Oracle Retail AI Foundation Cloud Services

Implementation Guide

Release 23.1.101.0

F76898-04

March 2023

ORACLE

Oracle Retail AI Foundation Cloud Services Implementation Guide, Release 23.1.101.0

F76898-04

Primary Author: Judith Meskill

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments xv

Preface xvii

1 Introduction

Oracle Retail AI Foundation Cloud Services and Business Agility 1-2
Oracle Retail AI Foundation Cloud Services Advanced Clustering 1-2
Reporting and Analysis 1-2
Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service 1-3
Dynamic Creation of Space Clusters 1-3
Conduct Micro-Space Optimization What-if Analysis 1-3
Preview Results Leveraging Shelf Preview Capabilities 1-3
Oracle Retail AI Foundation Cloud Services CDT Science and DT Science 1-3
Customer Decision Trees 1-3
Demand Transference Science 1-4
Oracle Retail AI Foundation Cloud Services Attribute Extraction 1-4
Oracle Retail AI Foundation Cloud Services Affinity Analysis 1-4
Common Workflow 1-4
Interacting with Oracle Retail AI Foundation Cloud Services 1-5
Hardware and Software Requirements 1-7
FAQs 1-7

2 Implementation Overview

Implementation Process 2-1
Implementation Steps 2-1
Configure the Application Roles and Users 2-1
Data Load Overview 2-1
Edit and Load Common Seed Data 2-2
Perform Attribute Preprocessing for CDT and DT, as Appropriate 2-3
Mandatory Configuration Parameters 2-3

3 Customer Decision Trees

Input Data 3-1
Overview 3-1
Transactions Data Requirements 3-1

Attribute Data Requirements 3-2
Attributes with a Large Number of Values 3-2
Grouped vs. Raw Attribute Values 3-4
Attribute Splitting 3-4
Functional-Fit Attributes 3-4
Customer Segments 3-5
Location Hierarchy 3-5
Setting Up Categories 3-5
Calculating Customer Decision Trees 3-5
Setting the Top Attribute 3-6
Excluding Attributes from the Calculation 3-6
Handling of the Brand Attribute 3-6
Limitations of the CDT Calculation 3-7
Choosing the Time Interval 3-7
Understanding the Filter Settings 3-7
Segments vs. Location 3-8
Setting the Escalation Path 3-8
How the CDT Score is Calculated 3-8
Understanding CDT Pruning 3-9
Overriding the CDT Calculation 3-9
Using the Calculation Stage 3-9
Setup Stage 3-9
Data Filtering Stage 3-9
Calculation Stage 3-10
Advanced Use 3-11

4 Demand Transference

DT and CDT 4-1
Demand Transference Model 4-1
An Example 4-2
Historical Similarity Data 4-2
Historical Sales Data 4-3
The Role of Attributes in Calculating Similarities 4-3
Attribute Data Requirements 4-4
Guidelines on Number of Attributes and Attribute Values 4-4
The Effect on Similarity Values 4-5
Avoiding Attributes with Many Values 4-5
Functional-Fit Attributes 4-6
Using Null as an Attribute Value 4-6
The Effect of Null Attribute Values on Similarity Values 4-7
Categories Containing Sub-Categories 4-7
Customer Segments 4-8
Location Hierarchy 4-9
Setting Up Categories 4-9

5 Using Demand Transference

Seasonality in Historical Sales Data 5-1

Assortment Elasticity 5-1
The Importance of Assortment Changes in Historical Data 5-2
Estimation of Assortment Elasticity 5-3
The Meaning of the Possible Values of Assortment Elasticity 5-4
The Substitutable Demand Percentage 5-4
No Requirement for a Time Interval 5-5
Segments vs. Locations 5-5
Setting the Escalation Path 5-6
Automatic Updating 5-6
Avoiding Categories with Small Assortments 5-7
Implementing DT for Fashion Categories 5-7
Proper Level for Fashion Categories 5-7
Seasonality (Life Cycle) Considerations 5-8

6 Advanced Clustering

Overview 6-1
Data Requirements 6-1
Multiple Hierarchies and Level Support 6-2
Clustering Criteria Supported in Store Clustering 6-3
Attributes in Store Clustering 6-4
Configuration Process 6-4
Copy Clusters Using Like Product Mapping 6-6
Multiple Clustering Approach 6-6
New Stores or Stores with Poor History 6-8
Outliers 6-9
Export to Excel 6-9

7 Customer Segmentation

Overview 7-1
Data Requirements 7-1
Multiple Hierarchies and Support 7-2
Supported Segment Criteria 7-2
Customer Segmentation Attributes 7-3
Configuration Process 7-4
Attribute Preprocessing 7-5
Segmenting Approach 7-6
Customer Segment Store Profile Generation 7-6
Preprocessing 7-6
Customer Metrics 7-7

8 Affinity Analysis

Overview 8-1
Data Requirements 8-1
MBA_ARM_SRVC_LOC_STG 8-2
MBA_ARM_SRVC_CONFIG_STG 8-2
Science Algorithms/Services 8-3

ARM_PH 8-3
ARM_PH_PROMO 8-3
ARM_PH_CS 8-3
Configurations 8-3
Data Output 8-4

9 Assortment and Space Optimization

Overview 9-1
ASO Data Input Requirements 9-3
Assortment Data 9-3
Planogram Data 9-4
Assortment-to-Planogram Mapping 9-5
Assortment to POG Mapping Process 9-5
Input Data 9-5
Automated Process 9-5
Mapping Errors 9-6
Product Images Data 9-6
Replenishment Data 9-7
Optimization Science 9-7
Optimization Algorithm Overview 9-7
Sales and Inventory Model 9-8
Sales and Inventory Modeling Considers All Possibilities 9-8
Inventory Levels After Replenishment 9-9
Calculating the Facing Capacity for a Product/Fixture Combination 9-9
Maximum Capacity of a Product 9-10
Replenishment Parameters 9-10
Sales Inventory Model Output 9-11
Objective Function 9-11
Constraints 9-12
Product Family Group Constraints 9-13
Diagnosis of Visual Guidelines 9-13
Product Groups 9-17
Validation Tool (Sanity Checker) 9-18
Diagnosis of Dropped Products 9-18
Checklist for Optimization Results Diagnosis 9-19
Monitoring Batch Processes 9-19
Overview 9-20
Batch Process Failure 9-20
Global Validation Issues 9-21
Sending Data in Data Files 9-23
Assortment-Related Files 9-23
POG-Related Files 9-24
Display-Style Files 9-24
POG Historical Data and Store CDAs 9-25
Mapping, Replenishment, and Other Files 9-25

10 Assortment Recommender

Prerequisites 10-1
Producing Better Assortments 10-1
Run Groups and Run Frequency 10-2
The Run Group Parameters 10-2
Data Used by the Assortment Recommender 10-3
Halo Effects 10-4
Troubleshooting 10-4

11 Size Profiles

Overview 11-1
Data Requirements 11-1
Hierarchy Data 11-1
Sales Data 11-2
Inventory Data 11-2
Product Images Data 11-3
Season 11-3
Size Range and Sub Size Range Definition 11-3
Size Definition and Mapping Between Size and Sub Size Range 11-4
Mapping Between SKU and Size 11-4
Mapping Between Style-Color/Store and Sub Size Range 11-5
Configurations 11-6
Data Output 11-8
Batch and Ad-Hoc Jobs 11-8

12 Offer Optimization

Overview 12-1
Project Planning 12-4
Functional 12-4
Data Interfaces 12-5
Forecasting Parameters 12-5
Optimization 12-5
Walkthrough 12-5
Security 12-9
Data Filtering 12-9
User Roles 12-10
Data Input Requirements 12-10
Hierarchy Data 12-10
Inventory Data 12-12
Price History Data 12-13
Retail Code 12-13
Retail Sales Data 12-14
Promotions 12-14
Warehouse Inventory Allocation 12-14
Competitor Prices 12-15
Product Attributes 12-15

Product Images Data 12-16
Season 12-16
Season Periods 12-16
Season Products 12-16
Price Ladder 12-16
Currency 12-17
Holidays 12-17
Pricing Product Groups 12-17
Budget 12-17
Strategy and Business Rules 12-17
Custom Columns 12-18
External Forecast Adjustment 12-18
Configuration and Expected Levels for Interfaces 12-18
Optional* Interfaces 12-20
Integration with Retail Pricing Cloud Service 12-20
POM Jobs 12-22
Forecasting Science 12-28
Parameter Estimation 12-28
Overview of the Parameter Estimation Stages 12-29
Data Preparation 12-29
Preprocessing 12-31
Elasticity 12-33
Seasonality 12-37
Promotion 12-39
Output and Review of Parameters 12-40
Day Level and Returns Metrics 12-41
Day of the Week Profiles 12-41
Returns Metrics 12-42
Demand Forecasting 12-43
New Stores 12-45
Optimization Science 12-46
Business Rules 12-46
Optimization Algorithm Overview 12-50
Model Apply 12-51
Objective Function 12-52
Constraints 12-52

13 Control and Tactical Center

Strategy and Policy Management 13-1
Forecast Configuration for MFP and AP 13-2
Run Type Configurations for MFP and AP to Set Up GA Runs 13-3
Batch and Ad-Hoc jobs for MFP and AP Forecasting 13-5
Forecast Configuration for RDF and AIF (including Inventory Optimization and Offer Optimization). 13-11
Batch and Ad-Hoc Jobs for RDF Forecasting 13-13
Workflow for RDF Implementation 13-18
Using the Add Multiple Run Types feature 13-19

Building Alternate Hierarchy in AIF 13-19
Custom Jobs Through Innovation Workbench (IW) 13-20

14 Inventory Optimization

Overview 14-1
Inventory Optimization Runs 14-2
Inventory Optimization UI Workflow 14-2
Security 14-3
Data Load Requirements 14-4
RAP Foundation Data (CSV and W_ interfaces) 14-4
RSP Foundation Data (RSE_% Tables) 14-5
Inventory Optimization Data (IO_% Tables) 14-5
Data Input Requirements 14-5
Hierarchy Data 14-5
Retail Sales Data 14-6
Inventory Data 14-6
Product Attributes 14-7
Product Images 14-7
Replenishment Attributes 14-7
Price and Cost 14-7
Season 14-7
Global Configurations 14-8
Inventory Optimization Integration with RMS 14-9
Forecast Engine Setup for IO 14-12
Batch and Ad Hoc Jobs for IO 14-13

15 Configuration

User Interface Authentication and Authorization 15-1
User Management Configuration: Configuring Users and Roles 15-1
User Roles 15-2
Configuration 15-2
RSE_CONFIG Table 15-2
Generic Configuration 15-3
Advanced Clustering Configurations 15-5
Assortment and Space Optimization Configurations 15-8
Customer Decision Tree Configurations 15-14
Demand Transference Configurations 15-16
Returns Logistics Configurations 15-19
Affinity Analysis Configurations 15-19
Resource Bundles 15-22
Manage Notebooks 15-22
Internationalization 15-22
Configuration Updates 15-22
Configuration Tables 15-23
Email Notification Configuration 15-25

16 Attribute Processing

Attribute Preprocessing 16-1
Process Overview 16-2
Product Attribute Loading 16-3
Introduce New Attribute 16-4
Determine the Attribute Source and Define in the Tables 16-4
W_PRODUCT_D or W_PRODUCT_ATTR_D 16-4
W_RTL_ITEM_GRP1_D or W_RTL_ITEM_GRP2_D 16-4
Populate RSE_PROD_ATTR_GRP_VALUE_STG Interface 16-4
Populate RSE_PROD_ATTR_VALUE_XREF_STG Interface 16-5

17 AI Foundation Web Services

AI Foundation Web Services 17-1
Authentication and Authorization 17-1
Summary of Web Services 17-2
Access to RSE_CONFIG Table 17-2
Access to RSE_CONFIG_CODE Table 17-3
Advanced Clustering Export 17-3
Customer Segment Export 17-4
ASO Exports 17-5
Retail Science Integration with XStore 17-8
Endpoint 17-8
Authentication 17-9
Retail Science and Retail Insight Integration 17-9
Configuration 17-9
POM Server Detail Configuration 17-9
POM Server Credentials 17-9

18 Batch Processing

Overview 18-1
Custom Batch Requests 18-1
Managing Custom Batch Requests 18-1
Handling Data Files 18-2
Supported PROCESS_QUEUE Trigger Values 18-2
Incremental Exports 18-3
Batch Process Flow 18-3
Configuring Additional Data Files 18-5
File Transmissions 18-6

19 Innovation Workbench

Components 19-1
Retailer Workspace Schema 19-2
Oracle APEX 19-2
Workspace 19-2
Users and Roles 19-2
Oracle Advanced Analytics 19-5

Oracle Data Mining 19-6
How to Invoke Oracle Data Mining 19-6
Notebooks 19-9
Scheduling Innovation Workbench Python Notebook 19-14
IDCS Setup 19-14
Scheduling Jobs 19-14
REST API Documentation 19-15
Example 19-15
Notebook and Paragraph IDs 19-19
Considerations 19-19
REST API 19-20
Restful Service 19-20
Integration 19-23
Custom Data Loads 19-23
Required Files 19-23
Zip File Contents 19-24
Context File Details 19-24
Data Load Feedback 19-25
Invoking the Data Load 19-25
Custom Data Exports 19-25
Required Configuration 19-25
Invoking the Export 19-26
Sample Extensibility Use Case 19-26
Troubleshooting 19-27
DBMS Scheduler 19-28
Schema Objects 19-28

20 Process Orchestration and Monitoring

Batch Administration 20-1
Batch Monitoring 20-1
Monitoring a Batch Cycle 20-2
Triggering Process from POAM 20-2
Running a Nightly Batch Process 20-2
Monitoring the Process Executions 20-3
Running AdHoc Batch Processes 20-3
Enabling AdHoc 20-3
Monitoring Process 20-4
Disabling AdHoc 20-4
Intraday Cycle 20-4
Error Handling 20-5

21 FAQs

General Questions 21-1
DT Questions 21-1
ASO Questions 21-3
Attribute Processing Questions 21-6

Offer Optimization Questions 21-6

A Appendix: Innovation Workbench Workshop

Overview A-1
Lab Innovation Workbench A-5
Lab Analysis A-7
Browse Existing Data A-8
Using ReST to Load Data A-11
Other Methods for Loading Data A-15
Explore Data Analysis A-16
Review Customer Behavior Analysis using Text Mining A-16
Innovate A-24
Churn Analysis using ODM A-25
Decision Tree Model Details A-26
Display Decision Tree Rules A-33
Lab Share Insights - ReST API A-39
Lab Share Insights - Task Navigation A-41
Database Tools A-43
How to Allocate Privileges A-44
How to Execute a Job using DBMS Scheduler A-44

Glossary of Acronyms

Send Us Your Comments

Oracle Retail AI Foundation Cloud Services Implementation Guide, Release 23.1.101.0

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

Oracle Retail AI Foundation Cloud Services Implementation Guide provides detailed information useful for implementing and configuring the application. It helps you to understand the behind-the-scenes processing of the application.

Audience

This document is for users and administrators. This includes merchandisers, buyers, business analysts, implementation team partners, the ORC, and administrative personnel.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle AI Foundation Cloud Services documentation set:

- *Oracle Retail AI Foundation Cloud Services Administration Guide*
- *Oracle Retail AI Foundation Cloud Services Implementation Guide*
- *Oracle Retail AI Foundation Cloud Services Release Readiness Guide*
- *Oracle Retail AI Foundation Cloud Services User Guide*
- *Oracle Retail AI Foundation Cloud Services Assortment and Space Optimization User Guide*
- *Oracle Retail AI Foundation Cloud Services Inventory Optimization User Guide*
- *Oracle Retail AI Foundation Cloud Services Promotion Markdown and Offer Optimization User Guide*
- *Oracle Retail Analytics and Planning Cloud Services Data Interfaces*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Improved Process for Oracle Retail AI Foundation Cloud Services Documentation Corrections

To more quickly address critical corrections to Oracle Retail AI Foundation Cloud Services documentation content, Oracle Retail AI Foundation Cloud Services documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle AI Foundation Cloud Services document may at times not be attached to a numbered software release; instead, the Oracle Retail AI Foundation Cloud Services document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

Oracle Retail AI Foundation Cloud Services documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail AI Foundation Cloud Services document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail AI Foundation Cloud Services Documentation on the Oracle Technology Network

Oracle Retail AI Foundation Cloud Services product documentation is available on the following web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. You can obtain these documents through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

The Oracle Retail AI Foundation Cloud Services combines AI, machine learning, and decision science with data captured from Oracle Retail AI Foundation Cloud Services SaaS applications and third-party data. The unique property of these learning-enabled applications is that they detect trends, learn from results, and increase their accuracy the more they are used, adding massive amounts of contextual data to obtain a clearer picture on what motivates outcomes.

The Oracle Retail AI Foundation Cloud Services are comprised of the following Cloud Services:

- Oracle Retail AI Foundation Platform Cloud Service
- Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service
- Oracle Retail AI Foundation Promotion and Markdown Optimization Cloud Service
- Oracle Retail AI Foundation Offer Optimization Cloud Service

The Oracle Retail AI Foundation Platform Cloud Service provides retailers with a data science toolkit that supports specific use-cases in planning, operations and execution and can be expanded to support broader retail uses. This includes Advanced Clustering, Customer Segmentation, Demand Transference, and Customer Decision Tree capabilities, as well as the recently introduced Attribute Extraction/Binning and Innovation Workbench capabilities.

The Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service is used to determine the optimal selection and arrangement of products within stores by optimizing the product assortment and product placement on a virtual planogram.

The Oracle Retail AI Foundation Promotion and Markdown Optimization Cloud Service and Oracle Retail AI Foundation Offer Optimization Cloud Service reflect the evolution of our price and promotion optimization capabilities into an integrated life-cycle price optimization offering that enables retailers to engage their customers in an omnichannel environment while maximizing profits. The modular approach to offering life cycle pricing for promotions and markdowns separate from targeted offers enables retailers to innovate at the speed of their customer, while also accounting for the maturity of loyalty data necessary for targeted offers. The combined capabilities provide the following benefits to retailers:

- Drive optimal promotion and pricing decisions for the entire product life cycle
- Engage customers with targeted and contextual offers
- Execute consistently, incorporating price and promotion plans, projected receipts, and returns.
- Simplify decision-making through high-automation, exception-driven processes and what-if optimizations
- Maximize accuracy and scale using artificial intelligence, machine learning, and optimization on the Oracle AI Foundation data science infrastructure

Oracle Retail AI Foundation Cloud Services and Business Agility

AI Retail Foundation Cloud Services are hosted in the Oracle Cloud with the security features inherent to Oracle technology and a robust data center classification, providing significant uptime. The Oracle Cloud team is responsible for installing, monitoring, patching, and upgrading retail software. Included in the service are continuous technical support, access to software feature enhancements, hardware upgrades, and disaster recovery. The Cloud Service model helps to free customer IT resources from the need to perform these tasks, giving retailers greater business agility to respond to changing technologies and to perform more value-added tasks focused on business processes and innovation.

Oracle Retail AI Foundation Software Cloud Service is acquired exclusively through a subscription service (SaaS) model. This shifts funding from a capital investment in software to an operational expense. Subscription-based pricing for retail applications offers flexibility and cost effectiveness.

Oracle Retail AI Foundation Cloud Services Advanced Clustering

Oracle Retail AI Foundation Cloud Services Advanced Clustering is an enterprise-specific clustering solution that leverages data mining capabilities to create store groupings at various product levels using multiple inputs. These inputs include performance data, product attributes, store attributes, third-party data such as demographic data as well as consumer segments. Using embedded science and automation capabilities, retailers are able to identify patterns within available data to create the necessary customer-centric and targeted clusters to be used by downstream assortment planning, allocation/replenishment, pricing, and promotions planning processes.

The store clustering process enables the creation, review, and approval of store clusters for downstream solution use, while also providing the ability to define and use clustering templates that can be specific to given product/location combinations.

Oracle Retail AI Foundation Cloud Services Advanced Clustering provides retailers with multiple clustering generation approaches and methods. These include the creation of simple, nested, and mixed attribute clusters using multiple methods, including those that support discrete and non-discrete attributes.

The types of clusters include the following:

- Performance-based clusters (Sales Revenue, Sales Units, Gross Profit%, and so on)
- Product attribute-based clusters (Brand, Color Family, Price Band, and so on)
- Location attribute-based clusters (Store Size, Climate, Population Size, and so on)
- Consumer profile-based clusters (Consumer Segment Profiles)

In addition to the above, users have the ability to create multiple clustering scenarios within a single cluster run. This enables the ability to leverage embedded rankings, scoring logic, as well as solution recommendations to define and approve the most appropriate clusters for use in intended planning or execution processes.

Reporting and Analysis

Users can access and review the following reporting information to drive decisions related to the clustering process.

Users can perform the following:

- Determine what categories or merchandise classifications benefit most from clustering; determine the level of product or location hierarchy at which to cluster; and determine what attributes should be leveraged.
- Analyze details related to the available cluster recommendations, assessing areas such as cluster composition, performance, and attributes, as well as store level scores (in relation to total clusters).
- Review cluster scenario comparison features, visually assessing differences between the respective store cluster details.

Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service

Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service can help maximize return on space, sales, revenue, and profits while improving customer satisfaction by optimizing assortments and facings to available space.

Leveraging key inputs such as optimization goals, demand transference science, and visual guidelines as well as inventory and replenishment factors, retailers are presented with a recommended shelf/fixture layout that can be leveraged in downstream execution processes.

Dynamic Creation of Space Clusters

Leveraging available fixture data, Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service dynamically groups stores (known as space clusters) with common fixture dimensions, enabling retailers to optimize and refine their assortments at the planogram or store level.

Conduct Micro-Space Optimization What-if Analysis

Oracle Retail AI Foundation Assortment and Space Optimization Cloud Service provides retailers with the ability to conduct 'what-if' analysis by adjusting fixture lengths during an optimization run. The solution allows for a visual review, comparison, and validation of the results. This provides the ability to dynamically manage and assess the impacts of adding or removing fixture space from a particular store (or store group). The solution can help plan for and conduct store projects by recommending the re-allocation of space to planograms with an optimal return on space.

Preview Results Leveraging Shelf Preview Capabilities

Prior to approving optimization results for downstream execution, retailers are able to review shelf previews, assessing variation from current or historical planograms as well as confirming that recommended results align with expectations. Updates to the respective shelf preview may be made in near real-time, with forecasted results updated in a real-time manner.

Oracle Retail AI Foundation Cloud Services CDT Science and DT Science

This section describes CDT Science and DT Science.

Customer Decision Trees

Oracle Retail AI Foundation Customer Decision Tree Science and Demand Transference Science enable retailers to create customer segment-specific decision trees using available

transaction level data. These customer decision trees are specific to their customer segments and the respective geographies they operate within, and retailers are provided a better understanding of their most important products and product attributes. Using this detailed information, the retailer is able to effectively analyze assortment coverage and identify the duplication of item types as well as prevent the removal of core items that would cause a loss of customers.

Demand Transference Science

Using Customer Decision Tree and Demand Transference Science, retailers can analyze a significant number of households (for example, in the thousands) to identify and rank which products are truly unique and whose sales are incremental, as opposed to those that can be discontinued because they are repetitive in nature and can be substituted with other products.

Understanding the incremental and substitutable sales associated with each item within an assortment, category managers can optimize the breadth of their assortments, as experienced by their customer's purchase preferences, with the optimal number of SKUs, given space constraints or financial goals.

Oracle Retail AI Foundation Cloud Services Attribute Extraction

Attribute Extraction (AE) is an enterprise-specific solution that uses machine learning to extract product attributes from free-form product description strings.

The application's embedded science and automation helps you to extract the attributes (such as brand, color, flavor, and so on) of each product in a particular category and to normalize the attribute values by correcting short forms, misspellings, and other inconsistencies. The product attributes can be used by Demand Transference, Customer Decision Trees, Advanced Clustering, and other retail applications that require product attributes in a structured format.

The AE application consists of the following tabs: Overview, Edit Labels, Annotation, Errors, Normalization, and Results. You use the Overview tab to select one of the previously added product categories or to add a new category. You use the Edit Labels to define category-specific attributes that you want to extract. In the Annotation and Errors tabs, you follow an iterative process to extract attributes and correct any mislabeled attributes. In the Normalization tab, you can use the embedded List of Values (LOV) or create your own LOV to standardize the attribute values. You use the Results tab to review and export the table of attributes.

Oracle Retail AI Foundation Cloud Services Affinity Analysis

Affinity Analysis (AA) lets retailers review the analysis about their customer market baskets. The system calculates association rules from the provided sales transaction data, which provides insight into customer shopping patterns. The process examines sales transaction data and identifies associations between different types of products. Such information can help a retailer understand that promoting one product is sufficient to help drive sales of another product, given the sales associations they exhibit.

Common Workflow

The AIF solutions have a similar workflow and user interface (UI). The workflow lets users implement new science applications using similar techniques. For example, a retailer who uses Demand Transference Science and Customer Decision Tree Science may then be able to more easily learn and use Advanced Clustering and other aspects of demand modeling. This approach lowers the future total cost of implementing various science applications.

The *Oracle Retail AI Foundation Cloud Services User Guide* provides details about using these applications.

Interacting with Oracle Retail AI Foundation Cloud Services

Two connection channels are used for interaction with the AIF:

Browser-Based

The application is accessed through a URL. The user is authenticated in order to gain access to the application. Access rights are controlled by the customer administrator through a Web application (Oracle Access Manager). Note that a role-based security policy is used. This allows the administrator to specify which applications and the tasks associated with those applications are accessible to which users.

Bulk Data Movement

A scheduled ETL extraction process must be used to extract the required data on the customer side and send it to the application through SFTP. Similarly, a schedule-based set of processes must be set up to process data coming in the opposite direction: from the application to on-premise. Note that this connection is still initiated from on-premise. Data is made available by the application for the download to on-premise location and processed further. All the necessary processes and credentials are set up during implementation.

Web Services

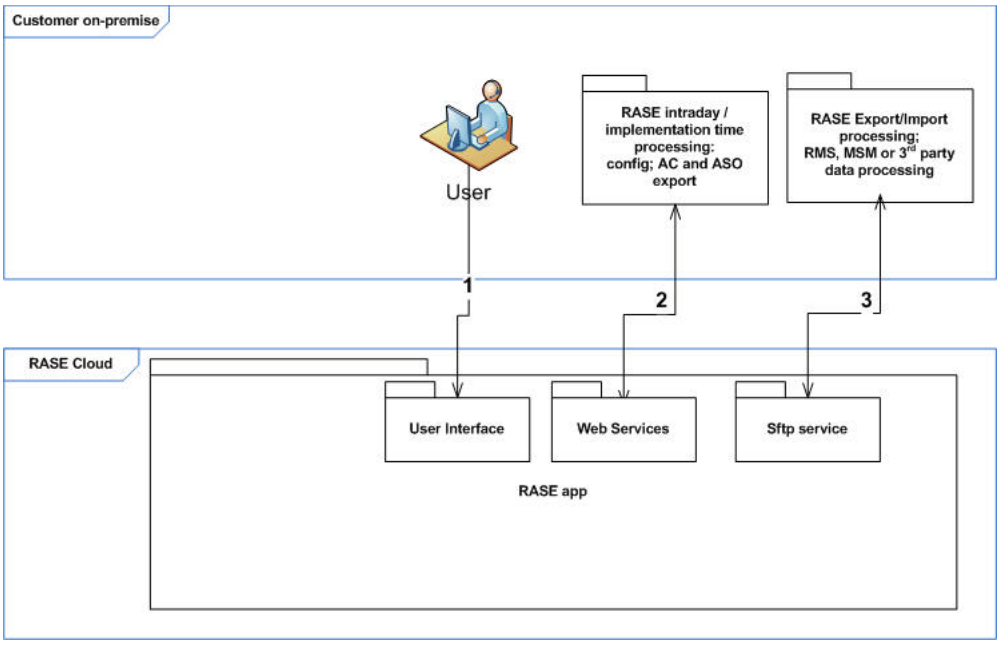
AI Foundation web services are REST-based. AI Foundation web services provide access to some of AI Foundation application data and functionality but do not fully mirror the user interface or the export and import features of the backend. They are not a replacement for bulk data export, which must still be done at a scheduled time as part of batch processing. However, access to the configuration can be used during implementation and upgrade time, while AC and ASO export web services can serve as the means of obtaining incremental update data from a specified point in time (driven by a query parameter) as means of intra-day processing.

In the Cloud

The application processes are hooked up to a cloud scheduler to work in concert with what is sent (uploaded) from on-premise and what must be published to the outgoing SFTP directory for on-premise download.

The interactions with the application are illustrated in Figure 1–1.

Figure 1–1 Interacting with Oracle Retail AI Foundation Cloud Services



Here are the major steps.

Table 1–1 Interacting with Oracle Retail AI Foundation Cloud Services

Step #	Protocol	Direction	External/ Internal	Description	Type of Data Sent
1	https/443	Inbound	External (Internet)	Used by customer to communicate with the application UI, OAM (login), OIM.	Cluster, DT, CDT, space optimized results parameters
2	sftp/2222	Inbound	External (Internet)	Data synchronization content (from RMS); Import/export files uploaded or downloaded by customer scripts (ODI for RMS-sourced data).	Includes hierarchies, calendar, sales transactions, and so on. Clusters generated, like entities, CDT xml, similarity files, space optimized results.
3	nfs/2049	Copy to/from local NFS mount	Internal	Copy files uploaded via SFTP to application server for processing with the data processing jobs. Copy exported result files to SFTP server for customer download.	Clusters generated, like entities, CDT xml, similarity files, and so on.
4	nfs/2049	Copy from local NFS mount	Internal	Pull files uploaded to SFTP server to application server for processing by Oracle Data Integrator (ODI) data load jobs.	Hierarchies, calendar, sales transactions, and so on.
5	SQLnet/1521	App->DB	Internal	DB create, update and delete operations from the application.	Cluster parameters, query parameters, and so on.

Hardware and Software Requirements

Oracle Retail AI Foundation Cloud Services has the following requirements:

Client System Requirements

The following operating system is supported:

- Microsoft Windows 10 with Microsoft Office 2013

> **Note:** AIF assumes that the retailer has ensured its Operating System has been patched with all applicable Windows updates.

The following browsers are supported:

- Mozilla Firefox
- Microsoft Edge
- Google Chrome (Desktop)

Microsoft has deprecated Internet Explorer 11 in Windows 10 and recommends using Edge as the default browser. Refer to the Oracle Software Web Browser Support Policy (https://www.oracle.com/middleware/technologies/browser-policy.html) for additional information.

Other Requirements

The user's source IP address must be communicated to the application cloud administration team for security purposes.

The SFTP client used for uploading and downloading data must be compatible with the SFTP protocol used by the application. Examples include:

- Putty Command line client
- Win SCP
- WS_FTP Pro Version 9

Note that all file exchange must be carried out in binary format.

FAQs

For answers to frequently asked questions, see Chapter 21, "FAQs."

2

Implementation Overview

This chapter provides an overview of the implementation of Oracle Retail AI Foundation Cloud Services.

Implementation Process

This section provides details about the implementation process. It assumes that the underlying platforms have been properly implemented. This includes servers, Oracle database, RADM, and WebLogic application servers.

Note: Data interfaces are required by the application to support the various supported modules. For details about the data interface, see the following:

*Oracle Retail Analytics and Planning Cloud Services Data Interface*s

---

Implementation Steps

The order of steps provided here is designed to simplify the process. The advanced user may be able to change the process order or skip some steps; however, that is not recommended and not documented here.

Note: See Chapter 15, "Configuration" for details about application configurations that can be modified as part of a deployment.

Configure the Application Roles and Users

Before any user can log into any application, you must set up application roles, add users, and assign users to the correct roles. To do this, complete the steps described in Chapter 15.

Data Load Overview

Note: Prior to running any installed .ksh scripts, you must source the RSE Environment Setup file located here: $RSE_HOME/common/scripts/lib/rse.env. To source this file, use the command

. $RSE_HOME/common/scripts/lib/rse.env

During an implementation of any applications, several steps are required. This section provides some details about this process.

The rse_config.ksh and the rse_master.ksh script are located in the $RSE_HOME/common/scripts/bin directory. In addition, similar scripts are located within each of the application directories, for example, $RSE_HOME/cdm/cis/scripts/bin has a cis_config.ksh and a cis_master.ksh script. All of the *config.ksh and *master.ksh are similar in nature, so this section focuses on the rse_config.ksh and rse_master.ksh as examples. However, the concepts apply equally to the application-specific *config.ksh and *master.ksh scripts.

Edit and Load Common Seed Data

All the applications share a set of configurable parameters that must be loaded into the RSE_CONFIG table. All have default values and are configurable and editable by the administrator. This section explains how to load and, if desired, edit these parameters.

The .ctl files for common configuration data must be edited and loaded into the staging tables. This data is common to all the applications. The application-specific .ctl files are located in their own application seed_data folders (for example, orase\installer\orase16\so\db\seed_data).

Review the .ctl files in that directory and adjust any configurations needed for the environment. Some configurations cannot be changed after the application has been used; therefore, you must carefully consider the parameters listed in Table 2–1. The remainder are optional and default to reasonable valuables.

The following configuration parameters must be initialized during setup. The values for hierarchy level and type are recommended for any installation that integrates with the CMPO installation and must match for all installed applications.

Table 2–1 Mandatory Common RSE Database Configuration Parameters

Application	Parameter	Description	Value
RSE	CAL_PERIOD_LEVEL	This is the calendar hierarchy level that is used to drive RSE processing.	4
RSE	CHAIN_LEVEL_DESC	The description to use for any top level hierarchy element, when one must be manually created.	CHAIN
RSE	CMGRP_HIER_TYPE	The hierarchy ID to use for the CMPO. Recommendation is for a normal installation with CMPO.	1
RSE	CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy where the CMPO level exists (installation configuration). Recommendation is for a normal installation with CMPO.	5
RSE	MT_TZ	The time zone that is used by application server(s), that is, by the middle-tier. It must match SELECT tzname FROM V$TIMEZONE_NAMES.	America/New_York
RSE	PRIMARY_LANGUAGE_CODE	The name of the language code to use for all RSE data sourced from RI.	EN
RSE	RA_FISCAL_CAL_ID	The ID of the calendar to use from RI (since RI supports multiple calendars).	1240
RSE	TRADE_AREA_HIER_TYPE	The hierarchy ID to use for the Trade Area (installation configuration).	6
RSE	UI_TZ	The time zone for display. It must match SELECT tzname FROM V$TIMEZONE_NAMES.	America/New_York

Perform Attribute Preprocessing for CDT and DT, as Appropriate

Product attributes are required by CDT and DT and are stored in the RADM. Attribute preprocessing is independent of the AIF database and happens in RI or flat files generated by the user. Once these tables and files are correct, you can load the resulting attributes in the AIF schema as part of the data load process.

Here are the basic attribute pre-processing steps:

1. Populate RADM with raw attribute data.
2. Optionally, perform translation.
3. Parse.
4. Cleanse and standardize.
5. Categorize and label.
6. Define the attributes.
7. Bin and group.

For details on these steps, see Chapter 16, "Attribute Processing".

Mandatory Configuration Parameters

Table 2–2 contains the mandatory configuration parameters for CDT.

Table 2–2 Mandatory CDT Configuration Parameters

Application	Parameter	Description	Value
CDT	CDT_CAL_HIER_TYPE	The hierarchy ID to use for the fiscal calendar (installation configuration).	11
CDT	CDT_CAL_LEVEL_ID	The hierarchy level ID that contains the level of the calendar hierarchy that CDT operates on. (This should equate to the Week - Installation configuration).	4
CDT	CDT_CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy that CDTs are created for (installation configuration).	5
CDT	CDT_CUSTSEG_HIER_TYPE	The hierarchy ID to use for customer segment (installation configuration).	4
CDT	CDT_CUSTSEG_LEVEL_ID	The hierarchy level ID that contains the level of the customer segment hierarchy that CDTs are created for (installation configuration).	2
CDT	CDT_LOC_HIER_TYPE	The hierarchy ID to use for location (installation configuration).	2
CDT	CDT_LOC_LEVEL_ID	The hierarchy level ID that contains the level of the location hierarchy that CDTs are created for (installation configuration).	Best equivalent for trade area level
CDT	CDT_PROD_HIER_TYPE	The hierarchy ID to use for the CMPO (installation configuration). The recommendation is for a normal installation with CMPO.	1
CDT	DEF_NUM_WEEKS_FOR_ SIMILARITY	The default number of weeks of sales transaction data to be used by the similarity process. This is used when the user does not specify time intervals.	15

Table 2–3 contains the mandatory configuration parameters for DT.

Table 2–3 Mandatory DT Configuration Parameters

Application	Parameter	Description	Value
DT	AE_CALC_INT_LENGTH	The number of weeks to group together for in an interval for the AE calculation.	4
DT	CDT_SIMILARITY_AVAILABLE	Whether the CDT similarity has been made available to DT.	Y
DT	DT_CAL_HIER_TYPE	The hierarchy ID to use for the fiscal calendar.	11
DT	DT_CAL_LEVEL_ID	The hierarchy level ID that contains the level of the calendar hierarchy that DT operates on. (It should equate to week.)	4
DT	DT_CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy that DTs are created for.	5
DT	DT_LOC_HIER_TYPE	The hierarchy ID to use for location.	5
DT	DT_LOC_LEVEL_ID	The hierarchy level ID that contains the level of the location hierarchy that DTs are created for.	Best equivalent for trade area level
DT	DT_PROD_HIER_TYPE	The hierarchy ID to use for the CMPO.	1
DT	PR_LOC_STATUS_LAST_ COMPLETED_WK	The last completed week for the SKU/Store ranging data copying.	Week ID from RSE_ CAL_HIER
DT	WGT_CALC_INTERVAL_ LENGTH	The number of weeks to group into an interval that is then used to perform weight calculations.	4

Table 2–4 contains the mandatory configuration parameters for AC.

Table 2–4 Mandatory AC Configuration Parameters

Application	Parameter	Description	Value
CIS	CIS_DFLT_CALENDAR_HIER_ TYPE_ID	the default calendar hierarchy for clustering.	11
CIS	CIS_DFLT_LOCATION_HIER_ TYPE_ID	The default location hierarchy for clustering.	2
CIS	CIS_DFLT_PRODUCT_HIER_ TYPE_ID	The default product hierarchy for clustering.	1
CIS	PERF_CIS_APPROACH	The approach to use for performance based clustering. The available options are CDT and DT.	CDT

Note: There are no mandatory configuration parameters for MBA.

Table 2–5 Mandatory ASO Configuration Parameters

Application	Parameter	Description	Value
SO	SO_CAL_HIER_TYPE	The hierarchy ID to use for the calendar (installation configuration).	10
SO	SO_FISCAL_CAL_HIER_TYPE	The hierarchy ID to use for the fiscal calendar (installation configuration).	11
SO	SO_LOC_HIER_TYPE	The hierarchy ID to use for location (installation configuration).	2
SO	SO_PROD_HIER_LEVEL_FOR_LEAF_NODE	The product hierarchy level number for leaf node.	7
SO	SO_PROD_HIER_TYPE	The hierarchy ID to use for the product (installation configuration).	1

3

Customer Decision Trees

This chapter provides details about the use of the Customer Decision Tree Science application.

Input Data

This section describes setting up the data that the CDT application uses to calculate CDTs.

Overview

The calculation of CDTs is based on a retailer's historical data, especially customer-linked transactions data that includes subsets of transactions from the same customer. The CDT calculation does not require any data about the customer; it does require that the transactions are flagged to indicate that they came from the same customer.

The CDT calculation uses this customer-linked transactions data to determine, for a particular category at a particular store, the switching behavior of the customer among the items in the category at that store. By seeing what fraction of all historical customers of the category consider two specific items substitutable, CDT generates a similarity for the two items, which is a number between 0 and 1 that indicates how substitutable those two items are.

It is important to have data from a large numbers of customers shopping in the category in order to be more certain of the similarity values. In general, it is not recommended to perform CDT calculation for categories where customer-linked transactions data is available only for a few hundred customers.

The CDT calculation also relies on attributes, since attributes are at the nodes of the CDT. The CDT calculation applies the similarity calculation to attribute values as well as to items in order to find the similarities between attribute values. The CDT is then generated by examining the relationships between the attribute-value similarities and the item-level similarities. So good attribute information is also important.

Notice that the CDT calculation is all within a particular category, and thus the CDT models the customer's choice process only within a category. The CDT calculation generates separate CDTs, using separate calculations, for each category that the user chooses.

The CDT calculation does not filter out the effects of promotions or price changes, because these effects can cause customers to switch to a different item. This is valuable since the basis of the CDT calculation is examining switching behavior among the customers. Generally, more switching behavior in the historical data leads to a better CDT.

Transactions Data Requirements

The historical transactions data for the CDT calculation must meet the following requirements:

Customer Linked

Since the calculation involves examining switching behavior by customers, it is necessary to identify which transactions came from the same customer. This can be done using a loyalty card or a generated customer ID. No actual information about a specific customer is required; all that is needed is a way to identify which transactions come from the same customer. Note that it is possible to have customer IDs from a retailer where the customer ID is not that of an actual customer but rather a cashier loyalty card that was used for many different customers. These customer IDs, and their associated history, cannot be used for the CDT calculation, since the data comes from a large number of different customers. The data load for CDT automatically filters out such "fake customers."

Repeat Purchase

The category used for the calculation must be one where the typical customer performs several transactions per year. Examples include grocery items such as milk or yogurt, which are typically purchased weekly, and batteries, which are typically purchased several times per year. Item such as electronics are not appropriate, as such items may only be purchased every few years. Note that it can be possible to trade off purchase frequency and history length. It is also possible to trade off purchase frequency with the number of customers who shop in the category.

Attribute Data Requirements

The attribute values for the CDT calculation must meet the following requirements:

Set of Attributes

Each category is characterized by a unique set of attributes. These attributes differ from category to category. For example, for yogurt, the attributes might be size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes might be size, brand, milk/dark, nut type, and package type. Two categories can both have brand, but that the brand attribute will have different values for each of the categories. So brand is actually a different attribute for each category.

Mapping

Each item in the category must be mapped to its set of attribute values. This information must be obtained from the retailer. Null values are acceptable as long as they are not too numerous. The CDT application can still run even if some attribute values are listed as null for some items in a category, but too many null values decrease the reliability of the generated CDTs.

Significance

The attributes for a category must be the ones that the customers actually pay attention to when shopping in the category. They are attributes that actually affect the customers' purchasing decisions.

The process of obtaining attributes for a category and performing a mapping of items in the category to attribute values is likely to require a significant amount of time and labor, even if the retailer has the information available, since this has to be done for every category.

Attributes with a Large Number of Values

A raw attribute is one that has a large number of attribute values. For example, the brand attribute for yogurt may be a list of 50 different brands at a large grocer. Using the raw attributes directly for the CDT calculation presents a problem, because each node of a CDT expands into a set of branches whose number is equal to the number of attribute values of the attribute at the node. An expansion into 50 different branches, one for each brand, is not desirable because the CDT would become too large to examine or interpret.

Such raw attributes must be turned into grouped attributes. This involves grouping the attribute values into a small number of bins. This grouping should be done in consultation with the retailer, who may have specific requirements. For example, a retailer may want to group soft-drink brands into Brand A, Brand B, and a third group that includes all other brands.

Another approach is to divide the values into two attributes (known as attribute splitting). For example, if the color attribute has many values, the single color attribute can be divided into two attributes, with one attribute representing the primary color and the second attribute representing a modifier. The CDT application's schema directly supports attribute grouping.

Attributes with a large number of values (for example, in the hundreds) can cause the CDT Calculation Stage to require a lot of time. Here are some approaches for handling categories that have attributes with a large number of values. The retailer should help determine which approach is appropriate.

Position of the Attribute Within the Tree

Typically, an attribute with a large number of values must not be at the top or near the top of the CDT. With such a large number of values, it is unlikely customers are first selecting from among such a large number of values and then selecting from other attributes with a smaller number of values. In addition, if such an attribute were at the top of the tree, the tree would be extremely wide and shallow. It would be extremely wide because the tree would then split into as many branches as there are attribute values. If there are 100 brands, and Brand was at the top of the tree, the tree would split into 100 different branches. As a result, the CDT would not be useful to the retailer. The tree would also be quite shallow; with 100 different branches, each branch would probably have very few SKUs, and so the branch could not be expanded much further.

In such a category, customers generally first use another attribute with a smaller number of values, and then choose an attribute with a large number of values. The following example demonstrates what happens when an attribute with a large number of values is lower in the tree.

In this example, the top attribute in the tree indicates the Market Segment, so that the SKUs in the category are split into various sub-categories. The Brand has a large number of attribute values. Because the Brand is below Market Segment, the branch for each segment only has a small subset of the Brands. Although Brand has many attribute values along each branch, only a subset apply, because each Brand only applies to one or two market segments. As a result, CDT never branches by all of the values in Brand, and only branch by a small subset of Brand.

It is possible for the CDT calculation to move Brand lower in the tree by itself, but in order to improve the performance of the CDT calculation in such a case, it can be helpful to direct the calculation to move the attribute with a large number of values lower in the tree.

There is no direct way in the CDT application to force an attribute to be lower, but here are some indirect strategies to use:

- In the Calculation Stage, set the Market Segment attribute to be a Top Attribute. This forces the Market Segment to be at the top of the tree, and so Brand will not be at the top of the tree. This can improve the performance of the Calculation Stage of the CDT application because it has fewer options to consider when expanding the tree.
- Set multiple attributes as the Top Attribute. It is possible that multiple attributes in combination delineate the market segment, for example. There could be a main segment and a sub-segment, as two separate attributes. In such a case, set both of them as Top Attribute. The CDT Calculation Stage will still determine the ordering of the Top Attributes.

For more information, see Functional-Fit Attributes.

Grouped vs. Raw Attribute Values

CDT supports grouped attributes, which turns the raw attribute into one with many fewer values by grouping the attribute values into a small number of bins (for example, 3 to 5). CDT does not have an automated way of performing this grouping, so it is best if the grouping is done in consultation with the retailer, who may have specific requirements.

For example, a retailer may want the following grouping of soft-drink brands: {Soda A, Soda B, everything else}, because they are most interested in their two main brands and are willing to bin all other smaller brands together. The grouped-attribute approach is primarily useful for retailers who are not concerned with every specific value of an attribute that may have a large number of values.

A CDT with a grouped attribute will have branches only for the groups, and not for the original raw values within the groups. In the example above, the branches for Brand would consist only of Soda A, Soda B, Everything Else. This approach can provide additional insights into shopping behavior that would not be available with the raw-attribute approach. For example, if the Brands were grouped into three groups, each representing a particular price tier (say High, MainStream, and Budget), then the CDT can show the behavior of Mainstream customers vs. the behavior of Budget customers. The portion of the CDT underneath Mainstream would look different from the portion of the CDT underneath Budget. In essence, the portions are CDTs that show how each type of customer shops. This insight would not be available using only the raw Brand attribute.

Attribute Splitting

Another approach to handling attributes that have a large number of attribute values is to break them into two attributes (known as attribute splitting). For example, if the Color attribute has many values representing combinations of colors, it may be best to break the single color attribute into two attributes, with one attribute representing the primary color and the second attribute representing a modifier. However, this is an advanced technique, and grouping is recommended over attribute splitting.

Functional-Fit Attributes

A functional fit attribute is one where there is no substitution across the attribute's values. For example, batteries of different sizes cannot be substituted for one another. Any category where size determines the functional suitability of the item will have size as a functional-fit attribute. Information about which attributes are functional fit ones must be loaded into the CDT application.

Designating an attribute as functional fit can also be useful any time the attribute is unlikely to have substitution across it (for example, caffeinated vs. decaffeinated coffee). This is not exactly functional fit; however, substitution is unlikely, so it is better to mark the attribute as functional fit.

Functional-fit attributes always appear at the top of the CDT. The order of the functional-fit attributes will be some arbitrary order, but the order is not meaningful since there is no sense in which one functional fit attribute is more important than another. What the functional-fit attributes do is partition the set of items in a category into their own separate groups, each of which then has its own CDT.

This same effect can be achieved via the UI in the Calculation Stage, by using the Top Attribute functionality of the pop-up called Category Attributes Setup. Using the Top Attribute check box in this pop-up indicates to the Calculation Stage that the attribute should be put at the top of the tree.

Customer Segments

The CDT application can calculate CDTs by customer segment. Customer segments are set of groupings of the customer IDs that are used to identify the transactions. The retailer must provide the customer segment information as a data feed.

Customer segments are useful for retailers who believe that different customer segments shop differently. They may want CDTs by segment only for particular categories. The Calculation stage provides check boxes that control whether or not the run generates CDTs by segment. You can use these check boxes to calculate CDTs by segment for particular categories only.

Frequently, the groups will overlap, since a customer can belong to more than one segment. The CDT application functions normally in this case, since a separate CDT calculation is done per segment.

Location Hierarchy

The CDT application supports calculating CDTs by location hierarchy. The lowest level of the hierarchy should be above store; in general, calculating CDTs per store is not recommended. Per-store CDTs may have too little data to be reliable, and the calculation time involved can be quite long.

Some retailers may have stores that vastly differ in size and assortments. For example, a grocery chain may have both convenience stores and supermarkets. It may be necessary to arrange a separate calculation of CDTs for convenience stores vs. supermarkets, because people may shop differently at the two types of stores and the assortments may be different at the two types of stores.

One approach to this is to arrange a separate calculation by creating separate store clusters for convenience stores vs. supermarkets. The CDT application has the capability of calculating CDTs for each element of the location hierarchy, so it can calculate CDTs for the separate store clusters and thus produce separate CDTs for convenience stores vs. supermarkets.

Setting Up Categories

In general, a category is a set of items that are substitutable with each other (if there are no functional-fit attributes). The categories at a retailer can all be derived by choosing the correct level of the merchandise hierarchy at the retailer. The CDT application configuration supports choosing which level of the merchandise hierarchy is to be used as the category level.

A retailer may want categories that consist of unions of nodes of its merchandise hierarchy, because no level of its merchandise hierarchy suffices as the category level. The CDT application does support this, in that it allows defining an alternate merchandise hierarchy, where the categories can consist of arbitrary collections of items. However, before investing time in setting up an alternate hierarchy, make sure that it is necessary for meaningful CDT calculations.

Consider a category that consists of related though not substitutable products. For example, a single category of hair products can include both shampoo, conditioner, and items that are both shampoo and conditioner. There may be other hair-care related products in the category as well. A reasonable approach is to create an attribute called "Item Type" or "Market Segment" to indicate why the customer is buying it. The Market Segment attribute will segment the category into several sub-categories, and in the Market Segment, attribute should be set as a Top Attribute (see Setting the Top Attribute).

Calculating Customer Decision Trees

This section suggests ways to using the stages of the CDT application effectively.

Setting the Top Attribute

The Category Attribute Setup pop-up in the Calculation Stage contains check boxes that force particular attributes to the top of the tree. This is useful in several cases:

- The category has an attribute that has a large number of values (50 or more). See Position of the Attribute Within the Tree.
- The category has a functional fit attribute. See Functional-Fit Attributes.
- The category has an attribute that distinguishes market segments or product use. See Setting Up Categories.
- You want to use the same top attribute that is present in a CDT from another source in order to make comparisons with that CDT easier.

It is possible to set more than one top attribute by checking multiple check boxes. In this case, all of the attributes will be at the top, but the Calculation Stage will determine the ordering of the attributes along each branch. This is useful if, for example, there are several attributes that together determine market segment.

In the case of using the top attribute as a market-segmenting attribute, it is possible to experiment with not using this attribute as the top attribute and letting the Calculation Stage determine the attribute ordering. This is useful if the market-segmenting attribute is not truly a functional fit attribute; that is, consumers can substitute across some of the market segments. For example, in the Cookie category, most likely customers can substitute across most of the market segments, because almost all cookies are desserts. In such a case, the Calculation Stage can give additional insight, by showing, for example, that customers actually shop by brand, so that even when they substitute across market segments they tend to stay with the same brand. This can be valuable information. However, if the retailer is interested only in substitutions within market segments, then it is proper to designate the market segment attribute as the Top Attribute.

However, in the case where the category has a very large number of items (greater than 1000), or the category has an attribute with a large number of values (50 or more), it is unwise to try such experiments, because the Calculation Stage may run too long. For such categories, setting the market-segmenting attribute as the top attribute is the best approach.

Excluding Attributes from the Calculation

The Category Attribute Setup pop-up in the Calculation Stage allows excluding particular attributes from the calculation. Use this to avoid meaningless attributes in the tree and also to decrease the calculation time of the Calculation Stage. Include only attributes in the tree that actually affect customers' purchasing. For example, Country of Origin may or may not be a relevant attribute, depending on whether it is visible on the package and plays a role in customers' decisions. Excluding such attributes will not only create a more useful CDT, but will also cut down on the execution time of the Calculation Stage

Handling of the Brand Attribute

Almost all categories will have a Brand attribute. The power of brands is well-known in retail, and in most categories, customers tend to stick with the same brand. Because of this, the Brand attribute will tend to show up near the top of the CDT. This is the correct scientific result, but not necessarily a useful one, for two reasons:

- It is known that customers shop by brand.
- Brand may have many attribute values, and the resulting tree will be shallow if Brand is high in the tree (see Position of the Attribute Within the Tree).

Here are some strategies for getting around these effects:

- Exclude Brand from the tree (see Excluding Attributes from the Calculation). The resulting tree will describe customer behavior in the other attributes. This indicates customer behavior, assuming that they shop by Brand. Given that they shop by brand, what are the effects of the other attributes on their purchasing behavior? CDT answers that question.
- Use the Top Attribute functionality to move Brand lower in the tree. See Setting the Top Attribute.
- Group the brands, so that Brand becomes a grouped attribute. See Grouped vs. Raw Attribute Values. This is a reasonable approach if taken in conjunction with the retailer, and can offer additional insight into shopping behavior not available without grouping. However, this approach is best taken as a phase 2 task, rather than immediately.

Limitations of the CDT Calculation

Because the CDT calculation uses historical data, the resulting CDT depends on the historical assortment represented in the data. If a particular attribute value does not have any representation in historical assortments among a particular group of stores, then the CDT for those stores will not have this attribute value in it. Similarly, if the assortments carried many more items of a particular attribute value compared to another attribute value, which limits the customer's choices, this can affect the CDT.

It is important to select historical data that reflects the retailer's current assortment, if the retailer has changed assortments in the last few years.

Choosing the Time Interval

The data used to calculate CDTs can be restricted to specific time intervals in the Data Setup stage. Thus, it is not necessary to use all of the available historical data to calculate CDTs. Some possible reasons for restricting the data to specific time intervals are:

The retailer may decide that particular time intervals, such as the two months before Christmas, represent periods where the buying behavior of its customers is significantly different for certain categories. In this case, you can run the CDT application for just for these categories. Choose these categories in the UI, and then also choose the particular time intervals where the retailer believes shopping behavior is different.

If the retailer has changed business practices for certain categories, it is better to exclude the historical data from before the change, so that the CDTs reflect the retailer's current business practices and assortments, not the past ones.

One caution about selecting time intervals: there is always the danger of selecting too narrow a time interval, so that the amount of historical data in the interval is too little. See Transactions Data Requirements.

In general, it is better not to restrict the data too much.

Understanding the Filter Settings

The Data Filtering stage of the CDT UI can be used to filter the data in order to remove outlier data that may result in incorrect CDTs. The user can adjust the values for the filters in order to control the extent of the filtering. The Data Filtering stage has five filters.

The three absolute filters have values that represent absolute limits that the data in question must pass in order not to be filtered out. For example, the maximum on missing attribute values states an absolute maximum that items must meet in order to be used in the CDT calculation. Items that have more than the maximum allowable missing attribute values will not be used in the CDT calculation.

The two relative filters have field values that are relative to the median of each category. The filters use median instead of the more-common average because the median is less likely to be affected by extreme outliers in the data. The average can be brought up (or down) by a single extreme outlier; this cannot happen with the median.

For example, the Transaction History Minimum is a percentage of the median transaction history length for a particular category. It is possible that the transaction history length varies by category. In generating a CDT for a particular category, the goal is to filter out customers who have transaction histories that are too short.

The default value of the filter for Transaction History Minimum is set to 1%, which filters out the customers that are truly outliers for the category because their history length is much smaller than median.

Segments vs. Location

Calculating CDTs by both segment and location hierarchy is not recommended. This calculation generates a large number of CDTs, since it will generate one CDT for each combination of location and segment, which takes a large amount of computation time. The large number of CDTs generated are not considered practically useful. You should either generate CDTs by segment, using Location Chain, or generate CDTs by location, using Segment Chain.

For a first calculation of CDTs, it is best to calculate them at Segment Chain/Location Chain. This quickly generates one CDT per category. It is a good way to check that everything has been done correctly and that the CDTs being produced are not unreasonable.

Setting the Escalation Path

The last stage in the CDT application involves setting the escalation path. If you are using only the segment hierarchy or only the location hierarchy, the escalation path is simply the hierarchy that you are using, and you set the escalation path according to the hierarchy. If you are using both a location hierarchy and a segment hierarchy, then usually you should set the escalation path to go up the segment hierarchy first, and then the location hierarchy. It is better to use only one of the hierarchies.

When using both hierarchies, the escalation path is necessary in order to tell the application which parent it should go to when moving up from a given segment/location node. With both hierarchies in play, every segment/location node has multiple higher-level nodes that do not lie along a single path. The escalation path is necessary to tell the application in what order the higher-level nodes should be considered. When only one hierarchy is used, the higher-level nodes form a single path.

How the CDT Score is Calculated

The terminal nodes of a CDT are the lowest-level nodes in the tree. The terminal nodes of the tree partition the items in the category. The items within each terminal node should be quite similar to each other, and less similar to the items in the other terminal nodes. The terminal nodes provide a clustering of the items in the category. A numerical score for the clustering given by the terminal nodes can be calculated.

Unconstrained clustering using any of the standard clustering algorithms using the similarities as the distance measure can also be created. This clustering can be compared with the clustering score for the clustering by terminal nodes. The terminal-node cluster score will be lower than the score for the unconstrained clustering because the unconstrained clustering had no constraints when performing the clustering. The closer the terminal-node clustering score is to the unconstrained score, the better the CDT. The CDT score in the CDT application is represented as a percentage of the unconstrained clustering score.

Typically, you should eliminate any CDT that has a score of below 60 percent, using the Pruning stage of the application.

Understanding CDT Pruning

The Evaluation Stage of the CDT application performs an operation called pruning, in which entire CDTs are removed. In the Evaluation Stage, the CDT as a whole is deemed reliable or not. An unreliable CDT is removed in its entirety; there is no automatic mechanism for making small adjustments to a CDT. The only mechanism the CDT application has for making small adjustments to a CDT is the manual editing of a CDT allowed in the CDT editor.

Overriding the CDT Calculation

It may be necessary, because of prior knowledge concerning the business of the retailer, or knowledge about the historical transactions at the retailer, to override portions of the calculation performed in the Calculation Stage of the CDT application. The override mechanism there allows you to specify what the topmost attributes of the CDT should be. For example, from an understanding of the retailer's business, it may be clear that in a particular category, brand should be at the top level of the tree. The override mechanism allows you to specify brand as the top level of the tree. The override mechanism is also flexible enough to allow specifying only the top level of the tree, while the rest of the tree is filled in by the usual calculation.

While it is possible to obtain the same effect by manual editing of the CDT, manual editing is much slower, especially if you have generated multiple CDTs for each category.

Using the Calculation Stage

This section provides step-by-step instructions for setting up the Calculation Stage, with a few comments on using the other stages. The focus here is mainly on the Calculation Stage, because the settings in this stage can directly affect how the CDTs look and because the Calculation Stage generally takes up most of the execution time.

If you are just beginning to use the CDT application, experiment with smaller categories (fewer than 1,000 items) initially. Smaller categories are easier to work with because they take less execution time in the Calculation Stage than larger ones, so it is easier to do multiple runs and examine results.

Setup Stage

When first starting to use the CDT application, it is best to set up only one category at a time in the Setup Stage. In this way, each run is for one single category. It requires some experience to include multiple categories in the same run, and it is not recommended as a starting point. The instructions assume only one category has been set up in this run.

Before selecting a category for a CDT run, review the data requirements in Transactions Data Requirements to be sure that your desired category meets the data requirements.

Data Filtering Stage

This stage is usually straightforward, in that the default values of the fields are usually suitable. However, it is important to check the Data Filtering Summary at the bottom of the screen *after* the stage has completed running. You must click **Refresh** in the summary table in order to see the results related to the latest run. Check each filter in the summary to see how much data it filtered out. If too much data was filter out, then determine whether the data may have a problem, or whether you need to adjust the filter so that it is less stringent.

If this is the first time you have run the Data Filtering Stage on a particular category, then you should run only the Setup Stage and the Data Filtering stage on the category, without running the Calculation Stage. This allows you to check the Data Filtering Summary before spending time running the Calculation Stage. Once you have run the Data Filtering Stage on a category and have checked the Data Filtering Summary, then you can re-run the Data Filtering Stage on the same category without checking the Summary, unless you have loaded new data for the category.

Calculation Stage

The steps here are simplified to help you get started in properly using the Calculation Stage. Once you become familiar with using this process, you can alter and expand them to use more of the capabilities of the CDT application. The process presented here represents the minimal set of steps needed to produce CDTs and to get you going in the right direction.

Take care during this process so that the Calculation Stage can complete within 1 or 2 hours for categories that have more than 1,000 items or that have an attribute with more than 50 values. The steps detail any additional consideration needed for large categories. After performing a run with these steps, if the time for the Calculation Stage to run turns out to be acceptable, then these restrictions can be relaxed on subsequent runs of the category.

Each step may reference sections of this chapter that can provide further details.

1. Check both top level check boxes (one for Segments and one for Location). With these settings, the Calculation Stage will generate only one CDT, representing the CDT for all customers and all locations. This is the recommended way to start using the Calculation Stage. In particular, these settings are recommended for very large categories, where the calculation time for multiple CDTs may be quite prohibitive and not worth the investment until you have generated one CDT.
2. Exclude any unnecessary attributes (see Excluding Attributes from the Calculation). It is good practice to exclude unnecessary attributes, but it is even more so when working with large categories in order to avoid unnecessary computation. For large categories, consider excluding attributes that you know are less important to the retailer, even if they may have an effect on customers' purchasing in the category.
3. Handle Brand properly. Brand frequently has many attribute values, and handling it properly is especially important when the category also has a large number of items (1,000 or more). You can skip this step if the category has fewer than 1,000 items. See Handling of the Brand Attribute.
4. Set Top Attributes properly. In particular, if the category has some type of market segment, product type, or product usage attributes, then force these attributes to be at the top of the tree. If the category has a large number of items, then it is likely to require some of these attributes, because with that many items, the items will likely have different segments or types. It is unlikely that the entire set of items is completely interchangeable in the customers' mind, and so it is proper to put segmenting attributes at the top of the tree. In addition to being scientifically proper, this decreases the execution time of the Calculation Stage because there are fewer combinations for the stage to consider and because other attributes such as Brand with a very large number of values will be moved down the tree, where fewer items are involved in the calculation. For more information, see Setting the Top Attribute.
5. For large categories, consider setting the SKU Percentage of the termination condition to a lower value (possibly 0%). A value of X in this field in the Calculation Stage UI specifies that a branch will end when a node on the branch contains fewer than X% of the items in the category. If X = 5%, which is the default, and the category contains 2,000 items, then the branch ends when a node on the branch contains fewer than 100 items. A threshold of 100 is probably too high, and if left at 5%, various branches may not be expanded to their

full extent. If the value is expanded until a node on the branch contains fewer than 10 items, the SKU percentage field must be set to 10 / 2,000 = 0.5%. However, the field only accepts integer percentages, and so it must be set to 0%, which will let the Calculation Stage use other internal criteria to end a branch. This field can also be used in a reverse manner; that is, by setting a higher value the tree will become shallower and the calculation time will be reduced. For trial runs, you may wish to leave it at 5%, and see how far the branches are expanded before trying a run with a setting of 0%. For smaller categories, with fewer than 1,000 items, using the default of 5% is likely to be reasonable and no adjustment is needed. The use of percentage in this field can also be handy if you are performing runs that have more than one category. If the categories are related, so that you want trees of roughly the same depth for them, the percentage nature of the field will help produce this result.

Advanced Use

The process described in Calculation Stage is intended as a starting point, and is the shortest path to getting one CDT per category. Once that has been achieved, it is possible to consider some more advanced uses of the CDT application. Here are some suggestions for these more advanced uses. These are not steps to be performed but individual suggestions.

- Generate CDTs that are specific to location or segment. The Calculation Stage can generate CDTs that are segment specific or location specific, by unchecking the appropriate Top level check boxes in the stage. This makes it possible to see whether purchasing behavior differs by segment or by location. Note that on large categories with more than 1,000 items, such a calculation can take many hours, because the stage must calculate one CDT per segment or per location. Note also that it is *not* advisable to uncheck both check boxes, as that will produce a CDT for each segment/location combination. This is a large number of CDTs that will take a very long time to run.
- Set up grouped attributes. See Grouped vs. Raw Attribute Values.
- Experiment with setting different attributes as the top attribute, or with not setting a top attribute at all. See Setting the Top Attribute. Different settings here can produce different insights. However, keep in mind the points raised in Handling of the Brand Attribute.

4

Demand Transference

This chapter provides details about the use of the Demand Transference application.

DT and CDT

The DT and CDT applications differ in significant ways. The CDT application has more stringent requirements for data than the DT application. CDT requires customer-linked, frequent transactions. Many retailers in various areas of retail do not have this type of data readily available. DT only requires SKU-store-week sales-units aggregates.

Demand Transference Model

A mathematical model of how the transference happens is required in order to calculate the transfer of demand in response to assortment changes. It is essential to understand the model at a basic level in order to best use DT. DT generates parameters that go into the model, so an understanding of the model can help when using the DT parameters.

The model is known as a cannibalization model. In this type of model, each item in an assortment has an associated value called its "full demand," which is the demand the item would have if it were the only item in the assortment. The full demand of an item is then multiplied by a factor, called the "cannibalization factor," which has a value of 1 if there are no other items in the assortment, but becomes progressively less than 1 as more and more items are added to the assortment. As the assortment becomes larger, the demand for each of the items decreases from its full demand because of cannibalization. The reverse is also true. If items are removed from the assortment, then the cannibalization factors increase, representing demand transferred from the removed item to the items remaining in the assortment. The cannibalization factors decrease from a value of 1 when the assortment becomes larger, and increase (up to a limit of 1) when the assortment becomes smaller.

The degree of change in an item's cannibalization factor indicates how similar the added items are. Item A's cannibalization factor will decrease more for added items that are very similar to A. The similarity of items is a key input to the Demand Transference model.

The cannibalization factor of an item accounts for similarities and also for a quantity called "assortment elasticity." The assortment elasticity determines how much of a decrease in the cannibalization factor occurs due to the addition of items of a particular similarity. The assortment elasticity is a number that depends on the particular category for which demand transference is being calculated. In one category, adding item B to the assortment may cause item A's cannibalization factor to go from 0.7 to 0.6, whereas in another category, adding an item Y may cause item X's cannibalization factor to go from 0.7 to 0.5, even though the similarity of X and Y is the same as the similarity of A and B. In other words, similarities alone are not enough to calculate cannibalization factors. The assortment elasticity is necessary to tell

us, for each category, how much change in cannibalization factors will occur for items of a given similarity.

The two components of the cannibalization factor, the similarities and the assortment elasticity, are calculated by DT from historical data. (The similarities can also be imported instead of calculated.) DT then exports the similarities and the assortment elasticities to any applications that want to calculate demand transference. It is up to the consuming application to properly use the cannibalization model in conjunction with the exported similarities and assortment elasticities to calculate transfers of demand when assortments change.

Note that demand transference only occurs within a category. All calculations are based on items cannibalizing each other, and there are no complimentary (halo) effects. DT calculates assortment elasticities at a level always higher than item. A single category/segment/location combination receives just one assortment elasticity.

An Example

This simple example explains how applications such as Category Management Planning and Optimization (CMPO) use the demand transference model to generate forecasts after assortment changes. In an assortment in the Cookies subcategory, one cookie SKU is removed from the assortment. The cannibalization factors of the rest of the items increase, because each is now cannibalized less after the removal of the one SKU. Because the cannibalization factors increase, the model predicts an increase in sales in accordance with the increase in the cannibalization factors. The removal of the SKU caused these increases, and some of the SKU's demand has transferred to the other SKUs.

Historical Similarity Data

DT has two different options for obtaining similarities. It can calculate them by itself or it can import them from CDT.

If the retailer has not implemented CDT for a category, then of course only the first option is possible.

The second option is recommended for a category only if the retailer has implemented CDT and run it for the category, since only in that case are similarities for the category available from CDT.

If CDT similarities are available, the recommendation is that you use them, instead of having the Similarity Calculation stage calculate its own similarities. The similarities from CDT are generally preferable to the attribute-based similarities that DT can calculate on its own because CDT similarities do not rely on attributes. They are extracted purely from historical transactions data.

The transactions data held in the RADM schema is used to feed both CDT and to generate the SKU-store-week aggregates for DT, so in option 2, consistency between the similarities and the SKU-store-week aggregates is automatic.

The similarities obtained from CDT may not cover all of the SKUs that are currently in the historical data loaded for DT. For example, it is possible that since the CDT similarities were calculated, the retailer has added some new SKUs to some assortments. This situation requires no special handling, because DT can extend the CDT similarities to cover the added SKUs. This extension does require attribute values for the new SKUs.

Note that the CDT similarities for a category may be only at Segment-Chain/Location-Chain. In this case, there is only one set of similarities for the category, or they may exist at various levels of the location hierarchy or the segment hierarchy, depending on what options the user selected in the CDT application's Calculation stage.

Historical Sales Data

DT requires SKU-store-week sales-units aggregates. The data loader for DT automatically produces the needed SKU-store-week sales-units aggregates from transactions data that is held in the RADM schema, so it is not necessary to implement a separate loader for SKU-store-week aggregates.

Typically, the data cannot be aggregated to a higher location level than store because different stores usually have different assortments. Some atypical cases can occur in which aggregation across some stores is legitimate because the assortments are the same or nearly so, but this is generally not the case.

In addition to the SKU-store-week sales-units aggregates, DT also requires promotion data. A flag indicates which SKU-store-weeks contain major promotions (ones that caused a very large increase in sales units, such as three times normal). DT uses the promotion data to flatten the promotional spikes in the SKU-store-week sales-units aggregates. DT uses the flattened data called the baseline to calculate assortment elasticity.

DT calculates the assortment elasticity by examining the historical assortment changes and seeing their effect on base sales rates of the items remaining in the assortment. Promotional spikes can affect this calculation by obscuring the true effect on base sales rates. These promotional spikes are removed to decrease the sales rates back to their base rates.

For example, suppose the historical data for a store S indicates that the Cookies assortment has one fewer SKU in week 10 compared to week 1. That is, a cookie SKU was removed. To see where the demand from this SKU transferred to in week 10, the sales units of the remaining SKUs between week 1 and week 10 are compared. This comparison is made across many pairs of weeks (though not all possible pairs of weeks). A promotion in week 10 of particular items can interfere with the analysis of the changes in demand that were due to the assortment changes.

Note that, in CDT, the effect of promotions is left in because it is an external influence that helps cause switching behavior in customers. However, in DT, promotions can affect the calculation of demand transference in the case where one item in the category is promoted and another is not, which is why the promotions are flattened.

An alternative to flattening the promotions is to ignore the SKU-store-weeks where promotions occurred. However, to implement this, it is necessary to ignore all SKU-store-weeks in any week where a SKU-store was promoted, because it is not clear what the effect is on the other items when one SKU is promoted and the others are not. Removing that many SKU-store-weeks can leave little data remaining, especially since many retailers promote quite frequently. For this reason, it is better to flatten promotions and keep more data instead.

Similarly to the way in which promotions are handled, the calculation of the baseline also involves removing short-term downward spikes that are due to very short-term stock outs. Note that inventory information is not an input to the DT system, and so the algorithm finds large but short-lasting dips in weekly sales units and fills those in. This handling of out-of-stock is not related to long-term out-of-stock conditions, discussed in a later section.

The Role of Attributes in Calculating Similarities

Without customer-linked transactions data, DT must use the attribute values of the SKUs to calculate similarities. The similarity of two SKUs is based in part on how many attribute values they have in common (the more in common, the higher the similarity of the two SKUs). The attributes used in the calculation are the raw attributes, not the grouped attributes that CDT uses. So it is not necessary to group the attribute values for DT.

Because the attributes play such an important role in calculating similarities, attribute quality is important when DT performs the calculation.

Attributes are also used in performing any necessary extensions of the CDT similarities to cover new SKUs.

Note that the similarities calculated by DT are only at Segment-Chain/Location-Chain. In contrast, the CDT similarities can be at multiple levels.

Attribute Data Requirements

The attribute values for the DT calculation must meet the following requirements:

Set of Attributes

Each category is characterized by a unique set of attributes. These attributes differ from category to category. For example, for yogurt, the attributes might be size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes might be size, brand, milk/dark, nut type, and package type. Two categories can both have brand, but the brand attribute will have different values for each of the categories. So brand is actually a different attribute for each category.

Mapping

Each item in the category must be mapped to its set of attribute values. This information must be obtained from the retailer. Null values are acceptable as long as they are not too numerous. DT can still run even if some attribute values are listed as null for some items in a category, but too many null values decrease the reliability of the generated DTs. In particular, too may SKU pairs may come out as less similar than they should be, which would decrease demand transference between those pairs (which leads to an underestimate of demand transference in applications such as CMPO).

Null values have a particular use in accommodating categories that are actually a union of more than one category. See "Setting Up Categories" for more information.

Significance

The attributes for a category must be the ones that the customers actually pay attention to when shopping in the category. They are attributes that actually affect the customers' purchasing decisions.

Note that the similarity calculation will still complete even with attributes that do not affect customer behavior, but the similarities produced will be less distinguishing. For example, a category has a Supplier attribute, which indicates for a given product which supplier shipped an item to the grocer. This attribute may be important to the grocer for accurate bookkeeping, but it has no effect on the customer's purchasing behavior because it is not reflected in the item itself nor is it something that the customer is concerned about. However, if it is included when setting up attributes, then the effect would be to increase the similarity of items that were from the same supplier. This is a false similarity, since it does not reflect how the customer actually views these items. In particular, if the supplier is a duplicate of the Brand attribute, then the similarity of products within the same Brand would be unintentionally increased.

The process of obtaining attributes for a category and performing a mapping of items in the category to attribute values is likely to require a significant amount of time and labor, even if the retailer has the information available, since this must be done for every category.

Guidelines on Number of Attributes and Attribute Values

The number of attributes and attribute values must be enough to distinguish the SKUs within a category. That is, for a given set of attribute values, the number of SKUs in the category all having those values must be a small number. A maximum of seven SKUs is recommended. For example, the Cookie category at a grocer has only three attributes, Brand, Package Size, and

Organic. If Brand has seven values, Package Size has three values, and Organic has two values (either Yes or No), then the total number of combinations of attribute values is 7 x 3 x 2 = 42. For 600 different cookie SKUs, the average for each combination of attribute values will represent 600 / 42 = approximately 14 different SKUs. The distribution of SKUs among the 42 different sets of values will not be an even 14, as some sets of values will have much more than 14, while others will have less. The three attributes alone are not enough to provide enough distinguishing power among the cookie SKUs. If Flavor is an important determinant of customer purchases, it should be added to the Cookie category. The guideline of a maximum of seven indicates that additional attributes are necessary. It is worth examining those sets of attribute values that have the largest number of SKUs associated with them in order to see what attributes can be added to reduce the number of SKUs.

It is not just the number of attributes that is important, but how many values each attribute has. For example, if Brand had 100 values instead of seven values, then the total number of attribute-value combinations is 100 x 3 x 2 = 600. It might seem that an easy way to achieve the maximum of seven is to expand the number of values in the attributes. However, this results in each SKU being similar to only a small number of other SKUs. For a single attribute for cookies with 600 different values, it might then be possible to assign one value to each cookie SKU, separating all of the 600 cookies SKUs with a single attribute. However, this would make each cookie SKU completely dissimilar (similarity of 0) from all other cookie SKUs, and the result would be no transference between the SKUs. Putting all 600 SKUs each into a separate Brand causes a complete loss of any similarity information among the SKUs, and no transference will result. For an opposite example, consider 11 attributes, each with only two values. There are a total of 2^11 = 2048 combinations of values, so that may be enough to encode 2,000 SKUs, even though there are only 2 x 11 = 22 distinct attribute values over the 11 attributes. In general, having more attributes is better, and it is better to increase the number of attributes rather than increase the number of attribute values of a single attribute. However, this is not always possible, and it is better to have the attribute with many values than not have the attribute at all. Flavor, for example, can have many values, as can Color. See "Avoiding Attributes with Many Values" regarding attributes that have many values.

The more SKUs in the category, the more attributes and attribute values will be needed to achieve the maximum of 7.

The Effect on Similarity Values

Suppose the set A of SKUs consists of 22 SKUs, all with the same attribute values, and the set B of SKUs consists of 25 SKUs, all with the same attribute values (but a different set of attribute values from set A). If the set A consists of cookie SKUs all with a package size of Small, and set B consists of cookie SKUs with the same attribute values as A except the size is Medium, then every SKU in A has a similarity of 1 to every other SKU in A, and every SKUs in A is similar to at least 22 other SKUs. Every SKU in A is similar to every SKU in B, since they only differ in one attribute value (namely size). So a SKU in set A is similar to at least 21 + 25 = 46 SKUs, which means that if a SKU in A were deleted from an assortment, its demand would have significant transference to about 46 other SKUs, assuming all 46 remained in the assortment. It is possible that a SKU in A being similar to 46 other SKUs in fact represents reality, but if it does not, then using additional attributes that distinguish the SKUs in A and in B will reduce the number of similar SKUs.

Avoiding Attributes with Many Values

Attributes with a large number of values occur frequently. For example, a color attribute in any clothing category might have several shades of each color. Midnight blue, sea blue, and sky blue may all be separate attribute values of the Color attribute; the problem is that in the similarity calculation, a midnight blue item and a sea blue item would be considered completely dissimilar colors, because the two color attribute values are different; in reality, because they

are both shades of blue, they should be somewhat similar. One solution is to split the color attribute into two separate attributes, a primary color attribute and a modifier. In this example, the primary color would be blue and the modifiers midnight, sea, and sky.

Functional-Fit Attributes

A functional fit attribute is one where there is no substitution across the attribute's values. For example, batteries of different sizes cannot be substituted for one another. Any category where size determines the functional suitability of the item will have size as a functional-fit attribute.

Information about which attributes are functional fit ones must be loaded into DT. The information is used to perform the similarity extension process of CDT similarities and to correctly calculate attribute-based similarities.

In either case, the functional-fit attributes are used to set the similarity of two SKUs to be 0 if the SKUs differ in any functional-fit attribute. Without the functional-fit information, the two SKUs may have non-zero similarity, and there would be erroneous demand transference between the two SKUs, such as batteries of different sizes.

Designating an attribute as functional fit can also be useful any time the attribute is unlikely to have substitution across it (for example, caffeinated vs. decaffeinated coffee). This is not exactly functional fit; however, substitution is unlikely, so it is better to mark the attribute as functional fit.

One approach to avoiding having to define large numbers of attributes and attribute values is to use functional-fit attributes. This approach does not help achieve the maximum of 7, but it can help decrease the number of SKUs that are similar to a given SKU. For example, with the sets A and B of cookie SKUs, if size were designated as functional fit, then the similarity between SKUs in A and SKUs in B would become 0. However, that designating size as functional fit does nothing about the 22 SKUs in A that all have a similarity of 1 to each other, since their attribute values are all the same. (Similar comments apply to set B.)

If the attributes and attribute values are insufficient to reach the maximum of 7 SKUs per set of attribute values, functional-fit attributes can be used to decrease the number of SKUs to which transference occurs. This is a second-best approach, and it is better to design a proper set of attributes and attribute values, in order to:

- Achieve the maximum of 7 SKUs
- Provide transference between SKUs that should have transference. Using functional-fit attributes reduces transference, but it may reduce it too much and remove transference from pairs of SKUs that should have transference. For example, in the sets A and B, the similarity between a SKUs in A and a SKU in B becomes 0, which does not reflect reality since the SKUs in A and in B share common attribute values except for size.
- Keep the second-best approach as a last resort, in case time is insufficient for designing a good set of attributes for a category.

Using Null as an Attribute Value

Null is a legitimate attribute value for an item to have; it denotes either of the following:

- The attribute value is unknown. In this case, the item has a value for this attribute, but the value is unknown. Attribute data for a category is rarely perfect, it is entirely possible that attribute values for some items were never recorded or are known to be incorrect. The use of null because of imperfections in the attribute data must only be for those cases where it is truly necessary.
- The attribute does not apply to the item. Not all attributes in a category apply to all items, especially when the category consists of sub-categories (see below). The ideal is to define

the items in the category so that every attribute for the category applies to every item, but this may not be possible. This usually occurs when an attribute applies to an item only if the item has a particular value for another attribute. For example, in the meat snacks category, the attribute grass fed only applies if the item has beef for the product type attribute. If the item is turkey then the grass fed attribute must be set to null for the item. The grass fed attribute must be true or false only for beef products.

Using null is not the ideal way of handling the second case listed above. If possible, it is preferable to split the items in the category into separate categories. For example, this means putting beef products into their own category, so that the grass fed attribute applies only to this new category and can be removed from the non-beef products. This is only reasonable if the beef products are not substitutes for the other products in the meat snacks category (which is probably not true since the customer might reasonably substitute turkey jerky for beef jerky). In the case of meat snacks, the best solution is to use null for the grass fed attribute for non-beef products.

The Effect of Null Attribute Values on Similarity Values

A null attribute value for an item means that the item does not match any other item in that attribute, even if the other item also has null for the same attribute. Null as the value of an attribute makes the item less similar to every other item, including items that also have null for the same attribute. If an item has null for every attribute, then this item has a similarity of 0 to every other item, regardless of what values the other items have for their attributes. A value of null for an attribute is thus treated as special in similarity calculations. It is recommended to keep the use of null to a minimum in order to avoid spuriously indicating that items are dissimilar when they are actually similar.

If the intention is to have null for an attribute value, it is essential to actually use null and not create a string such as "none" to use where null is actually meant. Designating a string such as "none" to use in place of null actually does the opposite of what is described above, because two items with a value of none for the same attribute will now be considered similar in that attribute since they have the same value for the attribute. This is directly opposite to the intention of null, as described above.

Categories Containing Sub-Categories

As discussed in "Setting Up Categories," it is possible for a category to consist of items from several categories. As it may be impractical to separate the category into its component categories, it is necessary to create attributes for the category that can approximate the separation. The following process describes how to do this, using " the category Oral Care. This category consists of: toothbrushes, toothpaste, dental floss, and mouth wash; it is really four separate categories whose items have all been classified together into a single category.

1. Create one attribute that can separate the category into its component categories. For example, for Oral Care, create an attribute called Product Type that has four values: "toothbrush," "toothpaste," "floss," and "mouthwash." The values of this separating attribute should describe the function of the product. It should separate the category into groups of items in which the items within each group are unlikely to substitute for items in another group. For example, toothbrushes are unlikely to substitute for toothpaste. Constructing a separating attribute is not always as simple as this, as discussed below.
2. Designate this separating attribute as functional fit. This means that no transference will take place between items that are in the different groups specified by the separating attribute. In this case, there is no transference between the four component groups of Oral Care.
3. Create a single set of attributes suitable for all of the component categories. This typically involves taking the union of the attributes that describe each component category

separately. For example, the set of attributes that describe toothpaste are different than the set of attributes that describe toothbrushes, but in this scheme, just take the union of both sets of attributes. Toothbrushes have an attribute called "Bristle hardness," which does not apply to toothpaste, and likewise toothpaste has a flavor attribute that does not apply to toothbrushes. Nonetheless, take all of the attributes together, including the ones for floss and mouthwash. It is possible that an attribute can apply across several of the component categories. For example, only a single flavor attribute is necessary, instead of having three attributes called Floss-Flavor, Toothpaste-Flavor, and Mouthwash-Flavor. Either approach is correct, but having three separate flavor attributes may require more work to create.

4. Assign null values appropriately. For example, for attributes that apply only to toothpaste, set them to null for any toothbrush. Likewise, for attributes that apply only to toothbrushes, set them to null for any toothpaste. Table 4–1 provides complete details for this example.

In step 1, you may not find a separating attribute. Instead, you may need to use an attribute that separates the items into groups but in which some of the groups may have items that can substitute for each other. For example, consider a category that consists of Shampoo, Shampoo and Conditioner, and Conditioner, where Shampoo and Conditioner are products that have shampoo and conditioner in one bottle. In this case, a separating attribute has three values, Shampoo, Shampoo and Conditioner, and Conditioner. If this attribute is set as functional fit, there is no transference between Shampoo and Shampoo and Conditioner. In this example, the approximation may be good enough, since although substitution can take place between Shampoo and Shampoo and Conditioner, it is probably not large.

Here is an example of the Oral Care category and how to set null for its attributes. Assume a separating attribute for Oral Care as described above, and assume, in addition, the following attributes: Thickness, Length, Contains Alcohol, and Flavor. (In reality there would be more.) Table 4–1 shows which attributes apply to which products. Null indicates that the attribute does not apply to the product, and therefore should be set to null for the product.

Table 4–1 Oral Care Category Example

Product Type	Thickness	Length	Contains Alcohol	Flavor
Floss	Applies	Applies	Null	Applies
Mouthwash	Null	Null	Applies	Applies
Toothpaste	Null	Null	Null	Applies
Toothbrush	Null	Null	Null	Null

Customer Segments

DT can calculate assortment elasticities by customer segment. This involves dividing the customer IDs into groups (the groups do not have to be disjoint). Retailers who want to use segments must, as with CDT, create the necessary groupings of customer IDs. DT uses the segments to produce segment-SKU-store-week aggregates of sales units, instead of just SKU-store-week aggregates. The segment-SKU-store-week aggregates are produced by aggregating transactions data, just as with the SKU-store-week aggregates. The difference is that the aggregation is by segment.

There is always a Segment-Chain for the segment hierarchy, and so there is always a segment that contains all customers. The Segment-Chain level of segment-SKU-store-week aggregates is not necessarily the sum of the lower-level segment-SKU-store-week aggregates, because it is possible that the segments are not disjoint (meaning a customer can belong to more than one segment). The Segment-Chain-level aggregates are produced by a separate aggregation of transactions data instead of by aggregating lower-level aggregates.

Using segments allows DT to calculate separate assortment elasticities for each segment. This means that demand transference can differ by segment.

Note that when using customer segments, references in this document to "SKU-store-week" data should be read as "segment-SKU-store-week" data. For example, the SKU-store-week sales-units aggregates mentioned above become segment-SKU-store-week sales-units aggregates.

Location Hierarchy

DT supports calculating assortment elasticities by location hierarchy. The lowest level of the hierarchy should be above store; in general, assortment elasticities should not be calculated per store. Per-store assortment elasticities may have too little data to be reliable. The calculation time involved can be quite large to handle all stores individually. The calculation of assortment elasticities depends on having assortment changes in the historical data, and the store level may contain too few assortment changes to produce reliable assortment elasticities.

Some retailers may have stores that differ in size and assortments. For example, a grocery chain may have both convenience stores and supermarkets. It may be necessary to arrange a separate calculation of CDTs for convenience stores vs. supermarkets, because people may shop differently at the two types of stores and the assortments may be different at the two types of stores.

One approach to this is to arrange a separate calculation by creating separate store clusters for convenience stores vs. supermarkets. DT has the capability of calculating CDTs for each element of the location hierarchy, so it can calculate CDTs for the separate store clusters and thus produce separate CDTs for convenience stores vs. supermarkets.

Setting Up Categories

In general, a category is a set of items that are substitutable with each other (if there are no functional-fit attributes). The categories at a retailer can all be derived by choosing the correct level of the merchandise hierarchy at the retailer. The DT configuration supports choosing which level of the merchandise hierarchy is to be used as the category level.

Demand transference can only occur within the category, since the categories define the sets of items that cannibalize each other.

A retailer may want categories that consist of unions of nodes of its merchandise hierarchy because no level of its merchandise hierarchy suffices as the category level. DT does support this, in that it allows defining an alternate merchandise hierarchy, where the categories can consist of arbitrary collections of items. However, before investing time in setting up an alternate hierarchy, make sure that it is necessary for meaningful DT calculations.

For example, it is possible that the set of all yogurt SKUs at a retailer is not at any level of the merchandise hierarchy. The retailer may have the category Dairy Products, which is too large because it contains yogurt and milk, and the retailer might have the category Store-brand Yogurt, which is too small because it leaves out the yogurt SKUs that are not store brand. In such a case, it may be necessary to set up an alternate hierarchy so that all the yogurts can be put together in their own set. On the other hand, if a level of the existing merchandise hierarchy contains most of the yogurt SKUs, but not quite all of them, an alternate hierarchy may not be worth the effort.

Frequently, retailers will have categories that are actually unions of categories. For example, a retailer might have a Hair Care category that contains shampoo, conditioner, and hair oil. The retailer may not want to separate out this category into three separate categories of Shampoo, Conditioner, and Hair Oil, if, for example, a single person in the organization is responsible for all three. The problem is that these three types of products do not share a common set of attributes. The attributes describing Hair Oil are not the same ones needed for describing

Shampoo or Conditioner. The types of products may share common attributes, such as Scent, but each type of product also needs its own set of attributes. The solution is to define, in addition to the common attributes, a set of attributes for each product type. If an attribute applies only to Shampoo, and not to Conditioner or Hair Oil, then Conditioner SKUs and Hair Oil SKUs should have Null for the value of that attribute. This is a common use of null attribute values and makes it possible to handle the case of a category that is really the union of smaller sub-categories.

5

Using Demand Transference

This chapter provides details about using Demand Transference.

Seasonality in Historical Sales Data

The DT application assumes that, within a category, all of the items at a store have a common seasonality. This assumption is generally correct for long life cycle categories, where each item does not have a predetermined point of obsolescence or where the point of obsolescence is years from the point of introduction of the item. Examples include most grocery categories and basic clothing items. Electronics items frequently have defined life cycles, generally measured in years.

It is important to address the situation in which different items in the same store have different life cycles and those life cycles are short. In this situation, the store may have items that are at various points in their life cycles and there is no common seasonality. This frequently occurs with fashion merchandise (see "Implementing DT for Fashion Categories").

Assortment Elasticity

An assortment elasticity of 0 turns all cannibalization factors into constant 1, meaning the assortment has no cannibalization. This is unlikely. However, it does show that a small-magnitude value for assortment elasticity indicates a category where cannibalization is small. Similarly, a high magnitude of assortment elasticity indicates a category where cannibalization is large. It is possible for the magnitude to be too large.

It is also possible for the Calculation stage of DT to produce assortment elasticities that are positive. Such positive values for assortment elasticity are an indication that there is some unidentified problem with the data, because a positive assortment elasticity means cannibalization factors increase with increasing assortment size, which in turn means each item in the assortment sells more the larger the assortment gets. In the Evaluation stage of DT, such positive assortment elasticities are removed and replaced by assortment elasticities from escalation.

Here is an example that explains the cannibalization model. In this example, all the cannibalization factors are equal; in a real example, the factors would all be different.

Identical, or nearly identical, items are being added to an assortment. If only one of these items is added to the assortment, it takes the entire market share for the item. When another item that is extremely similar is added, the two items split the market share evenly between them. The cannibalization factors are now half for both items. The addition of a third such item creates a three-way even split, one-third for each and the cannibalization factors are each one-third. As more such items are added, the cannibalization factors approach but never reach zero. (Parenthetically, this example also shows how adding items to an assortment does not

necessarily produce more market share overall for the assortment, since the new item may simply siphon off sales of existing items.)

The cannibalization factor is actually a power-law, that is, the assortment elasticity enters into the cannibalization factor as an exponent. The cannibalization factor consists of a positive value, the Total Assortment Effect (TAE), raised to the assortment elasticity. Each item in an assortment has its own TAE; the TAE increases as items are added to the assortment. Therefore, the assortment elasticity is a negative number, in order for the cannibalization factor to decrease as TAE increases. (In the example, the TAE could simply be the count of the number of items added so far, and the assortment elasticity would then be -1, thus producing 1/2, 1/3,....) Conversely, the TAE also decreases as items are removed from the Assortment.

Note the similarity to the more conventional idea of power-law price elasticity, which involves a price raised to a negative power (the negative power being the price elasticity). In the cannibalization model, the TAE plays the role of price. Like price, the TAE can change week to week because the assortment can change week to week and the TAE is designed to reflect the current assortment.

The cannibalization factor also accounts for the similarity of the items being added to the assortment, so that similar items cannibalize each other more than non-similar ones. The similarity values are used to calculate the TAE; higher similarities produce a larger TAE, providing a larger decrease in cannibalization factors.

The TAE also depends on the relative sales rates of the items in the assortment. Although higher similarities increase TAE, as discussed above, that cannot be the whole story, because cannibalization depends on the sales rate of the item doing the cannibalizing. For example, consider two items A and B that have equal similarity to an item C that is already in the assortment. Adding item A will cannibalize item C, and so will adding item B; however, if A is a low selling item and B is a high selling item, adding B will cannibalize C more than adding A will. So the TAE must account for the relative sales rates of the items in the assortment in addition to accounting for similarity. The TAE does this through the sales index of an item, which is the sales of the item divided by the average sales of the items in the assortment (total sales divided by the number of items in the assortment). This calculation is done in each week, with the assortment in that week.

The following example demonstrates these concepts together and also shows some of the robustness of this model. Consider an assortment that includes items A and B and in which A has a vastly higher sales rate than B. The TAE for item A includes a term for item B, and as discussed above, the contribution of B to the TAE depends on its sales index. In this case, the sales index of B is quite small, because A is so large. Removing B from the assortment will change the TAE of A very little, and so the change in A's sales due to removal of B is quite small. In a situation where A's sales become increasingly large, the sales index of B will go to 0, and the effect of the removal of B on A tends to 0. This is as it should be, since when A becomes so much larger than B, the removal of B cannot have much effect on the sales of A.

The cannibalization factor depends on both the similarity values and the assortment elasticity. It might seem that similarity alone determines cannibalization, as a similarity of 0.5 between items A and B means that A takes half of B's share if A is added, but that is not the case. In particular, by separating the concepts of TAE and assortment elasticity, the model is more robust; if all of the similarity values are biased lower or higher for some reason, the bias can be accounted for by adjusting the magnitude of the assortment elasticity so that the cannibalization factors are still correct.

The Importance of Assortment Changes in Historical Data

In order to calculate assortment elasticity, DT requires historical data that contains assortment changes, because DT examines historical data to determine how much cannibalization occurred when historical TAEs changed. From the relationship between changes in historical TAEs and

changes in cannibalization, DT then calculates the assortment elasticity. This is similar to calculations of price elasticity. In order to determine price elasticity from historical data, it is necessary to have price changes in the historical data, and the more changes the better.

Using the example from "Assortment Elasticity", suppose, in the historical data for a particular store *S*, that the cookies assortment has one fewer SKU in week 10 compared to week 1 (that is, some cookie SKU was removed). The TAEs for the other remaining cookie SKUs will all decrease between week 1 and week 10 because of the removal of the one SKU. DT then examines the changes in historical sales units of the SKUs in the cookie assortment at *S* between week 1 and week 10. By relating the changes in the sales units to the changes in TAEs, DT can calculate the assortment elasticity. A larger magnitude elasticity will result if the changes in TAE caused a large increase in sales units; a smaller magnitude elasticity will result if the increases are moderate.

In reality, the comparisons in this historical analysis that DT does are always more complex than in this simple example. It is rare to find a pair of weeks where the assortment change was just removal of a single SKU. Typically, in each pair of weeks, there are many assortment changes, involving both additions and removals, and the changes in TAE are a result of all of those changes. In the end, though, the relationship between the changes in TAE and the changes in sales units are summarized in a single number, the assortment elasticity, across all pairs of weeks. Because this single number summarizes the number of pairs of weeks and SKUs where TAEs changed, it is necessary an average over all the pairs of weeks and SKUs in the historical data and is not tuned to any particular SKU.

If Category Management Planning and Optimization (CMPO) is used to remove a single SKU from an assortment, it is likely that no pair of weeks in sales history exists in which exactly this SKU was removed and only this SKU was removed. For forecasting the results of this removal, CMPO makes an extrapolation from the historical analysis described above and uses the assortment elasticity that is not tuned to this particular situation of removing only this one particular SKU.

Estimation of Assortment Elasticity

One possible approach to the estimation of assortment elasticity is to estimate it jointly using base rates of sale for each SKU-store and seasonality parameters. However, the estimates of base rates of sale or of seasonality are not required, and an estimation technique that will estimate only assortment elasticity is used. Seasonality is taken into account by assuming that the entire category of items has similar seasonality at a given store, and then applying a differencing technique to get rid of seasonality, leaving base demand and assortment elasticity. For a particular SKU X at store S, the estimation uses pairs of weeks as follows: Suppose W1 and W2 are a pair of weeks where the assortment for the category at W2 at S is different from that at W1. The ratio of sales of X at W2 to sales of X at W1 is taken. This ratio is the dependent variable of the regression, and the independent variable is the assortment difference between W1 and W2. As explained above, the effect of the assortment on X is encoded as the TAE of X and this ratio of sales must be explained in terms of the ratio of the TAE of X between W1 and W2. The input to the regression is many such pairs of weeks for each SKU and each store, and the output of the regression is a single assortment elasticity over all the SKUs and stores of the input.

The weeks in each pair are reasonably close together, so that the weeks are more likely to represent similar conditions and provide a better estimate. That is, with such weeks, the only change is likely the assortment change, and it is more likely that the difference in sales units is due to the assortment change than to other external factors.

Estimation performs a weighted regression, with the observations from lower-selling SKUs weighted less than observations from higher-selling SKUs, because lower-selling SKUs have

more volatile sales and thus should be weighted less according to statistical theory concerning linear regression.

Note that since the estimation uses ratios of sales units of SKU X, the absolute quantity of the sales units does not matter (except for the weighting in the weighted regression) for the estimation.

The removal of seasonality described above requires that the weeks W1 and W2 in a pair share common items in the assortment within a store. For the pair of weeks to be in the regression, the weeks must have at least five items in common, though this is configurable. Category-store combinations in which the store carries very few items from the category may not meet the requirement of five common items, and a value smaller than five may be necessary, depending on the importance of such category-store combinations. Note that requiring fewer common items results in a poorer seasonality correction, and that five is already a very low number. For example, grocery clients for the demand-transference application usually have assortments numbering in the hundreds.

The requirement for common items within-store means each store can have its own seasonality, though this requires a number of common items.

In the above description of the regression, W1 and W2 were weeks, but in typical use, the regression uses longer time periods, with a default of four weeks. So W1 and W2 in the default configuration are four-week periods, and the regression uses the average sales of SKU X over those periods. This helps smooth out any noise in the sales of X, and also makes the regression more amenable to low-selling SKUs. Note that extreme low sellers, such as items that sell fewer than one unit in four weeks, will likely still present a problem for the regression.

The Meaning of the Possible Values of Assortment Elasticity

An assortment elasticity of 0 turns all cannibalization factors into constant 1, meaning the assortment has no cannibalization. This is highly unlikely. However, it does show that small-magnitude assortment elasticity indicates a category where cannibalization is small. Likewise, a high magnitude of assortment elasticity indicates a category where cannibalization is large. It is possible for the magnitude to be too large (see "The Substitutable Demand Percentage").

It is also possible for the Calculation Stage of DT to produce assortment elasticities that are positive. Such positive values for assortment elasticity are an indication that there is some unidentified problem with the data, because a positive assortment elasticity means cannibalization factors increase with increasing assortment size, which in turn means each item in the assortment sells more the larger the assortment becomes. This is presumed to be a nonsensical result, and, in the Evaluation Stage of DT, such positive assortment elasticities are removed and replaced by assortment elasticities from escalation (that is, the elasticities are replaced with higher-level ones).

It is possible, with sufficient data analysis, to figure out what problem with the historical data caused the positive assortment elasticity. However, such analysis is difficult to automate, and escalation is used instead.

The Substitutable Demand Percentage

The substitutable demand percentage, or just substitutable percentage, of an item in an assortment is the fraction of its demand that is retained by the assortment if the item is removed from the assortment. It is a measure of how substitutable the item is. For example, if the substitutable percentage is 100 percent, then removing the item will not decrease the total sales units of the assortment, since all of the demand for the item will transfer to the other items that remain in the assortment. If, on the other hand, it is 50 percent, then the removal of the item

from the assortment means that 50 percent of its demand is lost, and 50 percent is retained. The total assortment sales units will decrease if this item were to be removed from the assortment.

The magnitude of the assortment elasticity has a influence on the substitutable percentage. Increasing the magnitude of the assortment elasticity increases the substitutable percentage. DT only calculates the assortment elasticity for the entire category (not per item), so changing the value of the assortment elasticity changes the substitutable percentage for all items in the category all at once.

It is possible for the magnitude of the assortment elasticity to be too large. This is indicated by a substitutable percentage for several of the items in the assortment that is over 100 percent. A few items can have substitutable percentages over 100 percent, because those are probably outliers. If the assortment is large, it is likely that a few such outliers exist. If 10 percent of items in the assortment are over 100 percent, then the results should be examined.

DT provides a tool for examining the substitutable percentage and for decreasing the assortment elasticity if too many items have a substitutable percentage over 100 percent. Here are some guidelines for using this tool.

- When selecting the time interval for the tool, select one that is likely to contain assortments that are representative of the retailer's current assortments. Since the retailer is going to be using the assortment elasticity in forecasts of what happens when current assortments are modified, it makes sense to test the assortment elasticity against assortments that are as similar as possible to the current ones.
- It is possible to use the tool to dial down the assortment elasticity. Using Setting maximum substitution percentage, DT calculates an assortment elasticity that results in substitution percentages that do not exceed the set maximum. When using this feature, you may want to set the maximum to a value higher than 100 percent if there are some outlier items that have high substitution percentages. Forcing these outliers down to 100 percent may result in a small-magnitude assortment elasticity, which may mean unacceptably small substitution percentages for all except the outlier items. You may want to select a maximum that is higher than 100 percent but that still brings most items down to 100 percent, leaving a few outliers above 100 percent.
- It is possible to use this tool to set the maximum percentage even if all substitution percentages are already below 100 percent. You may have business knowledge, or a directive from the retailer, and know that a particular category must exhibit a substation percentage of at most 70 percent. In this case, this tool can be used to bring the substitution percentages down to 70 percent. This can make the difference between acceptance and rejection by the client.

No Requirement for a Time Interval

A time interval for the CDT calculation can be set in the CDT Data Setup stage. No equivalent exists in the Data Setup stage of DT.

The cannibalization factor directly incorporates information about the assortment through the TAE, and so the cannibalization model can handle large assortment changes. This makes it less necessary to use a time interval for DT, compared to CDT, because historical assortment changes can be directly accounted for in the model as changes in TAE.

Segments vs. Locations

In the Calculation Stage for both DT and CDT, it is possible to set up the calculation so that it is performed at all combinations of levels of the segment hierarchy and the location hierarchy. This is a more practical possibility for assortment elasticity than for the CDT calculation, because the assortment elasticity is not examined directly by people (unlike the CDTs), and

producing thousands of values will not cause an issue. However, it is recommended to use only one of the two hierarchies in the Calculation Stage. Set either the segment hierarchy or the location hierarchy (or both) to be Chain. Because the calculation of assortment elasticity requires assortment changes in history, generating assortment elasticities at all levels may mean that, at lower levels, the data does not contain enough assortment changes in history. You may want to use your business knowledge of the particular retailer or particular category here, since you may know for the retailer or for the category whether assortment changes are frequent or not in the historical data you have. If the assortment changes are infrequent, you may be better off calculating a Segment- Chain/Location Chain assortment elasticity only.

Setting the Escalation Path

The last stage in DT involves setting the escalation path. If you are using only the segment hierarchy or only the location hierarchy, the escalation path is simply the hierarchy that you are using, and you set the escalation path according to the hierarchy. If you are using both a location hierarchy and a segment hierarchy, then usually you should set the escalation path to go up the segment hierarchy first, and then the location hierarchy. It is better to use only one of the hierarchies.

When using both hierarchies, the escalation path is necessary in order to tell the application which parent it should go to when moving up from a given segment/location node. With both hierarchies in play, every segment/location node has multiple higher-level nodes that do not lie along a single path. The escalation path is necessary to tell the application in what order the higher-level nodes should be considered. When only one hierarchy is used, the higher-level nodes form a single path.

Automatic Updating

DT can automatically and periodically update the assortment elasticities as new sales history is available. This feature is unique to DT; CDT does not perform automatic updating because it makes less sense to automatically produce new CDTs. New assortment elasticities can be loaded into the consuming applications and thus immediately used; however, the value of new CDTs is less clear.

When new historical transactions enter the RA schema, DT will automatically aggregate them and produce new SKU-store-week sales-units aggregates. These new aggregates are then appended to the older SKU-store-week aggregates, and the resulting data set is then used in a new calculation of assortment elasticities.

Note the following about this calculation:

- It does not in any way run the full DT application, that is, re-run all of the stages. The calculation is more targeted and just calculates assortment elasticity.
- It only updates assortment elasticities, not the similarities from the Similarity Calculation Stage.
- Because it uses a mix of old data and more recent data, the values of the assortment elasticities will change slowly over time as the data set becomes more tilted towards newer data. This is by design. It is not desirable to have sudden changes in assortment elasticity, since that would result in sudden changes in cannibalization and demand transference.
- Any assortment elasticities that were overridden using the Substitutable Percentage tool (see "The Substitutable Demand Percentage"), stay overridden, and are not updated.

Avoiding Categories with Small Assortments

It is possible for a retailer to have categories where the assortments are very small, that is, 20 or fewer items in the assortment. Such categories can pose a problem for DT because of the small amount of sales data for just 20 items, and also the number of assortment changes may be quite few.

It is better if the assortment is small but items from a much larger set have been added or removed frequently from the assortment. That is, the category has a much larger set of items, but only 20 of them are in an assortment at a given time. It is possible that the assortment changes were frequent enough that more than 20 items have sales history, and in this case DT results may be reliable even though the assortment is small.

Implementing DT for Fashion Categories

For various reasons, fashion categories require some special consideration. This section describes what is different about them and how to handle the differences.

Proper Level for Fashion Categories

The lowest level of data must not be the SKU level, that is, the Size level of the merchandise hierarchy. Because size is a functional-fit attribute, or nearly so, the level of calculation must be at least one level above size (Style-Color). The historical sales-units data must be aggregated at least up to Style-Color. This also helps avoid problems with low sales rates and noisy sales rates at the SKU level, both common problems for fashion categories. It also helps decrease the number of SKUs within a category, since a multitude of sizes are possible for each Style-Color.

It is worth considering whether color is necessary. Aggregating to the Style level means that transference among colors cannot be calculated. However, it is not clear how useful calculating transference among colors would be, since the colors change for every selling season, and calculating historical transferences among colors may not be particularly useful. A possible halfway approach here might be to aggregate to Style-Primary-Color, where there are only a few primary colors. The primary colors chosen can be the ones that are stable season after season, so that historical transferences among them might be useful in future selling seasons. The primary colors can be chosen to be groupings of the actual colors (so for example, midnight blue and sky blue would become blue). In general, in fashion, the number of colors can be large, and it is unlikely that calculating transferences among such colors would be useful. Aggregating to Style-Primary-Color or even to Style can help avoid low sales rates and noisy sales rates.

It is possible to employ different approaches in grouping the colors. One is to use the primary color. It is also possible to group the colors based on the type of customer the color is designed to attract. For example, the colors can be grouped in "trendy colors" vs. "basic colors." The grouping should be decided in consultation with the retailer, to determine how the retailer uses colors in the category. The retailer may already have a grouping of colors that it uses, and the simplest approach may be to use this grouping.

Here, it is assumed that the aggregation of the historical sales-units data to be either Style-Primary-Color or Style. The term "item" should be understood to mean Style-Primary-Color or Style, depending on the chosen aggregation.

Typically, for fashion, the number of colors is large because of all the color variants. If it is necessary to retain all of the colors instead of following the recommendations above, then it will be necessary to split the color attribute into at least two attributes, a primary color and a secondary color. For more information, see "The Role of Attributes in Calculating Similarities".

Seasonality (Life Cycle) Considerations

DT makes the assumption that the items within a category at a store all have a common seasonality (see "Seasonality in Historical Sales Data"). Because of the short (tens of weeks) life cycles of fashion items, and because items within a category may have different introduction times within the same store, the assumption of common seasonality across the items in a category is probably not valid for fashion categories. It is possible within the category to have items that are at various points in their life cycles within the same store. At the same time, some items may be in the uptrend part of their life cycle, while other items are in the downtrend part of their life cycle.

There are some ways to deal with this by properly setting up the input data for DT.

One simple approach is to approximate the life cycle of an item by using the SKU-store-ranging described in Demand Transference. In this approximation, the range for an item is set to start at *x* percent sell-through of the item and end at *y* percent sell-through. Choose *x* to be 5, and *y* to be 70. *X* must not be 0 (that is, the point of introduction of the item), since it takes sometime after the point of introduction of the item for customers to start buying it in quantity and for the item to start having any kind of cannibalization effect on the other items. **Y** must be set to a point where significant numbers of customers have started to either lose interest in the item or where the item no longer has sufficient numbers of sizes available. In either case, customers are now transferring their demand to the other items in the assortment, so it is as if the item were no longer in the assortment. This is an approximation as the item is still in the assortment and is still selling, just at a significantly lower rate than its peak sales rate.

6

Advanced Clustering

This chapter describes the Advanced Clustering application. It provides details on configuration and implementation.

Overview

Advanced Clustering (AC) lets users create store clusters based on common features such as customer demographics in order to manage merchandise assortments and pricing strategies in a targeted way. Clusters can help retailers understand who shops in their stores and what their preferences are. Clusters can be used to inform decisions about assortment, pricing, promotion, forecasting, allocation, and supply chain processes based on selling patterns in stores. An understanding of the characteristics of the customers who shop in a store and what they buy can help a retailer target specific customers

The application optimizes clusters in order to determine the minimum number of clusters that best describes the historical data used in the analysis and that best meets the business objectives defined when the clusters are designed. Users can define a hierarchy cluster of stores based on a store attribute such as format and then cluster further using performance attributes in order to determine which stores have high, medium, and low sales. What-if scenarios and ranking can be used to compare how cohesive and well separated clusters are in each scenario as the number of cluster centers is increased. The application uses scoring to indicate which clusters fall below defined thresholds and may require manual intervention. Business Intelligence graphics illustrate the patterns in the data and the attributes that are important in each cluster.

The key features available in AC are:

- Dynamic nested clustering, in which a user can cluster on a criteria, analyze the results, and then decide whether or not to further sub-cluster.
- Mixed attribute clustering. A cluster can be created on continuous attributes such as performance (sales, revenue, and gross profit) as well as discrete attributes such as store size, demographics, and seasonality, all at the same time.
- Configurable clustering criteria such as customer profiles, product attributes, performance, and store attributes.
- Recommendations are made for the optimal number of clusters and the scores for each cluster. These are based on the quality of the clusters: how cohesive and well separated the clusters are.

Data Requirements

Advanced Clustering relies on following data and it uses ETL to load the data.

Table 6–1 Data Requirements

Objects	Granularity	Required/Optional
Hierarchies	Product, Location, and Fiscal	Required
Location Attribute	Store	Required
Product Attribute	SKU	Required
Aggregate Sales Data	Week/SKU/Store	Required
Customer Segment Profiles	Store/Customer Segment/(Category or All Merchandise)	Optional
Alternate Hierarchy	CM Group or Trade Area	Optional
Like Locations	Store	Optional
Product Attribute Group and Value	Category or Subcategory	Optional
Aggregate Forecast Sales Data	Week/SKU/Store	Optional

Multiple Hierarchies and Level Support

Store clusters can be generated for the following combinations of hierarchies.

Product Hierarchies

This includes:

- Core merchandise hierarchy
- Alternate hierarchy

Clusters can be defined for either of the hierarchies and for different hierarchy levels. For example, clusters can be defined for Chain, Department, or Category along the product hierarchy.

Note that store clustering can only be defined for a product hierarchy level higher than Item. Item level store clustering is not supported. However, store clusters can be generated for item groups by defining an item group as a level in the alternate hierarchy.

Location Hierarchies

This includes:

- Core location hierarchy
- Alternate hierarchy (optional)

Clusters can be defined for either of the hierarchies and for different hierarchy levels. For example, clusters can be defined for Chain, Trade Area, or Region along the location hierarchy. Store clusters can be generated for channel, if channels are configured as a level in the location hierarchy.

Calendar Hierarchies

This includes:

- Core fiscal calendar hierarchy (week, month/period, quarter, half, year)
- Gregorian calendar (week, month, quarter, half, year). Leverages a start and stop date (day level date range)

- Planning period. Leverages alternate hierarchies, including planning period, buy periods, and defined holiday time periods such as back to school and fourth of July. This is optional.

Clusters can be defined for any of these three calendar hierarchies (the cluster effective period). Note that the source time period for historical data only uses the core fiscal calendar hierarchy, supporting hierarchy levels that include week, month/period, quarter, half, and year.

Clustering Criteria Supported in Store Clustering

In store clustering, the cluster criteria are a set of attributes that define store clusters. These attributes can be either discrete or continuous. A group of these clusters is called "Cluster by." For example, demographic data such as income and store properties such as store formal can be grouped into a store attribute Cluster by.

These default Cluster by are supported.

Customer Profile

Stores are clustered based on the similarity in the mix of customer profiles shopping in the stores and trading areas. These clusters form the basis for further analysis to understand which customers shop in which stores and how they shop. Retailers obtain market data from market research firms such as the Nielsen Corporation and use the data to create customer profiles for their stores. An application feed can be used to provide this information to AC at the category or all merchandise level.

Location Attribute

Stores are clustered based on how shopping behavior varies by store attribute. This provides information about who is shopping in a store or trading area as well as demographic data such as ethnicity, income levels, education, household size, and family status. Retailers can analyze the cluster composition and related business intelligence in order to better understand the shopping behavior of their customers. This can help retailers make assortment and pricing decisions.

Product Attributes

In this type of clustering, stores are grouped together that have similar sales shares for one or more product attributes (for example, coffee brands such as premium, standard, and niche). The percentage of each store’s contribution is calculated using the Sales Retail $ for each product attribute value to the total retail sales of the category or subcategory of each location.

Product attributes can be configured only at category or sub-category levels.

Performance Criteria

Stores are clustered based on historic sales by considering performance at different merchandise levels while performing store clustering and analyzing how the shopping behavior varies by category. This can help to identify high, medium, and low volume stores.

Forecast Criteria

Stores are clustered based on forecast sales by considering future sales at different merchandise levels while performing store clustering and analyzing how the shopping behavior varies by category. This can help to identify high, medium, and low volume stores based on the predicted sales.

Mixed Criteria

Discrete and continuous attributes are combined together. Retailers can cluster stores using attributes from all the above defined criteria at the same time.

Attributes in Store Clustering

Cluster by uses a collection of attributes, including consumer profile attributes, sales metric attributes, location attributes, and product attributes.

Sales Metrics

Store clustering uses a fixed set of sales metrics. These attributes cannot be extended. Supported attributes include Sales Retail $, Sales Unit, Sales AUR, Gross Margin R, and Gross Margin %.

Forecast Sales Metrics

Store clustering uses a fixed set of predicted sales metrics. These attributes cannot be extended. Supported attributes include Forecast Sales Retail $, Forecast Sales Unit, Forecast Sales AUR, Forecast Gross Margin R, and Forecast Gross Margin %.

Location Attributes

Store clustering relies on location attributes that can loaded into the application as either core attributes or as user-defined attributes. These attributes are defined for stores. They define the store properties, including demographic and geographic details. During installation, only attributes that have 15 distinct values are configured. Attributes with higher discrete values are not considered for store clustering.

Product Attributes

Product attributes based on store clustering use two types of attributes, raw attributes and grouped attributes. The former are product attributes, identified as important attribute values. The latter are fed into the application and are available to the store clustering process only when the CDT application is enabled. The clustering process groups together stores that have similar attributes values for a product category. A store share is calculated using Sales Retail $, which is the ratio of the sales retail of each product attribute value to the total sales retail of the category or subcategory of the specific location.

Each category or subcategory must have raw attributes and grouped attributes.

Grouped attributes (the default) classify the items in the class or subclass. This set of attributes differs from class to class. For example, for yogurt, the attributes are: size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes are: size, brand, milk/dark, nut type, and package type. The two classes can both have the attribute of brand, but the brand attribute will have different values for each of the categories. Group attributes have a mapping of each item in the category to its set of attribute values. This information is provided as a data feed to the application. If grouped attributes are not available then raw attributes are used for the store clustering of product attributes.

Raw attributes typically have a large number of attribute values. For example, the brand attribute for yogurt may list 50 different brands at a large grocer. Using raw attributes, the system runs a preprocess to identify *n* (default = 3) attributes values that are most frequently sold for each attribute in a category or subcategory.

Configuration Process

Default configuration occurs during the installation and upgrade. The configuration process is responsible for installing or upgrading any new attributes in the application. This process ensures that any existing manual overrides introduced by the retailers are not overridden and any new additions are brought into the clustering process. The default configuration includes the following:

- All attributes are enabled by default and weights are normalized among all the configured attributes.
- Any discrete location attribute that has more than *n*=15 attributes values is not configured by default. Note that the value of *n* is a configuration and can be modified at the time of deployment.
- The UI formatting of each attribute is identified based on the data type of the attributes.
- Nesting is enabled by default for all types of Cluster by (except mixed, which is an alternative approach to clustering).
- The deployment of clusters at multiple hierarchies or levels is enabled.

The following configurations may require manual overrides if the default configuration is not acceptable or data is not available.

Table 6–2 Manual Overrides

Name	Description
Enable or disable Cluster by	Disable Cluster by. For example, for the consumer profiles for each store for a retailer, the Cluster by for the consumer segment must be turned off.
Enable or disable nesting	Allow multiple nesting levels under an existing Cluster by. For example, cluster first by product performance, with nested clustering by store attributes.
Enable or disable attributes	Enable an attribute to be considered for clustering or contextual BI. For example, label the population density attribute as a BI attribute instead of a clustering attribute, as very few stores have data for population density, and it is not significant enough for store clustering.
Change UI formatting	Change formatting associated with the attributes, such as label, decimals, percent, and currency. These are configurations for each attribute and do not rely on XLIF entries.
Cluster deployment	Enable or disable hierarchy at which clusters can be deployed. For example, if CMPO store clusters are only defined for categories, then only clusters at the category level will be approved. The other levels can be enabled if needed.
Outlier Rule	Change default outlier rules for a Cluster by. By default, the distance from centroid rule is enabled. See section below for other supported outlier rules.
New Store Rule	Change default store rules for a Cluster by. By default, the like location rule is enabled. See section below for other supported new store rules.

Table 6–3 Enable or Disable Cluster By

Cluster By	Description	Example	Enable
Customer segment profile	Cluster store using consumer segment distribution	20% soccer mom, 30% empty nesters	Enable if consumer segment profiles are available for each store.
Store attribute	Cluster store using location attributes	Income, climate, size, store format	Enable if location attributes are available for each store
Performance	Cluster store using sales metrics	Sales revenue, sales unit, gross margin $	Enable if retailer wants to cluster store using performance metrics

Table 6–3 (Cont.) Enable or Disable Cluster By

Cluster By	Description	Example	Enable
Product attribute	Cluster store using product attribute sales shares	Brand, color, seasonality, size/fit	Enable if retailer wants to cluster store using product attributes sales share
Mixed attributes	Cluster store using mixed attributes, combining attributes across all the cluster criteria	Income, climate, size, store format, sales revenue, sales unit, gross margin	Enable if retailer wants to cluster store using combination of attributes
Product forecast	Cluster store using predicted sales metric	Forecast sales revenue $, forecast sales unit, forecast gross margin $	Enable if retailer wants to cluster store using future sales metrics

Copy Clusters Using Like Product Mapping

Advanced Store Clustering supports copying clusters from other products by defining like product mapping. Some of a retailer's products, such as tea, may not be mature enough, so the retailer must copy clusters from a mature category such as coffee. The solution copies clusters that have already been approved from that category to a new category and then make the clusters available to the execution solution during export. This feature also supports the requirement in which categories in certain departments may want to borrow clusters from a higher product hierarchy such as Department or Division. In both use cases, clusters must be borrowed from another product (department or category) with minimal effort by the end user to define and approve the cluster criteria.

The retailer can define a like product interface that the retailer can use to provide mapping between a mature product and a new or poorly performing product. This product mapping can vary by location (for example, the tea category mapped to coffee in Trade Area East, and mapped to other beverages in Trade Area West).

This category mapping is selected automatically during export, based on the like products, and is assigned clusters from the mapped category that have already been approved by the end user. Cluster copy will respect product mapping if it is updated or removed using the effective period, and exports will be generated accordingly. Retailers can review the cluster composition of the mapped category via the UI from which it is borrowing the clusters; however, the copied clusters for the category will not be available for the user to review in the UI.

The Copy Clusters feature using like product mapping is only supported for the basic Store Clustering interface that is available for Oracle Retail Predictive Application Server applications such as Category Management; it is not be supported as part of the Store Clustering generic interface or the REST API.

Multiple Clustering Approach

Store clustering supports three types of clustering: simple, nested, and mixed. By default, all three approaches are enabled. These approaches are applicable to all Cluster by. Store clustering functionality supports dynamic nesting capabilities. For example, the user can cluster on a criteria, analyze the results, and then decide to further sub-cluster.

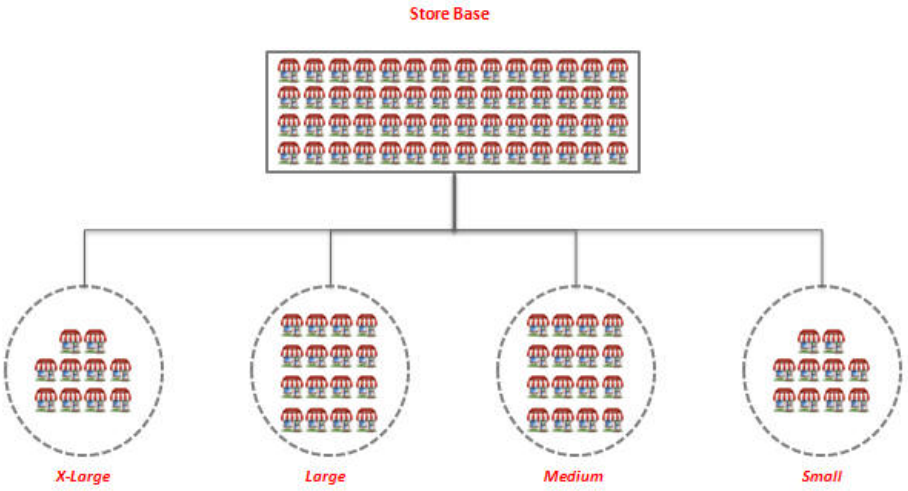
Mixed attribute clustering is also supported. For example, it is possible to cluster on continuous attributes such as performance (sales, revenue, and gross profit) as well as discrete attributes (store size and demographics) at the same time.

Simple

Users can select attributes from a Cluster by. For example, users can select location attribute Cluster by and generate clusters using location attributes such as store size.

Users select the most important store attribute (based on category) and group the stores accordingly. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibility of the planning team.

Figure 6–1 Simple Clustering

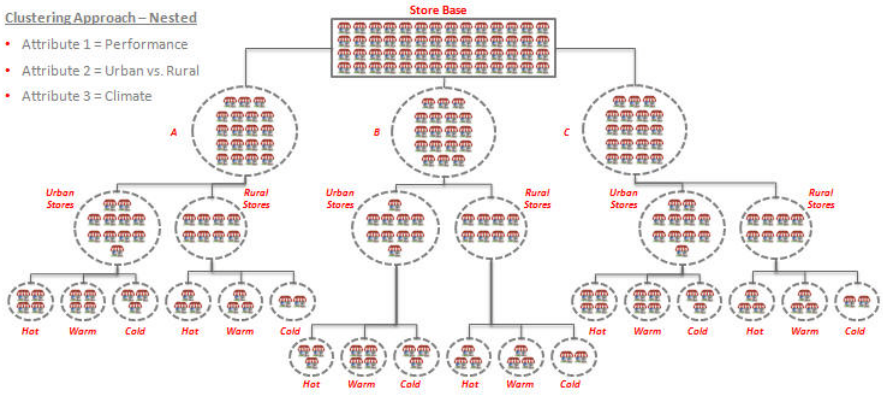


Nested

All Cluster by, except Mixed Attributes, can use a nested hierarchy by default. For example, performance clusters can be further clustered using location attributes. This approach allows a dynamic hierarchy of clusters. Nesting can be enabled or disabled in the AC configurations.

Users can select the most important store attributes (based on the category and the group of stores selected during the wizard process) and group the stores accordingly to ensure that a more refined assortment can be created by category or location selection. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibilities of the planning team. Once the initial clusters are created, users can further cluster using attributes to define nested clusters. This can be done within a trade area as well as at the total company level. The number of clusters is granular, as compared to mixed attributes.

Figure 6–2 Nested Attribute Clustering



Mixed

Mixed attributes, including Cluster By such as consumer segment, location attributes, performance, and product attributes, are supported by default. Users can combine attributes from different Cluster by. For example, users can combine attributes from consumer segment and performance Cluster by and generate clusters using sales revenue and consumer segment distributions.

Users can use the most important store attributes (based on the category and the group of stores selected during the wizard process) and group the stores accordingly. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibilities of the planning team. These clusters are the final ones used in assortment process. The number of clusters in this case is confined, compared to the nested clustering approach.

Figure 6–3 Mixed Attribute Clustering

New Stores or Stores with Poor History

Store clustering supports three rules for allocating new stores to a cluster. These rules can be configured for each Cluster by. A rule is applied after the clusters are generated.

These rules require a feed into the application that defines a mapping between a location and like locations. This mapping can be configured by merchandise. One location can be mapped to multiple locations with different weights.

Like Stores (Default Rule)

Stores with new history or poor history are allocated to the same cluster in which the like locations are allocated. For example, a new store or a store whose poor history has been corrected can be allocated to a valid performance cluster.

Largest Clusters

New stores or stores with a poor history can be allocated to the largest cluster identified by the clustering analytics. For example, a new store that has not yet formed a customer base can be allocated to a larger cluster until significant customer profiles have been collected.

Cohesive Clusters

New stores or stores with poor history are allocated to the most compact cluster identified by the clustering analytics. For example, stores can be assigned to a cluster that has not been not affected by outliers.

Outliers

Store clustering supports two rules that identify stores as outliers in a cluster. These rules can be configured for each Cluster by. The rule is applied after the clusters are generated.

Distance from Centroid (Default Rule)

The distance from a store to the centroid is identified. If the distance is beyond a defined limit of the configured threshold from the centroid, then the cluster is identified as having outliers. The user must investigate such clusters.

Cluster Size

The percentage of stores that are allocated to certain clusters is identified. If they fall beyond a certain limit in comparison to total number of stores, the cluster is identified as having outliers. The user must investigate such clusters.

Export to Excel

To use any reporting tool with an Excel file exported from Advanced Clustering, you may need to adjust the format. Here are some examples of possible formatting adjustments.

- The Text column should remain as is.
- If the Percentage column uses a percent symbol, then that symbol must be removed.
- If the Currency column uses either commas or currency symbols, then those must be removed.
- If the Number column uses commas, then they must be removed.

7

Customer Segmentation

This chapter provides details about the use of the Customer Segmentation application.

Overview

Customer Segmentation (CS) lets users create segments of customers based on common attributes, such as customer demographics, in order to help a retailer manage merchandise and sales strategies in a targeted way. Segments can help retailers understand the types of customers who shop in their stores and gain insight into their typical shopping patterns. This understanding can help retailers target specific customers.

The application optimizes segments in order to determine the minimum number of segments that best describes the data used in the analysis and that best meets the business objectives defined when the segments are designed. What-if scenarios and ranking can be used to compare how cohesive and well separated the segments are in each scenario as the number of segments is increased. The application uses scoring to indicate which segments fall below defined thresholds and may require manual intervention. Business Intelligence graphics illustrate the patterns in the data and the attributes that are important in each segment.

The key features available in CS are:

- Recommendations are provided for the important attributes to use in creating segments.
- Segments can be created on continuous attributes such as sales performance as well as discrete attributes such as customer gender.
- Configurable segmenting criteria such as demographics and RFM (Recency and Frequency Measures) are provided.
- Recommendations are made for the optimal number of segments and the scores for each segment. These are based on the quality of the segments: how cohesive and well separated the segments are.

Data Requirements

Customer Segmentation relies on following data, and it uses ETL to load the data.

Table 7–1 Data Requirements

Object	Granularity	Required/Optional
Hierarchy	Product, Location, and Fiscal	Required
Customer Attribute	Demographic or User Defined	Required for demographic segmentation

Table 7–1 (Cont.) Data Requirements

Object	Granularity	Required/Optional
Sales Transaction	Customer-identified, Product, Date, Transaction ID	Required for RFM segmentation
Alternate Hierarchy	CM Group or Trade Area	Optional

Multiple Hierarchies and Support

Customer segments can be generated for the following combinations of hierarchies.

Product Hierarchy

One of the following hierarchies can be used:

- Core merchandise hierarchy
- Alternate hierarchy

The level at which segments should be created can be configured. The user interface is used to create the specific node identified. This allows the creation of different segments of customers for different geographic regions. For example, different segments can be defined for Canada vs. France.

Location Hierarchy

One of the following hierarchies can be used:

- Core location hierarchy
- Alternate hierarchy (optional)

The level at which segments should be created is configurable. The user interface is then used to create the specific node identified. This allows the creation of different segments of customers for different geographic regions. For example, different segments can be defined for Canada vs. France.

Calendar Hierarchy

This includes:

- Core fiscal calendar hierarchy (week, month/period, quarter, half, year)
- Gregorian calendar (week, month, quarter, half, year). Leverages a start and stop date (day level date range)
- Planning period. Leverages alternate hierarchies, including planning periods, buy periods, and defined holiday time periods such as back to school and Fourth of July. This is optional.

Segments can be defined for any of these three calendar hierarchies (the segment effective period). Note that the source time period for historical data only uses the core fiscal calendar hierarchy. A configuration permits data aggregation at either the fiscal period or fiscal quarter, so the user can select any level that is at that level or above in the user interface.

Supported Segment Criteria

In customer segmentation, the segment criteria consist of a set of attributes that define customer segments. These attributes can be either discrete or continuous. A group of these segments is called "Segment by." For example, demographic data, such as income and gender, and store properties, such as store and formal, can be grouped into a Customer Demographic Segment by.

Here is the list of the default Segment bys that are supported.

Customer Demographics

Customers are segmented based on the similarity in the values of the various customer attributes. Examples include gender, income, educational background, age, and range. Additionally, it possible to use several user-defined numeric or discrete attributes.

RFM and Customer Behavior

Customers are segmented based on attributes that are comprised of aggregate metrics regarding their purchase behavior. Examples include the number of purchases, amount of sales, average basket size, and share of products purchased while being promoted. Retailers can analyze the segment composition and related business intelligence in order to better understand the customer shopping behavior associated with the segments.

Category Purchase Behavior

Customers are segmented in a similar manner to RFM and Customer Behavior, with the difference that these attributes are calculated for a selection of the most important product categories. The category level used here is configurable. It helps enable segmentation by product category purchase behavior.

Customer Segmentation Attributes

Segment by uses a collection of attributes, including demographics, purchase behavior, product purchase behavior, product profiles, and user defined. Each quarter, the batch processing creates new versions for each location at the configured level of the location hierarchy. During this process, attributes are summarized and their data is analyzed for usefulness for segmentation. An attribute can have a different level of usefulness for each of the different versions. For example, if the majority of customers in the Canada location provide a gender attribute, and the majority of the customers in the France location do not provide a gender attribute, then gender can be used in Canada, but not in France.

Furthermore, within the same location, it is possible for an attribute to be considered useful for the most recent quarter, while in the prior quarter it was not useful because there were insufficient values available during that time. The aggregate statistics about the attributes for a version can be seen using the Explore Data screen after selecting the segment criteria. Segments created in the previous quarter have different statistics than those that are created during a different quarter.

Demographics

Demographic segmentation relies on customer attributes that are loaded into RI. The set of attributes used from RI's customer dimension is fixed. If alternative attributes are needed, see User Defined. Once loaded into RI, they can be used by Customer Segmentation. They define details about each customer that can be used for creating customer segments using those values. During the time when a new version is created, only attributes that have 15 or fewer discrete values are used. Attributes with a higher number of discrete values are not considered for customer segmentation.

Purchase Behavior

Customer segmentation uses a fixed set of sales transaction metrics, which are obtained from sales transactions identified by a customer ID. The values include Sales Quantity, Sales Retail $, Gross Margin $, Promotional Sales Quantity, Promotional Sales Retail $, Promotional Gross Margin $, Number of Transactions, Average Transaction Count per configured period, SKU count, Transaction Basket Size, and Promoted Sales Share of Total Sales.

Product Purchase

Customer segmentation uses a selection of the top categories to portray the shopping patterns for each customer for some product categories. Each time a version is created, the categories with the highest amount of sales are picked as the categories for which Product Purchase based assessments are done. These attributes are similar in concept to the Purchase Behavior segment; however, these are specific to the top categories for the location associated with the version. The attributes include Sales Quantity, Sales Retail $, Gross Margin $, Promotional Sales Quantity, Promotional Sales Retail $, Promotional Gross Margin $, Number of Transactions, Average Number of Transactions, SKU Count, Transaction Basket Size, Promoted Sales Share of Total Sales, Average Basket Sales $, and Average Basket Gross Margin $.

Product Profile

Customer Segmentation uses a fixed set of category-based sales profile values. For the same set of top categories described in Product Purchase, the share of the customer's total purchases for the category is calculated. The share of the Promotional Sales Retail $ of the customers total sales is also calculated.

User Defined

For any attribute that is available for the customer, but is not accounted for in the default set of attributes, there are provisions for loading a set of customer or user defined attributes into RI. These attributes can be either numeric values or discrete values. If the attribute value is numeric (such as a zip code), but must be treated as discrete rather than a ranged numeric value, then the attribute must be loaded to an appropriate text attribute column in RI. Any attribute that has more than 15 distinct values will not be used by the segmentation process.

Once an attribute is defined, it is possible to adjust the configuration data in the database to assign a more context-suitable name for the attribute. This enables the user interface to identify the attribute as a specific attribute, and not just as a generic Custom Text Attr or Custom Number Attr.

Configuration Process

Default configuration occurs during the installation and upgrade. The configuration process is responsible for enabling or disabling any attributes in the application. This ensures that the desired attributes are available for use during the segmentation process.

- All attributes are enabled by default.
- Any discrete attribute that has more than n=15 attributes values is not configured by default. Note that the value of n is a configuration and can be modified at the time of deployment.
- The UI formatting of each attribute is identified based on the data type of the attributes and by the name of the attribute.

The following configurations may require manual overrides if the default configuration is not acceptable or data is not available.

Table 7–2 Manual Overrides

Name	Description
Enable or disable Segment by	Disable Segment by. For example, if there are no customer demographic attributes, then the Customer Demographics segment by can disabled.

Table 7–2 (Cont.) Manual Overrides

Name	Description
Enable or disable attributes	Enable an attribute to be considered for segmentation. For example, if there is no intention on loading user defined attributes, they can all be disabled so that when new versions are created, no processing time is spent analyzing those attributes.
Change UI formatting	Change formatting associated with the attributes such as label, decimals, percent, and currency. These are configurations for each attribute, and do not rely on XLIF entries.
Location hierarchy level	If customer segments are desired for each location at a given location level, then the configuration can be adjusted so that all processing is done at that level. This also requires an adjustment of the approval level so that it allows the segments to be approved.
Outlier rule	Change default outlier rules for a Segment by. By default, the distance from centroid rule is enabled. See the section below for other supported outlier rules.

Table 7–3 Enable or Disable Segment By

Segment By	Description	Example	Enable
Customer Demographics	Segment customers using demographic values	Age range, income, gender	Enable if consumer demographic attributes are available
RFM and Customer Behavior	Segment stores using location attributes	Income, climate, size, store format	Enable if location attributes are available for each store
Category Purchase Behavior	Segment stores using sales metrics	Sales revenue, sales unit, gross margin	Enable if retailer wants to segment stores using performance metrics

Attribute Preprocessing

Before customer segments are created, the available customer data must be preprocessed in order to identify the sample of customers to use for segmentation and to determine which attributes are the most beneficial for use while creating segments. A few configurations can be manipulated to help improve this process. The following table defines some configurations that can be adjusted to control how attributes are used by the system. These values can be manipulated in the table CIS_TCRITERIA_ATTR.

Table 7–4 Attribute Configuration

Column	Description
DELETE_FLG	Set value to Y to prevent an attribute from being used by any processing.
SAMPLE_FLG	Set value to Y so that the attribute to be used has a stratified sample of customers. This should help ensure that an appropriate selection of customers are represented in the sample. Up to three attributes can be set to Y. If no attributes are configured with a Y value, then a random sample of customers will be used.
DISPLAY_FORMAT_ID	This can be adjusted to use different display formats, as defined in RSE_DISPLAY_FORMAT.
NUM_BUCKETS	Allows a different number of buckets for use when showing summary metrics for a numeric attribute.

Table 7–4 (Cont.) Attribute Configuration

Column	Description
ATTR_IMP_FLG	When set to a Y value, the importance of this attribute is analyzed for usefulness during segmentation. If an attribute will never be used for segmentation, setting the value to N will exclude it from the attribute importance calculations.

Another configuration that can adjusted is the CIS_BUS_OBJ_TCRITERIA_ATT_XREF.VALIDATING_ATTR_FLG. For any attribute that has a Y value for this column, the attribute is shown in the Insights portion of the UI. If it is determined that an attribute should not be displayed, then this value can be changed to a value N so that it is excluded from display in the Insights results.

Segmenting Approach

Customer segmentation uses Oracle Data Mining for the creation of its segments. The k-means approach that is used results in the creation of segments in a hierarchal manner. The process automatically determines which attribute is the best attribute to split into an additional segment. This process is continued until the desired number of clusters has been achieved.

Customer Segment Store Profile Generation

Customer Segmentation calculates the sales share of customer segments for each store. These store profiles can be generated by the user for the approved customer segments from the user interface. They can then be consumed by RI to generate business reports and Store Clustering to generate customer centric store clusters.

Preprocessing

As part of preprocessing when a version is created, Customer Segmentation first filters the customer and then samples the customer.

Filter Customer

The filtering step allows the implementer to set the following conditions in order to remove outlier customers during different phases of Customer Segmentation,

Table 7–5 Filter Step Conditions

Filter Rule	Operation
Fake Customers	This rule discards customers if the number of transaction per day exceeds the value *x* (the max daily transaction threshold to identify a fake customer - default 10)
Discarding Customers	Keep customers where customer sales fall between min (sales or # of transaction) and max (sales or # of transaction). Minimum/Maximum percentile for amount of sales transactions. (default .001) Minimum/Maximum percentile for number of sales transactions. (default .001)

Table 7–5 (Cont.) Filter Step Conditions

Filter Rule	Operation
New Customer Rules	Include new customers if there is enough existing sales history for the customer. Use configurations that control the min/max values for average transaction count and average transaction amounts used to include new customers: Minimum/Maximum percentile for amount of sales transactions. (default .001) Minimum/Maximum percentile for number of sales transactions. (default .001)
Top Categories	A configuration that can limit the number of categories (*n*) that a user picks. The system picks it based on the default - Sales Revenue of the category to be used to calculate the top categories. Allowable values are limited to SLS_AMT, SLS_QTY, PROFIT_AMT, and SHARE variations of the same (i.e., SLS_AMT_SHARE).

Sample Customer

The sampling step allows the implementer to enable and adjust sampling customers.

Table 7–6 Sample Step Conditions

Sample Rule	Operation
Target Sample	Set the value to *Y* so that the attribute to be used has a stratified sample of customers. This should help ensure that an appropriate selection of customers is represented in the sample. Up to three attributes can be set to *Y*. If no attributes are configured with a *Y* value, then a random sample of customers will be used. This can be adjusted by changing CIS_TYPE_CRTIERIA in Data Management UI.
Sample Size	The sampling percentage for Customer Segmentation. This can be adjusted by changing RSE_CONFIG in the Data Management UI.

Customer Metrics

The Retail Science process supports the following derived customer metrics, which can be used for analysis by an application such as Customer Engagement. These metrics are calculated using customer-linked transaction data. In addition to helping understand how customers have behaved in past, these metrics can also help predict future behavior.

Table 7–7 Customer Metrics

Metric	Description
Customer Latency/Time Between Visits	The number of days between each of a customer's transactions sales or return.
Customer Lifespan	The time between a customer's first and last purchase.
RFMThe	RFM (recency, frequency, monetary) score determines quantitatively which customers are the best ones by examining how recently a customer has purchased (recency), how often the customer purchases (frequency), and how much the customer spends (monetary).
Projected next purchase date	Prediction of the next likely customer purchase date.
Location Loyalty	How loyal are customers to a specific location? A value of 100% indicates that they always shop at a particular location.
Style Loyalty	How loyal are customers to a particular style? A value of 100% indicates that they always prefer one specific style.

Table 7–7 (Cont.) Customer Metrics

Metric	Description
Color Loyalty	How loyal are customers to a particular color? A value of 100% indicates that they always prefer one specific color.
Brand Loyalty	How loyal are customers to a particular brand? A value of 100% indicates that they always prefer one specific brand.
Price Efficiency Loyalty	How efficient are customers in getting a promotion price? A value of 100% indicates that the customer always buys items on promotions or is very efficient in obtaining a good price.

8

Affinity Analysis

This chapter describes the Affinity Analysis application.

Overview

Affinity Analysis (AA) is used to gain insights into customer shopping patterns. A key component of AA is the process of Association Rule Mining (ARM). This process examines sales transaction data and identifies associations between types of products. Such information can help a retailer understand that promoting one product is sufficient to help drive sales of another product, given the sales associations they exhibit.

The processing of these algorithms occurs each week as part of the weekly batch execution, and a set of output files are provided to expose the association rules that have been discovered by the process.

Data Requirements

AA relies on the following data elements. These must be provided via text files, which are then loaded.

Table 8–1 Data Elements

Object	Notes	Required/Optional
Product Hierarchy	The ARM processing mainly operates at Sub Class, but it can be configured to different levels.	Required
Location Hierarchy	-	Required
Fiscal Calendar	-	Required
Sales Transactions	Must contain transaction IDs as part of the data. If the transactions include Customer ID, then customer segment results are possible.	Required
Customer Segments	Customer IDs and their association to a segment allows customer segment-specific results.	Optional
MBA_ARM_SRVC_LOC_STG	Can be used to limit the scope of locations processed, or to specify a set of locations to exclude from processing.	Optional

In order to calculate association rules, it is necessary to receive sales transaction data that include a transaction ID. This is used to identify which products were purchased by a customer as part of a single transaction. If the customer transactions also include a customer ID to

identify the customer who purchased the transaction, and a customer segment dimension is provided that links customer IDs to customer segments, then it is possible to provide some results for each customer segment.

MBA_ARM_SRVC_LOC_STG

When specifying which locations to process or which locations to not process, the MBA_ ARM_SRVC_LOC_STG interface can be used to limit the scope of locations to be processed. The data in this interface can be at any level of the location hierarchy. A customer may want to limit the scope of locations for the following reasons.

- Improve performance by only sampling some locations.
- Exclude locations that contain many wholesale transactions, where the transactions contain data for more than a single customer.
- Exclude locations that are experiencing a significant interruption to their normal sales pattern (for example, when undergoing a large scale renovation).
- Exclude locations that normally do not include customer-linked transactions from the ARM_PH_CS implementation, since suitable data to include for processing will not be available.

The SRVC_NAME column of this interface allows the specification of the service that must be filtered. If, however, all executions must have the same set of locations, then this column can be provided as a NULL value. The effect will be to use the same dataset for all the services. If, however, it is necessary to have some services use a different set of locations, then it is possible to provide the data specific to the different services. If data is provided for a SRVC_NAME, then the data must be provided with a SRVC_NAME specified. The valid SRVC_NAME values that can be provided are: ARM_PH (Product Hierarchy results), ARM_PH_PROMO (Product Hierarchy with Promotions results), and ARM_PH_CS (Product Hierarchy and Customer Segment, with Promotions results).

MBA_ARM_SRVC_CONFIG_STG

It is possible to provide a configuration file that will be used as the configuration for all automated batch runs, as well as the default values for any user-created runs. The expected process flow for this interface is that a user initially uses the UI to create user runs for the various types of processing that is supported. These runs must be for the same configuration levels that the application is configured to use. The user creates runs for some sample weeks, and gradually adjusts the various configuration points, until a run is created with an acceptable amount of output.

Once this has been done for each of the different types of calculations, then the configuration values that were used can then be provided via this interface so that all automated runs will use these settings.

If the system is configured to run the calculation at a level of the location hierarchy other than the entire company, such as chains instead, and it is necessary to have different configuration values for each of those chains, then a different set of values can be provided here for each location node.

The SRVC_NAMEs used in this interface are as follows:

- ARM_PH - Used for any Dept, Class, or Subclass calculations where promotions are not involved.
- ARM_PH_PROMO - Used for any calculations that focus on the involvement of a promotion.
- ARM_PH_CS - Used for any calculations that process results for a customer segment.

- ARM_ITEM - Used by user-created runs where the result level is at the Item level. This is not used by batch automation, as Item results are not calculated automatically.
- If there is a need for different set of configuration values for each level of the hierarchy, the HIER_LEVEL_NAME can be provided with the appropriate value to indicate what the configurations are for (SBC, CLS, DEPT).

Science Algorithms/Services

This section describes the science algorithms and services.

ARM_PH

This implementation calculates association rules for a configurable set of product hierarchy levels. It supports the creation of association rules for Sub Classes, Classes, and Department, which can be controlled by a system configuration. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table, and are named with "ARM_PH_" as the prefix. Because this implementation supports being run for multiple hierarchy levels, if there is a need to set a configuration uniquely for a specific hierarchy level, this can be accomplished via the RSE_CONFIG_CODE table using the hierarchy level name as the PARAM_CODE value. If no such row exists in RSE_CONFIG_CODE, then the configuration will be taken from the corresponding RSE_CONFIG row.

ARM_PH_PROMO

This implementation calculates association rules at the Sub Class level of the hierarchy and is restricted to only rules where the IF side of the rule is promoted and the THEN side of the rule is not promoted. In order to be able to execute this and have results for this implementation, it is necessary to provide promotion details with the sales transaction data. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table and are named with "ARM_PROMO_" as the prefix.

This implementation is used to focus on how products are associated when that promotion is in effect. This data can help a retailer understand the sales patterns that exist when promotions are involved, which can help the retailer avoid promoting too many items in an effort to help improve profit.

ARM_PH_CS

This implementation calculates association rules at the Sub Class level of the hierarchy and is restricted to rules where the IF side of the rule is promoted and the THEN side of the rule is not promoted. In order to be able to execute this and have results for this implementation, it is necessary to provide promotion details with the sales transaction data. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table and are named with "ARM_CS_" as a prefix.

This implementation provides the same type of information as the ARM_PH_PROMO implementation; however, it provides results that are specific to a customer segment. Therefore, this implementation requires the receipt of transactions that include the customer ID of the customer who purchased the transaction and the customer segment dimension, along with the association of the customers to each customer segment.

Configurations

There is a consistent pattern in the naming of the configurations for the AA implementation. As described above, each implementation has a specific naming prefix. The suffixes are also similar across the implementations. These suffixes are described in more detail in Table 8–2.

Table 8–2 Implementation Suffixes

Suffix	Example	Description
HIER_LEVEL	SBC	Indicates the name of the hierarchy level that the process is to be executed for. The values here are the same values as provided as LEVEL_NAME values in the W_PROD_CAT_DH interface. Not applicable to ARM_PH.
TOP_LEVEL	SBC	Indicates the highest level of the product hierarchy for which processing should be executed. Can contain SBC, CLS, or DEPT. Only applicable to ARM_PH.
MIN_SUPPORT	.001	Expresses the minimum percentage of transactions that are required to have the set of items in the same transaction.
MIN_SUPPORT_TXN_CNT	1000	In the event that sales volume is low, this is another way to express the minimum number of sales transactions that are required for the set of items to be sold together. The implementation uses the greater of the two values.
MIN_CONFIDENCE	.05	The minimum confidence value as calculated by the rule mining algorithm for an association rule.
MIN_REV_CONFIDENCE	.05	The minimum confidence as calculated by reversing the placement of the numbers in the calculation. Setting this value higher can help prevent redundancy in the rule expressions where the IF and THEN items are transposed.
MIN_LIFT	.05	The minimum lift as calculated by the rule mining algorithm for an association rule.
MAX_LIFT	100	The maximum lift as calculated by the rule mining algorithm for an association rule.
MAX_SET_SIZE	2	The maximum number of hierarchy members to include in the resulting rules. The set size includes the count of both the IF and THEN components. The maximum allowed is four, although it can be an expensive to calculate that many components.
MAX_RULE_COUNT	9999	The maximum number of rules that are retained per execution of the algorithm, per set size. This allows for the reduction of results to eliminate less important results.
WEEK_CNT	1	The number of weeks that are processed when the execution runs. Care should be taken when changing this to more than one week, as this can negatively affect performance.

Data Output

The results of the association rule mining can be obtained from two export interface files. One export interface contains summary information (mba_arm_run_exp) about the execution, along with various metrics that explain what the results are for. The second export interface file (mba_arm_result_exp) contains the details for each execution of the process. It is possible that a run may not contain any results to be exported. The data between the two interfaces can be joined to each other by the first column in each interface file (the RUN_ID).

In addition to metrics that quantify the rule (its frequency, its confidence, and its lift), the results also include sales values for the different components of the association rule. These sales values can help quantity the involved sales volume that is involved in the association rule.

Even if the weekly process that runs is executed for a single week each time, it is still possible to estimate the effects of the rule across multiple weeks by aggregating data across the weeks. The process for doing this requires locating the same product set across the different weeks within the same execution type. This means to join data in the mba_arm_result_exp interface by if_prod_ext_key1, if_prod_ext_key2, if_prod_ext_key3, if_promo_flg1, if_promo_flg2, if_promo_flg3, then_prod_ext_key, then_promo_flg, and the data in the mba_arm_run_exp interface by run_type, if_hier_level, then_hier_level, loc_ext_key, custseg_ext_key. The data between the mba_arm_run_exp and mba_arm_result_exp files are joined by the run_id.

Once this appropriate data has been gathered, the various sales metrics can be aggregated as needed. In order to calculate a new set of Frequency, Confidence, Lift, or Reverse Confidence values for a rule, it is possible to recalculate the values, as shown below. Note that in these calculations, the following abbreviations are used: run = mba_arm_run_ext, result = mba_arm_result_exp.

Frequency = SUM(result.rule_txn_count) / SUM(run. tot_txn_cnt)

Confidence = SUM(result.rule_txn_cnt) / SUM(result.if_tot_txn_count)

Reverse Confidence = SUM(result.rule_txn_cnt) / SUM(result.then_tot_txn_count)

Lift = SUM(result.rule_txn_cnt) * SUM(run.tot_txn_cnt) / SUM(result.if_tot_txn_count) / SUM(result.then_tot_txn_count

9

Assortment and Space Optimization

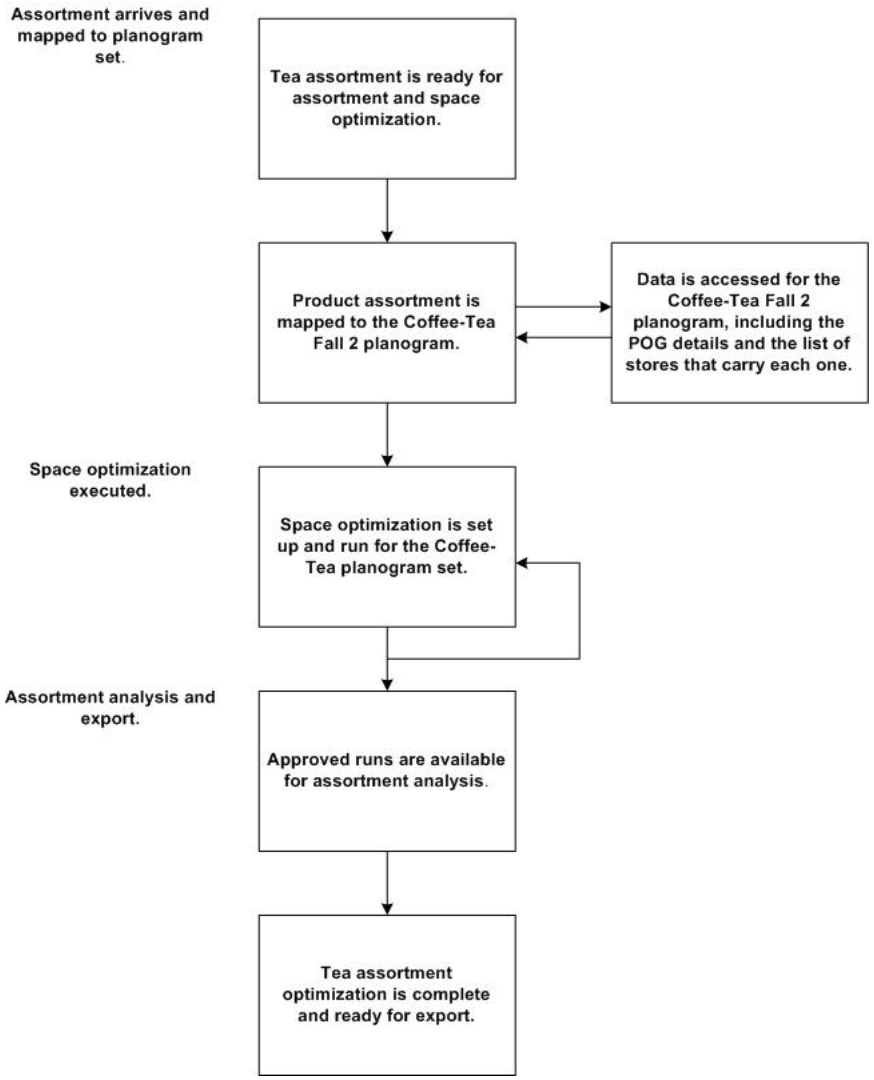
This chapter describes the Assortment and Space Optimization application.

Overview

Assortment and Space Optimization (ASO) is used to determine the optimal selection and arrangement of products within stores by optimizing the product assortment and product placement on a planogram. ASO uses information about available space in stores, product dimensions, fixture configurations, expected demand and demand transference, replenishment schedules, target service levels, merchandising rules, visual guidelines, and category goals to create VPOGs that optimize total performance. In short, ASO generates space-informed optimal product assortments in the form of a virtual planogram. (A VPOG is ASO's assortment recommendations in product facings, depth, stacking, and SKU order in form of a planogram.)

The products that are selected for the VPOGs are the ones that ASO recommends for the finalized assortment. The recommended product level data is used inside CMPO, while the virtual planograms created in ASO are available for use in space planning applications. Figure 9–1 shows the flow of an assortment through ASO.

Figure 9–1 ASO Assortment Flow



ASO supports store level and space cluster level optimizations runs. Space clusters are ad-hoc groups of stores used for optimization. Each space cluster includes stores that are in the same assortment, have the same product list, and have the same current planogram length (or same planogram length, depth, height, and fixture type, if additional planogram attributes are selected). Space clusters are a level between assortment cluster and store. ASO creates space clusters by partitioning stores from an assortment cluster into smaller groups of stores that have the same product list and planogram attributes. In a store level optimization run, each store is optimized independently using the store's specific data to generate one optimal planogram for each store. On the other hand, in a space cluster-level optimization run, every space cluster is optimized independently, using aggregated store data to produce one optimal planogram for each space cluster.

ASO supports shelves, pegboards, freezer chests, and a combination of shelf and peg-board planogram fixture types. The smart start process in ASO creates fixture details using a combination of default values and user selection and assigns shelves to the base fixture of the fixture profile.

At a high level, ASO starts with an assortment that is ready for optimization. The assortment is mapped to one or more planograms and one or more optimization runs. Approved runs are then available for assortment analysis and can then be finalized and exported.

Figure 9–2 shows an overview of ASO workflow, which is described here:

- Receive Assortment: ASO receives a preliminary assortment list from CMPO that is to be space optimized. The assortment is mapped to a set of available planograms through assortment-to-planogram mapping files.
- Set up Optimization: Allows users to specify the location level for the optimization, select planogram and optimization locations, select or update available fixture configurations, view or modify product merchandising options, and demand and replenishment data. Note that the majority of this information is pre-loaded into the solution.

Figure 9–2 ASO Workflow Overview

- Assortment and Merchandising Rules: Allows users to specify visual blocking and sorting rules, product and planogram constraints, assortment rules like the list of products that must be kept together, pick at least three from a list of five products, and so on. Note that mandatory items from CMPO are automatically available here.
- Analyze Optimized Space and Assortment: Allows users to view and analyze optimized assortment and associated facings recommendations as well as a visualization of the optimized POG. The user can also review KPIs such as optimized service levels, expected sales, and profit.
- Finalize Cluster and Store Virtual POG: Allows users to interact with VPOG and override recommendations before finalizing and approving optimized assortments.

ASO Data Input Requirements

This section provides information about setting up the data that the ASO application uses to generate optimal product assortment and placement on virtual planograms. See *Oracle Retail AI Foundation Cloud Services Implementation Guide* for detailed file formats and definitions.

Assortment Data

ASO requires product assortment data that contains information about assortments, assortment clusters and products within an assortment cluster, placeholder product information, like-items for placeholder products, price, cost, and forecast data for products in the assortment.

Note that this section does not discuss the assortment files related to placeholder attributes, finalized assortment placeholder products, or assortment finalization data.

Here is a list of the assortment data files.

- Assortment staging file: This file specifies the assortment header and general information about the assortment, including assortment goal.
- Assortment cluster file: This file provides information on assortment clusters.
- Assortment cluster membership file: This file contains information on stores to assortment cluster assignments.
- Assortment product store/cluster file: This file defines the list of products (including placeholder products) for a given assortment and store/assortment-cluster combination. It also contains the IPI and product priority indices.

- Assortment product location forecast file: This file contains the weekly forecast for a list of products (including placeholder products) for a given assortment and store/assortment-cluster combination.
- Assortment product location price and cost file: This file provides the regular retail price and the cost for the list of products (including placeholder products) for a given assortment and store/assortment-cluster combination.
- Assortment placeholder product like product file: This file contains the list of placeholder products and a like product for each placeholder product in the assortment. ASO uses the like product information available in the file to extract product size data, product attributes, and replenishment data.

Planogram Data

ASO requires planogram (POG) data that is used to define POG dimensions, categories, seasonal information, and product display geometry. CMPO provides ASO with the files it requires. Thus, CMPO customers can export POG data to files that ASO can import readily. Customers who use another POG or space planning solution must provide POG data to meet the ASO interface definitions.

The following POG files are needed for ASO.

- POG definition file: This file defines the major characteristics of a POG, including name, category, status, season, and dimensions.
- POG store file: This file maps a POG to a particular store or a set of stores.
- POG display style file: This file lists the display styles used in certain planograms.
- Product display style orientation file: This file is a cross reference between display styles and orientation. This lists the valid orientations for each display style. It is required that each display style must be mapped to at least one orientation.
- POG bay configuration file: This file provides a list of bays used by the POG.
- Fixture definition file: This file provides a list of the fixtures that define the POG. Fixture attributes specific to pegboards and freezer chests are also defined in this file.
- Fixture configuration file: This file describes the fixture layout in a bay. A fixture can be a shelf, pegboard, or freezer chest.
- Display style compatibility file: The file cross references fixture types and display styles. It lists the fixtures for which the display style is valid.
- Shelf definition file: This file is required for shelf fixture planograms. It provides the details for each individual shelf in the fixture.
- Shelf configuration file: This file describes the shelf layout in a fixture.
- Product display style file: This file specifies product to display style mappings. It provides a list of display styles that are available for a specific product.
- Display style definition file: This file provides the display style product settings and dimensions.
- Shelf product configuration file: This file describes the product layout on the shelf fixture.
- Pegboard/freezer product configuration file: This file describes the product layout on the pegboard/freezer fixture.
- Store custom attributes: This file provides user-defined POG attributes for every store/POG.

It is recommended that, at implementation time, planogram data is imported in bulk rather than on ad-hoc basis. Additional planogram data can be loaded incrementally.

Assortment-to-Planogram Mapping

A partially automated process in ASO attempts to map assortments to planograms. The process matches the seasons associated with planograms and assortments and considers demand spread factors for products in assortments that are assigned to multiple planograms at one time. The user can achieve the desired mappings by creating or editing a pair of mapping files. The assortment-to-planogram mapping files must be created before either of the two components can be used in ASO. Introduction of new assortments or planograms mandates the update and load of new mapping files before using the new assortments or planograms in ASO.

Here is the list of mapping files.

- POG to assortment mapping file: This file contains the POG hierarchy to assortment product mapping information, and it is used to identify which POG should be used for each product in the assortment.
- POG season-to-assortment mapping file: This file contains the POG season-to-assortment date mapping. Once the mapping from product to POG has been performed, a second pass examines this table to identify the specific season for the POG to use based on the assortment start date.

Assortment to POG Mapping Process

The assortment to POG mapping process has two main components. An automated mapping process assigns assortments from APO to POG sets. Afterwards the mapping, the user can access a UI to review and potentially override the mapping results.

Input Data

The following data is input for the mapping process.

- APO assortment, with associated data (SO_ASSORTMENT_STG)
  - Product list from APO assortment, at either the store level or the cluster level (SO_ASSORT_PRODUCT_STRCLTR_STG)
  - Stores within the assortment clusters, including effective assortment dates (SO_ASSORT_CLUSTER_STG and SO_ASSORT_CLUSTER_MEMBER_STG)
- Historical POG data, including POG hierarchy information (that is, POG department, POG category, POG sub-category) and POG seasonal attributes (SO_POG_STG)
  - Store to POG data. List of stores that have each POG, along with the effective dates for the POG at the store level (SO_POG_STORE_STG)
- Mapping tables
  - Product to POG node mapping table (SO_POG_ASSORT_MAPPING_STG)
  - Seasonal attribute mapping table (SO_POG_ASSORT_SEAS_MAPPING_STG)
- Merchandise hierarchy (Core data provided separately)

Automated Process

The automated process consists of the following steps:

1. Find one or more POG set(s) for each product.

- The process iterates through the list of products associated with each cluster or store (depending on the assortment level) and performs a lookup to identify a POG node and demand spread factor for each product.
- ASO searches the mapping tables for the lowest level merchandise hierarchy node that matches that product, starting at the product level, then, if the mapping is not there, looks for the product's parent, the product's grandparent, and so on. The highest level for escalation is the assortment's product category.
- Exception case: If no matching row for the product (or ancestors) is found in the mapping table, the product will be considered unmapped and will be handled manually via the UI. Nothing else is done for that product by the automated process.

2. Find the seasonal attribute.
   - For each location within the assortment cluster and mapped POG node found for the products above, ASO performs a year-independent lookup for the seasonal attribute using the location's start date against the season's provided in the mapping tables. If no match is found, the process will use the current season. Current season is defined as the one that corresponds to the POG with the latest (most recent) effective_start date for any loaded POG within the POG node.
3. Find a POG for each one of the stores in the assortment cluster associated with the product. Stores within the same assortment clusters will very likely have different POGs assigned to them. This step finds those POGs
   - If a season is found, then for every store, the process looks for a specific POG within the POG node and seasonal attribute identified above.
   - If a season is not specified, then the application uses the Current Season to find specific POGs for each store.
   - If a season is specified in the mapping table but there is no such season in the historical POGs, then the process treats this as if the season was not specified and uses the Current Season.
   - Exception case: If none the above techniques find a match for some stores, then the store will be unmapped and that information will be made available in the UI. The user can attempt to fix this by selecting some other POG node/season. Stores that remain unmapped after the POG mapping/manual overrides, will fall out and do not go through SO.

Mapping Errors

The following mapping errors can occur:

- Demand Spread Factor (DSF) out of range (1-100). The product is mapped to more than one POG set and the total DSF across these POG sets for the product is less than 1 or greater than 100.
- Unmapped Store. After matching the products from the assortment and looking for POGs within the identified POG sets, none of the stores in the cluster have a matching POG. This means that the assortment was mapped to a POG set for a product for which ASO cannot optimize any store.
- Unmapped Product. This is raised because mapping data either does not exist for the product because it was not provided or it was provided but the POG Set does not exist.

Product Images Data

Business users can view images that are provided on a customer-hosted web server on ASO virtual planograms. The W_RTL_PRODUCT_IMAGE_DS.dat interface contains a column

called PRODUCT_IMAGE_ADDR that can contain the full URL to an image of the product. This URL must be in the following format:

http[s]://servername[:port]/location/filename.extension

For example:

PRODUCT_IMAGE_NAME = imagename.png

PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png

PRODUCT_IMAGE_DESC= Short description of the image

The ASO application running in the cloud does not directly access these images, so there is no need to expose these images outside of the customer's firewall. As long as the user of the ASO application has access to the URL while running the ASO application, then the user's web browser will be able to resolve the URL and retrieve the images for display when they choose this option. The images must be in a file format that the web browser can display. Since the images shown in the UI are small, these images do not need to be high quality images. The size of the image files will affect the time it takes to render the planogram with the images.

Replenishment Data

ASO consumes replenishment parameters at the product/location level and simulates a (s, S) inventory model to estimate service levels as a function of number of facings, which is one of the key inputs to the optimization engine. Lost sales are estimated as a part of service level calculations.

In addition to mapping and replenishment feeds, ASO also had another feed related to the Assortment and Space Optimization Product Stack Height Limit file. For details, see *Oracle Retail AI Foundation Cloud Services Implementation Guide*.

Optimization Science

This section describes the science behind the optimization algorithms used in the ASO application. It does not provide all the details of the algorithm. It does provide some guidance so that the user can troubleshoot and resolve issues quickly during an implementation.

Optimization Algorithm Overview

The optimization algorithm analyzes trade-offs and selects the best set of items from a given assortment. It then informs the user where to place these items, using how many facings, in a given planogram. The algorithm uses sophisticated mathematical modeling techniques to analyze all possible solutions to generate the best possible planogram. The optimization algorithm is provided an objective (for example, maximize total profits over all items in the assortment), business rules or restrictions, and the sales for a particular item *i* with *k* number of facings and at a particular elevation in the planogram (see Sales and Inventory Model). The algorithm analyzes the trade-offs between all possible solutions and picks the solution that gives the best value for the objective. All the restrictions imposed by the user are treated as required; that is, all possible solutions must satisfy that particular criterion.

To provide a sense of the analysis, here are some of the trade-offs that are analyzed.

- Is there sufficient space to pick all items and assign the maximum number of facings? Or instead, because space is limited, which items must be dropped?
- Should a particular item be included in the ultimate solution? Is there sufficient space? Or, since space is limited, does giving space to this item result in the dropping of a more profitable item?

- Does picking an item cause a conflict with the imposed business rules? For example, a user might determine that the item must be placed at eye-level, but there are other items that are equally profitable that do not have any elevation restrictions.
- Does dropping an item have a significant impact on the demand lost and thus revenues? Or instead, does dropping an item causes no significant loss in demand since some of the demand is transferred to other items that are selected?
- For an item that is picked, is it profitable to add another facing? Or instead, is it profitable to select another item? Does adding another facing for an item cause the item to be dropped since it cannot be fit anywhere in the planogram?
- Does adding another facing help in meeting the service level requirement? Or instead, can the minimum service level requirement ever be met and that item must be dropped?

Sales and Inventory Model

At a high level, this model generates sales for a particular item *i* with *k* number of facings and at a particular elevation in the planogram. This model consumes the replenishment parameters and historical sales and computes the expected sales for a particular item *i* with *k* number of facings and at a particular elevation in the planogram.

The optimization that ASO performs makes decisions about which products to put on the shelves and how many facings to give each product. The basis for these decisions is demand information for each product and how much of the demand can be satisfied with a given amount of inventory for the item. Items with high demand may need more shelf space to hold the necessary inventory, and the ASO optimization balances that against other products that may have lower demand and also lower shelf space requirements.

The term "sales" refers to that portion of the product's demand that can be satisfied by a given amount of inventory. If inventory is sufficiently high, then all of the demand can be satisfied, and then demand is equal to sales. However, for lower levels of inventory, sales may be less than demand. For each product, optimization requires the relationship between the product's demand, sales, and inventory, since ultimately sales are what matters.

The feature of ASO called Sales and Inventory modeling performs the calculations of how much of a product's demand will turn into sales for a given level of inventory. The results of these calculations are fed to the optimizer. The sales units of a product are dependent on the following:

- Demand for the product
- Replenishment of the product, which determines how much inventory is available to meet demand

Replenishment of the product in turn depends on:

- Various replenishment parameters, such as the frequency of replenishment. These replenishment parameters are for each product-store combination.
- Inventory levels immediately after replenishment. This level is a separate calculation on its own, and is not a single parameter, which is why it is discussed separately.

Sales and Inventory Modeling Considers All Possibilities

Because the Sales and Inventory modeling occurs before optimization, the modeling does not know on which shelf or fixture the product will ultimately end up or how many facings the products will have. Thus, the modeling performs separate calculations for each fixture that can possibly hold the product and for each possible facing value. This is a type of what-if calculation, which gives the optimizer the information about what the sales of a product would be if it were to be placed on a particular fixture with a particular number of facings. The

optimizer requires this information for all possible fixtures and facings for each product so that it can make the best choices.

Inventory Levels After Replenishment

When replenishment of a product occurs, the portion of the shelf belonging to the product is re-stocked. The level to which the product is restocked (the order-up-to level) is partly determined by how much product will fit in the shelf space that the product will occupy. The calculation of how many units of the product will fit depends on the number of facings and how many units will fit in each facing. The latter is the units per facing capacity of the product, and depends on the product and the fixture the product is being placed on. For simplicity, units per facing capacity is called just "facing capacity".

Calculating the Facing Capacity for a Product/Fixture Combination

The calculation of facing capacity consists of several steps:

1. Orientation. The orientation of the product is how the product sits on the shelf. For simplicity, in the remaining text, the terms "height, width, and depth" refer to the dimensions of the product *after* the product has been oriented correctly on the shelf. The orientation can certainly affect the number of units of the product that can fit on the shelf, and thus it is essential for each product to have its correct orientation for the shelf-space capacity calculation to produce correct results.
2. Stacking portion of Facing Capacity. This is the portion of Facing Capacity where the products are stacked one unit atop another, or nested inside one another, if the product allows nesting. The number of units that can be stacked when not nested depends on the height of the shelf, the height of the product, and the product's Max Stack Quantity, which takes precedence over the number of units allowed by the shelf height. If the product allows nesting, then the number of units that can be stacked depends, in addition, on the nesting height. When one item is nested inside another, some portion of its height is above the height of the item it is nested in. This portion of the height is the nesting height. This is then multiplied by the number of units of the product that will fit one behind the other within the depth of the fixture. This calculation also depends on the Above Gap and Behind Gap of the product.
    - Elements in this calculation: shelf height, product height, nesting height, product's Max Units High, fixture depth, product depth, Above Gap, Behind Gap.
3. Cap portion of Facing Capacity. This is the number of units of the product that can be put on top of the stacking portion with the product in a different orientation. The Facing Capacity is then the sum of the stacking portion and the cap portion. The cap space has dimensions shelf depth x product width x remaining height, where remaining height is the height left over after the product has been stacked. The cap-units calculation simply determines how many units of product will fit into this cap space, using the product's Cap Height, Cap Depth, and Cap Width. The product's Max Cap Quantity is a maximum of how many units of product can be stacked in the cap space.
    - Elements in this calculation: product's dimensions, Cap Height, Cap Depth, Cap Width, Max Cap Quantity.

If the results from sales-inventory modeling show that a product has very low service levels, meaning the product is not receiving the inventory that it needs to serve its demand, check whether the facing capacity is sufficient to meet the demand of the product. This may involve checking the elements identified above to see if they are correct.

If the facing capacity is insufficient for the demand of the product, then service levels will be low, regardless of the values for the other inventory-related settings for the product, since there is not enough room on the shelf for sufficient quantities of the product.

Maximum Capacity of a Product

In addition to the units per facing capacity, the other determinant of the order up-to level of a product is the product's Maximum Capacity. This number can be interpreted in three different ways, depending on the product's settings:

- The number of units of the product. This is the most straightforward, as the Maximum Capacity is itself just a count of units of the product.
- The number of case packs of the product. In this case, the case pack size of the product (given in the replenishment parameters for the product) is multiplied by the Maximum Capacity to convert case packs to units. (The case pack size is given in units of product.)
- The number of days of supply of the product. Here, the weekly demand for the product (from the replenishment parameters of the product) is divided by the number of days in a week, and multiplied by the Maximum Capacity to convert the Maximum Capacity into units.

Elements in this calculation: Maximum Capacity of the product, casepack size, weekly demand of the product.

The Maximum Capacity is a setting for the product and does not depend on the number of facings that the product is given.

For a given number of facings, the sales and inventory modeling determines the order up-to level by comparing two unit quantities:

- The product of the number of facings and the units per facing capacity
- The Maximum Capacity, after it is converted to units.

The smaller of these two values is the order up-to level used by the Sales and Inventory modeling for this number of facings. Thus, for example, if the fixture simply does not have enough space to hold the necessary inventory for a product, it will not help to increase the Maximum Capacity. If you attempt to increase the Maximum Capacity, and still the inventory for the product is not enough to meet demand, then it is time to check the Facing Capacity.

In general, if the results of the sales and inventory modeling show that a product is not receiving sufficient inventory, in addition to the elements affecting the facing capacity, also check the elements affecting the Maximum Capacity calculation.

Replenishment Parameters

The replenishment frequency can also affect whether the product receives sufficient inventory. The product must receive enough inventory to meet demand at least until the next review point. Thus, for example, if replenishment occurs only once per week for a product, the product's order-up-to level must be high enough to sustain at least seven days of demand for the product. If the fixture is too small to hold that much product (see the Facing Capacity section above), then increasing the replenishment frequency may help.

- Element to check: replenishment frequency

The transit time and trigger type are related replenishment parameters. The recommended trigger type is Demand Based, which means the order point, which is the inventory level that will trigger replenishment, is calculated by accounting for the demand and the transit time. With a trigger type Demand based, the order point is high enough to leave enough units to meet demand until the replenishment arrives, and thus a larger transit time will mean a higher order point.

It is possible for the transit time to be so large that the order point is too high to be contained on the fixture. For example, suppose the Facing Capacity is lower than the order point. In this case, the product will continually run out of inventory, and it is necessary to either give the product more room on the fixture or decrease the transit time for the product.

- Element to check: transit time

Sales Inventory Model Output

When the user runs ASO in space cluster mode, the store-sku-level parameters are used to run the simulations. For example, the replenishment parameters and price of the items are all input at the store-level. As a result, all simulation models are run at the store-level and then averaged over time for each store. The output is aggregated for space cluster level analysis.

Aggregation

If the user decides to optimize at the store-level, then no further aggregation is performed and the store-level output is sent and used as is in the optimization. On the other hand, if the user wants to optimize at the space cluster level, then the output of the simulation is aggregated to the space cluster level and is sent to the optimization. Thus, metrics such as Demand, Revenue, Gross Profit, Sales Units, and On Hand Units are all summed up where the service level is averaged (a simple average that sums up all service levels and divides by number of stores) across all the stores to arrive at one service level at the space cluster-level for a given possibility, as described in earlier sections. The optimization algorithm uses the cluster-level metrics to analyze trade-offs between all possibilities and ultimately reports the analysis in the Results tab.

Objective Function

The objective function specified by the user plays a major role in determining which solutions are considered best. For example, if the user specifies the objective function as maximize profit margin, then the algorithm will look for solutions that are superior on profit margin and not necessarily on the other KPIs such as revenue and sales volume. Sometimes, understanding why an item is included or dropped and why it is given so many facings might be as trivial as looking at the objective function contribution of that particular item to the objective function.

Note that the objective function not only considers contributions from products that are included in the final planogram solution but also from the dropped products, by using the Demand Transference application. The key idea here is that, for a dropped product, no sales are lost; some sales may be transferred back to other substitutable products. The Demand Transference application generates the demand transference values that may not be consumed *as is*. ASO provides flexibility for the ASO users to dampen the demand transference values if they are deemed too high.

There are a few choices for specifying an objective.

- Maximize Sales Units. This tells the optimization to fill the planogram with items that will result in best possible sales units for selected items/facings.
- Maximize Sales Revenue. For an item, sales revenue is calculated as price times the sales units. This tells the optimization to fill the planogram with items that will result in the best possible sales revenue for selected items/facings.
- Maximize Gross Profit. For an item, gross profit is obtained by multiplying the difference between price and cost, and sales units. This tells the optimization to fill the planogram with items that will result in the best possible gross profit for selected items/facings.
- Maximize Sales Revenue/On Hand Units. For each item, the ratio is calculated and added together. This objective tells the optimization to fill the planogram with items that result in the best possible revenue to the inventory units carried for each item.
- Maximize Gross Profit Return on Investment. This objective tells the optimization to fill the planogram with items that result in the best possible total profit but that at the same time minimize the total inventory cost at the planogram-level. As one can imagine, carrying too much inventory will result in higher revenues or profits but at a higher cost of

carrying more inventory. This metric lets the user strike a balance between these two metrics. There are two versions supported: Generic version when the user is unable to provide the store-level inventory. To use generic version GENERIC_GMROI_FLG is set to 'Y' in RSE_CONFIG.

- Maximize Sales Units (Weighted). This objective is similar to Maximize Sales Units, except that here each item's contribution to objective is weighted by the IPI values provided by Category Manager.
- Maximize Sales Revenue (Weighted). This objective is similar to Maximize Sales Revenue, except that here each item's contribution to objective is weighted by the IPI values provided by Category Manager.
- Maximize Gross Profit (Weighted). This objective is similar to Maximize Gross Profit, except that here each item's contribution to objective is weighted by the IPI values provided by Category Manager.
- Maximize Sales Revenue/On Hand Units (Weighted). This objective is similar to Maximize Sales Revenue/On Hand Units, except that here each item's contribution to objective is weighted by the IPI values provided by Category Manager.
- Maximize Gross Profit Return on Investment (Weighted). This objective is similar to Maximize Gross Profit Return on Investment except that here each item's contribution to the objective (in numerator and denominator) is weighted by the IPI values provided by CMPO.

Weighted metrics use IPI values provided by the Category Manager. It is generally expected that IPI values are positive and hence negative or 0 values for products will be flagged as warnings since it can result in products getting dropped.

Constraints

If the objective function focuses on the best possible solution, then constraints work in the opposite direction, by restricting the set of possible solutions. For example, if the objective function says to select the most profitable item A and assign the maximum number of facings (for example, 15), a constraint on item A may say it is not possible to have more than three facings for item A in order to provide a minimum number of facings to the items of the other brands.

Optimization enforces all constraints as required except for Product Family Group constraints; that is, it finds all the solutions that satisfy all the constraints except for Product Family Group constraints that are specified by the user.

Sometimes, inadvertently, the user might specify conflicting constraints that can result in no solution or unexpected solutions. Often, a resolution can be found by just understanding the implications of individual constraints. Some examples of how to analyze the constraints:

- Why was item B dropped? Is the Average Weekly Demand too low?
- Why did item C receive so many facings? Is this item part of Match Facings group? Or what is the minimum number of facings rule?
- Why is an item placed at this elevation? What are the elevation restrictions on this item?
- Why is an item in second bay and not in the first bay? Is there a blocking rule in place?

More often than not, the user must analyze the interplay between two or more constraints to see why an item is included or dropped and why it is assigned so many facings, and why is it placed in a particular bay and at a particular elevation. A Sanity Checker is provided in the ASO Actions menu that can be used to identify some of the logical conflicts that have arisen due to the definition of the constraints.

Product Family Group Constraints

Among all the constraints specified by the user, Product Family Group constraints are enforced softly. To elaborate on the PFG constraints: Product Family Group constraints are defined using the sort attributes (up to three attributes) mentioned in the Visual Guidelines screen. Note that these Sort Attributes are also used in laying out items within a shelf. These constraints tell the optimization that products belonging to a particular combination of sort attribute1, sort attribute 2 and sort attribute 3 are to be kept together or "close enough". An example of a Product Family Group is Deodorants belonging to a Brand, Size and Scent to be placed together.

It should be noted that Product Family Group constraints are different from Same Shelf constraints or Visual Guidelines as follows:

- Product Family Group constraints are a relaxation of same shelf constraints - products do not have to be kept in same shelf but in nearby shelves.
- Product Family Group constraints are different from Visual Guidelines as they do not enforce any order between the groups.

Some recommendations on using Product Family Group Constraints:

- ASO provides three tools to help in achieving the intended layout: Same Shelf, Visual Guidelines, and Product Family Group. Note that Sorting Attributes are dual purpose as they also help in laying out items within a shelf. Figure 9–3 explains the three concepts.

Figure 9–3 Blocking, Product Family Group, Same Shelf, Sorting Illustrated

- The user should make use of different tools provided to achieve the intended layout. For example, the user can define the order using the Visual Guidelines and further keep products close enough within a block, and the user can define Product Family Group constraints. PFG constraints provide the ability to keep the products together within a block (for example, vertical block + horizontal block).
- An example of a redundant PFG strategy would be to use same attribute to define visual guidelines (for example, use brand to define primary vertical blocking) and for product family group (for example, use Brand for defining Product Family Group). In such a case, Visual Guidelines will keep products for each Brand together, and create blocks using the order specified. Product Family Groups are redundant here.

Diagnosis of Visual Guidelines

Visual guidelines are imposed as a set of constraints that restrict the placement of an item. Specification of the visual guidelines contains the following:

- Blocking strategy
- Primary blocking attributes (up to two attributes)
- Secondary blocking attributes (up to two attributes)

A blocking strategy can have a design in which primary blocking is vertical and secondary blocking is horizontal or primary blocking is horizontal and secondary blocking is vertical. When the primary blocking is vertical, the application allows the user to specify horizontal

blocking attributes by each vertical block. Similarly, when primary blocking is horizontal, the user can specify the vertical blocks by each horizontal block. Note that both strategies allow the user to specify additional horizontal blocks called top or bottom, which allows the user to put items in the top shelves or bottom shelves.

The following illustration clarifies these strategies. Assume that there are four brands, A, B, C, and D. Users also have the ability to create a merged block, as shown in the following example. Brand C and Brand D are merged to create a combined vertical block, as the user believes that these are premium brands that do not have as many SKUs as Brand A and Brand B. In the first scenario, the user wants the items placed in the following order. From left to right, the user wants to see Brand A, then Brand B, and then Brand C and D. Further, from top to bottom, the user wants to distinguish Brand A by size (since the user believes the size plays a major part in customer purchasing decision for Brand A). For Brand B, the user wants the flavors arranged from top to bottom. The (Vertical, Horizontal) strategy is shown in Table 9–1. Alternatively, the user can decide that size is the most important attribute across all brands and specify that horizontal rows of shelves should be of same size. The (Horizontal, Vertical) strategy is shown in Table 9–2.

Table 9–1 Vertical, Horizontal Strategy

Brand A	Brand B	Brand C
12 oz	Vanilla	-
24 oz	Chocolate	-
36 oz	Strawberry	Brand D
-	Multipack	-

Table 9–2 Horizontal, Vertical Strategy

Brand A	Brand B	Brand C and D
12 oz	12 oz	12 oz
24 oz + vanilla	24 oz + vanilla	24 oz + vanilla
24 oz + others	24 oz + others	24 oz + others
36 oz	36 oz	36 oz

These two strategies provide the flexibility for the user to define which takes precedence, vertical or horizontal, using only attributes. Note that the user does not actually indicate which shelves or elevation a product should be placed at. Instead, the optimization decides the shelves or elevations, based on the order of items specified by the user, using attributes. When the primary attribute is vertical, the optimization tries to follow the order specified for vertical blocks and then tries to place products by the horizontal blocks defined. In contrast, when the primary attribute is horizontal, the optimization tries to generate solutions that adhere to the horizontal blocking order and vertical blocking order that is common to all horizontal blocks. In the example above, for the former, Brand A has three horizontal blocks, Brand B has four horizontal blocks, and the merged block has two horizontal blocks. For latter, there are four horizontal blocks and three vertical blocks for all horizontal blocks.

An example of solution to the primary as vertical and secondary as horizontal strategy stated in Table 6-1 is shown in Figure 9–4.

Figure 9–4 Primary Strategy as Vertical

<table>
<tr><th>Bay 1</th><th colspan="2">Bay 2</th><th>Bay 3</th><th colspan="2">Bay 4</th></tr>
<tr><td>Brand A+12 oz</td><td>Brand A + 12 oz</td><td>Brand B+Vanilla</td><td>Brand B + Vanilla</td><td rowspan="2">Brand B + Vanilla</td><td>Brand C</td></tr>
<tr><td>Brand A+12 oz</td><td>empty space</td><td>Brand B+Vanilla</td><td>Brand B + Vanilla</td><td>Brand C</td></tr>
<tr><td>Brand A+12 oz</td><td rowspan="2">Brand A + 24 oz</td><td rowspan="2">Brand B + Chocolate</td><td>Brand B + Vanilla</td><td>Brand B + Vanilla</td><td>Brand C</td></tr>
<tr><td>Brand A + 24 oz</td><td>Brand B + Chocolate</td><td>empty space</td><td>Brand C</td></tr>
<tr><td>Brand A + 24 oz</td><td>Brand A + 24 oz</td><td>Brand B + Chocolate</td><td>Brand B + Chocolate</td><td>Brand B + Chocolate</td><td>Brand D</td></tr>
<tr><td>Brand A + 36 oz</td><td>empty space</td><td>Brand B + Chocolate</td><td>Brand B + Chocolate</td><td>Brand B + Chocolate</td><td>Brand D</td></tr>
<tr><td>Brand A + 36 oz</td><td>Brand A + 36 oz</td><td>Brand B + Strawberry</td><td rowspan="2">Brand B + Multipack</td><td>empty space</td><td>Brand D</td></tr>
<tr><td>Brand A + 36 oz</td><td>Brand A + 36 oz</td><td>Brand B + Multipack</td><td>empty space</td><td>Brand D</td></tr>
</table>

An example of solution to the primary as horizontal and secondary as vertical strategy stated in Table 6-2 is shown in Figure 9–5.

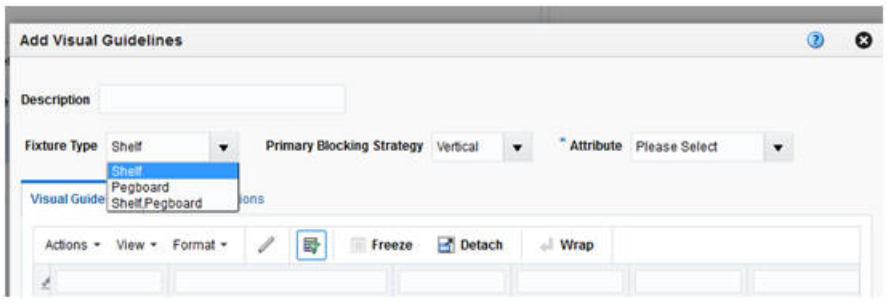
Figure 9–5 Primary Strategy as Horizontal

<table>
<tr><th>Bay 1</th><th colspan="2">Bay 2</th><th>Bay 3</th><th colspan="2">Bay 4</th></tr>
<tr><td>Brand A+12 oz</td><td>Brand A + 12 oz</td><td>Brand B+12 oz</td><td>Brand B + 12 oz</td><td rowspan="2">Brand B + 12 oz</td><td rowspan="2">Brand C&D + 12 oz</td></tr>
<tr><td>Brand A+12 oz</td><td>empty space</td><td>Brand B+12 oz</td><td>Brand B + 12 oz</td></tr>
<tr><td>Brand A+12 oz</td><td rowspan="2">Brand A + 24 oz + Vanilla</td><td rowspan="2">Brand B + 24 oz + Vanilla</td><td>Brand B + 24 oz + Vanilla</td><td>Brand B + 24 oz + Vanilla</td><td>Brand C&D + 24 oz + Vanilla</td></tr>
<tr><td>Brand A + 24 oz + Vanilla</td><td>Brand B + 24 oz + Vanilla</td><td>Brand B + 24 oz + Vanilla</td><td>Brand C&D + 24 oz + Vanilla</td></tr>
<tr><td>Brand A + 24 oz + Others</td><td>Brand A + 24 oz + Others</td><td>Brand B + 24 oz + Others</td><td>Brand B + 24 oz + Others</td><td>empty space</td><td>Brand C&D + 24 oz + Others</td></tr>
<tr><td>Brand A + 24 oz + Others</td><td>Brand A + 24 oz + Others</td><td>Brand B + 24 oz + Others</td><td>Brand B + 24 oz + Others</td><td>Brand B + 24 oz + Others</td><td>Brand C&D + 24 oz + Others</td></tr>
<tr><td>Brand A + 36 oz</td><td>Brand A + 36 oz</td><td>Brand B + 36 oz</td><td rowspan="2">Brand B + 36 oz</td><td>empty space</td><td>Brand C&D + 36 oz</td></tr>
<tr><td>Brand A + 36 oz</td><td>Brand A + 36 oz</td><td>Brand B + 36 oz</td><td>Brand B + 36 oz</td><td>Brand C&D + 36 oz</td></tr>
</table>

Visual Guidelines provide flexibility to model various layouts, and yet improper specification of visual guidelines can result in empty spaces or sparsely populated optimal planograms. The next few paragraphs provide some recommendations on how to use Visual Guidelines:

1. The user can specify visual guidelines by fixture type or common to all fixture types as shown in Figure 9–6. The user must define separate guidelines for different fixture types since products or product attributes are not common to both fixture types.

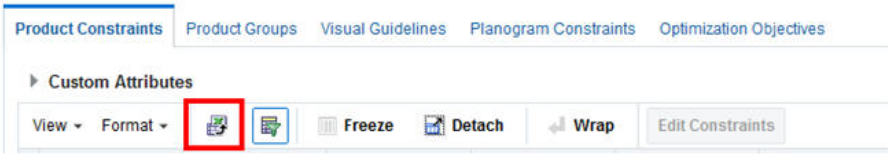
Figure 9–6 Add Visual Guidelines



2. In general, adding visual guidelines restricts the optimization, so the user should create a run without any visual guidelines. Later, the user can create runs with a variety of visual guidelines. Comparing the no-VG and VG runs provides the impact of imposing visual guidelines, in terms of revenue, sales units, and so on.

3. It is preferred that when adding visual guidelines, the user should bear in mind to start with a simple blocking strategy (for example, primary only) and to check the results and increase complexity in terms of additional and secondary strategies.
4. As stated above, there are two block strategies that are supported by ASO: (a) Primary is Vertical, Secondary is Horizontal (b) Primary is Horizontal, Secondary is Vertical. In case of shelf fixtures only, each horizontal block will need at least one shelf to satisfy and thus it is a restrictive strategy than the former. Further, the second strategy, in case of shelf fixtures, needs at least one shelf for each horizontal block and thus, the number of horizontal blocks is limited to number of shelves in the bay. For example, it is pointless to define ten horizontal blocks for strategy defined in Table 9–1 as each bay has at most eight shelves
5. It is essential that the user does a preliminary analysis on the product counts and KPIs like Sales Units to see how well the blocks are defined. Very few products will result in thin blocks or empty spaces; low KPIs typically result in sacrificing the block for another block with better products. ASO provides Export to Excel on Product Constraint tab so that the user can check product counts. Here is an example of the process:
    a. The user can select Custom Attributes (up to three attributes at a time) and click the **Show Attributes** button.
    b. The user can then click **Export to Excel** and download the products with attributes information into Excel as shown in Figure 9–7.

Figure 9–7 Export to Excel



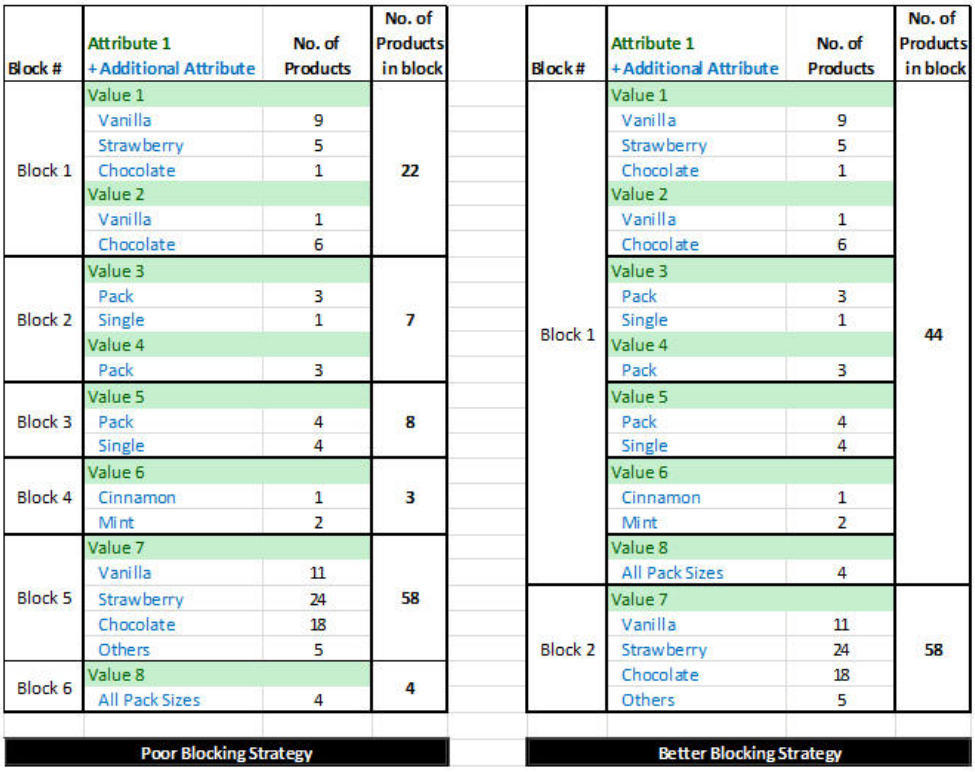
    c. An example of the analysis is shown below: the user would like to define blocks using two attributes: Segment and Pack-size.

    Exporting data to Excel and performing a pivot on the attributes gives us the Table 6-3. Attribute values highlighted in green color are Segment and others are Pack-size values.

    Notice that the Mouthwash segment seems the largest in terms of product counts whereas all other segments have few products. Given the disproportionate distribution of product counts, it might be better to start with only two blocks, one for Mouthwash and rest all in others. This gives even distribution of products in all the blocks.

    Next, the user needs to check the value of each block. It is possible that some blocks are not valuable and optimization can trade that block for another block with higher value. ASO Visual Guidelines screen provides the ability to see the KPIs like Sales Units, and Revenue. This gives the user the ability to understand the value of each block.

Figure 9–8 Product Counts for Blocking Strategies



Block #	Attribute 1 +Additional Attribute	No. of Products	No. of Products in block
Block 1	Value 1		22
	Vanilla	9	
	Strawberry	5	
	Chocolate	1	
	Value 2		
	Vanilla	1	
	Chocolate	6	
Block 2	Value 3		7
	Pack	3	
	Single	1	
	Value 4		
	Pack	3	
Block 3	Value 5		8
	Pack	4	
	Single	4	
Block 4	Value 6		3
	Cinnamon	1	
	Mint	2	
Block 5	Value 7		58
	Vanilla	11	
	Strawberry	24	
	Chocolate	18	
	Others	5	
Block 6	Value 8		4
	All Pack Sizes	4	

Poor Blocking Strategy

Block #	Attribute 1 +Additional Attribute	No. of Products	No. of Products in block
Block 1	Value 1		44
	Vanilla	9	
	Strawberry	5	
	Chocolate	1	
	Value 2		
	Vanilla	1	
	Chocolate	6	
	Value 3		
	Pack	3	
	Single	1	
	Value 4		
	Pack	3	
	Value 5		
	Pack	4	
	Single	4	
	Value 6		
	Cinnamon	1	
	Mint	2	
	Value 8		
	All Pack Sizes	4	
Block 2	Value 7		58
	Vanilla	11	
	Strawberry	24	
	Chocolate	18	
	Others	5	

Better Blocking Strategy

Finally, the user needs to decide whether to define primary as vertical or primary as horizontal. In general, the rule of thumb is to define primary as vertical since it is less restrictive than primary as horizontal.

After optimization, the user can quickly check how the blocks are formed. Optimization tries its best to satisfy all constraints and provide optimal amount of space for each product. This can result sometimes in empty spaces within each block. In such a case, the user should revisit blocking strategy; perhaps few smaller blocks can be merged into the bigger block.

Product Groups

Product groups provide a set of constraints for the optimization that specify the relation between any pair of items. For example, retailer has to pair a high-margin product with low-margin product or retailer has to match number of facings for shampoos and conditioners. ASO provides a few variants on these constraints.

Table 9–3 Product Group Constraints

Constraint	Description
At Least	At least *m* items must be selected in the final assortment.
Exact	Exactly *m* items must be selected in the final assortment.
At Most	At most *m* items must be selected in the final assortment.
All or Nothing	If one item from this group is selected, then all other items in this group must be selected in the final assortment.
Match Facings	Whatever items are selected, all the items selected must be given same number of facings.

Table 9–3 (Cont.) Product Group Constraints

Constraint	Description
Same Shelves	Whatever items are picked, they must be placed on the same shelf in the final planogram (only applicable for shelf fixture type).

Validation Tool (Sanity Checker)

This feature helps the user to identify logical conflicts that may occur because of the constraints imposed and provides guidance on how to resolve the issues. The validation feature generates two types of alerts: Error and Warning. Error is generated when the set of constraints result in no solution. Warning is informative in nature and does not necessarily result in no solution. The user should try to understand why an error or warning is generated and examine the resolution provided.

Diagnosis of Dropped Products

The Dropped Products tab provides information on which products are dropped and the high-level reason why they are dropped. Coupled with Validation Errors/Warnings, the user can get an idea on why a product is dropped. The user should try to understand why an error or warning is generated and examine the resolution provided.

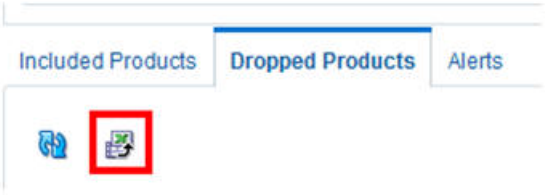
Further, the user should use Export to Excel to understand the high-level issues for dropped products. They are categorized primarily as follows:

- Do Not Include.
- No Solution. When the optimizer cannot satisfy the constraints specified, then it does not return any solution. For example, if all items need minimum of ten facings and are mandatory. This is not physically possible on the POG with such constraints and hence optimizer will return no solution.
- User Constraints. When a product cannot fit into any shelf due to its geometry or display orientation. To identify which products, the user should refer to Validation Errors/Warnings panel.
- Missing Sales and/or Forecast Data. As stated, a product does not get included because there is missing data. To identify which data elements are missing, the user should refer to Validation Errors/Warnings panel.
- Invalid Sales and/or Forecast Data. As stated, a product does not get included because there is invalid data (for example, price = 0). To identify which data elements are invalid, the user should refer to Validation Errors/Warnings panel.
- Minimum Service Level. When a product cannot satisfy the minimum service level (for example, 90%) then the product is dropped from the POG. Refer to the Sales and Inventory Model section of the Implementation Guide on how it can be improved.
- Solver Choice. This can be due to the objective defined and contribution of the item compared to the space needed for that item. Some examples of why Solver (or Optimization) drops the items:
  - Item has very low demand; the user should check why the demand is low.
  - Item got dropped due to low service levels since it needs many facings to meet minimum service level requirement.
  - Item got dropped because optimization compared the item's contribution to the objective function to the space needed and decided that it is best if that space is given to another product.

In general, the user can conduct the diagnosis of the errors/dropped products as follows:

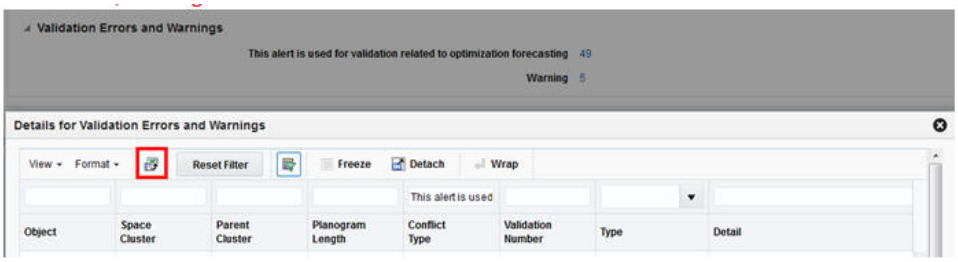
1. Go to Dropped Product panel and use the Export to Excel functionality. Looking at the Reason column, the user can determine why a product was dropped.

Figure 9–9 Dropped Products



2. To identify further specifics about the errors, the user can click **Validation/Errors** and click the hyperlink and use the Export to Excel functionality. This gives specific product-cluster errors/warnings.

Figure 9–10 Validation Errors



Checklist for Optimization Results Diagnosis

Before an SR is submitted, the user should review this checklist to identify any issues. If the issue is not resolved by walking through this checklist, then an SR can be submitted. Note that when an SR is submitted, the user must provide the following information:

- Issue description and expected behavior. As the below details are gathered, be sure the selections in the UI match the expectations and that the constraints were not specified for the wrong cluster.
- Validation Errors/Warnings Excel (using Export to Excel feature)
- Dropped Products Excel
- Replenishment Data
- Visual Guidelines Excel (using Export to Excel feature in Visual Guidelines UI)
- Analysis of Visual Guidelines Excel ((for example, Product Counts by Attributes used in defining the Visual Guidelines. See section Diagnosis of Visual Guidelines)
- VPOG Screenshot (Using All Visual Guidelines display option)
- Run Number, Space Cluster ID

Monitoring Batch Processes

Monitoring the batch processes can help you make sure that the complete data required for ASO to function correctly is being used.

Overview

Here is an overview of the process.

1. Make sure the daily files (RI_RMS_DATA.zip) are sent.
2. Approximately an hour after an intraday process starts, retrieve the ORASE_INTRADAY_ extract.zip from the FTP server and remove it. Do not send a new intraday set of data files before you retrieve and remove the extract file.
3. If the extract file is not available on the FTP site, the intraday process has failed. If this is the case, you will receive an email from Oracle Support regarding the errors that caused the failure (see"Batch Process Failure"). Do not send a new intraday set of data files before you have resolved the errors. Any data sent in the intraday zip file that failed is not guaranteed to have loaded. (The failure may have prevented the loading of some files.) Once you have resolved the cause of the load failure, send a new intraday zip file that includes the data previously provided.
4. Review the global validation issues in the so_global_val_detail.txt and so_global_val_ smry.txt files provided in the extract file for possible issues (see "Global Validation Issues"). Resolve the issues identified, and resend the appropriate data required to fix those issues.

Batch Process Failure

This section describes the process for addressing a batch process failure.

When a batch process failure occurs, Oracle Support sends you an email notification similar to the following example.

> Subject: ORASE Batch Process - ORASE - LOADERRORS

>

> This is an auto-generated email

>

> Customer: Department Store

> Environment: Production

>

> Batch Type: ORASE Batch

> Batch Frequency: intraday

> ################ Data Load Error Summary

> INTERFACE ERROR_DATE ERROR_ID NUM_ERRORS ERROR_ DESCR

> ------------------------------ ------------------- --------- ------------ ----------

> SO_DISPLAY_STYLE_STG 2018/04/04 02:02:15 2014024 3 Display style key does not exist in so_prod_display_style_stg (incoming display styles).

In this example, the email message indicates that three records in so_display_style_stg.txt have keys that do not have corresponding display style keys in so_prod_display_style_stg .txt. In other words, a value was provided in the so_display_style_stg key column that did not exist in the so_prod_display_style_stg display_style_key column. To resolve the error, you must add the missing display style keys to so_prod_display_style_stg .txt.

In order to see the data that is in error, you must review the corresponding BAD table. Log into the application as a user with the enterprise role of FORECAST_MANAGER_JOB. Then, using the Data Management/Manage Configuration menu option, select the appropriate table that corresponds to the interface. For example, interface SO_DISPLAY_STYLE_STG contains error records in table SO_DISPLAY_STYLE_BAD.

Global Validation Issues

This section describes the process for addressing global validation issues.

Here is an example file:

2025040|SO_GV_MAPPING_ASSORT_POG_NO_POGSET|Assortment to POG mapping data is using a POG Set that does not exists.|SO_POG_ASSORT_MAPPING_STG|POG_DEPT_KEY|POG_CATEGORY_KEY|POG_SUB_CATEGORY_KEY|Y|2017-10-22

2025041|SO_GV_MAPPING_ASSORT_POG_SEASON_NO_POGSET|Assortment to POG seasonal mapping data is using a POG Set that does not exists.|SO_POG_ASSORT_SEAS_MAPPING_STG|POG_DEPT_KEY|POG_CATEGORY_KEY|POG_SUB_CATEGORY_KEY|Y|2017-10-22

In this example, the errors indicate that the assortment-to-POG mapping data and assortment-to-POG seasonal mapping data are using POG sets that were not provided. The list of the missing sets is available in the so_global_val_detail.txt file. To resolve the error, you must send the missing planograms in the appropriate data files (see "Sending Data in Data Files").

Figure 9–4 contains a list of global validation error conditions that are checked for, and may be reported in the so_global_val_smry.txt and so_global_val_detail.txt files. The information contained in the table can help identify the interface that has an issue, along with a reference of columns that are related to the data issue.

Table 9–4 Global Validation Errors

Error ID	Name	Description	Interface	Column1 Name	Column2 Name	Column3 Name
2025001	SO_GV_DS_NO_ORIENTATION	Display Style is missing orientation data.	SO_DISP_STYLE_ORIENTATION_STG	DISPLAY_STYLE_KEY	-	-
2025002	SO_GV_DS_NO_FIXTURE	Display Style is missing fixture data.	SO_DISPLAY_STYLE_FIXTURE_STG	DISPLAY_STYLE_KEY	-	-
2025010	SO_GV_POG_MISSING_STORES	POG is missing store information.	SO_POG_STORE_STG	POG_KEY	-	-
2025011	SO_GV_POG_MISSING_BAYS	POG is missing bay information.	SO_POG_BAY_STG	POG_KEY	-	-
2025012	SO_GV_POG_MISSING_FIXTURE	POG is missing fixture information.	SO_BAY_FIXTURE_STG	POG_KEY	-	-
2025013	SO_GV_POG_BAY_MISSING_FIXTURE	Bay is missing fixture information.	SO_BAY_FIXTURE_STG	BAY_KEY	POG_KEY	-
2025014	SO_GV_POG_SHELF_FIXTURE_NO_SHELVES	A shelf fixture is missing shelves data.	SO_SHELF_STG	FIXTURE_KEY	POG_KEY	-

Table 9–4 (Cont.) Global Validation Errors

Error ID	Name	Description	Interface	Column1 Name	Column2 Name	Column3 Name
2025015	SO_GV_POG_WRONG_FIXTURE_TYPE	Non shelf fixture has shelves assigned to it.	SO_FIXTURE_STG	KEY	POG_KEY	-
2025016	SO_GV_POG_WRONG_FIXTURE_COMBINATION	POG has an invalid fixture combination (Shelf, Pegboard, Freezer or Shelf/Pegboard).	SO_FIXTURE_STG	POG_KEY	-	-
2025020	SO_GV_ASSORTMENT_NO_CLUSTERS	Assortment does not have mandatory assortment clusters.	SO_ASSORTMENT_STG	ID	-	-
2025021	SO_GV_ASSORTMENT_NO_PRODUCT	Assortment does not have mandatory products.	SO_ASSORTMENT_STG	ID	-	-
2025022	SO_GV_ASSORTMENT_CLUSTER_NO_STORE	Assortment cluster has no stores.	SO_ASSORT_CLUSTER_STG	ASSORTMENT_ID	CLUSTER_KEY	-
2025023	SO_GV_ASSORTMENT_CLUSTER_NO_PRODUCT	Assortment cluster (cluster level assortment) has no products.	SO_ASSORT_CLUSTER_STG	ASSORTMENT_ID	CLUSTER_KEY	-
2025024	SO_GV_ASSORTMENT_STORE_NO_PRODUCT	Assortment store (store level assortment) has no products.	SO_ASSORT_PRODUCT_STRCLTR_STG	ASSORTMENT_ID	CLUSTER_STORE_KEY	LOCATION_KEY
2025025	SO_GV_ASSORTMENT_NO_FORECAST	Assortment is missing forecast data.	SO_ASSORT_PROLOC_FCST_STG	ASSORTMENT_ID	-	-
2025026	SO_GV_ASSORTMENT_PRODLOC_NO_FORECAST	Assortment product/location is missing forecast data.	SO_ASSORT_PROLOC_FCST_STG	ASSORTMENT_ID	PRODUCT_KEY	LOCATION_KEY
2025027	SO_GV_ASSORTMENT_NO_PRICE_COST	Assortment is missing price and cost data.	SO_ASSORT_PROLOC_FCST_STG	ASSORTMENT_ID	-	-
2025028	SO_GV_ASSORTMENT_PRODLOC_NO_PRICECOST	Assortment product/location is missing price/cost data.	SO_ASSORT_PROLOC_PRICECOST_STG	ASSORTMENT_ID	PRODUCT_KEY	LOCATION_KEY
2025029	SO_GV_ASSORTMENT_PRODUCT_NO_DISPLAY_STYLE	Assortment product does not have a display style.	SO_DISPLAY_STYLE_STG	ASSORTMENT_ID	PRODUCT_KEY	-

Table 9–4 (Cont.) Global Validation Errors

Error ID	Name	Description	Interface	Column1 Name	Column2 Name	Column3 Name
2025030	SO_GV_ ASSORTMENT_ PRODUCT_NO_ REPLENISHME NT	(Assortment) Product/location does not have replenishment data.	SO_PROD_ LOC_REPL_ PARAM_STG	ASSORTMENT_ ID	PRODUCT_KEY	LOCATION_ KEY
2025031	SO_GV_ ASSORTMENT_ SAME_ CATEGORY	Assortment overlaps with another assortment for the same product category.	SO_ ASSORTMENT_ STG	ID	-	-
2025040	SO_GV_ MAPPING_ ASSORT_POG_ NO_POGSET	Assortment to POG mapping data is using a POG Set that does not exists.	SO_POG_ ASSORT_ MAPPING_STG	POG_DEPT_ KEY	POG_ CATEGORY_ KEY	POG_SUB_ CATEGORY_ KEY
2025041	SO_GV_ MAPPING_ ASSORT_POG_ SEASON_NO_ POGSET	Assortment to POG seasonal mapping data is using a POG Set that does not exists.	SO_POG_ ASSORT_SEAS_ MAPPING_STG	POG_DEPT_ KEY	POG_ CATEGORY_ KEY	POG_SUB_ CATEGORY_ KEY

More details about so_global_val_smry.txt and so_global_val_detail.txt files can be found in the Data Interface document ri_orase-<release number>-intf.xlsx. The link to the document can be found on the documentation site under *Oracle Retail Analytics and Planning Cloud Services Data Interfaces*.

Sending Data in Data Files

This section describes the data that must be sent in specified files. Note that, for each set of files, the text indicates whether the files must be an incremental or a full dataset.

Assortment-Related Files

Each time an assortment is delivered, all the data elements that define that assortment must also be delivered within the related interface files. Note the following:

- If an assortment appears within so_assortment_stg file, it is expected that the data for that assortment will be delivered again within the other files as well. A full replacement of that assortment and all its components is performed, using the data in the set of files.
- A change in the assortment resets the mapping and optimization data used for runs. The optimization work for that assortment must be re-started.
- A new assortment can be delivered at any time without impacting an existing assortment.
- The last two files in Table 9–5 are used to signal the finalization of the assortment set. They are both optional (that is, the data does not have to be delivered in every submission).
- SO_ASSORT_PROLOC_PRICECOST_STG contains two fields: COST and INVENTORY_COST. The difference between them is that former is the unit product cost and the latter is the unit cost of inventory. The latter field is only useful when a user selects GMROI as the objective.

Table 9–5 Assortment-Related Files

File Name	Requirement
SO_ASSORTMENT_STG	Mandatory
SO_ASSORT_CLUSTER_STG	Mandatory
SO_ASSORT_CLUSTER_MEMBER_STG	Mandatory
SO_ASSORT_PRODUCT_STRCLTR_STG	Mandatory
SO_ASSORT_PROLOC_FCST_STG	Mandatory
SO_ASSORT_PROLOC_PRICECOST_STG	Mandatory
SO_ASSORT_PHPROD_LIKE_PROD_STG	Optional
SO_ASSORT_PHPROD_ATTR_STG	Optional
SO_ASSORTMENT_FINALIZED_STG	Optional
SO_ASSORT_PHPROD_FINALIZED_STG	Optional

POG-Related Files

Every time a planogram is delivered, all the data elements that define that planogram must also be delivered in the related interface files. Note the following:

- A new bay cannot simply be added to an existing POG. If the POG must be changed, all the components must be delivered again. A full replacement of the POG and all the data elements related to it is performed.
- The POG resets the mapping and optimization data used for runs. The optimization work for that POG SET must be re-started.
- A new POG can be delivered at any time. However, there can be an impact if the new POG belongs to an existing POG SET in which other POGs are already present. This also causes a reset to the mapping and optimization data used for runs.

Table 9–6 POG-Related Files

File Name	Requirement
SO_POG_STG	Mandatory
SO_POG_STORE_STG	Mandatory
SO_POG_BAY_STG	Mandatory
SO_BAY_FIXTURE_STG	Mandatory
SO_FIXTURE_STG	Mandatory
SO_BAY_FIXTURE_SHELF_STG	Mandatory
SO_SHELF_STG	Mandatory

Display-Style Files

Each time a display style is delivered, the related information must be delivered in these files. The existing information is refreshed with the new information, a full replacement of the display style and its components.

Adding or changing a display style for a product can be done at any time. It does not reset the mapping and optimization data used for runs. This information can be delivered.

Table 9–7 Display-Style Files

File Name	Requirement
SO_DISPLAY_STYLE_STG	Mandatory
SO_PROD_DISPLAY_STYLE_STG	Mandatory
SO_DISPLAY_STYLE_FIXTURE_STG	Mandatory
SO_DISP_STYLE_ORIENTATION_STG	Mandatory

POG Historical Data and Store CDAs

This data can be delivered at any time for POGs that were previously delivered or that are being delivered in the same batch. These files are optional. They do a full replacement of data that may have already been delivered for the same POGs.

Data from these files does not reset the mapping and optimization data used for runs.

Table 9–8 POG Historical Data and Store CDAs

File Name	Requirement
SO_FIXTURE_DISP_CONFIG_STG	Optional
SO_PEGBOARD_DISP_CONFIG_STG	Optional
SO_POG_STORE_CDA_STG	Optional
SO_POG_DISPLAY_STYLE_STG	Optional

Mapping, Replenishment, and Other Files

These interface files can be delivered incrementally. You do not need to provide the full set every time, only deltas and changes. For large files such as replenishment files, only updates are required, not the full set each time.

Data from these files does not reset the mapping and optimization data used for runs.

Table 9–9 Mapping, Replenishment, and Other Files

File Name	Requirement
SO_PROD_LOC_REPL_PARAM_STG	Mandatory
SO_POG_ASSORT_MAPPING_STG	Mandatory
SO_POG_ASSORT_SEAS_MAPPING_STG	Mandatory
SO_PROD_STACK_HEIGHT_LIMIT_STG	Optional

10

Assortment Recommender

This chapter provides details about the Assortment Recommender, a batch-run system that provides recommendations for changing the current assortments in each store and each category in order to increase revenue or gross profit from that category at that store.

Prerequisites

The Assortment Recommender relies on output from the DT application of the application. In particular, every category/store combination that is to receive assortment recommendations must have DT results in an approved DT version. Some of the concepts used in the Assortment Recommender, such as substitutable and incremental demand, are concepts that the DT application also uses.

Producing Better Assortments

The Assortment Recommender starts from the current assortments for each category in each store and finds assortment changes that will increase either the total revenue or total gross profit of the assortment. Each category-store combination is optimized separately.

The total revenue or total gross profit calculation accounts for

- Cannibalization effects, by using demand transference, and
- Halo effects, meaning additional revenue a SKU in category brings by encouraging the purchase of complementary SKUs in other categories.

The assortment changes recommended are those that improve revenue or gross profit after accounting for the above effects. For example, the Assortment Recommender may recommend dropping an item that is very similar to the other items in the assortment and replacing it with an item that is less similar to the other items, because dropping the similar item does not decrease total revenue by much, and adding the dissimilar item brings in additional revenue. Similarly, the Assortment Recommender may recommend dropping a SKU that brings little halo revenue in favor of a SKU that brings in more halo revenue (or gross profit).

The calculation of revenue or gross profit does not include any revenue that an assortment change in other categories may bring though halo effects. Each category at a store is optimized separately, so when optimizing a category B, it is not possible to include revenue that another category A may bring to B through the halo effect of a SKU in A on B.

The Assortment Recommender lets you choose which of the following to maximize:

- total assortment sales units
- total assortment revenue
- total assortment gross profit

These quantities include halo sales units, halo revenue, or halo gross profit, respectively

The Assortment Recommender runs as a batch process and on a set schedule produces recommended assortment changes for each category/store combination using whatever is the current assortment at the time it runs. See the following sections for how the schedule can be configured.

Run Groups and Run Frequency

It is unnecessary to obtain frequent assortment-change recommendations for every single category/store combination. For many category/store combinations, recommendations may only be necessary at infrequent intervals, or not at all. The Assortment Recommender provides interfaces to allow external control of the run frequency of category/store combinations in the following way.

The interfaces allow for defining a run group, which consists of a set of categories and a set of stores. When a run group executes, every combination of category/store from the set of categories and set of stores receives new recommendations. The run group as a whole is associated with a run frequency, which specifies how often the category/store combinations in it receive new recommendations. In this way, the run group is a mechanism to control:

- The category/store combinations that receive recommendations. For example, if, for category A, only stores 1, 2, and 3 should receive recommendations, then a run group can be set up with category A, and stores 1, 2, and 3.
- The frequency of recommendations. A retailer may be particularly interested in certain key categories, so it makes sense to schedule recommendations only for those categories. However, not every category may need frequent recommendations.

Keep in mind that when the run group executes, every combination of category/store in the run group receives recommendations. If the run group has 10 categories and 500 stores, this is 10 x 500 = 5,000 sets of recommendations and 5,000 separate calculations to produce those recommendations.

The Run Group Parameters

Various parameters control how the recommendation calculation proceeds when a run group executes. These parameters must be present in the run-group tables in the database. Within a run group, each category has its own set of these parameters because the parameters depend on the category. The parameters are:

- The set of Must-Keep SKUs. Each category in a run group can have a list of such SKUs, which indicate to the Assortment Recommender that these SKUs must not ever be removed from the assortment. For example, key items for the retailer in the category must be on the list. The list can be empty, in which case the assortment recommender is free to swap out any SKU currently in the assortment.
- The Assortment-Size Change. This is an integer that can be negative, positive, or 0. It indicates the change the Assortment Recommender must make to the number of SKUs in the assortment. Suppose the value is C. Then the Assortment Recommender will change the assortment size by C. If C is 0, then the assortment size stays the same. If negative, the assortment size decreases by -C, and if positive the assortment size increases by C. There is one value of C for each category in a run group. For a given category, C applies to all of the stores in the run group. It may not be possible for the Assortment Recommender to achieve a change of C for a particular category in a particular store, but it will attempt to get as close as possible.
- The Min-Keep Percent. This is a non-negative percentage. Suppose the value is M. Then the Assortment Recommender keeps at least M percent of the SKUs in the assortment to be

the same. Suppose M were 50 percent. Then at least half the assortment will consist of SKUs that are already in the assortment, but the Assortment Recommender is free to choose which 50 percent to change. This parameter, along with the Must-Keep SKUs, is useful for ensuring that the Assortment Recommender does not make too many changes to the assortment.

- The assortment metric to maximize. The choices here are: sales units, revenue, or gross profit. This is a total over the entire assortment, and the Assortment Recommender recommends assortment changes that increase the chosen metric from what it is for the current assortment. The choice of assortment metric is per category within a run group, and it applies to all of the locations in the run group. For example, if a certain category is a loss leader but drives the customer to make other purchases, then the choice of metric for this category might be sales units.

Data Used by the Assortment Recommender

The Assortment Recommender requires the following data to generate assortment recommendations. Each data element is derived automatically from other data in the application schema and fed into the Assortment Recommender. The following list describes the data and how it is derived.

Required Data

- The current assortment for each category at each store. As discussed above, for each category/store combination in a run group, the Assortment Recommender starts with the current assortment and makes changes to it to increase the chosen assortment metric. For each category/store combination, the system takes the SKUs that were selling in the store during the last available week of historical data.

  The weekly sales-units rate of each SKU in the current assortment. This is calculated through an average over the most recent four weeks of historical data.

- The price of each SKU in the current assortment. Historical price data is not used for this, but instead the total revenue over the most recent four weeks of historical data is divided by the total sales units over the same four weeks. This provides an average historical price, based on the last four weeks of historical data.

- The gross profit of each SKU in the current assortment. Historical price or cost data is not used; instead, an average is taken similarly to the price calculation. The total gross profit is taken over the most recent four weeks of historical data and divided by the total sales units over the same four weeks.

The sales-units rate, price, and gross profit are required in order to support the possible assortment metrics.

Notice that in addition to data about the current assortment, the Assortment Recommender requires data about possible SKUs that it can swap into the assortment, since otherwise, in the case of keeping the assortment sizes the same or expanding the assortment, no recommendations would be possible. (The case of decreasing assortment sizes is discussed separately below.) For each category/store combination in a run group, the system calculates the following for the possible SKUs to be swapped into an assortment:

SKU Data

- For a category/store combination, the set of new SKUs is the SKUs that are in the current assortments of other stores but not in the current assortment of this store. (See above for how the current assortment of a store is determined.)

- The price of the new SKU is the average of the prices in the stores in which it is part of the current assortment. The price at each store in which it is selling is determined as described in "Required Data".
- The gross profit of the new SKU is the average of the gross profits in the stores i which it is part of the current assortment. The gross profit at each store in which it is selling is determined as described in "Required Data".
- Assigning the sales units rate of the new SKU is the trickiest to handle, since the sales-units rate of the new SKU must be forecast based on the assumption that it is selling at a store where it may not have sold before. Here, the Assortment Recommender identifies like items of the new SKU among SKUs that are currently in the assortment at the store, and from the like items it makes a forecast of the new SKU's sales-units rate at this store. Identifying the like items is done through the use of similarities. For more information about similarities, see the section "The Role of Attributes in Calculating Similarities".

If the Assortment-Size Change parameter is set to a negative value, then it is possible to decrease the size of the assortment without having data about possible SKUs to swap into the assortment, as in this situation the Assortment Recommender would simply be finding SKUs to delete from the assortment while still maximizing the chosen assortment metric.

Halo Effects

As mentioned above, the Assortment Recommender accounts for halo effects when calculating the selected assortment metric. This calculation uses the output of the Affinity Analysis. AA determines halo effects at the sub-class level. That is, sub-class A of one category has a halo effect on sub-class B of another category, meaning some significant fraction of the people who purchase in a SKU in A also purchase a SKU in B. Suppose the Assortment Recommender is running for a particular category C. The Assortment Recommender, when it considers putting a SKU C into the assortment, adjusts upward the amount of the assortment metric that C brings in order to include the halo effect. For example, suppose C is a SKU in sub-class A, and sub-class A brings a halo lift of 10 percent to sub-class B. If the metric the Assortment Recommender is maximizing is sales units, then to the sales units U of C itself, the Assortment Recommender adds sales units of 0.1U to represent the sales units of B bought by purchasers of C. Similarly, if the chosen metric is revenue, then to the revenue brought by C itself, 0.1U times the average price of SKUs in sub-class B is added. The average price of SKUs in sub-class B is calculated by a weighted average of prices, with the weights being the weekly sales-units rates.

The handling of gross profit is similar to the handling of revenue, except that a weighted average of gross profits of B is used instead of the weighted average of prices.

The above discussion involves adding SKU C to the assortment, but the same discussion holds if the Assortment Recommender is removing SKU C from the assortment.

Troubleshooting

Several conditions can prevent the Assortment Recommender from producing recommendations for specific category/store combinations. When any of these occur, the Assortment Recommender will not produce an error but will simply not produce recommendations for the particular category/store combination.

- The DT application was not run for a particular category, or the category does not have an approved DT version associated with it. This means the category does not have results from the DT application, and without those results, it is not possible for the assortment recommender to run since it cannot account for demand-transference effects. In this case, the category will not receive any assortment recommendations regardless of store.

- The Assortment Recommender was not able to find any new SKUs for the particular category/store combination. In the above description about finding the set of new SKUs, it is possible that the procedure does not find any new SKUs at all, perhaps because all of the stores are assorted identically at that point in time for this particular category. This may happen with categories that are less important to the retailer, so that the retailer does not see any benefit in tailoring the assortment within each store.
- The Assortment Recommender was not able to find any assortment changes that result in an increased assortment metric. This can happen if:
  - There were no new SKUs available (see previous item).
  - The number of new SKUs available was very small.
  - The run-group parameters for the category are too restrictive. For example, too many SKUs are listed as must-keep SKUs or the min-keep percentage was set too high (greater than 80 percent).
- The Assortment Recommender may run, but without using halo effects if the halo effects are not available. For example, the AA may not have run or may not have produced halo effects for the category in question.

11

Size Profiles

This chapter describes the Size Profiles (SP) Cloud Service module.

Overview

Size Profiles (SP) is a module under AIF Profile Sciences and is used to estimate the distribution of demand across different sizes (size profile) for different merchandise and location levels.

Size profile is estimated at different levels of merchandise and location. The lowest level of estimation is style-color (for merchandise) and store (for location). Size profiles may also be estimated at higher levels of aggregation on both the merchandise and location dimensions (for example, at subclass-store or style/color-store cluster), depending on the retailer's requirements. Irrespective of the level of estimation, the final output for size profiles is at the lowest level (style-color/store), and size profiles are shown at the lowest level in the UI.

Data Requirements

Size Profiles relies on the following data elements. These must be provided via text files, which are then loaded.

Hierarchy Data

The three types of hierarchies are Location Hierarchy, Merchandise Hierarchy, and Calendar Hierarchy.

- Location Hierarchy. An example of location hierarchy is: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE.
- Merchandise Hierarchy. An example of merchandise hierarchy is as follows: CHAIN ' COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE ' COLOR ' SIZE (SKU). The Size Profiles application is used for fashion apparel. Therefore, it is expected that the extended merchandise hierarchy is provided, that is, Style and Color are provided through the relevant interfaces.

  The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_DS interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_DHS interface. In addition, there must be consistency between the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_DS is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy.

  Here are the specific fields:

  PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)

PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)

PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699).

The value of PROD_NUMs is the same as the value in the W_PRODUCT_DS.PROD_NUM interface.

Here is what this looks like:

Table 11–1 Hierarchy Data

PROD_NUM	PRODUCT_ATTR13_NAME	PRODUCT_ATTR14_NAME	PRODUCT_ATTR15_NAME
STYLE_PROD_NUM	STYLE_PROD_NUM		
STY/COL_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	
STY/COL_SIZ_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	STY/COL_SIZ_PROD_NUM

Note: For Size Profiles Cloud Services, the Extended Product Hierarchy must be loaded.

Note: The mapping of columns and the example mentioned above for product hierarchy can be different, depending on the source of the data. For example, if the data is loaded from RMS Cloud, the mapping will be as follows:

- PRODUCT_ATTRPRODUCT_ATTR14_NAME = sku ID (a.k.a. style/color/size)13_NAME = sku parent ID (a.k.a.style)
- PRODUCT_ATTR14_NAME = sku ID (a.k.a. style/color/size)
- PRODUCT_ATTR16_NAME = differentiator ID (for example color, or any differentiator that is used in-between style and sku).

- Calendar Hierarchy. This is one of the core hierarchies. The retailer can specify the calendar depending on their business requirements (for example, fiscal calendar).

Sales Data

Historical sales data is required at the sku-store-week level. If the sales transaction data is provided, it will be aggregated by the application to the sku-store-level. It is necessary that the sales transaction data be provided at the lowest level of product hierarchy (that is, the level in RSE_HIER_LEVEL that has LEAF_NODE_FLG = 'Y' for HIER_TYPE_ID = 3).

Alternatively, data can be provided directly at this level using the RSE_SLS_PR_LC_WK_STG interface. When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Inventory Data

Two methods are available for generating size profiles. For the Optimization method, historical inventory must be provided at the sku-store-week level. To generate profiles using the Normalize method, inventory data is not required.

Product Images Data

Product images that are available on a customer-hosted web server can be viewed in the Size Profiles UI. The W_RTL_PRODUCT_IMAGE_DS.dat interface contains a column called PRODUCT_IMAGE_ADDR, which can contain the full URL to an image of the product. This URL must be in the following format:

http[s]://servername[:port]/location/filename.extension

For example:

PRODUCT_IMAGE_NAME = imagename.png

PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png

PRODUCT_IMAGE_DESC= Short description of the image

The Size Profiles application running in the cloud does not directly access these images, so there is no need to expose these images outside of the customer's firewall. As long as the user of the SP application has access to the URL while running the SP application, then the user's web browser will be able to resolve the URL and retrieve the images for display when the user chooses this option. The images must be in a file format that the web browser can display. Since the images shown in the UI are small, these images do not need to be high quality images. The size of the image files will affect the time it takes to render them.

Season

The selected season for the run is used in two ways:

The historical data period is set based on the start/end date of the selected season, one year ago. In addition, size profiles are generated for products that are selected for the run *and* are mapped to the selected season.

The season definition can be loaded through the following interfaces: W_RTL_SEASON_D and W_RTL_PHASE_D. In addition, the W_DOMAIN_MEMBER_LKP_TL should have data for DOMAIN_CODE of SEASON and PHASE. The mapping between items and season is also required and can be loaded through W_RTL_SEASON_PHASE_IT_D.

Size Range and Sub Size Range Definition

All available size ranges and sub size ranges must be provided using the SPO_SZ_RANGE_ STG interface.

SPO_SZ_RANGE_STG sample data:

Table 11–2 Size Range - Sub Size Range Definition

SZ_RANGE_ EXT_KEY	SZ_RANGE_ NAME	SZ_RANGE_ LENGTH	SUB_SZ_ RANGE_EXT_ KEY	SUB_SZ_ RANGE_ NAME	SUB_SZ_ RANGE_ LENGTH	COPY_OF_ SZ_RANGE
100	Active apparel top	4	1001	S/M/L/XL	4	Y
100	Active apparel top	4	1002	S/L/XL	3	N
100	Active apparel top	4	1003	S/M/L	3	N
100	Active Apparel top	4	1004	M/L/XL	3	N
100	Active apparel top	4	1005	S/M/XL	3	N

Table 11–2 (Cont.) Size Range - Sub Size Range Definition

SZ_RANGE_EXT_KEY	SZ_RANGE_NAME	SZ_RANGE_LENGTH	SUB_SZ_RANGE_EXT_KEY	SUB_SZ_RANGE_NAME	SUB_SZ_RANGE_LENGTH	COPY_OF_SZ_RANGE
100	Active apparel top	4	1006	L/M	2	N
100	Active apparel top	4	1007	S/M	2	N
100	Active apparel top	4	1008	S/XL	2	N
100	Active apparel top	4	1009	L/XL	2	N

Size Definition and Mapping Between Size and Sub Size Range

All sizes across all merchandises and their mapping to sub size ranges must be provided using the SPO_SIZE_STG interface.

SPO_SIZE_STG sample data:

Table 11–3 Size Definition and Mapping

SIZE_EXT_KEY	SIZE_EXT_CODE	SIZE_RANK	EXPECT_KINK_FLG	SZ_RANGE_EXT_KEY	SUB_SZ_RANGE_EXT_KEY
1	SZ1	1	N	100	1001
2	SZ2	2	N	100	1001
3	SZ3	3	N	100	1001
4	SZ4	4	N	100	1001
1	SZ1	1	N	100	1002
1	SZ1	3	N	100	1002
4	SZ4	4	N	100	1002
1	SZ1	1	N	100	1003
2	SZ2	2	N	100	1003
3	SZ3	3	N	100	1003
2	SZ2	2	N	100	1004
3	SZ3	3	N	100	1004
4	SZ4	4	N	100	1004

Mapping Between SKU and Size

The mapping between sku and size must be provided using the SPO_PROD_SIZE_STG interface.

SPO_PROD_SIZE_STG sample data:

Table 11–4 Mapping Between SKU and Size

PROD_HIER_TYPE_NAME	PROD_EXT_KEY	SIZE_EXT_KEY
Extended Product Hierarchy	78715077631	6
Extended Product Hierarchy	78715077648	7

Table 11–4 (Cont.) Mapping Between SKU and Size

PROD_HIER_TYPE_NAME	PROD_EXT_KEY	SIZE_EXT_KEY
Extended Product Hierarchy	78715077655	8
Extended Product Hierarchy	78715077662	9
Extended Product Hierarchy	78715156367	1
Extended Product Hierarchy	78715156374	2
Extended Product Hierarchy	78715156381	3
Extended Product Hierarchy	78715991500	1
Extended Product Hierarchy	78715991517	2
Extended Product Hierarchy	78715991524	3
Extended Product Hierarchy	78715991531	4

Mapping Between Style-Color/Store and Sub Size Range

The mapping between style-color/store and sub size range must be provided using the SPO_SUB_SZ_RANGE_PRDLOC_STG interface.

SPO_SUB_SZ_RANGE_PRDLOC_STG sample data:

Table 11–5 Mapping - Style-Color/Store to Sub Size Range

SUB_SZ_RANGE_EXT_KEY	PROD_HIER_TYPE_NAME	PROD_EXT_KEY	LOC_HIER_TYPE_NAME	LOC_EXT_KEY
1001	Extended Product Hierarchy	86323224302	Location Hierarchy	732
1001	Extended Product Hierarchy	86323224302	Location Hierarchy	273
2003	Extended Product Hierarchy	86323226802	Location Hierarchy	778
2003	Extended Product Hierarchy	86323226802	Location Hierarchy	672
2003	Extended Product Hierarchy	86323226802	Location Hierarchy	679
2007	Extended Product Hierarchy	86323226803	Location Hierarchy	778
2007	Extended Product Hierarchy	86323226803	Location Hierarchy	773

Note: As an alternative to SPO_SZ_RANGE_STG, SPO_SIZE_STG, SPO_PROD_SIZE_STG and SPO_SUB_SZ_RANGE_PRDLOC_STG, the required data for size, size ranges, and the mappings explained above can be provided through W_RTL_DIFF_GRP_D and W_RTL_DIFF_GRP_D_TL. It is also necessary to set the value of SPO_DIFF_TYPE in RSE_CONFIG to be the same as the diff_type that is in w_rtl_diff_grp_d for size attribute.

In addition to these two interfaces, it is expected that the column prod_attr_grp_id will be populated in W_RTL_ITEM_GRP1_D for the PROD_GRP_TYPE that is associated with Size.

For the jobs that must be run to load the size, size range, and data, see "Batch and Ad-Hoc Jobs".

Note: It is necessary that all attributes, including the Size attribute, are provided at the lowest level of product hierarchy (that is, the level in RSE_HIER_LEVEL that has LEAF_NODE_FLG = Y).

Configurations

The Size Profiles science algorithm uses various configurations. These configurations can be viewed and overridden using the Strategy & Policy Management under Control & Tactical Center. In this dashboard, navigate to Manage System Configurations, select ALL for application and RSE_CONFIG for table and filter the table by APPL_CODE = 'SPO'. For more details, see the "Control & Tactical Center" chapter of the *Oracle Retail AI Foundation Cloud Services User Guide*.

The main configurations are shown in Table 11–6. These configurations are global and are applied to all runs. There are five configurations that can also be adjusted at the run level:

SPO_KINK_INNER_RATIO, SPO_KINK_OUTER_RATIO, SPO_CORR_LOW_SELL_FLG, SPO_LOW_SELL_AVG, SPO_SZ_PRO_GEN_METHOD.

These five configurations are available in the UI and can be overridden when creating a new run.

Table 11–6 Configurations

PARAM_NAME	PARAM_VALUE	DESCR
LOC_CHAIN_LVL_ID	2	To evaluate profiles by comparing them with chain level profile.
MIN_SKUS_PRNT_STR_WK	2	Minimum SKUs for determining SKU-Parent/Store Eligibility.
MIN_SLS_PRNT_STR_WK	4	Minimum Sales units for determining SKU-Parent/Store Eligibility.
MIN_WKS_PRNT_STR_WK	2	Minimum Weeks for determining SKU-Parent/Store Eligibility.
SPO_AVG_WK_SLS_PRNT	0.08	The default value for average weekly sales of sku-parent.
SPO_CORR_LOW_SELL_FLG	Y	The default value to indicate Yes or No, whether to Correct Low Sellers flag.
SPO_DIFF_ORG_CORR_PROF	1	This is a configurable parameter that measures the difference between the original and corrected profile for each size.
SPO_ELIG_FRACTION_SKU_STR	0.1	Eligible Fraction SKU-Store for determining SKU-Store Eligibility.
SPO_KINK_INNER_RATIO	1.1	The default value for kink inner ratio.
SPO_KINK_OUTER_RATIO	1.03	The default value for kink inner ratio.
SPO_LOC_HIER_PROCESSING_LVL	6	Level Identifier for the level of the Location Hierarchy at which we process Size Profile. Default 6 is to process at STORE Level.
SPO_LOC_HIER_TYPE	2	Hierarchy Type Identifier for the Location Hierarchy Type.
SPO_LOW_SELL_AVG	0.08	Average weekly sale units across a season, below which deems a SKU (Size) to be a Low Seller.
SPO_MIN_INV_SALES	1	Minimum Inventory and Sales for determining start and end dates.

Table 11–6 (Cont.) Configurations

PARAM_NAME	PARAM_VALUE	DESCR
SPO_MIN_INV_SALES_SKU_STR_WK	1	Minimum Inventory and Sales for determining SKU/Store/Week Eligibility.
SPO_MIN_SALES	0	Minimum Sales for determining start and end dates.
SPO_MIN_SALES_SKU_STR	1	Minimum Sales SKU-Store for determining SKU-Store Eligibility.
SPO_MIN_SALES_SKU_STR_WK	0	Minimum Sales for determining SKU/Store/Week Eligibility.
SPO_MIN_SEASON_LEN_SKU_STR	1	Minimum Season Length SKU-Store for determining SKU-Store Eligibility.
SPO_MIN_SKU_STR_WK	2	Minimum SKUs for determining SKU-Parent/Store Eligibility.
SPO_PCT_MAX	0.05	Percent Maximum for determining start and end dates.
SPO_PCT_MAX_SKU_STR_WK	0	Percent Max SKU-Store-Week for determining SKU/Store/Week Eligibility.
SPO_PCT_SZ_RNG_PRNT_STR_WK	0.41	Percent sub size range for determining SKU-Parent/Store Eligibility.
SPO_PROD_HIER_PARENT_LVL	8	Level Identifier for the level of the Product Hierarchy of the SKU-Parent. Default 8 is the STYLE-COLOR Level.
SPO_PROD_HIER_PROCESSING_LVL	9	Level Identifier for the level of the Product Hierarchy at which we process Size Profile. Default 9 is to process at SKU Level.
SPO_PROD_HIER_TYPE	3	Hierarchy Type Identifier for the Product/Merchandise Hierarchy Type.
SPO_PROF_EXP_LOC_LVL	6	Mark USER_BY_AIP=Y Where active Season Size Profiles created at Store level in Export DB view.
SPO_PROF_EXP_MERCH_LVL	6	Mark USER_BY_AIP=Y Where active Season Size Profiles created at Subclass level in Export DB view.
SPO_SKU_PRNT_STR_WK	2	Post Processing threshold: Min number of SKU-parent/store/weeks with at least one valid sale.
SPO_SP_CHART_COLOR_LVL_ID	8	Color level id for displaying color level items in the size profiles chart.
SPO_SP_CHART_DISTRICT_LVL_ID	5	District level id for filtering store level items in the size profiles chart.
SPO_SP_CHART_STYLE_LVL_ID	7	Style level id for filtering color level items in the size profiles chart.
SPO_SZ_PRO_GEN_METHOD	Optimization	The default value for size profile generation method.
SPO_TOT_UNIT_SLS	2	Post Processing threshold that is applied to determine the number of valid sizes (i.e., the sizes that meet this minimum sales threshold).

Table 11–6 (Cont.) Configurations

PARAM_NAME	PARAM_VALUE	DESCR
SPO_TOT_UNIT_SLS_PCT_SIZES	0.4	Post Processing threshold that is applied to the number of valid sizes in a sub size range. This filter is applied in connection with SPO_TOT_UNIT_SLS. In this example, 40% of the sizes must have two units or more for the profile to be considered valid.
SZ_PROF_CORR_THRESHOLD	0.75	The size profiles at the lower level of escalation are compared to size profiles at a higher level (e.g., at Chain level). If the correlation between the two profiles is more than this value, the size profile at the lower level of merchandise/location hierarchy is considered valid.
SPO_DIFF_TYPE	SIZE	Diff Type value to be used in loading data from RI to SPO. **Note**: This value must be the same as the diff_type that is in w_rtl_diff_grp_d for size attribute.

Data Output

The output of Size Profiles is reviewed and submitted by the user using the Size Profiles UI. The submitted profiles can be obtained from the export interface file spo_size_profile.csv, which is available under ORASE_WEEKLY_extract.zip.

The output contains the size profiles at the style-color/location as well as profiles at higher levels of merchandise (for example, class/location). In addition to the columns that indicate the merchandise-location-size dimension and the percentage of each size in the profile, there are two flags in the export file (USED_BY_AIP and USED_BY_RDF) that indicate which rows must be extracted by Retail Demand Forecasting (RDF) and Assortment Planning Cloud Service (AP CS).

In addition to spo_size_profile.csv, the ORASE_WEEKLY_extract.zip folder contains two other files (d1itpt.01 and spo_gid_label.txt) for integration with Allocation. For details about the content and format of these files, please refer to the documentation for Oracle Retail Allocation Cloud Service.

Batch and Ad-Hoc Jobs

The following batch and ad-hoc jobs are responsible for loading the size, size range, and season data into SPO.

Note: When running ad-hoc jobs for loading size and size range data, they must be run in the following sequence:

SPO_SIZE_LOAD_JOB; SPO_PROD_SIZE_LOAD_JOB; SPO_SIZE_RANGE_LOAD_JOB; SPO_SUB_SZ_RANGE_PRDLOC_LOAD_JOB

Table 11–7 Data-Loading Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_SEASON_ LOAD_START_JOB	SPO_SEASON_ LOAD_START_JOB	rse_process_state_ update.ksh	SPO_SEASON_ LOAD_PROCESS Start	SPO_Batch
SPO_SEASON_ LOAD_JOB	SPO_SEASON_ LOAD_JOB	spo_season_load.ksh		SPO_Interfac-es||SPO_ Interfac-es_CNE
SPO_SEASON_ LOAD_END_JOB	SPO_SEASON_ LOAD_END_JOB	rse_process_state_ update.ksh	SPO_SEASON_ LOAD_PROCESS End	SPO_Batch
SPO_SIZE_RANGE_ LOAD_START_JOB	SPO_SIZE_RANGE_ LOAD_START_JOB	rse_process_state_ update.ksh	SPO_SIZE_RANGE_ LOAD_PROCESS Start	SPO_Batch
SPO_SIZE_RANGE_ LOAD_JOB	SPO_SIZE_RANGE_ LOAD_JOB	spo_sz_range_load.ksh		SPO_Interfac-es||SPO_ Interfac-es_CNE
SPO_SIZE_RANGE_ LOAD_END_JOB	SPO_SIZE_RANGE_ LOAD_END_JOB	rse_process_state_ update.ksh	SPO_SIZE_RANGE_ LOAD_PROCESS End	SPO_Batch
SPO_SIZE_LOAD_ START_JOB	SPO_SIZE_LOAD_ START_JOB	rse_process_state_ update.ksh	SPO_SIZE_LOAD_ PROCESS Start	SPO_Batch
SPO_SIZE_LOAD_ JOB	SPO_SIZE_LOAD_ JOB	spo_size_load.ksh		SPO_Interfac-es||SPO_ Interfac-es_CNE
SPO_SIZE_LOAD_ END_JOB	SPO_SIZE_LOAD_ END_JOB	rse_process_state_ update.ksh	SPO_SIZE_LOAD_ PROCESS End	SPO_Batch
SPO_PROD_SIZE_ LOAD_START_JOB	SPO_PROD_SIZE_ LOAD_START_JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE_ LOAD_PROCESS Start	SPO_Batch
SPO_PROD_SIZE_ LOAD_JOB	SPO_PROD_SIZE_ LOAD_JOB	spo_prod_size_load.ksh		SPO_Interfac-es||SPO_ Interfac-es_CNE
SPO_PROD_SIZE_ LOAD_END_JOB	SPO_PROD_SIZE_ LOAD_END_JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE_ LOAD_PROCESS End	SPO_Batch
SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_START_JOB	SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_START_JOB	rse_process_state_ update.ksh	SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_PROCESS Start	SPO_Batch
SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_JOB	SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_JOB	spo_sub_sz_range_ prdloc_load.ksh		SPO_Interfac-es||SPO_ Interfac-es_CNE
SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_END_JOB	SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_END_JOB	rse_process_state_ update.ksh	SPO_SUB_SZ_ RANGE_PRDLOC_ LOAD_PROCESS End	SPO_Batch

The following jobs are for exporting the outputs.

Table 11–8 Output Exporting Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_EXPORT_ START_JOB	SPO_EXPORT_ START_JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE_ LOAD_PROCESS Start	SPO_Batch
SPO_EXPORT_PREP_ JOB	SPO_EXPORT_PREP_ JOB	spo_export_prep.ksh		SPO_Export
SPO_SZ_PROF_ EXPORT_JOB	SPO_SZ_PROF_ EXPORT_JOB	spo_sz_prof_export.ksh	#Sys-Opt.SPO_ HOME/data/outfile/spo _sz_prof_export.csv	SPO_Export
SPO_SZ_PROF_ EXPORT_ADHOC_ JOB	SPO_SZ_PROF_ EXPORT_ADHOC_ JOB	spo_sz_prof_export.ksh	#Sys-Opt.SPO_ HOME/data/outfile/spo _sz_prof_export.csv	SPO_Export
SPO_EXPORT_END_ JOB	SPO_EXPORT_END_ JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE_ LOAD_PROCESS End	SPO_Batch
SPO_EXPORT_PREP_ ADHOC_JOB	SPO_EXPORT_PREP_ ADHOC_JOB	spo_export_prep.ksh		SPO_Export
SPO_ALLOC_ EXPORT_JOB	Export data for allocation	spo_alloc_export.ksh		
SPO_ALLOC_ EXPORT_ADHOC_ JOB	Ad hoc job to export data for allocation	spo_alloc_export.ksh		SPO_Export
SPO_ALLOC_GID_ FILE_EXPORT_JOB	Generate a GID file	spo_alloc_gid_file_ export.ksh		
SPO_ALLOC_GID_ FILE_EXPORT_ ADHOC_JOB	Ad hoc job to generate a GID file	spo_alloc_gid_file_ export.ksh		SPO_Export

12

Offer Optimization

This chapter describes the Promotion and Markdown Optimization and Offer Optimization Cloud Services. In general, throughout the document the term "Offer Optimization" is used to refer to these two combined services. However, when a specific functionality is only available in one of these services, the complete service name is used.

Overview

Offer Optimization (OO) is used to determine the optimal pricing recommendations for promotions, markdowns or targeted offers. Promotions and markdowns are at the location level or a price zone. Targeted Offers can be specific to each customer and not just to a location. Pricing recommendations contain answers to the following questions: Which items? When (timing)? and How deep and Who (segments)? Promotion and Markdown Optimization caters to the retailers who are interested in only promotions and markdowns. Offer Optimization not only provides promotions and markdowns but also targeted offers that are specific to each customer segment. In order to use targeted offers, the retailer must provide customer-linked sales transactions data; to use promotions and markdowns, the retailer does not necessarily have to provide customer-linked sales transaction data.

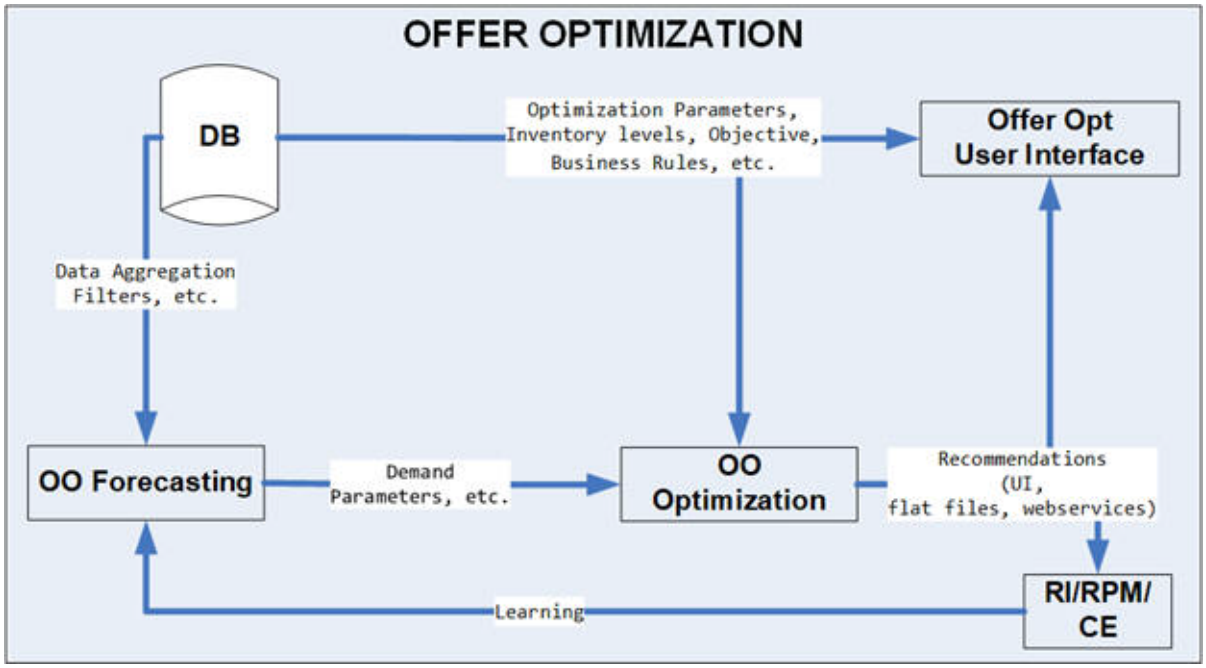
Certain aspects of promotions, markdowns, and targeted offers are important levers for managing the inventory over the life cycle of the product. The application helps in the following:

- Bring inventory to the desired level, not just during the full-price selling period.
- Maximize the total gross margin amount over the entire product life cycle.
- Assess in-season performance.
- Provide updated recommendations each week. This facilitates decision-making that is based on recent data, including new sales, inventory, price levels, planned promotions, and other relevant data.
- Provide targeted price recommendations at the segment-level.

Figure 12–1 shows the conceptual flow of different components in Offer Optimization. Oracle Retail AI Foundation Platform Cloud Service and Retail Insights Cloud Service Suite is the core data foundation layer that consumes the retailer data. Offer Optimization Forecasting analyzes and mines historical data (along with other data sources) using machine learning algorithms for developing a predictive model to provide the forecasting inputs to the Optimization Algorithm. The Optimization Algorithm obtains inputs such as objectives, budgets, and business rules from the UI, along with optimization parameters from AIF and RI. The algorithm analyzes the feasible price paths efficiently and generates price recommendations. These recommendations can be viewed in the Offer Optimization UI or can be exported to the price execution systems such as Price Management (PM) or Customer Engagement (CE). In addition, a feedback loop from the price execution systems can help the

OO Forecasting component to determine how the offers are performing and adjust the next set of offers based on the sales performance or the response rate. OO supports two kinds of runs, called "ad hoc runs" and "batch runs." Batch runs are scheduled to run automatically at regular intervals (for example, every week). Each batch run, using latest sales data and inventory levels, updates the parameters and budgets and produces the price recommendations. An analyst can review the results and further accept, reject, or override the price recommendations for each item. Once the analyst finishes the review, the item's recommendation status can be changed to Reviewed. A reviewed recommendation is sent to the buyer for approval. If the user with the buyer role likes the recommendations, that user can submit or approve the recommendation. At this point, the recommendation status is changed to Submitted or Approved. This indicates that the price recommendations will be sent to an export interface as well as to a price execution system such as RPM with the respective statuses.

Figure 12–1 Offer Optimization



An optimization can be carried out at the configured processing (or run) location or the price-zone, merchandise level, and calendar level. The user can configure the optimization to use either the price-zone or a node in the location hierarchy (but not both in the same instance). Once the optimization is complete, the recommendations can be generated at a lower level than the processing level (called recommendation levels for merchandise level and location or price-zone level). The location and merchandise level can be any level in the location hierarchy and merchandise hierarchy, respectively. Alternatively, price-zone can be used to define a set of stores (and/or online locations) and items. An example of a price zone is women's apparel in all university-based stores grouped into one price-zone. The usual levels for the run are Region or Price-zone, Department, and Week, and for the recommendation, the levels are Region or Price-zone and Style/Color or SKU (Style/Color/Size). If Targeted Offers is available, then the recommendations will also be generated at the Customer-Segment level, along with location or price-zone and merchandise level. The Promotions and Markdowns are always at the location or price-zone and merchandise level.

The optimization is set up as follows:

- Optimization is done at the run's merchandise and location setup levels. For example, if the run merchandise and location levels are Department and Region, then each optimization job is at the Department-Region level.

- Inventory is rolled to the desired recommendation level for merchandise. Further, the inventory is aggregated across all the locations to the run's location level.
- Price recommendations are generated at the configured recommendation levels for merchandise, location or price-zone, and calendar level and the customer segment level.

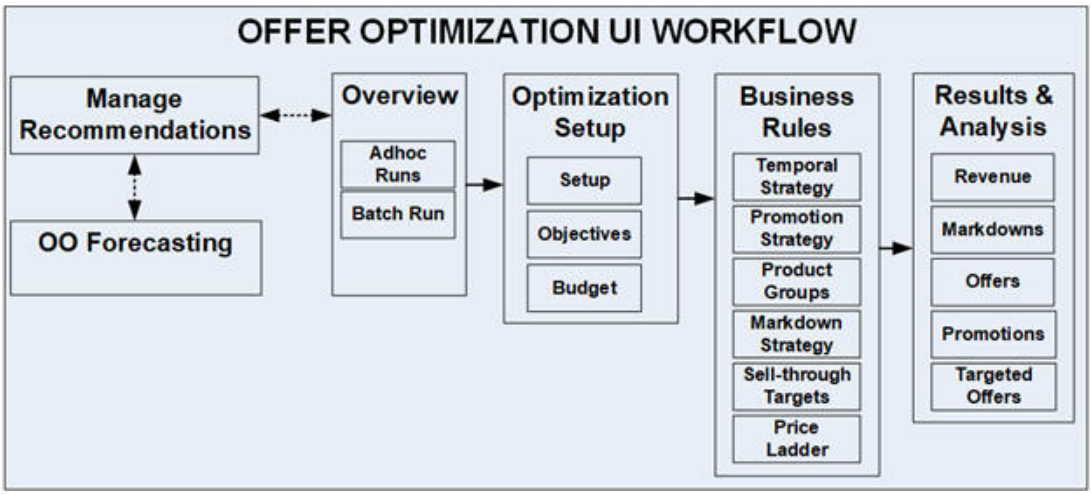
  For example, if the run location level is Price-zone, the run merchandise level is Department, the run calendar level is Week, and the recommendation level for merchandise is Style/Color, then the promotion and markdown recommendations are generated at Price-zone, Style/Color, Week and Price-zone, Style/Color, Week and Customer Segment for Targeted Offers.

Figure 12–2 shows an overview of the OO UI workflow, which consists of the following:

- Overview. This is the dashboard for the OO runs. In this tab, you can see a list of all existing runs, along with details that describe each run. The list includes runs created by other users, which you can open in read-only mode. You can create a run, copy a run, open a run, or delete a run. The runs overview has three components, Search, Run Status Tiles, and the Table of Runs.

  You can click an existing run or create a new run. Each run is opened in a run tab. The title of the run tab displays "Offer Optimization: <Run ID>". This is the main tab where you can specify the business rules and goals for the run. It provides a series of three stages that you progress through in order to set up, run, and analyze the results of the optimization run.
- Setup. Used to pick a season, location or price zone and department. It is also used to select the objectives and specify the budgets.
- Rules. Used to view or change business rules.
- Results. Used to view results and accept, reject, or override recommendations, and recalculate.
- Forecasting. These screens provide the user with the ability to review the way in which different components of forecast (for example, baseline, seasonality) affect the price recommendation for an item.
- Manage Recommendations. This screen is used to accept, reject, or override and recalculate recommendations. It also allows the user to review, submit, or approve recommendations so that they can be sent to a price execution system. In production, this screen is the starting point and is used most regularly, since the user can send recommendations from batch runs for execution or visit forecasting screens to review how item forecast looks, and finally, decide whether to create a new ad hoc run.

Figure 12–2 Offer Optimization UI Workflow



The goal of the implementation is to set up and configure an instance to generate optimal price recommendations that satisfy the retailer's business requirements. The implementation configures the application so that the batch runs complete successfully in a timely fashion and produce valid promotions, markdowns, and forecast recommendations that meet the retailer's requirements. The main implementation tasks involve configuring the following:

- The roles and permissions assigned to users.
- The loading of retailer data.
- The configuration parameters.
- The demand parameters, such as seasonality and price effects, that are used to determine optimal promotion, markdown, and targeted recommendations.
- The business rules that determine constraints that the application takes into account during the optimization process.

Project Planning

This section provides a high-level project planning for implementation by different buckets: functional, data interfaces, forecasting and optimization.

Functional

- Understand the existing business requirements for pricing; and proposed process changes with OO.
- Verify the functionality that the customer would like to leverage from OO and identify whether the customer can provide the necessary data.
- Review and identify the business rules (and strategy sets) with the customer.
- Identify the desired flex facts that the customer would like to see in the Manage Recommendation Screen.
- Identify the desired reports for the customer.
- Log and revisit any open requirements.
- Users and user roles, data security/filtering.

Data Interfaces

- Overview. Confirm the list of OO interfaces to be loaded for the customer
- Foundational Data. Decide on the business date and load the Calendar data. Plan for partitions.
- Foundational Data. Decide on the merchandise and location hierarchy levels; map it to the levels in the OO hierarchies.
- Foundation Data. Identify the fields in each interface and identify or map to a source in the customer's systems
- Foundational Data. Convert history for sales transactions
- Foundational Data. Convert history for inventory positions.
- Foundational Data. Convert other positional data (costs, prices)
- Foundational Data. Decide on whether to use Price Zones and load the price zone data.
- Validation. Load a small sample of data (for example,1 month) to verify data is flowing in correctly. Use Innovation Workbench or Data Visualizer for verifications.
- Full historical load. Plan for full load of history.
- Offer Optimization Interfaces.

Forecasting Parameters

- Setup. Create a forecast run type and relevant levels for forecast or recommendations
- Setup. Create a forecast run(s) with as many configurations as needed.
- Data Aggregation. Verify that the data aggregation is completed.
- Parameters. Review the forecasting parameters.
- Validation. Assess the forecast quality of the parameters and iterate on forecast runs as needed. Prepare comparison reports using Data Visualizer.

Optimization

- Setup. Configure all the relevant parameters (for example, warehouse virtual allocation algorithm).
- Setup. Verify the business rules and strategies.
- Ad hoc run. Optimize an ad hoc run for optimization for selected merchandise or locations.
- Validation. Review the recommendations and for the quality of the business rules and forecast parameters. Revisit the business rules and forecasting as necessary.
- Ad hoc batch job. Execute an ad hoc batch job to verify the runs are executed successfully for all merchandise/locations.

Walkthrough

Most of the AIF Platform's foundation data is pushed in two-step process. Any additional OO-specific data is directly pushed into the AIF Platform. First, data is loaded, using CSV and W_ interfaces, into Retail Analytics Platform Foundation Data, using appropriate jobs. Then, RADM_REFRESH_JOB must be run to refresh the table stats before any AIF job is run. Second, appropriate AIF jobs are used to push data into AIF. At the time of implementation, the user will only use ADHOC jobs, not any batch jobs. Batch jobs (described in "POM Jobs") are

necessary to put the system on a batch schedule. Third, OO-specific data (that is not available in RI interfaces) is directly pushed into the RSP.

The following sections assumes that the user has obtained access to POM. The POM UI URL is something like this: <host>/POMJetUI. If the user cannot access the POM UI, contact the administrator to obtain the relevant access/user roles. In addition, the user must have access to Innovation Workbench and/or Data Visualizer in order to query or visualize the data in order to verify that the data loaded matches the desired expectations.

Another point to note is that some of the jobs require relevant configuration parameters to be specified with client-specific values. If incorrect configuration values are used, a job may run without errors, but will not produce the desired data in the target tables. If the client or implementation team wants to view the progress of the job or any errors in order to file an SR, then RSP provides database logging in a table called RSE_LOG_MSG. Logging can be enabled only with a service request from the client/implementation partner to the support team.

Although this walk-through provides all the necessary steps, the user may want to refer to the relevant section in this guide for more details (for example, the specific fields in an interface).

RAP foundation data (CSV and W_ interfaces)

These interfaces provide the foundation data for the Retail Science Platform. Execute and verify that all the RAP foundation data has been loaded. For details, the RAP Implementation Guide:
https://docs.oracle.com/en/industries/retail/retail-analytics-platform/21.0/rapig/data-load-init-batch-proc.htm#data-load-init-batch-proc-F30BF774.

For Offer Optimization, this includes:

1. **Product Hierarchy**. When Style, Style Color must be loaded, appropriate columns must be populated.
2. **Location Hierarchy**.
3. **Price Zone Groups and Price Zones**. It is not required to provide price zones for OO, but if the user intends to do pricing by price zones, then these files are required. Price Zones are clusters of stores and Price Zone Groups allow the user to map a merchandise node or set of items to a Price Zone Group.
4. **Sales**. It is best to load at least 20 months of the data to obtain a good signal in the model training.
5. **Inventory**. It is a critical data element for OO, for, without inventory, the OO runs would fail.
6. **Inventory Receipts**. This data is used to identify when the item started selling. This data must be provided for OO.
7. **Customer Segments**. Even when the retailer is not planning to load or use customer segments, a dummy record. is required. RI provides the ability to insert a dummy record, and it is not required for the client or implementator to provide these interfaces: W_ PARTY_PER_DS and W_RTL_CUSTSEG_DS.
8. **Price and Cost**. This data must contain the price and cost for every product and location combination that had a sale in the sales history. At least a record must be provided for TRAN_TYPE 0 for every product and location combination so that OO can correctly identify the ticket price, the original price for the item, and the location. Note that this is the minimum required for this data. However, it is highly recommended that the entire price history for that item and location be provided. Otherwise, OO will incorrectly capture the ticket price changes or regular price changes. It is understandable that some items from the beginning of the sales history will not have the entire price history.

9. **Security**. RAF files must be provided when the user wants to impose security or data filters for the OO screens based on the merchandise and location nodes. The Manage OO Recommendation user filter will not show any location or merchandise nodes that are not allowed for a particular user.
10. **Product Images**. The URLs for rendering product images in OO screens can be provided.
11. **Pricing Groups**. If the user defines pricing groups such as Same Markdown Discount and so on, then they must be provided here for the batch OO runs to pick up.
12. **Flex Facts (or Custom Columns)**. If the user wants to see custom columns in Manage OO Recommendation, then these interfaces must be provided.
13. Before pushing any data into the Science engine, it is critical to make sure RAP foundation data is accurate and verified. Verification can be done through Data Visualizer or through the Innovation Workbench. Re-loading some of the RAP foundation data elements is non-trivial as it will require an SR to reset/truncate some of the target tables.

RSP foundation data (RSE_% tables)

Once the interface configuration is complete and the RAP Foundation data has been verified, the user can push the data from RAP into RSP. As mentioned before, once database logging has been enabled, users can check for any errors and the progress of the process in RSE_LOG_MSG. Before running the RSE_MASTER_ADHOC with appropriate flags to execute the following jobs, the user must set the correct values for configuration parameters in RSE_CONFIG. This can be done using Manage System Configurations (as described in Control and Tactical Center) and then selecting RSE_CONFIG from the drop-down list.

1. **Product Hierarchy**. The target table to check is RSE_PROD_HIER. When the user wants to load up to 9-level hierarchy (for example, fashion items with SBC ' STYLE ' STYLE/COLOR ' STYLE/COLOR/SIZE), then two configurations must be verified: LOAD_EXTENDED_PROD_HIER and PROD_ITEMUDA_SRC.
2. **Location Hierarchy**. The target table to check is RSE_LOC_HIER.
3. **Price Zone Groups and Price Zones**. The target tables to check are RSE_PRICE_ZONE, RSE_PRICE_ZONE_GRP, RSE_PRICE_ZONE_LOC, and RSE_PRICE_ZONE_PROD. The user must set the correct configuration level for the merchandise level that is mapped to a PRICE_ZONE_GROUP using this configuration in RSE_CONFIG: RSE_PZG_PROD_LEVEL.
4. **Sales**. The target table to check is RSE_SLS_TXN. This data will be further aggregated into multiple tables at different aggregate levels once the relevant aggregation jobs are executed. For OO, this table must be populated: RSE_SLS_PR_LC_CS_WK. Returns aggregation will not be required if the user does not plan on using the returns functionality.
5. **Inventory**. The target table to check for weekly inventory position at leaf node levels is RSE_INV_PR_LC_WK_A. Note that the warehouse inventory positions are also loaded into this target table as warehouse is treated like a leaf node on the location hierarchy. The configurations in RSE_CONFIG to check are INV_SRC_RI, RSE_INV_QTY_BOH_FLG, RSE_INV_QTY_IN_TRANSIT_FLG, RSE_INV_QTY_ON_ORD_FLG, RSE_WHSE_INV_QTY_BOH_FLG, RSE_WHSE_INV_QTY_IN_TRANSIT_FLG, and RSE_WHSE_INV_QTY_ON_ORD_FLG.
6. **Inventory Receipts**. The target table to check for inventory receipts at the leaf node level is RSE_INV_HIST_PR_LC_WK. The configuration to check is INV_RC_DT_SRC_RI.
7. **Customer Segments**. The target table to check is RSE_CUSTSEG_HIER. It is required to execute the job to load customer segments (even when there is just a dummy segment).
8. **Price and Cost**. Before pushing the price cost data into RSP, the user must make sure that the weekly sales aggregation job has been executed and that the customer segments have

been executed. The target table to check at leaf node levels is: RSE_PRICOST_PR_LC_WK.

9. **Security**. Target tables to check are RAF_FILTER_GROUP_MERCH, RAF_FILTER_GROUP_ORG, RAF_SEC_GROUP, RAF_SEC_USER, and RAF_SEC_USER_GROUP.

Direct loads to RSP

Some foundation data elements are loaded directly to RSE. Here is a description of some of the data elements that are used in OO.

These files must be prepared into ORASE_WEEKLY_ADHOC.zip.

1. Prepare the relevant _stg.txt file. (Note that some large files will be compressed into _stg.txt.gz files.)
2. Zip all the staging files into an archive named ORASE_WEEKLY_ADHOC.zip.
3. Upload ORASE_WEEKLY_ADHOC.zip to the SFTP site and copy the COMPLETE file to the COMMAND directory.

Then run the RSE_MASTER_ADHOC with the appropriate parameter. See: https://docs.oracle.com/en/industries/retail/retail-insights-cloud/21.0/rapog/mstr-dat-load-common-dsgn-ovrvw.htm

1. **Holidays**. The target table to check is RSE_HOLIDAY. This is a direct load to RSP and is loaded through RSE_HOLIDAY_STG. Any validation errors during loading are provided in the RSE_HOLIDAY_BAD table. As mentioned before, the bad tables can be viewed using Innovation Workbench.
2. **Warehouse Allocation Percentages**. RSE_INV_WHSE_LC_PR_ALLOC_STG. The target table to check is RSE_INV_WHSE_LC_PR_ALLOC. This is a direct load to RSP. The configuration to check is RSE_PRO_WHSE_ALLOC_FLG, which must be set to N when the client is loading allocation percentages.
3. **Product Attributes**. Product attributes are not required for OO; however, if the user wants to use them in the Manage OO Recommendation screen, then they must be provided directly to RSP. The target tables to check are RSE_PROD_ATTR_GRP_VALUE and RSE_PROD_ATTR_VALUE_XREF.

Forecasting

The next step is to create a forecast run type and run the model training to estimate the demand parameters. These steps are important to determine the appropriate aggregation levels for the next bucket, Optimization.

1. Create the forecast run type at the desired recommendation levels for the offers. For example, the recommendation levels can be STYLE COLOR, PRICE ZONE, and WEEK. Once the correct run type has been created, then user can click **Start Data Aggregation**. The configurations to check are RSE_INV_WHSE_ACTIVITY_USE_FLG (this flag must be set to N as otherwise the aggregation time would increase dramatically. The user can turn this flag on when the batches have started), PMO_PROD_HIER_TYPE (for example, this would be 3 for using extended hierarchy), and PMO_AGGR_INVENTORY_DATA_FLG. The target tables to check are PMO_ACTIVITIES and PMO_CUM_SLS.
2. Create the forecast run corresponding to the forecast run type and change the settings as needed. The user can select **Submit**, which will start training the model.
3. Once the forecast run is complete, the parameters are generated. If the user is satisfied with the parameters, then the user must Approve Demand Parameters, Approve Base Demand and Forecast. Then the user must activate the forecast run type and map it to the application. (In this case, it is Offer Optimization.)

Optimization

Once the forecasting run is complete, the next step is to supply the data for optimization. This data must be supplied before the user can optimize the runs to generate recommendations. To load this data, the user must use the PRO_MASTER_ADHOC job with the appropriate flags as described here:
https://docs.oracle.com/en/industries/retail/retail-insights-cloud/21.0/rapog/mstr-dat-load-oo-dsgn-ovrvw.htm

1. **Configurations**. The following configurations must be set up appropriately to supply the data at the relevant levels as well as to generate recommendations: PRO_CUST_HIER_PROCESSING_LVL, PRO_OPT_LOC_REC_LVL, and PRO_LOC_HIER_PROCESSING_LVL (which must be the same as PRO_OPT_LOC_REC_LVL). When recommendations are to be set by price zones then this is set to the Company level. Values are PRO_OPT_MERCH_REC_LVL, PRO_OPT_TIME_REC_LVL, PRO_PROD_HIER_PROCESSING_LVL (for example, class), PRO_PROD_HIER_RUN_SETUP_LVL (for example, dept), and PRO_PROD_HIER_TYPE (for example, 3 for extended hierarchy).
2. **Season-related data**. The target tables to check are PRO_SEASON, PRO_SEASON_PERIOD and PRO_SEASON_PRODUCT. It is not required to provide all historical seasons, as the optimization is concerned only with the ongoing seasons.
3. **Business Rules**. It is important that the rules are defined at the highest merchandise/location/season levels so that all items always find a rule to use. The target table to check is PRO_OPTIMIZATION_RULES.
4. **Price Ladder**. The target tables to check are PRO_PRICE_LADDER and PRO_PRICE_LADDER_DTL.
5. **Create OO runs**. The user can run the PRO_OPT_CREATE_RUNS_ADHOC_JOB with the FORCE option to create the OO runs for all merchandise and location/price zones.
6. **Optimize OO runs**. The user can run the PRO_OPT_ADHOC_JOB with the -batch option to optimize all the above created runs.

Manage OO Recommendations

The results will be available for the user to review/approve/export once the optimization job is complete.

Batch

Once the user is satisfied with the recommendations, then the user can put the system on a batch schedule to run at regular time intervals. Refer to "POM Jobs" about how to enable/disable the relevant jobs.

Security

Offer Optimization supports both data-based and user role-based filtering or security. Data filtering informs the application regarding which merchandise and locations can be accessed by a particular user (for example, the user 'tom' can only access locations within US). The user role restricts or allows which UI actions can be performed by a particular user.

Data Filtering

Data filtering can be specified (and integrated wherever applicable) in a similar fashion as discussed in Chapter 4, "Manage Data Filtering" of *Oracle Retail Merchandising Suite - Administration Guide* (https://docs.oracle.com/cd/E79623_01/rms/pdf/192000/merchcs-admin.pdf). When a customer does not have Oracle Retail

Merchandising Service, then the following interfaces can be leveraged to specify the user groups and users.

- RAF_FILTER_GROUP_MERCH
- RAF_FILTER_GROUP_ORG
- RAF_SEC_GROUP
- RAF_SEC_USER11.Pricing Groups. If the user to define pricing groups such as Same Markdown Discount, etc., then it must be provided here for batch OO runs to pick up.
- 12.Flex Facts (or Custom Columns). If the user desires to see custom
- RAF_SEC_USER_GROUP

User Roles

User roles are used to set up application user accounts through Oracle Identity Management (OIM). See *Oracle Retail AI Foundation Cloud Services Administration Guide* for details. Five roles are supported in this application:

- Pricing Administrator
- Pricing Manager
- Pricing Analyst
- Buyer
- Targeted Offer Role

Data Input Requirements

This section provides information about setting up the data that the Offer Optimization application uses to generate optimal price recommendations, including guidelines on the expectations for the data element requested and where it is used. Information about these files can be found in *Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface*.

Hierarchy Data

Hierarchies are part of the core data elements that are used in Offer Optimization, in both the forecasting and optimization modules. The four types of hierarchies are Location Hierarchy, Merchandise Hierarchy, Calendar Hierarchy, and Customer Segments Hierarchy. Hierarchy data is required.

Most of the W_% interfaces have numerous columns and thus, it is possible to define which columns will be populated in the interface by specifying the column headers (order of columns does not have to be the same) in the corresponding W_%.ctx file.

- Location Hierarchy. An example of location hierarchy is: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE. When the user wants to optimize by location hierarchy, then the run's optimization level must match a node in the location hierarchy. For example, you can run the optimization at the District level. Note that the E-com channel can be defined as part of the location hierarchy. Relevant interfaces are W_INT_ORG_ATTR_DS, W_INT_ORG_DHS, W_INT_ORG_DS.
- Price Zone Group and Price Zones. OO provides the user with the flexibility to group any set of stores as part of a price-zone and use it as the run setup level for optimization. A price-zone group contains the set of price-zones, and each price-zone contains the set of stores. Note that two price-zones with a price-zone group cannot have any overlapping

stores within a price-zone group. An example of Price-zone Group (PZG) and Price-zones (PZ) is as follows: PZG-100, mapped to DEPT1 with PZ-101 = Stores 1 to 10, PZ-102 =Stores 11 to 20. PZG-200, mapped to DEPT2 with PZ-201= Stores 1 to 5. PZ-202 = Stores 6 to 10. PZ-203 = Stores 11 to 20. Note that in OO, a price-zone cannot be defined above country-level since the currencies can be different. Further, the price-zone can be associated with a list of items. Different merchandise in a Department at a particular store can be part of different price zones. For example, women's university apparel in all university-based stores is grouped into one price zone and the women's non-university apparel belongs to another price-zone. If a price-zones are not defined or used, then the UI displays a default value of -1-WORLDWIDE.

Corresponding interfaces are W_RTL_CLSTR_GRP_DS, W_RTL_CLSTR_GRP_HDR_LC_DS, W_RTL_CLSTR_GRP_PRD_DS, W_RTL_CLSTR_HDR_DS. In C_ODI_PARAM, when the value for CLSTR_PROD_LEVEL is in (DEPT, CLS, SBC) then all MERCH_ID values map to the associated level in W_PROD_CAT_DH. On the other hand, in C_ODI_PARAM, when the value for CLSTR_PROD_LEVEL is in (ITEMLIST), then all MERCH_ID values map to an Item List ID in W_RTL_ITEM_GRP1_D. The later allows the retailer to provide any custom item lists to be mapped to a price zone.

- Merchandise Hierarchy. An example of merchandise hierarchy is as follows: CHAIN ' COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE ' COLOR ' SIZE (SKU). There can be up to nine levels in the merchandise hierarchy. When Offer Optimization is used for fashion apparel, it is expected that the you will provide Style and Color through the relevant interfaces. However, if the retailer deals with merchandise such as electronics, that does not necessarily have style or color, the application will work but some of the UI functionality will not be helpful. For example, Custom Rule has the merchandise selectors Class, Subclass, and Style. If no styles are loaded, then the last selector will not be helpful. Further, in such a situation, the recommendation levels will be at SKU level, as other levels may not be meaningful.

  The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_DS interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_DHS interface. In addition, there must be consistency among the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_DS, is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy. If the retailer wants to use an extended merchandise hierarchy, then the retailer must populate this interface.

  Here are the specific fields:

  – PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)

  – PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)

  – PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699). The value of PROD_NUMs is the same as the value in the W_PRODUCT_DS.PROD_NUM interface.

  Here is what this looks like:

PROD_NUM	PRODUCT_ATTR13_NAME	PRODUCT_ATTR14_NAME	PRODUCT_ATTR15_NAME
STYLE_PROD_NUM	STYLE_PROD_NUM		
STY/COL_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	
STY/COL/SIZ_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	STY/COL/SIZ_PROD_NUM

- Calendar Hierarchy. This is one of the core hierarchies. The retailer can specify the calendar depending on their business requirements (for example, fiscal calendar). The relevant interface for this is W_MCAL_PERIOD_DS.
- Customer Segments Hierarchy. Offer Optimization supports pricing recommendations at the customer segment level. To use this functionality, the retailer must load customer segments. The relevant interfaces to supply this data are W_RTL_CUSTSEG_DS and W_RTL_CUST_CUSTSEG_DS. Retailers can use customer segments for many purposes; however, for OO purposes, retailers can use one set of customer segments to determine forecasting parameters and optimal price recommendations.

Runs in the Offer Optimization can be set up to run at configured levels for location or price zone, merchandise, calendar, and customer levels.

- Location Level. The run's location processing level is configurable in RSE_CONFIG, and it is denoted as PRO_LOC_HIER_PROCESSING_LVL. For example, a typical level is Region. If the user specifies price-zones, then location level defaults to COUNTRY.
- Merchandise Level. The run's merchandise processing level is configurable in RSE_CONFIG, and it is denoted as PRO_PROD_HIER_PROCESSING_LVL. For example, a typical level is the Class level.
- Calendar Level. The run's calendar processing level is configurable in RSE_CONFIG, and it is denoted as PRO_CAL_HIER_PROCESSING_LVL. For example, a typical level is Week.
- Customer Level. The run's customer processing level is configurable in RSE_CONFIG, and it is denoted as PRO_CUST_HIER_PROCESSING_LVL. For example, a typical level is Segment or Chain (that is, segment-all).
- Setup Level. The run is set up at a level higher than the run's merchandise level. This is configurable in RSE_CONFIG, and it is denoted as PRO_PROD_HIER_RUN_SETUP_LVL. For example, a typical level is Department level.

In this guide, the typical processing levels are generally used as illustration. If necessary, further details will be used to clarify non-typical levels. These levels are also important for the forecasting module since it provides the relevant parameters at the appropriate levels so that it can be consumed by the optimization module. It is essential that the configuration levels be decided based on the business of the retailer at the time of implementation.

Hierarchies are assumed to be set up once for each season and can be revisited whenever it changes. It is expected that the hierarchies do not change from one week to next week within a year (or season). Since hierarchies are core foundational elements any change to hierarchies can be costly and must be planned accordingly. For example, if customer segments are redone, then the parameters must be recalculated. The retailer must plan for such changes as such changes cannot necessarily be completed in the normal weekly batch process or window.

Inventory Data

Required interface. Historical inventory data as well as daily inventory data can be specified through this interface: W_RTL_INV_IT_LC_DY_FS. In addition to the non-null fields, the minimum required data fields that used by OO are:

- INV_SOH_QTY - store on hand
- INV_ON_ORD_QTY - on order inventory quantity
- INV_IN_TRAN_QTY - in transit inventory quantity
- INV_UNIT_RTL_AMT_LCL

It is also recommend populating some other core fields on the table if the data exists, such as the RTL and COST values:

- INV_SOH_RTL_AMT_LCL - total retail value
- INV_ON_ORD_RTL_AMT_LCL - total on-order value
- INV_IN_TRAN_RTL_AMT_LCL - total in-transit value
- INV_AVG_COST_AMT_LCL - weighted avg cost (WAC)
- INV_UNIT_COST_AMT_LCL - current unit cost
- INV_SOH_COST_AMT_LCL - total OH cost value
- INV_ON_ORD_COST_AMT_LCL - total on order cost value
- INV_IN_TRAN_COST_AMT_LCL - total in transit cost value
- DOC_CURR_CODE - document currency of the record
- LOC_CURR_CODE - local currency for the RI system
- LOC_EXCHANGE_RATE - exchange rate (for BCF probably hard-coded to 1)

OO also uses these two fields, which cannot be provided directly through the above interface. However, they are calculated based on inventory receipts data provided in W_RTL_INVRC_IT_LC_DY_FS.

- LAST_INVRC_DT
- FIRST_INVRC_DT

Once the inventory data is loaded, the user can configure which inventory components are to be used in OO by setting the corresponding flags appropriately: RSE_INV_QTY_BOH_FLG, RSE_INV_QTY_ON_ORD_FLG, RSE_INV_QTY_IN_TRANSIT_FLG.

Price History Data

Required Interface. Historical price data as well daily price history can be loaded W_RTL_PRICE_IT_LC_DY_FS. In addition to non-null fields, these are the core fields to populate:

- PRICE_CHANGE_TRAN_TYPE. The PRICE_CHANGE_TRAN_TYPE is based on the RMS usage of the value in PRICE_HIST. For example, 0 = initial price, 4 = cost change, 8 = clearance price, 9 = promo price.
- SELLING_UNIT_RTL_AMT_LCL
- BASE_COST_AMT_LCL
- DOC_CURR_CODE
- LOC_CURR_CODE
- LOC_EXCHANGE_RATE

Retail Code

Required Interface. OO directly uses W_RTL_CODE_DS values to drive the markdown results, for example, if OO is generating a first markdown then the associated code/desc values for the first markdown that customer wants to see are obtained from this table. An example of this data is provided below.

Table 12–1 Sample Retail Code Data

CODE_TYPE	CODE	CODE_SEQ	CREATED_BY_ID	CHANGED_BY_ID	CREATED_ON_DT	CHANGED_ON_DT	DELETE_FLG	DATASOURCE_NUM_ID	ETL_PROC_WID	INTEGRATION_ID
MKDN	1	1	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~1
	2	2	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~2
	3	3	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~3
	4	4	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~4

Retail Sales Data

Retail Sales data is a required interface and is provided through W_RTL_SLS_TRX_IT_LC_DY_FS. OO requires you to provide retailer's historical sales data at the desired hierarchy levels. If you want to use the targeted offers, then you must provide the customer-linked transactions sales data. The retailer does not have to provide the data aggregated to the segment-level but must provide the customer ID and customer segment mapping as part of customer segment interface.

Note that the sales returns are handled in the same interface. The returns data identifies the original location where the item was purchased and when it was purchased. This information is useful in returns analysis and in calculating the returns parameters. If the user wants to separate the returns from gross sales, then retailer is required to provide these two values separately.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Promotions

A historical and future planned promotions-related interface is optional. OO allows the user to provide any promotions using the following interface: RSE_PLAN_PROMOTION_STG. This interface provides the retailer with the ability to specify the start and end dates of a promotion event and which merchandise or locations are part of the promotion. This information can help OO in two ways: identify when the promotions have occurred in the past and learn which future planned promotions are similar to the ones in the past. In this interface, the retailer must specify name, when (time), which items, and how deep (for example, for a Father’s Day Sale, 20% off all merchandise in Mens division and across all stores).

Warehouse Inventory Allocation

The Warehouse Inventory Allocation interface is optional. The OO user can provide the warehouse to store/online location mapping through this interface (RSE_INV_WHSE_LC_PR_ALLOC_STG). In addition, the user can provide the allocation percentage for each item (that is, how much of warehouse inventory is allocated to the particular store/online location). The

user can configure whether to include all components of warehouse inventory in the calculation: warehouse on order, warehouse in transit, warehouse on hand using the following flags: RSE_WHSE_INV_QTY_ON_ORD_FLG, RSE_WHSE_INV_QTY_IN_TRANSIT_FLG, and RSE_WHSE_INV_QTY_BOH_FLG. If the user does not provide the allocation percentages, then OO provides the user with ability to virtually allocate warehouse inventory using an in-built optimization algorithm. The optimization algorithm determines the allocation percentages based on the maximization of revenue across all warehouses. To turn on the warehouse virtual allocation algorithm, corresponding flag RSE_PRO_WHSE_ALLOC_FLG needs to be set to Y.

Competitor Prices

The Competitor Price interfaces, which are optional, are used to provide competitor prices. The two interfaces are W_RTL_COMP_PRICE_IT_LC_DY_FS and W_RTL_COMP_STORE_DS. The retailer can specify the price of an item at multiple competitors/locations and, depending on the configured option, either the minimum or the average of competitor prices is used for the optimization. Competitor prices are used to restrict the price recommendations to a certain percent range of the competitor price. For example, if the retailer wants to match the competitor's promotion of 20 percent off but with some tolerance, then this interface allows the retailer to specify the competitor price through the interface. Tolerance is specified as a global parameter in RSE_CONFIG. Note that in this release, the product can support competitor prices only through an interface.

Product Attributes

The product attributes that are useful in OO forecasting and in particular for targeted offers can be defined through this interface. Raw product attributes (for example, fabric, material, and so on) can be provided through this interface. Make sure the attribute values are as clean as possible. For example, COKE vs. COK vs. COEK must be normalized to COKE so that OO does not interpret them as different attributes.

The retailer must use W_RTL_ITEM_GRP1_DS, since it can support any number of attributes, such as BRAND, and the batch load process picks up the attributes added there automatically. The product attributes that are useful in OO forecasting and in particular for targeted offers can be defined through this interface. Raw product attributes (for example, fabric, material, and so on) can be provided through this interface. Make sure the attribute values are as clean as possible. For example, COKE vs. COK vs. COEK must be normalized to COKE so that OO does not interpret them as different attributes.

If the retailer wishes to load STYLE or COLOR as product attributes, then the W_RTL_ITEM_GRP1_DS interface is used to provide style and color attributes for the different products, using a value in PROD_GRP_TYPE of either STYLE or COLOR. The actual values for the Styles and Colors are provided in columns FLEX_ATTRIB_2_CHAR and FLEX_ATTRIB_4_CHAR using values that represent the ID for the Style (for example, 1234), and a description for the Style (for example, Loose Fit). The columns FLEX_ATTRIB_3_CHAR and FLEX_ATTRIB_10_CHAR contain the appropriate designation for STYLE or COLOR.

OO supports grouping of product attribute values and it is required to load these additional interfaces: RSE_PROD_ATTR_GRP_VALUE_STG and RSE_PROD_ATTR_VALUE_XREF_STG. It is not required to group the attribute values. Consider the following examples:

- Consider Brand attribute with values Adidas, Champion, Nike and Under Armour. Since we are not grouping any attribute values for attribute Brand, each value by itself will be a group.
- Consider the Color attribute. There might be too many variants of the same color and thus it might be meaningful to group the color values. The first interface defines how many groups of colors would be there - for example, Trending, Casual, Professional. The second

interfaces maps the actual attribute values to the groups - for example, Trending = {Flame Scarlet, Biscay Green}, Casual={Pink, Green}, Professional={Black, White, Grey}.

- These interfaces can also help with normalization of attribute values. Consider an example when there are a lot of misspellings (for example, COEK vs. COK vs. COEK). The retailer can use this interface to normalize the attribute values and create one group called COKE.

Product Images Data

Product images that are available on a customer-hosted web server can be viewed In OO via the Results screen. The W_RTL_PRODUCT_IMAGE_DS.dat interface contains a column called PRODUCT_IMAGE_ADDR, that can contain the full URL to an image of the product. This URL must be in the following format:

http[s]://servername[:port]/location/filename.extension

For example:

PRODUCT_IMAGE_NAME = imagename.png

PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png

PRODUCT_IMAGE_DESC= Short description of the image

The OO application running in the cloud does not directly access these images, so there is no need to expose these images outside of the customer's firewall. As long as the user of the OO application has access to the URL while running the OO application, then the user's web browser will be able to resolve the URL and retrieve the images for display when they choose this option. The images must be in a file format that the web browser can display. Since the images shown in the UI are small, these images do not need to be high quality images. The size of the image files will affect the time it takes to render them.

Season

PRO_SEASON_STG is used to define the name of the season with the start and end dates. For example, Halloween Season is used to identify merchandise that sells for Halloween. If the retailer does not have any seasons, then the retailer can define one long season.

Season Periods

PRO_SEASON_PERIOD_STG defines which periods belong to the season and lets users define whether a promotion or a markdown is allowed in a particular period. This interface can include the promotion and markdown calendar information of the retailer.

Season Products

PRO_SEASON_PRODUCT_STG defines which products are associated with a season. This interface indicates which products are to be included in the optimization for recommendations.

Price Ladder

PRO_PRICE_LADDER_STG can be used to supply price ladders. It supports three types of price ladders: Price Point (denoted by type A), Percentage Off Ticket Price (denoted by PT), and Percentage Off Full or Original Price (denoted by PO). The LADDER_COUNT can be used to define the ladder for the first and/or subsequent markdowns or promotions. The Ladder_count=0 must always be defined and is used in optimization. Any non-zero ladder_ count ladders (only type A) are used for post-processing of price recommendations. For example, Ladder_count=0 is ending with .99, and ladder_count=2 is ending with .98 and ladder_count=3 is ending with .97.

Currency

PRO_COUNTRY_LOCALE_STG can be used to define the currency based on the country. This allows the same instance to support multiple currencies. For example, US locations must generate price recommendations (and price-related metrics) in USD, and Canadian locations must generate price recommendations (and price-related metrics) in CAD. Budgets can be specified in one global currency and optimization runs can be in a different local currency based on PRO_COUNTRY_LOCALE_STG. In which case, budgets are converted based on exchange rates provided in W_EXCH_RATE_GS.

Holidays

RSE_HOLIDAY_STG can be used to define the holidays that are used by the retailer. Major holidays can cause additional traffic visiting the store or the web channel.

Pricing Product Groups

W_RTL_ITEM_GRP1_DS can be used to define the different pricing product groups that are used by the retailer. PRO_OPTIMIZATION_RULES_STG allows the retailer to specify the group types for each of the defined groups. For example, both these interfaces help retailer define which items comprise the group and that this group would need the same markdown discount. Such groups and corresponding constraints are picked up by batch optimization runs.

Budget

W_RTL_PLAN1_PROD1_LC1_T1_FS can be used to specify the budgets for markdowns and/or promotions used by the retailer. A budget can be fed in at the run's merchandise setup-level (for example, Division) or it can be fed in at the run's merchandise processing-level (for example, Department). The interface allows the user to provide separate budgets or a combined budget for promotions and markdowns. The relevant fields from the interface are: slspr_rtl_amt (promotion budget), slscl_rtl_amt (clearance budget), loc_exchange_rate, mcal_wid, delete_flg, prod_dh_wid, org_dh_wid. On loc_exchange_rate, if the retailer operates in only one country, then it can be 1. However, it is possible that when a customer spans multiple countries, the planning budget can be in one global currency and when it is applied, it can be in local currency. For example, to convert USD into GBP, set loc_exchange_rate as 1/0.75.

Strategy and Business Rules

PRO_OPTIMIZATION_RULE_STG allows the user to define business rules and associate them with a strategy set. This interface provides the following abilities for the user:

- There will always be a DEFAULT_SET strategy that is reserved to be used by batch runs in production. Other strategy sets are available for the user to do what-if runs.
- It is advised that retailers define all the business rules in the DEFAULT_SET strategy. It is not required that the user define complete set of rules again for a new strategy set. For example, an Aggressive Promotion Strategy set can define rules associated with promotions only. In such a case, the system will try to find the closest rule that is available in DEFAULT_SET when it is not available in the current strategy set.
- In any strategy set, the user can define rules at any merchandise and location levels. Rules will be applied to the merchandise and location that are closest to it. The term "Closest" can be defined as follows: for a given product level, move up the location hierarchy first to find any applicable rule. For example, the user defines Min First Markdown as 10% at Banner- All locations level and 20% for Men's Division-All locations.Then, merchandise in Men's division receives the 20% Min First Markdown rule, and all other merchandise receives the 10% Min First Markdown rule for all locations.

- Strategies can also be defined for price zone group (PZG) or price zone (PZ) level, when applicable. Note the following: (1) When the user wants to use PZG or PZ to specify rules, then the LOCATION_KEY must be -1 and LOCATION_HIER_LEVEL must be ALL. (2) When the user wants to specify a rule for all PZ Groups, then PZ GROUP = ALL (and PZ Key=-1). (3) When the user wants to specify a rule for all PZ within a PZG, then PZ GROUP must be specified (and PZ Key=-1). (4) When the user wants to specify a rule for a specific PZ , then PZ GROUP KEY = and PZ KEY must be specified. (5) When the user is not using PZG/PZ, then PZ GROUP KEY and PZ KEY must be -1.

Custom Columns

W_RTL_FLEXFACT_FS can be used to add any custom column to the recommendations in the Manage Recommendation screen. Custom columns are picked up by batch runs at the time of their creation and assigned to the item and location. It supports up to ten numeric columns (for example, last year sales), twenty currency columns (for example, revenue till date), ten text columns (for example, supplier), five percentage columns (for example, sell-through for last four weeks), and five date columns (for example, last markdown date).

External Forecast Adjustment

PRO_FCST_EXT_EFFECT_ADJ_STG is used to provide adjustments to the forecast. It may occasionally be possible that extraneous factors that are not visible to OO can impact the forecast (for example, Covid-19). In such a case, this interface can be leveraged to provide such factors to impact the OO forecast.

Configuration and Expected Levels for Interfaces

The expected levels for the interfaces are based on the configuration levels specified for the recommendation and optimization levels. There are five dimensions that are used to define these configurations: Customer (applicable only for targeted offers), Calendar, Merchandise, Location, and Price-zone. The optimization level determines the levels at which optimization is performed. For example, Division and Region (it can also be a price-zone). The recommendation level determines at what levels recommendations must be generated. Typically, these are lower levels of merchandise and location than the optimization levels (for example, Style/Color and Region). Note that the Calendar dimension tells the system at which level recommendations are generated (for example, Weekly vs Daily). The customer dimension tells the system that customer segments are available to generate targeted offers. Promotions and Markdowns are not generated at the customer segment level. If price-zones are not used, then the Price-zone column will be -1. When price-zones are used, then both optimization and recommendation levels will be price-zone. In such a case, both the location configurations, PRO_LOC_HIER_PROCESSING_LVL and PRO_OPT_LOC_REC_LVL, must be set to COUNTRY.

Table 12–2 Configuration: Expected Levels

Data	Configuration Parameters	Customer Segment	Calendar	Merchandise	Location	Price Zone
Optimization Level	Levels at which the optimization process is executed (Units of work)	PRO_CUST_HIER_PROCESSING_LVL	PRO_CAL_HIER_PROCESSING_LVL	PRO_PROD_HIER_PROCESSING_LVL	PRO_LOC_HIER_PROCESSING_LVL	-1 or PZ
Recommendation Level	Level at which the recommendation results are generated	PRO_OPT_CUST_REC_LVL	PRO_OPT_TIME_REC_LVL	PRO_OPT_MERCH_REC_LVL	PRO_OPT_LOC_REC_LVL	-1 or PZ

Depending on the above configuration levels, the interface levels expected are as follows:

Table 12–3 Interfaces: Expected Levels

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Season	PRO_ SEASON_ STG	Range of time periods (e.g., weeks) for the season. This has to match the calendar hierarchy loaded	N/A	N/A	N/A	N/A	N/A
Season Period	PRO_ SEASON_ PERIOD_ STG	Periods (e.g., weeks) within a season where the PRO_PROD_ HIER_ PROCESSIN G_LVL (e.g., CLASS) is active	N/A	PRO_CAL_ HIER_ PROCESSIN G_LVL Date range could span multiple calendar units (i.e. Multiple weeks)	PRO_PROD_ HIER_ PROCESSING _LVL or any hierarchy level higher	PRO_LOC_ HIER_ PROCESSIN G_ LVL or any other higher hierarchy level	-1 or PZ
Season Product	PRO_ SEASON_ PRODUCT_ STG	Products that must be included in given selling seasons	N/A	PRO_CAL_ HIER_ PROCESSIN G_LVL Date range can span multiple calendar units (i.e. Multiple weeks)	PRO_OPT_ MERCH_ REC_LVL	PRO_OPT_ LOC_REC_ LVL	-1 or PZ
Price Ladder	PRO_ PRICE_ LADDER_ STG	Product price ladder	N/A	N/A	PRO_PROD_ HIER_ PROCESS-IN G_LVL or any higher	PRO_LOC_ HIER_ PROCESSIN G_LVL	-1 or PZ
Country Locale	PRO_ COUNTRY_ LOCALE_ STG	Defines the specific locale for the location. Provided at the COUNTRY level	N/A	N/A	N/A	PRO_LOC_ HIER_ PROCESSIN G_LVL or any hierarchy level higher	N/A

Table 12–3 (Cont.) Interfaces: Expected Levels

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Planned Promotion	RSE_PLAN_ PROMOTIO N_STG	Planned promotion at PRO_PROD_ HIER_ PROCESSIN G_LVL (e.g., CLASS)	N/A	PRO_CAL_ HIER_ PROCESSIN G_LVL Date range could span multiple calendar units (i.e. Multiple weeks)	PRO_PROD_ HIER_ PROCESS-IN G_LVL	PRO_LOC_ HIER_ PROCESSIN G_ LVL	-1 or PZ
Holidays	RSE_ HOLIDAY_ STG	Major holidays	N/A	Date when the holiday is observed	Any level in merchandise hierarchy	Any level in location hierarchy	N/A
Strategy and Business Rules	PRO_ OPTIMIZATI ON_RULE_ STG	Business rules and associated strategies	N/A	N/A	Any level in merchandise hierarchy	Any level in location hierarchy	N/A

Table 12–4 Expected Levels for Optional Interfaces*

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Inventory	RSE_INV_ PR_LC_WK_ A_STG	Inventory available at the start of the period (e.g., week). Inventory is aggregated to the PRO_ LOC_HIER_ PROCESSIN G_LVL (e.g., Price Zone)	N/A	Week	SKU level (Style/Color/Si ze)	Store	N/A
Price Cost	RSE_ PRICOST_ PR_LC_WK_ STG	Price cost for the time period (e.g., week)	N/A	Week	SKU level (Style/Color/Si ze)	Store	N/A

Optional* Interfaces

The interfaces specified in this section are not required and are available to handle special retailer situations. When the retailer provides the data through the interfaces tagged as 'required', then the system automatically uses the primary interfaces.

- Inventory (RSE_INV_PR_LC_WK_A_STG)
- Price Cost (RSE_PRICOST_PR_LC_WK_STG)

Integration with Retail Pricing Cloud Service

Additional configuration parameters must be set to integrate OO with Retail Pricing Cloud Service. See the list of parameters shown in Table 12–5. In addition, the RPM username and password must be added under the credential store for Merchandise RPM Access. You can

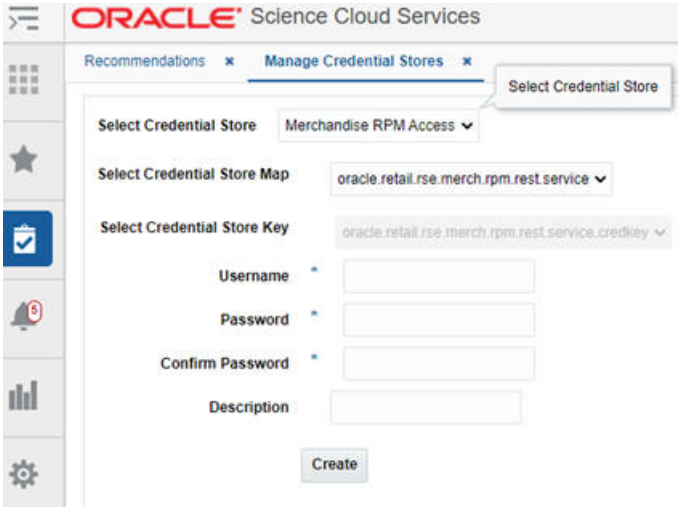
modify these values using the UI screen for Control and Tactical Center -> Manage Credential Stores. Control and Tactical Center is available under the task menu.

Setting up the Configuration Parameters for Integration with Retail Pricing Cloud Service

Table 12–5 Configuration Parameters for Integration with Retail Pricing CS

Parameter Name	Parameter Value	Dexcription
PRO_MERCH_RPM_FLG	N	Flag to identify whether to publish markdown recommendations to RPM.
PRO_MERCH_RPM_HOSTNAME		URL host name to publish recommendations to RPM. This must be same as CN name in the certificate.
PRO_MERCH_RPM_CONTEXT_ ROOT	RpmRestService/services/private	Context root of the RPM webservice url.
PRO_MERCH_RPM_ CLEARANCE_PATH	clearance/induction	Path of the webservice, e.g., clearance/induction.
PRO_MERCH_RPM_PORT	443	Port number to publish recommendations to RPM.
PRO_MERCH_RPM_HTTP_ PROXY_HOSTNAME		Proxy host name to publish recommendations to RPM.
PRO_MERCH_RPM_HTTP_ PROXY_PORT		HTTP proxy port number to publish recommendations to RPM.
PRO_MERCH_RPM_HTTPS_ PROXY_PORT		HTTPS proxy port number to publish recommendations to RPM.
PRO_MERCH_RPM_VERSION	19.0	RPM version to use for publishing customer segment to relate (format 16.0).
PRO_MERCH_RPM_USE_HTTPS	Y	Flag to identify whether to publish markdown recommendations to RPM over HTTPS.
PRO_MERCH_RPM_ PROMOTION_PATH	promotion	Path of the RPCS webservice for promotions.

Figure 12–3 Setting up the RPM Access in the Credential Store



POM Jobs

This section describes the batch jobs provided in the POM. Use this information to set up appropriate jobs for the regular batch process. Batch jobs do not expect any user input, and parameters are set to default values. Jobs are listed here by the functional area:

Foundation jobs that load product hierarchy, location hierarchy, calendar, product attributes, price zones, specify merchandise and location base user data filters (or security) are:

- ORASE_START_BATCH_JOB
- ORASE_START_BATCH_REFRESH_RESTR_JOB
- ORASE_START_BATCH_SET_ACTIVE_JOB
- ORASE_START_BATCH_END_JOB
- RSE_WEEKLY_INPUT_FILES_START_JOB
- WEEKLY_INPUT_FILES_WAIT_JOB
- WEEKLY_INPUT_FILES_VAL_JOB
- WEEKLY_INPUT_FILES_COPY_JOB
- RSE_WEEKLY_INPUT_FILES_END_JOB
- RSE_DIMENSION_LOAD_START_END_JOB
- RSE_DIMENSION_LOAD_START_START_JOB
- RSE_CAL_HIER_ETL_START_JOB
- RSE_PROD_HIER_ETL_START_JOB
- RSE_LOC_HIER_ETL_START_JOB
- RSE_REGULAR_MAIN_LOAD_JOB
- RSE_LOC_SRC_XREF_LOAD_JOB
- RSE_PROD_SRC_XREF_LOAD_JOB
- RSE_LOC_HIER_LOAD_JOB
- RSE_LOC_HIER_TC_LOAD_JOB
- RSE_LOC_HIER_DH_LOAD_JOB
- RSE_LOC_HIER_ETL_END_JOB
- RSE_FISCAL_MAIN_LOAD_JOB
- RSE_CAL_HIER_ETL_END_JOB
- RSE_PROD_HIER_LOAD_JOB
- RSE_PROD_TC_LOAD_JOB
- RSE_PROD_DH_LOAD_JOB
- RSE_PROD_GROUP_LOAD_JOB
- RSE_PROD_HIER_ETL_END_JOB
- RSE_CM_GRP_HIER_LOAD_START_JOB
- RSE_PRICE_ZONE_LOAD_START_JOB
- RSE_PROD_ATTR_LOAD_START_JOB
- RSE_PRICE_ZONE_GRP_LOAD_JOB

- RSE_PROD_ATTR_GRP_VALUE_STG_JOB
- RSE_PRICE_ZONE_LOAD_JOB
- RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB
- RSE_CM_GRP_HIER_LOAD_END_JOB
- RSE_PRICE_ZONE_LOAD_END_JOB
- RSE_PROD_ATTR_VALUE_XREF_STG_JOB
- RSE_DIMENSION_LOAD_END_START_JOB
- RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB
- RSE_DIMENSION_LOAD_END_END_JOB
- RSE_PROD_ATTR_LOAD_END_JOB
- RSE_SLS_START_START_JOB
- RSE_USER_DATA_FILTER_START_JOB
- RSE_LIKE_LOC_LOAD_START_JOB
- RSE_SLS_START_END_JOB
- RSE_SLS_TXN_START_JOB
- RSE_USER_DATA_FILTER_LOAD_JOB
- RSE_LIKE_LOC_STG_JOB
- RSE_USER_DATA_FILTER_END_JOB
- RSE_LIKE_LOC_LOAD_JOB
- RSE_SLS_TXN_SETUP_JOB
- RSE_LIKE_LOC_LOAD_END_JOB
- RSE_SLS_TXN_PROCESS_JOB
- RSE_SLS_TXN_END_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_PROCESS_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_START_JOB
- RSE_WKLY_SLS_CUST_SEG_START_
- RSE_WKLY_SLS_START_JOB
- RSE_WKLY_SLS_SEG_SETUP_JOB
- RSE_WKLY_SLS_SETUP_JOB
- RSE_WKLY_SLS_PROCESS_JOB
- RSE_WKLY_SLS_SEG_PROCESS_JOB
- RSE_WKLY_SLS_END_JOB
- RSE_WKLY_AGGR_SLS_START_JOB
- RSE_WKLY_AGGR_SLS_END_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_END_JOB
- RSE_WKLY_SLS_CUST_SEG_END_JOB
- RSE_SLS_END_START_JOB

- RSE_SLS_END_END_JOB
- RSE_INV_LOAD_START_JOB
- RSE_PRICE_LOAD_START_JOB
- RSE_INV_PR_LC_WK_STG_JOB
- RSE_PRICOST_PR_LC_WK_STG_JOB
- RSE_PRICOST_PR_LC_WK_LOAD_JOB
- RSE_PRICOST_PR_LC_WK_SETUP_JOB
- RSE_PRICOST_PR_LC_WK_PROCESS_JOB
- RSE_PRICE_LOAD_END_JOB
- RSE_INV_PR_LC_WK_LOAD_JOB
- RSE_INV_PR_LC_WK_SETUP_JOB
- RSE_INV_PR_LC_WK_PROCESS_JOB
- RSE_INV_LOAD_END_JOB
- ORASE_END_START_JOB
- ORASE_END_RUN_DATE_UPDT_JOB
- ORASE_END_END_JOB

Estimation and forecasting jobs that perform parameter estimation and generate forecast are:

- PRO_BASELINE_LOAD_START_JOB
- PRO_FCST_EXT_EFF_ADJ_START_JOB
- PRO_BASELINE_STG_JOB
- PRO_FCST_EXT_EFF_ADJ_STG_JOB
- PRO_PLAN_PROMOTION_LIFT_STG_JOB
- PRO_BASELINE_LOAD_JOB
- PRO_DOW_PROF_START_JOB
- PRO_FCST_EXT_EFF_ADJ_LOAD_JOB
- PRO_BASELINE_LOAD_END_JOB
- PRO_DOW_PROF_STG_JOB
- PRO_DOW_PROF_LOAD_JOB
- PRO_DOW_PROF_END_JOB
- PRO_FCST_EXT_EFF_ADJ_END_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_START_JOB
- PRO_MODEL_DATES_LOAD_START_JOB
- PRO_LIFECYCLE_FATIGUE_STG_JOB
- PRO_MODEL_DATES_STG_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_JOB
- PRO_MODEL_DATES_LOAD_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_END_JOB

- PRO_MODEL_DATES_LOAD_END_JOB
- PRO_PLAN_PROMOTION_LIFT_LOAD_JOB
- PRO_SEASONALITY_STG_JOB
- RSE_HOLIDAY_LOAD_START_JOB
- RSE_PLAN_PROMO_LOAD_START_JOB
- RSE_HOLIDAY_LOAD_STG_JOB
- RSE_HOLIDAY_LOAD_JOB
- RSE_HOLIDAY_LOAD_END_JOB
- RSE_PLAN_PROMO_STG_JOB
- RSE_PLAN_PROMO_LOAD_JOB
- RSE_PLAN_PROMO_LOAD_END_JOB
- PMO_HOLIDAY_LOAD_START_JOB
- PMO_HOLIDAY_LOAD_JOB
- PMO_HOLIDAY_LOAD_END_JOB
- PRO_SEASONALITY_LOAD_JOB
- PMO_ACTIVITY_LOAD_START_JOB
- PMO_ACTIVITY_STG_JOB
- PMO_ACTIVITY_LOAD_JOB
- PMO_ACTIVITY_LOAD_END_JOB
- PMO_RUN_EXEC_START_JOB
- PMO_CREATE_BATCH_RUN_JOB
- PMO_RUN_EXEC_SETUP_JOB
- PMO_RUN_EXEC_PROCESS_JOB
- PMO_CUMUL_SLS_START_JOB
- PMO_CUMUL_SLS_SETUP_JOB
- RSE_PUBLISH_FCST_PARAMETER_START_JOB
- PMO_CUMUL_SLS_PROCESS_JOB
- RSE_PUBLISH_FCST_PARAMETER_MODEL_DATE_JOB
- PMO_CUMUL_SLS_END_JOB
- RSE_MODEL_START_START_JOB
- RSE_MODEL_START_CALC_JOB
- RSE_PUBLISH_FCST_PARAMETER_SPREAD_DOWN_JOB
- RSE_PUBLISH_FCST_PARAMETER_SEAS_CURVE_JOB
- RSE_PUBLISH_FCST_PARAMETER_END_JOB
- RSE_MODEL_START_END_JOB
- PMO_RUN_EXEC_END_JOB
- RSE_FCST_BATCH_RUN_START_JOB

- RSE_CREATE_FCST_BATCH_RUN_JOB
- RSE_FCST_BATCH_PROCESS_JOB
- RSE_FCST_BATCH_RUN_END_JOB

Optimization jobs that load business rules and other related data are:

- PRO_COUNTRY_LOCALE_START_JOB
- PRO_OPT_RULE_START_JOB
- PRO_PLAN_PROMOTION_LOAD_START_JOB
- PRO_PROD_LOC_CDA_LOAD_START_JOB
- PRO_COUNTRY_LOCALE_STG_JOB
- PRO_OPT_RULE_STG_JOB
- PRO_COUNTRY_LOCALE_LOAD_JOB
- PRO_OPT_RULE_LOAD_JOB
- PRO_PROD_LOC_CDA_LOAD_JOB
- PRO_COUNTRY_LOCALE_END_JOB
- PRO_SEASON_LOAD_START_JOB
- PRO_OPT_RULE_END_JOB
- PRO_PLAN_PROMOTION_LOAD_JOB
- PRO_SEASON_STG_JOB
- PRO_PLAN_PROMOTION_LOAD_END_JOB
- PRO_SEASON_PRODUCT_STG_JOB
- PRO_SEASON_PERIOD_STG_JOB
- PRO_SEASON_CURR_OPT_METRIC_STG_JOB
- PRO_SEASON_PROD_MKDN_EDT_STG_JOB
- PRO_SEASON_LOAD_JOB
- PRO_SEASON_PRODUCT_LOAD_JOB
- RSE_BUDGET_START_JOB
- RSE_BUDGET_ALLOC_SETUP_JOB
- RSE_BUDGET_ALLOC_PROCESS_JOB
- RSE_BUDGET_END_JOB
- PRO_SEASON_PERIOD_LOAD_JOB
- PRO_SEASON_CURR_OPT_METRIC_LOAD_JOB
- PRO_SEASON_CURR_OPT_METRIC_SETUP_JOB
- PRO_SEASON_PROD_MKDN_EDT_LOAD_JOB
- PRO_SEASON_LOAD_END_JOB
- PRO_PROD_LOC_CDA_LOAD_END_JOB
- PRO_PRICE_LOAD_START_JOB
- RSE_WHSE_INV_ALLOC_LOAD_START_JOB

- PRO_PRICE_LOAD_SETUP_JOB
- RSE_WHSE_INV_ALLOC_STG_JOB
- PRO_PRICE_ELASTICITY_STG_JOB
- PRO_PRICE_LADDER_STG_JOB
- PRO_PRICE_LOAD_PROCESS_JOB
- PRO_PRICE_ELASTICITY_LOAD_JOB
- PRO_PRICE_LADDER_LOAD_JOB
- PRO_PRICE_LOAD_END_JOB
- RSE_WHSE_INV_ALLOC_LOAD_JOB
- RSE_WHSE_INV_ALLOC_PROCESS_JOB
- RSE_WHSE_INV_ALLOC_LOAD_END_JOB
- PRO_INVENTORY_LOAD_START_JOB
- PRO_INVENTORY_LOAD_SETUP_JOB
- PRO_INVENTORY_LOAD_PROCESS_JOB
- PRO_INVENTORY_LOAD_END_JOB
- PRO_OPT_START_JOB
- PRO_OPT_CREATE_RUNS_JOB
- PRO_OPT_JOB
- PRO_OPT_END_JOB
- PRO_SEASON_CURR_OPT_METRIC_PROCESS_JOB

Export and innovation workbench jobs that export the recommendations as well as grant permissions for objects in IW are:

- RSE_IW_GRANT_START_JOB
- RSE_IW_GRANT_JOB
- RSE_IW_GRANT_END_JOB
- RSE_EXPORT_START_START_JOB
- RSE_EXPORT_START_END_JOB
- PRO_EXPORT_START_JOB
- RSE_EXPORT_PREP_START_JOB
- PRO_EXPORT_PRICE_RECOM_JOB
- RSE_DAILY_BATCH_EXPORT_PREP_JOB
- PRO_EXPORT_RUN_RECOM_OPT_RESULT_JOB
- RSE_WEEKLY_BATCH_EXPORT_PREP_JOB
- PRO_EXPORT_RUN_TO_RESULT_JOB
- RSE_EXPORT_END_START_JOB
- RSE_QUARTERLY_BATCH_EXPORT_PREP_JOB
- PRO_TO_EXPORT_PRICE_RECOM_JOB

- RSE_EXPORT_END_END_JOB
- RSE_EXPORT_PREP_END_JOB
- PRO_EXPORT_END_JOB
- RSE_POST_EXPORT_DAILY_START_JOB
- RSE_POST_EXPORT_WEEKLY_START_JOB
- RSE_POST_EXPORT_DAILY_RESET_JOB
- RSE_POST_EXPORT_WEEKLY_RESET_JOB
- RSE_POST_EXPORT_DAILY_ZIP_JOB
- RSE_POST_EXPORT_WEEKLY_ZIP_JOB
- RSE_POST_EXPORT_WEEKLY_COPY_JOB
- RSE_POST_EXPORT_WEEKLY_FTP_JOB
- RSE_POST_EXPORT_WEEKLY_END_JOB
- RSE_POST_EXPORT_DAILY_COPY_JOB
- RSE_POST_EXPORT_DAILY_FTP_JOB
- RSE_POST_EXPORT_DAILY_END_JOB

Forecasting Science

Offer optimization forecasting has two modules, parameter estimation and demand forecasting. Parameter estimation uses the historical data to determine the seasonality, markdown elasticity, promotion elasticity, and promotion lift (or holiday lift) values. Demand forecasting leverages the parameters estimated from historical data and in-season sales data (the base demand is re-estimated on a regular basis as part of batch process) to determine the demand and generate the forecast.

Offer optimization forecasting supports the following features:

- Separate elasticity values for promotions and markdowns
- Customer segment level parameter values for seasonality, elasticity, and traffic lift for promotion and holidays
- The weather effects from historical data period can be incorporated into the estimation of parameters to remove the bias introduces by weather on forecasted sales.
- The ability to execute the following analysis:
  - Day of the week and time of the day profiles
  - Returns analysis

For both parameter estimation and demand forecasting stages, you can set up runs and configure various settings using the Manage Forecast Configurations functionality under Strategy & Policy Management. Based on the configuration settings, a batch job is used to initially estimate the parameters and then update them at specified intervals to reflect the latest sales trends. Similarly, a weekly batch job is used to generate demand forecasts every week after the weekly data has been updated.

Parameter Estimation

Parameter estimation consists of the following stages. Each stage has a series of configurable parameters and the corresponding default values. Modify the default values appropriately for

each retailer. You can modify these settings using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Overview of the Parameter Estimation Stages

The following sections describe the details of the settings in each stage.

1. Data preparation. Defines the levels to which data is aggregated on merchandise, location, customer segment, and time dimensions and the duration of the data used for parameter estimation.
2. Preprocessing. Filters the historical data and makes the first determination of item eligibility.
3. Elasticity. Determines the price elasticity for markdowns and promotions.
4. Seasonality. Estimates seasonality trends and identifies partitions with reliable seasonality.
5. Promotion lift. Estimates the traffic lift during promotion and holiday events. Corrects the seasonality curves to remove the traffic lift from promotion and holiday events.
6. Output. Generates parameter files in the format required by demand forecasting and offer optimization.

The parameter estimation stage generates markdown elasticity, promotion elasticity, seasonality and promotion lift values. Together they are referred to as demand parameters.

Parameter estimation requires at least 20 months of data. These requirements exist for the following reasons.

- The data from the beginning 4-6 weeks of the historical period must be removed from the data analysis so that the items that have truly started selling after the beginning of historical data period can be identified.
- The full life cycle data is required for all fashion/seasonal merchandise introduced during a fiscal year in order to estimate the seasonality curve for merchandise starting in every fiscal month. The least amount of data required is one year plus the typical season length of the merchandise.

Data Preparation

The data preparation stage defines the aggregation levels on merchandise, location, customer-segment, calendar dimensions, and duration of the data used for parameter estimation.

In addition to the above, configuration of the hierarchy level at which promotion data is processed is completed here.

Data Aggregation Sales data, inventory data, and returns data are aggregated on the following dimensions: Merchandise, Location, Price Zone (if available), Customer Segment (if available), and Calendar. The hierarchy type ID corresponding to each dimension is configured using the following parameters in the RSE_CONFIG table. Use the Manage System Configurations functionality to set the appropriate ID for each dimension.

The level of aggregation for each dimension can be configured while setting up the forecast run types in Manage Forecast Configurations. In addition, while creating a run type, the user can indicate whether Price Zone and Customer Segment should be used as additional dimensions in the aggregation process.

- Merchandise (PMO_PROD_HIER_TYPE). Multiple hierarchy types are available for Merchandise, Product, and Extended Product hierarchies. Choose the ID corresponding to the appropriate product hierarchy. Typically, data is aggregated to Style/ Style-Color level on the merchandise hierarchy. Select the appropriate merchandise level when setting up the forecast run type in Mange Forecast Configurations.

- Location (PMO_LOC_HIER_TYPE). Multiple hierarchy types are available for location, for example, Location Hierarchy and Trade Area Hierarchy. Choose the ID corresponding to the appropriate location hierarchy. Typically, data is aggregated to Store Cluster/Price Zone level. Choose the appropriate level. Select the appropriate location level when setting up the forecast run type in Manage Forecast Configurations.
- Customer Segment (PMO_CUST_SEG_HIER_TYPE). Choose the hierarchy type ID associated with the customer segment. By default, customer segment data is aggregated at two levels, by individual customer segments and Segment-All, which includes all the segments. The user does not have the option to select the aggregation level by customer segment.
- Calendar (PMO_FISCAL_CAL_HIER_TYPE). Choose the hierarchy type ID associated with the Calendar hierarchy.

Table 12–6 Data Aggregation

Setting Name and Description	Default Value	Range of Values
Merchandise Hierarchy Type ID: ID corresponding to the merchandise hierarchy used for aggregation of sales, inventory, and returns data. PMO_PROD_HIER_TYPE	3	1 - 15
Location Hierarchy Type ID: ID corresponding to the location hierarchy used for aggregation of sales, inventory, and returns data. PMO_LOC_HIER_TYPE	2	1 - 15
Customer Segment Hierarchy Type ID: ID corresponding to the customer segment hierarchy used for aggregation of sales, inventory, and returns data. PMO_CUST_SEG_HIER_TYPE	4	1 - 15
Calendar Hierarchy Type ID: ID corresponding to the calendar used for aggregation of sales, inventory, and returns data. PMO_FISCAL_CAL_HIER_TYPE	11	1 - 15
Merchandise Level: Level in the merchandise hierarchy at which data is aggregated for sales, inventory, and returns. Typically, data is aggregated to STYLE-COLOR. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	8	1 - 9
Location Level: Level in the location hierarchy at which data is aggregated for sales, inventory, and returns. Typically, data is aggregated to STORE CLUSTER. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	5	1 - 6
Calendar Level: Level in the calendar hierarchy at which data is aggregated for sales, inventory, and returns. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	4	1 - 5
Keep temporary tables: Turn this setting on to retain the intermediate tables generated during the parameter estimation process. PMO_KEEP_TEMP_TABLES	N	Y/N

Promotion Data Processing Level Promotions data can be present at various levels in the merchandise and location hierarchy. Choose the level at which promotions are defined for estimating traffic lift values. For estimating traffic lifts, we consider major events which drive significant traffic across the chain. For example: there events are defined at country-banner level.

Table 12–7 Hierarchy Levels Selection

Setting Name and Description	Default Value	Range of Values
Promotion processing level Merchandise. Choose the ID corresponding to the merchandise level at which traffic driving promotions are defined. Usually this is Company/Banner if there are multiple banners within the company. PMO_MIN_LOC_HIER_PROCESSING_LVL	2	1 - 9
Promotion processing level - Location: Choose the ID corresponding to the location level at which traffic driving promotions are defined. Usually this is Chain/Country level. PMO_MIN_PROD_HIER_PROCESSING_LVL	2	1 - 6

Data Validation Once the data preparation stage is complete, prepare data validation report and review it with the customer to confirm the data loaded matches with their expectations. Include the following in the data validation report.

- Summary of merchandise, location, customer segment, calendar hierarchies
- Sales units and amount by year at Chain and Division levels
- Summary metrics by Department-Price zone
- Sales, Inventory and Product count trends by Fiscal week and Fiscal year at Country-Banner level
- For the following data fields Ticket price, sales price, Item cost, Sales units, Sales amount, Inventory
  - Search for negative and null values
  - Significant percent of records with either negative or null values. Flag for review with customer.

Validating data with the customer is key to identifying potential data related problems early on. Investigate any potential data issues and work with the customer to resolve them. This ensures that the following stages are using the appropriate data.

Preprocessing

The Preprocessing stage filters the historical data to produce a subset of data that produces reliable demand parameters. The Preprocessing stage applies filters at the partition and week level. It performs the initial pruning of bad activity data. It does the first stage of determining eligibility and calculates certain values that can later be used in the calculation of elasticity and seasonality parameters. You can modify the configuration settings for the preprocessing stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Data Filters Week Level Table 12–8 shows the various week level filters used in the Preprocessing stage. These filters are used to remove individual weekly records of data that do not meet the requirement defined as specified for each filter.

Table 12–8 Data Filters Week Level

Filter Name and Description	Default Value	Range of Values
Store count greater than 0. Used to filter activities with null or zero store count. IWF_STORE_GTR_ZERO	N	Y/N
SKU store count greater than 0. Used to filter activities with null or zero sku store count. IWF_SKU_STORE_GTR_ZERO	N	N/Y
Inventory data present. Used to indicate that the inventory data is available and reliable. If inventory data is available and reliable, then Preprocessing will use inventory data for filtering and other calculations. If inventory data is not available or unreliable, then it is ignored. IWF_INV_DATA_PRESENT	Y	N/Y
Sales unit threshold. Used to filter sales data records with sales units below threshold values. IWF_SALES_UNIT_THRESHOLD	1	Greater than or equal to 1.
Inventory threshold. Used to filter sales data records with inventory units below threshold values. IWF_INV_THRESHOLD	1	Greater than or equal to 0.
Use store count for Inventory threshold. Defines the criteria used for inventory threshold. Setting to Y will use Store count with inventory as inventory threshold criteria instead of actual inventory value. IWF_SKU_STORE_COUNT	N	Y/N
Threshold combination. Use AND/OR combination for Sales unit threshold and Inventory threshold. IWF_FILTER_COMBO	AND	AND/OR
Life cycle sell thru %. The item start and end dates are calculated using percentage sell through. The start date is the date when Life cycle sell through % (Start) is reached. The end date is the date when Life cycle sell through % (End) is reached. Life cycle sell through % (Start) is expressed relative to 0% so entering 2 means that the start date is when 2% of total sales has been achieved. Life cycle sell through % (End) is expressed relative to 100% so entering 2 means that the end date is when (100-2)% i.e., 98%, of total sales has been achieved. IWF_LOW_LCYCLE_SELL_PCT and IWF_HIGH_LCYCLE_SELL_PCT	Start: 2.0 End: 2.0	Start: greater than or equal to 0; less than or equal to 50. End: greater than or equal to 0; less than or equal to 50.
Relative price. Relative price thresholds are used to filter out item weeks with a ratio of sales price to maximum ticket price that fall outside the specified range for the start date and the end date. IWF_LOW_RELATIVE_PRICE and IWF_HIGH_RELATIVE_PRICE	Start: 0.2 End: 1.0	Start: greater than 0. End: greater than the start value.

Partition Filters These filters are used to exclude all the weekly records from a Merchandise-Location-Customer segment partition at the aggregation level, for partitions with values that do not meet the requirement specified for each filter, as described below.

Table 12–9 Partition Filters

Filter Name and Description	Default Value	Range of Values
Minimum number of eligible weeks. Aa certain number of weeks are necessary in order to determine item eligibility. IF_MIN_ELIGIBLE_WEEKS	6	Greater than 0.
Minimum season (weeks). A certain season length (the number of weeks between the first and last activity) is required in order to determine item eligibility. IF_MIN_SEASON_WEEKS	6	Greater than 0.
Minimum sales units. The total number of units sold must be at least this value. IF_MIN_SALES_UNITS	10	Greater than 0.
Fraction of eligible weeks. The percentage of eligible weeks, expressed as a fraction of the season length. The season length is the number of weeks between the start and end dates. (See Life cycle sell thru % above.) IF_FRACTION_ELIGIBLE_WEEK	0.6	Greater than 0.0; less than or equal to 1.0.

Elasticity

The Elasticity stage determines the promotion and markdown elasticity values at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the elasticity value by customer segment. You can modify the configuration settings for the elasticity stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

In this stage, parameters must be set for

- Additional data filtering to reduce the noise in the elasticity estimation
- Merchandise and location hierarchy levels for which parameters are estimated
- Identifying markdowns weeks
- Identifying promotion weeks
- Reliability thresholds for elasticity estimation
- Transformation of elasticity values

Data Filters Weekly Additional data filters are applied during the estimation of elasticity. These filters are used to reduce the noise from very deep price cuts or price cuts towards the end of life. An option is also available to the limit the data used for elasticity estimation.

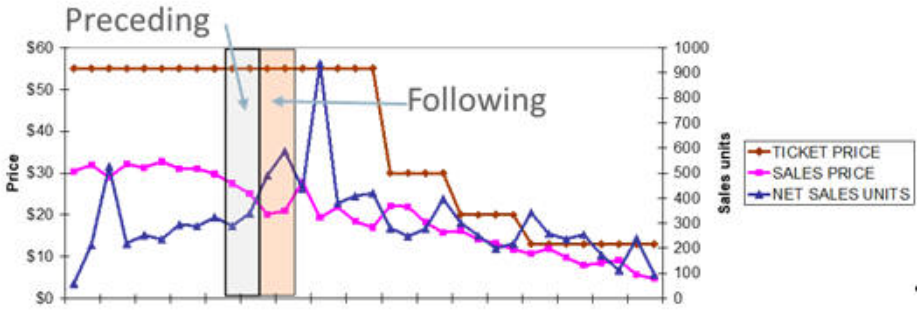
Table 12–10 Data Filters Weekly

Filter Name and Description	Default Value	Range of Values
Relative price. Used to filter out item-weeks with a ratio of sales price to maximum ticket price that falls outside the specified range. WDF_RELPRICE_LOW and WDF_RELPRICE_HIGH	Low: 0.2 High: 1.0	Low > = 0; High > low value
Relative inventory. The upper and lower bounds for the value for inventory relative to maximum inventory. WDF_PCTSALES_LOW and WDF_PCTSALES_HIGH	Low: 0.2 High: 1.0	Greater than or equal to 0. Low > = 0 High > low value
Range filter. Used to eliminate unreliable data using start date and end date for the period. Both the start date and the end date are Null by default. A Null value means that the field is not used in the filter. One or both of the fields can have a value of Null. If both of the fields are Null, then the data is not filtered. WDF_RANGEFILTER_MINCALWK and WDF_RANGEFILTER_MAXCALWK	Start date: Null End date: Null	-

Hierarchy Levels Selection Select the merchandise and location levels for which parameter estimation should calculate demand parameters. Parameter estimation calculates demand parameters for the partitions of the levels that are the cross-products of all the levels you select. The highest level is set based on PMO_MIN_LOC_HIER_PROCESSING_LVL (typically, Banner) and PMO_MIN_PROD_HIER_PROCESSING_LVL (typically, Department). The lowest level is set based on the aggregation levels that the user selects while creating the forecast run types. In Manage Forecast Configurations, the user can select from the list of levels that are between the highest and lowest levels.

Markdowns The parameters described below are used to identify markdowns in the aggregated weekly sales data. During a markdown window, the application is interested in a drop in prices from the preceding window to the following window. During a promotion, the application is looking for similar prices during the preceding and following window but a drop in prices during the promotion window. Similar parameters are used to identify Promotions as well.

Figure 12–4 Markdown Window



Time window defines the weeks before and after the markdown. The preceding weeks value is the number of weeks before the markdown occurred. The following weeks value is the number of weeks after the markdown and includes the week of the markdown. A week is a calendar

week. A time window that satisfies all the markdown parameter settings is classified as a markdown window.

Minimum eligible weeks is the minimum of the actual number of weeks in the Time Window that have data. The Time Window is in calendar weeks, and not every calendar week actually has sales data. So the actual number of weeks with data that are within the Time Window can actually be smaller than the Time Window.

Maximum deviation is the deviation of the sales price in the weeks before and after the markdown. Provides stability on the variance.

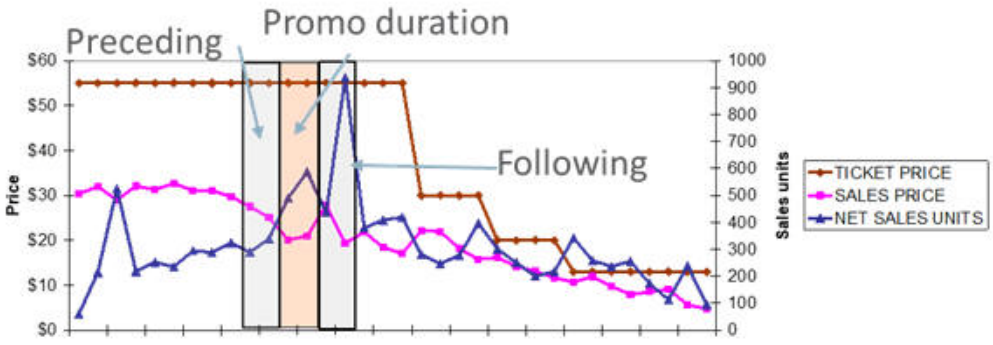
Min Markdown depth is the drop in average sales price from preceding weeks to the following weeks is higher than this threshold value. Markdown depth = 1- (Avg Price following weeks/Avg Price preceding weeks). In other words, this parameter controls how much of a price decrease is required in order for the price decrease to count as a markdown.

Table 12–11 Markdown Parameters

Name and Description	Default Value	Range of Values
Time Window MKDN_TIMEWINDOW_PRECEEDING and MKDN_TIMEWINDOW_FOLLOWING	Preceding: 2 Following: 2	1–3 1–3
Eligible Weeks MKDN_ELIGWKS_PREC and MKDN_ELIGWKS_FOLL	Preceding: 2 Following: 2	1–3 1–3
Maximum Deviation MKDN_MAXDVT_PRECEEDING and MKDN_MAXDVT_FOLLOWING	Preceding: 0.1 Following: 0.1	0–1 0–1
Min Markdown Depth MKDN_MINMKDN_DEPTH	0.1	0–1
Use Ticket Price: Both Ticket price and sales price values from the preceding and following weeks satisfy min markdown depth criteria. USE_TICKET_PRICE	N	Y/N

Promotion One Week For identifying promotions, a few additional parameters are defined below in addition to Time window, Eligible weeks, and maximum deviation, which are defined above.

Figure 12–5 Promotion Window



Min Promotion depth is the drop in average sales price during the week of promotion compared to preceding/following weeks. In other words, this parameter controls how much of a price decrease is required in order for the price decrease to count as a promotion.

Time window defines the weeks before promotion, during promotions, and after the promotion. The preceding weeks value is the number of weeks before the markdown occurred. The following weeks value is the number of weeks after the markdown and includes the week of the markdown. A week is a calendar week. Time window that satisfies all the promotion parameter settings is classified as a promotion window.

Promotion depth = 1- (Avg. Price during promotion weeks/ Avg. price during preceding and following weeks).

Time Window - Promo duration: Duration of the promotion event, this is set to 1 for 1-week promotion.

Max deviation - Preceding/Following: Avg. price deviation across preceding and following weeks must be less than this threshold. This ensures that the sales price is similar in the weeks before and after the promotion.

Table 12–12 Promotion Parameters

Name and Description	Default Value	Range of Values
Time Window PRWK1_TIMEWINDOW_PRECEEDING, PRWK1_TIMEWINDOW_FOLLOWING and PRWK1_TIMEWINDOW_PROMOWEEKS	Preceding: 1 Following: 1 Promo Duration: 1	1–3 1–3 1
Eligible Weeks PRWK1_ELIGWKS_PREC and PRWK1_ELIGWKS_FOLL	Preceding: 1 Following: 1	1–3 1–3
Maximum Deviation PRWK1_MAXDVT_PRECEEDING, PRWK1_MAXDVT_FOLLOWING and PRWK1_MAXDVT_PRECFOLLOW	Preceding: 0.1 Following: 0.1 Preceding/Following: 0.1	0–1 0–1 0–1
Min Promotion Depth PRWK1_MINPROMO_DEPTH	0.1	0–1

Promotion Two Week Promotion duration parameter is set to 2 for identifying two week promotions. Promotion parameters for one-week and two-week promotions can be set to different values.

Table 12–13 Promotion Two Weeks

Name and Description	Default Value	Range of Values
Time Window PRWK2_TIMEWINDOW_PRECEEDING, PRWK2_TIMEWINDOW_FOLLOWING and PRWK2_TIMEWINDOW_PROMOWEEKS	Preceding: 1 Following: 1 Promo Duration: 2	1–3 1–3 2
Eligible Weeks PRWK2_ELIGWKS_PREC and PRWK2_ELIGWKS_FOLL	Preceding: 1 Following: 1	1–3 1–3

Table 12–13 (Cont.) Promotion Two Weeks

Name and Description	Default Value	Range of Values
Maximum Deviation PRWK2_MAXDVT_PRECEEDING, PRWK2_MAXDVT_FOLLOWING and PRWK2_MAXDVT_PRECFOLLOW	Preceding: 0.1 Following: 0.1 Preceding/Following: 0.1	0–1 0–1 0–1
Min Promotion Depth PRWK2_MINPROMO_DEPTH	0.1	0–1

Reliability Settings Determine the partitions with reliable elasticity values.

Table 12–14 Reliability Settings

Name and Description	Default Value	Range of Values
Outlier threshold. Percentile threshold for removing outlier data points from elasticity estimation. Eg., Threshold is set to X%, Top X Percentile and Bottom X percentile sales ratio and price ratio data points are excluded from elasticity estimation. RS_OUTLIER_THRESHOLD	0.05	0–0.05
Eligible items threshold - Elasticity RS_ELIGIBLEITEMS_THRESHOLD	25	Greater than 1.
Std. Error threshold RS_STDERROR_THRESHOLD	0.3	0–1

Transformation Transform the elasticity values obtained after the pruning based on reliability to cap the very low or very high elasticity values. The transformation settings also enable the user to shift the elasticity values to a desired range.

Table 12–15 Transformation

Name and Description	Default Value	Range of Values
Transform Percentile - Low TRANSPERCENTILE_LOW	0.1	0–1
Transform Percentile - High TRANSPERCENTILE_HIGH	0.9	0–1
Elasticity range - Min: Select a value closer to the transform percentile - Low and higher than 1.2 ELASTICITYRANGE_MIN	1.3	Greater than 1.
Elasticity range - Max: Select a value closer to the transform percentile - high. ELASTICITYRANGE_MAX	2.8	Greater than elasticity range–min.

Seasonality

The Seasonality stage determines the seasonality values at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the seasonality value by customer segment. You can modify the configuration settings for the seasonality stage using the Manage Forecast Configuration functionality under Strategy & Policy Management.

In this stage, parameters must be set for

- Seasonality curve set up
- Reliability filters

Seasonality Curve Setup. The Seasonality Curve Setup stage determines the season codes used for building seasonality curves, length of the seasonality curve, item count threshold, padding curve weight, and seasonality coverage of the final curves.

Season codes create additional partitioning in the dataset by introducing a time dimension. This stage can be used to partition seasonality curves by fiscal start month, fiscal start week, for example, Partition the Class-Store cluster-Customer segment data based on the merchandise time of introduction. Group merchandise based on the fiscal month of the start date. This ensures items starting in the same month receive the same season code. Weekly curves are useful for modeling short life cycle merchandise, while monthly curves are useful for medium to long life cycle curves.

Length of the seasonality curve: Seasonality curves are generated for 52 weeks. They can also be generated for longer duration if the merchandise life cycle length is longer. This also increases the duration of historical data required to prepare the seasonality curves.

Actual sales for a given merchandise-location-customer segment-Start month can be present for less than 52 weeks. To generate a curve for the entire 52 weeks duration, a padding curve is used.

Padding curve: The padding seasonality curve is determined by creating a basic curve for the highest merchandise/location partition. The final seasonality curve is calculated as weight * padding curve + (1 - weight) * seasonality curve.

Table 12–16 Seasonality Curve Setup

Name and Description	Default Value	Range of Values
Curve Type. Determines the duration of time partition based on fiscal week or fiscal month. CURVE_TYPE	Monthly	Monthly or Weekly
Basic Curve. Basic curves are generated by ignoring the partitioning on time dimension for a given merchandise-location-customer segment partition. BASIC_CURVE	Y	True or FalseY/N
Seasonality curve length. Determines the length of the final seasonality curves for regular (non-basic) season codes. The length from the start determines how many weeks after the start date are in the curve. SEASON_CURVE_LENGTH	52	Greater than or equal to 6.
Eligible items threshold. The minimum number of items that a Merchandise Hierarchy/Location Hierarchy/Customer Segment/Season Code partition must contain so that Raw-AP can produce a seasonality curve for the partition. ELIGIBLE_ITEM_THRESHOLD	5	Greater than or equal to 1.
Padding curve weight. The value used in determining the final seasonality curve. PADDING_CURVE_WEIGHT	0.4	Greater than 0.0; less than 1.0.

Table 12–16 (Cont.) Seasonality Curve Setup

Name and Description	Default Value	Range of Values
Basic curve start week. First week of the basic curve. This is represented as number of weeks from the most recent week with data. Should be greater than 52 and less than number of weeks of historical data.	57	52 - 80
Seasonality Coverage. First Fiscal year SEASON_COVERAGE_FIRST_FY	Latest fiscal year with historical data	Greater than or equal to 2020
Seasonality coverage. Last fiscal year SEASON_COVERAGE_LAST_FY	Latest fiscal year with historical data + 2.	Greater than or equal to 2020.

Reliability Filters The filters in this stage are used to identify reliable seasonality curves based on the number of items in the partition, average sales volume, and season length threshold.

Table 12–17 Reliability Filters

Name and Description	Default Value	Range of Values
Keep top level curves. All the highest level curves are kept, regardless of threshold values. KEEP_TOP_LEVEL_CURVES	Y	Y/N
Prune curves with missing dates. Pruning of basic curves with missing dates is permitted. PRUNE_CURVES_MISSING_DATES	Y	Y/N
Eligible items threshold. Partitions with fewer numbers of eligible items than the threshold value are removed. Eligibility is defined during preprocessing.	5	Greater than or equal to 0.0.
Average weekly sales threshold. Partitions with average weekly sales below the threshold are removed. Weekly sales are the sum of all sales for all activities for a given week. AVG_WK_SALES_THRESHOLD	25	Greater than or equal to o.
Raw seasonality length threshold. Curves can be discarded when the number of weeks from the first non-zero seasonality value to the last non-zero seasonality value is less than the threshold. In the seasonality stage, it is possible that many seasonality values of zero have been added to the curves. Note that basic season codes have a length of 53, so picking a value greater than 53 will prune out any basic season codes. Seasonality curves before padding are referred to as raw seasonality curves. RAW_SEASONALITY_LEN_THRESHOLD	10	Greater than or equal to 0.

Promotion

The promotion stage determines the traffic lift value associated with promotions and holiday events at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the promotion lift value by customer segment. A list of holidays and promotions, along with event start date and end date, is sent as a data feed.

In this stage, parameters must be set for determining the baseline to be used for estimation of lift values, Outlier threshold, min lift value. and max lift value. You can modify the configuration settings for the promotion stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Table 12–18 Promotion

Name and Description	Default Value	Range of Values
Baseline. The baseline type determines how to smooth the seasonality curve. If you select the linear option, parameter estimation looks at event effective start date and event effective end date and draws a straight line between them. Effective start date is the weekend prior to the event start date and effective end date is the second weekend after the event end date. PROM_LIFT_BASELINE_TYPE	Linear	Linear, Min, or Max.
Outlier percentile. Very high lift values are flagged. PROM_LIFT_OUTLIER_PERCENTILE	0.05	0–1
Lift value -High. Lift values flagged as outliers are capped to outlier percentile lift value. PROM_LIFT_HIGH	Y	Y/N
Lift value-min. Sets the threshold for min lift value. Any promotions with lift value lower than this threshold are not assigned any lift value. PROM_LIFT_MIN_VALUE	1.05	Greater than or equal to 1.

Output and Review of Parameters

The Propagation and Parameter Export to Target stages do not have separate settings. The Propagation stage populates the seasonality curves for all the years from the Seasonality coverage first fiscal year to the Seasonality coverage last fiscal year.

The Parameter Export to Target stage populates the parameters at the lowest level of estimation selected in the appropriate output tables. Offer Optimization typically expects the parameters at the lowest level of estimation. For example, Offer Optimization expects the demand parameters at Class-Store cluster level. Use escalation to fill in the parameters missing at Class-Store level. The Output stage escalates on location hierarchy first and then on merchandise hierarchy.

Run the parameter estimation by starting the batch process during the implementation phase. Details on the batch processing are covered in "Batch Processing". Perform the following checks after each stage in the parameter estimation is complete.

Elasticity (pmo_elasticity_parameters)

- Histogram of elasticity values, promotion vs markdown: Distribution of elasticity values follows the expected trend. Typically follow a normal distribution around the median elasticity value.
- Elasticity escalation: Percentage of partitions receiving elasticity values from higher level.

Seasonality (pmo_seasonality_parameters; pmo_seasonality_curve_tbl)

- Review number of partitions impacted by reliability filters:
- Visually check the top-level seasonality curves
- Seasonality escalation: Percentage of partitions receiving elasticity values from higher levels.

Promotion lift (pmo_promtion_lift)

- Plot histogram of promotion lift values by event and compare across segments.

Day Level and Returns Metrics

This section covers the settings used for determining day of the week profiles, time of the day profiles, return rate, and average time to return.

Day of the Week Profiles

Day of the week profiles determine the relative sales strength across various days in a given week. These profiles are used to spread the forecasted demand to the day level. In order to capture the variation across merchandise, location, segment, and calendar dimension, the configuration on each of the four dimensions is supported. In the current version, you can choose two configurable levels for estimating day of the week profiles. Level 1 corresponds to the estimation level at which profiles are calculated, and Level 2 profiles correspond to the escalated profiles. They are only used when profiles are not available at the estimation level.

Table 12–19 Day of the Week Profiles

Name and Description	Default Value	Range of Values
Merchandise level. ID corresponding to the merchandise level at which day of the week profiles are calculated. Usually at DEPARTMENT or higher. RSE_PMO_DLYPROF_MH_LVL_SQC1	4	1 - 9
Location level. ID corresponding to the location level at which day of the week profiles are calculated. Usually STORE CLUSTER or higher. RSE_PMO_DLYPROF_LH_LVL_SQC1	4	1 - 6
Merchandise level. Merchandise level - Escalated: ID corresponding to the higher merchandise level at which day of the week profiles are calculated. Usually at Division or higher. RSE_PMO_DLYPROF_MH_LVL_SQC2	3	1 - 3
Location level. Location level - Escalated: ID corresponding to the higher location level at which day of the week profiles are calculated. Usually Region or higher. RSE_PMO_DLYPROF_LH_LVL_SQC2	3	1 - 3
Min week. Calendar hierarchy ID corresponding to the oldest week used for day of the week profile calculation. PMO_DLYPROF_MIN_WK	User determined	
Max Week. Calendar hierarchy id corresponding to the latest week used for day of the week profile calculation. PMO_DLYPROF_MAX_WK	User determined	
Week count threshold. Partitions with data for less than the week count threshold are excluded from day of the week profile calculation. PMO_DLYPROF_WK_CNT_THLD	2	> 1

Table 12–19 (Cont.) Day of the Week Profiles

Name and Description	Default Value	Range of Values
Sales threshold. Total sales from the merchandise, location, segment and calendar partition must be higher than this threshold value for the partition to be eligible for estimation of day of the week profile. PMO_DLYPROF_TOT_SLS_THLD	1000	Greater than 100.
Output level - Merchandise. Day of the week profiles are generated at this merchandise level for export. Any missing levels are filled using the higher level profiles.	4	1 - 9
Output level - Location. Merchandise: Day of the week profiles are generated at this location level for export. Any missing levels are filled using the higher level profiles.	4	1 - 6

Returns Metrics

For returns parameter estimation, the data is divided into two parts, one for building the model and the other for testing the model. Usually, one full year is used to build the model, and at least 13 weeks are used for testing the model. The most recent data corresponding to the build and test durations is used for returns modeling from the available data. Returns are modeled at various locations in the merchandise hierarchy, typically, Class, Department, and Division. For each of the merchandise levels, modeling is done for both customer segment and segment all levels on the customer segment dimension and at the data aggregation level on the location dimension. After applying the reliability filters, the missing parameters at the lowest merchandise level - location: data aggregation level - customer segment level are filled by escalation.

The return rate is estimated at the data aggregation level for the Merchandise-Location-Customer Segment level every week. This information, along with the returns parameters and sales forecast, is used to generate the returns forecast.

Table 12–20 Returns Metrics

Name and Description	Default Value	Range of Values
Merchandise level 1. The topmost merchandise level used for returns parameters. Typically Division or Banner. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL1	2	1 - 6
Merchandise level 2. The intermediate merchandise level used for returns parameters. Typically Department. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL2	4	1 - 6
Customer segment level 1 Merchandise level 2. The lowest merchandise level used for returns parameters. Typically Class. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL3	5	1 - 6

Table 12–20 (Cont.) Returns Metrics

Name and Description	Default Value	Range of Values
Duration of data for Build. Number of weeks of data to be used for building the model. PMO_RETURN_BUILD_MODEL_WKS	52	52 - 80
Duration of data for Test: Number of weeks of data to be used for testing the model. Most recent weeks are used for testing the model. PMO_RETURN_TEST_MODEL_WKS	13	13 - 26
Sales threshold. Merchandise-Location-Customer segment partitions with total sales lower than the threshold are excluded from returns parameter estimation. PMO_RETURN_LOW_SALES_THRESHOLD	10	> 10
Model fit threshold. Merchandise-Location-Customer segment partitions with adjusted r square value lower than the threshold are pruned. This eliminates partitions where the model fit is poor. PMO_RETURN_RELIABLE_ADJST_R_SQR_THSLD	0.5	> 0.3
Data sufficiency threshold. Row count: Merchandise-Location-Customer segment partitions: Total with data less than the row count threshold are excluded from returns parameter estimation. PMO_RETURN_RELIABLE_ROW_COUNT_THSLD	500	>100

Demand Forecasting

The following multiplicative demand model is used as follows:

Demand = Base demand* Seasonality *Price effect* Promotion Traffic Lift*Store count* Other Effects

Base demand is updated in season every week at the Optimization recommendation level, for example, Style Color-Store cluster-Customer segment level. Demand forecasting leverages the parameters estimated from historical data and in-season sales data to generate a demand forecast. Elasticity for determining price effect is assigned using merchandise and location hierarchy. For assigning seasonality curve and promotion lift, the start date is required to determine the right partition on time dimension in addition to merchandise and location hierarchy. Demand forecasting set up involves configuration for start date and demand strategy.

Demand forecasting stage leverages the parameters estimated from historical data and in-season sales data to generate a demand forecast. Demand forecast values along with historical parameters are sent to Offer Optimization for generating optimal price recommendations and targeted offers.

In this stage, the following parameters must be set by the user to determine the model start date, season code and demand model settings. For all the parameters prefixed with RSE, you can modify the values in the RSE_CONFIG table using Manage System Configurations. For other parameters, use the Manage Forecast Configurations functionality under Strategy & Policy Management to modify parameters for Base Demand estimation.

Note that the merchandise/location/calendar hierarchy types do not need to be set for base demand separately, as they will be the same as the values set for PMO_PROD_HIER_TYPE,

PMO_LOC_HIER_TYPE, and PMO_FISCAL_CAL_HIER_TYPE. In addition, the merchandise/location/calendar levels for base demand estimation do not need to be set separately, as they will be the same as the aggregation levels that are selected when creating a run type (AGGR_PROD_HIER_LEVEL, AGGR_LOC_HIER_LEVEL, and AGGR_CAL_HIER_LEVEL).

Table 12–21 Demand Forecasting

Name and Description	Default Value	Range of Values
Merchandise level -Store Weights. The merchandise level at which store weights are calculated. Choose the ID corresponding to the merchandise level, typically at Sub Class. RSE_BD_SW_PROD_HIER_LEVEL	6	1 - 9
Start date - sell thru %. The earliest week ending date when a Style-Color - Store Cluster combination (data aggregation level used for generating forecast and price recommendations) achieves 2% sell thru is used to determine its start date. The start date is used to determine the seasonality curve assigned to a style color - store cluster combination. OPTPARAM_PCT_SLS_THRESHOLD	2%	0%–5%
Start date - max weeks from first sale. After the first sale date if the 2% sell thru is not achieved before this threshold then first sale date + max weeks from first sale is used to define the start date. OPTPARAM_WEEK_LIMIT	3	1–5
Seasonality - Max weeks from start. This setting is used to switch from a fashion curve to a basic curve if an item lives longer than expected. OPTPARAM_SEAS_CURVE_THRESHOLD	42	Greater than 35.
Demand Interval. Number of in-season weeks to be used for demand forecast-ing.0 implies the use of entire history from current season to generate the demand forecast. BD_PERIOD_LEN	4	0–6
Demand Interval - Poisson. Number of in-season weeks to be used for demand forecasting using Poisson demand strategy.0 implies use entire history from current season to generate the demand forecast. BD_POISSON_WKS	0	0 - 6
Alpha: Poisson. Smoothing parameter used for Poisson demand strategy. This parameter determines how quickly the weights decay for past weeks in-season data. BD_POISSON_PARAM	0.5	0–1
Demand Interval - Log Linear. Number of in-season weeks to be used for demand forecasting using Log linear demand strategy.0 implies use entire history from current season to generate the demand forecast. BD_EST_LOG_LINEAR	4	0 - 6

Table 12–21 (Cont.) Demand Forecasting

Name and Description	Default Value	Range of Values
Alpha - Log Linear. Smoothing parameter used for Log Linear demand strategy. This parameter determines how quickly the weights decay for past weeks in-season data. BD_LOG_LINEAR_PARAM	0.7	0 - 1
Base demand - Sales units Threshold. Threshold used for determining the base demand strategy to use for an item. Items with sales lower than the threshold will use Poisson demand strategy other item will use log linear demand strategy. BD_LL_SLS_THRESH_QTY	20	15 - 30
Base demand - weeks used for Threshold. Number of weeks of data used for calculating the Base demand - Sales units threshold. BD_AE_SLS_LEN_WKS	4	1 - 5
Store weights - Duration of data. Number of weeks of data used for calculation of store weights. BD_SW_SLS_LEN_WKS	13	13 - 52

New Stores

Retailers regularly open or close stores in different markets. The base demand forecasting process above accounts for stores that are operational. However, sometimes, retailers plan for new stores and would like the forecasting process to account for the additional increase (or decrease) in demand that will be generated due to the planned store opening (or closure). For example, a retailer might be planning to open a new store ten weeks in the future and would like to account for this in the demand forecasting process. OO provides this functionality in which the retailer must provide a like store through the RSE_LIKE_LOC_STG interface. Once this mapping is provided, the OO forecasting module adjusts the demand accordingly and the optimization considers the additional increase (or decrease) in demand for the new store (or closed store). Once the new store is operational, the sister store sales potential is discarded and it switches to the store's base demand forecasting process. This functionality helps retailers appropriately forecast demand based on planned store openings or closures. The following two examples illustrate the functionality.

Example 1: Sales at the Beginning of Week 2

Table 12–22 Sales at the Beginning of Week 2

Store	Week 1	Week 2	Week 3	Week 4	Week 5	Total Season
Store 1	100	120	100	120	100	540
Store 2	100	120	100	120	100	540
Store 3			Opens 50	50	80	180
Store 4		Opens 40	50	60	80	230
Total Sales	200	280	300	350	360	1490

Example 2: Sales at the Beginning of Week 4

Table 12–23 Sales at the Beginning of Week 4

Store	Week 1	Week 2	Week 3	Week 4	Week 5	Total Season
Store 1	150	150	140	140	140	720
Store 2	100	120	100	120	100	540
Store 3			Opened 100	100	160	360
Store 4		Opened 40	50	60	80	230
Total Sales	250	310	390	420	480	1810

Optimization Science

This section describes the business rules that are available and the science behind the optimization algorithms used in the Offer Optimization application. It does not provide all the details of the algorithm. However, it does provide some guidance so that you can troubleshoot and resolve issues quickly during an implementation.

Business Rules

The implementation of Offer Optimization is determined by a retailer's individual business requirements. A complete and accurate configuration of business rules is essential to the accuracy and success of the recommendations and forecasts.

Some of the business requirements or rules include:

- The retailer's pricing strategy for promotions, markdowns, and targeted offers. Does the retailer want to maximize the revenue or profit margin?
- The level of the merchandise hierarchy and of the location hierarchy at which the sales data will be provided.
- The day of the week that markdowns are effective. Typical start and end date for promotions during a week.
- Whether markdowns are effective on the same day or different days for all departments.
- Whether promotions, budget, and distribution center data will be included in the data feed.
- How eligible items (items to be processed by OO in a given week) are defined (for example, if an item exists in the most recent sales data feed and has a threshold number of units on hand).
- The kinds of inventory that must be considered by the batch process for optimization.

It is essential that considerable time be spent on understanding the business rules so that the global defaults can be populated in RSE_CONFIG with APPL_CODE ='PRO' and so most of the departments will not require further tweaking of the batch runs or for ad hoc runs. The configuration that must be created in conjunction with the business requirements are:

Table 12–24 Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
DEFAULT_APPL_USER	OO_BATCH_USR	User identifier to be used for batch activities that require user tracking.
PRO_PROD_HIER_TYPE	3	The hierarchy ID to use for the product (Installation configuration).
PRO_LOC_HIER_TYPE	2	The hierarchy ID to use for the location (Installation configuration).
PRO_CAL_HIER_TYPE	11	The hierarchy ID to use for the calendar (Installation configuration).
PRO_CUSTSEG_HIER_ TYPE	4	The hierarchy ID to use for the customer segments (Installation configuration).
PRO_CAL_HIER_ PROCESSING_LVL	4	The calendar hierarchy level at which PRO will define RUNs for optimization.
PRO_PROD_HIER_RUN_ SETUP_LVL	4	The merchandise hierarchy level at which PRO will setup/create RUNs.
PRO_PROD_HIER_ PROCESSING_LVL	5	The merchandise hierarchy level at which PRO will optimize RUNs.
PRO_LOC_HIER_ PROCESSING_LVL	2	The location hierarchy level at which PRO will define RUNs for optimization.
PRO_CUST_HIER_ PROCESSING_LVL	2	Default customer segment level at which PRO will define RUNs for optimization.
PRO_OPT_LOC_REC_LVL	2	Default location level at which price recommendations will be generated.
PRO_OPT_CUST_REC_LVL	1	Default customer segment level at which price recommendations will be generated (whole population).
PRO_OPT_TIME_REC_LVL	4	Default calendar level at which price recommendations will be generated (week).
PRO_OPT_MERCH_REC_ LVL	9	Default merchandise level at which price recommendations will be generated (STYLE-COLOR).
PRO_OPT_MECH_REC	NONE	Default targeted offer mechanics for which price recommendations will be generated.
PRO_OPT_MKTG_REC	NONE	Default targeted offer marketing aspect for which price recommendations will be generated.

Table 12–24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_SALVAGE_VALUE	0	Salvage value is the value of the product after the season ends. default value is 0.
PRO_TR_NO_TOUCH_ AFTER_LANDING	0	Default no touch after landing (2 weeks).
PRO_TR_LENGTH_OF_ PROMOTIONS	0.6	Default length of promotion as a percentage of the whole season.
PRO_TR_MAX_LENGTH_ OF_PROMOTION	1	Default maximum length (weeks) of a promotion.
PRO_TR_LENGTH_OF_ MKDN	0.4	Default length of markdown as a percentage of the whole season.
PRO_TR_NO_TOUCH_END_ LIFE	0	Default no touch at the end of life (1.5 weeks).
PRO_PR_FIRST_PROMO_ MIN_DISC_PCT	0	Default minimum discount percentage for the first promotion.
PRO_PR_FIRST_PROMO_ MAX_DISC_PCT	1	Default maximum discount percentage for the first promotion.
PRO_PR_OTHER_PROMO_ MIN_DISC_PCT	0	Default minimum discount percentage for promotions other than the first one.
PRO_PR_OTHER_PROMO_ MAX_DISC_PCT	1	Default maximum discount percentage for promotions other than the first one.
PRO_PR_MIN_TIME_ BETWEEN_PROMOS	1	Default minimum time separation between any two consecutive promotions (1 week).
PRO_PR_PROMO_START_ DAY	MONDAY	Day of the week promotions start
PRO_PR_PROMO_END_ DAY	SUNDAY	Day of the week promotions end
PRO_PR_DAY1_WGT	0.14	Promotion weight for day 1 of the week
PRO_PR_DAY2_WGT	0.14	Promotion weight for day 2 of the week
PRO_PR_DAY3_WGT	0.14	Promotion weight for day 3 of the week
PRO_PR_DAY4_WGT	0.14	Promotion weight for day 4 of the week
PRO_PR_DAY5_WGT	0.14	Promotion weight for day 5 of the week
PRO_PR_DAY6_WGT	0.15	Promotion weight for day 6 of the week
PRO_PR_DAY7_WGT	0.15	Promotion weight for day 7 of the week

Table 12–24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_MR_FIRST_MKDN_ MIN_DISC_PCT	0	Default minimum discount percentage for the first markdown.
PRO_MR_FIRST_MKDN_ MAX_DISC_PCT	1	Default maximum discount percentage for the first markdown.
PRO_MR_OTHER_MKDN_ MIN_DISC_PCT	0	Default minimum discount percentage for markdowns other than the first one.
PRO_MR_OTHER_MKDN_ MAX_DISC_PCT	1	Default maximum discount percentage for markdowns other than the first one.
PRO_MR_MIN_TIME_ BETWEEN_MKDN	1	Default minimum time separation between any two consecutive markdowns (1 week).
PRO_MR_MKDN_START_ DAY	MONDAY	Day of the week markdown start
PRO_MR_MKDN_END_ DAY	SUNDAY	Day of the week markdown end
PRO_MR_DAY1_WGT	0.14	Markdown weight for day 1 of the week
PRO_MR_DAY2_WGT	0.14	Markdown weight for day 2 of the week
PRO_MR_DAY3_WGT	0.14	Markdown weight for day 3 of the week
PRO_MR_DAY4_WGT	0.14	Markdown weight for day 4 of the week
PRO_MR_DAY5_WGT	0.14	Markdown weight for day 5 of the week
PRO_MR_DAY6_WGT	0.15	Markdown weight for day 6 of the week
PRO_MR_DAY7_WGT	0.15	Markdown weight for day 7 of the week
PRO_ST_END_REGULAR_ SEASON	0.15	Default percentage of sell-through for each individual product at end of regular periods.
PRO_ST_END_ CLEARANCE_SEASON	0.85	Default percentage of sell-through for each individual product at end of clearance season.
PRO_ST_TGT_TYPE	HARD	Default sell-through target type for each individual product.
PRO_ST_HARD_TGT_FLG	Y	Default value for sell-through hard target flag.
PRO_DFLT_RETURN_PCT	0.01	Default percentage of sales returned.

Table 12–24 (Cont.) Business Rules

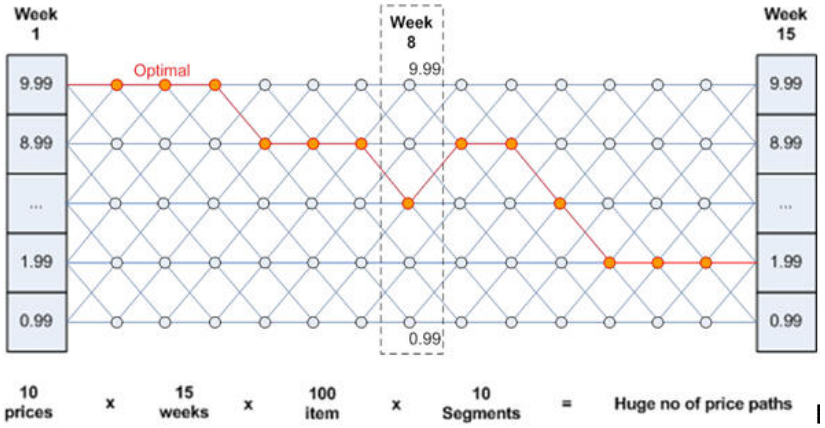
PARAM_NAME	PARAM_VALUE	DESCR
PRO_DFLT_RETURN_ LEAD_TIME	2	Default return lead time in weeks (time when the return happens after the purchase is made).
PRO_PR_MIN_COST	Y	Promotions recommendations cannot be lower than the cost of the product.
PRO_MR_MIN_COST	Y	Markdown recommendations cannot be lower than the cost of the product.
PRO_BASE_INITIAL_ BUDGET	9999999	Initial Budget for Base Scenarios.
PRO_SEAS_CURVE_ WEEKS_SWITCH	42	Seasonality Curve.
PRO_OFFERMAX	3	Maximum number of offers for segment in a given time period.
PRO_LOWRR	0.20	threshold value used to pick at least one low redemption rate offer for a segment and given time period.
PRO_HIGHRR	0.80	High Redemption Rate. This is a threshold value used to pick at least one high redemption offer for a given segment and given time period.
PRO_MAX_OFFERS_PER_ SEGMENT	3	This is the number of top offers that will be selected per customer_segment/class.
PRO_INV_QTY_BOH	Y	Specifies if the inventory on hand must be included into the total inventory.
PRO_INV_QTY_ON_ORD	Y	Specifies if the inventory on order must be included into the total inventory.
PRO_INV_QTY_IN_ TRANSIT	Y	Specifies if the inventory in transit must be included into the total inventory.
PRO_USE_ABSOLUTE_ CAL_FLG	Y	Y/N Indicator. Identifies if the absolute calendar must be used for Temporal Rules.
PRO_USE_LIFECYCLE_ FATIGUE	N	Y/N Indicator. Identifies if the life cycle fatigue must be enabled or not.

Optimization Algorithm Overview

The optimization algorithm analyzes trade-offs between a set of available price paths and picks the best price path. It then informs the user when to promote, how deep to promote, when to mark down, how deep to mark down, when to provide a targeted offer, and how deep the targeted offer must be. Note that the Targeted Offers are aligned to be in the same period when the promotion occurs.

The algorithm uses sophisticated mathematical modeling techniques to analyze all possible solutions to generate the best possible solution. The optimization algorithm is provided an objective (for example, maximize total revenue over all items in the season), business rules or restrictions, and the demand parameters for a particular item. The algorithm analyzes the trade-offs between all possible solutions (see Figure 12–6) and picks the solution that provides the best value for the objective. All the restrictions imposed by the user are treated as required; that is, all possible solutions must satisfy that particular criterion. The only constraint that can be enforced as soft is the sell-through target constraint. The user can make this constraint soft, which tells the optimization that this constraint does not have to be met. Optimization will try to satisfy this constraint, and, if it cannot, it will not return an infeasible or no solution.

Figure 12–6 Optimization Algorithm



To provide a sense of the complexity of the optimization, here is a sample of the trade-offs that are analyzed.

- Is it better to provide a promotion early in the life or wait until later in life to mark down? Will conducting in a promotion now help me given a shallower markdown at a later point in life?
- There are planned promotions scheduled at different points in time. Can these promotions be used and figure out whether an item needs further promotions or markdowns?
- Is providing a promotion at a particular week useful? Will it help meet the sell-through target? Will it increase revenue?
- There are customer segments with different response to price cuts. How can targeted offer provide a promotion that will entice the customer in customer segment A?
- Does providing a promotion or markdown or targeted offer for an item cause a conflict with the imposed business rules? For example, a user might determine that the item cannot be given more than 30% discount.

Model Apply

Forecasting is applied in the optimization using the demand parameters supplied. Suppose the user wants to generate weekly price recommendations. Depending on the day of the week when markdown is effective, forecasting is adjusted using daily weights to reflect that there are two prices in effect for that week. The same logic applies for promotions as well.

Objective Function

The objective function specified by the user plays a major role in determining which solutions are considered best. For example, if the user specifies the objective function as maximize profit margin, then the algorithm will look for solutions that are superior on profit margin and not necessarily on the other KPIs such as revenue and sales volume. Sometimes, understanding why an item got such a price recommendation might be as trivial as looking at the objective function contribution of that particular item to the objective function.

Constraints

If the objective function focuses on the best possible solution, then constraints work in the opposite direction, by restricting the set of possible solutions. For example, if the objective function says to select the most profitable discount for an item A, a constraint on item A may say it is not possible to have more than certain discount percentage.

Optimization enforces all constraints as required; that is, it finds all the solutions that satisfy all the constraints that are specified by the user. Sometimes, inadvertently, the user might specify conflicting constraints that can result in no solution or unexpected solutions. Often, a resolution can be found by just understanding the implications of individual constraints. More often than not, the user must analyze the interplay between two or more constraints to understand the solution. PRO_RUN_SANITY_CHECK_RSE_VW contains information on the errors/alerts/warnings generated.

OO supports a variety of constraints, and it is essential for the user to understand the purpose and role of each constraint in the optimization.

13

Control and Tactical Center

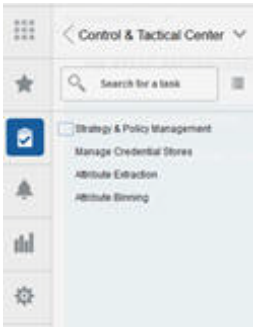
This chapter describes the Control and Tactical Center, where an administrative user can access and manage configurations for different applications.

Under the Control and Tactical Center, a user can access Strategy and Policy Management, which is the central place for managing the configurations of different applications and for setting up and configuring the demand forecasting runs. In addition, the modules for managing product attributes (Attribute Extraction and Attribute Binning) as well as the link for managing web service-related credentials can be also found under the Control and Tactical Center.

Depending on the role, a user will see one or multiple of the following links, as shown in Figure 13–1. Typically, the administrative user has access to all of the following.

- Strategy and Policy Management
- Manage Credential Stores
- Attribute Extraction
- Attribute Binning

Figure 13–1 Control and Tactical Center



Strategy and Policy Management

In Strategy and Policy Management, a user can edit the configurations of different applications using the Manage System Configurations screen. Only the tables that are editable can be accessed in this screen. Within each table, one or multiple columns are editable and a user can override the values in those columns.

Manage Forecast Configurations can be used to set up, manage, and configure the demand forecasting runs for different applications such as Offer Optimization (OO), Inventory Optimization (OO), Retail Demand Forecasting (RDF), Assortment Planning (AP) and Merchandise Financial Planning (MFP).

Forecast Configuration for MFP and AP

To configure the forecast process for MFP and AP, complete the following two steps:

1. Use the Manage System Configurations screen to review and modify the global configurations in RSE_CONFIG. For further details, see "Configuration Updates". The relevant configurations in RSE_CONFIG that must be reviewed and edited as required are listed in Table 13–1.

Table 13–1 Configurations

APPL CODE	PARAM NAME	PARAM VALUE	DESCR
RSE	EXTENDED_HIERARCHY_SRC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PROD_HIER	Default value is Y. If the product hierarchy data had nine levels, keep this value as Y. If it has seven levels, change this value to N.	Y/N Value. This parameter is used by the product hierarchy ETL to determine if the extended product hierarchy is required.
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has nine levels (i.e., it has an extended hierarchy), keep this value as 3. If it has seven levels, change this value to 1.	The hierarchy ID to use for the product (installation configuration).
RSE	PROD_HIER_SLSTXN_HIER_LEVEL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color, or 9 Style/color/Size). It *must* match the extended hierarchy leaf level.

2. Use the Manage Forecast Configurations screen to set up the forecast runs for MFP and AP, as follows.

   In Manage Forecast Configurations, start by setting up a run type in the Setup train stop. Click the + icon above the table and fill in the fields in the pop-up. For MFP and AP forecasting, the forecast method must be selected as Automatic Exponential Smoothing. You must create a run type for each forecast measure/forecast intersection combination that is required for MFP and/or AP.

   Once you are done with setting up the run types, click **Start Data Aggregation** in the Setup train stop. Select all the run types that were created and click **Submit**. When the aggregation is complete, the aggregation status will change to Complete. At this point, ad-hoc test runs and batch runs can be created and executed to generate a forecast.

   To create an ad-hoc run, go to the **Test** train stop. First, click a run type in the top table and then click the + icon in the bottom table to create a run. In the Create Run pop-up, you can change the configurations parameters related to estimation process and forecast process in their respective tabs. For example, if you want to test a run using the Bayesian method, edit the Estimation Method parameter in the Estimation tab using the **Edit** icon above the table. After modifying and reviewing the configuration parameters, click **Submit** to start the run. Once the run is complete, the status will change to Forecast Generation Complete.

   Doing test runs is an optional step. In addition to that, you must modify and review the configurations of the run type, activate the run type, enable auto-approve, and map the run type to the downstream application (in this case to MFP or AP). In the Manage train stop,

select a row, click **Edit Configurations Parameters**, and edit the estimation and forecast parameters as necessary. Once you are done, go to Review tab and click **Validate** and then close tab.

Note: If the run type is active, you will only be able to view the parameters. In order to edit the parameters, the run type must be inactive.

To activate the run type and enable the auto-approve, select a run type in the table and click the corresponding buttons above the table. Lastly, to map the run type to MFP or AP, go to Map train stop and click the + icon to create a new mapping.

Run Type Configurations for MFP and AP to Set Up GA Runs

To set up GA runs, create one run type for each forecast intersection/measure combination. Table 13–2 shows the configurations for each run type.

Table 13–2 Run Type Configurations

	Forecast Level (Merchandise/Location/ Calendar)	Forecast Measure	Forecast Method	Data Source
Location Plan (MFP)	Department/Location/Week	Total Gross Sales Units	Automatic Exponential Smoothing for all run types	Store Sales for all run types
		Total Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
Location Target (MFP)	Company/Location/Week	Total Gross Sales Units		
		Total Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
Merchandise Plan (MFP)	Subclass/Area/Week	Clearance Gross Sales Units		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Units		
		Regular and Promotion Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
Merchandise Target (MFP)	Department/Area/Week	Clearance Gross Sales Units		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Units		
		Regular and Promotion Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		

Table 13–2 (Cont.) Run Type Configurations

	Forecast Level (Merchandise/Location/ Calendar)	Forecast Measure	Forecast Method	Data Source
Forecast Level 1 (AP)	Subclass/Location/Week	Clearance Gross Sales Unit		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Amount		
		Regular and Promotion Gross Sales Amount		
Forecast Level 2 (AP)	SKU/Location/Week	Clearance Gross Sales Units		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Units		
		Regular and Promotion Sales Units		

Note: For all batch runs, in addition to the pre-season forecast, the Bayesian forecast is also generated by default. In the Manage train stop, you must only set the method for the pre-season forecast (either as Automatic Exponential Smoothing or Seasonal Exponential Smoothing).

Batch and Ad-Hoc jobs for MFP and AP Forecasting

The Batch and Ad-Hoc jobs listed in Table 13–3 are used for loading foundation data (product hierarchy, location hierarchy, calendar, product attributes, and so on).

Table 13–3 Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
ORASE_START_BATCH_JOB	ORASE_START_BATCH_JOB	rse_process_state_update.ksh
ORASE_START_BATCH_SET_ ACTIVE_JOB	ORASE_START_BATCH_SET_ ACTIVE_JOB	rse_batch_type_active.ksh
ORASE_START_BATCH_REFRESH_ RESTR_JOB	ORASE_START_BATCH_REFRESH_ RESTR_JOB	rse_batch_freq_restriction.ksh
ORASE_START_BATCH_END_JOB	ORASE_START_BATCH_END_JOB	rse_process_state_update.ksh
RSE_WEEKLY_INPUT_FILES_ START_JOB	RSE_WEEKLY_INPUT_FILES_ START_JOB	rse_process_state_update.ksh
WEEKLY_INPUT_FILES_WAIT_JOB	WEEKLY_INPUT_FILES_WAIT_JOB	rse_batch_zip_file_wait.ksh
WEEKLY_INPUT_FILES_VAL_JOB	WEEKLY_INPUT_FILES_VAL_JOB	rse_batch_zip_file_extract.ksh
WEEKLY_INPUT_FILES_COPY_ JOB	WEEKLY_INPUT_FILES_COPY_ JOB	rse_batch_inbound_file_copy.ksh

Table 13–3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_WEEKLY_INPUT_FILES_ END_JOB	RSE_WEEKLY_INPUT_FILES_ END_JOB	rse_process_state_update.ksh
RSE_PROD_HIER_ETL_START_JOB	RSE_PROD_HIER_ETL_START_JOB	rse_process_state_update.ksh
RSE_PROD_SRC_XREF_LOAD_JOB	RSE_PROD_SRC_XREF_LOAD_JOB	rse_prod_src_xref_load.ksh
RSE_PROD_HIER_LOAD_JOB	RSE_PROD_HIER_LOAD_JOB	rse_prod_hier_load.ksh
RSE_PROD_TC_LOAD_JOB	RSE_PROD_TC_LOAD_JOB	rse_prod_tc_load.ksh
RSE_PROD_DH_LOAD_JOB	RSE_PROD_DH_LOAD_JOB	rse_prod_dh_load.ksh
RSE_PROD_GROUP_LOAD_JOB	RSE_PROD_GROUP_LOAD_JOB	rse_load_prod_group.ksh
RSE_PROD_HIER_ETL_END_JOB	RSE_PROD_HIER_ETL_END_JOB	rse_process_state_update.ksh
RSE_LOC_HIER_ETL_START_JOB	RSE_LOC_HIER_ETL_START_JOB	rse_process_state_update.ksh
RSE_LOC_SRC_XREF_LOAD_JOB	RSE_LOC_SRC_XREF_LOAD_JOB	rse_loc_src_xref_load.ksh
RSE_LOC_HIER_LOAD_JOB	RSE_LOC_HIER_LOAD_JOB	rse_loc_hier_load.ksh
RSE_LOC_HIER_TC_LOAD_JOB	RSE_LOC_HIER_TC_LOAD_JOB	rse_loc_hier_tc_load.ksh
RSE_LOC_HIER_DH_LOAD_JOB	RSE_LOC_HIER_DH_LOAD_JOB	rse_loc_hier_dh_load.ksh
RSE_LOC_HIER_ETL_END_JOB	RSE_LOC_HIER_ETL_END_JOB	rse_process_state_update.ksh
RSE_LOC_ATTR_LOAD_START_ JOB	RSE_LOC_ATTR_LOAD_START_ JOB	rse_process_state_update.ksh
RSE_CDA_ETL_LOAD_LOC_JOB	RSE_CDA_ETL_LOAD_LOC_JOB	rse_cda_etl_load_location.ksh
RSE_LOC_ATTR_LOAD_END_JOB	RSE_LOC_ATTR_LOAD_END_JOB	rse_process_state_update.ksh
RSE_LIKE_LOC_LOAD_START_ JOB	RSE_LIKE_LOC_LOAD_START_ JOB	rse_process_state_update.ksh
RSE_LIKE_LOC_STG_JOB	RSE_LIKE_LOC_STG_JOB	rse_like_loc_stg.ksh
RSE_LIKE_LOC_COPY_JOB	RSE_LIKE_LOC_COPY_JOB	
RSE_LIKE_LOC_STG_CNE_JOB	RSE_LIKE_LOC_STG_CNE_JOB	
RSE_LIKE_LOC_LOAD_JOB	RSE_LIKE_LOC_LOAD_JOB	rse_like_loc_load.ksh
RSE_LIKE_LOC_LOAD_END_JOB	RSE_LIKE_LOC_LOAD_END_JOB	rse_process_state_update.ksh
RSE_LIKE_PROD_LOAD_START_ JOB	RSE_LIKE_PROD_LOAD_START_ JOB	rse_process_state_update.ksh
RSE_LIKE_PROD_STG_JOB	RSE_LIKE_PROD_STG_JOB	rse_like_prod_stg.ksh
RSE_LIKE_PROD_COPY_JOB	RSE_LIKE_PROD_COPY_JOB	
RSE_LIKE_PROD_STG_CNE_JOB	RSE_LIKE_PROD_STG_CNE_JOB	
RSE_LIKE_PROD_LOAD_JOB	RSE_LIKE_PROD_LOAD_JOB	rse_like_prod_load.ksh
RSE_LIKE_PROD_LOAD_END_JOB	RSE_LIKE_PROD_LOAD_END_JOB	rse_process_state_update.ksh
RSE_DATA_STAGING_START_JOB	RSE_DATA_STAGING_START_JOB	rse_process_state_update.ksh
RSE_MD_CDA_VALUES_STG_JOB	RSE_MD_CDA_VALUES_STG_JOB	rse_md_cda_values_stg.ksh
RSE_MD_CDA_VALUES_COPY_ JOB	RSE_MD_CDA_VALUES_COPY_ JOB	
RSE_MD_CDA_VALUES_STG_ CNE_JOB	RSE_MD_CDA_VALUES_STG_ CNE_JOB	

Table 13–3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_PR_LC_CDA_STG_JOB	RSE_PR_LC_CDA_STG_JOB	rse_pr_lc_cda_stg.ksh
RSE_PR_LC_CDA_COPY_JOB	RSE_PR_LC_CDA_COPY_JOB	
RSE_PR_LC_CDA_STG_CNE_JOB	RSE_PR_LC_CDA_STG_CNE_JOB	
RSE_PR_LC_CAL_CDA_JOB	RSE_PR_LC_CAL_CDA_JOB	rse_pr_lc_cal_cda_stg.ksh
RSE_PR_LC_CAL_CDA_COPY_JOB	RSE_PR_LC_CAL_CDA_COPY_JOB	
RSE_PR_LC_CAL_CDA_STG_CNE_JOB	RSE_PR_LC_CAL_CDA_STG_CNE_JOB	
RSE_MD_CDA_STG_JOB	RSE_MD_CDA_STG_JOB	rse_md_cda_stg.ksh
RSE_MD_CDA_COPY_JOB	RSE_MD_CDA_COPY_JOB	
RSE_MD_CDA_STG_CNE_JOB	RSE_MD_CDA_STG_CNE_JOB	
RSE_MD_CDA_LOAD_JOB	RSE_MD_CDA_LOAD_JOB	rse_md_cda_load.ksh
RSE_MD_CDA_VALUES_LOAD_JOB	RSE_MD_CDA_VALUES_LOAD_JOB	rse_md_cda_values_load.ksh
RSE_DATA_STAGING_END_JOB	RSE_DATA_STAGING_END_JOB	rse_process_state_update.ksh
RSE_PROD_ATTR_LOAD_START_JOB	RSE_PROD_ATTR_LOAD_START_JOB	rse_process_state_update.ksh
RSE_CDA_ETL_LOAD_PROD_JOB	RSE_CDA_ETL_LOAD_PROD_JOB	rse_cda_etl_load_product.ksh
RSE_PROD_ATTR_GRP_VALUE_STG_JOB	RSE_PROD_ATTR_GRP_VALUE_STG_JOB	rse_prod_attr_grp_value_stg.ksh
RSE_PROD_ATTR_GRP_VALUE_COPY_JOB	RSE_PROD_ATTR_GRP_VALUE_COPY_JOB	
RSE_PROD_ATTR_GRP_VALUE_STG_CNE_JOB	RSE_PROD_ATTR_GRP_VALUE_STG_CNE_JOB	
RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB	RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB	rse_prod_attr_grp_value_load.ksh
RSE_PROD_ATTR_VALUE_XREF_STG_JOB	RSE_PROD_ATTR_VALUE_XREF_STG_JOB	rse_prod_attr_value_xref_stg.ksh
RSE_PROD_ATTR_VALUE_XREF_COPY_JOB	RSE_PROD_ATTR_VALUE_XREF_COPY_JOB	
RSE_PROD_ATTR_VALUE_XREF_STG_CNE_JOB	RSE_PROD_ATTR_VALUE_XREF_STG_CNE_JOB	
RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB	RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB	rse_prod_attr_value_xref_load.ksh
RSE_PROD_ATTR_LOAD_END_JOB	RSE_PROD_ATTR_LOAD_END_JOB	rse_process_state_update.ksh
RSE_CM_GRP_HIER_LOAD_START_JOB	RSE_CM_GRP_HIER_LOAD_START_JOB	rse_process_state_update.ksh
RSE_CM_GRP_XREF_LOAD_JOB	RSE_CM_GRP_XREF_LOAD_JOB	rse_cm_grp_xref_load.ksh
RSE_CM_GRP_HIER_LOAD_JOB	RSE_CM_GRP_HIER_LOAD_JOB	rse_cm_grp_hier_load.ksh
RSE_CM_GRP_TC_LOAD_JOB	RSE_CM_GRP_TC_LOAD_JOB	rse_cm_grp_tc_load.ksh
RSE_CM_GRP_DH_LOAD_JOB	RSE_CM_GRP_DH_LOAD_JOB	rse_cm_grp_dh_load.ksh
RSE_CM_GRP_HIER_LOAD_END_JOB	RSE_CM_GRP_HIER_LOAD_END_JOB	rse_process_state_update.ksh

Table 13–3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_TRADE_AREA_HIER_LOAD_START_JOB	RSE_TRADE_AREA_HIER_LOAD_START_JOB	rse_process_state_update.ksh
RSE_TRADE_AREA_SRC_XREF_LOAD_JOB	RSE_TRADE_AREA_SRC_XREF_LOAD_JOB	rse_trade_area_src_xref_load.ksh
RSE_TRADE_AREA_HIER_LOAD_JOB	RSE_TRADE_AREA_HIER_LOAD_JOB	rse_trade_area_hier_load.ksh
RSE_TRADE_AREA_TC_LOAD_JOB	RSE_TRADE_AREA_TC_LOAD_JOB	rse_trade_area_tc_load.ksh
RSE_TRADE_AREA_DH_LOAD_JOB	RSE_TRADE_AREA_DH_LOAD_JOB	rse_trade_area_dh_load.ksh
RSE_TRADE_AREA_HIER_LOAD_END_JOB	RSE_TRADE_AREA_HIER_LOAD_END_JOB	rse_process_state_update.ksh
RSE_CUST_CONS_SEG_HIER_ETL_START_JOB	RSE_CUST_CONS_SEG_HIER_ETL_START_JOB	rse_process_state_update.ksh
RSE_CUSTSEG_SRC_XREF_LOAD_JOB	RSE_CUSTSEG_SRC_XREF_LOAD_JOB	rse_custseg_src_xref_load.ksh
RSE_CUSTSEG_HIER_LOAD_JOB	RSE_CUSTSEG_HIER_LOAD_JOB	rse_custseg_hier_load.ksh
RSE_CUSTSEG_HIER_TC_LOAD_JOB	RSE_CUSTSEG_HIER_TC_LOAD_JOB	rse_custseg_hier_tc_load.ksh
RSE_CUSTSEG_CUST_XREF_LOAD_JOB	RSE_CUSTSEG_CUST_XREF_LOAD_JOB	rse_custseg_cust_xref_load.ksh
RSE_CONSEG_LOAD_JOB	RSE_CONSEG_LOAD_JOB	rse_conseg_load.ksh
RSE_CONSEG_ALLOC_LOAD_JOB	RSE_CONSEG_ALLOC_LOAD_JOB	rse_conseg_alloc_load.ksh
RSE_CUSTSEG_ALLOC_LOAD_JOB	RSE_CUSTSEG_ALLOC_LOAD_JOB	rse_custseg_alloc_load.ksh
RSE_CUST_CONS_SEG_HIER_ETL_END_JOB	RSE_CUST_CONS_SEG_HIER_ETL_END_JOB	rse_process_state_update.ksh
RSE_CAL_HIER_ETL_START_JOB	RSE_CAL_HIER_ETL_START_JOB	rse_process_state_update.ksh
RSE_REGULAR_MAIN_LOAD_JOB	RSE_REGULAR_MAIN_LOAD_JOB	rse_regular_main_load.ksh
RSE_FISCAL_MAIN_LOAD_JOB	RSE_FISCAL_MAIN_LOAD_JOB	rse_fiscal_main_load.ksh
RSE_CAL_HIER_ETL_END_JOB	RSE_CAL_HIER_ETL_END_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_END_START_JOB	RSE_DIMENSION_LOAD_END_START_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_END_END_JOB	RSE_DIMENSION_LOAD_END_END_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_START_START_JOB	RSE_DIMENSION_LOAD_START_START_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_START_END_JOB	RSE_DIMENSION_LOAD_START_END_JOB	rse_process_state_update.ksh
RSE_SLS_START_START_JOB	RSE_SLS_START_START_JOB	rse_process_state_update.ksh
RSE_SLS_START_END_JOB	RSE_SLS_START_END_JOB	rse_process_state_update.ksh
RSE_SLS_END_START_JOB	RSE_SLS_END_START_JOB	rse_process_state_update.ksh
RSE_SLS_END_END_JOB	RSE_SLS_END_END_JOB	rse_process_state_update.ksh
ORASE_END_START_JOB	ORASE_END_START_JOB	rse_process_state_update.ksh

Table 13–3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
ORASE_END_RUN_DATE_UPDT_JOB	ORASE_END_RUN_DATE_UPDT_JOB	rse_batch_run_date_update.ksh
ORASE_END_END_JOB	ORASE_END_END_JOB	rse_process_state_update.ksh
RSE_MASTER_ADHOC_JOB	Run RSE master script	rse_master.ksh
PMO_MASTER_ADHOC_JOB	Run PMO master script	pmo_master.ksh

The Batch and Ad-hoc jobs listed in Table 13–4 are used to prepare weekly data and to run weekly batches for MFP and AP.

Table 13–4 Batch and Ad Hoc Jobs for MFP and AP Forecasting

JobName	Description	RmsBatch
PMO_ACTIVITY_LOAD_START_JOB	PMO Activity load start job	rse_process_state_update.ksh
PMO_ACTIVITY_STG_JOB	PMO Activity data staging	pmo_activity_load_setup.ksh
PMO_ACTIVITY_LOAD_JOB	PMO Activity load job	pmo_activity_load_process.ksh
PMO_ACTIVITY_LOAD_END_JOB	PMO Activity load end job	rse_process_state_update.ksh
RSE_FCST_SALES_PLAN_START_JOB	Start job for sales plan data load for Bayesian method	rse_process_state_update.ksh
RSE_FCST_SALES_PLAN_LOAD_JOB	Load job for sales plan data for Bayesian method	rse_fcst_sales_plan_load.ksh
RSE_FCST_SALES_PLAN_END_JOB	End job for sales plan data load for Bayesian method	rse_process_state_update.ksh
RSE_ASSORT_PLAN_LOAD_START_JOB	Loading assortment plan through interface for AP	rse_process_state_update.ksh
RSE_ASSORT_PLAN_LOAD_STG_JOB	Loading assortment plan through interface for AP	rse_assort_plan_per_stg.ksh
RSE_ASSORT_PLAN_LOAD_COPY_JOB	Loading assortment plan through interface for AP	
RSE_ASSORT_PLAN_LOAD_STG_CNE_JOB	Loading assortment plan through interface for AP	
RSE_RDX_ASSORT_PLAN_IMPORT_JOB	Importing assortment plan from RDX to AIF for AP	rse_rdx_assort_plan_import.ksh
RSE_ASSORT_PLAN_LOAD_JOB	Loading assortment plan through interface for AP	rse_assort_plan_per_load.ksh
RSE_ASSORT_PLAN_LOAD_END_JOB	Loading assortment plan through interface for AP	rse_process_state_update.ksh
RSE_PLANNING_PERIOD_LOAD_START_JOB	Loading assortment period through interface for AP	rse_process_state_update.ksh
RSE_PLANNING_PERIOD_STG_JOB	Loading assortment period through interface for AP	rse_planning_period_stg.ksh
RSE_PLANNING_PERIOD_COPY_JOB	Loading assortment period through interface for AP	
RSE_PLANNING_PERIOD_STG_CNE_JOB	Loading assortment period through interface for AP	

Table 13–4 (Cont.) Batch and Ad Hoc Jobs for MFP and AP Forecasting

JobName	Description	RmsBatch
RSE_RDX_ASSORT_PERIOD_IMPORT_JOB	Importing assortment period from RDX to AIF for AP	rse_rdx_assort_period_import.ksh
RSE_PLANNING_PERIOD_LOAD_JOB	Loading assortment period through interface for AP	rse_planning_period_load.ksh
RSE_PLANNING_PERIOD_LOAD_END_JOB	Loading assortment period through interface for AP	rse_process_state_update.ksh
RSE_MFP_FCST_BATCH_RUN_START_JOB	Start job for executing MFP and AP forecast batch runs	rse_process_state_update.ksh
RSE_CREATE_MFP_BATCH_RUN_PROC_JOB	Create MFP and AP batch runs	rse_create_mfp_batch_run_proc.ksh
RSE_MFP_FCST_BATCH_PROCESS_JOB	Execute MFP and AP forecast batch	rse_fcst_mfp_batch_process.ksh
RSE_MFP_FCST_BATCH_RUN_END_JOB	End job for executing MFP and AP forecast batch runs	rse_process_state_update.ksh
RSE_CREATE_MFP_BATCH_RUN_PROC_ADHOC_JOB	Ad hoc job to create MFP and AP batch runs	rse_create_mfp_batch_run_proc.ksh
RSE_MFP_FCST_BATCH_PROCESS_ADHOC_JOB	Ad hoc job to execute MFP and AP forecast batch	rse_fcst_mfp_batch_process.ksh

The forecast values generated by runs that are associated with active run types are exported to the RDX schema. The jobs for exporting the outputs to the RDX schema are listed in Table 13–5.

Table 13–5 Export Jobs for MFP and AP from AIF to RDX

JobName	Description	RmsBatch
RSE_FCST_EXPORT_START_JOB	Start job for MFP and AP forecast export	rse_process_state_update.ksh
RSE_MFP_FCST_EXPORT_JOB	Export MFP forecast	rse_mfp_export.ksh
RSE_AP_FCST_EXPORT_JOB	Export AP forecast	rse_ap_export.ksh
RSE_FCST_EXPORT_END_JOB	End job for MFP and AP forecast export	rse_process_state_update.ksh
RSE_MFP_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export MFP forecast	rse_mfp_export.ksh
RSE_AP_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export AP forecast	rse_ap_export.ksh
RSE_RDX_ASSORT_ELASTICITY_EXPORT_JOB	Export assortment elasticity for AP	rse_rdx_assort_elasticity_export.ksh
RSE_RDX_ASSORT_ELASTICITY_EXPORT_ADHOC_JOB	Ad hoc job to export assortment elasticity for AP	rse_rdx_assort_elasticity_export.ksh
RSE_RDX_LOC_CLUSTER_EXPORT_JOB	Export store clusters for AP	rse_rdx_loc_cluster_export.ksh

Table 13–5 (Cont.) Export Jobs for MFP and AP from AIF to RDX

JobName	Description	RmsBatch
RSE_RDX_LOC_CLUSTER_EXPORT_ADHOC_JOB	Ad hoc job to export store clusters for AP	rse_rdx_loc_cluster_export.ksh
RSE_RDX_SIZE_PROFILE_EXPORT_JOB	Export size profiles for AP	rse_rdx_size_profile_export.ksh
RSE_RDX_SIZE_PROFILE_EXPORT_ADHOC_JOB	Ad hoc job to export size profiles for AP	rse_rdx_size_profile_export.ksh

Forecast Configuration for RDF and AIF (including Inventory Optimization and Offer Optimization)

To configure forecast process for RDF, do the following.

1. Use the Manage System Configurations screen to review and modify the global configurations in RSE_CONFIG. For further details, see "Configuration Updates". The relevant configurations in RSE_CONFIG that must be reviewed and edited as required are listed in Table 13–6.

Table 13–6 RSE Configurations

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	EXTENDED_HIERARCHY_SRC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PROD_HIER	Default value is Y. If the product hierarchy data had 9 levels, keep this value as Y. If it has 7 levels, change this value to N.	Y/N Value. This parameter is used by the product hierarchy ETL to know if the extended product hierarchy is needed.
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has 9 levels (i.e. it has extended hierarchy), keep this value as 3. If it has 7 levels, change this value to 1.	The hierarchy id to use for the product (Installation configuration).
RSE	PROD_HIER_SLSTXN_HIER_LEVEL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color or 9 Style/color/Size). It MUST match the extended hierarchy leaf level
PMO	PMO_AGGR_INVENTORY_DATA_FLG	Default value is Y.	Specifies if inventory data is present and if it should be used when aggregating activities data. Set this value to N if inventory data is not loaded (inventory data is not required for MFP and AP forecasting but it is required for other applications like Offer Optimization, Inventory Optimization and Retail Demand Forecasting).

Table 13–6 (Cont.) RSE Configurations

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	SLS_TXN_EXCLUDE_ LIABILITY_SLS	Default value is N.	Y/N flag indicating if liability sales columns should be excluded when retrieving sales transaction data.
RSE	RSE_LOC_HIER_EXCL_ FRANCHISE	Default value is N.	Y/N flag indicating if franchise locations should be excluded from the location hierarchy.
RSE	PROMOTIONAL_SALES_ AVAIL	Default value is Y.	Y/N flag indicating if promotion data is available for Causal-Long Life Cycle run types. Set this value to N if promotion data is not yet loaded into AIF.
RSE	DFLT_LIFE_CYCLE	Default value is SLC.	This parameter identifies the main life cycle of the items. Keep this value as SLC if most of the items have short life cycle and change this value to LLC is most of the items have long life cycle.

2. Use the Manage Forecast Configurations screen to set up the forecast runs, as follows.

   In Manage Forecast Configurations, start by setting up a run type in the Setup train stop. Click the + icon above the table and fill in the fields in the pop-up. The customer should decide the majority life cycle of the items and accordingly set the value of the parameter DFLT_LIFE_CYCLE in the RSE_CONFIG table. All the items that are of the opposite life cycle should be stored in the table RSE_FCST_LIFE_CYCLE_CLSF (populated through an interface). If the customer has both Short Life Cycle and Long Life Cycle items, you must create one run type for each. The run types will either use the items present in the table RSE_FCST_LIFE_CYCLE_CLSF or use the rest of the items depending on the values of the Forecast Method for the run type and the DFLT_LIFE_CYCLE parameter. You must create a run type for each forecast intersection combination that is required.

   Once you have finished setting up the run types, click **Start Data Aggregation** in the Setup train stop. Select all the run types that were created and click **Submit**. When the aggregation is complete, the aggregation status will change to **Complete**. At this point, ad-hoc test runs and batch runs can be created and executed to generate a forecast.

   To create an ad-hoc run, go to the **Test** train stop. First, click a run type in the top table and then click the + icon in the bottom table to create a run. In the Create Run pop-up, you can change the configurations parameters related to estimation process, base demand, and forecast process in their respective tabs. After modifying and reviewing the configuration parameters, click **Submit** to start the run. Upon submit, a validation process runs to validate the value of configuration parameters. If there is any error, correct the corresponding parameter and submit again. Once the run is complete, the status will change to Forecast Generation Complete.

   Once you are done with testing, you must modify and review the configurations of the run type, activate the run type, enable auto-approve, and map the run type to the downstream application (for example, RDF). In the Manage train stop, select a row, click **Edit Configurations Parameters**, and edit the estimation and forecast parameters as necessary. Once you are done, go to Review tab and click **Validate** and then close tab.

Note: If the run type is active, you will only be able to view the parameters. In order to edit the parameters, the run type must be inactive.

To activate the run type and enable the auto-approve, select a run type in the table and click the corresponding buttons above the table. Lastly, to map the run type to RDF, go to Map train stop and click the + icon to create a new mapping.

Note: For RDF, you will be able to map one or multiple run types to the same "external key" run type. The run types that are being mapped to the same external key must have same forecast intersection, same price zone and customer segment flag, and same data source, but opposite life cycle.

Batch and Ad-Hoc Jobs for RDF Forecasting

The configuration and main data load jobs listed in Table 13–3 are used for loading foundation data (product hierarchy, location hierarchy, calendar, product attributes, and so on).

The Batch and Ad-Hoc jobs listed in Table 13–7 are used for preparing weekly data and running weekly batches for RDF.

Table 13–7 Batch and Ad Hoc Jobs for RDF Forecasting

JobName	Description	RmsBatch
PMO_ACTIVITY_LOAD_START_JOB	PMO Activity load start job	rse_process_state_update.ksh
PMO_ACTIVITY_STG_JOB	PMO Activity data staging	pmo_activity_load_setup.ksh
PMO_ACTIVITY_LOAD_JOB	PMO Activity load job	pmo_activity_load_process.ksh
PMO_ACTIVITY_LOAD_END_JOB	PMO Activity load end job	rse_process_state_update.ksh
RSE_PROMO_HIER_ETL_START_JOB	Start job for loading promotion hierarchy data	rse_process_state_update.ksh
RSE_PROMO_SRC_XREF_LOAD_JOB	Job for loading promotion hierarchy data	rse_promo_src_xref_load.ksh
RSE_PROMO_HIER_LOAD_JOB	Job for loading promotion hierarchy data	rse_promo_hier_load.ksh
RSE_PROMO_HIER_DH_LOAD_JOB	Job for loading promotion hierarchy data	rse_promo_hier_dh_load.ksh
RSE_PROMO_HIER_ETL_END_JOB	End job for loading promotion hierarchy data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC_START_JOB	Start job for loading promotion lift data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC_LOAD_JOB	Job for loading promotion lift data	pmo_activity_load_pl_llc.ksh
PMO_ACTIVITY_LOAD_PL_LLC_END_JOB	End job for loading promotion lift data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC_LOAD_ADHOC_JOB	Ad-hoc job for loading promotion lift data	pmo_activity_load_pl_llc.ksh
PMO_ACTIVITY_LOAD_OFFERS_INITIAL_ADHOC_JOB	Initial one-time job for loading promotion offers data	pmo_activity_load_offers_initial.ksh

Table 13–7 (Cont.) Batch and Ad Hoc Jobs for RDF Forecasting

JobName	Description	RmsBatch
PMO_ACTIVITY_LOAD_OFFERS_START_JOB	Start job for loading promotion offers data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_OFFERS_LOAD_JOB	Job for loading promotion offers data	pmo_activity_load_offers_weekly.ksh
PMO_ACTIVITY_LOAD_OFFERS_END_JOB	End job for loading promotion offers data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_OFFERS_LOAD_ADHOC_JOB	Ad-hoc job for loading promotion offers data	pmo_activity_load_offers_weekly.ksh
RSE_FLEX_GROUP_LOAD_START_JOB	Loading flex group data through interface	rse_process_state_update.ksh
RSE_FLEX_GROUP_DTL_STG_JOB	Loading flex group data through interface	rse_flex_group_dtl_stg.ksh
RSE_FLEX_GROUP_DTL_COPY_JOB	Loading flex group data through interface	
RSE_FLEX_GROUP_DTL_STG_CNE_JOB	Loading flex group data through interface	
RSE_FLEX_GROUP_DTL_LOAD_JOB	Loading flex group data through interface	rse_flex_group_dtl_load.ksh
RSE_FLEX_GROUP_LOAD_END_JOB	Loading flex group data through interface	rse_process_state_update.ksh
RSE_FCST_SPREAD_PROFILE_LOAD_START_JOB	Loading spread profile data through interface	rse_process_state_update.ksh
RSE_FCST_SPREAD_PROFILE_STG_JOB	Loading spread profile data through interface	rse_fcst_spread_profile_stg.ksh
RSE_FCST_SPREAD_PROFILE_COPY_JOB	Loading spread profile data through interface	
RSE_FCST_SPREAD_PROFILE_STG_CNE_JOB	Loading spread profile data through interface	
RSE_FCST_SPREAD_PROFILE_LOAD_JOB	Loading spread profile data through interface	rse_fcst_spread_profile_load.ksh
RSE_FCST_SPREAD_PROFILE_LOAD_END_JOB	Loading spread profile data through interface	rse_process_state_update.ksh
RSE_FCST_LIFE_CYCLE_CLSF_LOAD_START_JOB	Loading exception life cycle items through interface	rse_process_state_update.ksh
RSE_FCST_LIFE_CYCLE_CLSF_STG_JOB	Loading exception life cycle items through interface	rse_fcst_life_cycle_clsf_stg.ksh
RSE_FCST_LIFE_CYCLE_CLSF_COPY_JOB	Loading exception life cycle items through interface	
RSE_FCST_LIFE_CYCLE_CLSF_STG_CNE_JOB	Loading exception life cycle items through interface	
RSE_FCST_LIFE_CYCLE_CLSF_LOAD_JOB	Loading exception life cycle items through interface	rse_fcst_life_cycle_clsf_load.ksh
RSE_FCST_LIFE_CYCLE_CLSF_LOAD_END_JOB	Loading exception life cycle items through interface	rse_process_state_update.ksh
PMO_RUN_EXEC_START_JOB	Start job for PMO run execution	rse_process_state_update.ksh
PMO_CREATE_BATCH_RUN_JOB	Create batch run for PMO execution	rse_create_pmo_batch_run_proc.ksh

Table 13–7 (Cont.) Batch and Ad Hoc Jobs for RDF Forecasting

JobName	Description	RmsBatch
PMO_RUN_EXEC_SETUP_JOB	Setup job for PMO run execution	pmo_run_exec_setup.ksh
PMO_RUN_EXEC_PROCESS_JOB	Process job for PMO run execution	pmo_run_exec_process.ksh
PMO_RUN_EXEC_END_JOB	End job for PMO run execution	rse_process_state_update.ksh
PMO_RUN_EXEC_ADHOC_JOB	Ad-hoc job for PMO run execution	pmo_run_execution.ksh
RSE_FCST_BATCH_RUN_START_JOB	Start job for forecast batch run	rse_process_state_update.ksh
RSE_CREATE_FCST_BATCH_RUN_JOB	Create forecast batch run	rse_create_fcst_batch_run_proc.ksh
RSE_FCST_BATCH_PROCESS_JOB	Execute Base Demand and Demand Forecast for forecast run	rse_fcst_batch_process.ksh
RSE_FCST_BATCH_RUN_END_JOB	End job for creating forecast batch run	rse_process_state_update.ksh
RSE_CREATE_FCST_BATCH_RUN_ADHOC_JOB	Adhoc job to create forecast batch run	rse_create_fcst_batch_run_proc.ksh
RSE_FCST_BATCH_PROCESS_ADHOC_JOB	Adhoc job to execute Base Demand and Demand Forecast for forecast run	rse_fcst_batch_process.ksh

The forecast values generated by runs that are associated with active run types and mapped to RDF are exported to RDX schema. Table 13–8 lists the jobs to export the main forecast data.

Table 13–8 Jobs for Exporting Forecast Outputs from AIF to RDX

JobName	Description	RMSBatch
RSE_RDF_FCST_EXPORT_JOB	Export RDF forecast	rse_rdf_export.ksh
RSE_RDF_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export RDF forecast	rse_rdf_export.ksh

Table 13–9 list the additional jobs for exporting different data such as promotions, and configuration parameters for run types.

Table 13–9 Additional Jobs for Exporting Data from AIF to RDX

JobName	Description	RmsBatch
RSE_HIST_PROMO_OFFER_SALES_EXPORT_ADHOC_JOB	Initial one-time job for exporting promotion offer sales figures	rse_rdf_offer_sales_hist_exp.ksh
RSE_PROMO_OFFER_EXPORT_START_JOB	Start job for exporting promotion offer	rse_process_state_update.ksh
RSE_PROMO_OFFER_EXPORT_JOB	Export promotion offers	rse_rdf_offers_hier_exp.ksh
RSE_PROMO_OFFER_SALES_EXPORT_JOB	Export promotion offer sales figures	rse_rdf_offer_sales_exp.ksh
RSE_PROMO_OFFER_EXPORT_END_JOB	End job for exporting promotion offer	rse_process_state_update.ksh
RSE_PROMO_OFFER_EXPORT_ADHOC_JOB	Ad hoc job to export promotion offers	rse_rdf_offers_hier_exp.ksh
RSE_PROMO_OFFER_SALES_EXPORT_ADHOC_JOB	Ad hoc job to export promotion offer sales figures	rse_rdf_offer_sales_exp.ksh

Table 13–9 (Cont.) Additional Jobs for Exporting Data from AIF to RDX

JobName	Description	RmsBatch
RSE_FCST_RUN_TYPE_CONF_EXPORT_START_JOB	Start job for exporting runtype config	rse_process_state_update.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_SETUP_JOB	Setup job for exporting runtype config	rse_fcst_run_type_conf_exp_setup.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_PROCESS_JOB	Process job for exporting runtype con	rse_fcst_run_type_conf_exp_process.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_END_JOB	End job for exporting runtype config	rse_process_state_update.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_SETUP_ADHOC_JOB	Ad-hoc setup job for exporting runtype config	rse_fcst_run_type_conf_exp_setup.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_PROCESS_ADHOC_JOB	Ad-hoc process job for exporting runtype config	rse_fcst_run_type_conf_exp_process.ksh

The jobs for importing forecast parameters from RDX schema are listed in Table 13–10:

Table 13–10 Jobs for Importing Forecast Parameters from RDX to AIF

JobName	Description	RMSBatch
RSE_RDX_FCST_PARAM_START_JOB	Start job for importing forecast parameters	rse_process_state_update.ksh
RSE_RDX_FCST_PARAM_SETUP_JOB	Setup job for importing forecast parameters	rse_rdx_fcst_param_setup.ksh
RSE_RDX_FCST_PARAM_PROCESS_JOB	Process job for importing forecast parameters	rse_rdx_fcst_param_process.ksh
RSE_RDX_FCST_PARAM_END_JOB	End job for importing forecast parameters	rse_process_state_update.ksh
RSE_RDX_FCST_PARAM_SETUP_ADHOC_JOB	Ad-hoc setup job for importing forecast parameters	rse_rdx_fcst_param_setup.ksh
RSE_RDX_FCST_PARAM_PROCESS_ADHOC_JOB	Ad-hoc process job for importing forecast parameters	rse_rdx_fcst_param_process.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_START_JOB	Start job for importing enablement flags for new item forecast	rse_process_state_update.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_SETUP_JOB	Setup job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_setup.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_PROCESS_JOB	Process job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_process.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_END_JOB	End job for importing enablement flags for new item forecast	rse_process_state_update.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_SETUP_ADHOC_JOB	Ad-hoc setup job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_setup.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_PROCESS_ADHOC_JOB	Ad-hoc process job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_process.ksh
PMO_EVENT_IND_RDF_START_JOB	Start job for importing event indicators for preprocessing	rse_process_state_update.ksh
PMO_EVENT_IND_RDF_SETUP_JOB	Setup job for importing event indicators for preprocessing	pmo_event_ind_rdf_setup.ksh

Table 13–10 (Cont.) Jobs for Importing Forecast Parameters from RDX to AIF

JobName	Description	RMSBatch
PMO_EVENT_IND_RDF_ PROCESS_JOB	Process job for importing event indicators for preprocessing	pmo_event_ind_rdf_process.ksh
PMO_EVENT_IND_RDF_END_ JOB	End job for importing event indicators for preprocessing	rse_process_state_update.ksh
PMO_EVENT_IND_RDF_SETUP_ ADHOC_JOB	Ad-hoc setup job for importing event indicators for preprocessing	pmo_event_ind_rdf_setup.ksh
PMO_EVENT_IND_RDF_ PROCESS_ADHOC_JOB	Ad-hoc process job for importing event indicators for preprocessing	pmo_event_ind_rdf_process.ksh
RSE_LIKE_RDX_RSE_START_ JOB	Start job for importing new item parameters	rse_process_state_update.ksh
RSE_LIKE_RDX_RSE_SETUP_ JOB	Setup job for importing new item parameters	rse_like_rdx_rse_setup.ksh
RSE_LIKE_RDX_RSE_PROCESS_ JOB	Process job for importing new item parameters	rse_like_rdx_rse_process.ksh
RSE_LIKE_RDX_RSE_END_JOB	End job for importing new item parameters	rse_process_state_update.ksh
RSE_LIKE_RDX_RSE_SETUP_ ADHOC_JOB	Ad-hoc setup job for importing new item parameters	rse_like_rdx_rse_setup.ksh
RSE_LIKE_RDX_RSE_PROCESS_ ADHOC_JOB	Ad-hoc process job for importing new item parameters	rse_like_rdx_rse_process.ksh

The approved forecast is exported out of RDX into AIF, which then goes to a flat file. The jobs are listed in Table 13–11.

Table 13–11 Approved Forecast Export Jobs

JobName	Description	RMSBatch
RSE_RDX_APPD_FCST_START_ JOB	Start job for exporting approved forecast data	rse_process_state_update.ksh
RSE_RDX_APPD_FCST_SETUP_ JOB	Setup job for exporting approved forecast data	rse_rdx_appd_fcst_setup.ksh
RSE_RDX_APPD_FCST_ PROCESS_JOB	Process job for exporting approved forecast data	rse_rdx_appd_fcst_process.ksh
RSE_RDF_APPR_FCST_EXPORT_ JOB	Export approved weekly forecast to a flat file	rse_rdf_appr_fcst_export.ksh
RSE_RDF_APPR_FCST_DAY_ EXPORT_JOB	Export approved daily forecast to a flat file	rse_rdf_appr_fcst_day_export.ksh
RSE_RDX_APPD_FCST_END_JOB	End job for exporting approved forecast data	rse_process_state_update.ksh
RSE_RDX_APPD_FCST_SETUP_ ADHOC_JOB	Ad-hoc setup job for exporting approved forecast data	rse_rdx_appd_fcst_setup.ksh
RSE_RDX_APPD_FCST_ PROCESS_ADHOC_JOB	Ad-hoc process job for exporting approved forecast data	rse_rdx_appd_fcst_process.ksh
RSE_RDF_APPR_FCST_EXPORT_ ADHOC_JOB	Ad hoc job to export approved weekly forecast to a flat file	rse_rdf_appr_fcst_export.ksh
RSE_RDF_APPR_FCST_DAY_ EXPORT_ADHOC_JOB	Ad-hoc job to export approved daily forecast to a flat file	rse_rdf_appr_fcst_day_export.ksh

Workflow for RDF Implementation

The AIF-RDF workflow can be implemented in two ways.

The main dependency for the first process is that even before exporting first round of forecast from AIF to RDX, parameters need to be imported from RDX to AIF (the reason for this is RDF prefers to receive forecast for a subset of the prod/locs for which forecast is generated in AIF). For the import of parameters to work properly, run types must be mapped to RDF and assigned an ext key. Here are the steps for the first process:

1. Create run types in AIF in Setup train stop screen.
2. Create test runs in AIF in Test train stop screen to see if the setup has been correct.
3. Map run types to RDF and assign ext keys in Map train stop screen.
4. Import parameters from RDX to AIF (RDF sends parameters using the same ext keys used in Step 3).
5. Run forecasts again in Test train stop screen (this time, the forecasts will use the imported parameters).
6. Approve the estimation and forecast runs in Test train stop screen.
7. Activate the run types in Manage train stop screen.
8. Export forecasts from AIF to RDX.
9. Cycle continues:
    a. Import parameters from RDX to AIF.
    b. Run forecasts.
    c. Approve runs.
    d. Export forecasts from AIF to RDX.

In the second process, there is no dependency of importing parameters from RDX to AIF before exporting forecast from AIF to RDX. (This simplifies the workflow and saves time during implementation; however, the batch processes will still have the dependency.) Here are the steps for the second process:

1. Create run types in AIF in Setup train stop screen.
2. Create test runs in AIF in Test train stop screen to see if the setup has been correct.
3. Approve the estimation and forecast runs in Test train stop screen.
4. Activate the run types in Manage train stop screen.
5. Map run types to RDF and assign ext keys in Map train stop screen.
6. Export forecasts from AIF to RDX (this will export all prod/locs for which AIF has generated forecast and not just a subset; however, this export will take longer time because it's a huge amount of data to be exported -- so, here is the tradeoff between the two implementation processes).
7. Cycle continues as in Step 9 of the first process.

The internal (w.r.t. AIF) difference between the two processes is as follows: When the table in AIF that stores imported parameters from RDX is empty, AIF will export forecast for all prod/locs (second process); otherwise, AIF will export forecast for only a subset of prod/locs as imported from RDX (first process).

Note that once a forecast run has been exported from AIF to RDX, it can't be exported again. For an active run type, the export code finds the latest forecast run that has completed, has been approved, and has not been exported before.

Using the Add Multiple Run Types feature

An "Add Multiple Run Types" feature is available in the Setup train stop within the Manage Forecast Configurations screen. To add/edit/delete the rows visible within the "Add Multiple Run Types" table, please edit the two tables, RSE_CUSTOM_PROCESS_HDR and RSE_CUSTOM_PROCESS_DTL, available in the Manage System Configurations screen.

Building Alternate Hierarchy in AIF

AIF has the ability to build alternate location and alternate product hierarchies. Alternate hierarchy information should be available in RI. RSE_ALT_HIER_TYPE_STG and RSE_ALT_HIER_LEVEL_STG are the two relevant tables available in the Manage System Configurations screen. It is possible to add/edit/delete rows in these tables. Information about the alternate hierarchy types and levels need to be provided through these tables. Then, the alternate hierarchy jobs need to be executed to generate the alternate hierarchies in AIF. Table 13–12 lists the relevant jobs.

Table 13–12 Alternate Hierarchy Jobs

JobName	Description	RMSBatch
RSE_ALT_HIER_SETUP_START_JOB	Start Job for Alternate Hierarchy Type Setup	rse_process_state_update.ksh
RSE_ALT_HIER_LOAD_JOB	Setup alternate hierarchy types by loading data from RSE_ALT_HIER_TYPE_STG table	rse_alt_hier_load.ksh
RSE_ALT_HIER_SETUP_END_JOB	End Job for Alternate Hierarchy Type Setup	rse_process_state_update.ksh
RSE_ALT_LOC_HIER_START_JOB	Start Job for Alternate Location Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_LOC_HIER_SRC_XREF_LOAD_JOB	Load the RSE_LOC_SRC_XREF table with alternate location hierarchy data	rse_alt_loc_hier_process.ksh
RSE_ALT_LOC_HIER_LOAD_JOB	Load the RSE_LOC_HIER table with alternate location hierarchy data	rse_alt_loc_hier_load.ksh
RSE_ALT_LOC_HIER_TC_LOAD_JOB	Load the RSE_LOC_HIER_TC table with alternate location hierarchy data	rse_alt_loc_hier_tc_load.ksh
RSE_ALT_LOC_HIER_DH_LOAD_JOB	Load the RSE_LOC_HIER_DH table with alternate location hierarchy data	rse_alt_loc_hier_dh_load.ksh
RSE_ALT_LOC_HIER_END_JOB	End Job for Alternate Location Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_PROD_HIER_START_JOB	Start Job for Alternate Product Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_PROD_HIER_SRC_XREF_LOAD_JOB	Load the RSE_PROD_SRC_XREF table with alternate product hierarchy data	rse_alt_prod_hier_process.ksh
RSE_ALT_PROD_HIER_LOAD_JOB	Load the RSE_PROD_HIER table with alternate product hierarchy data	rse_alt_prod_hier_load.ksh

Table 13–12 (Cont.) Alternate Hierarchy Jobs

JobName	Description	RMSBatch
RSE_ALT_PROD_HIER_TC_LOAD_JOB	Load the RSE_PROD_HIER_TC table with alternate product hierarchy data	rse_alt_prod_hier_tc_load.ksh
RSE_ALT_PROD_HIER_DH_LOAD_JOB	Load the RSE_PROD_HIER_DH table with alternate product hierarchy data	rse_alt_prod_hier_dh_load.ksh
RSE_ALT_PROD_HIER_END_JOB	End Job for Alternate Product Hierarchy Load	rse_process_state_update.ksh

To perform forecasting in AIF using the created alternate location (product) hierarchy, the PMO_LOC_HIER_TYPE (PMO_PROD_HIER_TYPE) parameter value in RSE_CONFIG table (APPL_CODE = PMO) available in the Manage System Configurations screen must be set up correctly.

Custom Jobs Through Innovation Workbench (IW)

Custom processes can be created through IW. AIF provides three in-built custom jobs that can be used for setting up some custom steps which will be executed before the estimation runs for forecasting. Basically, anything that can enhance the estimation results should be setup using these jobs. Table 13–13 lists the relevant jobs.

Table 13–13 Custom Jobs for Pre-Estimation Custom Processes

JobName	Description	RMSBatch
PMO_PRE_ESTIM_CUSTOM_START_JOB	Start job for PMO pre-estimation custom process	rse_process_state_update.ksh
PMO_PRE_ESTIM_CUSTOM_1_JOB	PMO pre-estimation custom job 1	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_2_JOB	PMO pre-estimation custom job 2	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_3_JOB	PMO pre-estimation custom job 3	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_END_JOB	End job for PMO pre-estimation custom process	rse_process_state_update.ksh
PMO_PRE_ESTIM_CUSTOM_1_ADHOC_JOB	PMO pre-estimation custom ad hoc job 1	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_2_ADHOC_JOB	PMO pre-estimation custom ad hoc job 2	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_3_ADHOC_JOB	PMO pre-estimation custom ad hoc job 3	rse_custom_job.ksh

Here are the steps to get the custom jobs working:

1. In IW, create a package and procedure for the process that needs to be run as a custom job. After logging into Innovation Workbench -> Manage Workbench -> SQL Workshop, on the right-hand side click on the "Package" under "Create Object" and procced with creating the package body, specification and create the procedure inside the package.
2. Using IW or the Manage System Config screen, modify the table RSE_CUSTOM_JOB_CFG and edit values for the following columns:

a. OB_NAME: This value is not editable and indicates a placeholder where a custom process can be added.

b. PACKAGE_NAME: Enter the name of package that was created in IW.

c. PROCEDURE_NAME: Enter the name of procedure that was created in IW.

d. PROCEDURE_DESCR: Enter a description if desired.

e. RUN_TIME_LIMIT: The run time limit is 900 seconds by default. It can be changed to a smaller value but not to a larger value. If the custom process runs for longer than the value indicated in RUN_TIME_LIMIT when running as a part of the batch process, the custom process will stop and move on to the next job/process.

f. CONNECTION_TYPE: Valid values are LOW and MEDIUM. This value should almost always be LOW unless the job is supposed to run a process that would need multiple threads. HIGH is not a valid value. If HIGH is entered, it will switch to LOW by default when the job runs.

g. ENABLE_FLG: Set this value to Y to indicate that this job should be executed as part of the batch process.

14

Inventory Optimization

This chapter describes the Oracle Retail AI Foundation Inventory Optimization Cloud Service.

Overview

Inventory Optimization (IO) determines the optimal replenishment policies, reorder point (RP), and receive up-to level (RUTL), at the item/location level for both store and warehouse locations. The optimization engine uses simulation-based optimization and machine learning methods to calculate the trade-offs between the service level and the inventory cost, and the trade-off analysis is leveraged to generate the optimal replenishment policies for achieving a desired target service level. These data-driven policies are pushed to Oracle Retail Merchandising System (RMS) to generate and execute purchase orders and transfers. To provide full visibility, the replenishment policies are also used within IO to calculate optimal transfers and purchase orders. In addition, IO recommends optimal rebalancing transfers between stores to increase sell-through and to avoid markdowns. This type of strategy can be turned off when not applicable (for example, for grocery categories).

IO leverages historical sales and inventory, business requirements such as lead time and review schedule, and the demand forecast to optimize the replenishment policies. The demand forecast that is generated by the forecast engine within AI Foundation considers different factors such as price effect, holidays, and promotions, and variation across customer segments.

The conceptual flow of different components in IO is as follows. AIF/RI is the core data foundation layer that consumes the retailer data, including merchandise and location hierarchy and historical sales and inventory. The IO algorithm obtains inputs such as replenishment strategies and objectives from the UI and from the strategy rules interface that can be used for specifying strategies that are not supported in the UI, along with replenishment attributes such as lead time, review schedule, and presentation stock from AIF/RI. The other key input to IO is the demand forecast, which is generated and provided by the AIF forecast engine.

The simulation-based optimization and machine learning algorithms in IO analyze historical data and different future demand scenarios to calculate the trade-offs between service level and inventory cost and generate optimal replenishment policies (that is, the Re-order Point (RP) and the Receive up-to Level (RUTL) for each item-location). These optimal policies, along with data for inventory positions and replenishment attributes such as minimum order quantity, are leveraged to determine the recommended transfers and purchase orders (PO). The optimal replenishment policies as well as the recommended transfers and purchase orders can be viewed in the IO UI. The replenishment policies that are auto-approved or are approved by the user will be exported to the retailer's replenishment system, such as RMS, to generate and execute orders. The user can also review and submit the transfers and purchase orders within IO. In this release, the recommendations for purchase orders and replenishment transfers (warehouse-to-store and warehouse-to-warehouse) are not integrated with the retailer's order execution system and primarily provide information about the time phase view of orders. The submitted recommendations will not be pushed to other systems.

The IO optimization algorithm also analyzes various rebalancing transfers efficiently and generates optimal transfers that are in line with the retailer's objective, such as minimizing total transfer costs and maximizing sell-through. The recommended rebalancing transfers can be reviewed and edited in the IO UI. The replenishment policies that are auto-approved or are approved by the user will be exported to the retailer's order management system, such as RMS or any external system, to generate and execute orders. If the rebalancing transfers are submitted within IO, they will be exported to the retailer's replenishment system for final approval by the user and then execution.

Inventory Optimization Runs

IO supports two kinds of runs: user runs and batch runs. Batch runs are scheduled to run automatically at regular intervals (for example, weekly or daily). User runs are typically created by the user to do what-if analysis and/or to override the recommendations generated by the latest batch run. IO runs generate recommendations at the item-location level. The items that are included in a run can be selected by specifying one, a multiple, or all of the nodes at a higher merchandise level, such as department. This level is configurable. Similarly, the locations that are to be included in a run can be selected by specifying one, a multiple, or all of the nodes at a higher location level, such as area. This level is configurable.

Each run, using the latest sales data and inventory levels, updates the replenishment policies and PO/transfer recommendations. The number of item-locations that are processed in each run can vary, based on the retailer's review schedule. During each run, only the item-locations that are due for review for replenishment on the next day are processed and their replenishment policies are updated. For example, a run that is scheduled for a nightly batch on Sunday 01/08/2023 would process and update the policies for item-locations that have a next review date of Monday 01/09/2023 in the retailer's replenishment system. This allows the user to review and approve the policies that were not auto-approved on Monday before they are pushed to the replenishment system for generating and executing orders.

An analyst can review the results of runs and approve the replenishment policies for each item-location. In addition, the analyst can change the target service level strategy for the item-location or at higher levels of merchandise-location based upon the cost-service level trade-offs. Changes in service level strategies will take effect from the next batch run onwards. The user can also review the PO and transfers and change the order quantity, submit the recommended order, or approve the recommended order. Submitted recommended orders are exported to the order execution system in worksheet mode and require final approval by the end user within the order execution system. Approved recommendations are exported to the order execution system in final approved mode and are executed on the order date. In this release, the approve action is available only for rebalancing transfers and for replenishment policies.

Inventory Optimization UI Workflow

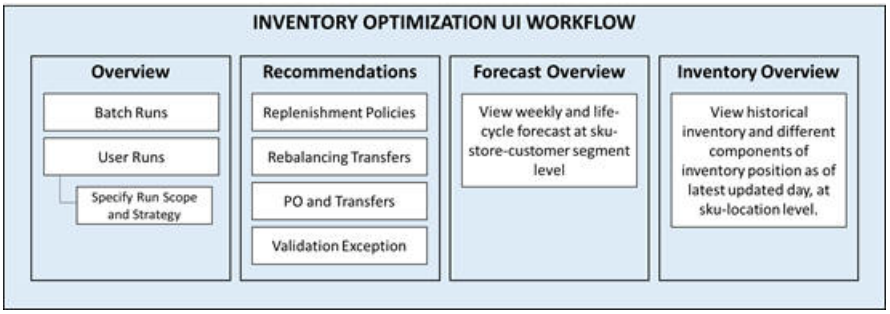
Figure 14–1 shows an overview of the IO UI workflow, which consists of the following:

- Run Overview: This is the dashboard for the IO runs. In this tab, the user can see a list of all existing runs, along with details that describe each run. Runs have four different statuses: Setup, In Progress, Successful, and Failed.

  From the run overview table, the user can create a run, copy a run, open a run, or delete a run. When the user clicks on a successful run, it is opened in a new tab with three sub-tabs to show replenishment policies, rebalancing transfers, and PO and transfers, as well as a sub-tab to show the summary of inputs of the run. For a failed run, only the input summary sub-tab is displayed. To create a run, the user must specify the run scope and strategy. As part of the run scope, the user specifies the areas and departments for the run and the type of analysis to be performed. As part of run strategy, the user can specify the business rules.

- Recommendations: This is the main screen where the user can view, override, and submit/approve different types of recommendations. This screen shows the approved/submitted recommendations across all runs and the ready for review recommendations from the most recent run. This screen consists of five sections.
  - Overview–A calendar view that shows the summary of different types of recommendations and provides a link for the user to open and view the details of recommendations for each date.
  - Replenishment Policies–Used to review and approve replenishment policies and to set service level strategies.
  - Rebalancing Transfers–Used to review, override, and submit/approve rebalancing recommendations.
  - PO/Transfers–Used to review, override, and submit recommended PO and transfers. These include warehouse/supplier to store and warehouse/supplier to warehouse.
  - Validation Exception–When the user overrides the recommended order quantity for rebalancing, PO, or transfers, the overridden value may exceed the available supply at the origin or may be more or less than the quantity required at the destination. In the validation exception section, the user can see these types of warnings and revert the override if desired.
- Inventory Overview–Used to view the inventory levels and the historical trend of the inventory at the item-location level.
- Forecast Overview–Used to view the demand forecast of the item-locations and the historical trend of sales across all customers as well as for each customer segment.

Figure 14–1 Inventory Optimization UI Workflow



The following sections explain the main implementation tasks that involve configuring the following:

- The roles and permissions assigned to users.
- The loading of retailer data.
- The configuration parameters.
- The business rules that determine constraints that the application takes into account during the optimization process.

Security

User roles are used to set up application user accounts through Oracle Identity Management (OIM). See *Oracle Retail AI Foundation Cloud Services Administration Guide* for details. The following roles are required.

- ADMINISTRATOR_JOB in order to access the Control & Tactical Center
- INVENTORY_ANALYST_JOB in order to access the Inventory Optimization application.

In addition to above roles, verify the following:

- Access to Innovation Workbench and/or Data Visualizer: This is necessary in order to query or visualize the data and verify that the data loaded matches the desired expectations.
- Access to POM to execute ad-hoc and batch jobs: The POM UI URL is something like <host>/POMJetUI. If the user cannot access the POM UI, contact the administrator to obtain the relevant access/user roles.

Data Load Requirements

This section provides information about setting up the data that the IO application uses, including guidelines regarding the expectations for the data element requested and where it is used. Information about these files can be found in *Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface*.

Most of the AI Foundation Cloud Services (AIF) data is pushed in two-step process. Any additional IO-specific data is directly pushed into the AIF.

1. First, data is loaded, using CSV and W_ interfaces, into Retail Insights Cloud Service (RI), using appropriate Retail Analytics & Planning (RAP) jobs. Then, RADM_REFRESH_JOB must be run to refresh the table statistics before any AIF job is run (though REFRESH_ RADM_JOB is automatically executed as part of most ad hoc processes in RI).
2. Second, appropriate AIF jobs are used to push data into AIF. At the time of implementation, the user will only use ADHOC jobs, not any batch jobs. Batch jobs (described in "POM Jobs") are necessary to put the system on a batch schedule.
3. Third, IO-specific data (that is not available in RI interfaces) is directly pushed into the AIF.

Note that some of the jobs require relevant configuration parameters to be specified with client-specific values. If incorrect configuration values are used, a job may run without errors, but will not produce the desired data in the target tables. If the client or implementation team wants to view the progress of the job or any errors in order to file an SR, then AIF provides database logging in a table called RSE_LOG_MSG. Logging can be enabled only with a service request from the client/implementation partner to the support team.

RAP Foundation Data (CSV and W_ interfaces)

These interfaces provide the foundation data for the AI Foundation Cloud Services. First, the user must execute and verify that all the RAP foundation data has been loaded. For details, the RAP Implementation Guide can be accessed at:

https://docs.oracle.com/en/industries/retail/retail-analytics-platform/21.0/rapig/data-load-init-batchproc.htm#data-load-init-batch-proc-F30BF774

For Inventory Optimization, the following data is required:

- Product Hierarchy. When Style, Style Color must be loaded, appropriate columns must be populated.
- Location Hierarchy
- Sales. It is recommended to load at least 24 months of historical data to obtain a good signal in the model training and a reliable parameter estimation for the demand forecasting.

- Inventory. It is a critical data element for IO and for demand forecasting of fashion items that are considered short life cycle.
- Inventory Receipts. This data is used to identify when the item started selling. This data must be provided for forecasting fashion items that are considered short life cycle.
- Customer Segments. Even when the retailer is not planning to load or use customer segments, a dummy record is required. RI provides the ability to insert a dummy record, and it is not required for the client or implementer to provide these interfaces: W_PARTY_PER_DS and W_RTL_CUSTSEG_DS.
- Price and Cost. This data is required for demand forecasting and for calculating some of the KPIs that are shown in IO screens.
- Product Images. The URLs for rendering product images in IO screens can be provided.

Before pushing any data into the AIF, it is critical to make sure RAP foundation data is accurate and verified. Verification can be done through Data Visualizer or through the Innovation Workbench. Re-loading some of the RAP foundation data elements is non-trivial as it will require an SR to reset/truncate some of the target tables.

RSP Foundation Data (RSE_% Tables)

Once the interface configuration is complete and the RAP Foundation data has been verified, the user can push the data from RAP into AIF. As mentioned before, once database logging has been enabled, users can check for any errors and the progress of the process in RSE_LOG_MSG. Before running the RSE_MASTER_ADHOC with appropriate flags to execute the following jobs, the user must set the correct values for configuration parameters in RSE_CONFIG.

Inventory Optimization Data (IO_% Tables)

Once RSE_MASTER_ADHOC is successfully run, data can be loaded to IO tables by running the IO_MASTER_ADHOC.

Data Input Requirements

This section describes the data requirements.

Hierarchy Data

Hierarchies are part of the core data elements that are used in Inventory Optimization, in both the forecasting and optimization modules. The four types of hierarchies are Location Hierarchy, Merchandise Hierarchy, Calendar Hierarchy, and Customer Segments Hierarchy. Most of the W_% interfaces have numerous columns and thus, it is possible to define which columns will be populated in the interface by specifying the column headers (the order of columns does not have to be the same) in the corresponding W_%.ctx file.

Location Hierarchy

Here is an example of a location hierarchy: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE. The run's scope level must match a node in the location hierarchy. For example, you can specify the run scope at the Area level and include one, multiple, or all areas in the run. Batch runs by default include all values of the scope level (for example, all areas). Note that the E-com channel can be defined as part of the location hierarchy. Relevant interfaces are: W_INT_ORG_ATTR_DS, W_INT_ORG_DHS, W_INT_ORG_DS.

Merchandise Hierarchy

An example of merchandise hierarchy is as follows: CHAIN 'COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE 'COLOR ' SIZE (SKU). There can be up to nine levels in the merchandise hierarchy. When Inventory Optimization is used for fashion apparel, it is expected that Style and Color are provided through the relevant interfaces. The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_DS interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_ DHS interface. In addition, there must be consistency among the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_DS is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy. If the retailer wants to use an extended merchandise hierarchy, then the retailer must populate this interface. Here are the specific fields:

- PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)

- PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)

- PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699). The value of PROD_NUMs is the same as the value in the W_PRODUCT_ DS.PROD_NUM interface.

Calendar Hierarchy

This is one of the core hierarchies. A retailer can specify the calendar depending on that retailer's business requirements (for example, fiscal calendar). The relevant interface for this is W_MCAL_PERIOD_DS.

Customer Segments Hierarchy

The AIF forecast engine in IO supports demand forecasting at the customer segment level. This is leveraged by the optimization algorithm to recommend customer-aware rebalancing transfers. To use this functionality, the retailer must load customer segments. The relevant interfaces to supply this data are: W_RTL_CUSTSEG_DS and W_RTL_CUST_CUSTSEG_ DS.

Runs in IO can be set up to run at configured levels for location and merchandise.

- **Location Level**. The location level of a run's scope is configurable in RSE_CONFIG, and it is denoted as IO_LOC_HIER_FILTER_LVL. For example, a typical level is Area.
- **Merchandise Level**. The merchandise level of a run's scope is configurable in RSE_ CONFIG, and it is denoted as IO_PROD_HIER_FILTER_LVL. For example, a typical level is Department.
- **Calendar Level**. The calendar processing level is configurable in RSE_CONFIG, and it is denoted as IO_CAL_HIER_PROCESSING_LVL. For example, a typical level is Day.

Retail Sales Data

Retail Sales data is a required interface and is provided through W_RTL_SLS_TRX_IT_LC_ DY_FS. IO requires the retailer's historical sales data at the SKU-Store-Day level. When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Inventory Data

Historical inventory data is a required at two levels: the historical weekly inventory at sku-location level and the inventory at sku-location for the latest day. The latest weekly

inventory data is obtained as part of the weekly batch process. The daily inventory data for the latest day is obtained through the daily batch process. In both interfaces, various components of inventory, including on-hand, on-order, back-order, and so on, are required. Historical weekly inventory data as well as daily inventory data can be specified through this interface: W_RTL_INV_IT_LC_DY_FS.

Product Attributes

Product attributes are not required for IO, but they are useful for the demand forecasting, and can be used for demand forecasting of new items. Product attributes must use W_RTL_ITEM_GRP1_DS since it can support any number of attributes.

If the retailer wishes to load STYLE or COLOR as product attributes, then the W_RTL_ITEM_GRP1_DS interface is used to provide style and color attributes for the different products, using a value in PROD_GRP_TYPE of either STYLE or COLOR. The actual values for the Styles and Colors are provided in columns FLEX_ATTRIB_2_CHAR and FLEX_ATTRIB_4_CHAR using values that represent the ID for the Style (for example, 1234), and a description for the Style (for example, Loose Fit). The columns FLEX_ATTRIB_3_CHAR and FLEX_ATTRIB_10_CHAR contain the appropriate designation for STYLE or COLOR.

Product Images

The Images URL can be provided through W_RTL_PRODUCT_IMAGE_DS.dat interface. This interface has a column called PRODUCT_IMAGE_ADDR that contains the full URL to an image of the product. This URL must be in the following format: http[s]://servername[:port]/location/filename.extension

For example:

- PRODUCT_IMAGE_NAME = imagename.png
- PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png
- PRODUCT_IMAGE_DESC= Short description of the image

Replenishment Attributes

Replenishment attributes such as lead time, review time, presentation stock, and so on, are required for any sku-location that is expected to be processed by IO to generate an optimal replenishment policy. These attributes must be provided at sku-location level through W_INVENTORY_PRODUCT_ATTR_DS:

Price and Cost

Price and cost data at the end of the week for each item-location is required. This data can be loaded through RI. Alternatively, a retailer can directly load price and cost data to AIF using the W_RTL_PRICE_IT_LC_DY_FRI interface. Alternatively, a retailer can directly load price and cost data to AIF using the RSE_PRICOST_PR_LC_WK_STG interface.

Season

Data for the name of the season along with start and end dates can be provided through IO_SEASON_STG. If the retailer does not have any seasons, then the retailer can define one long season.

Global Configurations

Table 14–1 shows the configurations that must be provided at the time of implementation. These configurations can be modified in RSE_CONFIG table using the Manage System Configuration screen under Control & Tactical Center in the AIF dashboard.

Table 14–1 Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
RSE	EXTENDED_HIERARCHY_ SRC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PROD_ HIER	Default value is Y. If the prod-uct hierarchy data had 9 levels, keep this value as Y. If it has 7 levels, change this value to N.	Y/N Value. This parameter is used by the product hierarchy ETL to know if the extended product hi-erarchy is needed.
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the prod-uct hierarchy data has 9 levels (i.e. it has extended hierar-chy), keep this value as 3. If it has 7 levels, change this value to 1.	The hierarchy id to use for the product (Installation configuration).
PMO	PMO_AGGR_INVENTORY_ DATA_FLG	Default value is Y. Set this value to N if inventory data is not loaded (inventory data is not required for MFP forecasting but it is required for other applications like Offer Optimization, Inventory Optimization and Retail Demand Forecasting).	Specifies if inventory data is pre-sent and if it should be used when aggregating activities data.
RSE	PROD_HIER_SLSTXN_HIER_ LEVEL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color or 9 Style/color/Size). It MUST match the extended hierarchy leaf level.
IO	INV_REBALANCE_FLG	Y	Indicates whether inventory re-balancing analysis should be per-formed in batch runs. Also the inventory rebalancing section in the UI create run screen will be shown or hidden based on this flag. Must be Y or N.
IO	IO_CURRENT_DATE	A Dummy Date that represents the Current Date for the Inventory Optimization process.	This date is to be used for Dev/Test/Demo purposes because of our use of Historical Data. Date format YYYYMMDD. In production environment and for go live set this value to NULL.

Table 14–1 (Cont.) Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
IO	IO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has 9 levels (i.e. it has extended hierarchy), keep this value as 3. If it has 7 levels, change this value to 1.	Hierarchy Type Identifier for the Product/Merchandise Hierarchy Type.
IO	IO_REBAL_DEF_REGION	usastates	Specifies the default region in the rebalancing details map view
IO	IO_LOC_HIER_FILTER_LVL	3	Level Identifier for the level of the Location Hierarchy at which IO runs can be set up.
IO	IO_PROD_HIER_FILTER_LVL	4	Level Identifier for the level of the Product Hierarchy at which IO runs can be set up.
IO	IO_ALLOCATION_BUFFER_ DAYS	7	The number of days before selling start date that the initial shipment must be received. This is used for initial allocation recommendations.
IO	IO_AUTO_APPR_NEW_ PARAMS	N	Indicates whether net new pa-rameters in a batch should be au-to-approved. Default to N (No).
IO	IO_AUTO_APPR_ THRESHOLD_PCT	0.2	The threshold percentage change of Reorder Point (RP) and Receive Up To Limit (RUTL) between batches that is acceptable for au-to-approval of the parameters.
IO	IO_BATCH_RUN_INV_REBAL_ FLG	N	Y/N flag that indicates whether inventory rebalancing analysis is performed as part of batch run.
IO	IO_BATCH_RUN_PO_TNSFR_ RECOMM_FLG	Y	Y/N flag that indicates whether po transfer recommendation is performed as part of batch run.
IO	IO_BATCH_RUN_TIME_ PHASED_FLG	Y	Y/N flag that indicates whether time phased simulation is performed as part of batch run.
IO	IO_BATCH_RUN_TRADEOFF_ SIM_FLG	N	Y/N flag that indicates whether trade-off simulation is performed as part of batch run.
IO	IO_CAL_HIER_PROCESSING_ LVL	4	Level Identifier for the level of the Calendar Hierarchy at which we process Inventory. Default 4 is to process at Week Level.
IO	IO_TIME_PHASED_HORIZON	14	Number of periods in the future for which the time phased plan is generated.

Inventory Optimization Integration with RMS

Figure 14–4 shows the conceptual diagram for the integration of IO with RMS. When a sku-store is set up in RMS for replenishment, if the optimization flag in RMS is set to Y, the

optimal replenishment policies for that sku-store is sent back to RMS via Web Service integration.

Additional configuration parameters must be set to integrate IO with RMS. The configuration parameters that must be set for integration of replenishment policies and rebalancing transfers are listed in Table 14–2 and Table 14–3, respectively.

In addition, the RMS username and password must be added under the credential store for Merchandise RMS Access. You can modify these values using the UI screen for Data Management -> Manage Credential Store, as shown in Figure 14–2 and Figure 14–3. Data Management screen is available under the task menu in the main dashboard.

The replenishment policies and rebalancing transfers that are pushed to RMS can be viewed in io_repl_param_export_vw and io_rebalance_export_vw, respectively.

Table 14–2 Configuration Parameters for Integration of Replenishment Policies with RMS

Parameter Name	Parameter Value	Description
IO_MERCH_RMS_ CONTEXT_ROOT	RmsReSTServices/services/private This value must be set based on the client environment.	Context root of the RMS Restservice url.
IO_MERCH_RMS_FLG	Y	Flag to identify whether to publish replenishment parameter to RMS.
IO_MERCH_RMS_ HOSTNAME	msp00aay.us.oracle.com This value must be set based on the client environment.	URL host name to publish replenishment parameter to RMS. This must be same as CN name in the certificate.
IO_MERCH_RMS_PORT	8002 This value must be set based on the client environment.	Port number to publish replenishment parameter to RMS.
IO_MERCH_RMS_REPL_ PATH	inventory/replenishment/create ReplSched	Path of the Restservice e.g., inventory/replenishment/create ReplSched.
IO_MERCH_RMS_VERSION	16.0 This value must be set based on the client environment.	RMS Version to use for publishing replenishment parameter to RMS (format 16.0).
IO_MERCH_RMS_AUTH_ SERVICE_TYPE	OAUTH	Used in determining whether to use OAuth or basic authentication for IO RMS integration.

Table 14–3 Configuration Parameters for Integration of Rebalancing Transfers with RMS

Parameter Name	Parameter Value	Description
IO_REBAL_TRANSF_ MERCH_RMS_FLG	Y	Flag to identify whether to publish rebalancing transfers to RMS.
IO_REBAL_TRANSF_ MERCH_RMS_HOSTNAME	msp00aay.us.oracle.com This value must be set based on the client environment.	URL host name to publish rebalancing transfers to RMS. This must be same as CN name in the certificate.

Table 14–3 (Cont.) Configuration Parameters for Integration of Rebalancing Transfers with RMS

Parameter Name	Parameter Value	Description
IO_REBAL_TRANSF_RMS_PORT	443 This value must be set based on the client environment.	Port number to publish rebalancing transfers to RMS
IO_REBAL_TRANSF_RMS_SERVICE_VERSION	v1 This value must be set based on the client environment.	Transfer Management Service Version to use for publishing rebalancing transfers to RMS (format v1).
IO_TRAN_NUM_LOWER_LIMIT	9000000000	Lower range dedicated to the ID for store-to-store transfers that are generated in IO and sent to RMS.

Figure 14–2 and Figure 14–3 show adding a username and password for the credential store.

Figure 14–2 Selecting the Credential Store

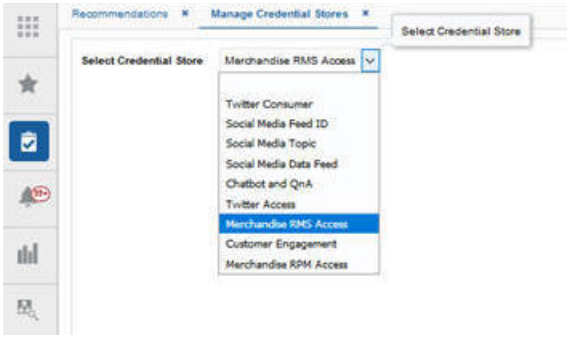


Figure 14–3 Adding the Username and Password for the Credential Store

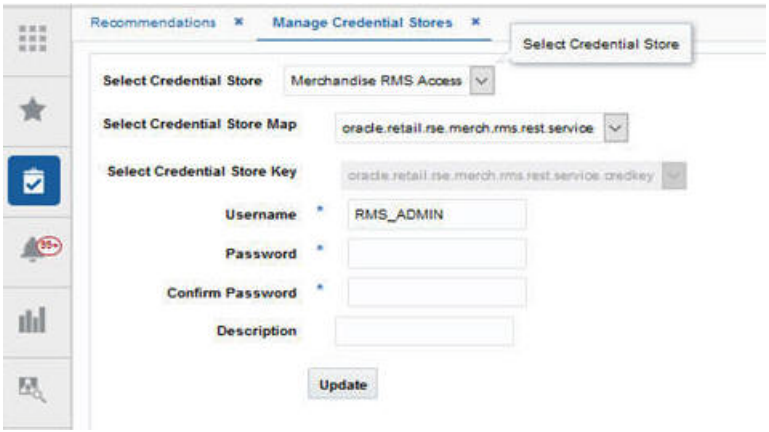
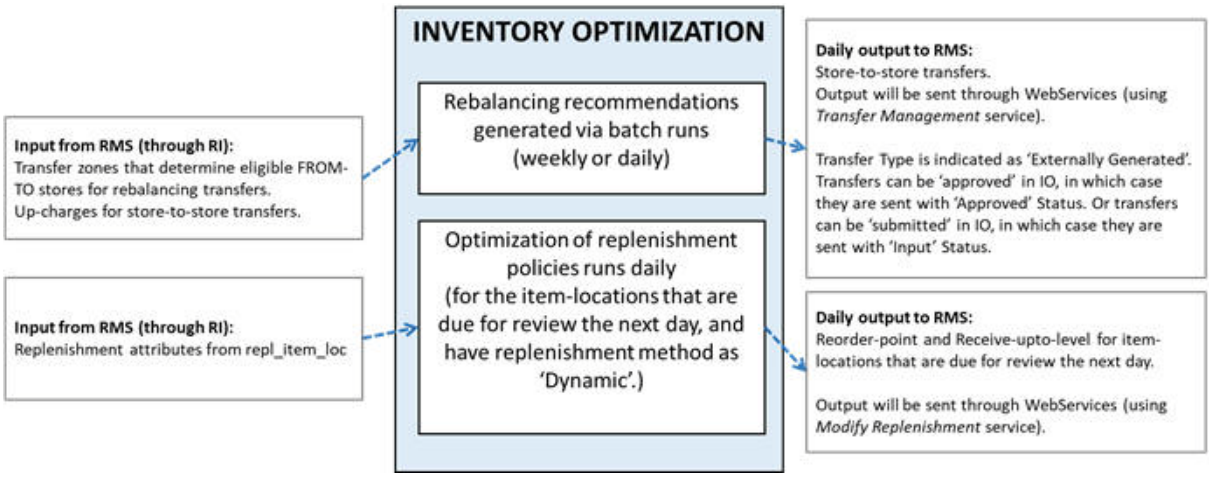


Figure 14–4 Inventory Optimization Integration with RMS



Forecast Engine Setup for IO

The forecast engine that generates demand forecast and feeds into the Inventory Optimization consists of two modules: parameter estimation and demand forecasting. Parameter estimation uses the historical data to determine the parameters, including seasonality, price elasticity, promotion elasticity, and promotion lift (or holiday lift) values. Demand forecasting leverages the parameters estimated from historical data and in-season sales data (the base demand is re-estimated on a regular basis as part of batch process) to determine the demand and generate the forecast.

To set up the parameter estimation and forecast runs and configure various settings, you can use the Manage Forecast Configurations functionality under Strategy & Policy Management. Based on the configuration settings, parameter estimation can be executed through the UI to initially estimate the parameters and then updated at specified intervals to reflect the latest sales trends. Similarly, a weekly batch job is used to generate demand forecasts every week after the weekly data has been updated.

For details regarding setting up run types and runs and configuring parameters, refer to the Control and Tactical Center chapter in the implementation guide.

Here are the points to take into consideration when setting up forecast runs types for inventory optimization.

- Forecast level: data must be aggregated at the lowest level (Sku/store/week level) and the option for spreading the forecast to day level must be enabled when setting up the run type. If customer segment data is available and is desired to be used as additional dimension in the forecast, enable the option for customer segments.
- Forecast method: if the items are fashion/short life cycle, select Causal-Short Life Cycle. If items are basic items which live for at least 16 weeks and often longer, select Causal-Long Life Cycle.
- Forecast measure: select Total Gross Sales Units.
- Spread Forecast to Day level: must be enabled.
- Spread Profile Level: Spread profile can be provided through rse_fcst_spread_profile_stg.txt. If the spread profile is not available, leave the profile level selection empty. In such case, a default uniform profile will be applied for spreading forecast from week to day level.
- Run the data aggregation in the Setup train stop.

- Once the aggregation is complete, use Test train stop to create an ad-hoc run. Select Run estimation and base demand and enable Run forecast. Once the run is successfully complete, if desired, you can review the results using Innovation Workbench.
- Use the Manage train stop to activate the run type and override the configurations for the run type if desired. In addition, turn on auto-approve batch runs.
- Use Map train stop to map the run type to Inventory Optimization.
- Make sure proper POM jobs (listed in Batch and Ad-Hoc Jobs for RDF Forecasting in the Control & Tactical Center chapter) are enabled so the forecast is generated via batch jobs.

Batch and Ad Hoc Jobs for IO

The following jobs with mentioned options must be executed to load required data for IO and forecast engine.

Table 14–4 Batch and Ad Hoc Jobs for IO

JobName	Description	RmsBatch	Parameters
RSE_MASTER_ADHOC_AIF_JOB The batch job is	Run RSE master script	rse_master.ksh	You can run this with parameter '-A' which will run all data loads. You can also provide parameters to load certain data only. These parameters can be found in the AIF Operations Guide. The parameters that are relevant for IO are listed below this table.
IO_MASTER_ADHOC_AIF_JOB	Run IO master script	io_master.ksh	You can run this with parameter '-A' which will run all data loads. Alternatively, you can specify specific parameters to load certain data. List of parameters are listed below this table
IO_WAREHOUSE_LOAD_START_JOB IO_WAREHOUSE_LOAD_JOB IO_WAREHOUSE_LOAD_END_JOB	Load job for warehouses	io_warehouse_load.ksh	
IO_SEASON_LOAD_START_JOB IO_SEASON_LOAD_STG_JOB IO_SEASON_LOAD_COPY_JOB IO_SEASON_LOAD_STG_CNE_JOB IO_SEASON_LOAD_JOB IO_SEASON_LOAD_END_JOB	Load job for seasons	io_season_load.ksh	

Table 14–4 (Cont.) Batch and Ad Hoc Jobs for IO

JobName	Description	RmsBatch	Parameters
RSE_RULES_TO_IO_LOAD_START_JOB RSE_RULES_TO_IO_LOAD_JOB RSE_RULES_TO_IO_LOAD_END_JOB	Job to load RSE rules to IO	rse_rul_eng_io_load.ksh	
IO_CREATE_BATCH_RUN_ADHOC_JOB	IO batch run creation ad hoc job	io_create_batch_run.ksh	
IO_RUN_ANALYTICS_ADHOC_JOB	IO analytics run ad hoc job	io_run_analytics.ksh	
IO_PUBLISH_MERCH_RMS_START_JOB IO_PUBLISH_MERCH_RMS_JOB IO_PUBLISH_REBAL_RMS_JOB IO_PUBLISH_MERCH_RMS_END_JOB	Publish job for pushing replenishment param-eters and Rebalancing transfers to RMS	io_publish_merch_RMS.ksh io_publish_rebal_transfer.ksh	

Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Table 14–5 Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
RSE Product Hierarchy ETL (including CM Groups Hierarchy) (-p)	
RSE Location Hierarchy ETL (-l)	
Trade Area alternate hierarchy (-t)	
RSE Calendar Hierarchy ETL (-d)	
RSE Promotion Hierarchy ETL (-r)	Required if client has promotion data that they want to use in forecasting
RSE Customer Segment Hierarchy ETL (-g)	
RSE Product Attributes (-P)	
Location attributes (-L)	
Like Location / Product data load (-K)	Required if client wants to use like location and/or like product in forecasting
Holiday load (-h)	
Inventory data load (-i)	
Sales transaction data (-x)	
Weekly Aggregate Sales data (Load or Calc) (-w)	
Aggregate Sales data processing (-a)	
Price and Cost data load (-C)	
UDA Load (-u)	
Weekly Return transactions (-T)	

Table 14–5 (Cont.) Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
Weekly Return Aggregation (-e)	
Weekly Sales Return Price Consolidation (-S)	
Promotion data load (-n)	
Daily data load (-D)	
Supplier, supplier item, daily supplier cost and supplier inventory management ETLs (-U)	w_party_org_d, w_party_attr_d, w_rtl_it_supplier_d supplier inventory management is optional data for order scaling rules, which are sourced from w_rtl_repl_sup_ mgmt_d
Seasons, phases and related items ETL (-N)	
Load rules engine data to IO (-j)	Optional. It's needed for loading IO rules that are defined in Rules & Strategies into IO.
Buyer, allocation, purchase order and transfer ETLs (-H)	Optional If provided it can be used for value assessment to compare optimized history with actual history
load forecast spread profiles (-M)	Optional. Source: rse_fcst_spread_profile_stg.txt It's needed for spreading week level forecast to day level. If not provided, a uniform profile will be applied.
Load life cycle classification data (-V)	Optional Source: rse_fcst_life_cycle_clsf_stg.txt This allows user to classify product/locations as being short or long life cycle. By default, all prod-uct/location combinations are considered to have the life cycle classification which is indicated by DFLT_LIFE_CYCLE in RSE_CONFIG. Exceptions to DFLT_LIFE_CYCLE can be provided through rse_fcst_life_cycle_clsf_stg.txt and loaded by running this job. This information is forecast en-gine to apply the proper forecast method ac-cording to the life cycle of the item.

Parameters for running IO_MASTER_ADHOC_AIF_JOB

Table 14–6 Parameters for running IO_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
Warehouses (-W)	Source: w_int_org_attr_d
Replenishment Attributes at Product/Location or Group level (-a)	Source: w_inventory_product_attr_d
Seasons (-s)	Source: io_season_stg.txt Data for the name of the season along with start and end dates can be provided through io_season_stg.txt. If the retailer does not have any seasons, then the retailer can define one long season.

15

Configuration

This chapter describes the major application configuration points, including:

- User Interface Authentication and Authorization
- User Management Configuration: Configuring Users and Roles
- Configuration
- Internationalization
- Affinity Analysis Configurations

> **Note:** Since AA is distinct from the other applications, much of what is described here is not applicable for AA. For clarity, AA implementation, configuration, operations and data model are described separately in Chapter 8, "Affinity Analysis.".

User Interface Authentication and Authorization

> **Note:** For more information, see *Oracle Retail AI Foundation Cloud Services Administration Guide*.

For authorization, the applications have been built with role-based access. Access to application user interface components is done by assigning application roles. Application roles are defined as part of the application and deployed as part of the installation process. Application roles are mapped to enterprise roles during initial environment provisioning. Enterprise roles exist as LDAP groups in OID. Refer to the *Oracle Retail AI Foundation Cloud Service User Guide* for the definition of standard user roles.

User Management Configuration: Configuring Users and Roles

This section provides detailed instructions on setting up enterprise-level user management using Oracle WebLogic 12c with Enterprise Manager. The user management configuration is handled using the WLS Console and the WLS Enterprise Manager (EM).

User Roles

The roles listed in Table 15–1 are required to update the configurations in the data management section of the application. For more information, see *Oracle Retail AI Foundation Cloud Services Administration Guide*.

Table 15–1 Data Management Configuration Roles

Roles	Access to Application Configuration
ANALYTIC_EXPERT_JOB	CDT
ANALYTIC_EXPERT_JOB	DT
FORECAST_MANAGER_JOB	ASO
CLUSTERING_ADMINISTRATOR_JOB	AC
CUSTOMER_SEGMENT_ADMINISTRATOR_JOB	CS
ATTRIBUTE_EXTRACTION_JOB	AE
RETURN_LOGISTICS_JOB	RL
SOCIAL_ANALYTICS_JOB	SA
PRICING_ADMINISTRATOR_JOB	PRO
MARKET_BASKET_ANALYSIS_JOB	MBI

Configuration

This section provides details about application configurations that can be modified as part of a deployment. The list of configurations is limited to those settings that most likely need to be reviewed and adjusted before the applications are used. Some configuration points cannot be adjusted after initial setup, so a careful review of the configurations should be done. For a complete list of configurable values, see the complete list of configuration values in RSE_CONFIG.

Any installation configuration must be set before the application is initialized with data and used.

Note: Before beginning any advanced customization, you must consult with development.

RSE_CONFIG Table

The RSE_CONFIG table is a generic table that contains various application configurations. The DESCR column describes the configuration and provides any relevant information regarding what each specific configuration controls. The APPL_CODE column is a useful filter to organize the configurations by the application module.

The following application abbreviations are used

- AE - Attribute Extraction
- CDT - Customer Decision Trees
- CIS - Advanced Clustering
- DT - Demand Transference
- IO - Inventory Optimization
- MBA - Affinity Analysis

- PMO - Forecasting
- PRO - Offer Optimization
- RL - Return Logistics
- RSE - Retail Science Platform (generic configurations)
- SA - Social Analytics
- SPO - Size Profile Optimization
- SO - Space Optimization
- The PARAM_NAME is the name of the configuration.
- The PARAM_VALUE is the value currently associated with the configuration.
- The CONFIGURABLE_FLG column indicates whether the configuration is safe to change.
- The UPDATEABLE_FLG column is used to control whether application code is able to update the configuration.

> **Note:** Use caution when changing a value that the application code can update after the application has started to be used.

The RSE_CONFIG_CODE table is an extension table to the RSE_CONFIG table. It allows for overrides of certain configurations when an additional qualifier is present. All descriptions for the columns in RSE_CONFIG apply here as well. The additional column CONFIG_CODE allows overriding a value for a specific qualifier. Application code does not always support the use of these configurations, but Affinity Analysis, for example, uses this to set up configurations for different hierarchy levels.

Generic Configuration

The following is a list of configurations that can be adjusted. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='RSE'.

Table 15–2 Generic Configurations

Parameter Name	Example Value	Description	Notes
CAL_PERIOD_LEVEL	4	This is the calendar hierarchy level that is used to drive the processing (installation configuration).	Usually this value should be set to the Fiscal Week level of the Fiscal Calendar. The number 4 refers to the fourth level in RSE_HIER_LEVEL table. This cannot be adjusted after initial setup.
CMGRP_HIER_TYPE	1	The hierarchy ID to use for the CM Group (installation configuration).	If an alternate hierarchy is to be used, then this configuration must specify that hierarchy type. Otherwise it must reflect the primary product hierarchy type. The value represented here relates to the ID in RSE_HIER_TYPE. This cannot be adjusted after initial setup.
CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy where the CM Group level exists (installation configuration).	Once the CMGRP_HIER_TYPE is configured, this level must be set to indicate the level of the hierarchy (RSE_HIER_LEVEL) that defines the categories.
PRIMARY_LANGUAGE_CODE	EN	The name of the language code to use for all RSE data sourced from RI (installation configuration).	Data values stored in the database are not multi-language capable, and are not affected by the UI language settings, like the UI labels are. This setting must select the language code for which data should be shown in the UI.
RA_FISCAL_CAL_ID	1240	ID of the calendar to use from RI since RI supports multiple calendars (installation configuration).	This value must match the ID of the desired fiscal calendar loaded into W_MCAL_PERIOD_DS.MCAL_CAL_ID.
CHAIN_LEVEL_DESC	CHAIN	The description to use for any top level hierarchy element when one must be manually created.	This description must be adjusted if a different description for the top level of the hierarchy is desired.
DEFAULT_LOCALE	en_US	The default locale to use for rendering elements that cannot support multiple locales.	Adjust this value so that it contains the correct LOCALE setting for most users of the application.

Table 15–2 (Cont.) Generic Configurations

Parameter Name	Example Value	Description	Notes
DISPLAY_DATE_FORMAT	Mon dd, yyyy	The default date mask to be used by UI.	Adjust this so that the format of dates can be displayed as desired.
FAKE_CUST_DAY_TXN_ THRESHOLD	10	The maximum daily transaction threshold to identify a fake customer.	This setting must be adjusted to the maximum number of daily transactions that a normal customer usually has. This allows all other customers with higher transactions counts to be excluded from processing.
UI_TZ	America/New_York	Timezone for display. Must match SELECT tzname FROM V$TIMEZONE_NAMES.	This setting must be adjusted so that it contains a proper time zone setting indicating where most users run the UI.

Advanced Clustering Configurations

The following is a list of configurations that can be adjusted for the Advanced Clustering application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_ CODE='CIS'.

Store Exclusion can be used to filter and identify which stores are to be excluded for Clustering. To enable this functionality, EXCLUDE_LOC_ATTR is set by specifying a SHORT_DB_NAME from the RSE_BUSINESS_OBJECT_ATTR_MD BOAM) table in RSE_ CONFIG. This can be configured via the Control and Tactical Center. Locations in RSE_LOC_ ATTR table with the BUSINESS_OBJECT_ATTR_MD_ID value similar to the ID of from BOAM table, which is configured in the config table, will be excluded.

For example, EXLUDE_LOC_ATTR of RSE_CONFIG is set to CFA_ATTRIB_2345 as its value. A check is performed to fetch the ID from the BOAM table that contains this SHORT_ DB_NAME. This ID is then used to obtain all locations from the RSE_LOC_ATTR table. These locations are then flagged to be excluded from clustering.

Table 15–3 AC Configurations

Parameter Name	Example Value	Description	Notes
PERF_CIS_APPROACH	CDT	The approach to use for performance-based clustering. Available options are CDT and DT.	If the CDT application is in use, then this configuration must be set to CDT, which will result in the use of attribute groups for product attribute clustering. Otherwise, it must be set to DT, which will result in the use of the top raw attribute values. This cannot be adjusted after initial setup.
ATTR_NAME_SEPARATOR	-	The separator character(s) to use to separate the different components of the attribute names in CIS_TCRITERIA_ATTR.	Adjust this value if a different separator is desired in the UI for attribute names built from multiple values. This cannot be adjusted after initial setup.
CIS_CONTR_SLS_SRC_COLUMN	SLS_AMT	Source column for contribution BI chart sales values. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_CONTR_X_SRC_COLUMN	SLS_AMT	Source column for the x axis of the contribution BI chart. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_CONTR_Y_SRC_COLUMN	SLS_QTY	Source column for the y axis of the contribution BI chart. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_DFT_PIVOT_LVL	6	Default pivot level to show in explore data.	This configuration indicates the lowest level of the organization hierarchy (see rse_hier_level) that should be shown in the Explore Data pivot table. A value of 6 allows store locations to be visible, but this can be adjusted to a higher level if this level of detail is not desired.
CIS_IDX_AVG_SRC_COLUMN	SLS_AMT	Name of the column to use for index-to-average BI calculations. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows index to averages based on the desired sales column.
CIS_NUMERIC_DFT	0	Default attribute value for numeric.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.

Table 15–3 (Cont.) AC Configurations

Parameter Name	Example Value	Description	Notes
CIS_STRING_DFT	UNKNOWN	Default attribute value for string.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.
CIS_VARIABILITY_IDX_SRC_COLUMN	SLS_AMT	Name of the column to use for variability index BI calculations. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows variability indexes based on the desired sales column.
DEFAULT_NUM_ATTR_VALUE	15	Constant for number of discrete values allowed for store attribute clustering.	Any attribute that has more than this number of distinct categorical values is not available for use as an attribute.
DEFAULT_STR_CATEGORICAL_ATTR	UNKNOWN	Default string description for row added in cis_tcriteria_attr_type_value table for unmatched grouping.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.
EXCLUDE_LOC_ATTR	CFA_ATTR5_2345	Contains the name of a SHORT_DB_NAME from RSE_BUSINESS_OBJECT_ATTR_MD table that is used to exclude specific stores that are in RSE_LOC_ATTR to be excluded from AC processing.	Setting this configuration to NULL disables the Store Exclusion interface.
INSIGHT_BI_SALES_AMT_MARGIN	N	Flag to show sales BI based on sales amount and margin.	Either INSIGHT_BI_SALES_AMT_MARGIN or INSIGHT_BI_SALES_AMT_UNIT can be assigned a Y so that the BI displays the BI using the desired columns.
INSIGHT_BI_SALES_AMT_UNIT	Y	Flag to show sales BI based on sales amount and units	-
MAX_HIST_WEEK_CNT	104	The maximum number of weeks that must be selectable by the UI when processing historic data.	-
MAX_ITEMS_IN_GRAPH_CLUSTER_DETAIL	-1	Maximum number of clusters to be displayed in cluster details graph.	A value of -1 results in an unlimited number of values. Adjust if necessary.
MAX_ITEMS_IN_GRAPH_CLUSTER_LIST	-1	Maximum number of clusters to be displayed in cluster list graph.	A value of -1 results in an unlimited number of values. Adjust if necessary.
MNG_RUN_NO_WKS	26	Display run for past n weeks.	-
PERF_ATTR_TOPN_COUNT	3	The number of attribute values to be used per product category for performance-based clustering.	If PERF_CIS_APPROACH configuration is DT, then this configuration will limit the number of attribute values to this number of values with the greatest sales.

Table 15–3 (Cont.) AC Configurations

Parameter Name	Example Value	Description	Notes
PERF_NUM_WEEKS_FOR_SLS_SHARE	16	The number of weeks to be used while calculating the sales share for the product attributes.	-
PERF_NUM_WEEKS_FOR_TOPN_CALC	16	The number of weeks to be used while identifying the top *n* attributes	-
SELECT_ALL_MERCH_NODES	N	Flag to identify if all (or only first) merchandise node(s) to be selected by default.	-
SUMM_CAL_ALL_LVL	N	Flag to identify whether performance summarization allowed at all available calendar levels.	-
SUMM_MERCH_ALL_LVL	N	Flag to identify whether performance summarization allowed at all available merchandise levels.	-

Assortment and Space Optimization Configurations

The following is a list of configurations that can be adjusted for the ASO application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='SO'. Note that the WebLogic server must be rebooted whenever a configuration change is made for it to take effect. It is recommended that you make all configuration changes at the same time so that you only need to reboot once.

Table 15–4 ASO Configurations

Parameter Name	Example Value	Description	Notes
SO_LOC_HIER_TYPE	2	The hierarchy ID to use for location (installation configuration).	To use an alternate location hierarchy for ASO, adjust this configuration accordingly.
SO_PROD_HIER_TYPE	1	The hierarchy ID to use for the product (Installation configuration).	To use an alternate product hierarchy for ASO, adjust this configuration accordingly.
ALRT_LESS_THAN_PCT_USED_SPACE	0.8	An alert will be triggered if the run optimization results use less space than the value specified by this global parameter.	-
ALRT_LESS_THAN_SERVICE_LEVEL_AMT	0.80	An alert will be triggered if the run optimization results have a sales service level lower than the value specified by this global parameter.	-
ALRT_LESS_THAN_SERVICE_LEVEL_QTY	0.80	An alert will be triggered if the run optimization results have a quantity service level lower than the value specified by this global parameter.	-

Table 15–4 (Cont.) ASO Configurations

Parameter Name	Example Value	Description	Notes
ALRT_MORE_THAN_CNT_ PRODUCT_DROPPED	10	An alert will be triggered if the run optimization results dropped more products than the value specified by this global parameter.	-
ALRT_MORE_THAN_PCT_ PRODUCT_DROPPED	0.2	An alert will be triggered if the run optimization results dropped a percentage of product higher than the value specified by this global parameter.	-
ALRT_NO_FEASIBLE_ SOLUTION	0	An alert will be triggered if the run optimization results have no results.	-
ALWAYS_REVIEW_MAPPING_ RES_FLG	N	Default value is N. A Y flag indicates that a user mapping review is always required (regardless of results or errors). A N flag triggers a review based on other flags and conditions.	-
CAPACITY_RANGE_UNITS	25	Capacity range units used by SO Solver. This parameter value maps to a CRU row with this value ID within so_prod_ constr_range_values table.	-
DEFAULT_BAY_MERGE_ CONSTR_FLG	N	Default indicator for the use of merging bays constraint.	-
DEFAULT_BLOCKING_ CONSTR_FLG	Y	Default indicator for the use of blocking constraint.	-
DEFAULT_SPACING_ CONSTR_FLG	Y	Default indicator for use of spacing constraint.	-
DEFAULT_USABLE_SPACE_ CONSTR_FLG	N	Default flag indicating if space constraint must be used.	-
DEFAULT_USABLE_SPACE_ CONSTR_PCT	1	Default usable space constraint percentage.	-
DEMAND_DISTRIBUTION	Normal	Demand distribution used by SO Solver.	-
DFLT_HORIZONTAL_ BLOCKING_FLG	N	A Y value in this flag indicates the analytics that combining adjacent attribute blocks should be done (when possible).	-
DFLT_REPL_CASEPACK	1	Default replenishment parameter for casepack.	-
DFLT_REPL_FACINGS_LIFT	0	Default facing lift.	-
DFLT_REPL_SHELF_PARAM	0	Default shelf replenishment parameter.	-
DFLT_REPL_SHELF_TT	2	Default replenishment type.	-
DFLT_REPL_TYPE	2	Default replenishment type.	-

Table 15–4 (Cont.) ASO Configurations

Parameter Name	Example Value	Description	Notes
DFLT_SHELF_THICKNESS	1	This is the default shelf thickness that is used by the POG-shelf interface to create the initial-bottom shelf for empty shelf fixtures.	-
GV_DAYS_TO_VALIDATE_WO_CHANGES	21	Number of days without direct changes the validation process considers data objects for validation.	-
GV_RESULT_DETAIL_LEVEL	SUMMARY	Level of detail for each validation that is used to produce the results (DETAIL:rows for every failure or SUMMARY: a row at the data object level).	-
GV_VALIDATION_SECTIONS_TO_RUN	ASSORTMENT_POG_MAPPING_DS	Global validations are executed for the selected data objects. ASSORTMENT, POG, MAPPING and DS (Display Style).	-
MAX_CAPACITY_RANGE	80	Maximum capacity range used by SO Solver.	-
MAX_HEIGHT_RANGE	72	Maximum height range used by SO Solver.	-
MAX_NUMBER_OF_FACINGS	5	Maximum number of facings used by SO Solver.	-
MAX_NUM_OPT_LOC_BLOCK	10	Maximum number of blocks per optimization location.	-
MAX_SHELF_THICKNESS	2.5	This is the maximum shelf thickness that can be used for shelf fixture edits.	-
MAX_STACK_DFLT_LIMIT	5	The default maximum stack limit that is used for display styles that have a missing value.	-
MIN_CAPACITY_RANGE	0	Minimum capacity range used by SO Solver.	-
MIN_HEIGHT_RANGE	0	Minimum height range used by SO Solver.	-
MIN_NUMBER_OF_FACINGS	1	Minimum number of facings used by SO Solver.	-
MIN_SHELF_DEPTH	2	This is the minimum shelf depth that can be used for shelf fixture edits. The maximum shelf depth is defined by the fixture depth.	-
MIN_SHELF_THICKNESS	0.5	This is the minimum shelf thickness that can be used for shelf fixture edits.	-

Table 15–4 (Cont.) ASO Configurations

Parameter Name	Example Value	Description	Notes
MIN_SHELF_VERTICAL_GAP	2.5	This is the specific smallest allowable vertical offset (SAVO) value. Ensure that any edit action leaves at least this much space between shelves.	-
MNG_ASSORT_NO_WKS	52	Display assortments for past *n* weeks.	-
MNG_RUN_NO_WKS	52	Display run for past *n* weeks.	-
NUMBER_OF_SIMULATED_DAYS	1000	Number of simulated days used by SO Solver.	-
OPT_LOC_LVL1_NAME_STR	All Locations	This value is used entirely or as a prefix to generate the pogset location and optimization location top-level names.	-
OPT_LOC_LVL2_NAME_STR	PC_	This value is used as a prefix to generate the pogset location and optimization location mid-level names.	-
OPT_LOC_LVL3_NAME_STR	SC_	This value is used as a prefix to generate the pogset location and optimization location bottom-level names.	-
PC_SUM_CAPRANGE	Set Capacity Range	Capacity range label for product constraint summary.	-
PC_SUM_ELEVATION	Elevation	Elevation label for product constraint summary.	-
PC_SUM_ELEVRANGE	Set Elevation Range	Elevation range label for product constraint summary.	-
PC_SUM_FACERANGE	Set Facing Range	Facing range label for product constraint summary.	-
PC_SUM_FACINGS	Facings	Facings label for product constraint summary.	-
PC_SUM_INCLUSION	Inclusion	Inclusion label for Product constraint summary	-
PENALTY_PFG_MAX	10	Maximum product family group penalty.	-
PENALTY_PFG_NORM	0.15	Normalized value that affects how close products of the same family are placed together.	-
POGC_SUM_MERGEBAYS	Merge Adjacent Bays	Label to display in the UI for POG Constraint - Merge Adjacent Bays.	-
POGC_SUM_PRODSPACE	Use Product Spacing	Label to display in the UI for POG Constraint - Use Product Spacing.	-
POGC_SUM_SERVICELEVEL	Set Minimum Service Level	Label to display in the UI for POG Constraint - Minimum Service Level.	-

Table 15–4 (Cont.) ASO Configurations

Parameter Name	Example Value	Description	Notes
POGC_SUM_USABLESPACE	Set Usable Space	Label to display in the UI for POG Constraint - Set Usable Space.	-
POG_SET_LVL1_NAME_STR	All Planograms	This value is used to generate the name for the top level node on planogram list.	-
PRODUCT_INCLUSION	2	Product Inclusion rule used by SO Solver. This parameter value maps to a IN row with this value ID within so_prod_ constr_range_values table.	-
PRODUCT_STACKING_ HEIGHT_LIMIT	24	Product stacking height limit that applies as a global setting to all top products.	-
PROD_ATTR_NAME_ DELIMITER	-	This value is used as a delimiter between the product name/descr and the attribute name/descr when setting up POG attributes. A NULL value here results in no concatenations.	-
REVIEW_DSF_ERROR_FLG	Y	A Y flag indicates a user review is required for DSF errors. N lets the process move forward to the next stage using the DSF available.	-
REVIEW_UNMAPPED_PROD_ FLG	Y	A Y flag indicate a user review is required for unmapped products. N lets the process move forward to next stage eliminating unmapped products. This is not desired for products.	-
REVIEW_UNMAPPED_ STORE_FLG	Y	A Y flag indicates a user review is required for unmapped stores. N lets the process move forward to the next stage eliminating unmapped stores.	-
SO_MIN_SERVICE_LEVEL	0.8	Minimum target service level for SO optimization process.	-
SO_PROD_HIER_LEVEL_FOR_ LEAF_NODE	7	Product hierarchy level number for leaf node.	-
STD_ADJUSTMENT_ COEFFICIENT_1	0.05	Analytical parameter. Demand standard deviation adjustment parameter 1.	-
STD_ADJUSTMENT_ COEFFICIENT_2	0.19	Analytical parameter. Demand standard deviation adjustment parameter 2.	-
TOP_SHELF_STACKING_ HEIGHT_LIMIT	18	Top shelf stacking height limit that applies as a global setting to all top shelves.	-

Table 15–4 (Cont.) ASO Configurations

Parameter Name	Example Value	Description	Notes
UI_CONFIG_PC_RENDERED_COL_7	N	UI configuration for product constraints render column 7. Default Y means column will be rendered.	-
UI_CONFIG_PC_RENDERED_COL_8	N	UI configuration for product constraints render column 8. Default Y means column will be rendered.	-
UI_CONFIG_PC_RENDERED_COL_9	N	UI configuration for product constraints render column 9. Default Y means column will be rendered.	-
UI_CONFIG_PC_VISIBLE_COL_1	N	UI configuration for product constraints visible column 1. Default Y means column will be visible.	-
UI_CONFIG_PC_VISIBLE_COL_2	N	UI configuration for product constraints visible column 2. Default Y means column will be visible.	-
UI_CONFIG_PC_VISIBLE_COL_3	N	UI configuration for product constraints visible column 3. Default Y means column will be visible.	-
UI_CONFIG_PC_VISIBLE_COL_4	Y	UI configuration for product constraints visible column 4. Default Y means column will be visible.	-
UI_MAX_POG_CONFIG_LENGTH	600	UI configuration for maximum length bound for the Create Lengths pop-up.	-
UI_MAX_POG_CONFIG_NO_OF_BAYS	10	UI configuration for maximum number of bays bound for the Create Lengths pop-up.	-
UI_MIN_POG_CONFIG_NO_OF_BAYS	1	UI configuration for minimum number of bays bound for the Create Lengths pop-up.	-
UI_THRESHOLD_SL	Y	UI configuration for Thresholds Configurable for Service Level Formatting.	-
UI_THRESHOLD_SL_MAX	0.85	UI configuration for Thresholds MAX after which color green would be shown.	-
UI_THRESHOLD_SL_MIN	0.75	UI configuration for Thresholds MIN below which color red would be shown.	-
USE_OPT_DT	N	SO global indicator for applying DT.	If the DT application is not in use, then this configuration should be set to an N to disable this feature.
USE_SERVICE_LEVEL_CONSTRAINT	Y	SO global indicator for applying service level constraints.	-

Customer Decision Tree Configurations

The following is a list of configurations that can be adjusted the CDT application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='CDT'. See Chapter 3 for more details.

Table 15–5 CDT Configurations

Parameter Name	Example Value	Description	Notes
CDT_CAL_LEVEL_ID	4	The hierarchy level ID that contains the level of the calendar hierarchy that CDT operates on (should equate to Week - Installation configuration).	Normally, this configuration must be the same value as the common CAL_PERIOD_ LEVEL configuration.
CDT_CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy that CDTs are created for (installation configuration).	Normally, this must be the same setting as the common CMGRP_LEVEL_ID configuration.
CDT_LOC_HIER_TYPE	2	The hierarchy ID to use for location (installation configuration).	This setting must be set to use either the trade area hierarchy type or the organization hierarchy type.
CDT_LOC_LEVEL_ID	4	The hierarchy level ID that contains the level of the location hierarchy that CDTs are created for (installation configuration).	If CDT_LOC_HIER_TYPE is set to trade area, this must be set to a value of 2. Otherwise, it must be set to the level of the organization hierarchy for which CDT output is desired.
CDT_PROD_HIER_TYPE	1	The hierarchy ID to use for the CM Group (installation configuration).	Normally, this setting must be the same value as the common CMGRP_HIER_TYPE configuration. It can either be the primary product hierarchy or an alternate product hierarchy.
CDT_CALC_RAW_ATTR_SIM	Y	Determines whether or not to execute Raw Attribute Value Similarities routines.	If the RI application is in use, then this setting must be set to a Y to enable calculation of data indented to be used by RI.
CDT_UI_DEF_CALC_PARENT_ SEGMENT_FLG	Y	UI default for the calculate only parent consumer segments flag.	-
CDT_UI_DEF_CALC_PARENT_ TRADE_AREA_FLG	N	UI default for calculate only parent trade areas flag.	-
CDT_UI_DEF_CDT_SCORE_ HIST_CNT	20	UI default for the number of histogram buckets for the CDT scores histogram.	-
CDT_UI_DEF_DATA_FILTER_ HIST_CNT	20	UI default for the number of histogram buckets for the data filtering histograms.	-
CDT_UI_DEF_EXCLUDE_ CUST_CNT	1000	UI default for minimum require customer counts for pruning process.	-

Table 15–5 (Cont.) CDT Configurations

Parameter Name	Example Value	Description	Notes
CDT_UI_DEF_EXCLUDE_MIN_SCORE	0.25	UI default for minimum CDT score required for the pruning process.	-
CDT_UI_DEF_EXCLUDE_SKU_CNT	10	UI default for minimum number of SKUs for the pruning process.	-
CDT_UI_DEF_EXCLUDE_TREE_LEVEL_CNT	2	UI default for minimum number of levels of the tree for the pruning process.	-
CDT_UI_DEF_LOWEST_EXPANSION_LEVEL	15	UI default for lowest number of levels allowed for a tree.	-
CDT_UI_DEF_MAX_CUST_AVG_DY_TXN	100	UI default for maximum number of times more than average a customer's daily transaction count can be.	-
CDT_UI_DEF_MAX_MISS_ATTR_CNT	3	UI default for maximum number of missing attributes a SKU can have	-
CDT_UI_DEF_MIN_ATTR_SKU_CNT	5	UI default for minimum number of SKUs assigned to an attribute, to be used by the process.	-
CDT_UI_DEF_MIN_ATTR_VALUE_SKU_CNT	5	UI default for minimum number of SKUs assigned to an attribute value, to be used by the process.	-
CDT_UI_DEF_MIN_CUST_TXN_CNT	0.01	UI default for minimum number of transactions required for a customer, as a percent of the average number.	-
CDT_UI_DEF_MIN_NODE_ITEM_CNT_PCT	0.05	UI default for the minimum percent of SKUs required for a node of the tree before it is considered a terminal node.	-
CDT_UI_DEF_MIN_SKU_TXN_CNT	0.01	UI default for minimum number of transactions required for a SKU, as a percent of the average number.	-
CDT_UI_DEF_PRUNING_HIST_CNT	20	UI default for the number of histogram buckets for the pruning histograms.	-
CDT_XML_PRECISION	4	Default precision of weight field in CDT XML.	Adjust this to control the amount of precision in the generated CDT XML files.

Table 15–5 (Cont.) CDT Configurations

Parameter Name	Example Value	Description	Notes
HISTOGRAM_DEFAULT_BIN_APPROACH	C	The default histogram bin approach (C = Custom, W = Width)	-
HISTOGRAM_DEFAULT_NUM_BINS	7	The default number of bins to display for CDT histograms	-
MAX_NUM_WEEKS_FOR_SIMILARITY	104	The maximum number of weeks of sales transaction data to be used by the similarity process. This prevents the process from using too much data.	Adjust this to reduce the use of too much input data for the process. Enabling a high number of weeks will result in slower performance; however, it may be necessary if suitable data is not available in a smaller number of weeks.

Demand Transference Configurations

The following is a list of configurations that can be adjusted for the DT application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='DT'.

Table 15–6 DT Configurations

Parameter Name	Example Value	Description	Notes
AE_CALC_INT_LENGTH	8	The number of weeks to group together for in an interval for the AE calculation.	This setting must be adjusted according to the data quality and quantity available for a customer. Higher values improve performance; however, it limits the amount of data available to be processed.
ATTRIBUTE_LIST_SEPARATOR	-	A separator to be used to display a list of attributes in Similarity Calculation screen.	-
CDT_SIMILARITY_AVAILABLE	Y	Whether CDT similarity has been made available to DT.	If the CDT application is being used, this setting enables DT to use the similarities that CDT may have calculated.
DT_CAL_LEVEL_ID	4	The hierarchy level ID that contains the level of the calendar hierarchy that DT operates on (should equate to Week).	Normally, this configuration must be the same value as the common CAL_PERIOD_LEVEL configuration.
DT_CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy that DTs are created for.	Normally, this must be the same setting as the common CMGRP_LEVEL_ID configuration.
DT_LOC_HIER_TYPE	2	The hierarchy ID to use for location.	This must be adjusted to trade area hierarchy or be left at the default organization hierarchy.

Table 15–6 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
DT_LOC_LEVEL_ID	4	The hierarchy level ID that contains the level of the location hierarchy that DTs are created for.	If DT_LOC_HIER_TYPE is set to trade area, this must be set to a value of 2. Otherwise, it must be set to the level of the organization hierarchy for which DT output is desired.
DT_MDL_AP_EXP_WKS_BACK_END	1	The number of weeks back from the last date that range data has been loaded for (PR_LOC_STATUS_LAST_COMPLETED_WK) to end using for model apply export.	Adjust this and MDL_AP_EXP_WKS_BACK_START to control which weeks should be used during data export.
DT_PROD_HIER_TYPE	1	The hierarchy ID to use for the CM Group.	Normally, this setting must be the same value as the common CMGRP_HIER_TYPE configuration. It can either be the primary product hierarchy or an alternate product hierarchy.
DT_REMOVE_REDUNDANCY	N	If set to Y, then remove redundancy while calculating attribute-based similarities.	-
DT_SIM_DISPLAY_ROWNUM	9999999	The number of distinct similarity values to show in the UI pop-up. Setting to a high number effectively eliminates this limit.	-
GENERIC_SEPARATOR		A separator to be used to display a list of items, for example. SKU prod_ext_code name.	This value is used in the UI to separate lists of items. For example, when a list of attributes is shown in the UI, they will be delimited by this value.
HISTOGRAM_DEFAULT_BIN_APPROACH	W	The default histogram bin approach (C = Custom, W = Width).	-
HISTOGRAM_DEFAULT_NUM_BINS	7	The default number of buckets in the contextual BIs.	-
MAX_NUM_WEEKS_FOR_ATTR_WGT	104	The maximum number of weeks of input data to use for calculating attribute weights.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_AVG_SLS	104	The maximum number of weeks of input data to use for calculating the average sales.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.

Table 15–6 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
MAX_NUM_WEEKS_FOR_FILTERING	104	The maximum number of weeks of input data to use for data filtering. Setting this value lower than the other MAX_NUM_WEEKS_FOR* configurations overrides those other configurations.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_MDL_CALC	104	The maximum number of weeks that should be used during model build calculation.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_MDL_UPDT	104	The maximum number of weeks that should be used during model build update calculation.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_SIMILARITY	104	The maximum number of weeks of input data to use for calculating similarity.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MDL_AP_EXP_WKS_BACK_START	4	The number of weeks back from the last date that range data has been loaded for (PR_LOC_STATUS_LAST_COMPLETED_WK) to start using for model apply export.	Adjust this and DT_MDL_AP_EXP_WKS_BACK_END to control which weeks should be used during data export.
PRUNED_CATEGORIES_SEPARATOR	,	A separator to be used to display a list of pruned attributes in the Calculation screen.	-
UI_DEF_CALC_PARENT_CS_ONLY_FLG	N	The UI default for calculate only parent customer segments flag.	-
UI_DEF_CALC_PARENT_TA_ONLY_FLG	N	The UI default for calculate only parent trade areas flag.	-
UI_DEF_MAX_MISS_ATTR_CNT	3	The maximum number of missing attributes for a SKU, before requiring it to be filtered from use.	-
UI_DEF_MIN_SKU_CNT	10	The UI default for minimum number of SKUs required for a segment/store.	-

Table 15–6 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
UI_DEF_MIN_SKU_TXN_LEN_PCT	0.01	The UI default for minimum SKU transaction length as a percentage of the CM Group average.	-
UI_DEF_MIN_TOT_SLS_UNIT_PCT	0.01	The UI default for minimum total sales units as a percentage of the CM group average.	-
WGT_CALC_INTERVAL_LENGTH	4	The number of weeks to group into an interval that is then used to perform weight calculations with.	This setting can be adjusted according to the data quality and quantity available for a customer. Higher values improve performance; however, it limits the amount of data available to be processed.

Returns Logistics Configurations

The following is a list of configurations that can be adjusted for the Returns Logistics process. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='RL'.

Table 15–7 Returns Logistics Configurations

Parameter Name	Example Value	Description	Notes
RL_NUM_TOP_CATEGORIES	10	The number of top selling categories for Returns Logistics to process.	-
RL_NO_OF_SIMULATION_RUNS	100	The number of simulated forecast runs to perform.	-
RL_IN_SEASON_DEMAND_WEIGHT	0.99	The percentage of the in-season demand and returns rate to use. This is used to calculate the adjusted demand and returns rate.	-
RL_HISTORICAL_DEMAND_WEIGHT	.01	The percentage of the historical demand and returns rate to use. This is used to calculate the adjusted demand and returns rate.	-
RL_GUR_TIMELIMIT	60	GUROBI parameter. Time limit.	-
RL_GUR_THREADS	0	GUROBI parameter. Number of allowed threads.	-
RL_PROCESSING_THREADS	0	The number of threads to create for data processing. A value of 0 creates one thread for each machine processor.	-

Affinity Analysis Configurations

The following is a list of configurations that can be adjusted for the AA application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='MBA'. Some of the

configurations note that overrides are possible. When this is required, see the RSE_CONFIG_CODE configuration table and use the appropriate Hierarchy Level value as the PARAM_CODE to override the value for only that hierarchy level.

Table 15–8 AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_CS_HIER_LEVEL	SBC	The highest level of the hierarchy to run the process for. Valid values are: CLS, SBC.	-
ARM_CS_MAX_LIFT	100	The maximum lift required for an association rule.	-
ARM_CS_MAX_RULE_COUNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_CS_MAX_SET_SIZE	2	The maximum number of hierarchy members to include in an association rule.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_CS_MIN_CONFIDENCE	.05	The minimum confidence required for an association rule.	-
ARM_CS_MIN_LIFT	.05	The minimum lift required for an association rule.	-
ARM_CS_MIN_REV_CONFIDENCE	.05	The minimum reverse confidence required for an association rule.	-
ARM_CS_MIN_SUPPORT	.001	The minimum percentage of transactions require for an association rule.	-
ARM_CS_MIN_SUPPORT_TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_CS_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.
ARM_PH_MAX_LIFT	100	The maximum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MAX_RULE_COUNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_PH_MAX_SET_SIZE	2	The maximum number of hierarchy members to include in an association rule. Override with PARAM_CODE of the hierarchy level name.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_PH_MIN_CONFIDENCE	.05	The minimum confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_LIFT	.05	The minimum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-

Table 15–8 (Cont.) AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_PH_MIN_REV_ CONFIDENCE	.05	The minimum reverse confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_SUPPORT	.001	The minimum percentage of transactions require for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_SUPPORT_ TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_PH_TOP_LEVEL	CLS	The highest level of the hierarchy to run the process for. Valid values are: DEPT, CLS, SBC.	-
ARM_PH_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.
ARM_PROMO_HIER_LEVEL	SBC	The highest level of the hierarchy to run the process for. Valid values are: CLS, SBC.	-
ARM_PROMO_MAX_LIFT	100	The maximum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MAX_RULE_ COUNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_PROMO_MAX_SET_ SIZE	2	The maximum number of hierarchy members to include in an association rule. Override with PARAM_CODE of the hierarchy level name.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_PROMO_MIN_ CONFIDENCE	.05	The minimum confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_LIFT	.05	The minimum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_REV_ CONFIDENCE	.05	The minimum reverse confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-

Table 15–8 (Cont.) AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_PROMO_MIN_SUPPORT	.001	The minimum percentage of transactions require for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_ SUPPORT_TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_PROMO_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.

Resource Bundles

Oracle Retail AI Foundation Cloud Services comes packaged with resource bundles, files that contain text resources. These text resources appear throughout the application as instructions, messages, labels, errors. Retailers can customize the text resources to suit their requirement. The capability to change the text for a resource is provided through the Resource Bundles Section of Retail Home. For more information, see *Oracle Retail Home Administration Guide*.

Manage Notebooks

Different types of notifications are sent to specific groups of users for the tasks and jobs accomplished. Retail Home provides administrators with the ability to set up groups of user roles that must receive each type of notification. For more information, see *Oracle Retail Home Administration Guide*.

Internationalization

The user interface supports multiple languages in a single instance, but the underlying database only supports a single language in an instance.

The database default language is selected at installation. Once set, there is no support for switching the database language.

The application user interfaces adhere to the language setting for each user's browser. For example, to change the language for the Firefox browser:

1. Select Tools from the menu bar.
2. Select Options.
3. Select Choose.
4. Select the language to add.

The following language are supported: English, German, Greek, Spanish, French, Croatian, Hungarian, Italian, Japanese, Korean, Dutch, Polish, Brazilian Portuguese, Russian, Swedish, Turkish, Simplified Chinese, and Traditional Chinese.

Configuration Updates

Use the Manage Configuration user interface screen in order to edit the values in the configuration tables. In order to access this screen, you must be assigned the role of

ADMINISTRATOR_JOB. The tables you can edit within Manage Configuration depends upon which application roles you have been assigned, as described in Table 15–1.

Once you have accessed Manage Configuration, you select, from the configuration tables available to you, the table that you want to review and edit. Note that each configuration table allows specific actions. You may or may not be able to view the table, edit some or all of the columns, or add and delete rows.

Certain of the tables in Manage Configuration are not intended for configuration but instead are provided to display data that may be useful during implementation. For each application interface file, a table is available that contains any records that have failed the data validation rules. Such tables have an extension of BAD. For example, for the rse_prod_attr_grp_value_stg.txt interface, the data is loaded into a table named RSE_PROD_ATTR_GRP_VALUE_STG. The error table is named RSE_PROD_ATTR_GRP_VALUE_BAD.

You can see all the error tables for all the interfaces via Manage Configuration. When an error occurs with an interface, yo can review the BAD table to see what records failed validation and why. Each BAD table includes the following columns: ERROR_ROWID, ERROR_ID, ERROR_DESCR, and ERROR_DT. The ERROR_ID is useful as a reference. The ERROR_DESCR provides a message to explain the validation error.

Configuration Tables

The following tables can be viewed in the Manage Configuration screen.

Table 15–9 Customer Decision Tree Configuration Tables

Table Name	Description
CDT_EXCLUDE	Defines the various types of pruning filters that can be used to prevent a CDT from being used during the escalation phase of the CDT workflow.
CDT_FILTER	Defines the various types of data filters that can be used to filter sales transaction data used for the CDT calculation.
CDT_VERSION	Defines a version to collectively group together a batch of CDTs that have been created for a particular purpose.

Table 15–10 Advanced Clustering and Customer Segmentation Configuration Tables

Table Name	Description
CIS_ALGORITHM_ATTR	Defines the possible attributes for any algorithm.
CIS_BUSINESS_OBJECT	Hosts the list of applications that are registered and configured to use the clustering feature.
CIS_BUSSOBJ_OBJ_ALG_XREF	This cross reference is provided so that you can use the same algorithm to generate different cluster objectives. The same algorithm can be used to generate customer clusters as well as store clusters. At the same time it is possible to list multiple algorithms that can be used to achieve a similar objective.
CIS_BUSSOBJ_TCRIT_HIER_XREF	Specifies the possible hierarchy levels for each hierarchy type (merchandise and location) that are permitted for the combination of objective ID, business objective ID, and type criteria ID.
CIS_BUS_OBJ_HIER_DEPLOY_XREF	Identifies the business object, objectives, product hierarch types, and levels that can be selected for deployment. This ensures that only authorized types of clusters are exported to external systems (CMPO).
CIS_BUS_OBJ_NESTED_TCRITERIA	Determines the possible child cluster types for a parent cluster.

Table 15–10 (Cont.) Advanced Clustering and Customer Segmentation Configuration

Table Name	Description
CIS_BUS_OBJ_TCRITERIA_XREF	Specifies the possible cluster types that are permitted for the combination of objective ID and business objective ID.
CIS_BUS_OBJ_TCRIT_ALGO_ATTR	Defines possible attributes for any algorithm, business objective, objective, and criteria.
CIS_CLUSTER_OUTLIER_RULE	Specifies the possible outlier rules for a type of criteria.
CIS_EFFECTIVE_PERIOD	Specifies the planning period information.
CIS_ODM_SETTINGS	Temporary storage for the data mining settings provided to ODM. You can store multiple runs of settings by using different mining functions and run IDs. To provide this data to ODM, you must write a view that provides the appropriate rows of data.
CIS_SALES_TYPE	Specifies the type of sales information - historical, forecasted, or planned.
CIS_SRC_ENTITY	Defines the different database views available for use in the various clustering implementations. These settings help control how the attributes are used throughout the system.
CIS_TCRITERIA_SRC_XREF	Cross reference to SRC_ENTITY_NAME and for the settings for Partitioning, Information for the SRC_ENTITY_NAME attributes.
CIS_TCRIT_SRC_TYPE_XREF	Describes the xref of type_criteria and sales.
CIS_TYPE_CRITERIA	List of different clustering types and criteria that can be used to generate clusters.

Table 15–11 Demand Transference Configuration Tables

Table Name	Description
DT_EXCLUDE	Defines the different types of pruning filters available to prevent a DT result from being used during the escalation phase of the DT workflow.
DT_FILTER	Defines the different types of data filters used during the DT data filtering process.

Table 15–12 Generic Configuration Tables

Table Name	Description
RSE_AGGR_SRVC_CONFIG_LEVELS	Defines the different hierarchy types and levels that must have aggregated data created as part of the hierarchy configuration.
RSE_BUSINESS_OBJECT_ATTR_MD	Defines the attributes for business objects and also provides relevant details about where from external table sources to obtain the data for this attribute.
RSE_BUSINESS_OBJECT_DB_SRC	Defines the source database objects for the attributes.
RSE_CONFIG	Provides configuration names and their values for the settings that can be changed and affect the operation of the software.
RSE_CONFIG_CODE	Provides configuration values for those configurations that can have different values, depending on how other values are configured. For example, if a configuration is required for a default error tolerance, but department 1 requires a different value, then a row here with a PARAM_CODE of 1 will enable a different value than the base configuration in RSE_CONFIG for just that department.

Table 15–12 (Cont.) Generic Configuration Tables

Table Name	Description
RSE_EMAIL_CFG_DISTR	Intersection table between the email configuration table and the email distribution list table used to resolve the many-to-many relationships.
RSE_EMAIL_DISTR_LIST	Provides the email distribution lists.
RSE_EMAIL_NOTIF_CFG	Defines the email messaging configuration.
RSE_EXP_GRP	Hosts the list of applications that are registered and configured to use the clustering application.
RSE_HIER_LEVEL	Defines the various levels for all the hierarchies.
RSE_HIER_TYPE	Defines the hierarchies that are available for use by the RSE applications.
RSE_LOAD_SRVC_CFG	Defines a data loader that can be executed through the data loading framework.
RSE_LOAD_VALDT_ RULES_CFG	Defines the validation rules that a data loader performs, along with some configurable options that impact rows that fail the validation.
RSE_PROC_TASK_TMPL	Defines the templates for processing tasks used by the applications.
RSE_PROC_TMPL	Defines the processing templates for asynchronous or synchronous invocable from Java applications
RSE_SRVC_CONFIG	Defines all the database service routines that can be invoked through the database service framework in RSE.

Table 15–13 Assortment and Space Optimization Configuration Tables

Table Name	Description
SO_BI_ELEMENT	Contains configuration metadata for BI elements.
SO_BI_ELEMENT_CHART	Contains the metadata to configure BI element charts.
SO_INT_ TRANSFORMATION_KEY	Used to help perform interface data transformation. The values in this table are used to align data from external sources with the data expected by ASO. It helps to isolate translation conversion issues.
SO_POG_FIXTCONF_ALG_ PARAM	Stores the list of different algorithm parameters that a user can customize while running the fixture smart start process.
SO_POG_FIXT_CONFIG_ ALGORITHM	Stores the list of available algorithms to perform the shelf fixture smart start process to create shelves for empty fixtures.
SO_PROD_CONSTR_ RANGE_VALUES	Contains the list of product constraint values supported by the application.
SO_REPL_PARAM_ DESCRIPTION	Stores the list of replenishment parameters that a user can change. These parameters have a defined list of valid values that are stored in this table so that they can be used by the UI for display.
SO_RUN_OBJECTIVE_ FUNC	Contains the list of possible run objective functions that are supported by the application.
SO_STACK_CAP_STYLE	-

Email Notification Configuration

The RSE_EMAIL_CFG_DISTR, RSE_EMAIL_DISTR_LIST, and RSE_EMAIL_NOTIF_ CFG tables listed in the tables in Configuration Tables can be used to enable email notifications for certain batch processes. The RSE_EMAIL_NOTIF_CFG table contains a list of processes that can generate an email notification. Configurations are available for loading interface files

(PROCESS_TYPE=LOAD), as well as configurations for running the automated batch processes (PROCESS_TYPE=BATCH).

The RSE_EMAIL_DISTR_LIST table is used to define email distribution lists that should receive email notifications. If all email notifications are to be sent to the same email address, then a row with a DEFAULT_FLG=Y can be used so that the distribution list is used for all email notices.

If some processes require different email recipients, then additional distribution lists can be created for those in RSE_EMAIL_DISTR_LIST. Once created there, a record can be created in RSE_EMAIL_CFG_DISTR that links the distribution list with the email notification. Any email configuration without a specific entry in RSE_EMAIL_CFG_DISTR is sent to the default distribution list.

By default, all LOAD failures trigger an email notification with details about the load that failed. Additionally, all batch processes send an email notification when the entire batch process completes. This email indicates whether the batch process was successful or if a failure occurred.

16

Attribute Processing

This chapter addresses attribute preprocessing. It contains the following sections:

- Attribute Preprocessing
- Product Attribute Loading

Attribute Preprocessing

Attributes provide context about products and enhance the accuracy of DT and CDT models. Attributes are stored within RI and are derived from product descriptions and merchandise hierarchy.

RADM may or may not contain product attributes. Any attributes found in RADM may have been created for BI reporting or other purposes and may need mining or preprocessing to make them suitable for the application.

Some steps in attribute preprocessing require manipulating attribute data. Oracle Enterprise Product Data Quality is a licensed software package that facilitates some of the preprocessing data manipulation steps required to make attributes suitable for CDT and DT modeling.

Here is an example of product information for yogurt.

- Product description: Brand A non-fat organic 6 oz.
- Class description: Dairy product.
- Sub-class description: Yogurt.

SKU/Store attributes determined by preprocessing:

- Brand
- Price
- Size

Note that CDT modeling works optimally when there are five or fewer possible values for any given SKU-store attribute. For example, many price points are available for yogurt. For CDT, it is better to define between three and five price bins (that is, budget, regular, premium, and elite).

For ASO, the application itself does not have any specific requirements; the business requirements for the attribute values are what matters. ASO supports the use of attribute value groups to control the layout of products. If the business requirement states that products should be organized by many different attribute values, then, for ASO, the attribute value groups must have as many values as needed for the organization specified. Note that care must be taken, as these two requirements can contradict one another.

Process Overview

The basic steps for attribute preprocessing are as follows:

- Populating RADM with attribute data
- Translating (optional)
- Parsing
- Cleansing and standardizing
- Categorizing and labeling
- Defining attributes
- Binning and grouping

Populating RADM with Attribute Data

A few steps are required to make RADM attributes suitable for the application so that the applications can use this data.

The first requirement is to ensure that the attribute values are populated in RADM. This is the source for the attribute data and must be loaded there in order to be available to the application.

Regarding RADM attributes: In RADM, an attribute can be defined in multiple ways. Flex attributes are typically stored in a column of the W_PRODUCT_ATTR_D table. RADM has a table W_RTL_METADATA_G that contains a list of defined attribute locations. Consult this list to see if there is already a defined place to store a particular attribute value.

RI also offers the ability to store Item Differentiators for products. These are essentially User Defined Attributes (UDAs), which consist of a lookup code for the attribute and the attribute value. These lookup codes are then defined in RADM's standard translation table (W_DOMAIN_MEMBER_LKP_TL with domain codes of ITEM_UDA_HEAD and ITEM_UDA). The actual association of an item to one of the UDAs is performed in the W_RTL_ITEM_GRP 1_D table.

Once attributes are available in RADM, it is necessary to define these attributes in the RSE_BUSINESS_OBJECT_ATTR_MD table. This requires engagement with OCI in order to configure the data correctly for retrieval from the application. This table must be set up with appropriate metadata to define the source of the attributes from RADM. The sample seed_data file for this table contains some standard attributes that can be defined in RADM, but the table must be adjusted to contain the complete list of attributes that should be available for the applications to use. This must include Flex Attributes as well as User Defined Attributes.

Once attributes are defined in the RSE_BUSINESS_OBJECT_ATTR_MD, the next step is to provide custom lists of attributes that should be used per product category. This can be done through the RSE_PROD_ATTR_GRP_VALUE_STG and RSE_PROD_ATTR_VALUE_XREF_STG interfaces. The first interface is used to define the output of the binning and grouping of attributes. For example, if Coffee needs a Brand Tier attribute, and it should have values of Premium, Value, and Mainstream, then this interface would define this Coffee Brand Tier attribute, along with the values of Premium, Value, and Mainstream, and it should specify what source attribute is to be used for this (the source is in RSE_BUSINESS_OBJECT_ATTR_MD). The second table of the interface (RSE_PROD_ATTR_VALUE_XREF_STG), enables the association of specific Brand attributes to the binned/grouped attribute values from the first interface (RSE_PROD_ATTR_GRP_VALUE_STG).

One concept to consider for these attributes and attribute values, is that they must be unique across all product categories. This offers the ability to classify one Brand as Premium for one product category, while it could be Mainstream for another product category. Additionally, it enables a different selection of attribute values for each product category. For example, another

product category might not have a Premium Brand Tier, and therefore the interface would not include this value in this interface for that product category.

Translating

This step is only needed when the product data is in a different language than the customer's primary language.

Parsing

This step identifies and extracts target key words, such as "Brand A," "small," "blue," and "non-fat." from the source data (such as product description). It is done through semantic recognition, usually by software such as Oracle Enterprise Product Data Quality.

Cleansing and Standardization

This step edits the text and corrects spelling and grammar. For example, "Addr." will be recognized and converted into "Address" and "St." into "Street." EPDQ can facilitate this step.

Categorizing and Labeling

This step classifies targeted key words into the pre-defined categories, such as "Brand A" for "Brand," "small" for "Size" and "blue" for "Color." The product record can thus be labeled by the category values. EPDQ can facilitate this step.

Defining Attributes

With the extracted categories from the product data, attributes are defined. They can be some or all of the categories identified, based on contextual business knowledge and how populated the categories are.

Binning and Grouping

Binning and grouping are used to consolidate and reduce the number of possible values for an attribute into a manageable number.

- Binning divides numerical attributes, such as price, discounts, and mileage, into discrete sets of ranges, such as <=$10, $10~$25, and >$25.
- Grouping combines textual attributes that are too granular into a smaller set of attribute values. For example, tea weight can have dozens of values; grouping merges the values into coarser ranges (like small or large) and reduces the number of possible attribute values.

Product Attribute Loading

This section provides an example of adding an attribute for use by the application into all the relevant tables. In this example, a new attribute is added to represent Flavor within the Coffee product category.

The process flow for this involves:

1. Identify the need to add a new product attribute for a product category.
2. Determine where the attribute data is found within RADM.
3. Coordinate with OCI to add the attribute definition in the tables, if it not already present.
4. Coordinate with OCI to run the batch process to load attribute data from RADM.
5. Determine if the attribute data requires any special grouping or binning.

6. Populate the RSE_PROD_ATTR_GRP_VALUE_STG staging table with attribute definition and values.
7. Populate the RSE_PROD_ATTR_VALUE_XREF_STG staging table with data to associate raw RADM attribute values to the Attribute Groups defined above.
8. Coordinate with OCI to run the batch process that processes the interface staging tables.
9. Coordinate with OCI to update the CIS attribute data to reflect the new attribute (product attributes).
10. Coordinate with OCI to update the CIS attribute data to reflect the new attributes (non-product attributes).

Introduce New Attribute

The first step in the process is the catalyst that triggers the remaining steps. The catalyst is the new attribute that has been introduced and must be made available within the application.

Determine the Attribute Source and Define in the Tables

The new attribute is loaded from RADM for each of the products that require this attribute. RADM has multiple ways of loading attributes, so the approach used varies, depending on where and how the data is stored in RADM. The process involves defining the source table and then defining the column (or column filter values) used to identify the attribute. Once the source is determined, the appropriate values are loaded into RSE_BUSINESS_OBJECT_ATTR_MD and possibly RSE_BUSINESS_OBJECT_DB_SRC.

W_PRODUCT_D or W_PRODUCT_ATTR_D

RADM's W_PRODUCT_D table and W_PRODUCT_ATTR_D table can provide attributes from any of the available columns in these tables. The W_PRODUCT_D table contains named columns with data of a specific logical value, while the W_PRODUCT_ATTR_D table contains a more flexible set of Number, Text, and Date columns that can contain varying values, depending on the implementation. From an attribute point of view, these tables are effectively the same and require the same type of handling.

W_RTL_ITEM_GRP1_D or W_RTL_ITEM_GRP2_D

The W_RTL_ITEM_GRP1_D and W_RTL_ITEM_GRP2_D tables in RADM are different than the other product attribute sources, in that these tables can have attributes implemented as unique rows and specific columns. These tables contain a PROD_GRP_TYPE column, which defines the type of data in the table. Values of ITEMUDA are used for User Defined Attributes. Rows in which the PROD_GRP_TYPE corresponds to the BRAND, COLOR, FLAVOR, SCENT, FABRIC, and STYLE WID columns (ex. BRAND_WID) are also possible.

Populate RSE_PROD_ATTR_GRP_VALUE_STG Interface

Once the attribute data has been reviewed and groups have been defined, it is necessary to define the attribute groups and process them into the database. The output of the prior step must be loaded into the staging table for Attribute Value Groups (RSE_PROD_ATTR_GRP_VALUE_STG). This interface defines two sets of data and is used to load two different tables.

Table 16–1 RSE_PROD_ATTR_GRP_VALUE_STG

Column	Example	Description
PROD_HIER_TYPE_NAME	Product Hierarchy	Must match the NAME from RSE_HIER_TYPE that has the ID equal to the RSE_CONFIG for CMGRP_ HIER_TYPE.
PROD_EXT_KEY	CLS~1000~10000	The external key used to identify the product category (for example, Coffee Class). This value is the same as in RADM's INTEGRATION_ID of the W_PROD_ CAT_DH, and also the PROD_EXT_KEY of the RSE_PROD_SRC_XREF table.
ATTR_SHORT_DB_NAME	FLAVOR	This must match the SHORT_DB_NAME of the RSE_BUSINESS_OBJECT_ATTR_MD table for the newly added attribute.
PROD_ATTR_GRP_EXT_KEY	CLS~1000~10000~flavor_yn CLS~1000~10000~flavor_type	This must be a unique value to describe the attribute to be used by the applications. Since the source Flavor attribute is being defined as two different attributes, two example values are shown here.
PROD_ATTR_GRP_NAME	FlavorYN FlavorType	A name to be displayed in the UI for the new attribute. Two example values are shown here.
PROD_ATTR_GRP_DESCR	Flavor Y/N Indentifier Flavor Type	An optional/additional descriptive value that can be displayed in the UI for the new attribute.
PROD_ATTR_VALUE_KEY	(See additional table below)	A unique/external identifier for the new attribute values.
PROD_ATTR_VALUE_NAME	(See additional table below)	A name displayed in the UI for the attribute value.
PROD_ATTR_VALUE_DESCR	(See additional table below)	An optional/additional descriptive value that could be shown in the UI for the new attribute value.
FUNC_ATTR_FLG	N	This is a Y/N flag to indicate whether this attribute is considered to be an attribute associated with a specific function or role (Y) or not (N). For example, a customer cannot choose a product with a different value for the auto wiper blade size because each car model has a specific size requirements.

Here is a table showing the different values for adding the example Flavor Attribute Values.

Table 16–2 Flavor Attribute Values

PROD_ATTR_ GRP_NAME	PROD_ATTR_VALUE_KEY	PROD_ATTR_VALUE_ NAME	PROD_ATTR_VALUE_ DESCR
FlavorYN	CLS~1000~10000~flavor_yn~y	Y	Yes
FlavorYN	CLS~1000~10000~flavor_yn~n	N	No
FlavorType	CLS~1000~10000~flavor_type~non	Non Flavored	Non Flavored
FlavorType	CLS~1000~10000~flavor_type~fruit	Fruit Flavored	Fruit Flavored
FlavorType	CLS~1000~10000~flavor_type~mild	Mild Flavored	Mild Flavored
FlavorType	CLS~1000~10000~flavor_type~special	Specialty	Specialty

Populate RSE_PROD_ATTR_VALUE_XREF_STG Interface

Once the RSE_PROD_ATTR_GRP_VALUE_STG interface has been loaded, it is possible to load the RSE_PROD_ATTR_VALUE_XREF_STG interface with a mapping of actual product attribute values (otherwise known as base attributes) to the attribute groups that were loaded via

RSE_PROD_ATTR_GRP_VALUE_STG. The format of data to be loaded here depends on the format of the base attributes. Only one set of attribute value columns should be populated for this interface. These sets are MIN_ATTR_NUM_VALUE and MAX_ATTR_NUM_VALUE (for numeric attributes), ATTR_STRING_VALUE (for text attributes), MIN_ATTR_DATE_VALUE and MAX_ATTR_DATE_VALUE (for date attributes), ATTR_VALUE_EXT_CODE (for dimension-based attributes). The sets are mutually exclusive of each other for this interface.

Table 16–3 RSE_PROD_ATTR_VALUE_XREF_STG

Column	Example	Description
PROD_ATTR_VALUE_KEY	CLS~1000~10000~flavor_yn~y	Must match a PROD_ATTR_VALUE_KEY that was loaded via the RSE_PROD_ATTR_GRP_VALUE_STG interface.
MIN_ATTR_NUM_VALUE	0	Minimum numeric value to associate with this attribute group value. Only applicable if this attribute uses the ATTR_NUM_VALUE column to store the base attribute value.
MAX_ATTR_NUM_VALUE	7	The maximum numeric value to associate with this range. Only applicable in conjunction with MIN_ATTR_NUM_VALUE.
ATTR_STRING_VALUE	Y	A string value to associate with this attribute group value. Only applicable if this attribute uses the ATTR_STRING_VALUE column to store the base attribute value.
MIN_ATTR_DATE_VALUE	2010-01-01	The minimum date value to associate with this attribute group value. Default date format for provided control file is YYYY-MM-DD. Only applicable if this attribute uses the ATTR_DATE_VALUE column to store the base attribute value.
MAX_ATTR_DATE_VALUE	2010-01-31	The maximum date value to associate with this attribute group value. Default date format for provided control file is YYYY-MM-DD. Only applicable in conjunction with MIN_ATTR_DATE_VALUE.
ATTR_VALUE_EXT_CODE	32	For base attributes that are sourced from W_RTL_ITEM_GRP1_D, this column can be used to specify the key from the appropriate source column. This is applicable if this attribute uses ATTR_VALUE_EXT_CODE to store the attribute value.

Here is a table containing some examples for adding a new flavor attribute, using string-based attributes.

Table 16–4 Adding a New Flavor Attribute

PROD_ATTR_VALUE_KEY	ATTR_STRING_VALUE
CLS~1000~10000~flavor_yn~y	BLUEBERRY
CLS~1000~10000~flavor_yn~y	RASPBERRY
CLS~1000~10000~flavor_yn~y	VANILLA
S~1000~10000~flavor_yn~y	CARAMEL
CLS~1000~10000~flavor_yn~y	CINNAMON
CLS~1000~10000~flavor_yn~y	HAZELNUT
CLS~1000~10000~flavor_yn~n	PLAIN

Table 16–4 (Cont.) Adding a New Flavor Attribute

PROD_ATTR_VALUE_KEY	ATTR_STRING_VALUE
CLS~1000~10000~flavor_type~non	PLAIN
CLS~1000~10000~flavor_type~fruit	BLUEBERRY
CLS~1000~10000~flavor_type~fruit	RASPBERRY
CLS~1000~10000~flavor_type~mild	HAZELNUT
CLS~1000~10000~flavor_type~mild	VANILLA
CLS~1000~10000~flavor_type~special	CINNAMON
CLS~1000~10000~flavor_type~special	CARAMEL

17

AI Foundation Web Services

This chapter provides an overview of the AI Foundation Web Services. For additional details about any interfaces, see *Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface*.

AI Foundation Web Services

AI Foundation web services provide access to DB-bound configurations when a direct connection to the database is not desirable or possible; intra-day and ad-hoc access to certain application outputs is also provided. AI Foundation web services are REST-based; it is assumed that you are familiar with basic REST principles (such as the usage of HTTP verbs). AI Foundation web services provide access to a subset of application and output data, but do not fully mirror the user interface or export and import features of the backend. They are not a replacement for bulk data export, which must be done on a schedule as part of batch processing. However, access to the configuration can be used during implementation and upgrade periods, and AC and ASO export web services can serve as a means of obtaining incremental update data from a specified point in time (driven by a query parameter) as a means of intra-day processing.

All services support the query parameter contentType and the HTTP header Content-Type, with the supported values application/json and application/xml. The query parameter takes precedence; if no content type is supplied, then application/json serves as the default.

The AC and ASO export services have a dateMask query parameter that must adhere to the Java java.text.SimpleDateFormat rules (for example, dateMask=yyyyMMdd&exportDate=20151012). All date parameters must be sent in this format, and output dates and timestamps are returned according to this format.

The json/XML structure follows the corresponding DB table and view while being converted to the Java standard. That is, underscores are removed, camel case is used, and first letter is lowercase. For example, the RSE_CONFIG APPL_CODE column is returned as applCode. All data is returned as type string.

Authentication and Authorization

Basic authentication is used, so you may use any client software that supports it. Authorization is done for ADF-LDAP (OID) mapped roles, and only administrator roles are used (that is, the calling user must be in a duty that is mapped to the roles in Table 17–1).

Table 17–1 Mapped Administrator Roles

Service	Role	Mapped Role
DT	DemandTransferenceRole	ANALYTIC_EXPERT_JOB

Table 17–1 (Cont.) Mapped Administrator Roles

Service	Role	Mapped Role
CDT	CustomerDecisionTreeRole	ANALYTIC_EXPERT_JOB
AC	StoreClusterAdvancedRole	CLUSTERING_ADMINISTRATOR_JOB
ASO	Administrator	SPACE_ADMINISTRATOR_JOB
CS	CustomerSegmentAdvancedRole	CUSTOMER_SEGMENT_ADMINISTRATOR_JOB

Summary of Web Services

This section provides a summary of web services.

Access to RSE_CONFIG Table

Fetch Config Data

GET on /rase/resources/rse/parameters returns all RSE_CONFIG entries a specific user has access to (that is, all applCode RSE entries plus corresponding application entries). For example, if a user is in the ASO Administrator role, then the applCode SO will be returned as well. Here is a list of the fields:

- applCode
- paramName
- paramValue
- configurableFlg.
- descr

Fetch One Entry

GET on applCode,paramName. For example, /rase/resources/rse/parameters/RSE,PRIMARY_LANGUAGE_CODE returns a particular entry if the user has access to it.

Create an Entry

POST to /rase/resources/rse/parameters with a form having a single field called "content" and a value having the appropriate new parameter data in content type as specified by query parameter and header, if the user has access to it. The structure of the content must match what is returned by GET.

Update an Entry

PUT to applCode,paramName. For example, /rase/resources/rse/parameters/RSE, LOC_HIER_TYPE, with the form having fields called paramName, paramValue, descr, and configurableFlg, along with appropriate values, if the user has access to it.

Batch Update Entries

PUT to /rase/resources/rse/parameters with a form having a single field called "content" and a value having appropriate parameter data in the content type as specified by the query parameter and header. Note that if the user does not have access to a particular entry, it will be skipped. The structure of the content must match what is returned by GET.

Access to RSE_CONFIG_CODE Table

Fetch Data for an Entry

GET on applCode,paramName, paramCode. For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2 returns data, if user has access to it.

- applCode
- paramName
- paramCode
- paramValue
- configurableFlg
- descr

Create an Entry

POST to applCode,paramName,paramCode. For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2} with a form having a single field named "content" and a value having appropriate new parameter data in the content type as specified by query parameter and header, if the user has access to it. The structure of the content must match what is returned by GET.

Update an Entry

PUT to applCode,paramName, paramCode, For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2 with form having a single field named "content" and a value having appropriate parameter data in the content type as specified by query parameter / header, if user has access to it.The structure of the content must match what is returned by GET.

Delete an Entry

DELETE to applCode,paramName, paramCode. For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2, if the user has access to it.

Advanced Clustering Export

This section describes Advanced Clustering export.

Get Clusters

This service is based on rsestrclst.csv.

GET on /rase/resources/cis/export/cluster.

Parameters (in addition to dateMask and contentType) are all joined by logical AND:

Table 17–2 Advanced Clustering Export Parameters

Name	gType	Required	Query Logic for Corresponding Column if Parameter is Provided
exportedDt	Date	Required	Greater than or equal to this value
effStartDt	Date	Required	Greater than or equal to this value
effEndDt	Date	Required	Less than or equal to this value
prodHierTypeExtKey	String	Required	Equal to this value []

Table 17–2 (Cont.) Advanced Clustering Export Parameters

Name	gType	Required	Query Logic for Corresponding Column if Parameter is Provided
prodExtKey	String	Required	Equal to this value
locExtKey	String	Required	Equal to this value
locHierTypeExtKey	String	Required	Equal to this value

Customer Segment Export

This section describes Customer Segment export.

Get Segments

This service is based on rsestrclst.csv.

GET on /rase/resources/cis/export/segment.

Parameters (in addition to dateMask and contentType) are all joined by logical AND:

Table 17–3 Customer Segment Export Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
exportedDt	Date	Greater than or equal to this value
effStartDt	Date	Greater than or equal to this value
effEndDt	Date	Less than or equal to this value
prodExtKey	String	Equal to this value

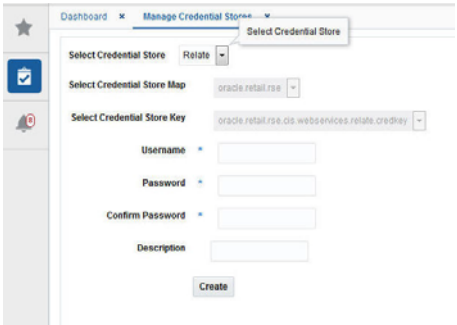
AIF Integration with ORCE (Customer Engagement)

This integration enables the application to send approved customer segments into Oracle Retail Customer Engagement (ORCE). Whenever customer segments are approved for merchandise, location in the application, a web service call is made to ORCE in order to save segments, along with its members and relevant attributes.

Credential Store

You can use the Manage Credential Stores screen, shown in Figure 17–1, to maintain ORCE credentials in the application.

Figure 17–1 Manage Credential Stores



Enter the following information into Manage Credential Stores:

Table 17–4 Manage Credential Stores Information

Field	Description
Username	Credential username created by ORCE to enable integration
Password	Credential password (at least one character) created by ORCE to enable integration
Confirm Password	Confirm password
Description	Credential store used for Customer Segment integration with Oracle Customer Engagement

Configuration

The following configuration must be updated for integration. By default, ORCE integration is disabled.

Table 17–5 Configuration

Configuration	Description
CUST_SEG_WS_RELATE_FLG	Flag to identify whether to publish customer segment to customer engagement. (Y/N)
CUST_SEG_WS_RELATE_ORGID	ORGID is the 3-letter ID provided by Oracle Customer Engagement to create and implement an authentication key. (REL)
CUST_SEG_ATTR_WS_RELATE_FLG	Flag to identify whether to publish customer segment attribute to customer engagement. (Y/N)
CUST_SEG_RELATE_HOSTNAME	URL host name to publish customer segment to customer engagement. This should be same as the CN name in the certificate.
CUST_SEG_RELATE_PORT	Port number to publish customer segment to customer engagement.
CUST_SEG_REL_SEGSERV_VERSION	Segment service version to use for publishing customer segment to customer engagement. (format vX_0 - v3_0)
CUST_SEG_REL_PROXY_HOSTNAME	Proxy host name to publish customer segment to customer engagement.
CUST_SEG_REL_HTTP_PROXY_PORT	HTTP proxy port number to publish customer segment to customer engagement.
CUST_SEG_REL_HTTPS_PROXY_PORT	HTTPS proxy port number to publish customer segment to customer engagement.
CUST_SEG_REL_BATCH_SIZE	Number of customers to add in batches after customer segment to customer engagement is published.

ASO Exports

This section describes ASO exports.

AIP/Replenishment Results

This service is based on so_assort_aiprepl_int.csv.

GET on /rase/resources/so/export/soAssortAipreplInt

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–6 AIP/Replenishment Results Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
assortmentSetId	String	Equal to this value
exportedDt	Date	Greater than or equal to this value

Output Aggregated Across Approved Runs

This service is based on so_assort_cm_int.csv.

GET on /rase/resources/so/export/soAssortCmInt.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–7 Output Aggregated Across Approved Runs Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
assortmentSetId	String	Equal to this value
exportedDt	Date	Greater than or equal to this value

Assortment Result Details

This service is based on so_assort_int.csv.

GET on /rase/resources/so/export/soAssortInt.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–8 Assortment Results Details Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
assortmentSetId	String	Equal to this value
exportedDt	Date	Greater than or equal to this value

Cross Reference Between Planograms and Finalized Assortments

This service is based on planogram_assortment.csv.

GET on /rase/resources/so/export/planogramAssortment.

Parameters (in addition to dateMask and contentType) all joined by logical OR:

Table 17–9 Cross Reference Between Planograms and Finalized Assortments Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
pogKey	String	Equal to this value
exportDate	Date	Greater than or equal to this value

POG Header

This service is based on planogram.csv.

GET on /rase/resources/so/export/planogram.

Parameters (in addition to dateMask and contentType) all joined by logical OR:

Table 17–10 POG Header Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
pogKey	String	Equal to this value
exportDate	Date	Greater than or equal to this value

POG Equipment Components

This service is based on equipment.csv.

GET on /rase/resources/so/export/equipment.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–11 POG Equipment Components Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
pogKey	String	Equal to this value
exportDate	Date	Greater than or equal to this value

POG/Stores Cross Refererence

This service is based on planogram_store.csv.

GET on /rase/resources/so/export/planogramStore.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–12 POG/Stores Cross Reference Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
pogKey	String	Equal to this value
storeKey	String	Equal to this value
exportDate	Date	Greater than or equal to this value

Finalized Assortment Product Hierarchies

This service is based on product_hierarchy.csv.

GET on /rase/resources/so/export/productHierarchy.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–13 Finalized Assortment Product Hierarchies Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
exportDate	Date	Greater than or equal to this value

Finalized Assortment Products

This service is based on product_position.csv.

GET on /rase/resources/so/export/productPosition.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–14 Finalized Assortment Products Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
pogKey	String	Equal to this value
skuKey	String	Equal to this value
exportDate	Date	Greater than or equal to this value

Product Display Style Information

This service is based on sku_details.csv.

GET on /rase/resources/so/export/skuDetails.

Parameters (in addition to dateMask and contentType) are all joined by logical OR:

Table 17–15 Product Display Style Information Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
skuKey	String	Equal to this value
skuName	String	Equal to this value
effectiveDate	Date	Greater than or equal to this value
expiryDate	Date	Greater than or equal to this value
exportDate	Date	Greater than or equal to this value

Retail Science Integration with XStore

Retail Science integrates with XStore; as part of this integration, XCenter broadcasts real time sales data with multiple transactions in one XML file to Retail Science. Retail Science, on receiving data, saves real time sales transactions, any price modifiers like promotions, price overrides, and returns in the database, which can be used as real time sales data for reporting.

This section provides the information necessary to send messages contained in version 1.0 of the POSLOG Services API. Information can be obtained using the Web Service Description Language (WSDL) in conjunction with a Simple Object Access Protocol (SOAP), XML Schema, and various methods contained in the classes of the API to provide the web service described below.

Endpoint

The URL for the WSDL is:

https://<host>:<port>/rasews/poslog/v1/PoslogStrReceiverApiService?wsdl, where <host> is the name or address of the server. For a cloud installation, <port> the default port number is 443.

The URL for the WSDL is: Retail Science SOAP endpoint is:

https://<host>:<port>/rasews/poslog/v1/PoslogStrReceiverApiService

This endpoint should be configured in the XCenter for Generic Poslog String broadcaster. For further details, see the XCenter configuration.

Authentication

Retail Science supports Basic Authentication integration with XCenter over https, using a username and password. XCenter integration with Retail Science relies on SSL to do the encryption of the username/password configuration. After a user is defined in Retail Science, XCenter must be configured with the userid/password, and Xcenter will use Base64-encoding and include it as part of the Http authorization header when invoking the SOAP service. XCenter service requests must be authenticated with a valid user/password combination and roles. Failure to provide a valid authorization header will result in a 403 - Forbiddenresponse.

The user must have following role set up:

POSLOGS_SERVICE_JOB - Point of Sales broadcast listener role to enable integration between Retail Science and Oracle XStore.

Retail Science and Retail Insight Integration

Retail Science internally triggers job LOAD_POSLOG_DATA_ADHOC; this job is predefined in Retail Process Orchestration and Monitoring (POM). As part of the configuration verification, ensure that LOAD_POSLOG_DATA_ADHOC is defined in POM.

Configuration

This section describes the configuration.

POM Server Detail Configuration

To ensure Retail Science to Retail Insights integration configuration is correct, verify RSE_CONFIG configuration for POM Server details.

Select Control & Tactical Center > Strategy & Policy Management

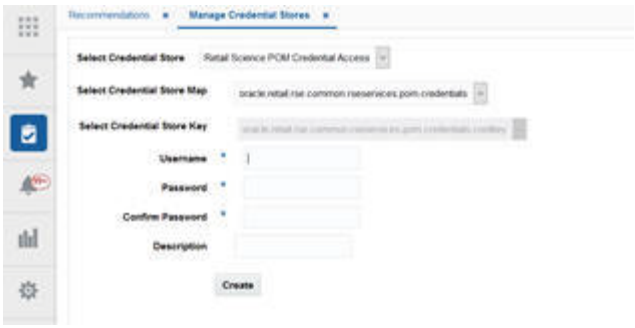
Filter APPL_CODE as "RSE" and PARAM_NAME as "RSE_POM_HOST". Determine if the value set as part of the deployment is correct. If the host is not correct, update the entries.

Figure 17–2 Edit Value

POM Server Credentials

Select Control & Tactical Center > Manage Credential Store to provide POM ReST endpoints credentials for Basic Authentication. All the POM ReST endpoints use Basic Authentication, with the same credentials as those used to log into the POM application. See Figure 17–3 to provide credentials.

Figure 17–3 POM Server Credentials



Once the configuration is complete, Retail Science will start triggering jobs for Retail Insights.

18

Batch Processing

This chapter provides an overview of the batch processing capabilities available for the application.

Overview

The implementation process involves loading data files for dimensions and fact data into the database. For new implementations, the best practice is to test the interfaces in a logical sequence, in small test cycles, using a Custom Batch Request process.

Once all required data has been loaded and all interfaces have been tested, the scheduled batch cycles that perform different tasks can be used, depending on the frequency involved. The application has INTRADAY processes that are used for ASO, as well as DAILY, WEEKLY, and QUARTERLY batch cycles, each of which performs different tasks, depending on which applications are being used.

Custom Batch Requests

This section describes processing that is valid through 18.x, and that will gradually be phased out as implementations migrate to 19.x. For information elated to 19.x, see Process Orchestration and Monitoring.

A custom batch request provides some flexibility in the execution of batch routines during the application initialization and setup stage. This process should not be used once the application is running its normal scheduled batch cycles. During this stage of the application setup, it is generally necessary to test interfaces to make sure they follow the correct formats and contain the proper data. In this way, an implementer can perform tests in a self-sufficient manner.

Managing Custom Batch Requests

To initiate a custom batch, upload a PROCESS_QUEUE file that contains entries to trigger the execution of the processes that are associated with those identifiers. Since most processes are triggered based on the receipt of inbound files or may be a request to trigger the execution of processes required to create an outbound file, the values that can be used inside the PROCESS_QUEUE file are generally the names of the data files. The values that can be used to trigger other batch steps are described in Table 18–1.

Once the PROCESS_QUEUE has been uploaded to the inbound directory of the FTP server, a PROCESS_QUEUE.complete file can be uploaded and created. This triggers the execution of the batch steps. Once the batch process is complete, a verification email notification is sent, provided the Manage Configuration screen has been configured for such email notification.

If the PROCESS_QUEUE contains a list of any inbound data files, these files must be uploaded prior to the creation of the PROCESS_QUEUE.complete file.

After the batch process completes, a file named PROCESS_QUEUE.log is created in the EXPORT directory of the FTP server. This file contains any details that may be relevant for the implementer. It may include SQL*Loader log file contents if errors occurred during the processing. Log files for the programs that were executed may also be included. Such information can help in determining the cause of the error. When the batch process completes, if any outbound files to be created are placed in the EXPORT directory on the FTP server so that they can be retrieved.

Handling Data Files

For the process described in this section, it is assumed that the PROCESS_QUEUE file contains the value of W_PRODUCT_DS.dat, which can trigger the execution of the batch processing for loading that file.

The data to be processed can be provided as a text file (for example, W_PRODUCT_DS.dat) or as a compressed file (for example, W_PRODUCT_DS.dat.gz). For RI interfaces, a context file can also be provided that lists the columns in the interface either as a text file (for example, W_PRODUCT_DS.dat.ctx) or a compressed file (for example, W_PRODUCT_DS.dat.ctx.gz). The PROCESS_QUEUE file specifies the interface name of W_PRODUCT_DS.dat, and the process that collects the data files then retrieves any file of these filename patterns.

If the process request requires that multiple files be processed, these files can also be provided in a zip file. The file handler looks for a file named ORASE_PROCESS_TRIGGER.zip, unzips the contents, and uses any files listed in PROCESS_QUEUE. If a file that was previously included in the ORASE_PROCESS_TRIGGER.zip file must be adjusted, it is possible to send that file individually, so that the entire zip file does not need to be recreated and retransmitted.

Supported PROCESS_QUEUE Trigger Values

In addition to supporting any inbound or outbound data files, some additional values, described in Table 18–1, can be used to trigger the execution of some specific batch processes.

Table 18–1 Trigger Values

Process Queue Trigger Text	Description
SIL_INIT	Initializes RI MCAL Current Date. This may be used as required to advance the business date.
SO_POST_PROC	Triggers the execution of a series of steps that perform the data processing required to operate after the successful staging and loading of individual SO data files.
EXPORT_PREP_DAILY	Many export files provide incremental data exports that have occurred since the most recent export process was run. This step resets the from/to date range for daily exports to include changes up through the time this process is executed. The from date is set to the date/time that was previously the to date value.
EXPORT_PREP_WEEKLY	Many of the application export files provide incremental data exports for periods that begin with the date of the last time the export process was run. This step resets the from/to date range for weekly exports so that it includes changes up through the time this process is executed. The From date is set to the date and time that were previously used for the To values.
EXPORT_PREP_QUARTERLY	Many of the application export files provide incremental data exports for periods that begin with the date of the last time the export process was run. This step resets the from/to date range for quarterly exports so that it includes changes up through the time this process is executed. The From date is set to the date and time that were previously used for the To date values.

Table 18–1 (Cont.) Trigger Values

Process Queue Trigger Text	Description
EXPORT_PREP_INTRADAY	Many of the application export files provide incremental data exports for periods that begin with the date of the last time the export process was run. This step resets the from/to date range for intraday exports so that it includes changes up through the time this process is executed. The From date is set to the date and time that were previously used for the To date values.

Incremental Exports

As described in Table 18–1, all incremental export files are controlled by a set of dates that define the beginning and ending range of data to be exported. This data is stored in a configuration table called RSE_EXP_GRP and can be seen in the Manage Configuration screen. Each incremental export has a date associated with the data to be exported. Only data that has a date/time value between the FROM_DT and TO_DT columns of the RSE_EXP_GRP that it is associated with, will be exported when the export file is created.

When testing an application, it is important to realize that if a test export of data is required, you must make sure that data is available to be exported and that the data is associated with a date that is in the range of the export group. If an export runs and does not produce any data in the file, you should check the values of the Export Group to ensure the dates were not set incorrectly.

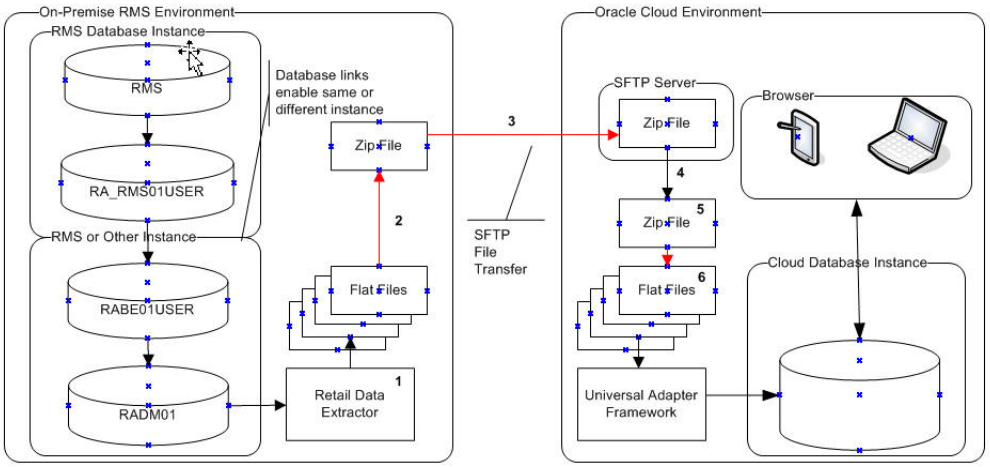
When you create or process data in the application user interface, want to test the export of that data, you must advance Export Group's date range by running the appropriate export preparation step as described in this chapter. This causes the date range to advance and enables the exporting of the data that is available for exporting. Note that if the Export Group date range is advanced too many times, the data that you want to export may no longer be in the current range for exporting.

You may encounter such issues when using this custom batch process to trigger the execution of exports; however, these issues will not occur once the application is running the batch routines in an automated manner, because the batch processes are only executed once per batch cycle.

Batch Process Flow

Figure 18–1 illustrates the batch process flow.

Figure 18–1 Batch Process Flow



Here is the process.

1. The on-premise batch shell script extracts data to files initiated by the customer scheduler.
2. Merch batch script creates the zip file named RI_RMS_DATA.zip. Additionally, zip files named RI_CE_DATA.zip and RI_MFP_DATA.zip should be created.
3. You should sftp the three zip files. Then, create a file named "COMPLETE" in the sftp directory COMMAND.
4. After the COMPLETE file is found in the COMMAND directory, the file watcher initiates the processing of files and places them in the landing directory of the cloud server.
5. The presence of the COMPLETE file in the landing directory releases the batch load processing.
6. The batch load process begins with tasks that
   a. Archive the files that have been received in a date/time stamped directory.
   b. Perform the presence validation exercise that verifies that all expected files for the customer's subscribed applications in the zipped files. This terminates if any expected files are missing.
   c. Clear the previous day's files from the $MMHOME/data/staging directory.
   d. Unzip the zip file into the $MMHOME/data/staging directory.

Table 18–2 lists the zip files.

Table 18–2 Supported Zip Files

Zip File Name	Frequency	File Type	Notes
RI_RMS_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
RI_CE_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
RI_MFP_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
ORASE_WEEKLY.zip	Weekly	Inbound	Any inbound file that does not start with W_* and has a weekly frequency can be placed in here.

Table 18–2 (Cont.) Supported Zip Files

Zip File Name	Frequency	File Type	Notes
ORASE_INTRADAY.zip	Intraday	Inbound	Any inbound file that has an intraday frequency can be placed in here.
ORASE_WEEKLY_extract.zip	Weekly	Outbound	Any outbound file that has a weekly frequency will be placed in here.
ORASE_INTRADAY_extract.zip	Intraday	Outbound	Any outbound file that has an intraday frequency will be placed in here.

Configuring Additional Data Files

It may be necessary to configure support for additional data files into AIF, beyond the ORASE_WEEKLY.zip and ORASE_INTRADAY.zip. If this is required, it is possible to configure additional files to go along with those. This section describes how to add support for this.

It is possible to receive additional zip files, with additional suffixes to the existing zip. In this example, the assumption is that a new zip file named ORASE_WEEKLY_IP.zip must be processed when ORASE_WEEKLY.zip is processed. The IP portion of this is what can be configured, as explained below.

Using the Retail AI Foundation Platform's UI, select the Data Management / Manage Configuration menu options. Then, select RSE_CONFIG_CODE as the table to configure. If you want to search for existing configurations, you can enter a search value of "%ZIP" in the PARAM_NAME Search field and select Search. There are no default extensions defined here, but once one has been created, this will display them.

To add a new entry to allow processing this additional file, select the Create Param icon to add a new configuration. You can enter values such as those shown in Table 18–3 into the dialog box.

Table 18–3 Additional Data Files

Field	Value	Notes
APPL_CODE	RSE	
PARAM_NAME	ORASE_WEEKLY_ZIP	The format of this is important. If adding an additional file for the Intraday batch, the value would be ORASE_INTRADAY_ZIP instead.
PARAM_CODE	IP	This is the suffix fthat the additional zip will use.
PARAM_VALUE	Y	Y to enable this, or N to disable this file.
CONFIGURABLE_FLG	Y	Fixed value.
UPDATEABLE_FLG	N	Fixed value.
DESCR	Additional zip for IP files.	Adjust this description so it describes the zip contents/source.

Based on the example values in Table 18–3, a new zip file named ORASE_WEEKLY_IP.zip will now be expected when the ORASE_WEEKLY.zip is expected.

If a file configured as defined above must be temporarily disabled, you can edit the PARAM_VALUE via the UI,so that it has a value of "N" instead of a value of "Y".

File Transmissions

When a file is sent to the SFTP server for processing, two approaches are available to process the file. After sending a file to be processed to the SFTP server, it is necessary to also send a "COMPLETE" file. Two approaches are available for this. One is to send a file named COMPLETE, in the COMMAND directory (for example, COMMAND/COMPLETE). When this is done, all files that were sent to the SFTP server will be pushed internally to an area accessible to the application. A secondary approach is to create an individual "complete" file to signal that the specific file is now ready to be pushed to the application area. This approach uses an additional suffix of ".complete" to signal that the file is completed.

If multiple servers are sending files to the SFTP server independently, then it is important to use individual complete files to signal when that file has been finished. Otherwise, it would be possible to move a file that is still being transferred.

Using the example of ORASE_WEEKLY.zip and ORASE_WEEKLY_IW.zip: if ORASE_WEEKLY.zip is transmitted at 1:00am, and completes at 1:10am, and ORASE_WEEKLY_IW.zip is transmitted 1:05am and completes at 1:06am, it would be problematic if, after the completion of ORASE_WEEKLY_IW.zip, a COMMAND/COMPLETE file was provided, as this would result in the movement of both of these zip files, even though ORASE_WEEKLY.zip has not yet finished being transferred. Therefore, in this situation, it is necessary to use a completion trigger of ORASE_WEEKLY.zip.complete and ORASE_WEEKLY_IW.zip.complete, which indicates that each of the respective files have completed their transmission.

If a file is sent to the SFTP server, and no "complete" file was provided, then the file will not become available for processing by the application.

19

Innovation Workbench

Innovation Workbench (IW) is a workspace that provides read-only access to application data objects and clean data by using Oracle APEX. This extension is a workspace for advanced analytics users to add new implementations by using Oracle Advanced Analytic (ODM) algorithms that are implemented as SQL/PLSQL functions. This chapter provides features, examples, and implementation details that are available in Innovation Workbench.

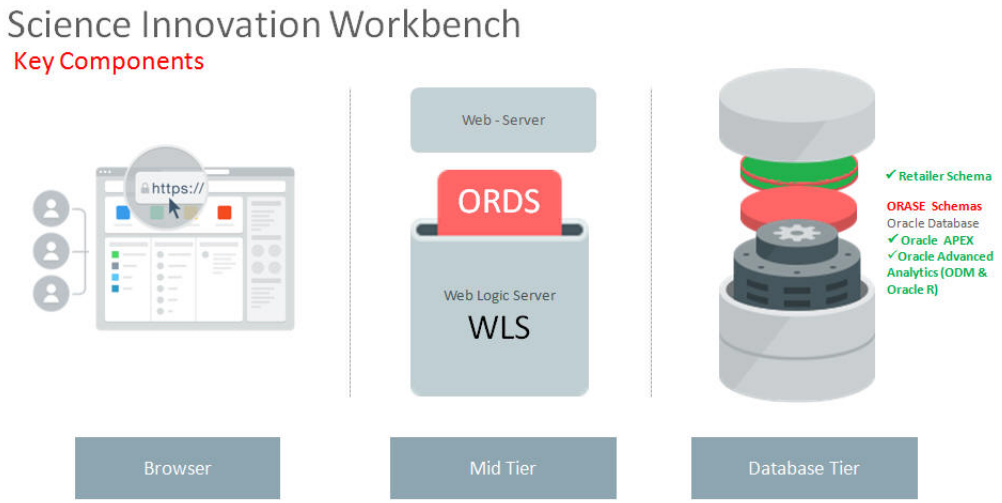
The key features available in Innovation Workbench are:

- Schema as a Service, in which a user can use AIF cleansed, read-only data, upload and combine data to gain insights, and upload retail application data to further analyze and mine data.
- Advanced Analytics, allows a user to use Oracle Machine Learning and Advanced Analytic algorithm that are implemented as SQL/PLSQL functions.
- Visualize Data, enables a user to explore data, develop and visualize data using charts, and review reports using the browser.
- RESTful Web Service, allows a user to declaratively create and edit RESTful service definitions using SQL Query.
- SQLWorkshop, enables a user to view and manage database objects.

Components

The key components of Science Innovation Workbench are Retailer Workspace Schema, Oracle APEX, and Oracle Advanced Analytics (Oracle Data Mining).

Figure 19–1 Innovation Workbench Key Components



Retailer Workspace Schema

Innovation Workbench provides retailer with a logical work area (Retailer Workspace) that is associated with a predefined database schema. The schema(s) store the database objects and provide read access to an existing retailer's application data objects. The retailer workspace schema is an independent schema for a retailer to use and these data objects are owned by retailer.

Oracle APEX

This section describes Oracle APEX, a web-based software development environment.

Workspace

A workspace <RETAILER_WORKSPACE> is a predefined workspace for the retailer where workspace users can create database objects and applications. The workspace has privileges to the allocated <RETAILER_WORKSPACE_SCHEMA> database schema.

Users and Roles

Innovation Workbench has two types of users: application developers and workspace administrators, and they can be created using Oracle APEX.

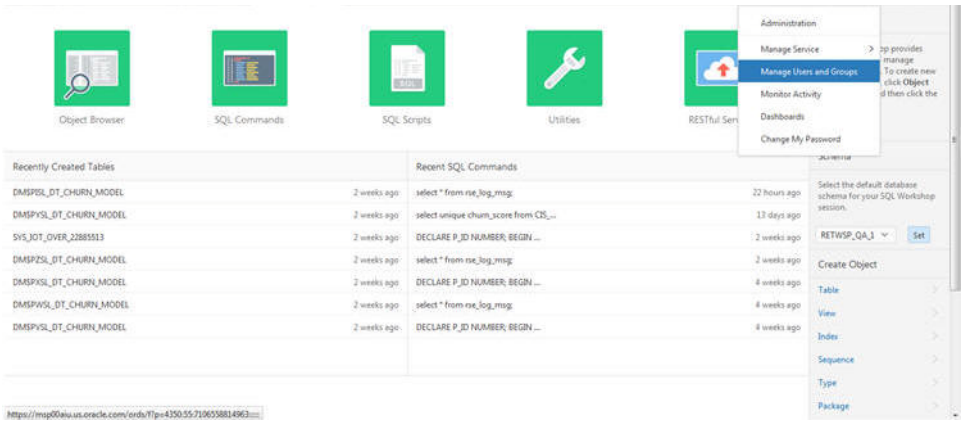
Innovation Workbench Administrators

The workspace administrator role is already created for the retailer; the administrator can create and edit developer accounts, manage groups, and manage development services.

Innovation Workbench Developer

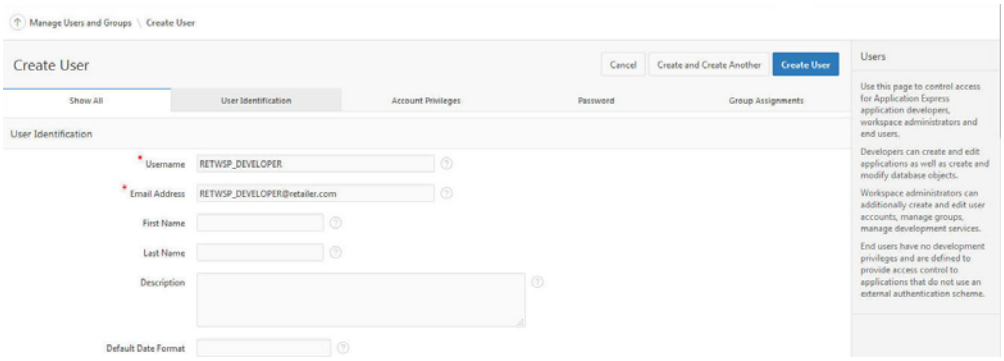
Workspace administrators can create workbench developers by selecting Manage Users and Groups. Developers can create and modify applications and browse database objects in an allocated workspace and schema. The retailer workspace schema has privileges required by Oracle Data Mining for executing analytic models.

Figure 19–2 Manage Users and Groups



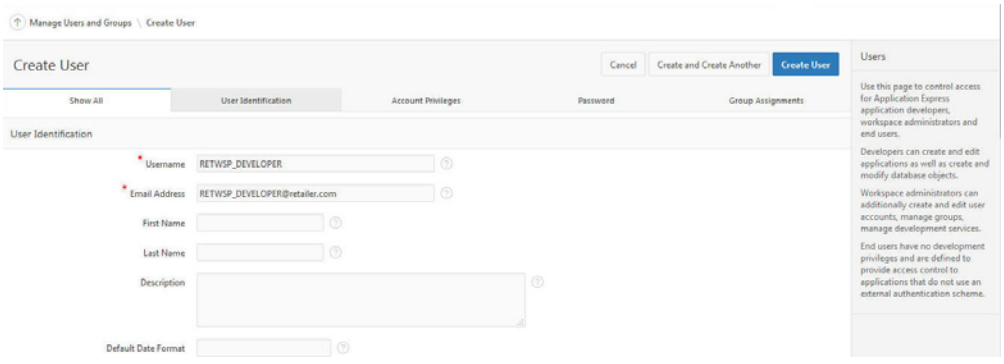
1. Create User. This area displays the Oracle APEX Create User screen that can be used to create a new Developer account and assign a RESTful Service group. Note that this user must also be created in Identity Management with the same username and password.

Figure 19–3 Create User



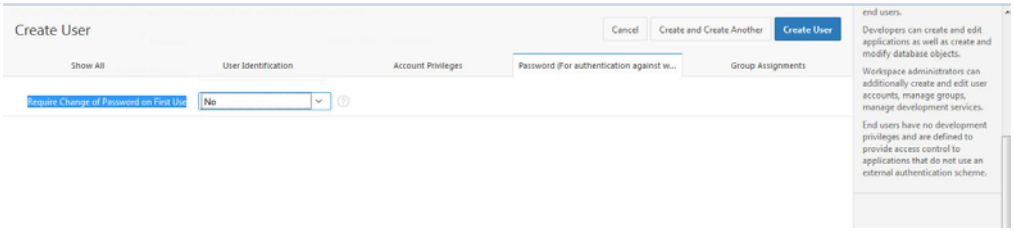
2. Assign Group RESTful Service.

Figure 19–4 Assign Group



3. Set Require Change of Password on First Use to No.

Figure 19–5 Require Change of Password



SQL Workshop

The SQL Workshop provides tools to view and manage database objects. To create new database objects, click **Object Browser** and then click **Create**.

Figure 19–6 SQL Workshop Object Browser: Reading Database Objects

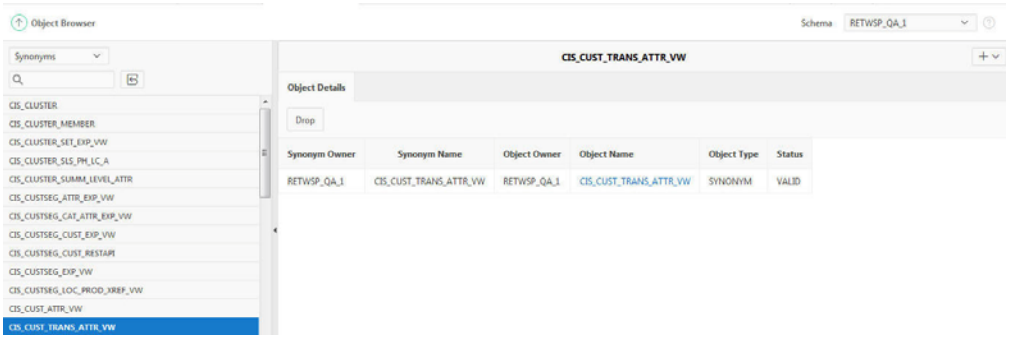


Figure 19–7 SQL Workshop Create Object

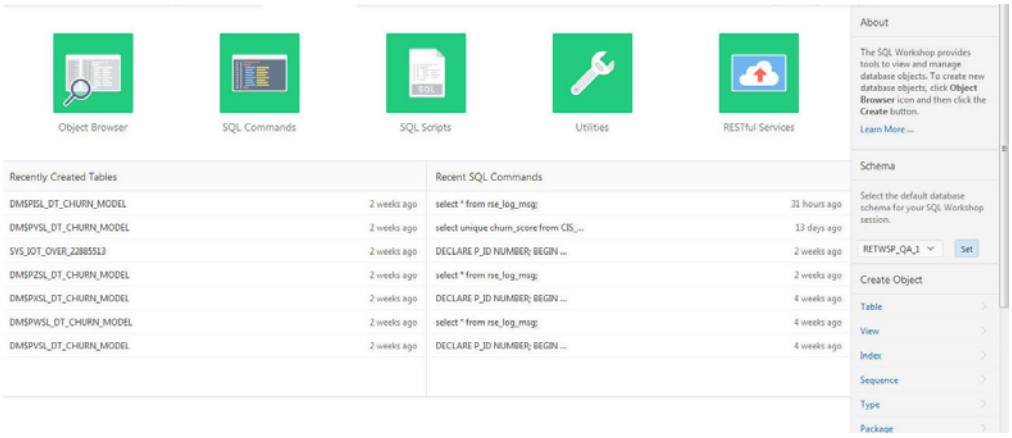


Figure 19–8 SQL Workshop SQL Command: Executing Ad Hoc SQLs

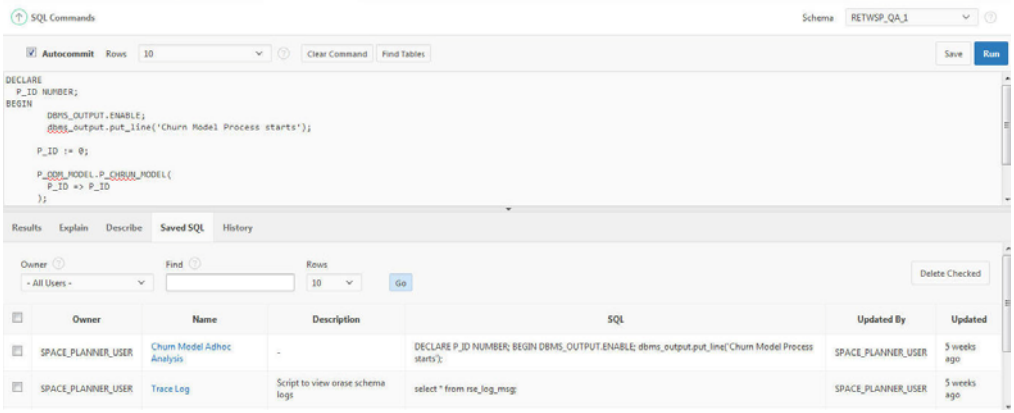
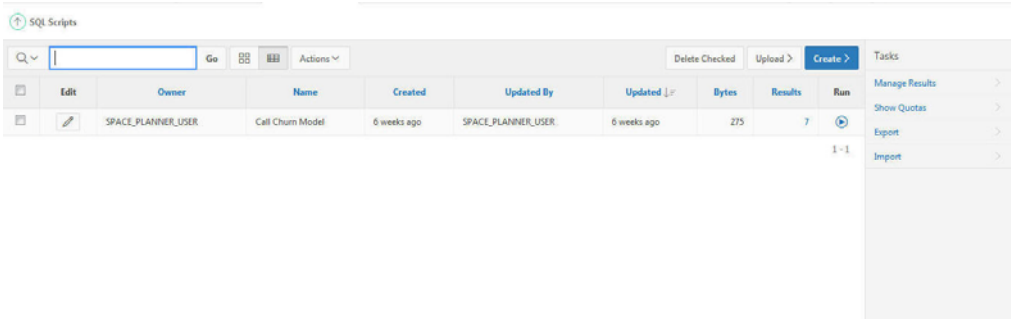


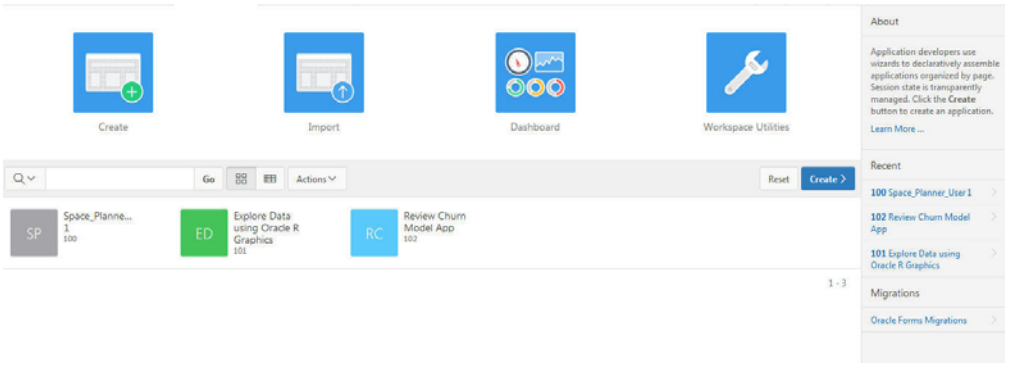
Figure 19–9 SQL Workshop SQL Scripts: Uploading, Executing, and Running



Application Builder

Application developers use wizards to declaratively assemble applications organized by page. The session state is transparently managed. Click **Create** to create an application.

Figure 19–10 Application Builder

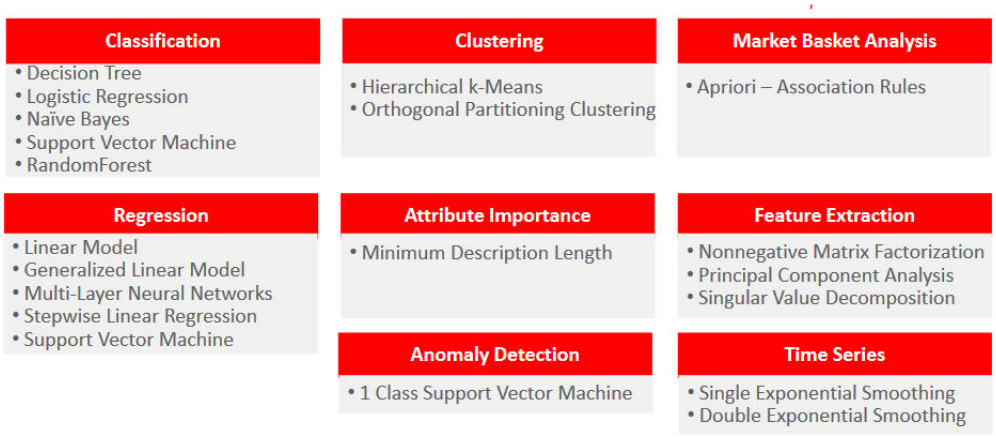


Oracle Advanced Analytics

Oracle Advanced Analytics has, as one of the components in Oracle Database Enterprise Edition, Oracle Data Mining. Innovation Workbench assigns privileges required by Oracle Data Mining for execution.

Oracle Data Mining offers a comprehensive set of in-database algorithms for performing a variety of mining tasks, such as classification, regression, anomaly detection, feature extraction, clustering, and affinity analysis.

Figure 19–11 Machine Learning Algorithms in Database



Oracle Data Mining

Oracle Data Mining can be used to build and deploy predictive and descriptive data mining applications, add intelligent capabilities to existing applications, and generate predictive queries for data exploration.

The Oracle Data Mining developers guide, samples, and tutorials are available at the following websites.

Oracle Data Mining Developer's Guide

http://www.oracle.com/pls/db121/vbook_subject?subject=dma

Data Mining Concepts

https://docs.oracle.com/database/121/DMCON/toc.htm

Oracle Data Mining Sample Programs

http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/odm-samples-194497.html

Samples can be downloaded odm12csampleprograms-2184025.7z

How to Invoke Oracle Data Mining

The following scripts show how to invoke Oracle Data Mining Classification script that creates a classification model using the Decision Tree algorithm.

```
CREATE OR REPLACE PACKAGE BODY pkg_odm_model
AS
  -- DESCRIPTION - This script creates a classification model using the Decision
Tree algorithm.
PROCEDURE proc_churn_model(
    p_id NUMBER)
IS
BEGIN
  DECLARE ---------- start drop RETWSP_CUST_CHURN_MODEL
    not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_found, -40203);
```

```
  BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHURN_MODEL Tables');
    DBMS_DATA_MINING.DROP_MODEL('RETWSP_CUST_CHURN_MODEL');
    dbms_output.put_line('End Drop RETWSP_CUST_CHURN_MODEL Tables');
  EXCEPTION
  WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHURN_MODEL not found');
  END; ------------- end drop RETWSP_CUST_CHURN_MODEL
  --------------------------
  -- CREATE A SETTINGS TABLE
  --
  -- The default classification algorithm is Naive Bayes. In order to override
  -- this, create and populate a settings table to be used as input for
  -- CREATE_MODEL.
  --
  DECLARE ---------- start drop RETWSP_CUST_CHMDL_SETTINGS
    not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_found, -40203);
  BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHMDL_SETTINGS Tables');
    EXECUTE IMMEDIATE 'DROP TABLE RETWSP_CUST_CHMDL_SETTINGS';
    dbms_output.put_line('End Drop RETWSP_CUST_CHMDL_SETTINGS Tables');
  EXCEPTION
  WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_SETTINGS not found');
  END; ------------- end drop RETWSP_CUST_CHMDL_SETTINGS
  DECLARE ---------- start drop RETWSP_CUST_CHMDL_COST
    not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_found, -40203);
  BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHMDL_COST Tables');
    EXECUTE IMMEDIATE 'DROP TABLE RETWSP_CUST_CHMDL_COST';
    dbms_output.put_line('End Drop RETWSP_CUST_CHMDL_COST Tables');
  EXCEPTION
  WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
  END; ------------- end drop RETWSP_CUST_CHMDL_COST
  DECLARE ---------- start create table RETWSP_CUST_CHMDL_SETTINGS
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
  BEGIN
    dbms_output.put_line('Start Create RETWSP_CUST_CHMDL_SETTINGS Tables');
    EXECUTE IMMEDIATE 'CREATE TABLE RETWSP_CUST_CHMDL_SETTINGS (
setting_name  VARCHAR2(30),
setting_value VARCHAR2(4000))';
    dbms_output.put_line('End Create RETWSP_CUST_CHMDL_SETTINGS Tables');
  EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('Exception not found');
  END; ------------- end create table RETWSP_CUST_CHMDL_SETTINGS
  DECLARE ---------- Create RETWSP_CUST_CHMDL_COST Tables begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
  BEGIN
    dbms_output.put_line('Start Create RETWSP_CUST_CHMDL_COST Tables');
    EXECUTE IMMEDIATE 'CREATE TABLE RETWSP_CUST_CHMDL_COST (
actual_target_value          NUMBER,
predicted_target_value       NUMBER,
cost                         NUMBER)';
    dbms_output.put_line('End Create RETWSP_CUST_CHMDL_COST Tables');
```

```
  EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
  END; ------------- Create RETWSP_CUST_CHMDL_COST Tables ends
  -- CREATE AND POPULATE A COST MATRIX TABLE
  --
  -- A cost matrix is used to influence the weighting of misclassification
  -- during model creation (and scoring).
  -- See Oracle Data Mining Concepts Guide for more details.
  --
  dbms_output.put_line('Start Insert records into RETWSP_CUST_CHMDL_COST');
  DECLARE ---------- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
  BEGIN
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,0,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,1,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,2,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,3,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,0,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,1,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,2,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,3,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,0,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,1,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,2,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,3,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,0,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,1,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,2,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,3,0)';
    dbms_output.put_line('End Insert Records');
  EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
  END; ------------- sub-block ends
  dbms_output.put_line('End Insert records into RETWSP_CUST_CHMDL_COST');
  -- Populate settings table
  DECLARE ---------- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
    v_stmt               VARCHAR2(4000);
    v_algo_name          VARCHAR2(100);
    v_algo_decision_tree VARCHAR2(100);
  BEGIN
    dbms_output.put_line('Start Populate settings table' || dbms_data_mining.algo_
name);
    dbms_output.put_line('Start Populate settings table' || dbms_data_mining.algo_
decision_tree);
    v_algo_name          := dbms_data_mining.algo_name;
    v_algo_decision_tree := dbms_data_mining.algo_decision_tree;
    v_stmt               := 'INSERT INTO RETWSP_CUST_CHMDL_SETTINGS (setting_name,
setting_value) VALUES (''' || v_algo_name || ''',''' || v_algo_decision_tree ||
''')';
    dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
    EXECUTE IMMEDIATE v_stmt;
  EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('Exception not found');
  END; ------------- sub-block ends
```

```
  DECLARE ---------- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
    v_table_name  VARCHAR2(100);
    v_matrix_cost VARCHAR2(100);
  BEGIN
    v_table_name  := dbms_data_mining.clas_cost_table_name;
    v_matrix_cost := 'RETWSP_CUST_CHMDL_COST';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_SETTINGS (setting_name,
setting_value) VALUES' || '(''' || v_table_name || ''',''' || v_matrix_cost ||
''')';
    dbms_output.put_line('End Populate settings table');
  EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('Exception not found');
  END; ------------- sub-block ends
  ---------------------
  -- CREATE A NEW MODEL
  --
  -- Build a DT model
  dbms_output.put_line('Start Create Churn Model');
  DBMS_DATA_MINING.CREATE_MODEL( model_name => 'RETWSP_CUST_CHURN_MODEL', mining_
function => dbms_data_mining.classification, data_table_name => 'cis_cust_attr_
vw', case_id_column_name => 'CUSTOMER_ID', target_column_name => 'CHURN_SCORE',
settings_table_name => 'RETWSP_CUST_CHMDL_SETTINGS');
  dbms_output.put_line('End Create Churn Model');
  -------------------------
  -- DISPLAY MODEL SIGNATURE
  --
  column attribute_name format a40
  column attribute_type format a20
  SELECT attribute_name,
    attribute_type
  FROM user_mining_model_attributes
  WHERE model_name = 'RETWSP_CUST_CHURN_MODEL'
  ORDER BY attribute_name;
END p_chrun_model;
END pkg_odm_model;
```

Test ODM Model

```
DECLARE
  RUN_ID NUMBER;
BEGIN
  DBMS_OUTPUT.ENABLE;
  dbms_output.put_line('Churn Model Process starts');
  RUN_ID := 1001;
  pkg_odm_model.proc_churn_model( RUN_ID => RUN_ID );
  dbms_output.put_line('Churn Model Process ends');
END;
```

Notebooks

Data scientists can use the Innovation Workbench Notebook to create notebooks, which are collections of documentation, snippets of code, and visualizations. These notebooks are bundled with key python modules for machine learning, data mining, natural language processing, network analysis, and optimization solvers.

Here is a list of some of the python packages that are bundled with AIF. These packages provide features that span data mining processes of from data exploration to data visualization.

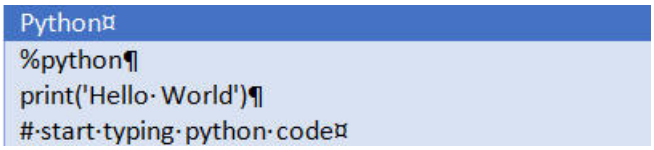
- Ensemble Machine Learning Algorithms with scikit-learn
- Data exploration and analysis using Pandas; NumPy; SciPy
- Data visualization using Matplotlib; Seaborn
- Data storage using cx_Oracle
- Graph algorithms using Networkx
- Optimization using Gurobi Solver

Data Studio graph analytics includes numerous built-in graph algorithms. Some of the classes of algorithms that it provides include:

- Community Detection
- Path Finding
- Ranking
- Recommendation
- Pattern Matching
- Influencer Identification

Invoking Python Code

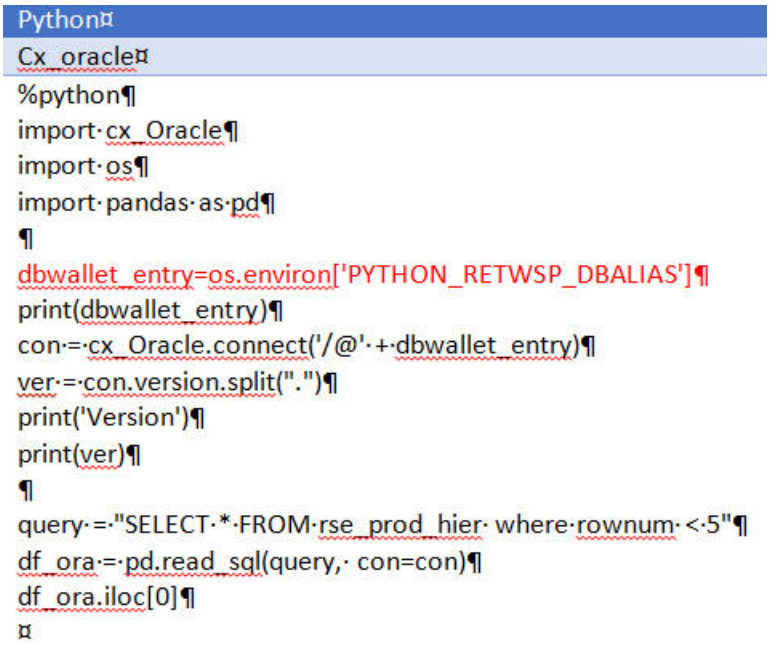
Figure 19–12 Invoking Python Code



Connecting to a database using cx_Oracle

Database connection string should be fetched from environment variable. Refer to code in Figure 19–13 in red.

Figure 19–13 Connecting to a Database Using cx_Oracle



Executing Gurobi code

Create a Gurobi environment by calling RseGurobiEnv.getGurobiEnv(). Refer to the code in red in Figure 19–14

Figure 19–14 Executing Gurobi Code

Python¤
Gurobi¤

```
%python¶
from·gurobipy·import·*¶
from·rsecommon.analytics.gurobi·import·*¶
try:¶
¶
····instance·=·rseenv.RseGurobiEnv.getInstance()¶
····env·=·instance.getGurobiEnv()¶
····m·=·Model("mip1",·env)¶
¶
····#·Create·variables¶
····x·=·m.addVar(vtype=GRB.BINARY,·name="x")¶
····y·=·m.addVar(vtype=GRB.BINARY,·name="y")¶
····z·=·m.addVar(vtype=GRB.BINARY,·name="z")¶
¶
····#·Set·objective¶
····m.setObjective(x·+·y·+·2·*·z,·GRB.MAXIMIZE)¶
¶
····#·Add·constraint:·x·+·2·y·+·3·z·<=·4¶
····m.addConstr(x·+·2·*·y·+·3·*·z·<=·4,·"c0")¶
¶
····#·Add·constraint:·x·+·y·>=·1¶
····m.addConstr(x·+·y·>=·1,·"c1")¶
¶
····m.optimize()¶
¶
····for·v·in·m.getVars():¶
········print('%s·%g'·%·(v.varName,·v.x))¶
¶
····print('Obj:·%g'·%·m.objVal)¶
¶
except·GurobiError·as·e:¶
····print('Error·code·'·+·str(e.errno)·+·":·"·+·str(e.message))¶
¶
except·AttributeError:¶
····print('Enco111untered·an·attribute·error')¶
¤
```

Invoking Oracle SQL/PL/SQL procedures

Figure 19–15 Invoking Oracle SQL/PL/SQL Procedures

Oracle·JDBC¤

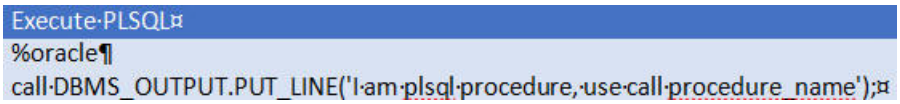
```
%oracle¤
```

Executing SQL

Figure 19–16 Executing SQL

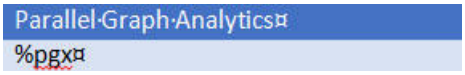
Executing PL/SQL

Figure 19–17 Executing PL/SQL



Invoking Parallel Graph Analytics

Figure 19–18 Invoking Parallel Graph Analytics



Creating a graph from scratch

Figure 19–19 Graph Creation

```
Build a graph from scratch
%pgx
builder = session.newGraphBuilder()

// create a few vertices
v1 = builder.addVertex(1).addLabel("Person").setProperty("name", "Charles").setProperty("age", 24)
v2 = builder.addVertex(2).addLabel("Person").setProperty("name", "Susan").setProperty("age", 36)
v3 = builder.addVertex(3).addLabel("Person").setProperty("name", "Philip").setProperty("age", 22)
v4 = builder.addVertex(4).addLabel("Place").setProperty("name", "McDonalds")
v5 = builder.addVertex(5).addLabel("Place").setProperty("name", "Wendy's")
v6 = builder.addVertex(6).addLabel("Place").setProperty("name", "Office")
v7 = builder.addVertex(7).addLabel("Car").setProperty("name", "VW Passat")
v8 = builder.addVertex(8).addLabel("Car").setProperty("name", "Toyota Highlander")
v9 = builder.addVertex(9).addLabel("Place").setProperty("name", "Central Park")

// vertices can have multiple labels
v4.addLabel("Restaurant")
v5.addLabel("Restaurant")

// create a few edges
builder.addEdge(0, v1, v2).setLabel("knows")
builder.addEdge(1, v2, v1).setLabel("knows")
builder.addEdge(2, v2, v3).setLabel("knows")
builder.addEdge(3, v4, v5).setLabel("connected")
builder.addEdge(4, v5, v6).setLabel("connected")
builder.addEdge(5, v4, v6).setLabel("connected")
builder.addEdge(6, v1, v7).setLabel("owns").setProperty("since", 1998)
builder.addEdge(7, v2, v8).setLabel("owns").setProperty("since", 2003)
builder.addEdge(8, v1, v6).setLabel("worksAt")
builder.addEdge(9, v2, v6).setLabel("worksAt")
builder.addEdge(10, v9, v4).setLabel("connected")
// store resulting graph in a variable 'graph'
graph = builder.build()
```

Scheduling Innovation Workbench Python Notebook

This section describes the scheduling process.

IDCS Setup

In order to use the Datastudio Notebook REST API, the Oracle Identity Cloud Service (IDCS) application is necessary. The IDCS application must be created with following grant_type as (password) and client_credentials.

Once the IDCS application is created, a new clientId:clientSecret will be generated. Note these credentials in order to perform a REST API request. A user must also be created in order to set up the Datastudio User Session.

Scheduling Jobs

Notebooks can be scheduled for automatic execution. To implement this, make a POST request to the using REST API call with the following payloads.

Execute once, immediately.

```
{
  "cronSchedule": "",
  "timeEnd": "",

}
```

Execute at a regular interval, and stop at a given date.

```
{
  "cronSchedule": "0/1440 * * * * *",
  "timeEnd": "2021-02-24T21:10:34.400Z",
  "id": "dsYVGG9Mvw"
}
```

Execute at a regular interval, indefinitely.

```
{
  "cronSchedule": "0/ 1440 * * * * *",
  " timeEnd": "",
  "id": "dsYVGG9Mvw"
   }
```

REST API Documentation

The REST API documentation details for request/response are located here:

http://datastudio.oraclecorp.com/docs/apidoc/swagger-ui.html#/

Example

Here is a REST API curl call example showing how to schedule a notebook.

Table 19–1 Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response

1	Fetch Authentication Token
	POST request
		curl --location --request POST 'https://<IDCS_HOST>/oauth2/v1/token' \
		--header 'Authorization: Basic <Base64 ClientID:ClientSecret>' \
		--header 'Content-Type: application/x-www-form-urlencoded' \
		--data-urlencode 'grant_type=password' \
		--data-urlencode 'scope=urn:opc:idm:__myscopes__' \
		--data-urlencode 'username=<username>' \
		--data-urlencode 'password=<password>'
			Response will have authentication token. To make subsequent request:
			{
			"access_token": "<Access Token>",
			"token_type": "Bearer",
			"expires_in": 3600
			}

2	Setup Session
	GET request
		curl --location --request GET '<APP_HOST>/datastudio/v2/sessions/user' \
		--header 'Authorization: Bearer <Token>'
			{
			"username": "<user_name>",
			"permissions": [
			"graph_create",
			"export_all",
			"view_permissions_tab",
			"import_notebook",
			"view_dashboard_tab",
			"view_credentials_tab",
			"create_notebook",
			"create_credential",
			"view_interpreter_tab",
			"delete_all"
			],
			"authToken": "<Token>"
			}

Note: The first two steps above are mandatory. Execute any of the following steps.			

3	Schedule Notebook
	POST request
		curl --location --request POST 'https://<APP_HOST>/datastudio/v2/notebooks/schedule' \
		--header 'Authorization: Bearer <Token>' \
		--header 'Content-Type: application/json' \
		--data-raw '{
		"cronSchedule": "",
		"timeEnd": "",
		"id": "dsYVGG9Mvw"
		}'
			{
			"id": "dsja4YWg",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

Table 19–1 (Cont.) Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response

4	Schedule Notebook Paragraphs to execute immediately
	POST request
		curl --location --request POST 'https://<APP_ HOST>/datastudio/v2/notebooks/schedule' \
		--header 'Authorization: Bearer <Token>' \
		--header 'Content-Type: application/json' \
		--data-raw '{
		"cronSchedule": "",
		"paragraphs": [
		{
		"id": "dsB0zM58"
		}
		],
		"timeEnd": "",
		"id": "dsYVGG9Mvw"
		}'
			{
			"id": "dsVBqlO3",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

5	Schedule Notebook to execute immediately.
	Execute a notebook every five minutes.
	POST request
		curl --location --request POST 'https://<APP_ HOST>/datastudio/v2/notebooks/schedule' \
		--header 'Authorization: Bearer <Token>' \
		--header 'Content-Type: application/json' \
		--data-raw '{
		"cronSchedule": "*/1440 * * * * *",
		"endDate": "2020-12-17T14:10:34.400Z",
		"id": "dsYVGG9Mvw"
		}'
			{
			"id": "dsja4YWg",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

Table 19–1 (Cont.) Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response

6	Schedule Notebook Paragraphs.
	Execute a notebook/paragraph every five minutes.
	POST request
		curl --location --request POST 'https://<APP_ HOST>/datastudio/v2/notebooks/schedule' \
		--header 'Authorization: Bearer <Token>' \
		--header 'Content-Type: application/json' \
		--data-raw '{
		"cronSchedule": "*/1440 * * * * *",
		"paragraphs": [
		{
		"id": "dsB0zM58"
		}
		],
		"timeEnd": "2020-12-17T14:10:34.400Z",
		"id": "dsYVGG9Mvw"
		}'
			{
			"id": "dsVBqlO3",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

7	Schedule Notebook Paragraphs.
	Execute a notebook/paragraph adhoc with parameters.
		curl --location --request POST 'https://<hostname>/datastudio/v2/notebooks /schedule' \
		--header 'Authorization: Bearer <Token>' \
		--header 'Content-Type: application/json' \
		--data-raw '{
		"cronSchedule": "",
		"paragraphs": [
		{
		"id": "dsa7kZYv",
		"message": "string",
		"params": {
		"FirstName": "Sandhya",
		"LastName": "Lonial"
		}
		}
		],
		"timeEnd": "2020-02-24T21:10:34.400Z",
		"id": "dsYVGG9Mvw"
		}'
			{
			"id": "dsOB0yra",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

Table 19–1 (Cont.) Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response

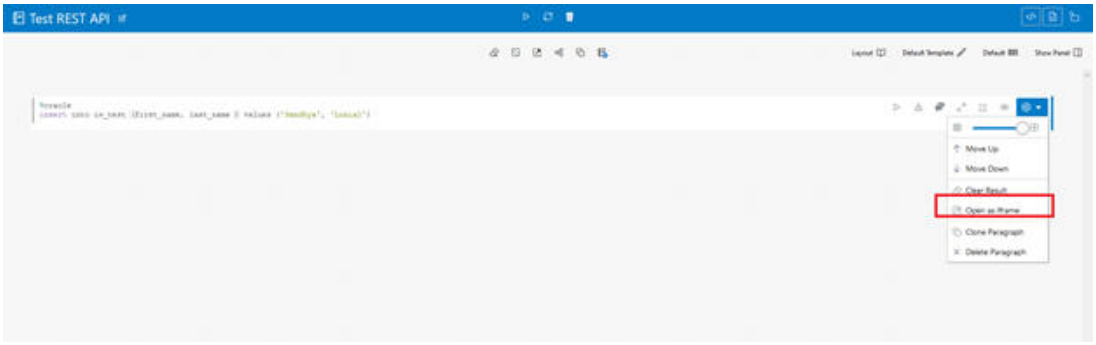
8	Schedule Notebook Paragraphs.
	Execute a multiple paragraphs.
		{
		"cronSchedule": "",
		"paragraphs": [
		{
		"id": "dsB0zM58"
		},
		{
		"id": "dsa7kZYv",
		"message": "string",
		"params": {
		"FirstName": "Sandhya",
		"LastName": "Lonial"
		}
		}
		],
		"timeEnd": "2020-02-24T21:10:34.400Z",
		"id": "dsYVGG9Mvw"
		}
			{
			"id": "dsOB0yra",
			"status": "SUBMITTED",
			"startTime": null,
			"endTime": null,
			"error": null,
			"tasks": []
			}

Notebook and Paragraph IDs

To obtain notebook and paragraph IDs, select a paragraph, using the Setting Context menu. Click **Open as Iframe**. A new browser tab opens. Copy the url in the address bar in browser and borrow IDs from the URL notebookId=dsYVGG9Mvw&pid=dsB0zM58

https://<host>/datastudio/?root=iParagraph&notebookId=dsYVGG9Mvw&pid=dsB0zM58&readOnly=false&showCode=true

Figure 19–20 Notebook and Paragraph IDs



Considerations

Consider the following:

- Make sure there are no empty paragraphs as this results in an 500 error response.
- To test, create a table and view records that are inserted in the table.

```
%oracle
```

```
insert into iw_test (first_name, last_name ) values (John, 'Doe')
%python
print('Hello World')
```

Verify using the following:

```
CREATE TABLE iw_test (
    person_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    PRIMARY KEY(person_id)
);
select count(*) from iw_test;
```

REST API

Here is a REST API curl call example showing how to schedule a notebook.

Table 19–2 Example

REST API purpose	REST API curl request	REST API curl response
Fetch Authentication Token	curl --location --request POST 'http://bur00afq.us.oracle.com:26000/datastudio/v2/sessions/login' \ --header 'Authorization: Basic b3Jhc2UxOnBhc3N3b3JkMQ==' \ --header 'Content-Type: application/json' \ --header 'Cookie: JSESSIONID=cmWRE8eaMH1e7ClA72Qx8QVxX3SKVMuEi1S_ErDgySh2qInJRbdZ!1637160649' \ --data-raw '{"credentials":"password1","principal":"orase1"}'	Response will have authentication token. To make subsequent request: authToken":"de6460f5-e36c-4592-b2c3-6ef18268e6c5"

Schedule Notebook	curl -X POST -H 'Content-Type: application/json' -H 'x-auth-token: de6460f5-e36c-4592-b2c3-6ef18268e6c5' -i 'https://<host>:<port>/datastudio/v2/notebooks/schedule' --data '{
	"cronSchedule": "0/5 * * * * *",
	"endDate": "2020-02-24T21:10:34.400Z",
	"id": "dsZgvK3G"
	}
		{"id":"dsbBVLgL","status":"SUBMITTED","startTime":null,"endTime":null,"error":null,"tasks":[]}

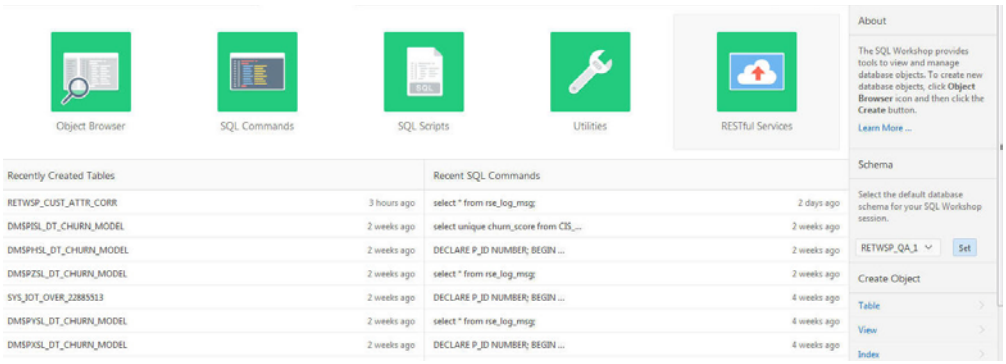
Restful Service

RESTful Services allow for the declarative specification of RESTful access to the database. They are created by configuring a set of Uniform Resource Identifiers (URIs) to a SQL query or anonymous PL/SQL block. The set of URIs is identified by a URI template.

To create a RESTful service:

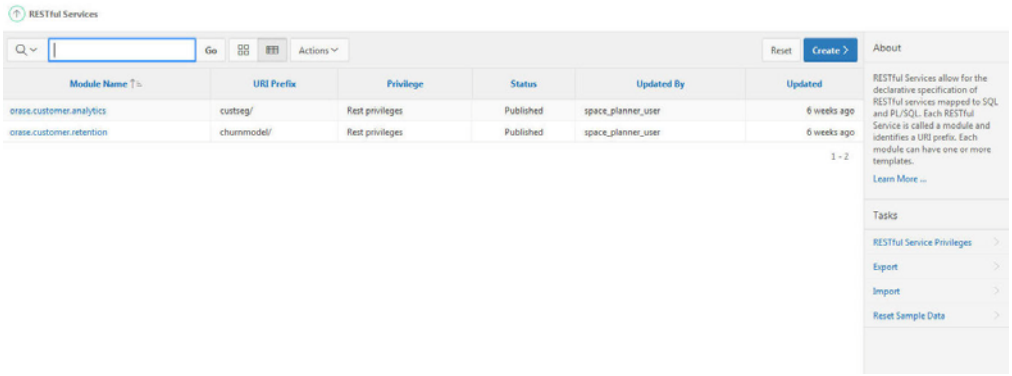
1. Select SQL Workshop RESTful Services.

Figure 19–21 RESTful Service



2. On selecting RESTful Services, you see the option to create a RESTful Service.

Figure 19–22 Create RESTful Service



3. A RESTful Service Module is a grouping of common templates, known as Resource Templates, under a common URI prefix. This prefix is prepended to all templates.

Figure 19–23 Create Module

4. A URI template is a simple syntax for describing URIs. You populate the required fields as shown in Figure 19–24 and make sure to set the required privileges as Rest Privileges.

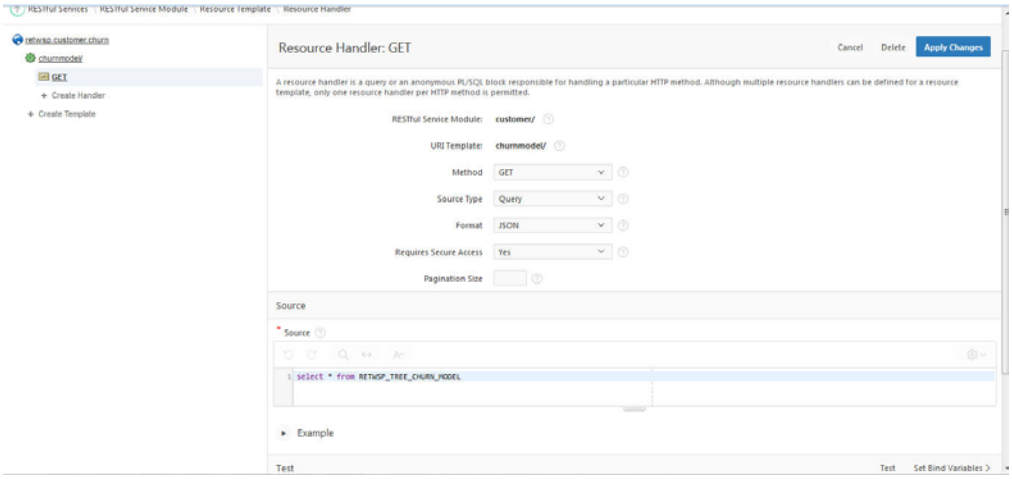
Figure 19–24 Resource Template

5. A Resource Handler is a query or an anonymous PL/SQL block responsible for handling a particular HTTP method. Multiple handlers can be defined for a Resource Template; however, only one handler per HTTP method is permitted. You can select method, source type, format, and SQL Query to read data from the schema.

Figure 19–25 Resource Handler

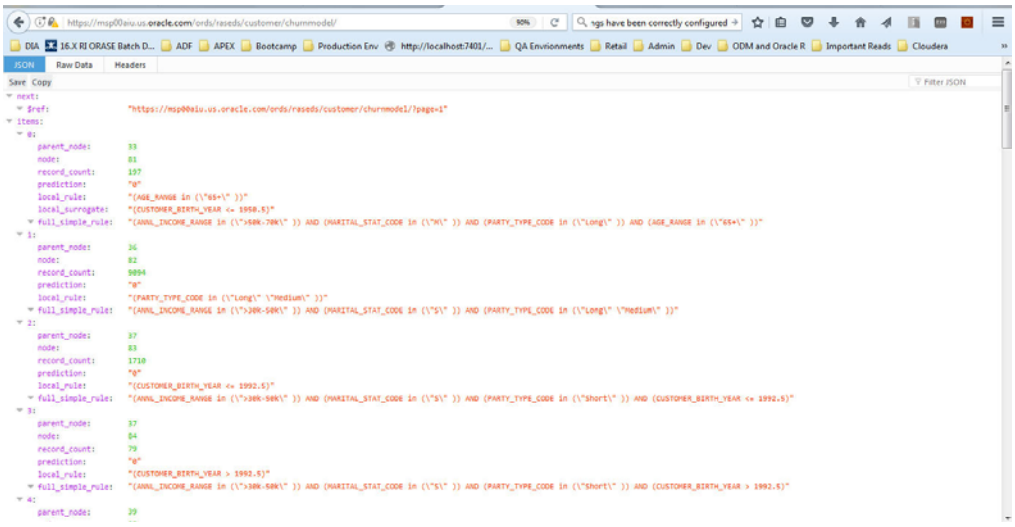
6. To test the rest API, click **Test Button** (see Test at the bottom right corner.)

Figure 19–26 Test the REST API



7. The page refreshes and displays data in JSON format.

Figure 19–27 RESTful Refresh



Integration

IW provides a customer with a way to integrate with other applications. A framework is provided that enables it to receive and read-in files from customers so that the data contained can be loaded into IW tables. They are thus able to use data from their other systems within IW itself.

Consequently, data within IW can also be exported to flat files so that their other applications can use the AIF-cleansed data that is available via IW.

More details on the framework for these capabilities are contained in the following Custom Data Loads and Custom Data Exports sections.

Custom Data Loads

AIF provides functionality to load data files from a customer site sent via Secure FTP, from which all other daily interface files are sent. A zip file, which can contain any number of files, can be provided, along with the specifications about the files to be loaded. (These files are referred to as context files here.) Once the data has been sent, a process is invoked from the Process Orchestration and Monitoring (POM) application to initiate loading the data into the appropriate tables inside the Innovation Workbench. Once the data load is complete, a zip will be sent to the outbound SFTP server to provide the data load logs, which will indicate whether or not there were any rejected records.

Required Files

The following are the prerequisites for performing a data load:

- The table must exist inside the Innovation Workbench.
- A zip file named ORASE_IW_DATA.zip must be created. It must contain the data to be loaded and the context files that describe details about the data load.
- After ORASE_IW_DATA.zip has been sent to the SFTP server, then a ORASE_IW_DATA.zip.complete file must be sent as well. This triggers the receipt of the data file by the application server. These two files must both be provided. The .zip must be sent first, followed by the .zip.complete file. Note that these filenames are case sensitive.

Zip File Contents

The ZIP file must contain a pair of files for each table to be loaded, a .dat file and a .ctx file.The data file can have any name, with the .dat extension. The context file must match the name of the data file, but with the .ctx extension. If a data file is provided without a .ctx file, then the data file will be ignored.

For example, to load a table called STG_WKLY_SLS, you require a file called STG_WKLY_SLS.dat to hold the data to be loaded and a file called STG_WKLY_SLS.dat.ctx to describe the details regarding the load.

No support exists for directory names inside the zip file, so if they are provided they will be ignored. For example, you cannot send directory1/STG_WKLY_SLS.dat and directory2/STG_WKLY_SLS.dat because they will end up overriding each other when the zip is extracted.

Although you can provide multiple files to be loaded in a single zip, none of them can be for the same table as another load. The load truncates the data in the table to be loaded, so if the same table is targeted by multiple files, then only the latest file will be loaded. Because this data is truncated prior to each load, you should follow a pattern in which these tables are the staging tables to be used to then populate data in other tables. In this way, the data is verified, which allow for the merging of data, via updates or inserts. This is not be possible if you loaded directly to your final table.

Context File Details

The .ctx file provide options for including details that describe the data load. These details are similar to the options used by SQL*Loader, so refer to documentation about the SQL*Loader for any additional details about concepts noted here.

Here is an example of a context file.

```
#TABLE#SLS_WKLY_SLS#
#DELIMITER#|#
#COLUMN#WEEK_DATE#DATE(10) "YYYY-MM-DD"#
#COLUMN#PRODUCT_KEY#
#COLUMN#LOCATION_KEY_FILLER#BOUNDFILLER#
#COLUMN#SLS_AMT#
#COLUMN#LOCATION_ID#"some_db_package.lookup_location_id(:LOCATION_KEY_FILLER)"#
#COLUMN#LOAD_DATE#"sysdate"#
```

In the above example contextual file:

- The #TABLE## line is required in order to specify the name of the table to be loaded.
- The #DELIMITER## line is optional, and if not provided, then a | will be considered the default value.
- The table to be loaded will be loaded via a Direct Path load (for efficiency sake). It will truncate the table prior to loading the table. This means that you cannot provide multiple files in a single transmission for the same table.
- The columns can be wrapped in " " if necessary.
- The data in the file must be provided as CHARACTERSET AL32UTF8.
- For the column specifications, the columns must be listed in the order in which they appear in the file.The format is #COLUMN#THE_COLUMN_NAME#Any special load instructions#
- Note that for Date columns, you must provide a proper date specification with the format that you are providing the data in.The example above for WEEK_DATE illustrates how to specify the date in a YYYY-MM-DD format.
- For numeric columns, you do not normally require any additional load instructions.

- If you are providing data that must be used as a BOUND FILLER during the load of another column, then you must specify BOUNDFILLER, just like you would in SQL*Loader.
- The LOCATION_ID example above illustrates how you can refer to a FILLER column, which is then used as a parameter to a function, which then returns the actual value to load to the column.
- If you want an value such as the system date/time to be loaded, then you can specify "sysdate" as shown above for LOAD_DATE column.
- When you are populating columns that are loaded via a special expression (such as LOCATION_ID and LOAD_DATE above), be sure to provide them last in the context file.
- When loading data character data, it maybe be necessary to specify the column like this: CHAR(50). This is commonly required by SQL*Loader when advanced character sets are used.
- The examples shown above for LOCATION_KEY_FILLER and LOCATION_ID are not usual/simple use cases, but are supported. For a better understanding of how that works, refer to the documentation for SQL*Loader.

Data Load Feedback

When the data load is complete, a zip file named ORASE_IW_DATA_extract.zip will be pushed to the SFTP server and it will contain the log and bad files for the data load.

Invoking the Data Load

In order to invoke the load process, you must use the POM application to invoke the following adhoc process:

RSE_LOAD_IW_FILES_ADHOC.

This controls the execution of all these steps. Once the load has been completed, you should be able to use the data as necessary inside the Innovation Workbench. See "Process Orchestration and Monitoring" for additional details.

Custom Data Exports

You can export data that has been created within the Innovation Workbench workspace. You first configure the table to be exported. A process is then executed that exports the table data to a text file.The data is then gathered into a zip file, and the zip file is moved to the SFTP server for retrieval.

Required Configuration

Using the Data Management menu option (described in *Oracle Retail Science Cloud Services User Guide*), you can manage the tables to be exported, using the formatting shown in Table 19–3. Once in the screen, select the table RSE_CUSTOM_EXP_CFG. Then, add rows and make adjustments to existing rows as required in order to define the tables to be exported.

Table 19–3 Export Table Formatting

Column Name	Data Type	Example	Comments
TABLE_NAME	Character(30)	IW_RESULTS_TABLE	The name of the table to be exported.
FILE_NAME	Character(80)	iw_results_table.csv	The name of the file to be created, excluding any directory names.
DESCR	Character(255)	Results table of XYZ Calculation	Any description to describe the table.
COL_DELIMITER	Character(1)	, (comma)	The character to use as a delimiter between columns.
COL_HEADING_FLG	Character(1)	Y	A Y/N value to indicate if the export should include a heading row (Y) that contains the names of the columns, or not (N).
DATE_FORMAT	Character(30)	yyyy-MM-dd HH:mm:ss	The format to use for any date columns. This format must be a valid format for Java date handling.
ENABLE_FLG	Character(1)	Y	A flag to indicate if this table should be exported (Y) or not (N) This flag can be used to temporarily disable an export, without the need to remove it completely.

Invoking the Export

In order to begin the export process, you must use the POM application to invoke the following AdHoc process:

RSE_IW_EXPORT_FILES_ADHOC / job: RSE_IW_EXPORT_FILES_ADHOC_JOB.

This process controls the execution of the steps to export the data. Once the export has been completed, a zip file named ORASE_IW_EXPORT_extract.zip is created. All files will be named according to the names specified in the RSE_CUSTOM_EXP_CFG table. See "Process Orchestration and Monitoring" for additional details on how to execute a Standalone/AdHoc Sample Extensibility Use Caseprocess.

Sample Extensibility Use Case

Clients can use IW's framework to address use-cases unique to their business requirements.

Here is one possible example of IW's extensibility.

A customer wants to expose AIF-cleansed sales transaction data from AI Foundation to a third-party system, using an integration software for connecting applications, data, and devices. This enables their other applications to access the exact calculations/AIF-cleansed data that they require.

IW can be extended as outlined below in order to address this use-case:

- Load sales data into AI Foundation using the existing nightly sales data load routine.
- Create a custom PL/SQL package to transform the data as required and load the transformed data into custom tables in IW. For example, sales can be aggregated per customer/transaction. This aggregated data is loaded into the custom IW table, with specific custom keys required to identify the records in external systems.

This custom PL/SQL can include status tracking using auxiliary tables to capture the status of the executed routine, as well as log pertinent information that can enable re-execution if necessary.

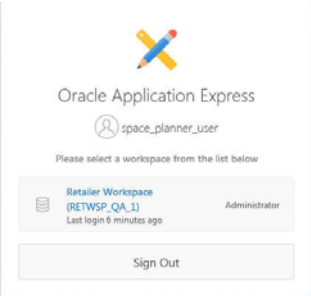
- Create ReST APIs in APEX that reference the custom IW tables, to expose the sales transaction data.
- Create and schedule load jobs in APEX that invoke the custom PL/SQL package to populate the custom tables. These daily programs must be scheduled using DBMS_SCHEDULER jobs that are aligned with the expected end-time of the daily Retail Insights batch.
- Real-time ReST API calls can then be made from the integration software and then into the third-party system that will use the data.

Troubleshooting

Science Innovation Workbench can also be used for monitoring application logs to troubleshoot issues in an application process due to an error in a batch or business process.

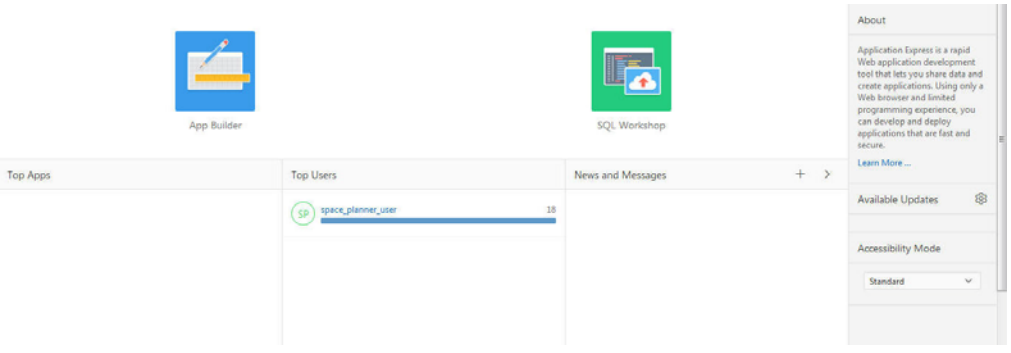
1. Click **Innovation Workbench**.
2. Select **Retailer Workspace**.

Figure 19–28 Retailer Workspace



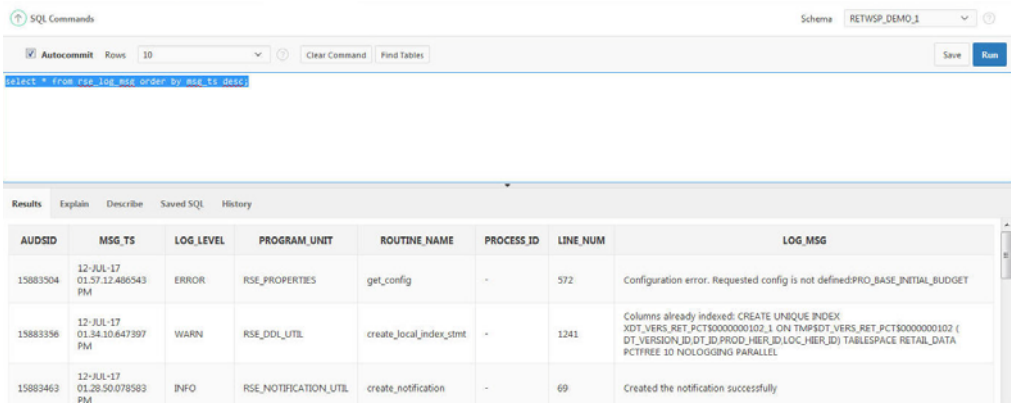
3. Select **SQL Workshop**.

Figure 19–29 SQL Workspace



4. Select **SQL Command**.
   a. Enter SQL select * from rse_log_msg order by msg_ts desc;
   b. Select run and view the log message in the bottom panel.
   c. You can describe objects using describe rse_log_msg;

Figure 19–30 SQL Command



DBMS Scheduler

Innovation Workbench can be used for data mining tasks that are long running and have to rely on database scheduling functions and procedures that can be called from any PL/SQL program.

Any jobs created in the Retailer Workspace using Innovation Workbench must be created under Job Class RETAILER_WORKSPACE_JOBS. Jobs created in a default job class or any other job class other than RETAILER_WORKSPACE_JOBS can be disabled by an Oracle Administrator while managing resources.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name           =>  'retwsp_churn_model',
   job_type           =>  'STORED_PROCEDURE',
   job_action         =>  'retwsp.pkg_customer_analytics.proc_churn_model',
   start_date         =>  '01-JAN-17 07.00.00 PM US/Pacific',
   repeat_interval    =>  'FREQ=YEARLY; BYDATE=0331,0630,0930,1231; ',
   end_date           =>  '31-DEC-17 07.00.00 PM US/Pacific',
   job_class          =>  'RETAILER_WORKSPACE_JOBS',
   comments           =>  'Retailer workspace churn model job');
END;
/
```

Schema Objects

Database objects owned by the various schemas in AI Foundation are available for the advanced analyst to use. Here are some examples:

Table 19–4 Schema Objects

Table Name	Description
rse_cal_hier	This table is used to hold all calendar hierarchies. Examples are the normal calendar hierarchy, and can also contain an alternate hierarchy for the fiscal calendar.
rse_prod_hier	This table is used to hold all product hierarchies. Examples are the normal product hierarchy, and can also contain an alternate category hierarchy.
rse_loc_hier	This table is used to hold all location hierarchies. Examples are the normal organizational hierarchy, and can also contain an alternate hierarchy for trade areas.

Table 19–4 (Cont.) Schema Objects

Table Name	Description
rse_prod_loc_status	Status of the item at this location for this time frame. A-Active; I-Inactive; C-Discontinued; D-Deleted
rse_ret_lc_wk_a	This table contains aggregate sales data for the dimensions of a location and a week.
rse_sls_lc_wk_a	This table contains aggregate sales data for the dimensions of a location and a week.
rse_sls_pr_lc_cs_wk	This table contains aggregate sales data for a Product, Location, Customer Segment and Week. The SLS_PR columns represent the metrics for that week that were on promotion, while the other metrics represent the sales metrics while the item was not on promotion.
rse_sls_pr_wk_a	This table contains aggregate sales data for the dimensions of a product and a week.
rse_sls_ph_lc_wk_a	This table contains aggregate sales transaction data for different product hierarchy/levels, at the store location/week dimension.
rse_sls_pr_lc_wk	This table contains aggregate sales data for a Product, Location, and Week. The SLS_PR columns represent the metrics for that week that were on promotion, while the other metrics represent the sales metrics while the item was not on promotion.
rse_sls_txn	This table contains sales transaction data.
rse_prod_attr	This is the table that holds product attributes.
rse_prod_attr_grp	This is the table used to load the associations of CM Groups to product attributes.
rse_prod_attr_grp_value	This is the table used to load the associations of CM Groups to product attributes and its values
rse_prod_attr_grp_value_map	This is the table used to load the associations of CM Groups to product attributes, group values and actual product attribute values
rse_like_loc	This is the table used to load the like stores for CMGroup or Category.
rse_hier_level	This table defines the various levels for all the hierarchies.
rse_hier_type	This table defines the available hierarchies for use within the RSE applications.
rse_fake_cust	Table for specifying customers who are considered as fake customers. A fake customer is a customer who purchases too many transactions to be considered a single customer. Examples are generic store cards.
rse_loc_src_xref	This table contains integration ID information that enables interaction with other systems, using IDs that other systems can accommodate for the Location Hierarchy.
rse_prod_src_xref	This table contains integration ID information that enables interaction with other systems, using IDs that other systems can accommodate for the Product Hierarchy.
rse_log_msg	This table contains messages logged while database, batch or business processing.
w_party_per_d	This table contains customer data and its attribute.
cis_cust_attr_vw	Customer Attributes - This view provides basic customer attributes.
cis_cust_trans_attr_vw	Customer Transaction Attributes - This view provides attributes for customer transactions.
cis_cust_trans_ph_attr_vw	Customer Transaction Attributes - This view provides product attributes for customer transactions.

Table 19–4 (Cont.) Schema Objects

Table Name	Description
cis_custseg_attr_exp_vw	This view provides an export of the attributes that define a segment.
cis_custseg_cat_attr_exp_vw	This view provides an export of the product attributes that define a segment.
cis_custseg_cust_exp_vw	This view provides the members for an exportable set of segments.
cis_custseg_exp_vw	This view provides an exportable set of clusters for customer segmentation.
cis_sls_cust_cal_a	This table contains aggregate customer sales data for a configurable level of the calendar hierarchy. The table is to be partitioned by Calendar and is also be suitable for sub partitioning by Customer using a Hash Partition strategy, so that subsequent uses can operate within the confines of a given Hash partition.
cis_sls_cust_ph_cal_a	This table contains aggregate customer sales data for a configurable level of the calendar hierarchy, for a selected set of product hierarchies. The table is to be partitioned by Calendar and is also be suitable for sub-partitioning by Customer using a Hash Partition strategy, so that subsequent uses can operate within the confines of a given Hash partition.
cis_sls_ph_a	This table contains aggregate sales data for all product hierarchy members of a configured hierarchy type and level. This can be used to identify the Top Categories for use by things like Customer Segmentation.
cis_cluster_set_exp_vw	This view provides an exportable set of clusters to send to Cat Man.
cis_store_cluster_exp_vw	This view provides an exportable set of clusters for stores.
cis_store_cluster_mem_exp_vw	This view provides the members with an exportable set of segments.
cis_store_cluster_prop_exp_vw	This view provides an exportable set of clusters for stores.
cis_cluster_sls_ph_lc_a	This table contains calendar level aggregates for the various clusters. The table is to be partitioned by Calendar.
cis_cluster_summ_level_attr	These are metrics generated at cluster/attribute/business object level. There are metrics that are generated at that level such as centroid.
cis_prod_attr_loc_share	This table contains aggregate sales data for product attribute values as well as the share that these values are with respect to the total sales for the product hierarchy, calendar, location. The share value is a configurable value, which can either be based on sales units, sales amount or profit amount.

20

Process Orchestration and Monitoring

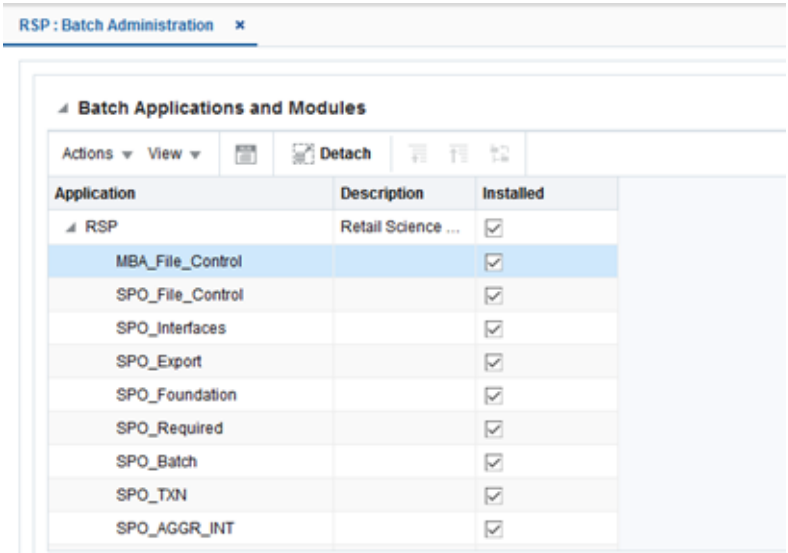
This chapter describes the basic workflow that triggers batch processes from Process Orchestration and Monitoring (POAM), including monitoring and error handling.

Batch Administration

Using the Batch Administration functionality, shown in Figure 20–1, the user can enable and disable application-specific batch processes as a whole or enable and disable individual batch processes at the job level. Any modifications made will become effective with the next scheduler day load (except batch throttling configurations). The execution of application batch processes can be controlled by selecting or deselecting the check box.

To access Batch Administration, select: Schedule > Administration > Batch Administration.

Figure 20–1 Batch Administration



Batch Monitoring

Using Batch Monitoring, shown in Figure 20–2, the user can monitor the current status of different batch processes at cycle level including nightly, ad hoc, and hourly. This screen displays a pie chart section that provides a summary view of the current cycle execution. A job can have eight different job statuses: Pending, Completed, Hold, Skipped, Running, Failed, Disabled, and System Held.

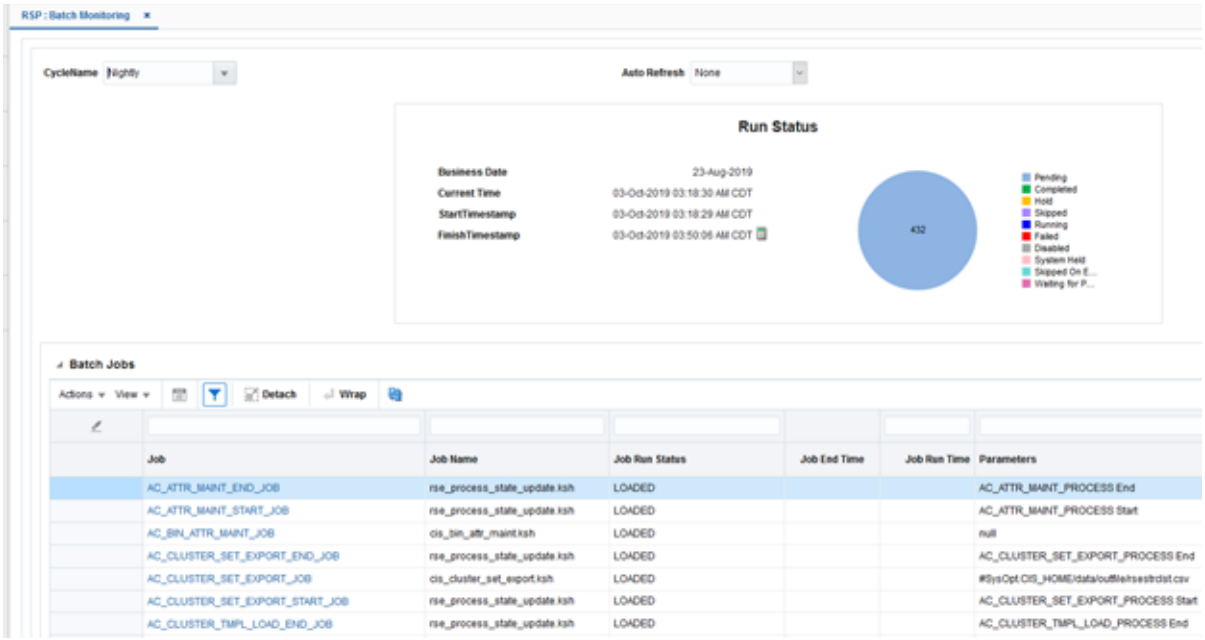
To access Batch Monitoring, select: Schedule > Monitoring > Batch Monitoring

Monitoring a Batch Cycle

To create a new scheduler day.

1. Access Batch Monitoring.
2. Click **Create Schedule**, then click **Yes** to confirm. A new scheduler day will be created.
3. From the Monitoring Cycle list box, select the Nightly/Ad hoc/Hourly cycle. Note that the individual nightly batch job with its run status is loaded.
4. A pie chart displays the summary view of the current cycle execution.

Figure 20–2 Batch Monitoring



Triggering Process from POAM

Running a Nightly Batch Process

To run a nightly batch process, complete the following steps:

1. Log into POAM using a valid user name and password.
2. Navigate to Schedules > RSP > Monitoring > Batch Monitoring.
3. From the CycleName drop-down list, select **Nightly**.
4. From the Auto Refresh drop-down list, select **Every 30 seconds**.
5. From Postman, do the following:
   a. Post the url and the process to be started in the following format, replacing POAM_URL with the correct value.

      POAM_URL/ProcessServices/services/private/executionEngine/schedules/RSP/execution

      For example:

http://<pom-server-url>.us.oracle.com/ProcessServices/services/private/executionEngine/schedules/RSP/execution.

b. In the Authorization section, provide the correct user name and password for the environment.

c. In the body, provide details for Cycle name, Flow name, and Request type. For example:

```
{
"cycleName" : "Nightly",
"flowName" : "Nightly",
"requestType" : "External Request"
}
```

d. Click **Post**. The body indicates that the execution has started. The process begins executing in POAM.

Monitoring the Process Executions

After triggering the process from Postman, the user can view the progress of the batch execution and the state of the batch process (either pass or fail) in the POAM UI. In the POAM UI, open the Batch Monitoring screen and select **Nightly** in the drop-down menu. Refresh the scheduler day you created. The batch schedule status can be viewed in the displayed graph; if Auto Refresh is set, then the graph is automatically updated.

The Job status is listed in the Batch Jobs tab.

Note the following:

- The nightly batch schedule, once executed successfully, opens the next day automatically.
- If any batch failures have occurred, the next day will not open automatically. In such cases, the failure must be resolved, the existing scheduler day must be closed, and the new scheduler day must be opened manually.
- RI and AIF must always be on same business dates. Triggering RI will automatically start the AIF batch processes. (Once RI completes, it starts AIF.)

Running AdHoc Batch Processes

In the RSE_CONFIG table, the parameter ADHOC_BATCH_ENABLED is disabled by default. To execute AdHoc batch processes, you must change the value of this parameter to **Y**.

Enabling AdHoc

To enable AdHoc,, complete the following steps:

1. Log into the POAM using a valid user name and password.
2. Navigate to Schedules > RSP > Monitoring > Batch Monitoring.
3. From the CycleName drop-down list, select **Adhoc**.
4. From the Auto Refresh drop-down list, select **Every 30 seconds**.
5. In Batch Jobs table, search for the job RSE_BATCH_ENABLE_ADHOC.
6. Select the job and click **Actions > Run**.

   Once RSE_BATCH_ENABLE_ADHOC is run, it will disable the checks that determine if it is a daily, weekly, or quarterly batch process. It will cause a new script added to the start of the daily batch to fail. It will then provide a warning message indicating that Adhoc is enabled and therefore the daily batch process is disabled.

7. Once the enabled AdHoc batch process passes the other AdHoc batch processes, the batch processes can be executed.
8. To execute the remaining Adhoc batch processes from Postman, complete the following steps:
   a. Post the url and the process to be started in the following format, replacing POAM_ URL with the correct value.

      POAM_ URL/ProcessServices/services/private/executionEngine/schedules/RSP/execution

      For example:

      http://<pom-server-url>.us.oracle.com/ ProcessServices/services/private/executionEngine/schedules/RSP/execution.

   b. In the Authorization section, provide the correct user name and password for the environment.
   c. In the body, provide details for Cycle name, Flow name, and Request type. For example:

```
{
"cycleName" : "Adhoc",
"flowName" : "Adhoc",
"requestType" : "External Request",
"processName": " RSE_SLS_TXN_ADHOC ",
"requestParameters": " jobParams.RSE_SLS_TXN_SETUP_ADHOC_JOB=-s20180101
-e20190101 -f Y "
}
```

      Note that he requestParameters can vary from one Adhoc Job to another. A spreadsheet has been created to show the list of Adhoc Processes, along with examples on the parameters they support. (SEE ORASE-adhoc-191.xlsx).

Monitoring Process

After triggering the process from Postman, the user can view the progress of the batch execution and the state of the batch process (either pass or fail) in the POAM UI. In the POAM UI, open the Batch Monitoring screen and select **Adhoc** in the drop-down menu. Refresh the scheduler day you created. The batch schedule status can be viewed in the displayed graph; if Auto Refresh is set, then the graph is automatically updated.

Disabling AdHoc

When Adhoc is enabled, the nightly batch processes are not permitted to run. Once Adhoc is no longer required, it must be disabled.

To disable the Adhoc batch process, complete the following steps:

1. In the Batch Jobs table, search for the job RSE_BATCH_DISABLE_ADHOC.
2. In the Auto Refresh drop-down menu, select **Every 30 seconds**.
3. Select that job and click **Actions > Run**.

Intraday Cycle

To create the intraday cycle:

1. Log into POAM using a valid user name and password.
2. Navigate to Schedules > RSP > Monitoring > Batch Monitoring.

3. In the CycleName drop-down list, select the required cycle (for example,Hourly_Cycle_1).
4. From Postman, complete the following steps:
   a. Post the url and the process to be started in the following format, replacing POAM_URL with the correct value.

      POAM_URL/ProcessServices/services/private/executionEngine/schedules/RSP/execution

      For example:

      http://<pom-server-url>.us.oracle.com/ProcessServices/services/private/executionEngine/schedules/RSP/execution.

   b. In the Authorization section, provide the correct user name and password for the environment.
   c. In the body, provide details for Cycle name, Flow name, and Request type. For example:

```
{
"cycleName"      :"Hourly_Cycle_1",
"flowName"       :"ASO_INTRADAY_CYCLE",
"requestType" : "External Request"
}
```

   d. Click **Post**. The body indicates that the execution has started. The process begins executing in POAM.

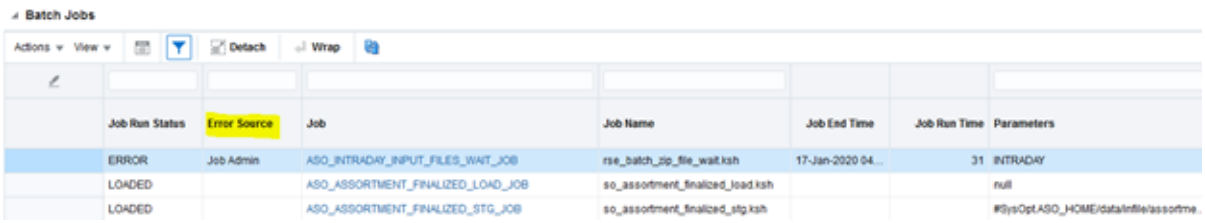
Error Handling

A batch process can fail because of the following:

- An actual batch process failure.
- A failure as a result of the loss of communication between POAM JOS components.
- 'The Error Source column lists the source of error. If the source is POAM, it is generally a connection issue.

If the source of the error is Job Admin, it can be either a connection failure or an actual job failure. This can be determined by examining the logs for JOS JOB ADMIN.

Figure 20–3 Error Handling



JOS JOB ADMIN

1. .Log into JOS JOB ADMIN using a valid user name and password.
2. Navigate to System Logs.
3. Select the name of the JOB that failed. The log will be displayed at the bottom of the screen.

Skipping a Failed Job

If any job has failed for a known reason, the user can bypass it. In this way, the schedule releases the dependency and can continue executing other jobs.

To skip a failed job, complete the following steps:

1. Navigate to POAM > Batch Monitoring.
2. Select the job row that you want to skip.
3. Click **Skip or Release Skip** from More Actions.
4. Add any appropriate comments.
5. The status of the job will change to SKIPPED if the action is SKIP or to LOADED if the action is Release Skip.
6. Note that a job that already has a status of Skipped can only be released.

21

FAQs

This chapter contains frequently asked questions and their answers.

General Questions

Here are some answers to typical questions of a general nature.

What delimiters are used in the data files?

The inbound data files must be pipe-delimited, while the outbound files are comma-delimited. The delimiters are not configurable.

When are W_RTL_RECLASS_DP_GP_TMP and W_RTL_RECLASS_IT_SC_CL_ TMP interfaces used?

These are generally used for implementations that also use RI's data warehouse reporting.

If CREATED_BY_ID and CHANGED_BY_ID columns in every table are set to -1, will it have negative impacts on the functionality?

The value of -1 is the normally expected/used value. The value used will not affect functionality.

CREATED_ON_DT and CHANGED_ON_DT and SRC_EFF_FROM_DT are given values for every table, but since the value is not used functionally, can it be set to a value of NOW?

The only interface where this approach is not appropriate is W_RTL_PROMO_DS. For this interface, the SRC_EFF dates must indicate when the promotion was in effect. (If sales transaction data is not provided, W_RTL_PROMO_DS.dat interface becomes pointless, and therefore defaulting to NOW will be fine everywhere else.)

DT Questions

Here are some answers to typical questions related to demand transference.

Demand transference can make use of similarities derived from customer-linked transactions data. What is the advantage of using this type of similarity, instead of attribute-based similarities?

The benefit to using customer-linked transactions is that the similarities are not dependent on good construction of attributes and attribute values; in addition, the similarities will agree with the CDT if the customer has also implemented CDT. However, keep in mind that to handle the case of new items, attributes are required anyway, so even when using transaction-based similarities, it is still necessary to construct attributes.

What are the disadvantages of using similarities derived from customer-linked transactions data?

Two key disadvantages are:

Retailers want the similarities to be intuitive to their business users, and attribute-based similarities are much better for this because they are more transparent, particularly when the attributes provided by the business users themselves are used.

The requirement for frequent repeat purchases is a high requirement for a category to meet. Many grocery and drug store categories can meet this, but even at grocery or drug stores, there are categories, such as small electronics, that cannot meet this requirement. There are also categories such as batteries or toothbrushes that do have repeat purchases, but the length between purchases is quite long. For such categories, it is probably better to use attribute-based similarities.

Is it possible to use transactions-based similarities for categories where repeat purchases occur but are very infrequent?

It is recommended that you use attributes for such categories, but if there is sufficient data for the category, it is still possible to use transactions-based similarities. You must have either (or both):

- A longer period of history for the category (for example, three years instead of one year) so that there is more chance of catching repeat purchases. Note that this does not necessarily mean that the actual data volume is higher. For example, for batteries, the data volume for three years might be equal to the data volume for six months of dairy, since battery sales have a much lower rate than dairy.
- A large number of customers (at least 100,000) for each segment of the category, if you are implementing segments, or just a large number of customers (at least 100,000) for the entire category, if you are not implementing segments. Having such a large number of customers increases the probability that a significant fraction of them will have made repeat purchases in the category in your historical data, even if each customer did not make many repeat purchases. For example, you could have 100,000 customers, each of whom made only one repeat purchase.

How can grouped attributes be used in the AIF platform?

Grouped attributes are recommended only for CDT, not DT. For CDT, the guideline is that attributes must have no more than seven values. Grouping is a way to achieve this goal, as it groups together many attribute values into a single value. Having attributes with a large number of attribute values can result in very large CDTs, especially if the attribute occurs at the top of the tree. This general guideline of seven values is a way to keep CDTs to a reasonable size. Note that retailers seem to prefer having large CDTs with ungrouped attributes.

What attributes are typically good candidates for grouping?

The most common attributes are Brand, Flavor, and Color, because these attributes typically have a large number (that is, more than 50) of values. Keep in mind, that retailers typically do not want grouped attributes.

How can the Brand attribute be grouped?

Brands can, for example, be grouped into High End, Mainstream, and Budget. The grouping must be driven by how the retailer thinks of the category and thinks of brands in the category.

Does the software automatically determine which attributes are functional fit? Can the software help me determine which attributes are functional fit ones?

There is no way to automatically detect functional-fit attributes, or close-to-functional fit attributes. The determination is done today using business users' intuition, not science.

Are there other uses of functional-fit attributes besides the obvious examples like size in wiper blades and clothing? What about functional-fit attributes for food items?

Functional-fit attributes are useful for items such as food for expressing strong customer preferences. For example, caffeinated can be set as a functional-fit attribute for the Coffee category to eliminate transference between caffeinated and de-caffeinated coffee. While it is possible that a small minority of people might transfer purchases of one to the other, it is unlikely enough that, when planning an assortment, transference should simply be eliminated. Note that this is based on the business user's intuition, since there is no automatic way to detect these situations.

What are the special considerations for categories containing very few items, such as 20 or fewer?

Such categories are very rare; typically, categories have hundreds to thousands of items. Small categories may cause a problem in determining demand transference because their historical data may contain very few assortment changes. The more items a category has, the more likely assortment changes will have occurred; conversely, small categories may have very few historical assortment changes, which makes determining transference in the category difficult. One solution is to merge the small category into a bigger category that has similar items. For example, merge a specialty bacon category containing 20 items with the regular bacon category (since both are bacon). This amounts to applying the assortment elasticity of the larger bacon category to the specialty bacon, and if the specialty bacon by itself has very few assortment changes, this is a reasonable solution.

How is new-item introduction handled by demand transference? Which cases of new items are handled?

The case the application handles is introducing new items at an existing store for an existing category that already has attributes. For this case, all that is required for the new item are its attribute values. No sales history is required for the new item. From the attribute values assigned to the new item, the similarity of the new item to existing items is calculated. This similarity is used to forecast a base rate of sale for the new item based on the sales rates of existing similar items. This base rate of sale is then used in demand-transference calculations.

How is W_RTL_PRODUCT_BRAND_DS interface used by DT?

Brand can be provided as an ITEMUDA, or it can be provided via W_RTL_ITEM_GRP1_DS with a PROD_GRP_TYPE=BRAND. If it is done as a PROD_GRP_TYPE=BRAND, then this interface receives the master list of Brands, which the W_RTL_ITEM_GRP1_DS refers to. If Brand is just provided as an ITEMUDA, then there is no need for this file.

How is W_RTL_PRODUCT_COLOR_DS interface used by DT?

In the same way as the W_RTL_PRODUCT_BRAND_DS interface, only the PROD_GRP_TYPE is COLOR in this case.

ASO Questions

Here are some answers to typical questions related to ASO.

An item cannot meet the service level with the defined facings. Should the solver give more facings to meet the minimum service level restriction?

Not necessarily. This depends on the feed from the sales and inventory model, which indicates what the possible service level is for a given number of facings. If it indicates that adding any number of facings will not necessarily help, then the solver drops that item if it is not mandatory or returns no solution if it is mandatory. It would be worthwhile for the user to review the replenishment data provided for the item.

Why are empty spaces present in the planogram?

The goal of the optimization is never to fill up the entire space. Empty spaces can arise for a variety of reasons. This can occur if more than sufficient space is available for all the products. In this case, the solver places the products in more than one elevation and location. Alternatively, many constraints may be in play at the same time and one of them can cause some products to be dropped.

The solution generated does not seem optimal, since I can generate a better solution. Why?

Such a situation does not arise normally. It can sometimes happen because the optimization searches for the best possible solution within an allocated time. The allocated time can vary, depending on how many jobs are running and how much processing power is available for this job. To negate such uncertainty, user can increase the time allocated for the optimization from the default value of 180 seconds.

Why are items not sorted across the shelves?

The sorting functionality sorts items within a particular shelf. It does not sort across shelves as this negates the optimal placement determined by the optimization and can sometimes result in infeasibility. For these reasons, sorting is not allowed across shelves. The user is advised to examine the visual guidelines to obtain the desired structure.

Why are the horizontal blocks arranged in a zigzag pattern instead of being truly horizontal?

This can arise for the shelf fixture type. If the bays are actually identical, then the optimization can generate horizontal blocks. However, most often, the shelves are not aligned perfectly across bays and so the optimization is asked to look for almost horizontal blocks. In the case of the pegboard fixture type, the optimization can generate actual horizontal blocks, as it is not limited by shelf elevations.

Why is an assortment not ready for optimization?

An assortment may stop at the pre-optimization stage and not be ready for optimization because errors were found either in the assortment data during data loading, during the checking of integration with other data sets (such as missing replenishment, display style, and so on), or during the mapping process and requiring user review. Mapping errors can be examined in the Assortment Mapping UI. In addition, assortments that are ready for SO may not be available for optimization if the POG set to which they are mapped has multiple assortments mapped to it and at least one of those assortments is not ready for optimization due to mapping errors. In such cases, the POG set is meant to optimize multiple assortments; having errors in at least one of them prevents the whole POG set from becoming available for optimization.

Can I copy an old run from an assortment for the same category rather than a new assortment and use it against the new assortment (most of the products/stores should be the same)?

No, a run's scope is limited to the POG set and specific assortments used in that run. Once the assortments from that run are exported and finalized, there is nothing more to do with it. The

user can still copy the run and perform some test scenarios, but the data used in the run corresponds to the old assortment and there is no relationship with newest data from more recent assortments.

I just sent some POG and mapping data to fix some of the errors in an assortment. However, the assortment still shows that mapping results require review. Why wasn't the newest data used to fix the mapping errors?

The assortment to POG mapping process runs every time an assortment is loaded into ASO and every time the direct assortment data is updated. In this case, the data that was delivered is not part of the assortment, so there is not an automatic mapping process for the assortment. An authorized user must use the Assortment Mapping UI and manually request the Re-map action from the drop-down menu. The mapping process will then run again. It will use the latest data available within the application, and, if all the data elements provided fix the old mapping errors, the assortment will map correctly and will be upgraded to the ready for optimization status.

When ASO exports data to APO, will the core and optional items that have been marked as 'Must Keep' in ASO be sent as mandatory items in APO?

APO data is used to initialize the run in ASO as follows: Product Priority=1 becomes Must Keep, Product Priority=2 or 3 become Can Keep and Product Priority=-1 becomes Do Not Include. The user has the option of changing the inclusion constraint in the UI. This inclusion information (Must Keep or Can Keep or Do Not Include) specified in the optimization is not sent out in the ASO Export interface.

What kind of mixed fixture type layouts are supported in ASO?

There are three fixture types that are supported in ASO: Shelf, Pegboard, and Freezer. ASO does allow the user to input a mixed fixture bay layout; however, it provides support when there is at most one transition from a fixture type to another fixture type. For example, it supports a layout that goes from Shelf to Pegboard or vice-versa. However, it does not support a transition such as shelf to pegboard and then back to shelf again.

How do Visual Guidelines work when there are multiple fixture types?

Visual Guidelines are defined per fixture type, and the blocking is carried over from one bay to another for each fixture type. There might be physical separation between the fixtures, but it is ignored for Visual Guidelines. However, when outputting the final position of the items, the actual bay/fixture layouts are respected.

What is the difference between using Gross Profit vs. Gross Profit Return on Investment (GMROI) as objectives?

When the user optimizes with GMROI, the optimization tries to achieve as much of Gross Profit as possible but with the least inventory cost. For example, to maximize gross profit, one could theoretically have lots of inventory. On the other hand, to keep inventory cost low, one could carry very low inventory. So the goal is to analyze the trade-offs between carrying too much inventory and achieving too much gross profit. The GMROI objective balances between these two competing metrics.

What is the difference between the Generic and the Non-Generic version of GMROI?

In the Generic version, the user does not have to provide store level inventory; in the non-generic version, the user is expected to provide store level inventory. Optimization uses the store level inventory for calculating the inventory cost.

I have selected primary as vertical and used the merge sequence for some of the primary vertical blocks. What does it do?

When a primary blocking strategy is vertical, and if the user specifies the merge sequence for a vertical block, then it automatically creates secondary horizontal blocks for this vertical block. This functionality is reversed when the primary is horizontal.

Attribute Processing Questions

Here are some answers to typical questions related to attribute processing.

Is it allowed for RSE_PROD_ATTR_VALUE_XREF_STG to have attribute values which is not linked to items in W_RTL_ITEM_GRP1_DS? For example, given that in a certain season W_RTL_ITEM_GRP1_DS has following item/attributes: Item A Attribute 1, Item B Attribute 2, Item C Attribute 3, and Item D Attribute 4. RSE_PROD_ATTR_VALUE_XREF_STG should have Attribute 1,2,3,4. In the next season, Item C is dropped: Item A Attribute 1, Item B Attribute 2, and Item D Attribute 4. In this case, RSE_PROD_ATTR_VALUE_XREF_STG should be rebuilt to have only Attribute 1,2,4?

Yes, it is allowed. The load will not fail if there is no data found, so it is not a data validation rule that requires there to be a matching attribute in W_RTL_ITEM_GRP1_DS. If there are no products associated with the attribute value, then the attribute value will not be used.

Can you have records with NULL for all the following five "VALUE" columns in RSE_PROD_ATTR_VALUE_XREF_STG? So, records will have value only for PROD_ATTR_VALUE_KEY and ATTR_VALUE_EXT_CODE columns: MIN_ATTR_NUM_VALUE, MAX_ATTR_NUM_VALUE, ATTR_STRING_VALUE, MIN_ATTR_DATE_VALUE, and MAX_ATTR_DATE_VALUE.

The interface provides multiple choices for identifying the attribute value. You must pick the one approach that works for the specific attribute and the remaining values are NULL. So for attributes that relate to data in W_RTL_ITEM_GRP1_DS, you provide a value for ATTR_VALUE_EXT_CODE and leave the other five indicated Attribute Value columns as NULL.

Offer Optimization Questions

Here are some answers to typical questions that you may encounter.

In a price-zone, there are stores with different inventory levels. Some stores have very high inventory levels, and others have low inventory levels. Does OO recommend different recommendations for the zone and such stores separately?

This is not possible in the current release. You can configure Offer Optimization to run either at the price-zone level or at the store-level. If the price-zone level is selected, then all the stores within that price-zone will receive the same price recommendations. If you want to have store-level recommendations, then you must configure the run-level to be at the store-level and let the algorithm dictate the recommendations. If the stores share very similar customer behavior, then it may be possible that they will receive similar recommendations.

I am not interested in the promotions; can the product handle only markdowns?

Yes, the product can handle only markdowns. You can define all the periods to allow markdowns, and no period should have promotions allowed.

I have a planned promotion and I know when the promotion should occur, but I do not know the items and the depth. Can the product recommend items and the depth?

No, the product requires planned promotion to specify the items, depth, and when the promotion should occur. However, if the retailer wants the product to recommend items and depth, then you can specify the promotion and markdown calendar using the interface provided and specify when the promotions and markdowns can be provided. Offer Optimization will determine which items should be promoted or the markdown among the set of periods provided. Note that you can also use the relative calendar option through the UI to specify the calendar.

Should I use Absolute Calendar or Relative Calendar?

The two calendars are provided for flexibility in handling two use cases. When all the items are introduced at almost the same time and exit at almost the same time, then it makes sense to use the absolute calendar. You can also use the relative calendar in such a use case. When the items are introduced at periodic intervals (for example, summer merchandise is introduced in May, June, and July) then it makes sense to use relative calendar. For example, the absolute calendar in such a scenario would probably say May, June as Regular and July as Clearance. This means that items that were introduced in July are going directly into a Clearance season. However, the relative calendar begins after the item is introduced. It is defined as the percentage of regular season and the percentage of the markdown season, based on the total length of the season available for the item.

Is the Style, Color needed as part of Merchandise Hierarchy?

Since the product supports fashion apparel, it is expected that the style and color be provided as part of merchandise hierarchy. If the retailer carries non-apparel items, then the retailer can specify some generic value for style and color, such as not applicable.

How does Offer Optimization handle the online channel?

The retailer can specify the online channel as a separate price zone. This means that the price recommendations for brick-and-mortar stores and online-stores will be different. If the retailer does not wish to have different price recommendations, then it is suggested that you combine the online channel into a brick-and-mortar store.

Can Offer Optimization handle complex promotions?

Offer Optimization handles only simple promotions (for example, percentage off). It does not handle complex promotions like BOGO.

Can Offer Optimization come up with bundles for complex promotions? How can this be handled?

Offer Optimization cannot come up with bundles for a promotion. A bundle can be specified using Product Groups under Business Rules.

Is budget constraint over the effective week or over the entire life of the item?

Budget is assumed to be for the entire remaining life of the item. The user can specify either different budgets for Promotions and Markdowns or a combined budget. Note that a planned promotions can be excluded or included from this budget.

What happens when a user accepts or rejects a recommendation? Do the numbers change for that week? Does it optimize the entire life again?

In the current release, the metrics in the Results or Manage Recommendations do not reflect the accept, reject, or override price recommendations. However, when the user selects the Recalculate button, then all the metrics are recalculated taking into account the price

modifications done by the user. But, when the user selects Reoptimize, then the price modifications are ignored.

Can Offer Optimization handle any type of price-zone?

Yes. In the current release, Offer Optimization can accept price-zones that are defined as any combination of locations and merchandise.

How does Offer Optimization handle multi-currency support?

Offer Optimization accepts the historical sales data in the local currency provided. Further, the location in the application can be specified a country locale. Each application run has a location, which is used to tie the price and price-related metrics (for example, revenue) to the specified currency. However, if you want to configure the run to be at a level that spans multiple currencies, then you must specify one currency.

How does Offer Optimization support dynamic clustering?

Any time you change the price-zone or re-cluster the customer segments, the change must be re-loaded as part of new hierarchies to Offer Optimization. It is considered a major change to the application as it affects the core data elements. The application will be re-calculating the demand parameters and other relevant optimization inputs, based on the new hierarchies. Further, the UI does not offer support to re-group stores within a price-zone based on certain attributes, for example, group stores by sell-through.

How are product images loaded?

Product images can be loaded via the interface provided. Offer Optimization expects that the images will be loaded on an image server, which can be a local server. An URL is provided with the image name that can be accessed by the application. The application searches for the image and populates the UI with the relevant image for that item in the Targeted Offer screen.

A

Appendix: Innovation Workbench Workshop

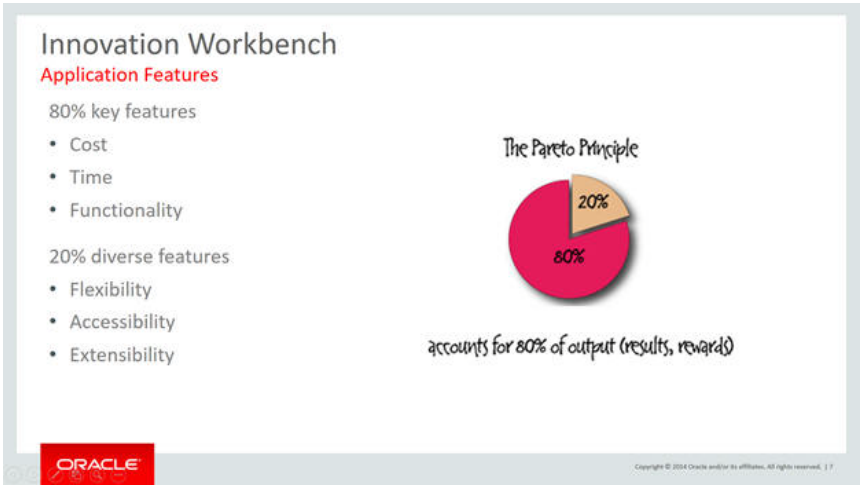
Science Innovation Workbench is a workspace that provides read-only access to the application data objects and clean data by using Oracle APEX. This extension is a workspace for advanced analytics users to add new implementations by using Oracle Advanced Analytic (Oracle R/ODM) algorithms that are implemented as SQL/PLSQL functions.

This appendix provides instructions for using the interface.

Overview

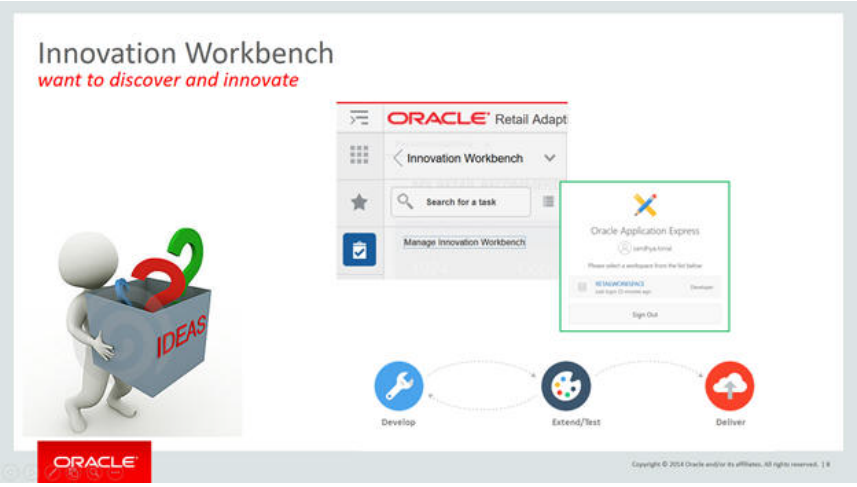
Open Analytics: understanding why, what, and how.

Figure A–1 Overview



Innovation Workbench helps in extending and delivering 20 percent of these features. It provides tools to learn, discover, and innovate existing application features.

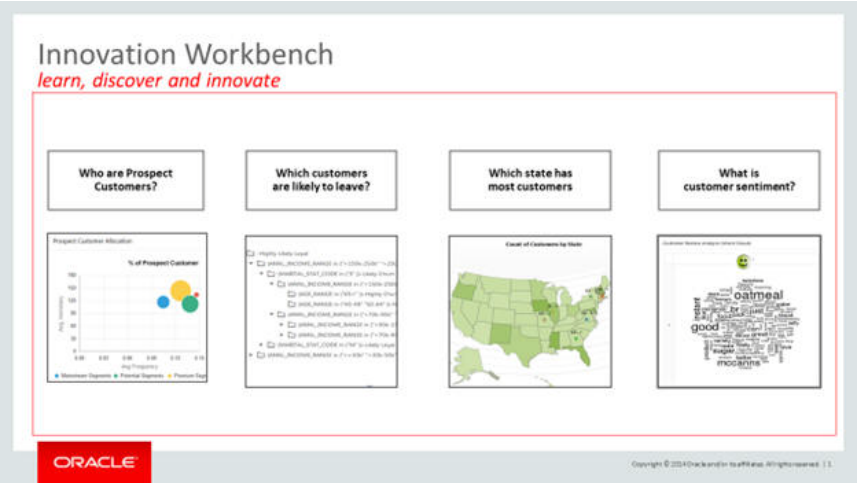
Figure A–2 Discover



Innovation Workbench helps answer questions such as:

- Which customers are likely to leave?
- What is customer feedback?
- Who are the prospect customers?

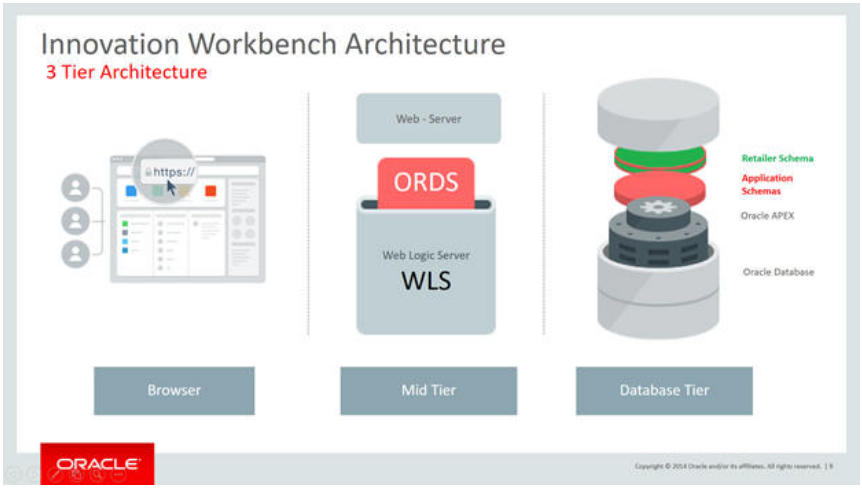
Figure A–3 Learn



With Innovation Workbench, the following new components are available:

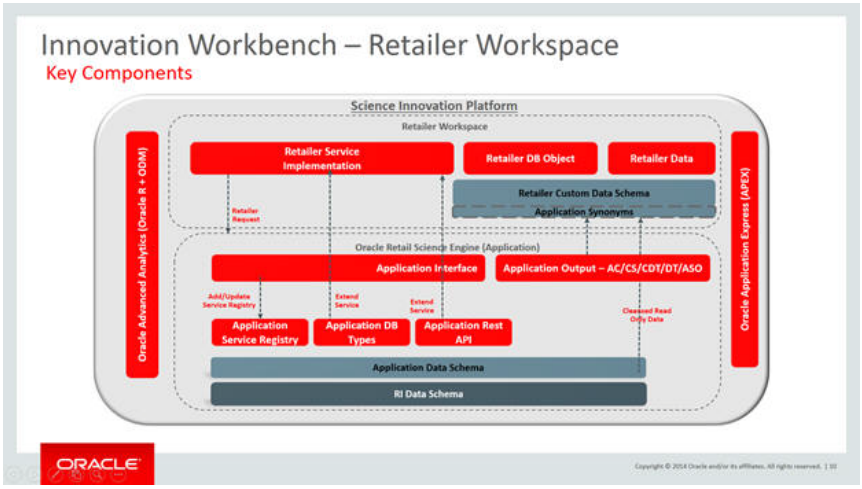
- Retailer schema (see green slice in Figure A–4)
- Retailer workspace
- SQL Workshop
- Application Builder

Figure A–4 Architecture



The Retailer Schema has read-only access to the Application Schema. All permissions are controlled using database permissions.

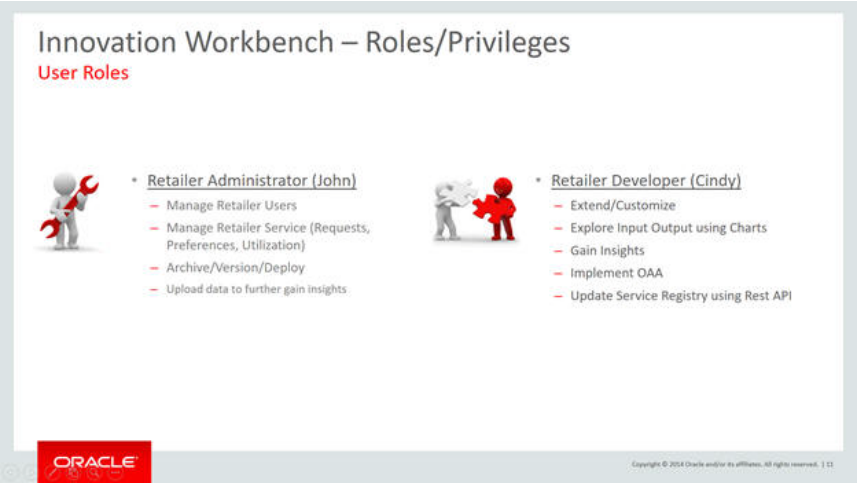
Figure A–5 Key Components



Roles for the analyst, scientist, developer. and integration are as follows:

- Oracle Developer identifies which objects can be accessed.
- Administrator manages users.
- Retailer Developer uses data shared, extend, explore and mine data.

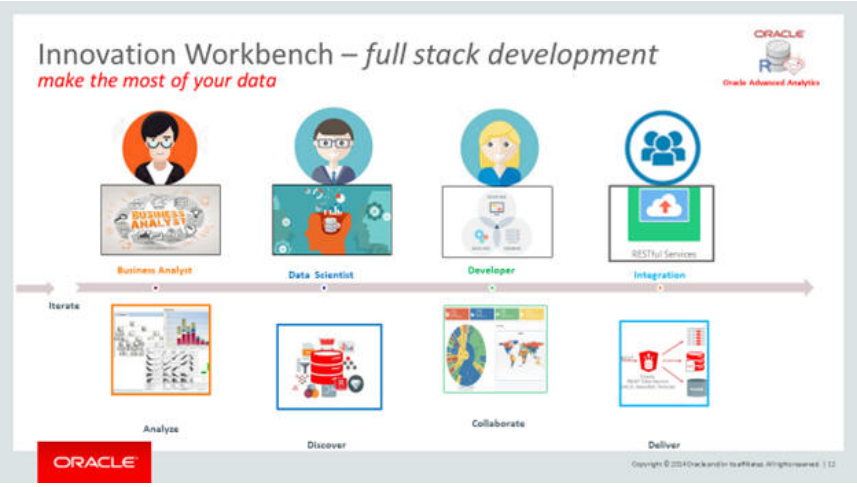
Figure A–6 User Roles



As part of Innovation Workbench, a retailer developer has the following roles:

- Business Analyst, explores data, gains insights
- Data Scientist, discovers patterns in data mining using ODM
- Developer, develops applications and shares insights via UI and web services

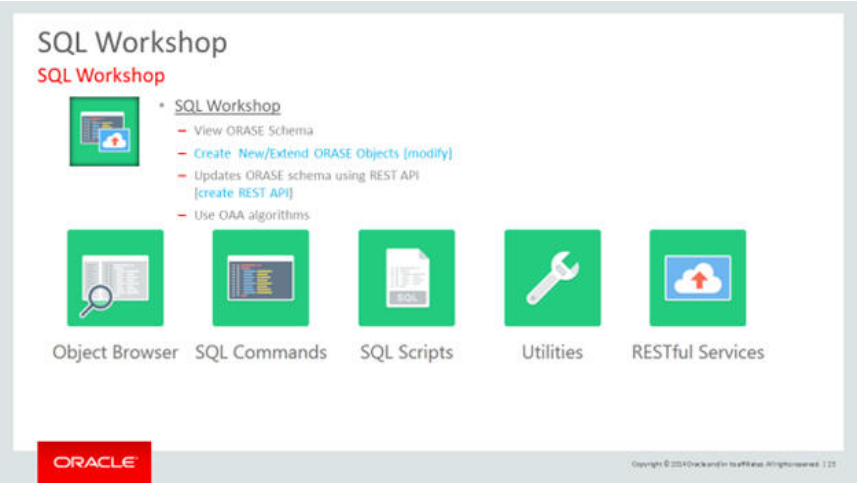
Figure A–7 Full Stack Development



Key components

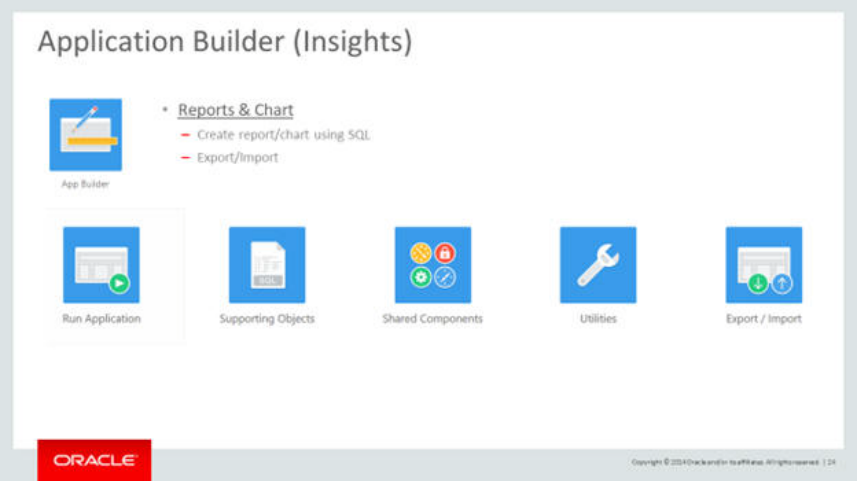
The SQL Workshop provides tools to view and manage database objects.

Figure A–8 SQL Workshop



The Application Builder provides tools to create reports and charts using wizards and page designers.

Figure A–9 Application Builder (Insights)

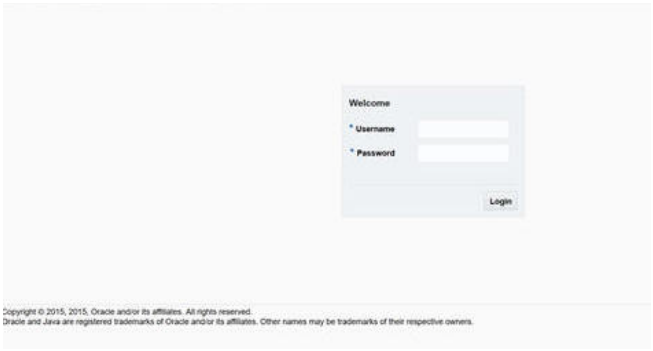


Lab Innovation Workbench

1. Login using http:// < url>

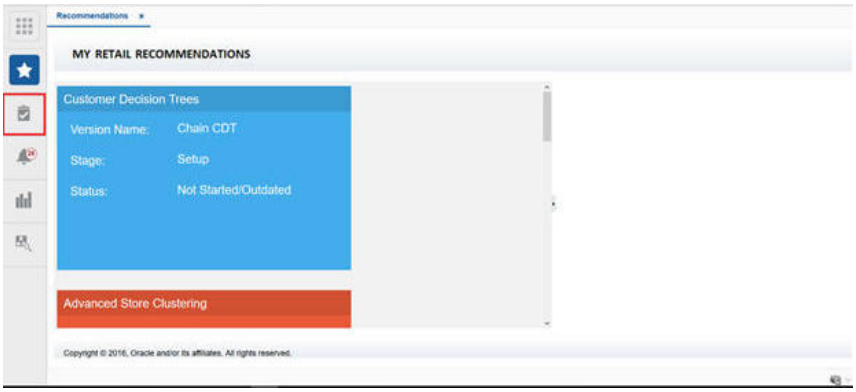
   Login/Password

Figure A–10 Login



This takes you to the application.

Figure A–11 Application



2. Select Innovation Workbench.

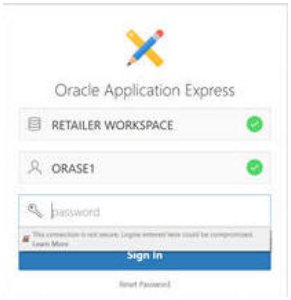
Figure A–12 Select Innovation Workbench

3. Select Manage Innovation Workbench.

Figure A–13 Manage Innovation Workbench

This takes you to Retailer Workspace where you can log in with same login and password credentials.

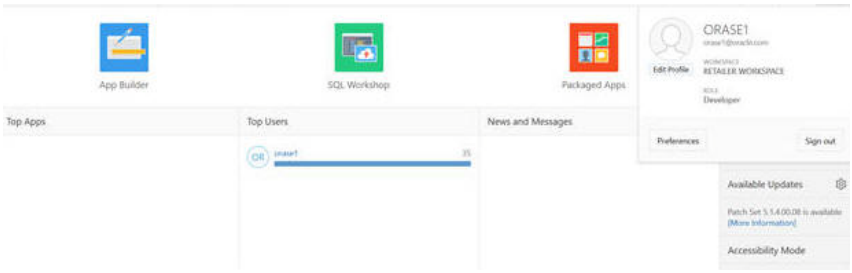
Figure A–14 Retailer Workspace Login



Retailer Workspace has a dedicated schema and work area to do full stack development.

- Dedicated Schema for retailer
  - Granting and revoking database object privileges
  - Dedicated service plan and consumer group for retailer workspace
  - Access to database utilities and scheduling for long-running transactions
- SQL Workshop
- Application Builder

Figure A–15 Retailer Workspace



Lab Analysis

Browse Existing Data

In this exercise you will browse the Application schema objects using SQL Workshop.

- Access Read Only
- Cleansed
- Aggregated
- Filtered
- Sample
- Inputs and Outputs

Figure A–16 Application Schema Objects

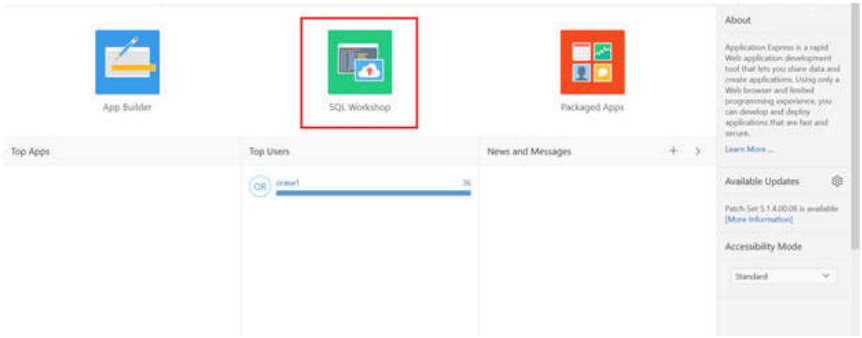


Figure A–17 SQL Workshop Object Browser

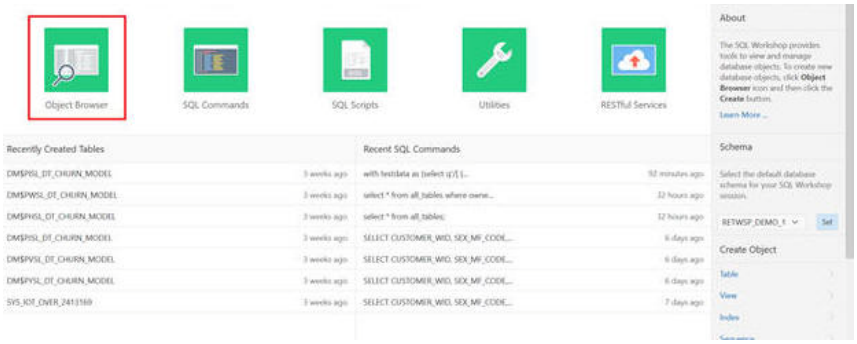
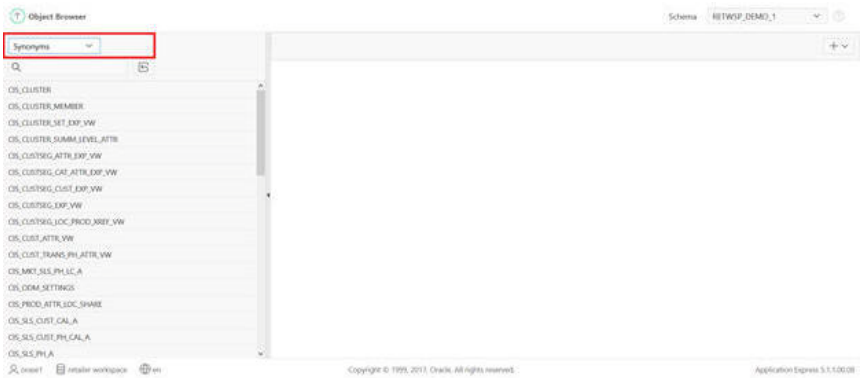


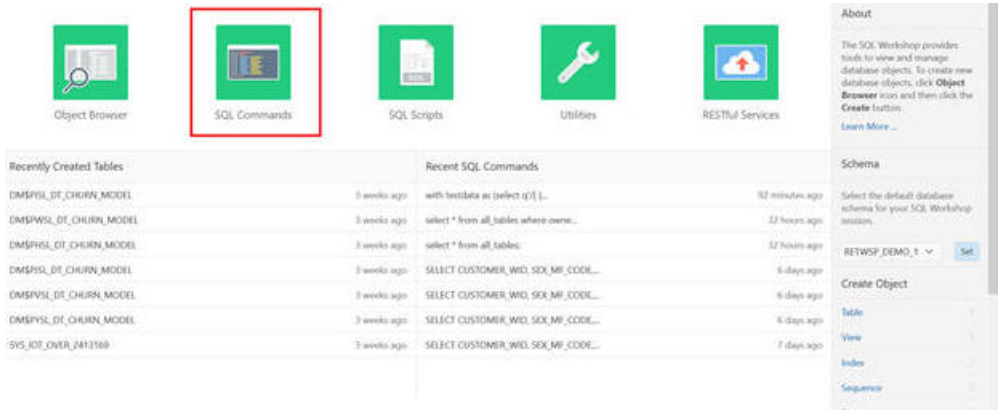
Figure A–18 SQL Workshop Object Browser Synonym



Note that this dedicated schema is only associated with the Application Schema and cannot be extended to any other new schema.

Data

Figure A–19 SQL Workshop SQL Command



This SQL has been saved, so you can type sql in the command and execute and save it.

Figure A–20 SQL Workshop SQL Command Save SQL

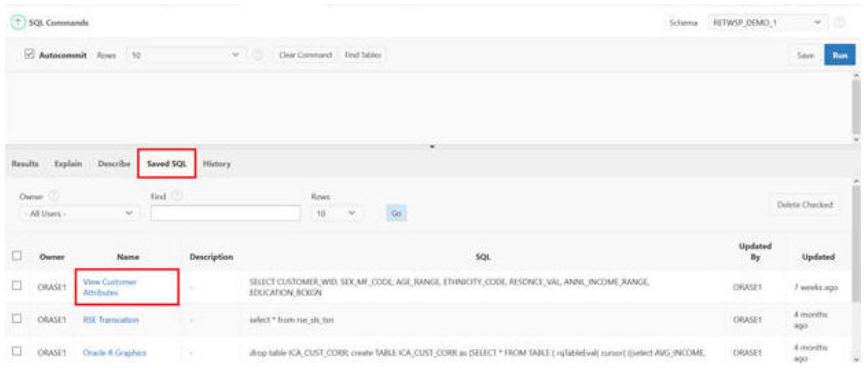


Figure A–21 SQL Workshop SQL Command Saved SQL View Customer Attributes

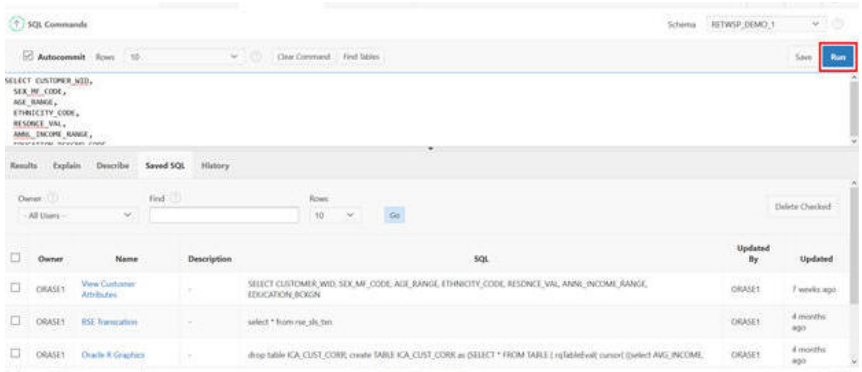


Figure A–22 Browse Data

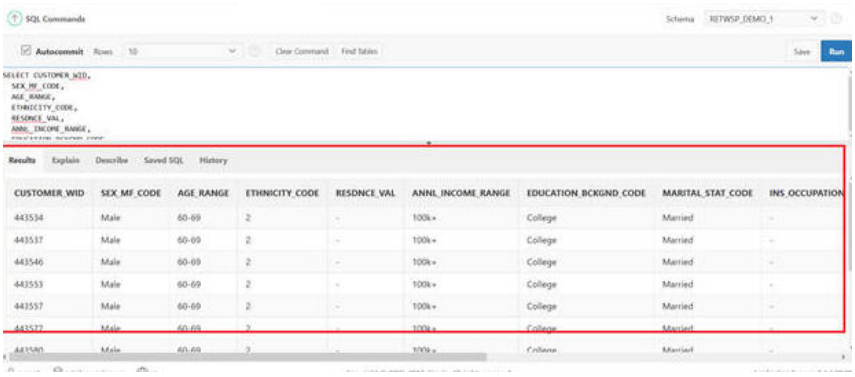
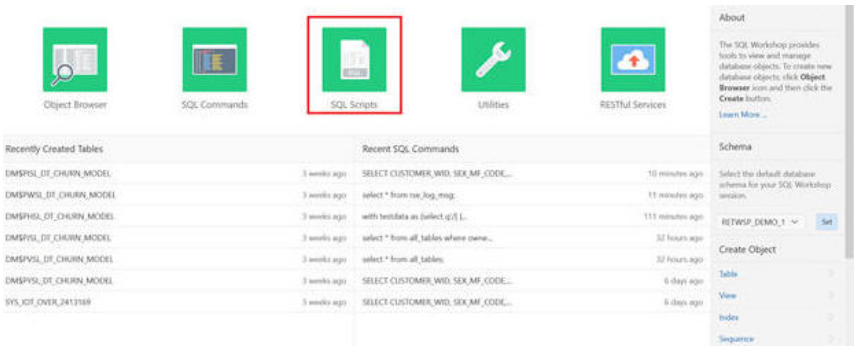


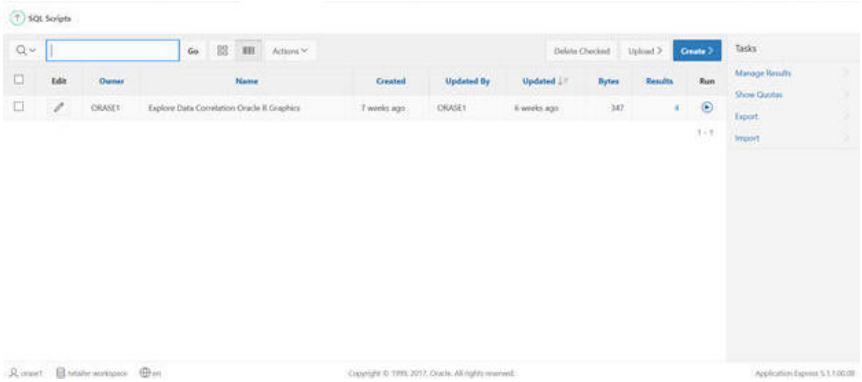
Figure A–23 SQL Workshop SQL Scripts



Using this you can do the following:

- Upload Scripts
- Create Scripts
- Edit Scripts

Figure A–24 Scripts



- Load data
- Use ETL feeds
- Use system feeds whereever possible. If the application feed is not available, then ReST objects can be used.

Using ReST to Load Data

Data can be loaded to the retailer schema using the ReST API in JSON format. These must be batched from the client end while uploading data into application.

SQL Workshop ' SQL Command

Create table that will hold Customer Reviews

DROP TABLE retwsp_cust_prod_rev_stg;

CREATE TABLE retwsp_cust_prod_rev_stg(

json_str clob

);

```
DROP SEQUENCE retwsp_cust_prod_rev_seq
CREATE SEQUENCE retwsp_cust_prod_rev_seq
 START WITH     1
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
reviewerID varchar2(255) NOT NULL,
```

DROP TABLE retwsp_customer_product_review;

CREATE TABLE retwsp_customer_product_review

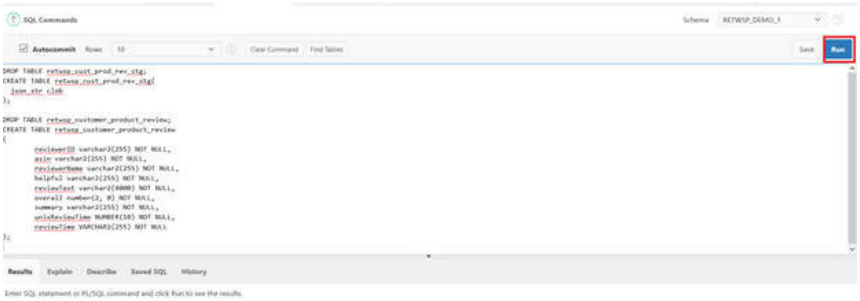
(

- reviewid NUMBER NOT NULL,
- reviewerID varchar2(255) NOT NULL,
- asin varchar2(255) NOT NULL,
- reviewerName varchar2(255) NOT NULL,
- helpful varchar2(255) NOT NULL,
- reviewText varchar2(4000) NOT NULL,
- overall number(2, 0) NOT NULL,
- summary varchar2(255) NOT NULL,
- unixReviewTime NUMBER(10) NOT NULL,
- reviewTime VARCHAR2(255) NOT NULL

);

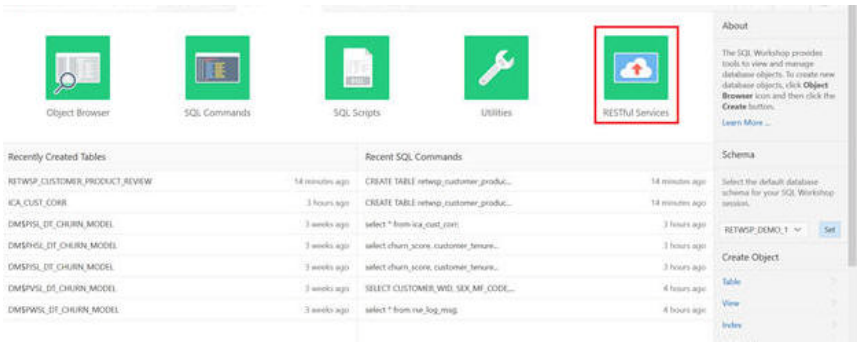
1. Copy and paste the above content and click **Run.**

Figure A–25 Load Data Script



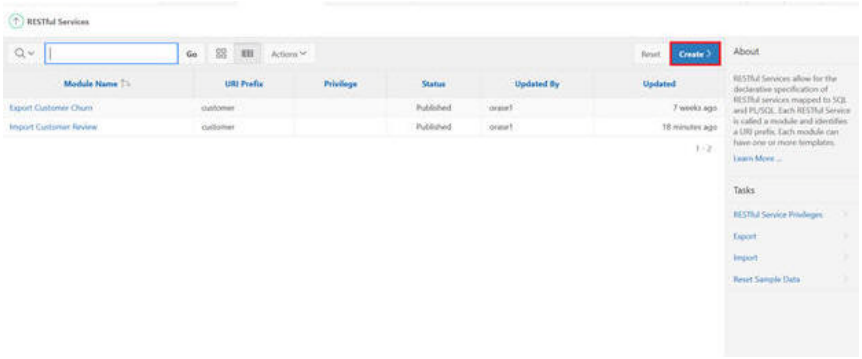
2. Create ReST API, which allows you to load customer reviews in the above table.

Figure A–26 Create REST API



3. Click **Create**.

Figure A–27 Create



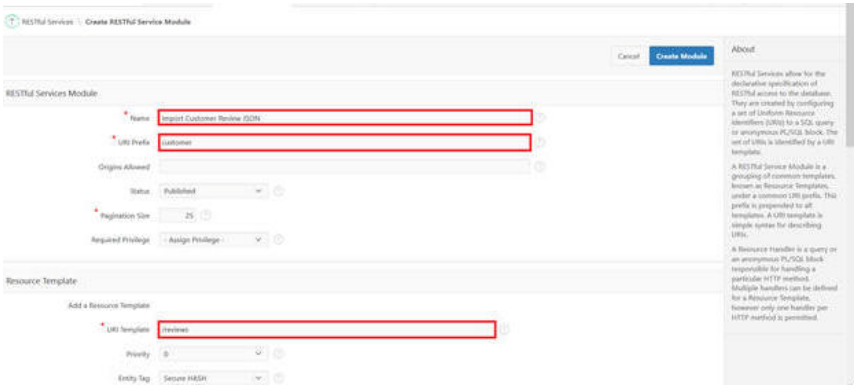
4. Here is the code to populate the JSON data to the relational table.

```
Declare
 customer_review_lblob blob;
 customer_review_cblob clob;
begin
    customer_review_lblob := :body;
    customer_review_cblob := wwv_flow_utilities.blob_to_clob(customer_review_
lblob);
    -- insert into stage table
    insert into retwsp_cust_prod_rev_stg values (customer_review_cblob);

    -- move to target table
    insert into retwsp_customer_product_review (reviewid, reviewerID, asin,
reviewerName, helpful, reviewText, overall, summary, unixReviewTime,
reviewTime)
    with review_data
    as (select json_str from retwsp_cust_prod_rev_stg)
    select retwsp_cust_prod_rev_seq.nextval, j.*
    from review_data rd,
    json_table(rd.json_str, '$[*]'
columns (reviewerID varchar2 path '$.reviewerID'
                      ,asin varchar2 path '$.asin'
                      ,reviewerName varchar2 path '$.reviewerName'
                      ,helpful number path '$.helpful'
                      ,reviewText varchar2 path '$.reviewText'
```

```
                        ,overall number path '$.overall'
                        ,summary varchar2 path '$.summary'
                        ,unixReviewTime number path '$.unixReviewTime'
                        ,reviewTime varchar2 path '$.reviewTime'
                        )
    ) j;
    delete from RETWSP_TMP_CUST_PROD_RW;
    commit;
end;
```

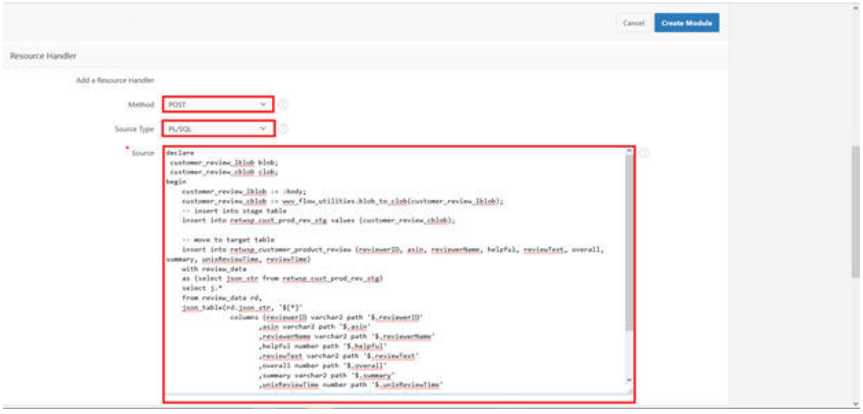
5. Provide the essential elements, including name, URI prefix, and template.

Figure A–28 ReST Information



6. Select the method. In this case it is POST, since the data is getting updated.

Figure A–29 Select Post



7. Populate the name for the RESTful Services Module. URI Prefix as customer and URI Template/reviews. The / is required.
8. Add resource handler method - POST and source type - PL/SQL
9.

```
Declare
 customer_review_lblob blob;
 customer_review_cblob clob;
begin
    customer_review_lblob := :body;
    customer_review_cblob := wwv_flow_utilities.blob_to_clob(customer_review_
```

```
lblob);
    -- insert into stage table
    insert into retwsp_cust_prod_rev_stg values (customer_review_cblob);

    -- move to target table
    insert into retwsp_customer_product_review (reviewid, reviewerID, asin,
reviewerName, helpful, reviewText, overall, summary, unixReviewTime,
reviewTime)
    with review_data
    as (select json_str from retwsp_cust_prod_rev_stg)
    select retwsp_cust_prod_rev_seq.nextval, j.*
    from review_data rd,
    json_table(rd.json_str, '$[*]'
columns (reviewerID varchar2 path '$.reviewerID'
                      ,asin varchar2 path '$.asin'
                      ,reviewerName varchar2 path '$.reviewerName'
                      ,helpful number path '$.helpful'
                      ,reviewText varchar2 path '$.reviewText'
                      ,overall number path '$.overall'
                      ,summary varchar2 path '$.summary'
                      ,unixReviewTime number path '$.unixReviewTime'
                      ,reviewTime varchar2 path '$.reviewTime'
                      )
    ) j;
    delete from retwsp_cust_prod_rev_stg;
    commit;
end;
```

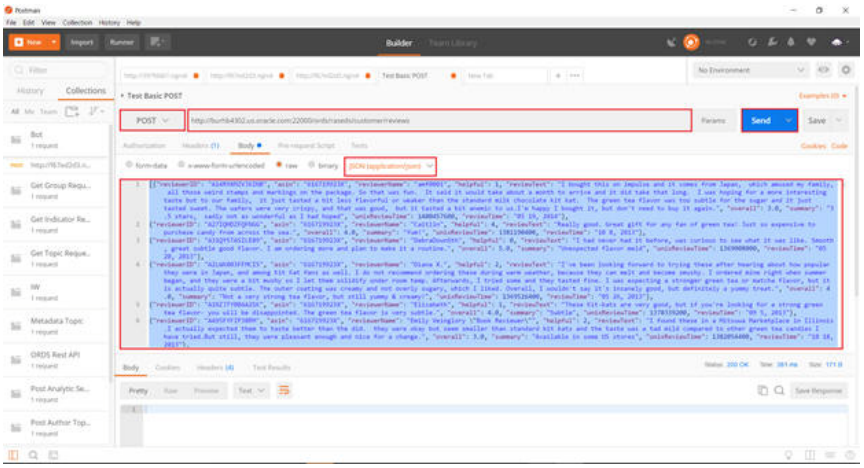
10. Post customer reviews using following the data http://<url>

    This posting can be done using POSTMaN or any ReST Client (such as java application posting data into Retailer Schema)

    This step requires PostMan or a client that will push reviews. Download and Install PostMan. If it cannot be installed, go to the next step.

    https://www.getpostman.com/apps

Figure A–30 Post



11. Select * from retwsp_customer_product_review;

Other Methods for Loading Data

These methods must be used judiciously, with smaller datasets.

Figure A–31 Data Workshop

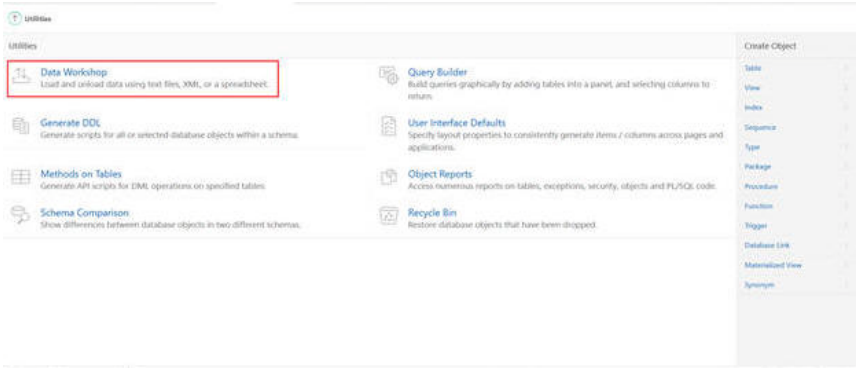


Figure A–32 Load and Unload Data

Prepare Data

Key components that can be used for refining data are Create Train and Test dataset.

The followings SQL shows how stratified split is used to preserve the categorical target distribution in the resulting training and test dataset.

Key items below are:

- target column. It must be categorical type.
- case id column. It must contain unique numbers that identify the rows.
- input data set. {percent of training dataset} - percent of training dataset. For example, if you want to split 60% of input dataset into a training dataset, use the value 60. The test dataset will contain 100%-60% = 40% of the input dataset. The training and test dataset are mutually exclusive.

Train Datsasets

```
create or replace view retwsp_rf_cust_attr_test_vw as (SELECT v1.* FROM (
-- randomly divide members of the population into subgroups based on target
classes
SELECT a.*, row_number() OVER (partition by EDUCATION_BCKGND_CODE ORDER BY ORA_
HASH(CUSTOMER_WID)) "_partition_caseid"
FROM
retwsp_rf_cust_attr_vw a) v1, (
-- get the count of subgroups based on target classes
```

```
SELECT
EDUCATION_BCKGND_CODE,
COUNT(*) "_partition_target_cnt" FROM
retwsp_rf_cust_attr_vw GROUP BY EDUCATION_BCKGND_CODE) v2
WHERE v1.EDUCATION_BCKGND_CODE = v2.EDUCATION_BCKGND_CODE
-- random sample subgroups based on target classes in respect to the sample size
AND ORA_HASH(v1."_partition_caseid", v2."_partition_target_cnt"-1, 0) <= (v2."_
partition_target_cnt" * 60 / 100));
```

Test Datasets

```
create or replace view retwsp_rf_cust_attr_test_vw as (SELECT v1.* FROM (
-- randomly divide members of the population into subgroups based on target
classes
SELECT a.*, row_number() OVER (partition by EDUCATION_BCKGND_CODE ORDER BY ORA_
HASH(CUSTOMER_WID)) "_partition_caseid"
FROM
retwsp_rf_cust_attr_vw a) v1, (
-- get the count of subgroups based on target classes
SELECT
EDUCATION_BCKGND_CODE,
COUNT(*) "_partition_target_cnt" FROM
retwsp_rf_cust_attr_vw GROUP BY EDUCATION_BCKGND_CODE) v2
WHERE v1.EDUCATION_BCKGND_CODE = v2.EDUCATION_BCKGND_CODE
-- random sample subgroups based on target classes in respect to the sample size
AND ORA_HASH(v1."_partition_caseid", v2."_partition_target_cnt"-1, 0) <= (v2."_
partition_target_cnt" * 40 / 100));
```

Explore Data Analysis

Now we have customer reviews we will review, build text to find document frequencies, and display them in charts. We will further extract key text for each review.

Review Customer Behavior Analysis using Text Mining

This script can be found under SQLWorkshop ' SQL Scripts [Text Mining Feature Extraction]

```
-------------------------------------------------------------Build Text and Token
-----------------------------------------
DECLARE
    v_policy_name   VARCHAR2(4000);
    v_lexer_name    VARCHAR2(4000);
BEGIN
    v_policy_name := 'RETWSP_POLICY';
    v_lexer_name := 'RETWSP_LEXER';
    ctx_ddl.drop_preference(v_lexer_name);
    ctx_ddl.drop_policy(
        policy_name   => v_policy_name
    );
    ctx_ddl.create_preference(
        v_lexer_name,
        'BASIC_LEXER'
    );
    ctx_ddl.create_policy(
        policy_name   => v_policy_name,
        lexer         => v_lexer_name,
        stoplist      => 'CTXSYS.DEFAULT_STOPLIST'
    );
```

```
END;
/
DROP TABLE retwsp_feature;

CREATE TABLE retwsp_feature
    AS
        SELECT
            'REVIEWTEXT' "COLUMN_NAME",
            token,
            value,
            rank,
            count
        FROM
            (
                SELECT
                    token,
                    value,
                    RANK() OVER(
                        ORDER BY value ASC
                    ) rank,
                    count
                FROM
                    (
                        SELECT
                            column_value token,
                            ln(n / COUNT(*) ) value,
                            COUNT(*) count
                        FROM
                            (
                                SELECT
                                    t2.column_value,
                                    t1.n
                                FROM
                                    (
                                        SELECT
                                            COUNT(*) OVER() n,
                                            ROWNUM rn,
                                            odmr_engine_text.dm_policy_tokens(
                                                'RETWSP_POLICY',
                                                reviewtext
                                            ) nt
                                        FROM
                                            retwsp_customer_product_review
                                    ) t1,
                                    TABLE ( t1.nt ) t2
                                GROUP BY
                                    t2.column_value,
                                    t1.rn,
                                    t1.n
                            )
                        GROUP BY
                            column_value,
                            n
                    )
            )
        WHERE
            rank <= 3000;

SELECT
```

```
    *
FROM
    retwsp_feature;

create table retwsp_review_df as
SELECT
    *
FROM
    (
        SELECT
            nested_result.attribute_name,
            COUNT(nested_result.attribute_name) count_val
        FROM
            (
                WITH
/* Start of sql for node: RETWSP_CUSTOMER_PRODUCT_REVIEW */ "N$10001" AS (
                    SELECT /*+ inline */
                        "RETWSP_CUSTOMER_PRODUCT_REVIEW"."ASIN",
                        "RETWSP_CUSTOMER_PRODUCT_REVIEW"."REVIEWERID",
                        "RETWSP_CUSTOMER_PRODUCT_REVIEW"."OVERALL",
                        "RETWSP_CUSTOMER_PRODUCT_REVIEW"."REVIEWTEXT"
                    FROM
                        "RETWSP_DEMO_1"."RETWSP_CUSTOMER_PRODUCT_REVIEW"
                )
/* End of sql for node: RETWSP_CUSTOMER_PRODUCT_REVIEW */,
/* Start of sql for node: Build Text */ "N$10002" AS (
                    SELECT /*+ inline */
                        "REVIEWERID",
                        "ASIN",
                        "OVERALL",
                        odmr_engine_text.dm_text_token_features(
                            'RETWSP_POLICY',
                            "REVIEWTEXT",
                            'RETWSP_FEATURE',
                            NULL,
                            50,
                            'IDF'
                        ) "REVIEWTEXT_TOK"
                    FROM
                        "N$10001"
                )
/* End of sql for node: Build Text */ SELECT
                    *
                FROM
                    "N$10002"
            ) output_result,
            TABLE ( output_result.reviewtext_tok ) nested_result
        GROUP BY
            nested_result.attribute_name
    )
ORDER BY count_val DESC;

select * from retwsp_review_df;
------------------------------------------------------------Feature Extraction
and Feature Comparison ------------------------------------------
-- Create the settings table
DROP TABLE RETWSP_ESA_settings;
CREATE TABLE RETWSP_ESA_settings (
    setting_name VARCHAR2(30),
    setting_value VARCHAR2(30));
```

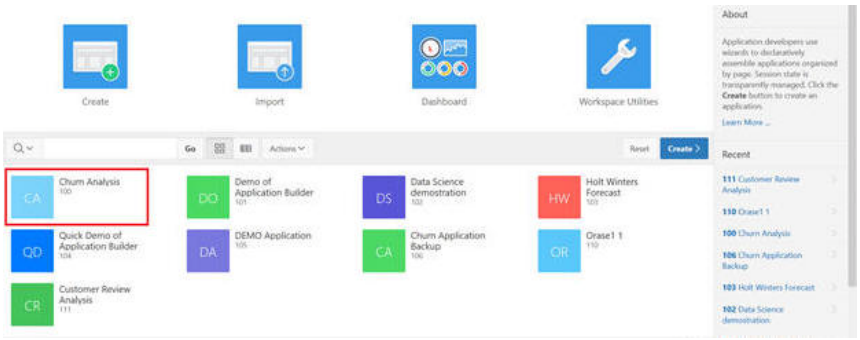
```
     DECLARE  ---------- sub-block begins
        already_exists   EXCEPTION;
        PRAGMA EXCEPTION_INIT(already_exists, -00955);
        v_stmt VARCHAR2(4000);
        v_param_name VARCHAR2(100);
        v_param_value VARCHAR2(100);
     BEGIN
      dbms_output.put_line('Start Populate settings table' || dbms_data_
mining.algo_name);
      dbms_output.put_line('Start Populate settings table' || dbms_data_
mining.algo_nonnegative_matrix_factor);
      v_param_name := dbms_data_mining.algo_name;
      v_param_value := dbms_data_mining.ALGO_NONNEGATIVE_MATRIX_FACTOR;
      v_stmt := 'INSERT INTO RETWSP_ESA_settings (setting_name, setting_value)
VALUES (''' || v_param_name || ''','''' || v_param_value || ''')';
      dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
      EXECUTE IMMEDIATE v_stmt;
      v_param_name := dbms_data_mining.prep_auto;
      v_param_value := dbms_data_mining.prep_auto_on;
      v_stmt := 'INSERT INTO RETWSP_ESA_settings (setting_name, setting_value)
VALUES (''' || v_param_name || ''','''' || v_param_value || ''')';
      dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
      EXECUTE IMMEDIATE v_stmt;
      EXCEPTION
          WHEN already_exists THEN
            dbms_output.put_line('Exception not found');
     END;  ------------- sub-block ends
/

create view RETWSP_CUST_PROD_REVIEW_VW as (select reviewid, reviewtext from
RETWSP_CUSTOMER_PRODUCT_REVIEW);

DECLARE
   v_xlst              dbms_data_mining_transform.TRANSFORM_LIST;
   v_policy_name       VARCHAR2(130) := 'RETWSP_POLICY';
   v_model_name        varchar2(50) := 'RETWSP_ESA_MODEL';
BEGIN
    v_xlst := dbms_data_mining_transform.TRANSFORM_LIST();

    DBMS_DATA_MINING_TRANSFORM.SET_TRANSFORM(v_xlst, 'REVIEWTEXT', NULL,
'REVIEWTEXT', NULL, 'TEXT(POLICY_NAME:'||v_policy_name||')(MAX_FEATURES:3000)(MIN_
DOCUMENTS:1)(TOKEN_TYPE:NORMAL)');
    DBMS_DATA_MINING.DROP_MODEL(v_model_name, TRUE);
    DBMS_DATA_MINING.CREATE_MODEL(
        model_name          => v_model_name,
        mining_function     => DBMS_DATA_MINING.FEATURE_EXTRACTION,
        data_table_name     => 'RETWSP_CUST_PROD_REVIEW_VW',
        case_id_column_name => 'REVIEWID',
        settings_table_name => 'RETWSP_ESA_SETTINGS',
        xform_list          => v_xlst);
END;
/

------------------
-- List top (largest) 3 features that represent are represented in each review.
-- Explain the attributes which most impact those features.
-- This can be used in UI to display all key features in each review.
select REPLACE(xt.attr_name,'"REVIEWTEXT".',''), xt.attr_value, xt.attr_weight,
```

```
xt.attr_rank
from (SELECT S.feature_id fid, value val,
       FEATURE_DETAILS(RETWSP_ESA_MODEL, S.feature_id, 5 using T.*) det
FROM
  (SELECT v.*, FEATURE_SET(RETWSP_ESA_MODEL, 3 USING *) fset
    FROM RETWSP_CUSTOMER_PRODUCT_REVIEW v
   WHERE reviewid = 1) T,
  TABLE(T.fset) S
order by val desc) X, XMLTABLE('/Details'
PASSING X.DET
COLUMNS
"ALGORITHM" VARCHAR2(30) PATH '@algorithm',
"FEATURE" VARCHAR2(30) PATH '@feature',
RECORDS XMLTYPE PATH '/Details') R,
XMLTABLE ('/Details/Attribute'
PASSING R.RECORDS
COLUMNS
"ATTR_NAME" VARCHAR2(30) PATH '@name',
"ATTR_VALUE" VARCHAR2(120) PATH '@actualValue',
"ATTR_WEIGHT" VARCHAR2(10) PATH '@weight',
"ATTR_RANK" VARCHAR2(10) PATH '@rank'
) XT
ORDER BY ATTR_RANK;
```

To display the prepared data in UI for end user to review, complete the following:

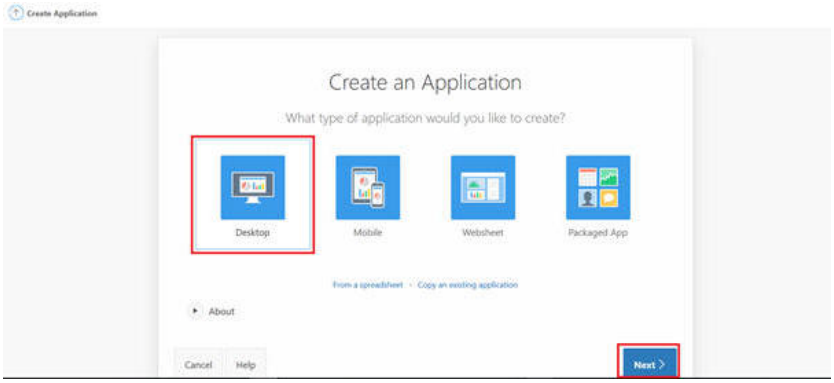
1. Select **Application Builder** to create the UI.

Figure A–33 Churn Analysis



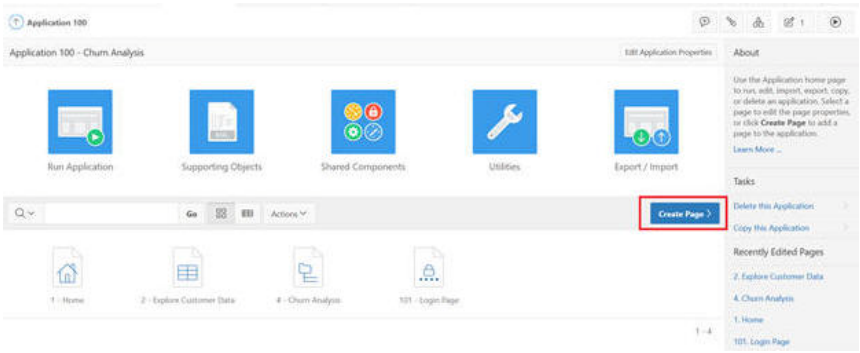
2. Click **Create**.

Figure A–34 Create an Application



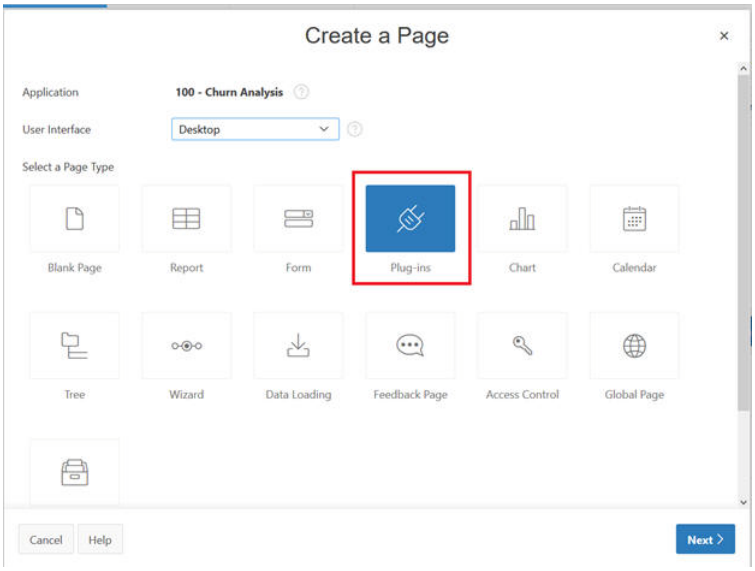
3. Select **Create Page**.

Figure A–35 Create Page



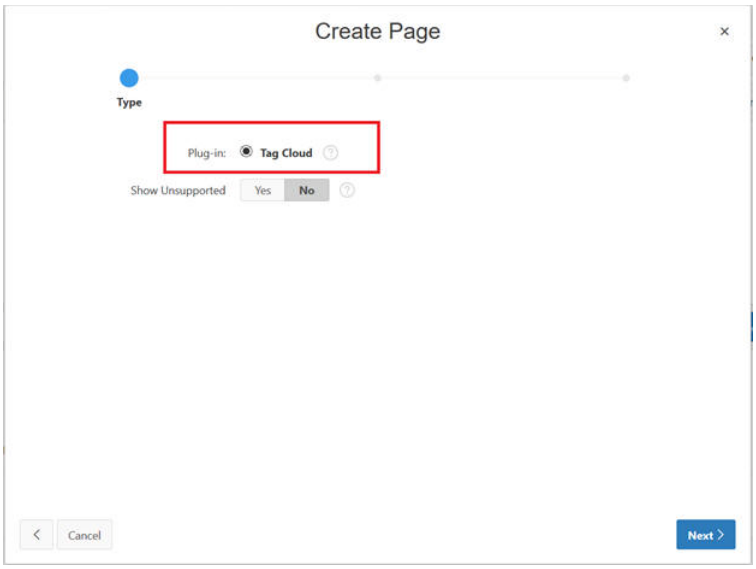
4. Select **Plugin** and click **Next**.

Figure A–36 Plug-In



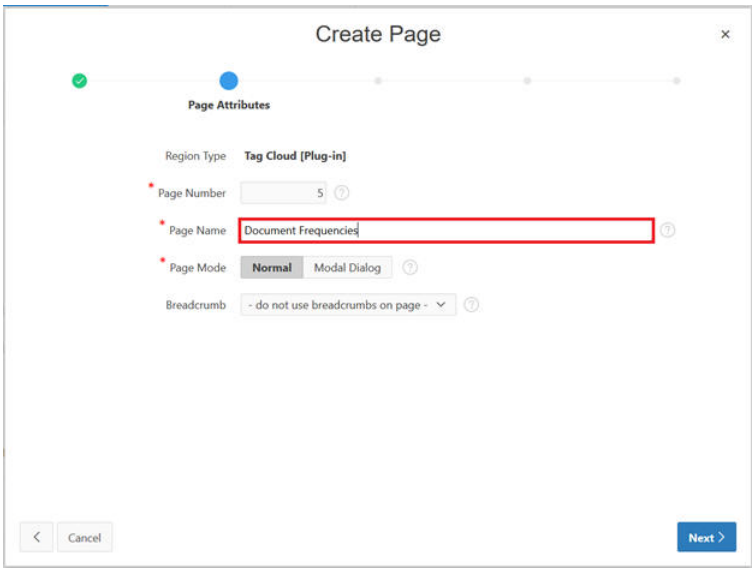
5. Select **Type**.

Figure A–37 Select Type



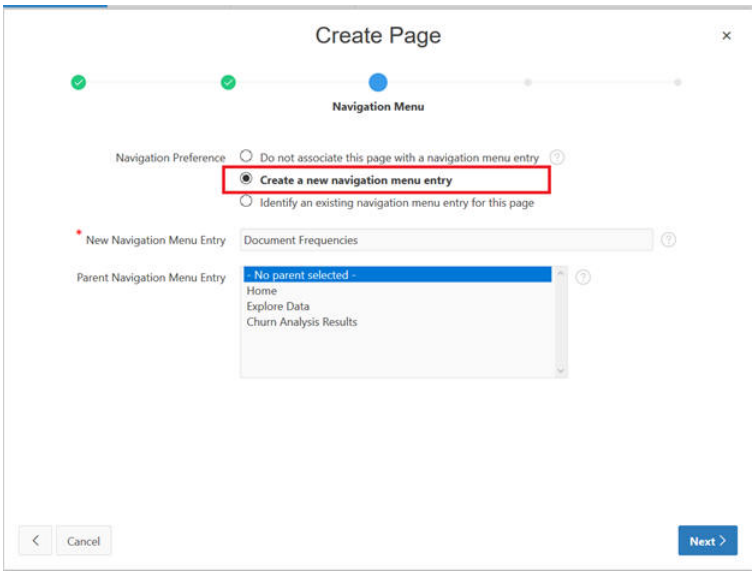
6. Populate Page Name and click **Next**.

Figure A–38 Populate Page Name



7. Select the Navigation Menu entries and click **Next**.

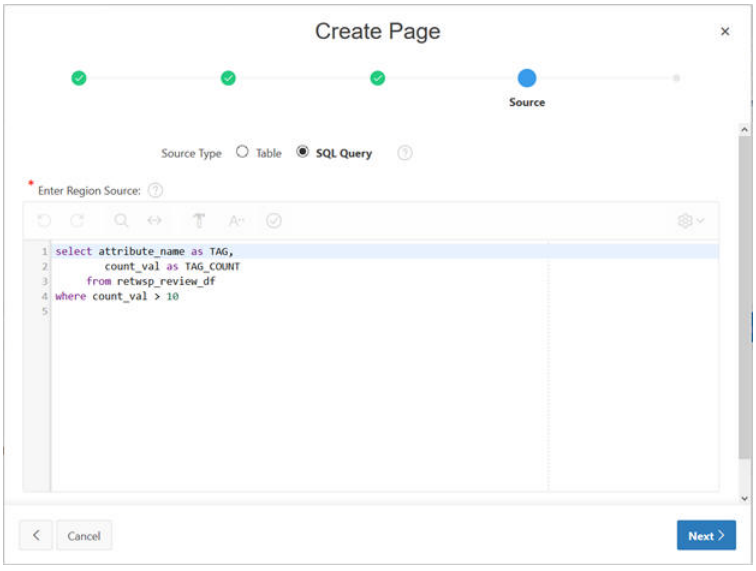
Figure A–39 Navigation Menu



8. Click **Next** and populate SQL to build the chart.

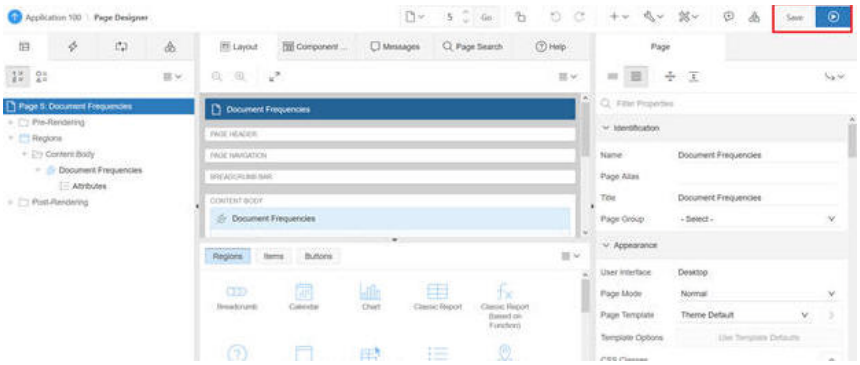
```
select attribute_name as TAG,
        count_val as TAG_COUNT
     from retwsp_review_df
where count_val > 30
order by attribute_name;
```

Figure A–40 SQL Query



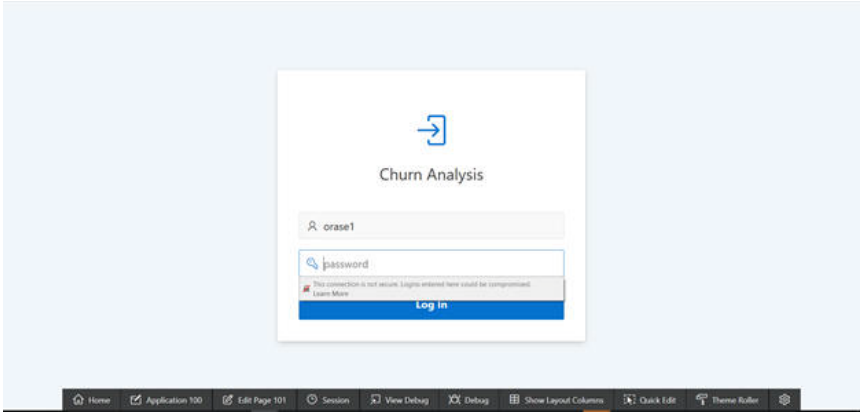
9. You see the Page Designer. Click **Save** and Run icon.

Figure A–41 Page Designer



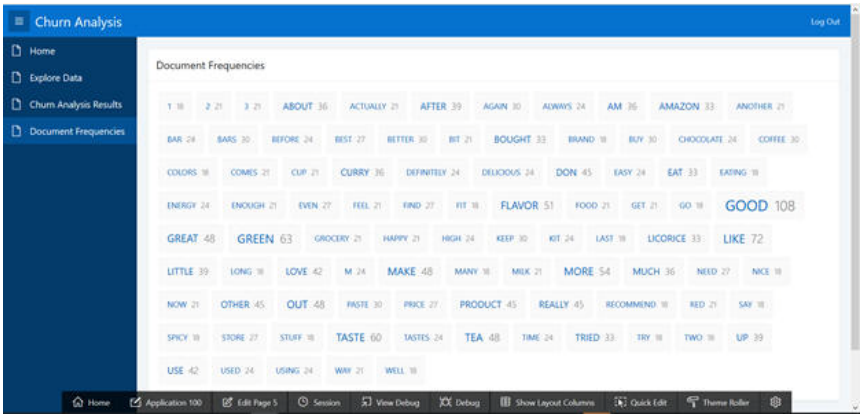
10. Log in with username/password.

Figure A–42 Churn Login



11. The just designed screen is displayed, with the following content.

Figure A–43 Churn Analysis

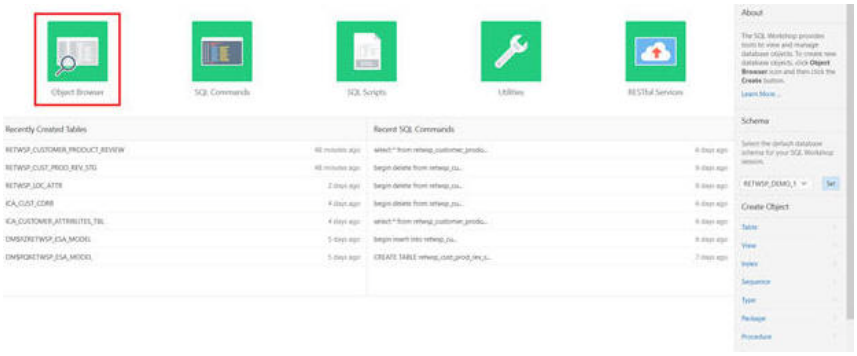


Innovate

Churn Analysis using ODM

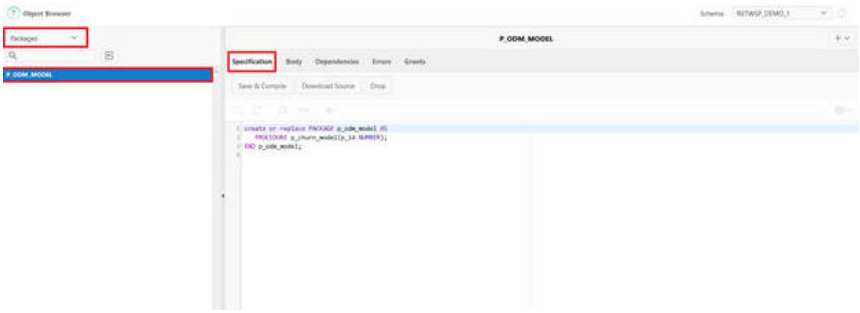
1. View SQL Workshop Object Browser.

Figure A–44 Object Browser



2. Select **Packages** and **Specification**.

Figure A–45 Packages and Specification



3. Select the body that creates a classification model using the Decision Tree algorithm.

   Key considerations in the code are

   - Note the default classification algorithm in ODM is Naive Bayes. In order to override, create and populate a settings table to be used as input to the model.

     Examples of other possible settings are:

     (dbms_data_mining.tree_impurity_metric, 'TREE_IMPURITY_ENTROPY')

     (dbms_data_mining.tree_term_max_depth, 5)

     (dbms_data_mining.tree_term_minrec_split, 5)

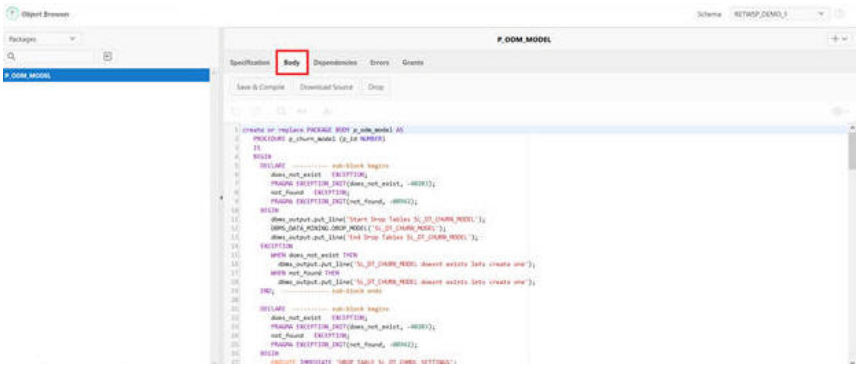
     (dbms_data_mining.tree_term_minpct_split, 2)

     (dbms_data_mining.tree_term_minrec_node, 5)

     (dbms_data_mining.tree_term_minpct_node, 0.05)

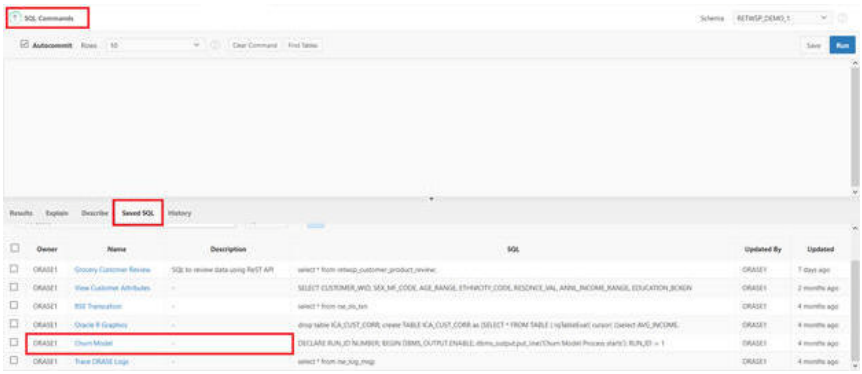
   - A cost matrix is used to influence the weighting of mis-classification during model creation and scoring.

Figure A–46 Cost Matrix



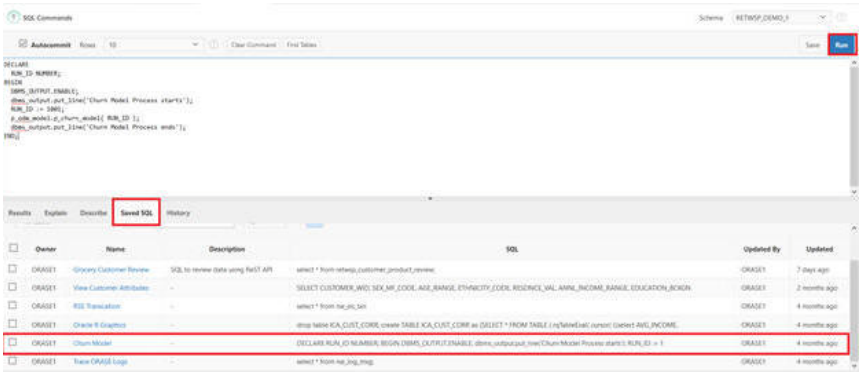
4. Execute the procedure via SQL Command.

Figure A–47 SQL Command



5. Execute Churn Model analysis and view the results.

Figure A–48 Model Analysis Results



Decision Tree Model Details

Here are the results.

```
SELECT
 dbms_data_mining.get_model_details_xml('SL_DT_CHURN_MODEL')
 AS DT_DETAILS
FROM dual;
```

Results of the XML is as follows, next step will be to parse the XML and

```
<PMML version="2.1">
  <Header copyright="Copyright (c) 2004, Oracle Corporation. All rights
reserved."/>
  <DataDictionary numberOfFields="4">
    <DataField name="AGE_RANGE" optype="categorical"/>
    <DataField name="ANNL_INCOME_RANGE" optype="categorical"/>
    <DataField name="CHURN_SCORE" optype="categorical"/>
    <DataField name="PARTY_TYPE_CODE" optype="categorical"/>
  </DataDictionary>
  <TreeModel modelName="SL_DT_CHURN_MODEL" functionName="classification"
splitCharacteristic="binarySplit">
    <Extension name="buildSettings">
      <Setting name="TREE_IMPURITY_METRIC" value="TREE_IMPURITY_GINI"/>
      <Setting name="TREE_TERM_MAX_DEPTH" value="7"/>
      <Setting name="TREE_TERM_MINPCT_NODE" value=".05"/>
      <Setting name="TREE_TERM_MINPCT_SPLIT" value=".1"/>
      <Setting name="TREE_TERM_MINREC_NODE" value="10"/>
      <Setting name="TREE_TERM_MINREC_SPLIT" value="20"/>
      <costMatrix>
        <costElement>
          <actualValue>0</actualValue>
          <predictedValue>0</predictedValue>
          <cost>0</cost>
        </costElement>
        <costElement>
          <actualValue>0</actualValue>
          <predictedValue>1</predictedValue>
          <cost>1</cost>
        </costElement>
        <costElement>
          <actualValue>0</actualValue>
          <predictedValue>2</predictedValue>
          <cost>2</cost>
        </costElement>
        <costElement>
          <actualValue>0</actualValue>
          <predictedValue>3</predictedValue>
          <cost>3</cost>
        </costElement>
        <costElement>
          <actualValue>1</actualValue>
          <predictedValue>0</predictedValue>
          <cost>1</cost>
        </costElement>
        <costElement>
          <actualValue>1</actualValue>
          <predictedValue>1</predictedValue>
          <cost>0</cost>
        </costElement>
        <costElement>
          <actualValue>1</actualValue>
          <predictedValue>2</predictedValue>
          <cost>2</cost>
        </costElement>
        <costElement>
          <actualValue>1</actualValue>
          <predictedValue>3</predictedValue>
          <cost>3</cost>
        </costElement>
```

```
          <costElement>
            <actualValue>2</actualValue>
            <predictedValue>0</predictedValue>
            <cost>3</cost>
          </costElement>
          <costElement>
            <actualValue>2</actualValue>
            <predictedValue>1</predictedValue>
            <cost>2</cost>
          </costElement>
          <costElement>
            <actualValue>2</actualValue>
            <predictedValue>2</predictedValue>
            <cost>0</cost>
          </costElement>
          <costElement>
            <actualValue>2</actualValue>
            <predictedValue>3</predictedValue>
            <cost>1</cost>
          </costElement>
          <costElement>
            <actualValue>3</actualValue>
            <predictedValue>0</predictedValue>
            <cost>3</cost>
          </costElement>
          <costElement>
            <actualValue>3</actualValue>
            <predictedValue>1</predictedValue>
            <cost>2</cost>
          </costElement>
          <costElement>
            <actualValue>3</actualValue>
            <predictedValue>2</predictedValue>
            <cost>1</cost>
          </costElement>
          <costElement>
            <actualValue>3</actualValue>
            <predictedValue>3</predictedValue>
            <cost>0</cost>
          </costElement>
        </costMatrix>
      </Extension>
      <MiningSchema>
        <MiningField name="AGE_RANGE" usageType="active"/>
        <MiningField name="ANNL_INCOME_RANGE" usageType="active"/>
        <MiningField name="CHURN_SCORE" usageType="predicted"/>
        <MiningField name="PARTY_TYPE_CODE" usageType="active"/>
      </MiningSchema>
      <Node id="0" score="2" recordCount="100427">
        <True/>
        <ScoreDistribution value="2" recordCount="46479"/>
        <ScoreDistribution value="1" recordCount="32131"/>
        <ScoreDistribution value="3" recordCount="13794"/>
        <ScoreDistribution value="0" recordCount="8023"/>
        <Node id="1" score="2" recordCount="71604">
          <CompoundPredicate booleanOperator="surrogate">
            <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
              <Array type="string">&amp;quot;20-29&amp;quot; &amp;quot;40-49&amp;quot;
&amp;quot;50-59&amp;quot; &amp;quot;70-79&amp;quot; </Array>
            </SimpleSetPredicate>
```

```
          <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;0k-39k&quot; &quot;40k-59k&quot;
&quot;80k-99k&quot; </Array>
          </SimpleSetPredicate>
        </CompoundPredicate>
        <ScoreDistribution value="2" recordCount="46479"/>
        <ScoreDistribution value="3" recordCount="13794"/>
        <ScoreDistribution value="1" recordCount="11331"/>
        <Node id="2" score="2" recordCount="43586">
          <CompoundPredicate booleanOperator="surrogate">
            <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
              <Array type="string">&quot;40-49&quot; &quot;70-79&quot; </Array>
            </SimpleSetPredicate>
            <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
              <Array type="string">&quot;Cautious Spender&quot; &quot;Mainstream
Shoppers&quot; &quot;Money and Brains&quot; </Array>
            </SimpleSetPredicate>
          </CompoundPredicate>
          <ScoreDistribution value="2" recordCount="29792"/>
          <ScoreDistribution value="3" recordCount="13794"/>
          <Node id="5" score="2" recordCount="41577">
            <CompoundPredicate booleanOperator="surrogate">
              <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
                <Array type="string">&quot;40-49&quot; </Array>
              </SimpleSetPredicate>
              <SimpleSetPredicate field="ANNL_INCOME_RANGE"
booleanOperator="isIn">
                <Array type="string">&quot;0k-39k&quot; &quot;40k-59k&quot;
&quot;60k-79k&quot; &quot;80k-99k&quot; </Array>
              </SimpleSetPredicate>
            </CompoundPredicate>
            <ScoreDistribution value="2" recordCount="27783"/>
            <ScoreDistribution value="3" recordCount="13794"/>
          </Node>
          <Node id="6" score="2" recordCount="2009">
            <CompoundPredicate booleanOperator="surrogate">
              <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
                <Array type="string">&quot;70-79&quot; </Array>
              </SimpleSetPredicate>
              <SimpleSetPredicate field="ANNL_INCOME_RANGE"
booleanOperator="isIn">
                <Array type="string">&quot;100k+&quot; </Array>
              </SimpleSetPredicate>
            </CompoundPredicate>
            <ScoreDistribution value="2" recordCount="2009"/>
          </Node>
        </Node>
        <Node id="3" score="2" recordCount="28018">
          <CompoundPredicate booleanOperator="surrogate">
            <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
              <Array type="string">&quot;20-29&quot; &quot;50-59&quot; </Array>
            </SimpleSetPredicate>
            <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
              <Array type="string">&quot;Livin Large&quot; &quot;Value
Seeker&quot; &quot;Young Professional&quot; </Array>
            </SimpleSetPredicate>
          </CompoundPredicate>
          <ScoreDistribution value="2" recordCount="16687"/>
          <ScoreDistribution value="1" recordCount="11331"/>
          <Node id="7" score="2" recordCount="15567">
```

```
            <CompoundPredicate booleanOperator="surrogate">
              <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
                <Array type="string">&quot;20-29&quot; </Array>
              </SimpleSetPredicate>
              <SimpleSetPredicate field="ANNL_INCOME_RANGE"
booleanOperator="isIn">
                <Array type="string">&quot;0k-39k&quot; &quot;80k-99k&quot;
</Array>
              </SimpleSetPredicate>
            </CompoundPredicate>
            <ScoreDistribution value="2" recordCount="10398"/>
            <ScoreDistribution value="1" recordCount="5169"/>
          </Node>
          <Node id="8" score="2" recordCount="12451">
            <CompoundPredicate booleanOperator="surrogate">
              <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
                <Array type="string">&quot;50-59&quot; </Array>
              </SimpleSetPredicate>
              <SimpleSetPredicate field="ANNL_INCOME_RANGE"
booleanOperator="isIn">
                <Array type="string">&quot;100k+&quot; &quot;40k-59k&quot;
</Array>
              </SimpleSetPredicate>
            </CompoundPredicate>
            <ScoreDistribution value="2" recordCount="6289"/>
            <ScoreDistribution value="1" recordCount="6162"/>
          </Node>
        </Node>
      </Node>
      <Node id="4" score="1" recordCount="28823">
        <CompoundPredicate booleanOperator="surrogate">
          <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;30-39&quot; &quot;60-69&quot; </Array>
          </SimpleSetPredicate>
          <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;100k+&quot; &quot;60k-79k&quot; </Array>
          </SimpleSetPredicate>
        </CompoundPredicate>
        <ScoreDistribution value="1" recordCount="20800"/>
        <ScoreDistribution value="0" recordCount="8023"/>
        <Node id="9" score="1" recordCount="16772">
          <CompoundPredicate booleanOperator="surrogate">
            <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
              <Array type="string">&quot;60-69&quot; </Array>
            </SimpleSetPredicate>
            <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
              <Array type="string">&quot;Livin Large&quot; &quot;Value
Seeker&quot; </Array>
            </SimpleSetPredicate>
          </CompoundPredicate>
          <ScoreDistribution value="1" recordCount="16772"/>
        </Node>
        <Node id="10" score="0" recordCount="12051">
          <CompoundPredicate booleanOperator="surrogate">
            <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
              <Array type="string">&quot;30-39&quot; </Array>
            </SimpleSetPredicate>
            <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
              <Array type="string">&quot;Mainstream Shoppers&quot; &quot;Young
Professional&quot; </Array>
```

```
          </SimpleSetPredicate>
        </CompoundPredicate>
        <ScoreDistribution value="0" recordCount="8023"/>
        <ScoreDistribution value="1" recordCount="4028"/>
      </Node>
    </Node>
  </Node>
 </TreeModel>
</PMML>
```

Parse XML and insert data into a table <RETWSP_TREE_CHURN_RULES> using SQL.

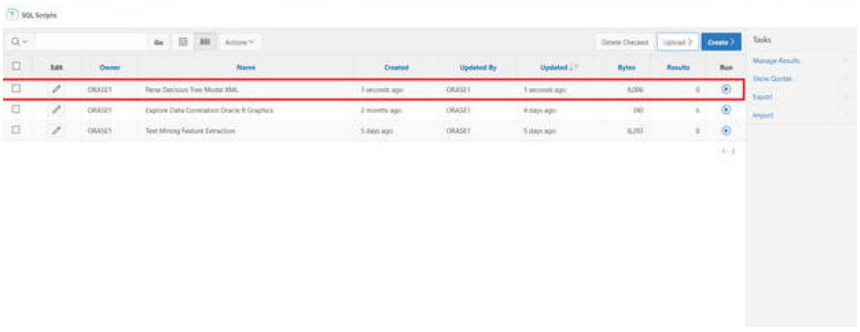
```
Create table RETWSP_TREE_CHURN_RULES as
WITH X as
(SELECT * FROM
 XMLTable('for $n in /PMML/TreeModel//Node
           let $rf :=
             if (count($n/CompoundPredicate) > 0) then
               $n/CompoundPredicate/*[1]/@field
             else
               if (count($n/SimplePredicate) > 0) then
                 $n/SimplePredicate/@field
               else
                 $n/SimpleSetPredicate/@field
           let $ro :=
             if (count($n/CompoundPredicate) > 0) then
               if ($n/CompoundPredicate/*[1] instance of
                   element(SimplePredicate)) then
                 $n/CompoundPredicate/*[1]/@operator
               else if ($n/CompoundPredicate/*[1] instance of
                   element(SimpleSetPredicate)) then
                 ("in")
               else ()
             else
               if (count($n/SimplePredicate) > 0) then
                 $n/SimplePredicate/@operator
               else if (count($n/SimpleSetPredicate) > 0) then
                 ("in")
               else ()
           let $rv :=
             if (count($n/CompoundPredicate) > 0) then
               if ($n/CompoundPredicate/*[1] instance of
                   element(SimplePredicate)) then
                 $n/CompoundPredicate/*[1]/@value
               else
                 $n/CompoundPredicate/*[1]/Array/text()
             else
               if (count($n/SimplePredicate) > 0) then
                 $n/SimplePredicate/@value
               else
                 $n/SimpleSetPredicate/Array/text()
           let $sf :=
             if (count($n/CompoundPredicate) > 0) then
               $n/CompoundPredicate/*[2]/@field
             else ()
           let $so :=
             if (count($n/CompoundPredicate) > 0) then
               if ($n/CompoundPredicate/*[2] instance of
                   element(SimplePredicate)) then
```

```
                    $n/CompoundPredicate/*[2]/@operator
                  else if ($n/CompoundPredicate/*[2] instance of
                      element(SimpleSetPredicate)) then
                    ("in")
                  else ()
                else ()
              let $sv :=
                if (count($n/CompoundPredicate) > 0) then
                  if ($n/CompoundPredicate/*[2] instance of
                      element(SimplePredicate)) then
                    $n/CompoundPredicate/*[2]/@value
                  else
                    $n/CompoundPredicate/*[2]/Array/text()
                else ()
              return
                <pred id="{$n/../@id}"
                      score="{$n/@score}"
                      rec="{$n/@recordCount}"
                      cid="{$n/@id}"
                      rf="{$rf}"
                      ro="{$ro}"
                      rv="{$rv}"
                      sf="{$sf}"
                      so="{$so}"
                      sv="{$sv}"
                />'
      passing dbms_data_mining.get_model_details_xml('SL_DT_Cchurn_MODEL')
            COLUMNS
              parent_node_id   NUMBER PATH '/pred/@id',
              child_node_id    NUMBER PATH '/pred/@cid',
              rec              NUMBER PATH '/pred/@rec',
              score            VARCHAR2(4000) PATH '/pred/@score',
              rule_field       VARCHAR2(4000) PATH '/pred/@rf',
              rule_op          VARCHAR2(20) PATH '/pred/@ro',
              rule_value       VARCHAR2(4000) PATH '/pred/@rv',
              surr_field       VARCHAR2(4000) PATH '/pred/@sf',
              surr_op          VARCHAR2(20) PATH '/pred/@so',
              surr_value       VARCHAR2(4000) PATH '/pred/@sv'))
select pid parent_node, nid node, rec record_count,
      score prediction, rule_pred local_rule, surr_pred local_surrogate,
      rtrim(replace(full_rule,'$O$D$M$'),' AND') full_simple_rule from (
select row_number() over (partition by nid order by rn desc) rn,
 pid, nid, rec, score, rule_pred, surr_pred, full_rule from (
 select rn, pid, nid, rec, score, rule_pred, surr_pred,
   sys_connect_by_path(pred, '$O$D$M$') full_rule from (
  select row_number() over (partition by nid order by rid) rn,
    pid, nid, rec, score, rule_pred, surr_pred,
    nvl2(pred,pred || ' AND ',null) pred from(
   select rid, pid, nid, rec, score, rule_pred, surr_pred,
     decode(rn, 1, pred, null) pred from (
    select rid, nid, rec, score, pid, rule_pred, surr_pred,
     nvl2(root_op, '(' || root_field || ' ' || root_op || ' ' || root_value ||
')', null) pred,
     row_number() over (partition by nid, root_field, root_op order by rid desc)
rn from (
     SELECT
       connect_by_root(parent_node_id) rid,
       child_node_id nid,
       rec, score,
       connect_by_root(rule_field) root_field,
```

```
       connect_by_root(rule_op) root_op,
       connect_by_root(rule_value) root_value,
       nvl2(rule_op, '(' || rule_field || ' ' || rule_op || ' ' || rule_value ||
')',  null) rule_pred,
       nvl2(surr_op, '(' || surr_field || ' ' || surr_op || ' ' || surr_value ||
')',  null) surr_pred,
       parent_node_id pid
       FROM (
        SELECT parent_node_id, child_node_id, rec, score, rule_field, surr_field,
rule_op, surr_op,
               replace(replace(rule_value,'" "', ''', '''),'"', '''') rule_value,
               replace(replace(surr_value,'" "', ''', '''),'"', '''') surr_value
        FROM (
          SELECT parent_node_id, child_node_id, rec, score, rule_field, surr_
field,
                 decode(rule_op,'lessOrEqual','<=','greaterThan','>',rule_op)
rule_op,
                 decode(rule_op,'in','('||rule_value||')',rule_value) rule_value,
                 decode(surr_op,'lessOrEqual','<=','greaterThan','>',surr_op)
surr_op,
                 decode(surr_op,'in','('||surr_value||')',surr_value) surr_value
          FROM X)
       )
       CONNECT BY PRIOR child_node_id = parent_node_id
     )
    )
   )
  )
  CONNECT BY PRIOR rn = rn - 1
         AND PRIOR nid = nid
  START WITH rn = 1
)
)
where rn = 1;
```

This sql is also listed in SQL Scripts.

Figure A–49 SQL Listed SQL Script



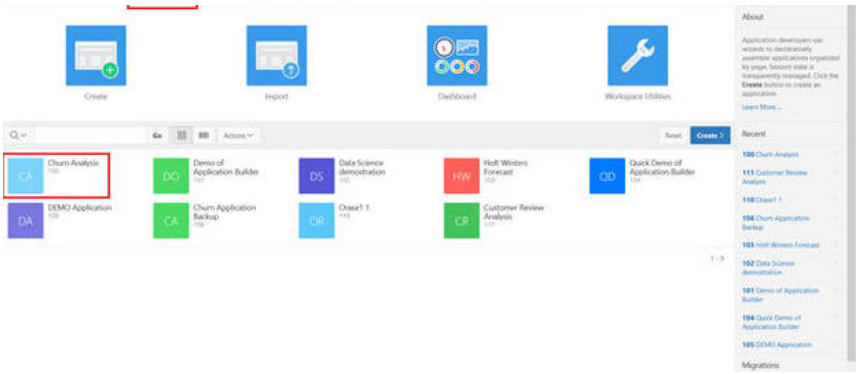
Save the churn rules by creating table RETWSP_TREE_CHURN_RULES as (<sql above>).

Display Decision Tree Rules

Create an application to display these rules.

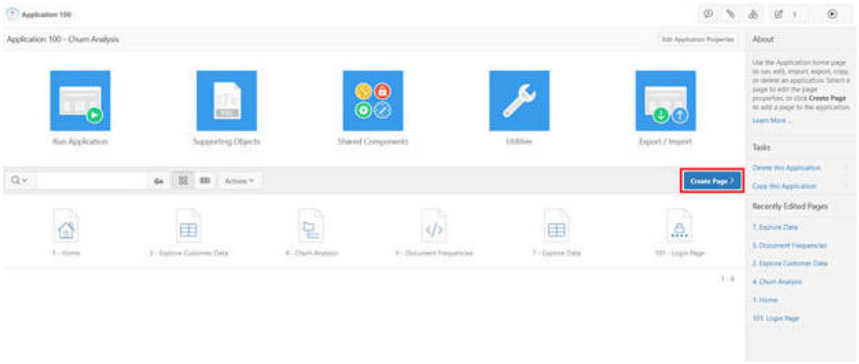
1. Select **Churn Analysis** from App Builder.

Figure A–50 Select Churn Analysis



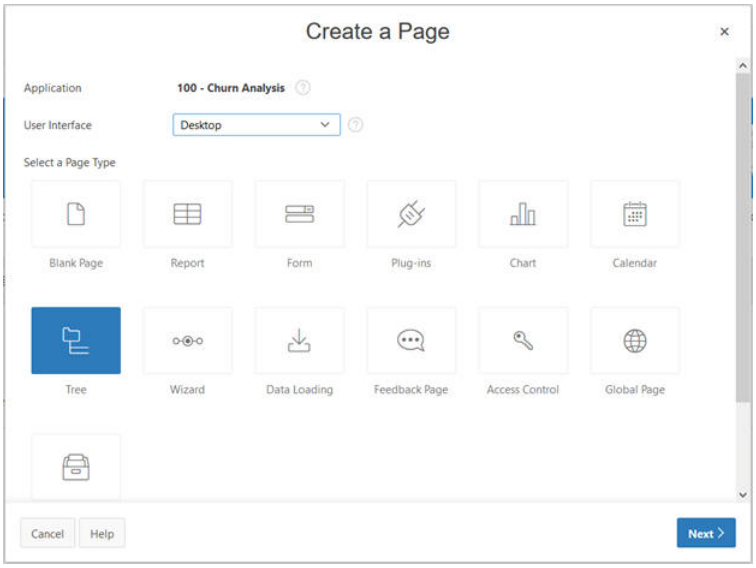
2. Create a page.

Figure A–51 Create Page



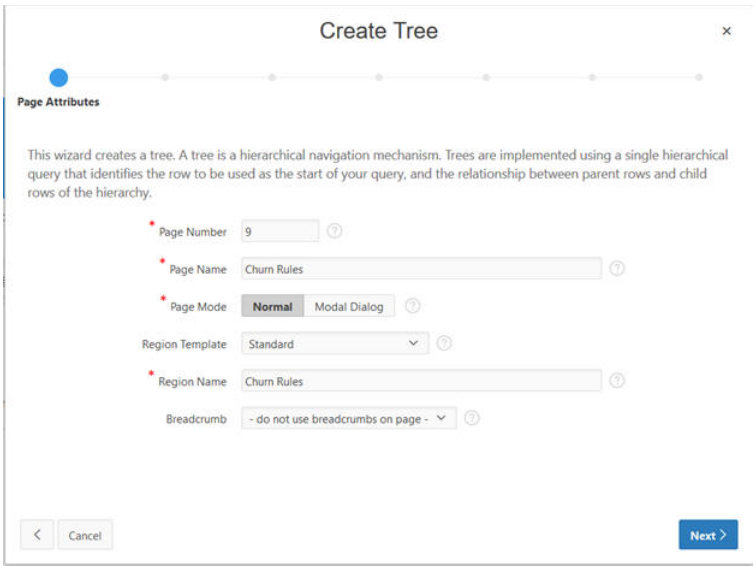
3. Select the Tree to display.

Figure A–52 Select Tree



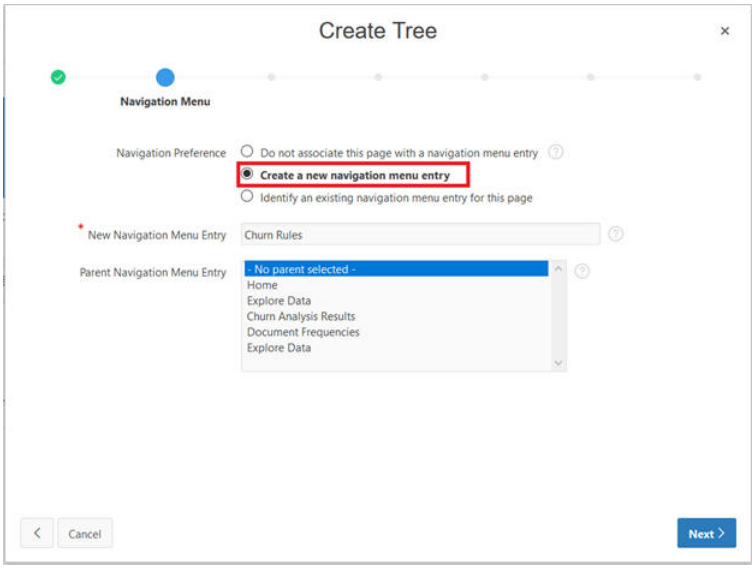
4. Populate Page Name as Churn Rules and click **Next**.

Figure A–53 Page Attributes



5. Select **Create a new navigation menu entry**.

Figure A–54 New Navigation Menu Entry



6. Select **RETWSP_TREE_CHURN_RULES** from Table/View name list.

Figure A–55 Table/View Name

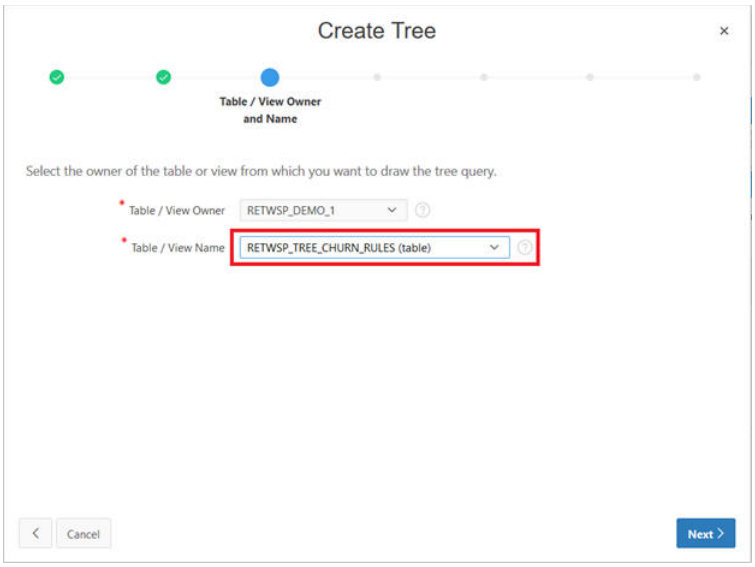


Figure A–56 Tree Query

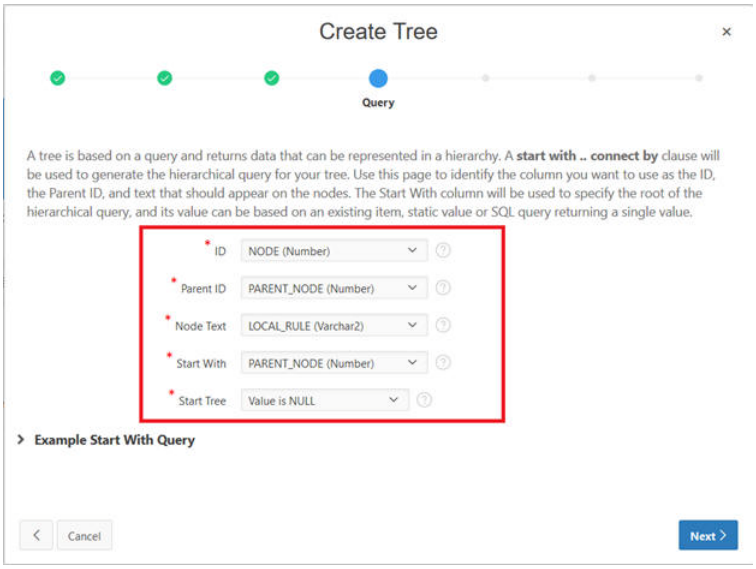


Figure A–57 Where Clause

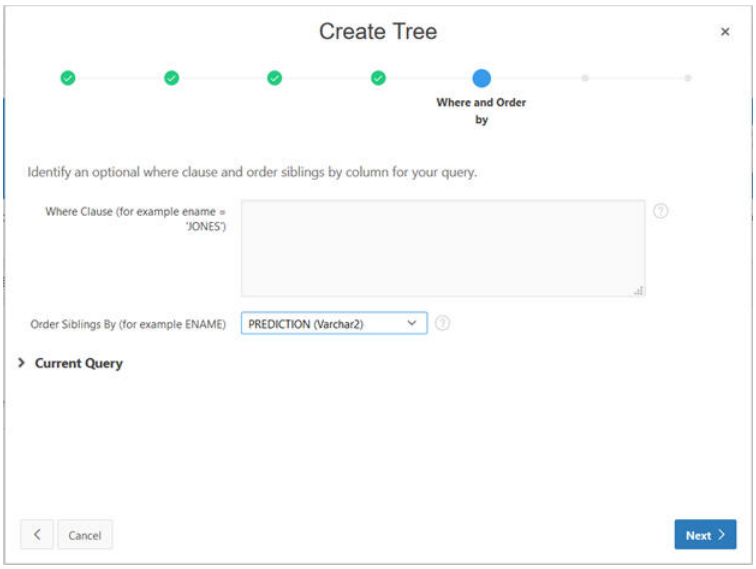


Figure A–58 Tree Attributes

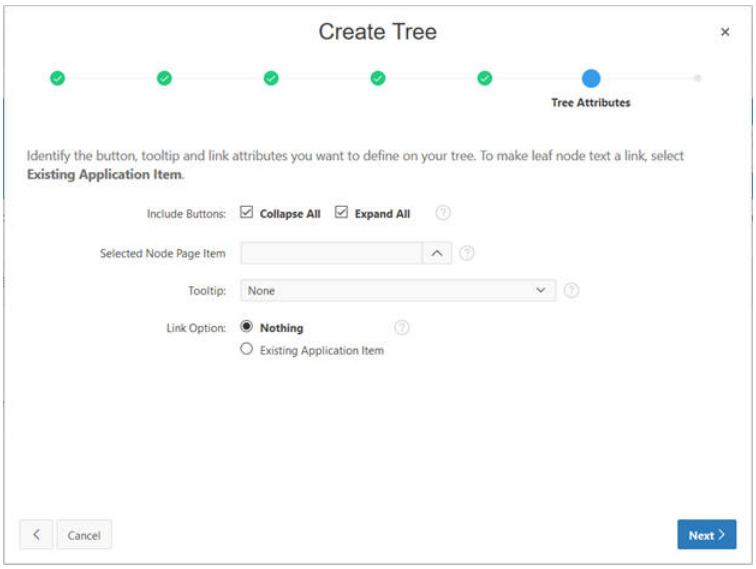
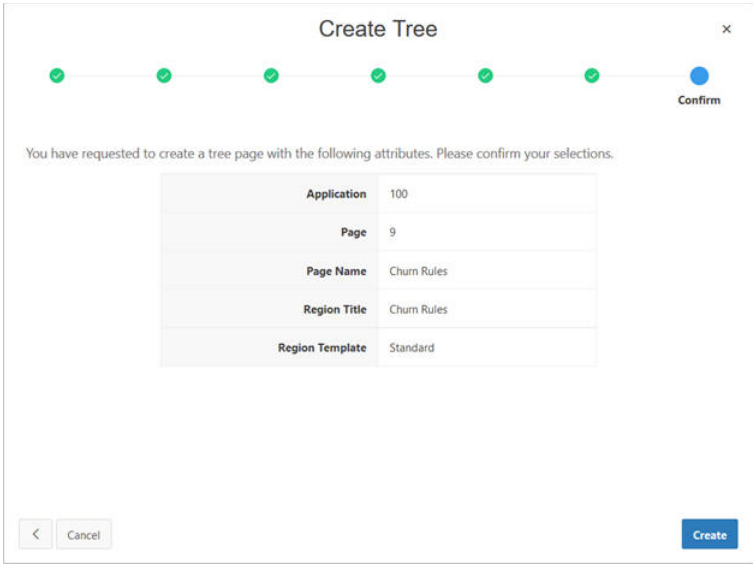
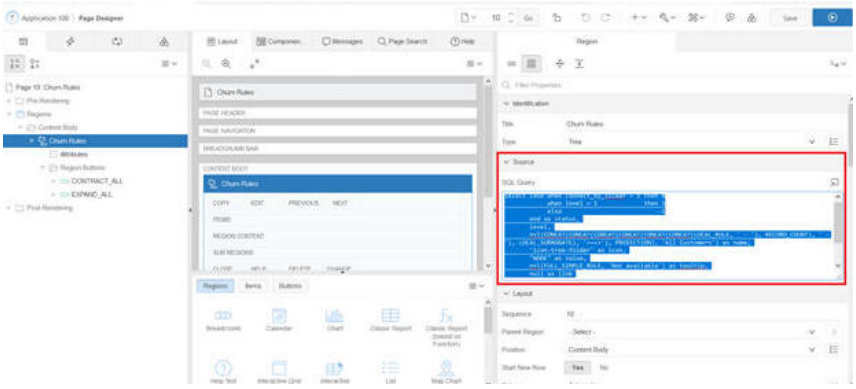


Figure A–59 Tree Confirmation



7. You see Page Designer.

Figure A–60 Page Designer

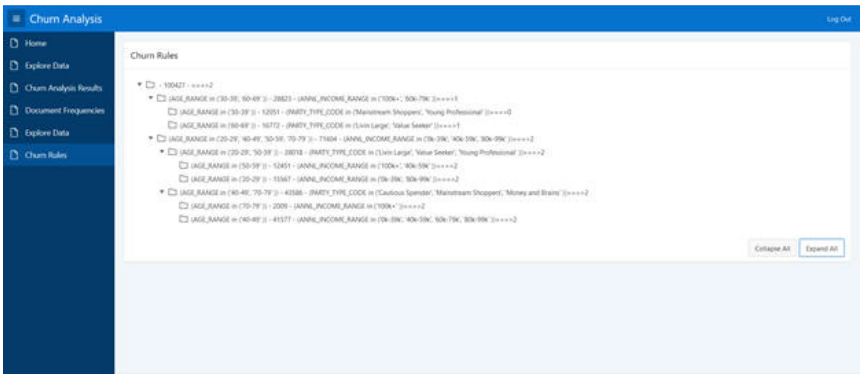


8. Replace the following query to better format and execute the run.

```
select case when connect_by_isleaf = 1 then 0
            when level = 1             then 1
            else                            -1
       end as status,
       level,
       nvl(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(LOCAL_RULE, ' - '),
RECORD_COUNT), ' - '), LOCAL_SURROGATE), '===>'), PREDICTION), 'All Customers')
as name,
       'icon-tree-folder' as icon,
       "NODE" as value,
       nvl(FULL_SIMPLE_RULE, 'Not available') as tooltip,
       null as link
from "#OWNER#"."RETWSP_TREE_CHURN_RULES"
start with "PARENT_NODE" is null
connect by prior "NODE" = "PARENT_NODE"
order siblings by "RECORD_COUNT"
```

9. Run a new window

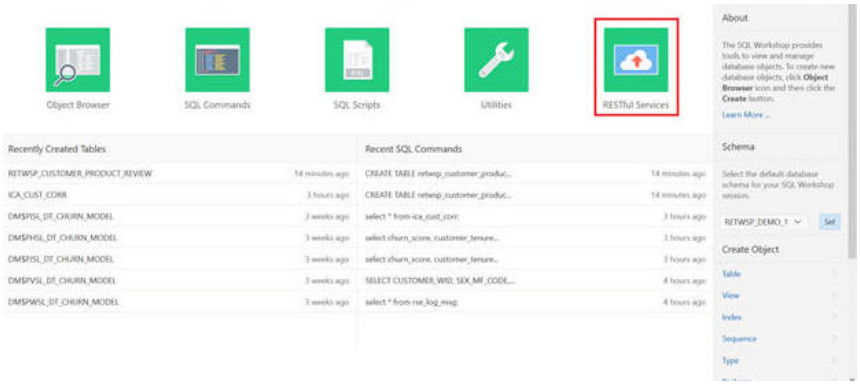
Figure A–61 Run New Window



Lab Share Insights - ReST API

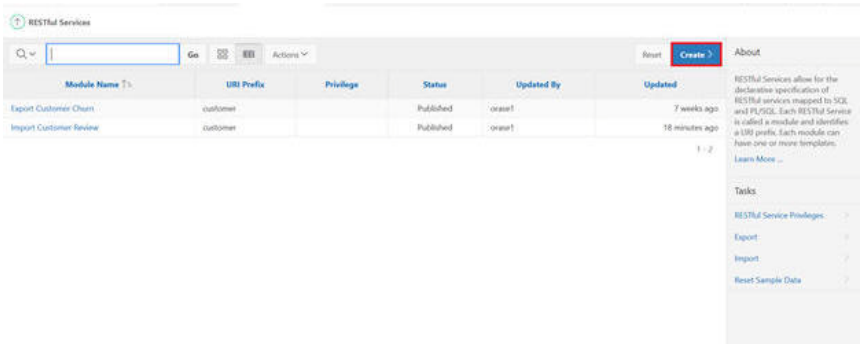
1. View RESTful Services
2. Create ReST API, which you can use to load customer reviews in the above table.

Figure A–62 ReST API



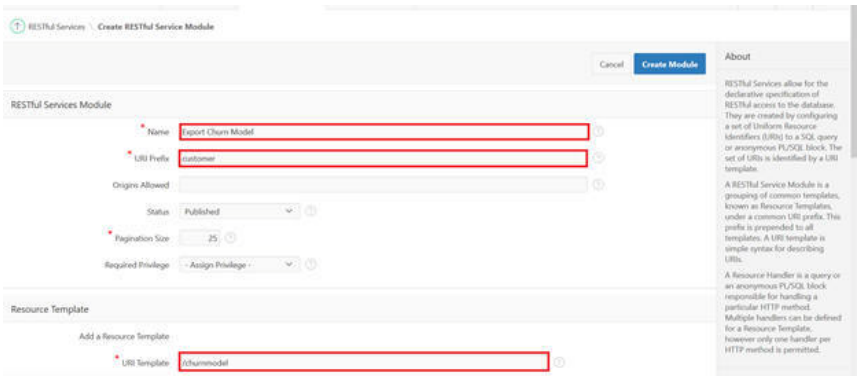
3. Click **Create**.

Figure A–63 Create



4. Populate with Name, URI Prefix, and URI Template.

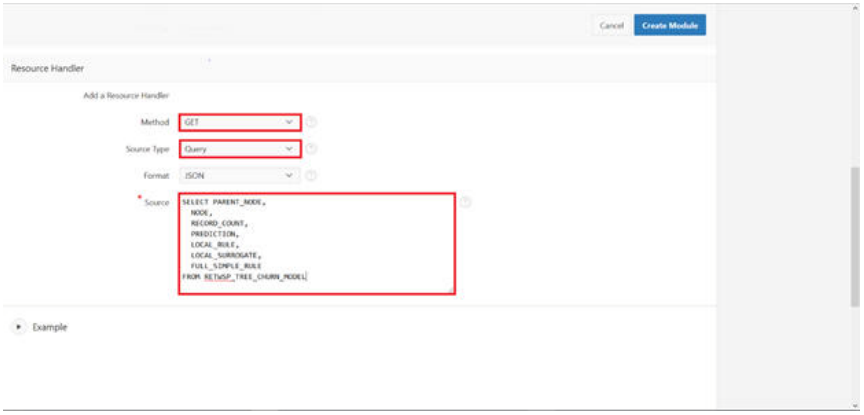
Figure A–64 Populate ReST Fields



5. Select **GET** and populate the SQL as:

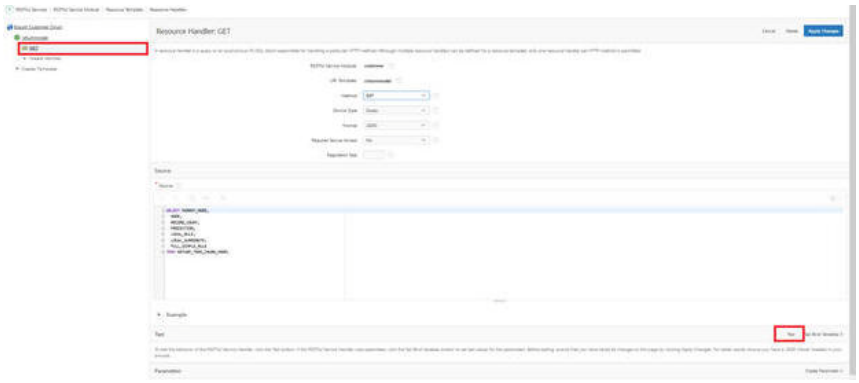
```
SELECT PARENT_NODE,
  NODE,
  RECORD_COUNT,
  PREDICTION,
  LOCAL_RULE,
  LOCAL_SURROGATE,
  FULL_SIMPLE_RULE
FROM RETWSP_TREE_CHURN_MODEL
```

Figure A–65 Get



6. Select **GET** and then click **Test**.

Figure A–66 Click Test



7. Data is displayed in the window and any request to this end point http:/<url> will return the data shown in Figure A–67 in JSON format.

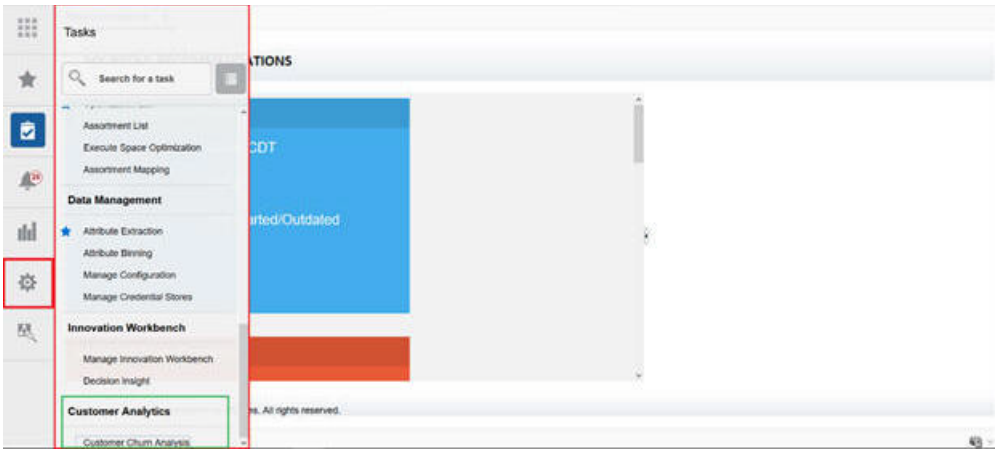
Figure A–67 Data

Lab Share Insights - Task Navigation

This exercise illustrates how to add a link to the application Task Navigation.

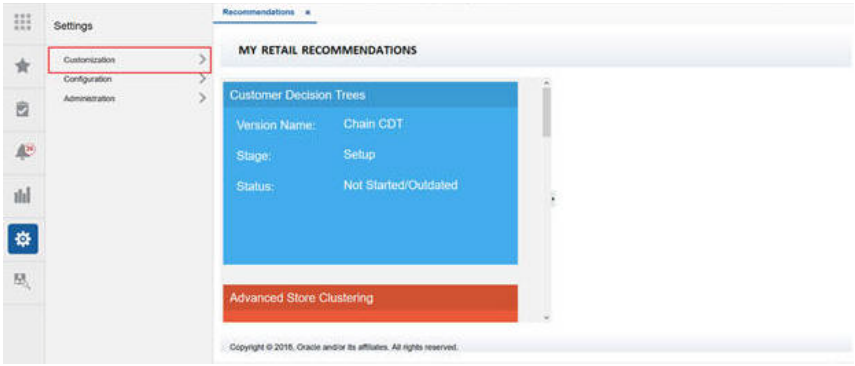
1. As an INSIGHT_APPLICATION_ADMINISTATOR, you can add a new task to the Tasks.

Figure A–68 Tasks



2. Click **Customization**.

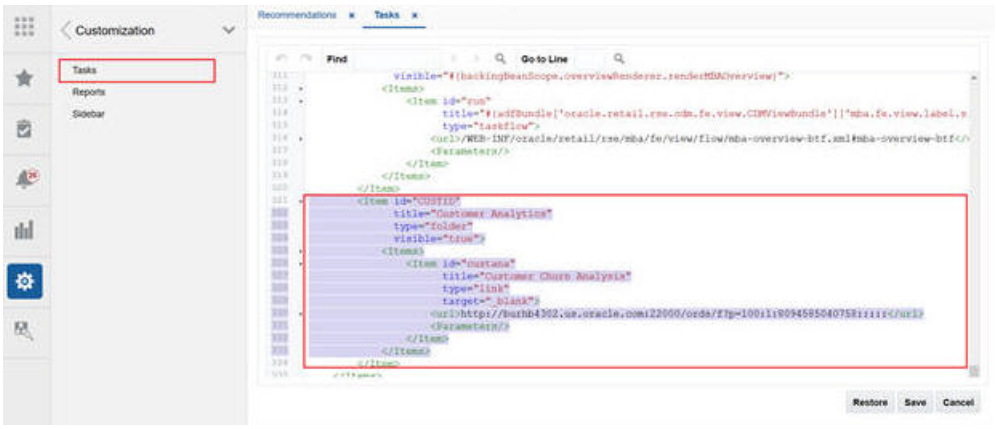
Figure A–69 Customization



3. The Content Area displays Navigation XML. At the end of the XML, add the following content (see Customer Analytics). Update the highlighted contents.

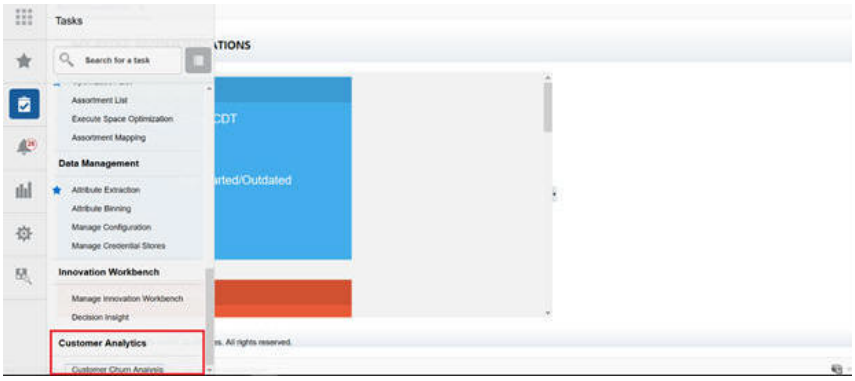
```
<Item id="CUSTID"
      title="Customer Analytics"
      type="folder"
      visible="true">
    <Items>
        <Item id="custana"
              title="Customer Churn Analysis"
              type="link"
              target="_blank">
            <url>http://<url>/ords/f?p=100:1:8094585040758:::::</url>
            <Parameters/>
        </Item>
    </Items>
</Item>
```

Figure A–70 Highlighted SQL



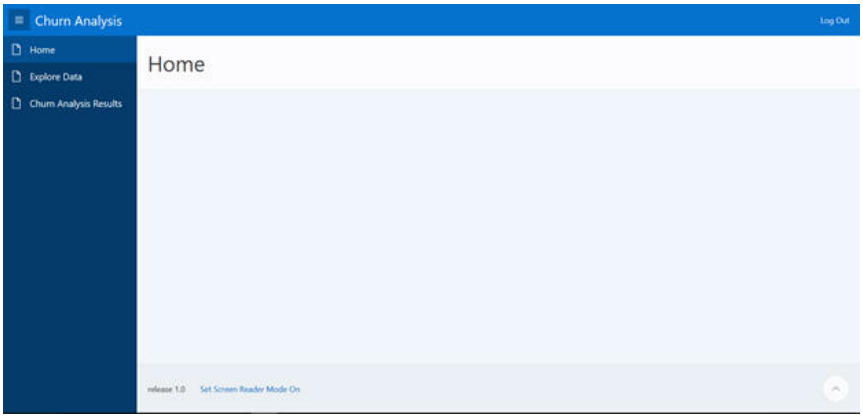
4. Click **Save**. Log out and then log back in. You should see:

Figure A–71 After Logout



5. Click the Customer Churn Analysis link system. A new tab is launched in the browser for the end user.

Figure A–72 New Tab



Database Tools

How to Allocate Privileges

None of the tables in Retailer Workspace Schema are accessible in the Application Schema.

Executing queries in the Application Schema will result in an error. Until and unless access is provided, the user will not be able to access object/data in another schema.

```
describe retwsp_demo_1.p_churn_model;
ERROR:
ORA-04043: object retwsp_demo_1.p_churn_model does not exist

select * from RETWSP_TREE_CHURN_RULES;
ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:
Error at Line: 1 Column: 15
```

How to Execute a Job using DBMS Scheduler

Long running jobs must be executed under RETAILER_WORKSPACE_JOBS. These are executed in resource groups dedicated to the retailer schema.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
job_name => 'retwsp_churn_model',
job_type => 'STORED_PROCEDURE',
job_action => 'retwsp.pkg_customer_analytics.proc_churn_model',
start_date => '01-JAN-17 07.00.00 PM US/Pacific',
repeat_interval => 'FREQ=YEARLY; BYDATE=0331,0630,0930,1231; ',
end_date => '31-DEC-17 07.00.00 PM US/Pacific',
job_class => 'RETAILER_WORKSPACE_JOBS',
comments => 'Retailer workspace churn model job');
END;
/
```

Glossary of Acronyms

AA

Affinity Analysis.

AC

Advanced Clustering, also known as CIS.

ASO

Oracle Retail Assortment and Space Optimization.

BI

Business Intelligence.

CDT

Customer Decision Tree.

DB

Database.

DT

Demand Transference.

MDS

Oracle MetaData Services.

POG Hierarchy

Defined by three levels: POG department, POG category, and POG subcategory. The POG hierarchy is used to organize POGs within POG sets. For example, a leaf to root path in the POG hierarchy: Grocery -> Snacks -> Crackers.

POG Node

A leaf node (POG subcategory) within a POG set.

POG Set

Historical POGs in the same POG subcategory and with the same seasonal attribute. An ASO term.

RI

Retail Insights, formerly known as Retail Analytics.

RADM

Retail Analytics Data Model, also known as RI Schema.

RCM

Oracle Retail Category Management.

RDF

Oracle Retail Demand Forecasting.

Seasonal Attribute

Refers to a specific year independent time period for an APO assortment and a POG set. Examples include Spring, holiday, back to school, year-round. (Also, Season Attribute.)