

Oracle® Cloud

Working with Data Modeler in Oracle Analytics Cloud



F25036-22
July 2023



Oracle Cloud Working with Data Modeler in Oracle Analytics Cloud,

F25036-22

Copyright © 2020, 2023, Oracle and/or its affiliates.

Primary Author: Rosie Harvey

Contributing Authors: Suzanne Gill, Pete Brownbridge, Stefanie Rhone, Hemala Vivek, Padma Rao

Contributors: Oracle Analytics development, product management, and quality assurance teams

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	vii
Documentation Accessibility	vii
Diversity and Inclusion	viii
Related Documents	viii
Conventions	viii

1 About Modeling Enterprise Data

Data Modeler Available for a Limited Time	1-1
Oracle Analytics Data Modeling Tools	1-1

2 Get Started with Data Modeler

Typical Workflow to Model Data Using Data Modeler	2-1
Open Data Modeler	2-2
Top Tasks for Data Modeler	2-3

3 Understand Data Modeling

About Modeling Data with Data Modeler	3-1
Plan a Semantic Model	3-2
Understand Semantic Model Requirements	3-2
Components of a Semantic Model	3-3
About Modeling Source Objects with Star Relationships	3-3
About Modeling Source Objects with Snowflake Relationships	3-4
About Modeling Denormalized Sources	3-4
About Modeling Normalized Sources	3-4

4 Start to Build Your Semantic Model

Use Data Modeler	4-1
Create a Semantic Model	4-2

Use the Left Pane in Data Modeler	4-2
Use the Right Pane in Data Modeler	4-3
Use Action Menus	4-4
Lock a Semantic Model	4-5
Validate a Semantic Model	4-5
Refresh and Synchronize Source Objects and Semantic Model Objects	4-5
Publish Changes to Your Semantic Model	4-7
Clear Cached Data	4-8
Rename a Semantic Model	4-8
Connect a Model to a Different Database	4-9
Export a Semantic Model	4-10
Import a Semantic Model	4-10
Delete a Semantic Model	4-10
Review Source Tables and Data	4-11
View Source Objects	4-11
Preview Data in Source Objects	4-11
Create Source Views	4-12
About Source Views	4-12
Add Your Own Source Views	4-13
Define Filters for Source Views	4-15
Add Fact Tables and Dimension Tables to a Semantic Model	4-15
About Fact Tables and Dimension Tables	4-16
Create Fact and Dimension Tables from a Single Table or View	4-16
Create Fact Tables Individually	4-18
Create Dimension Tables Individually	4-19
Edit Fact Tables and Dimension Tables	4-20
Add More Columns to Fact and Dimension Tables	4-21
Add Columns from Another Source to a Dimension Table	4-22
Join Tables in a Semantic Model	4-22
About Joins	4-23
Join Fact and Dimension Tables	4-23
Create a Time Dimension	4-23
Add Measures and Attributes to a Semantic Model	4-25
Edit Measures and Attributes	4-25
Specify Aggregation for Measures in Fact Tables	4-26
Create Calculated Measures	4-28
About Creating Calculated Measures	4-29
Create Derived Attributes	4-31
Create Expressions in the Expression Editor	4-31
About the Expression Editor	4-31
Create an Expression	4-32

Copy Measures and Attributes	4-33
Copy Model Objects	4-33

5 Define Hierarchies and Levels to Drill and Aggregate

Typical Workflow to Define Hierarchies and Levels	5-1
About Hierarchies and Levels	5-1
Edit Hierarchies and Levels	5-2
Set Dimension Table Properties for Hierarchies	5-2
Set Aggregation Levels for Measures	5-3
About Setting Aggregation Levels for Measures	5-3

6 Secure Your Semantic Model

Typical Workflow to Secure Model Data	6-1
Create Variables to Use in Expressions	6-1
About Variables	6-1
Define Variables	6-2
Secure Access to Objects in the Model	6-3
About Permission Inheritance	6-4
Secure Access to Data	6-5

7 Expression Editor Reference

Semantic Model Objects	7-1
SQL Operators	7-1
Conditional Expressions	7-3
Functions	7-5
Aggregate Functions	7-5
Analytics Functions	7-9
Date and Time Functions	7-10
Date Extraction Functions	7-11
Conversion Functions	7-13
Display Functions	7-14
Evaluate Functions	7-16
Mathematical Functions	7-16
Running Aggregate Functions	7-18
Spatial Functions	7-19
String Functions	7-20
System Functions	7-24
Time Series Functions	7-24
Constants	7-26

Types
Variables

7-27
7-27

Preface

Learn how to model data in Oracle Analytics Cloud using Data Modeler.

Topics:



Note:

Data Modeler is available for a limited time. Consider moving to Semantic Modeler as soon as possible.

- [Audience](#)
- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Related Documents](#)
- [Conventions](#)

Audience

Working with Data Modeler in Oracle Analytics Cloud is intended for business intelligence analysts and administrators who use Oracle Analytics Cloud:

- **Analysts** model enterprise data and create workbooks, analyses, dashboards, and pixel-perfect reports for consumers. Analysts can select interactive visualizations and create advanced calculations to reveal insights in the data.
- **Administrators** edit and upload data models built using Oracle BI Enterprise Edition or Oracle Analytics Server to Oracle Analytics Cloud. Analysts use the data models to build workbooks, analyses, dashboards, and pixel-perfect reports.

Documentation Accessibility

Oracle is committed to 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 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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For a full list of guides, refer to the Books tab on Oracle Analytics Cloud Help Center.

- <http://docs.oracle.com/en/cloud/paas/analytics-cloud/books.html>

Conventions

This document uses the standard Oracle text and image conventions.

Text Conventions

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	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.

Videos and Images

Skins and styles customize the look and feel of Oracle Analytics Cloud, dashboards, reports, and other objects. Videos and images used in this guide may not have the same skin or style that you're using, but the behavior and techniques shown are the same.

1

About Modeling Enterprise Data

Oracle Analytics Cloud offers several tools for modeling your enterprise data.

Topics:

- [Data Modeler Available for a Limited Time](#)
- [Oracle Analytics Data Modeling Tools](#)

Data Modeler Available for a Limited Time

Data Modeler is available only for a limited time. Consider moving to Semantic Modeler as soon as possible.

Semantic Modeler is a browser-based modeling tool that you use for creating, building, and deploying a semantic model. The Semantic Modeler editor is a fully-integrated Oracle Analytics component.

Use Semantic Modeler to create semantic models. See [Create an Empty Semantic Model](#).

If you're working with existing data models, you can migrate your data models to Semantic Modeler. For help with this migration, see [Plan Your Migration to Semantic Modeler](#).

For information about all available modeling tools, see [Oracle Analytics Data Modeling Tools](#).

Oracle Analytics Data Modeling Tools

Oracle Analytics offers several data modeling tools that you can use to create enterprise semantic models and self-service datasets.

Use this topic to learn the differences between the data modeling tools and which tool to use based on the type of data model that you want to create.

Tool	Use to create	Description
Semantic Modeler	Governed data models	<p>A browser-based modeling tool that developers use for creating, building, and deploying the semantic model to an .rpd file. The Semantic Modeler editor is a fully-integrated Oracle Analytics component.</p> <p>Because the Semantic Modeler generates Semantic Model Markup Language (SMML) to define semantic models, developers have the choice of using the Semantic Model editor, the native SMML editor, or another editor to develop semantic models. Semantic Modeler provides full Git integration to support multi-user development.</p> <p>You can use the Semantic Modeler to create semantic models from the data sources that it supports. Use the Model Administration Tool to create semantic models from data sources that Semantic Modeler doesn't support.</p> <p>See What Is Oracle Analytics Semantic Modeler? and Data Sources Available for Data Modeling.</p>

Tool	Use to create	Description
Model Administration Tool	Governed data models	<p>A mature, longstanding, heavyweight, developer-focused modeling tool that provides complete governed data modeling capabilities. Developers use the Model Administration Tool to define rich business semantics, data governance, and data interaction rules to fetch, process, and present data at different granularity from disparate data systems.</p> <p>Oracle recommends that you use the Semantic Modeler to create semantic models from the data sources Semantic Modeler supports, and that you use the Model Administration Tool to create semantic models from any data source that Semantic Modeler doesn't support. See About Creating Semantic Models with Model Administration Tool and Data Sources Available for Data Modeling.</p> <p>The Model Administration Tool is a Windows-based application that isn't integrated into the Oracle Analytics Cloud interface. You download the Model Administration Tool and install it onto and use it from your computer.</p> <p>If you previously modeled your business data with Oracle BI Enterprise Edition or Oracle Analytics Server, you don't have to start from scratch in Oracle Analytics Cloud. You can use the Model Administration Tool to upload a complete semantic model .rpd file to Oracle Analytics Cloud and immediately start using your subject areas in visualizations, dashboards, and analyses.</p> <p>Optionally, can use the Model Administration Tool to download, edit, and upload your semantic model .rpd files to Oracle Analytics Cloud.</p> <p>See Build Semantic Models Using Model Administration Tool.</p>
Data Model Editor	XML data structure for Pixel-Perfect Reports	<p>The Data Model editor enables you to combine data from multiple datasets into a single XML data structure for pixel-perfect reports.</p> <p>See Build Data Models for Pixel-Perfect Reports.</p>
Dataset Editor	Self-service data models	<p>A user-friendly data modeling and data preparation tool that data analysts and business analysts use to create datasets containing multiple tables with joins. A dataset can contain data from local and remote files, including more than 50 connections and subject areas.</p> <p>The Dataset editor is available from the Oracle Analytics interface and enables business users to create self-service data models on top of existing governed semantic models.</p> <p>See What Are Datasets?</p>

2

Get Started with Data Modeler

This topic describes how you access and begin working with Data Modeler.



Note:

Data Modeler is available for a limited time.

Intead of using Data Modeler, Oracle advises that you use Semantic Modeler to create semantic models. See [Create an Empty Semantic Model](#).

If you're working with existing data models, Oracle advises that you migrate them to Semantic Modeler. For help with this migration, see [Import the Semantic Model From Data Modeler](#).

Topics:

- [Typical Workflow to Model Data Using Data Modeler](#)
- [Open Data Modeler](#)
- [Top Tasks for Data Modeler](#)

Typical Workflow to Model Data Using Data Modeler

Here are the common tasks for modeling data with Data Modeler.

Task	Description	More Information
Read about Data Modeler	Get familiar with Data Modeler, including how to refresh your data, publish changes, and find the Action menus.	Use Data Modeler
Create a new model	Start a new model and connect it to your data source.	Create a Semantic Model
Browse source objects	Review source tables to determine how to structure your semantic model.	Review Source Tables and Data
Create new views in the database if needed	Create views for role-playing dimensions, or create views to combine multiple tables into a single view, as in snowflake or normalized sources.	Add Your Own Source Views
Add fact tables and dimension tables	Create fact tables and dimension tables from source objects.	Add Fact Tables and Dimension Tables to a Semantic Model
Join fact and dimension tables	Create joins between fact and dimension tables.	Join Fact and Dimension Tables
Add a time dimension	Create a time dimension table and database source table with time data.	Create a Time Dimension

Task	Description	More Information
Add aggregated and calculated measures	Specify aggregation for columns and create calculated measures using expressions.	Add Measures and Attributes to a Semantic Model
Add derived attributes	Specify custom attributes for dimension tables using expressions.	Create Derived Attributes
Create hierarchies and levels	Define hierarchies and levels based on relationships between groups of attribute columns.	Edit Hierarchies and Levels
Create variables	Optionally, create variables that dynamically calculate and store values for use in column expressions and data filters	Define Variables
Set up object permissions	Control who can access fact tables, dimension tables, and columns.	Secure Access to Objects in the Model
Set up data security filters	Define row-level data security filters for fact tables, dimension tables, and columns.	Secure Access to Data
Upload a semantic model .rpd file	If you've modeled your business data with Oracle Analytics Server, instead of building a semantic model from scratch using Data Modeler, you can use Console to upload your semantic model to the cloud.	Upload Semantic Models from a .rpd File Using Console

Open Data Modeler

Your administrator gives you access to Data Modeler.



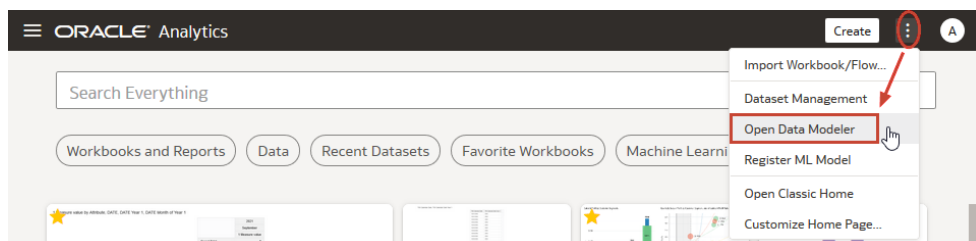
Note:

Data Modeler is available for a limited time.

Instead of using Data Modeler, Oracle advises that you use Semantic Modeler to create semantic models. See [Create an Empty Semantic Model](#).

If you're working with existing data models, Oracle advises that you migrate them to Semantic Modeler. For help with this migration, see [Import the Semantic Model From Data Modeler](#).

1. Sign in to Oracle Analytics Cloud.
2. Click the **Page** menu on the Home page, and select **Open Data Modeler**.



3. At the Data Modeler Migration page, click **Launch Data Modeler**.
4. At the Models page, open an existing model or create a new model.

Top Tasks for Data Modeler

The top tasks for data modeling with Data Modeler are identified in this topic.

- [Create a Semantic Model](#)
- [Review Source Tables and Data](#)
- [Add Your Own Source Views](#)
- [Create Fact and Dimension Tables from a Single Table or View](#)
- [Create Fact Tables Individually](#)
- [Create Dimension Tables Individually](#)
- [Join Fact and Dimension Tables](#)
- [Create Calculated Measures](#)
- [Create Derived Attributes](#)
- [Create a Time Dimension](#)
- [Edit Hierarchies and Levels](#)
- [Secure Access to Objects in the Model](#)
- [Publish Changes to the Data Model](#)

3

Understand Data Modeling

You build a model of your business data to enable analysts to structure queries in the same intuitive fashion as they ask business questions.

Topics:

- [About Modeling Data with Data Modeler](#)
- [Plan a Semantic Model](#)

About Modeling Data with Data Modeler

A semantic model is a design that presents business data for analysis in a manner that reflects the structure of the business. Semantic models enable analysts to structure queries in the same intuitive fashion as they ask business questions. Well-designed models are simple and mask the complexity of the underlying data structure.

Using Data Modeler you can model data from various source types, such as star and snowflake, in various ways that make sense to business users. You must have the BI Data Model Author role to use Data Modeler.



Note:

If you modeled your business data with Oracle BI Enterprise Edition, you don't have to start from scratch with Data Modeler. You can use the Model Administration Tool to upload your semantic model .rpd file to the cloud. See [Upload Semantic Model Files from Oracle BI Enterprise Edition or Oracle Analytics Server](#).

Although not all source objects have star relationships, Data Modeler presents data as a simple star structure in the semantic model. In other words, the semantic model represents measurable facts that are viewed in terms of various dimensional attributes.

When building a semantic model with Data Modeler, you perform the following tasks:

- Connect to the database containing your business data.
- Add source tables or views to the model and classify them as either a fact table or a dimension table.
- Define joins between fact and dimension tables
- Ensure that every dimension table maps to at least one fact table, and that every fact table maps to at least one dimension table.
- Specify aggregation rules for different fact columns, create derived measures based on expressions, create dimension hierarchies to support drilling, and create level-based measures.

- Publish your semantic model to permanently save the changes and make the data available for use in analyses.

After publishing your semantic model, you can start visualizing your data from your enterprise reporting Home page. Your semantic model displays as a subject area that you can use in visualizations, dashboards, and analyses. The name of the subject area matches the name of your semantic model.

When you model source objects with multiple star relationships, they're all part of the same semantic model and are included in the same subject area.

Can I Use My Existing Semantic Model .rpd File vs Data Modeler

Yes. This chapter describes how to create semantic models from scratch using Data Modeler. If you modeled your business data with Oracle BI Enterprise Edition, you can upload the complete semantic model .rpd to Oracle Analytics Cloud and immediately start using your subject areas in visualizations, dashboards, and analyses. See Upload Semantic Model Files from Oracle BI Enterprise Edition or Oracle Analytics Server.

If you upload an existing semantic model file in this way:

- Data Modeler is disabled.
You see the message "Please use Oracle BI Administration Tool to manage your model".
- You use Model Administration Tool to make the changes.
See Edit a Semantic Model in the Cloud.

Plan a Semantic Model

Before you start modeling your data, take some time to think about your business requirements and to understand data modeling concepts.

Topics:

- [Understand Semantic Model Requirements](#)
- [Components of a Semantic Model](#)
- [About Modeling Source Objects with Star Relationships](#)
- [About Modeling Source Objects with Snowflake Relationships](#)
- [About Modeling Denormalized Sources](#)
- [About Modeling Normalized Sources](#)

Understand Semantic Model Requirements

Before you can begin to model data, you must first understand your semantic model's requirements:

- What kinds of business questions are you trying to answer?
- What are the measures required to understand business performance?
- What are all the dimensions under which the business operates? Or, in other words, what are the dimensions used to break down the measurements and provide headers for the reports?

- Are there hierarchical elements in each dimension, and what types of relationships define each hierarchy?

After you have answered these questions, you can identify and define the elements of your business model.

Components of a Semantic Model

Fact tables, dimension tables, joins, and hierarchies are a semantic model's key components.

Component	Description
Fact Tables	<p>Fact tables contain measures (columns) that have aggregations built into their definitions.</p> <p>Measures aggregated from facts must be defined in a fact table. Measures are typically calculated data such as dollar value or quantity sold, and they can be specified in terms of hierarchies. For example, you might want to determine the sum of dollars for a given product in a given market over a given time period.</p> <p>Each measure has its own aggregation rule such as SUM, AVG, MIN, or MAX. A business might want to compare values of a measure and need a calculation to express the comparison.</p>
Dimension Tables	<p>A business uses facts to measure performance by well-established dimensions, for example, by time, product, and market. Every dimension has a set of descriptive attributes. Dimension tables contain attributes that describe business entities (like Customer Name, Region, Address, or Country).</p> <p>Dimension table attributes provide context to numeric data, such as being able to categorize Service Requests. Attributes stored in this dimension might include Service Request Owner, Area, Account, or Priority.</p> <p>Dimension tables in the model are conformed. In other words, even if there are three different source instances of a particular Customer table, the model only has one table. To achieve this, all three source instances of Customer are combined into one using database views.</p>
Joins	<p>Joins indicate relationships between fact tables and dimension tables in the model. When you create joins, you specify the fact table, dimension table, fact column, and dimension column you want to join.</p> <p>Joins allow queries to return rows where there is at least one match in both tables.</p> <p>Tip: Analysts can use the option Include Null Values when building reports to return rows from one table where there're no matching rows in another table.</p>
Hierarchies	<p>Hierarchies are sets of top-down relationships between dimension table attributes.</p> <p>In hierarchies, levels roll up from lower levels to higher levels. For example, months can roll up into a year. These rollups occur over the hierarchy elements and span natural business relationships.</p>

About Modeling Source Objects with Star Relationships

Star sources consist of one or more fact tables that reference any number of dimension tables. Because Data Modeler presents data in a star structure, working with star sources is the simplest modeling scenario. In star sources, dimensions are normalized with each dimension represented by a single table.

For example, assume that you have separate sources for Revenue Measures, Products, Customers, and Orders. In this scenario, you load data from each source to separate

database tables. Then, you use Data Modeler to create a fact table (Revenue Measures) and dimension tables (Products, Customers, and Orders). Finally, you create joins between the dimension tables and the fact table.

When you create your fact and dimension tables, you can drag and drop the source objects into the semantic model, or you can use menu options to create the fact and dimension tables individually.

See [Roadmap for Modeling Data](#) for a full list of data modeling tasks.

About Modeling Source Objects with Snowflake Relationships

Snowflake sources are similar to star sources. In a snowflake structure, however, dimensions are normalized into multiple related tables rather than in single dimension tables.

For example, assume that you have separate sources for Revenue Measures, Products, Customers, and Orders. In addition, you have separate sources for Brands (joined to Products) and Customer Group (joined to Customers). The Brands and Customer Group tables are considered to be "snowflaked" off the core dimension tables Customers and Products.

In this scenario, you load data from each source to separate database tables. Next, you create database views that combine the multiple dimension tables into a single table. In this example, you create one view that combines Products and Brand, and another view that combines Customer and Customer Group.

Then, you use Data Modeler to create a fact table (Revenue Measures) and dimension tables (Products + Brand view, Customers + Customer Group view, and Orders). Finally, you create joins between the dimension tables and the fact table.

See [Roadmap for Modeling Data](#) for a full list of data modeling tasks.

About Modeling Denormalized Sources

Denormalized sources combine facts and dimensions as columns in one table (or flat file). With a denormalized flat source, one data file is loaded into one table. The data file consists of dimension attributes and measure columns.

In some cases, the semantic model might consist of a hybrid model that involves a combination of star, snowflake, and denormalized sources. For example, a denormalized source might include information about revenue measures, products, customers, and orders - but all in a single file rather than in separate source files.

In this scenario, you first load the denormalized file as a single database table. Then, you use the Add to Model wizard to partition columns into multiple fact and dimension tables. In this example, you drag and drop revenue measure columns to create a fact table, then drag and drop columns for products, customers, and orders to create three separate dimension tables. Finally, you create joins between the dimension tables and the fact table.

See [Roadmap for Modeling Data](#) for a full list of data modeling tasks.

About Modeling Normalized Sources

Normalized or transactional sources distribute data into multiple tables to minimize data storage redundancy and optimize data updates. In a normalized source, you have

multiple data files that correspond to each of the transactional tables. Data from Oracle Cloud applications is likely partitioned into a normalized source.

Similar to snowflake sources, modeling normalized sources involves creating database views to combine columns from multiple source tables into individual fact and dimension tables. Some normalized sources are very complex, requiring a number of database views to organize the data into a star-type model.

For example, assume that you have source files for Products, Customers, Orders, and Order Items. Orders and Order Items both contain facts.

In this scenario, you first load the files as separate database tables. Next, you create a database view that combines the multiple fact columns into a single table. In this example, you create a view that combines columns from Orders and Order Items.

Then, you use Data Modeler to create a fact table (Orders + Order Items view) and dimension tables (Products and Customers). Finally, you create joins between the dimension tables and the fact table.

See [Roadmap for Modeling Data](#) for a full list of data modeling tasks.

4

Start to Build Your Semantic Model

This section provides information about first steps for building a semantic model, such as adding dimension tables, fact tables, and joins.

Topics:

- [Typical Workflow to Model Data Using Data Modeler](#)
- [Use Data Modeler](#)
- [Review Source Tables and Data](#)
- [Add Your Own Source Views](#)
- [Add Fact Tables and Dimension Tables to a Semantic Model](#)
- [Join Fact and Dimension Tables](#)
- [Create a Time Dimension](#)
- [Add Measures and Attributes to a Semantic Model](#)
- [Copy Model Objects](#)

Use Data Modeler

Data Modeler enables you to model the data that is needed to produce reports.



Note:

Data Modeler is available for a limited time.

Instead of using Data Modeler, Oracle advises that you use Semantic Modeler to create semantic models. See [Create an Empty Semantic Model](#).

If you're working with existing data models, Oracle advises that you migrate them to Semantic Modeler. For help with this migration, see [Import the Semantic Model From Data Modeler](#).

Topics:

- [Create a Semantic Model](#)
- [Use the Left Pane in Data Modeler](#)
- [Use the Right Pane in Data Modeler](#)
- [Use Action Menus](#)
- [Lock a Semantic Model](#)
- [Validate a Semantic Model](#)
- [Refresh and Synchronize Source Objects and Semantic Model Objects](#)

- [Publish Changes to Your Semantic Model](#)
- [Clear Cached Data](#)
- [Rename a Semantic Model](#)
- [Connect a Model to a Different Database](#)
- [Export a Semantic Model](#)
- [Import a Semantic Model](#)
- [Delete a Semantic Model](#)

Create a Semantic Model

Create a new semantic model from scratch in Data Modeler.



Note:

Data Modeler will be available only for a limited time. Instead of using Data Modeler, Oracle advises that you use Semantic Modeler to create semantic models. See [Create an Empty Semantic Model](#).

1. Open Data Modeler.
2. Click **Create model**.
3. Enter a name and description for your semantic model.
The subject area associated with this model gets the same name.
4. Connect the model to a **Database**.
If the database you want isn't listed, ask your administrator to set up the connection for you.

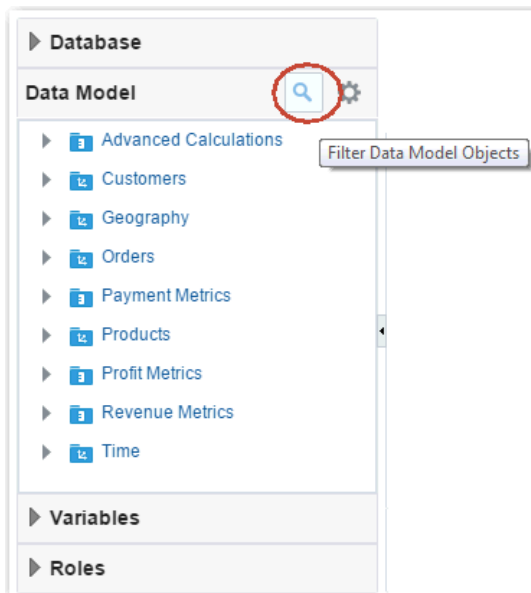
Use the Left Pane in Data Modeler

Various data modeling menus are available from the left pane in Data Modeler.

- **Database** — Lists source objects such as database tables and views
- **Data Model** — Lists model objects such as fact tables, dimension tables, hierarchies, fact columns, and dimension columns
- **Variables** — Lists variables for use in data security filters and in column expressions
- **Roles** — Lists roles that you can use when defining object permissions and data security filters

Filter a list to find exactly what you want.

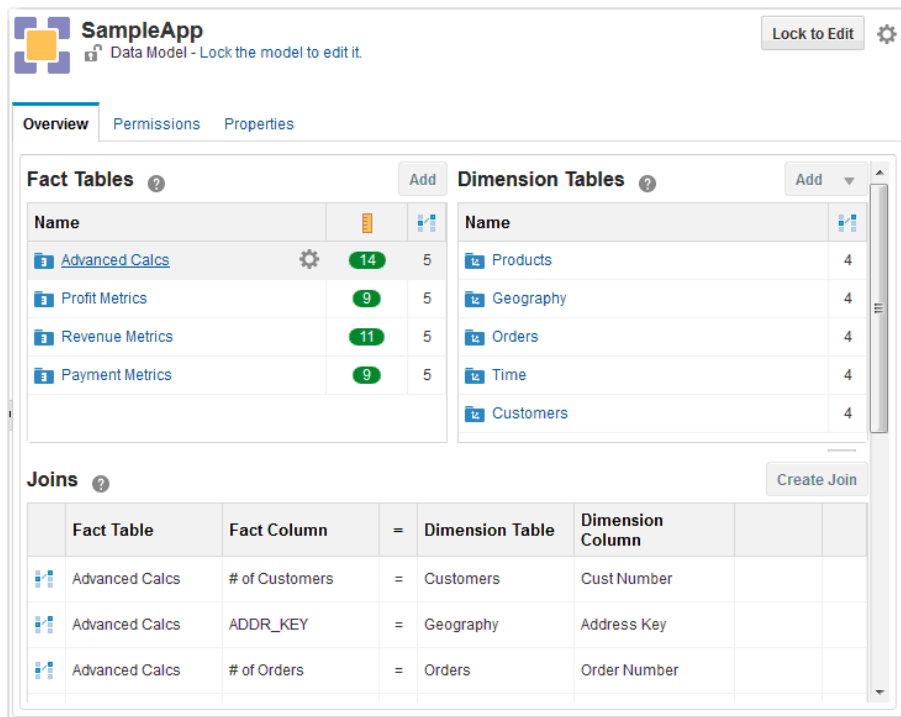
1. In Data Modeler, in the left pane, open the **Database**, **Data Model**, **Variables**, or **Roles** menu.
2. Click the **Filter** icon to the right of the selected menu.



3. In the Filter area, enter a string value for filtering the display.
4. Delete the text or click the **Filter** icon again to remove the filter.

Use the Right Pane in Data Modeler

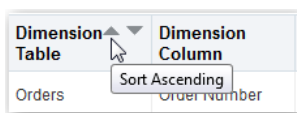
The right pane in Data Modeler is a contextual pane that changes depending on what task you're performing. After you have started modeling data, the default or home view shows the fact tables, dimension tables, and joins that you've defined so far.



- In the fact tables and dimension tables area you can see the number of joins for each fact and dimension table, as well as the number of measures in each fact table.



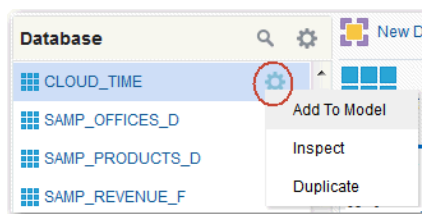
- Joins are listed below the fact and dimension tables. Click the up or down arrow in each column header to sort.



- When you click an object to open its editor, the editor appears in the right pane. For example, clicking a dimension table name from the Data Model menu in the left pane opens the dimension table editor in the right pane.
- Open the Permission tab to control who has access to the model and who is allowed to build reports from its associated subject area.
- Open the Properties tab to rename the model or connect the model to a different database.

Use Action Menus

Data Modeler provides action menus for most objects. When you select an object, you'll see a gear icon, which displays the menu (⚙️).



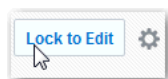
A global **Model Actions** menu in the upper right corner enables you to clear, close, refresh, or unlock the model.

You can also use action menus to delete individual objects that you have locked.

- You can delete source views but you can't delete source tables. Use SQL Workshop to drop tables in the source database.
- You can't delete model objects that other objects depend on.

Lock a Semantic Model

You lock a semantic model before making any changes. Click **Lock to Edit** to lock the semantic model.



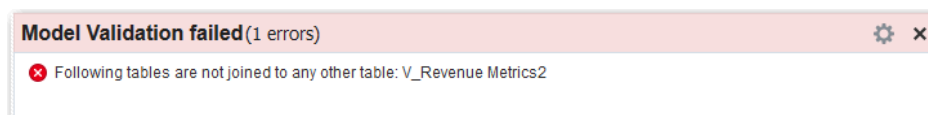
Tips:

- Publish changes regularly (browsers timeout after 20 minutes of inactivity).
- Publish changes before closing your browser to ensure that the lock is released.
- Lock your model before changing views.
- If you have administrative privileges, you can override locks set by other users.

Validate a Semantic Model

You can use the global **Validate** checkmark icon in the upper-left corner to check whether a semantic model is valid.

The semantic model is also validated automatically when you publish changes. Validation errors are shown at the bottom of the right pane.



Use the **Message Actions** menu to customize the types of messages displayed (Errors, Warnings, and Information).

Some tasks are validated when they're performed. For example, you can't save a source view unless its SQL query is valid. Expressions for calculated measures and derived columns must be valid before they can be saved. Validation messages that are displayed as you're performing tasks provide more information about any validation errors.

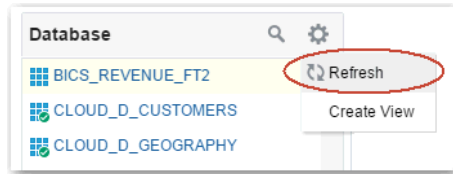
Refresh and Synchronize Source Objects and Semantic Model Objects

Data Modeler provides three ways to refresh data to ensure you're looking at the most up-to-date information. You can refresh source objects, refresh the semantic model, or synchronize the semantic model with source object definitions in the database.

Refresh Source Objects

You can refresh the Database pane to ensure that the source objects list reflects the latest objects in the database. For example, you can refresh the source objects list to include any new database tables that were added. The source objects list is not refreshed automatically after new objects are loaded in to the database.


To refresh source objects, select **Refresh** from the **Database Actions** menu in the left pane.



Refresh the Semantic Model

In some cases, other Data Modeler users might have locked the model and made changes. You can refresh the semantic model to ensure that Data Modeler is displaying the latest version of the model.

To refresh the semantic model, select **Refresh** from the **Data Model Actions** menu in the left pane.

Alternatively, select **Refresh Model** from the **Model Actions** gear menu  next to the **Lock to Edit** button.

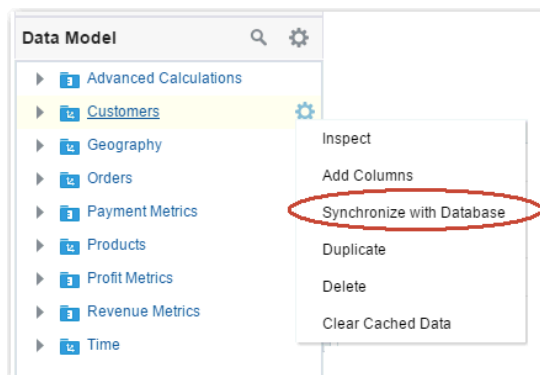
Synchronize with the Database

You can synchronize the semantic model with source objects in the database. Synchronization identifies objects in the model that have been deleted in the database, as well as tables and columns that are new. It also identifies other discrepancies like column data type mismatches.

To synchronize all model objects and source objects with the database, select **Synchronize with Database** from the global **Model Actions** menu in the upper right corner.

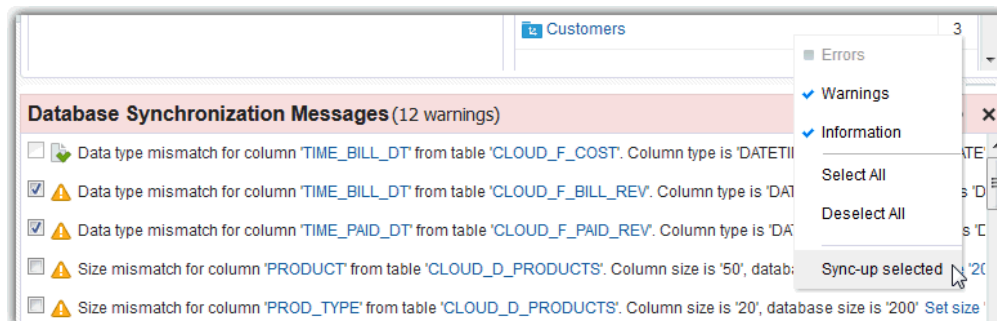
To synchronize individual fact tables or dimension tables, select **Synchronize with Database** from the **Actions** menu for the given fact table or dimension table in the Data Model objects list in the left pane. Then, click **OK**.

You must lock the semantic model to synchronize with the database.



Synchronization discrepancies are displayed in a message box at the bottom of the right pane. Use the **Message Actions** menu to customize the types of messages

displayed (Errors, Warnings, and Information), select or deselect all messages, and perform sync-up actions on selected messages. For example, you can select all data type mismatch warnings and then select **Sync-up selected** from the **Actions** menu to make the relevant synchronization changes.



Publish Changes to Your Semantic Model

As you update a semantic model, you make changes that you can save or discard. You publish a model to save the changes permanently and make the data available for use in reports. The published semantic model displays as a subject area.

Tip:

Although changes to the semantic model are saved as you work, they are saved in the browser session only. The changes aren't truly saved until you publish the model.

When you publish a semantic model, it is validated automatically. Any validation errors appear in the bottom of the right pane. If you see validation errors, fix them and then try to publish the semantic model again.

After making changes to your semantic model, you can perform these actions using the menus in the upper-right corner:

- **Publish and Unlock** — Verifies that the model is valid, saves the changes, and publishes the model for use with reports. The model is unlocked for other users.
- **Publish and Keep Lock** — Verifies that the model is valid, saves the changes, and publishes the model for use with reports. The lock is retained for further edits.
- **Unlock** — Removes the lock on the model so that other users can update it. Your unpublished changes to the model are discarded.
- **Revert** — Returns the model to its previously published state. Your unpublished changes to the model are discarded, but the model remains locked.
- **Clear**—Permanently deletes all objects in the model and removes them from any reports that are based on the model's subject area.

You can also click **Undo** and **Redo** in the upper right corner to revert or reapply individual changes.

 **Tip:**

You don't need to publish the model to save *database* changes. Changes made to database views and other source database objects are saved to the database when you complete the action, not to the semantic model. For database changes, **Undo** and **Redo** aren't available.

After publishing your model it takes up to two minutes for changes to the semantic model to reflect in reports and dashboards. To see changes immediately, open the report, click **Refresh**, and then **Reload Server Metadata**.

Oracle Analytics Cloud takes a snapshot when you or someone else publishes changes to the semantic model. If you're having some problems with the latest semantic model, you can ask your administrator to restore an earlier version.

Clear Cached Data

Oracle Analytics Cloud caches data to maximize performance. This means data updates may not immediately reflect in reports and Data Modeler.

After loading new data in your tables, you might want to clear the cache to see the very latest data.

- To see new data in Data Modeler, select the **Refresh Model** menu.
- To see new data in reports, manually clear the cache from the Data Model menu in the left pane
 - To clear cached data for a particular fact or dimension table, right-click the table and select **Clear Cached Data**.
 - To clear all cached data, click **Data Model Actions**, then select **Clear All Cached Data** to remove all data from the cache.

You can also select **Clear All Cached Data** from the global **Model Actions** menu in the upper-right corner.

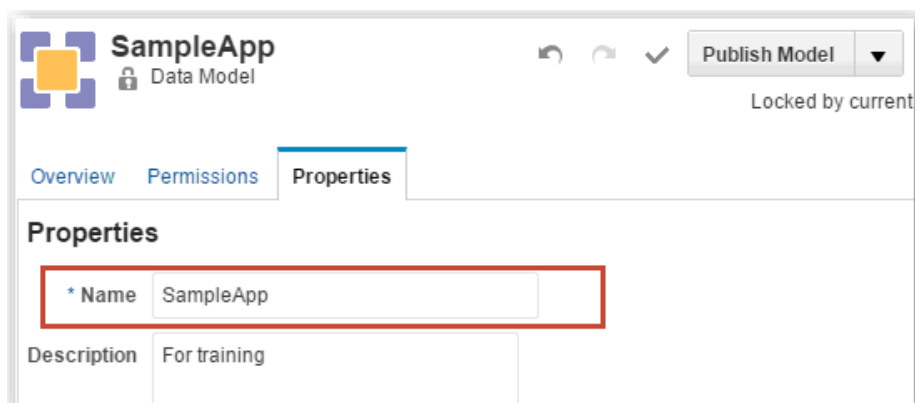
 **Tip:**

Always clear the cache after loading new data to ensure that the most recent data is displayed in reports.

Rename a Semantic Model

To rename a semantic model, lock it, select the Properties tab, and change the name.

This action also renames the corresponding subject area for reports.



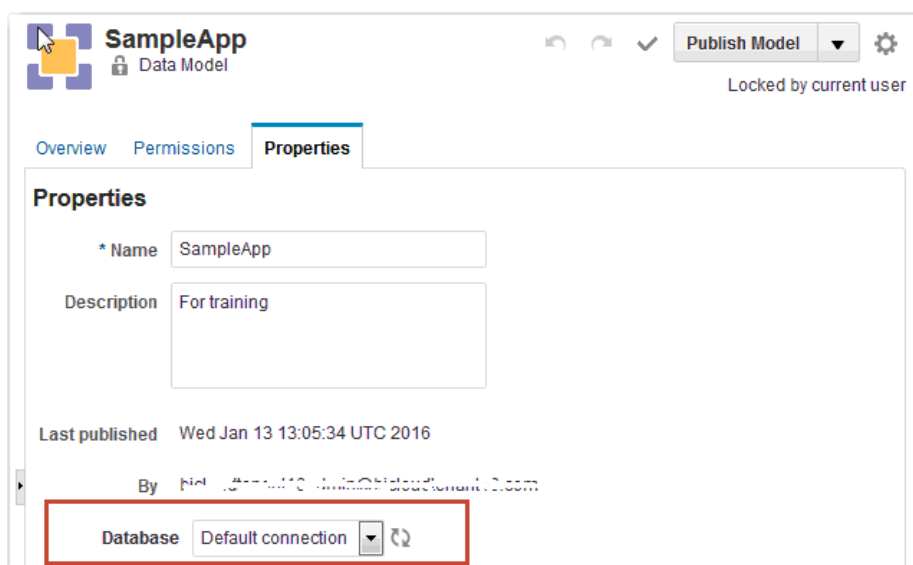
Connect a Model to a Different Database

When you start a new semantic model you're asked to select the database where your data is stored. All the tables and views in this database display in Data Modeler so you can add them to your model. Sometimes, data is moved or the source database changes. If this happens, change your model's database connection.

If you change the database, reports based on the model's subject area won't work unless all the required source objects are available in the new database.

1. In Data Modeler, lock your model for editing.
2. Click the **Properties** tab.
3. Select the **Database**.

If the database you want isn't listed, ask your administrator to set up the connection for you.



4. Synchronize your semantic model with the new database. Select **Synchronize with Database** from the **Model Actions** menu.

Export a Semantic Model

Individual semantic models can be exported to a JSON file and the information imported on another service. If you want to make minor changes to the model, you can edit the JSON before importing it. For example, you might want to change the name of the model (`modelDisplayName`) or the database connection (`connectionName`).

1. Open Data Modeler.
2. In the Models page, click the **Model Actions** icon for the model you want to export, and select **Export**.
3. Save the JSON file. The default name is `model.json`.

Import a Semantic Model

Individual semantic models can be exported to a JSON file and the information imported on another service. If you want to make minor changes to the model, you can edit the JSON before importing it. For example, you might want to change the name of the model (`modelDisplayName`) or the database connection (`connectionName`).

For any semantic model to work properly it must have access to the associated database tables. Before importing the semantic model, check whether Data Modeler can connect to the required database. If not, ask your administrator to set up the connection.

1. Open Data Modeler.
2. Click **Import Model**.
3. Browse to the JSON file that contains the semantic model you want to import.
4. Click **OK**.
5. Optional: Select a database connection for the model.

You're asked to select a database connection if Data Modeler doesn't recognize the connection name in the JSON file. If the connection you want isn't listed, ask your administrator to set up the connection and try again.

6. Optional: Choose whether to replace a semantic model with the same name. Click **Yes** to overwrite the model or **No** to cancel.

This happens when the model named in the JSON file clashes with another model in Data Modeler. If you don't want to replace the existing model, change the `modelDisplayName` attribute in the JSON file and try again.

Delete a Semantic Model

You can delete all objects from your semantic model if you want to clear your model and start over. Or you can delete an entire model along with its subject area.

- Clearing model content—Lock the model and select **Clear Model** from the global **Model Actions** menu in the upper right corner.

This permanently removes all the objects in the semantic model and also removes them from any reports that are based on the model's subject area.

- Deleting a model—Click **Data Modeler**, click the **Model Actions** menu for the model you don't want anymore, and select **Delete**.

This permanently removes the semantic model and its subject area.

Before clearing or deleting a model, we recommend that you or your administrator take a snapshot of the model as a backup

Review Source Tables and Data

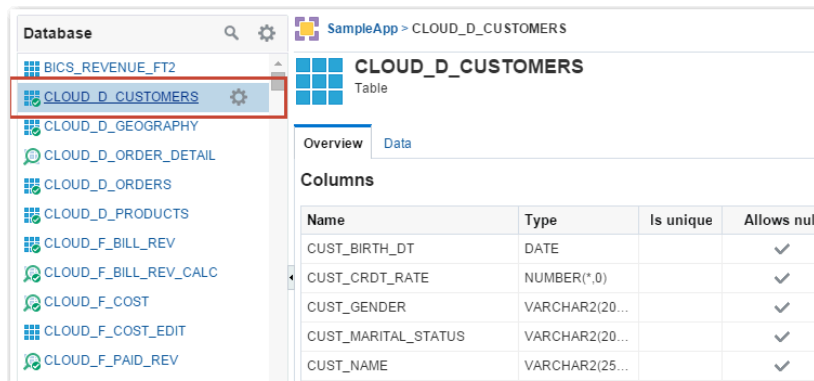
This topic describes how you can learn more about the source database objects that are available for your semantic model.

Topics:

- [View Source Objects](#)
- [Preview Data in Source Objects](#)

View Source Objects

You can see a list of source tables and views in the Database menu in the left pane. Click a table or view to see its properties.



The Overview tab for source tables and views shows column information, like column name, data type, whether it's unique, and whether it accepts null values.

Preview Data in Source Objects

You can preview the first 25 rows of data in your database tables and views. By reviewing the initial rows, you can get ideas for modeling the database tables and views as either dimension tables or fact tables.

1. Open Data Modeler.
2. From the Database menu in the left pane, click a database table or view to open it.
3. Click the **Data** tab.
4. Review the first 25 rows of data for the table or view. You can resize the columns in the display table if needed.

SHIPTO_ADDR_KEY	OFFICE_KEY	EMPL_KEY	PROD_KEY	ORDER_KEY	UNITS
379	11	1	4	5784	57
2257	15	2	10	5785	208
1306	2	12	3	5786	65

5. Click **Get Row Count** to retrieve a complete row count for the table or view. This task might take some time to complete if the table is large.
6. Click **Done**.

Create Source Views

Create source views as a base for model objects when you think you might want to perform subsequent changes.

Topics:

- [About Source Views](#)
- [Add Your Own Source Views](#)
- [Define Filters for Source Views](#)

About Source Views

Source views are saved queries of data in the database. You can think of a source view as a "virtual table."

You create source views when using a single table as a source for more than one dimension table. For example, you can create source views that use the Employee source table as a source for the Employee and Manager dimension tables.

You also create source views when creating a dimension table that is based on multiple source tables, as in a snowflake source. For example, you can create a source view that combines columns from the Customer and Customer Group source tables to create a single Customers dimension table.

You can also perform pre-aggregation calculations in a source view. For example, to create an Average Revenue column that is calculated pre-aggregation, you can include the calculation in the SQL query for the view:

```
SELECT
  "BICS_REVENUE_FT1"."UNITS",
  "BICS_REVENUE_FT1"."ORDER_KEY",
  "BICS_REVENUE_FT1"."REVENUE",
  "BICS_REVENUE_FT1"."PROD_KEY",
  "BICS_REVENUE_FT1"."REVENUE"/"BICS_REVENUE_FT1"."UNITS" AS AVERAGE_REVENUE
FROM
  "BICS_REVENUE_FT1"
```

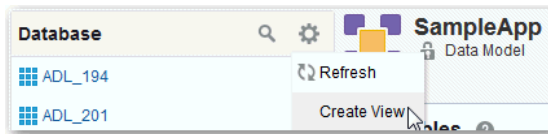
In general, create source views as a base for model objects when you think you might want to perform subsequent changes. Creating a semantic model based on source views provides greater flexibility than using source tables directly. For example, using source views makes it much easier to extend model objects, create filters, and add pre-aggregation calculations.

Add Your Own Source Views

You can add views to the source database from Data Modeler. For example, you can create a source view that combines the Brands and Products source tables to create a single source for your dimension table.

Create source views as a base for model objects when you think you might want to perform subsequent changes. You can create a view from scratch and add any column you want from other tables and views in the database. Alternatively, you can create a view by copying an existing source table or another source view.

1. In Data Modeler, lock the model for editing.
2. From the Database menu in the left pane, click **Actions**, then click **Create View**.

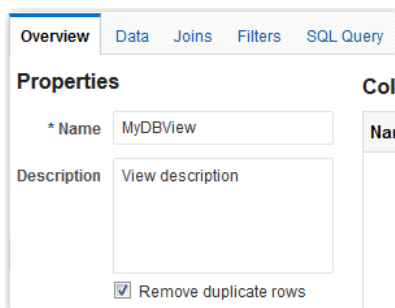


Initially the view is empty. You can add any column you want from other tables and views in the database.

Tip:

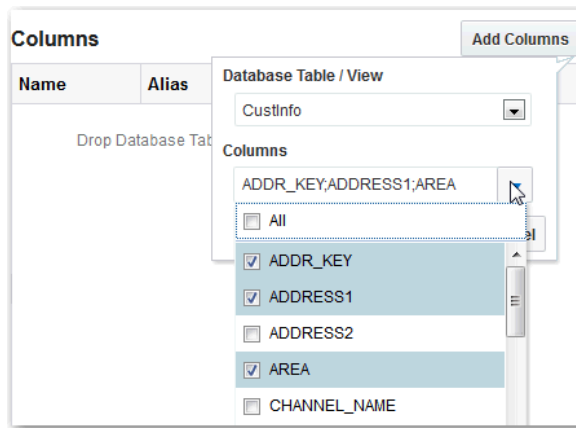
To create a view from an existing source table or source view, navigate to the database object you want to copy, click **Actions**, and then click **Duplicate**.

3. In the View editor, specify a name and description for the view. Optionally deselect **Remove duplicate rows** if you want to include duplicate rows in the view.

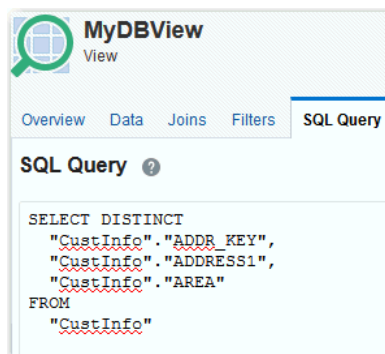


4. Add columns to the database view by dragging and dropping tables or views from the Database menu into the Columns area of the View editor.

Alternatively, click **Add Columns**, select a source database table or view, select columns, and then click **Add**.



5. Define aliases for columns if needed. You can also optionally move rows up or down using the **Action** menu for a specific row.
6. From the Joins tab, you can define joins for the view. Click **Create Join**, then specify the left side table, right side table, columns, and the join type. You must include more than one source table in your view to create joins.
7. From the Filters tab, you can define filters for the view.
8. From the SQL Query tab, review the code for the SQL query for the source view.



You can edit the SQL code for the query here, but do so only if you're familiar with SQL code. Entering invalid SQL code can produce unexpected results.

If you do edit the SQL query directly, simple updates are reflected back in the Overview, Join, and Filters tabs and you can use these tabs to further edit the view later. For example, you can include:

- Simple SELECT clause with aliases and DISTINCT keyword
- FROM clause with joins
- WHERE clause with filter conditions which combined with AND keyword

If you use the SQL Query tab to make more advanced code changes you cannot use the Overview, Joins or Filters tabs to further edit the view. For example, if you include:

- SQL aggregation functions, GROUP BY clause, HAVING clause
- ORDER BY clause

- OR keyword in WHERE clause
- Optional: Click the Data tab to preview the first 25 rows of data. You can also get a complete row count. It is best to view data only after defining joins between all tables for better performance.
 - Click **Save and Close**.

Define Filters for Source Views

A filter specifies criteria that are applied to columns to limit the results that are returned. In other words, a filter is the `WHERE` clause for the view statement. For example, you can define a filter where Customer Country is equal to USA.

- Create a view.
- Click the **Filters** tab.
- Click **Create Filter**.
- In the `WHERE` row, first select the column for the filter. Next, select the condition, such as "is not equal to" or "is greater than".

Finally, specify the value for the filter. You can specify a variable if needed.

	Column	Condition	Value
WHERE	CustInfo.AREA	is like	WESTERN

- Optional: Click **Create Filter** again to add an "and" row to the filter. Specify the column, condition, and value. Repeat as needed.
- To remove a row, click **Actions**, then select **Delete**.

Value
WESTERN

- Click **Save**.

Add Fact Tables and Dimension Tables to a Semantic Model

Use fact tables and dimension tables to represent aspects of your business that you want to understand better.

Topics:

- [About Fact Tables and Dimension Tables](#)
- [Create Fact and Dimension Tables from a Single Table or View](#)
- [Create Fact Tables Individually](#)

- [Create Dimension Tables Individually](#)
- [Edit Fact Tables and Dimension Tables](#)
- [Add More Columns to Fact and Dimension Tables](#)

About Fact Tables and Dimension Tables

Fact tables and dimension tables hold the columns that store the data for the model:

- Fact tables contain measures, which are columns that have aggregations built into their definitions. For example, Revenue and Units are measure columns.
- Dimension tables contain attributes that describe business entities. For example, Customer Name, Region, and Address are attribute columns.

Fact tables and dimension tables represent the aspects of your business that you want to understand better. See [Components of Data Models](#).

Before you begin modeling fact tables and dimension tables, make sure that the data that you need to model is available in the source tables list. Also ensure that you have created any source views upon which to base model objects.

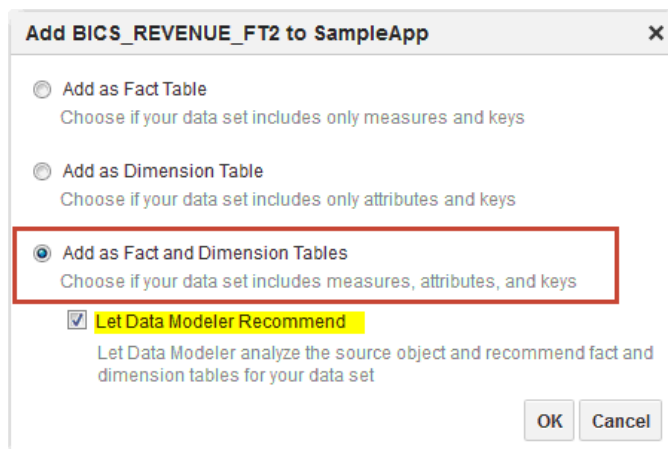
If you think the list of source objects in the database has changed since you opened Data Modeler, then you can click **Refresh** from the **Database Actions** menu. If the data that you need has not yet been loaded into the database, then you can load it.

Create Fact and Dimension Tables from a Single Table or View

Some source tables contain both facts and dimensions. For these source tables, Data Modeler provides a wizard to help you partition the fact and dimension columns into fact tables and dimension tables.

For example, you might have a source that contains both product and customer attributes, as well as revenue measures. Use the wizard to create the corresponding fact and dimension tables.

1. In Data Modeler, lock the model for editing.
2. In the Database menu in the left pane, right-click the source table that contains the fact and dimensional data that you want to model, select **Add to Model**, and then select **Add as Fact and Dimension Tables**.



3. To let Data Modeler suggest some fact tables, dimension tables, and joins for the source table, select **Let Data Modeler Recommend** and click **OK**. You can review suggestions in Step 4.

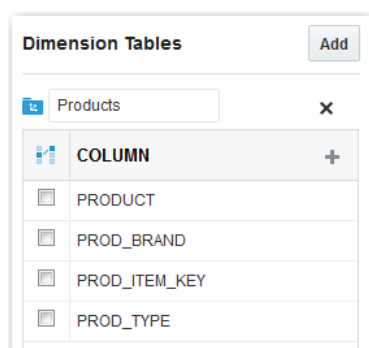
If you'd rather choose fact and dimension tables yourself from scratch:

- a. Deselect **Let Data Modeler Recommend** and click **OK**.
- b. Drag measures from the source table onto the fact table.

 **Tip:**

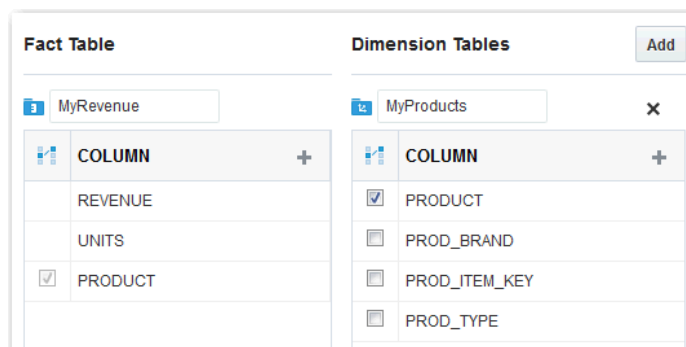
You can also click the **Plus** icon in the column header area to select a column to include in the fact table.

- c. Enter a name for the fact table, such as Costs or Measures.
- d. Add a dimension table for each group of related attributes, and enter a meaningful name, such as Products. Drag and drop related columns from the source table to the appropriate dimension table.



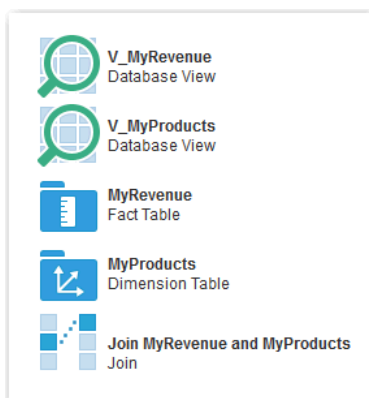
- e. To add more dimension tables, click **Add** and repeat the previous step.
- f. To delete a dimension table, click **X** next to the table name.
- g. Specify the join columns for each of the dimension tables. Select the box beside the appropriate columns to mark those columns as join columns.

If the join column you select is missing from the fact table, a corresponding column gets added automatically to the fact table.



4. Review fact tables, dimension tables, and join columns. For example:

- Rename fact and dimension tables.
 - Add or remove columns.
 - Add, delete, or merge dimension tables.
 - Move columns from one dimension table to another.
5. Click **Next**.
 6. Review the objects that will be created.



7. Click **Create**.
8. Click **Done**.

New fact tables, dimension tables, and joins display in Data Modeler. New views display in the Database pane.

Create Fact Tables Individually

You can add individual source tables containing fact data to your semantic model.

If you have distinct source tables with fact data, such as in a star source, then you can add them to your semantic model individually. For example, if you have a source table that contains only revenue measures, then you can use this method to create the corresponding fact table.

Alternatively, you might have sources with fact information spread across multiple tables, such as normalized transactional sources. In this case, create source views first to combine tables in a way that resembles a star model. For information about creating views, see [Add Your Own Source Views](#). For information about modeling different source types, see [Plan a Semantic Model](#).

Tip:

Create source views as a base for model objects when you think you might want to perform subsequent changes like extending model objects, creating filters, and adding pre-aggregation calculations. Creating a fact table based on source views provides greater flexibility than using source tables directly.

When you use this method to create individual fact tables, all columns in the source table or view are assigned to a single fact table and if the source has relationships with other tables or views, we'll offer to add them to your model.

After locking the model, perform one of the following actions to create fact tables individually:

- Drag the source table or view from the Database menu in the left pane to the Fact Tables area of the semantic model.
- From the Database menu in the left pane, right-click the table or view, then click **Add to Model**, then **Add as Fact Table**.
- From the Database menu in the left pane, click **Table Actions** or **View Actions**, click **Add to Model**, then **Add as Fact Table**.
- From the Database Table or View editor for a particular source table or view, click **Add to Model**, then **Add as Fact Table**.
- In the right pane, click **Add** in the Fact Tables area of the semantic model. Then, select one or more source tables and views from the Database Objects list and click **OK**.
- To copy an existing fact table, click **Fact Table Actions** for the fact table you want to copy, and then click **Duplicate**.

After adding the source table or view to the model, you can edit the fact table.

Create Dimension Tables Individually

You can add individual source tables containing dimension data to your semantic model.

If you have distinct dimensional source tables, such as in a star source, then you can add them to your semantic model individually. For example, if you have a source table that contains only customer attributes, then you can use this method to create the corresponding dimension table.

Alternatively, for snowflake or normalized (transactional) sources, create source views to combine source objects in a way that resembles a star model. For information about creating views, see [Add Your Own Source Views](#). For information about modeling different source types, see [Plan a Semantic Model](#).

Tip:

Create source views as a base for model objects when you think you might want to perform subsequent changes like extending model objects, creating filters, and adding pre-aggregation calculations. Creating a dimension table based on source views provides greater flexibility than using source tables directly.

When you use this method to create individual dimension tables, all columns in the source table or view are assigned to a single dimension table and if the source has relationships with other tables or views, we'll offer to add them to your model.

After locking the model, perform one of the following actions to create dimension tables individually:

- Drag the table or view from the Database menu in the left pane to the Dimension Tables area of the Data Model.

- From the Database menu in the left pane, right-click the table or view, click **Add to Model**, and then select **Add as Dimension Table**.
- From the Database menu in the left pane, click **Table Actions** or **View Actions** for a table or view, click **Add to Model**, and then select **Add as Dimension Table**.
- Click **Add** in the Dimension Tables area, and then select **Add Database Tables**. From the Database Objects list, select one or more sources and then click **OK**.
- From the Database Table or View editor for a particular source table or view, click **Add to Model** and then select **Add as Dimension Table**.
- To copy an existing dimension table, click **Dimension Table Actions** for the dimension table you want to copy, and then click **Duplicate**.

After adding the source table or view to the model, you can edit the dimension table.

Edit Fact Tables and Dimension Tables

You can edit properties of fact and dimension tables in your semantic model and preview the source data.

1. In Data Modeler, lock the model for editing.
2. Click the fact table or dimension table that you want to edit.
3. Change settings on the Overview tab as needed.
 - **Time dimension** - For dimension tables only. Specifies that hierarchies for this dimension table support a time dimension.
 - **Enable skipped levels and Enable unbalanced hierarchies** - For dimension tables only. Set properties for hierarchies associated with this dimension table.
 - **Column list** - Click the link for a column to edit that column in the Column editor. Or, right-click the row for the column and click **Edit**.
 - **Aggregation** - For fact tables only. Click to select a type of aggregation for the column from the list, or select **Set Aggregation** from the Column Actions menu. Aggregation types include:
 - For fact tables only. Click to select a type of aggregation for the column from the list, or select **Set Aggregation** from the Column Actions menu. Aggregation types include:
 - None**: Applies no aggregation.
 - Sum**: Calculates the sum by adding up all values.
 - Average**: Calculates the mean value.
 - Median**: Calculates the middle value.
 - Count**: Calculates the number of rows that aren't null.
 - Count Distinct**: Calculates the number of rows that aren't null. Each distinct occurrence of a row is counted only once.
 - Maximum**: Calculates the highest numeric value.
 - Minimum**: Calculates the lowest numeric value.
 - First**: Selects the first occurrence of the item.
 - Last**: Selects the last occurrence of the item.

Standard Deviation: Calculates the standard deviation to show the level of variation from the average.

Standard Deviation (all values): Calculates the standard deviation using the formula for population variance and standard deviation.

Tip: Some calculated measures show Pre-Aggregated for aggregation. These measures have calculations involving measures that already have an aggregation applied. To edit a calculation that contains pre-aggregated measures, click the column name.

- **Available** - Click to mark a column as **Available** or **Unavailable** to choose whether that column is displayed in analyses that are created. You can also select **Mark as Unavailable** or **Mark as Available** from the Column Actions menu.
- **Edit All** - You can click to edit properties for individual columns in the table, or select **Edit All** to edit all rows at once.
- **Add Column** - Click **Add Column** to display the Column editor and create a new column.

Columns						<input type="checkbox"/> Edit All	Add Column
Name	Source	Type	Joined ?	Aggregation	Available ?		
# of Customers	CUST_NUMBER	DOUBLE		Count Distinct	✓		
# of Orders	ORDER_KEY	DOUBLE		Count	✓		
# of Products	PROD_ITEM_KEY	VARCHAR(20)		Count Distinct	✓		
ADDR_KEY	ADDR_KEY	DOUBLE		None	—		
Average Order Size	Expression	DOUBLE		Pre-Aggregated	✓		
Average Unit Price	Expression	DOUBLE		Average	✓		
Billed Units	UNITS	DOUBLE		Sum	✓		

4. From the Source Data tab, you can preview the first 25 rows of source data for the table. Resize the columns in the display table if needed. Click **Get Row Count** to retrieve a complete row count for the table or view.
5. For dimension tables only: from the Hierarchies tab, edit the hierarchies and levels for the table.
6. From the Permissions tab, specify object permissions.
7. From the Data Filters tab, you can define data filters that provide row-level filtering for semantic model objects. See [Secure Access to Data](#).
8. Click **Done** to return to the semantic model.

Add More Columns to Fact and Dimension Tables

There are different ways to add more source columns to fact and dimension tables in your model.

- If new columns are added to a source table and you want to include them in fact tables and dimension tables in your model, synchronize the fact or dimension table with the database. Synchronization identifies any new columns and adds them to the fact or dimension table. See [Refresh and Synchronize Source Objects and Sematic Model Objects](#).

- Dimension tables can combine columns from multiple sources. See [Add Columns from Another Source to a Dimension Table](#).

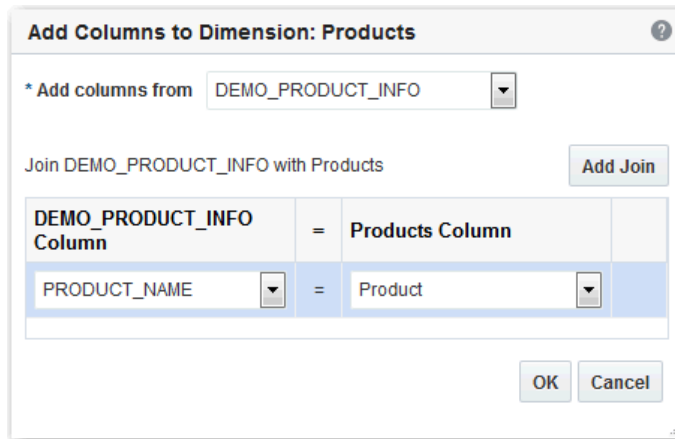
Add Columns from Another Source to a Dimension Table

You can add the columns from another source table or view to an existing dimension table. For example, you may want to include attributes from a Product Category table in your Products dimension table.

1. In Data Modeler, lock the model for editing.
2. Select the dimension table you want to edit so its Overview tab displays.
3. Drag and drop the source table or view that contains the columns you want to add from the Database pane to the dimension table (columns area).

Alternatively, right-click the dimension table you want to edit, click **Add Columns**, and then select the source table or view that contains the columns you want to add.

4. Select appropriate join columns and click **OK**.



View the dimension table to see the additional columns. The Source property shows that the dimension table is based on a new database view. Data Modeler creates a new database view whenever you add columns from another source.

Join Tables in a Semantic Model

A join in the model indicates a relationship between one fact table and one dimension table.

Topics:

- [About Joins](#)
- [Join Fact and Dimension Tables](#)

About Joins

A join in the model indicates a relationship between one fact table and one dimension table. When you use the Add to Model wizard to model data, the wizard creates joins automatically between a fact table and each of its corresponding dimension tables.

When you model fact and dimension tables individually, joins are automatically created between them if the join references exist in the source tables.

You can also manually create joins in the semantic model. To do this, you drag and drop a dimension table to a fact table, or click **Create Join** in the Joins area.

When you define a join between a fact table and dimension table, you select a join column from each table. You can create a join on more than one column.

Join Fact and Dimension Tables

Define joins between fact tables and dimension tables to enable querying of related data. For example, you can define a join between the Profit Metrics fact table and the Products dimension table.

1. In Data Modeler, lock the model for editing.
2. In the Dimensions Tables area, drag and drop a dimension table to the Fact Tables area. Or, in the Joins area, click **Create Join**.

Fact Table	Fact Column	=	Dimension Table	Dimension Column	
Profit Metrics	Select a column	=	Products	Select a column	✓ ✕

3. In the Joins area, specify the appropriate Fact Table, Fact Column, Dimension Table, and Dimension Column to use for the join.

For example, you might specify a billing date column and a calendar date column.

4. Click the checkmark icon to save the changes to the join.

If you want to remove your changes, then click the X icon. If you start to create a new join and click X, then the new row for the join is removed from the Joins table.

After you create joins, you can see the default hierarchies and levels when you click the Hierarchies tab for the given dimension table.

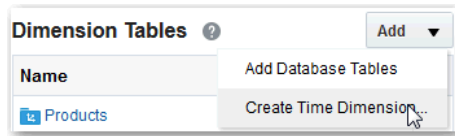
Create a Time Dimension

Time series functions provide the ability to compare business performance with previous time periods, enabling you to analyze data that spans multiple time periods. For example, time series functions enable comparisons between current sales and sales a year ago, a month ago, and so on. To use time series functions, the semantic model must include a time dimension

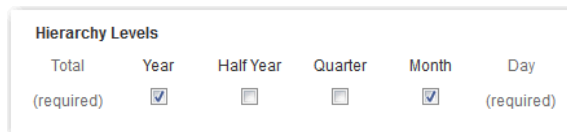
When you create a time dimension, the Create Time Dimension wizard creates a table in the database, populates it with time data, creates a corresponding time dimension table in the semantic model, and creates a time hierarchy.

The Create Time Dimension wizard populates the source table with time data from 01-JAN-1970 to 31-DEC-2020.

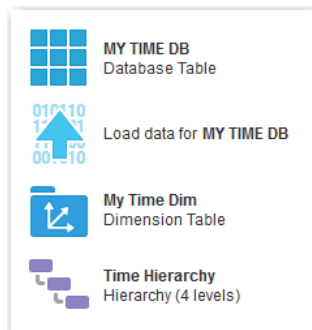
1. In Data Modeler, lock the model for editing.
2. In the Dimension Tables area, click **Add**, then **Create Time Dimension**.



3. In the Create Time Dimension wizard, specify names for the database table, the dimension table, and the hierarchy.
4. In the Hierarchy Levels, specify which levels to include, such as Year, Quarter, and Month.



5. Click **Next**.
6. On the next page, review the tasks that the wizard will perform to create the time dimension.



7. Click **Create** to enable the wizard to create the dimension.

The wizard adds a time dimension with data to the database and creates a corresponding dimension in the semantic model. This action might take up to 30 seconds.

8. Click **Done**.
9. To create joins between columns in the fact table and columns in the Time dimension table, click **Create Join** in the semantic model.

The time dimension has two unique columns. The DAY_TS column has the type TIMESTAMP, and the DATE_ID column has the type NUMBER. When you create a join, you specify either the column with the timestamp format or with the numeric format (depending on whether the column in the fact table has a date or number type).

10. In the Joins area for the new definition, select the appropriate fact column, then select the appropriate timestamp or numeric column from the Time dimension.

After you create the joins, you can display the Hierarchies tab in the Time Dimension editor to view the default hierarchies and levels.

11. Edit the tables in the model.
12. Click **Done** to return to the semantic model.

Add Measures and Attributes to a Semantic Model

This topic describes how to add measures and attributes to your semantic model.

Topics:

- [Edit Measures and Attributes](#)
- [Specify Aggregation for Measures in Fact Tables](#)
- [Create Calculated Measures](#)
- [Create Derived Attributes](#)
- [Create Expressions in the Expression Editor](#)
- [Copy Measures and Attributes](#)

Edit Measures and Attributes

Use the table editor to add, edit, and delete measures and attributes in your semantic model.

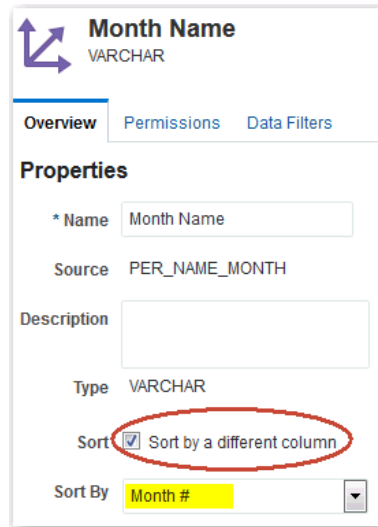
1. In Data Modeler, lock the model for editing.
2. Click the fact table or dimension table that contains the measure or attribute that you want to edit.
3. To edit all the columns directly in the table editor, select **Edit All**.

To edit, copy, or delete a selection of columns at the same time, Shift + click or Ctrl + click the rows you want.

Name	Source	Type	Joined	Aggregation	Available
# of Customers	CUST_NUMBER	DOUBLE		Count Distinct	✓
# of Orders	ORDER_KEY	DOUBLE		Count	✓
# of Products	PROD_ITEM_KEY	VARCHAR(20)		Count Distinct	✓
ADDR_KEY	ADDR_KEY	DOUBLE		None	—
Average Order Size	Expression	DOUBLE		Pre-Aggregated	✓
Average Unit Price	Expression	DOUBLE		Average	✓
Billed Units	UNITS	DOUBLE		Sum	✓

4. In the table editor, right-click a column and optionally click **Copy** or **Delete** as appropriate.
5. In the table editor, click the column that you want to edit or click **Add Column**.

6. Change settings on the Overview tab as needed.
 - Edit the display name and description.
 - Change the sort order.
By default, columns are sorted based on the data in the column and reports display data in this order. To sort a column based on the data in another column, select **Sort by a different column** and select the **Sort By** value you prefer. For example, instead of sorting a Month Name attribute alphabetically, you could sort by month number, such as 1 (January), 2 (February), 3 (March), and so on.



7. Change settings for calculated measures or derived attributes.
8. Optional: From the Permissions tab, modify object permissions.
9. Optional: From the Data Filters tab, define data filters that provide row-level filtering for semantic model objects. See [Secure Access to Data](#).
10. Optional: From the Levels tab for columns in a fact table, create a level-based measure. See [Set Aggregation Levels for Measures](#).
11. Click **Done** to return to the table editor.

Specify Aggregation for Measures in Fact Tables

You can specify aggregation for a measure in a fact table. For example, you can set the aggregation rule for a Revenue column to **Sum**.

1. In Data Modeler, lock the model for editing.
2. In the Fact Tables area, click the fact table for which you want to create measures.
3. In the Columns list, change the aggregation rule for the appropriate columns to specify that they're measures.

To apply the same aggregation rule to multiple columns, Shift + click or Ctrl + click the appropriate columns.

Aggregation options include:

None: No aggregation.

Sum: Calculates the sum by adding up all values.

Average: Calculates the mean value.

Median: Calculates the middle value.

Count: Calculates the number of rows that aren't null.

Count Distinct: Calculates the number of rows that aren't null. Each distinct occurrence of a row is counted only once.

Maximum: Calculates the highest numeric value.

Minimum: Calculates the lowest numeric value.

First: Selects the first occurrence of the item.

Last: Selects the last occurrence of the item.

Standard Deviation: Calculates the standard deviation to show the level of variation from the average.

Standard Deviation (all values): Calculates the standard deviation using the formula for population variance and standard deviation.

 **Tip:**

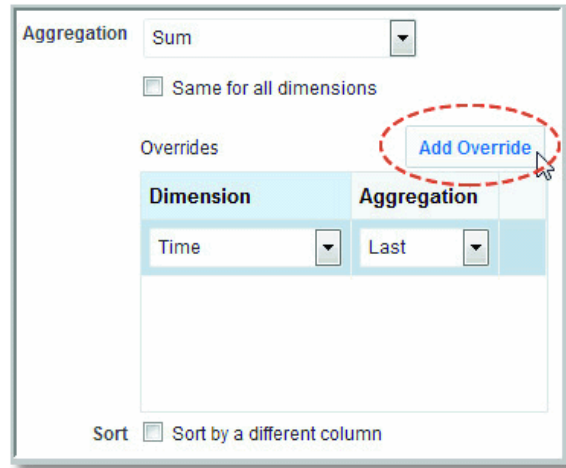
Some calculated measures are **Pre-Aggregated**. These measures have calculations involving measures that already have an aggregation applied. To edit a calculation that contains pre-aggregated measures, click the column name.

Name	Source	Type	Joined	Aggregation	Available
# of Customers	CUST_NUMBER	DOUBLE		Count Distinct	✓
# of Orders	ORDER_KEY	DOUBLE		Count	✓
# of Products	PROD_ITEM_KEY	VARCHAR(20)		Count Distinct	✓
ADDR_KEY	ADDR_KEY	DOUBLE		None	—
Average Order Size	Expression	DOUBLE		Pre-Aggregated	✓
Average Unit Price	Expression	DOUBLE		Average	✓
Billed Units	UNITS	DOUBLE		Sum	✓
COST_FIXED	COST_FIXED	DOUBLE		Sum	
COST_VARIABLE	COST_VARIABLE	DOUBLE		Sum	
Discount Ratio %	Expression	NUMERIC		Average	
Discount Value	DISCNT_VALUE	DOUBLE		Median	
Revenue	REVENUE	DOUBLE		Count	
TIME_BILL_DT	TIME_BILL_DT	DATE		Count Distinct	
				Maximum	
				Minimum	
				First	
				Last	
				Standard Deviation	
				Standard Deviation (all values)	

For most measures, the same aggregation rule applies for each dimension but for some measures you'll want to specify one aggregation rule for a given dimension and specify other rules to apply to other dimensions.

Time dimensions are most likely to require different aggregation. For example, Headcount (calculated measure) typically aggregates as SUM across Organization and Geography dimensions but SUM does not apply for a Time dimension. Aggregation for the Time dimension should be LAST, so you can show Headcount on the last week or day of the year.

4. To override the aggregation for specific dimensions:
 - a. Click the name of the measure column.
 - b. Deselect **Same for all dimensions**.



- c. Click **Add Override**.
- d. Select the dimension you want to aggregate differently, for example Time.
- e. Select an aggregation rule for the dimension.
- f. If required, override aggregation for another dimension.
- g. Click **Done**.

When dimension-specific aggregation rules are defined for a measure, you see an asterisk * next to the aggregation rule in the Columns table. For example, **Sum***.

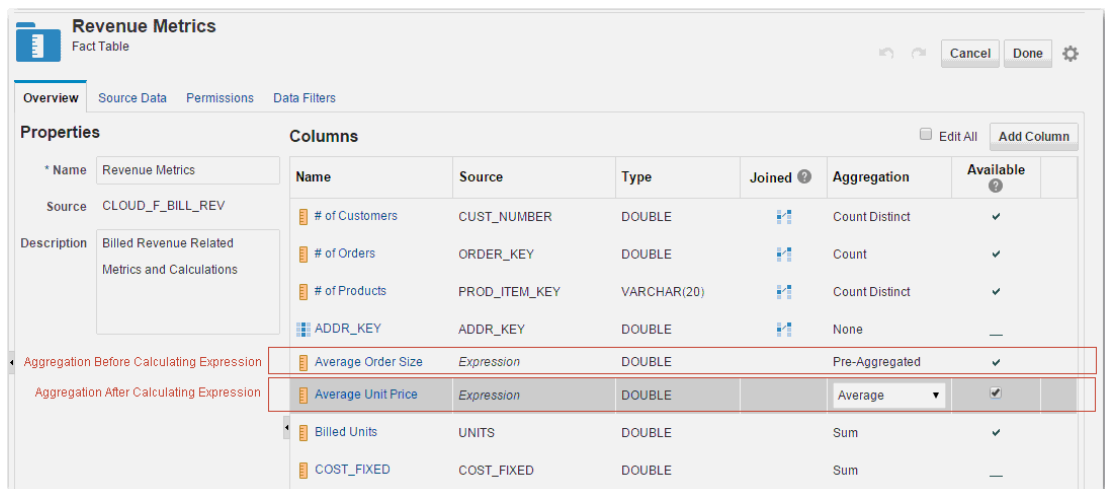
5. By default, all the columns in the fact table are displayed in reports. Deselect the **Available** box for any columns that you don't want to display. You can use Shift + click or Ctrl + click to select multiple rows.
6. Click **Cancel** to cancel any of your changes.
7. Click **Done** to return to the table editor.

Create Calculated Measures

If a fact table does not include all the measures that you need, then you can create calculated measures. For example, you can create a calculated measure called Average Order Size using the formula Revenue/Number of Orders.

1. In Data Modeler, lock the model for editing.
2. In the Fact Tables area, click the fact table for which you want to create measures.
3. In the Columns area, click **Add Column**.

4. In the New Column editor, enter a name and description for the column.
Then, enter an expression directly in the Expression box, or click **Full Editor** to display the Expression editor.
5. Expressions can contain measures that are already aggregated, as well as measures with no aggregation applied. Do one of the following:
 - Set Aggregation to **Before Calculating**, if your expression includes measures that are already aggregated or aggregation is not required.
 - Set Aggregation to **After Calculating** and select an aggregation rule, such as **Sum**, **Average**, **Count**, to apply aggregation after calculating the expression.
6. Click **Done** to return to the table editor.



About Creating Calculated Measures

Calculated measures, as the name suggests, are calculated from other measures. For example, you can create a measure that calculates Average Order Size using the formula Revenue/Number of Orders.

Calculations can contain measures that are already aggregated, as well as measures with no aggregation applied. For example:

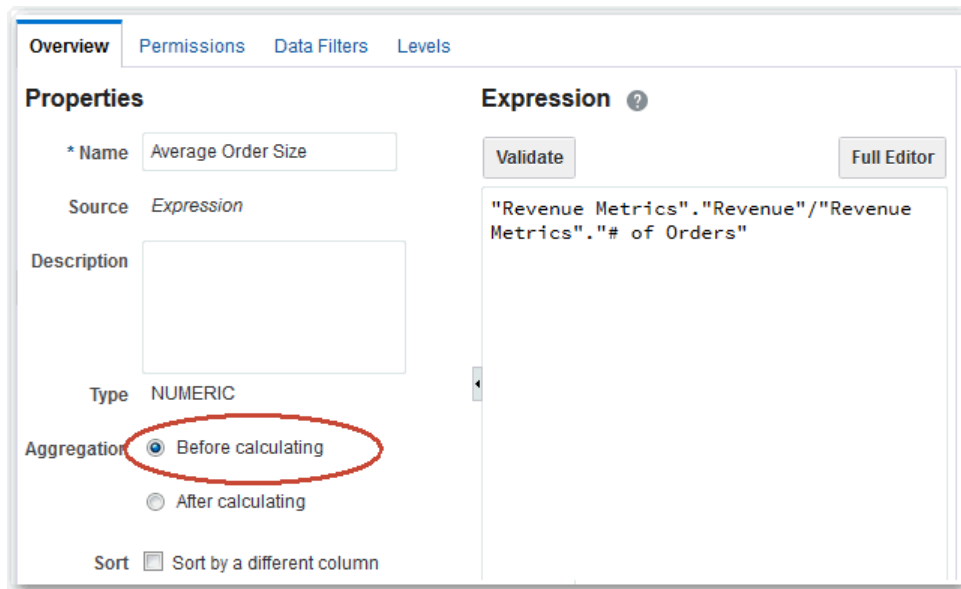
- Calculation includes aggregated measures: $\text{Sum}(\text{Revenue}) / \text{Sum}(\text{Orders})$
- Calculation includes measures with no aggregation applied: $\text{UnitPrice} \times \text{Quantity}$

If the measures in your calculation aren't pre-aggregated, such as `UnitPrice` and `Quantity`, you may apply aggregation after the calculation. For example, $\text{Sum}(\text{UnitPrice} \times \text{Quantity})$.

Check the measures in your calculations before choosing whether to apply aggregation **Before Calculating** or **After Calculating** your expression.

Calculations Include Measures Already Aggregated

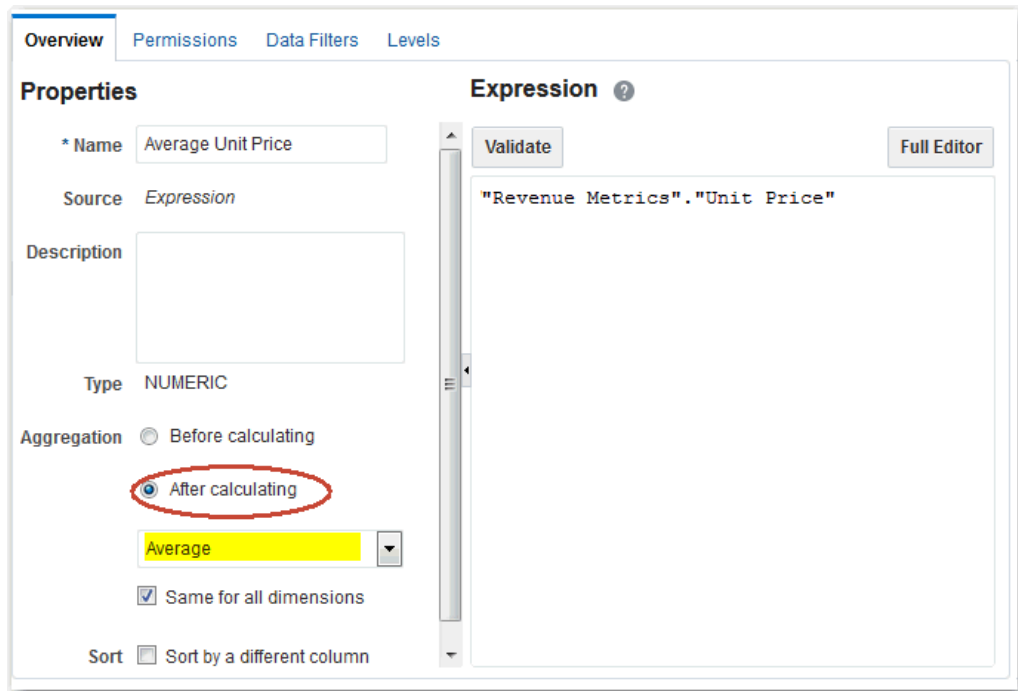
Set Aggregation to **Before calculating** if the calculation contains pre-aggregated measures. For example: $\text{Sum}(\text{Revenue}) / \text{Sum}(\text{Orders})$.



Calculations Include Non Aggregated Measures

Optionally, you can apply aggregation after your calculation. Set Aggregation to **After calculating** and then select an aggregation rule from the list. For example, **Sum**, **Average**, **Count** and so on.

Don't include expression columns in the calculation. If you include aggregated columns in the calculation, aggregation on the columns is ignored.

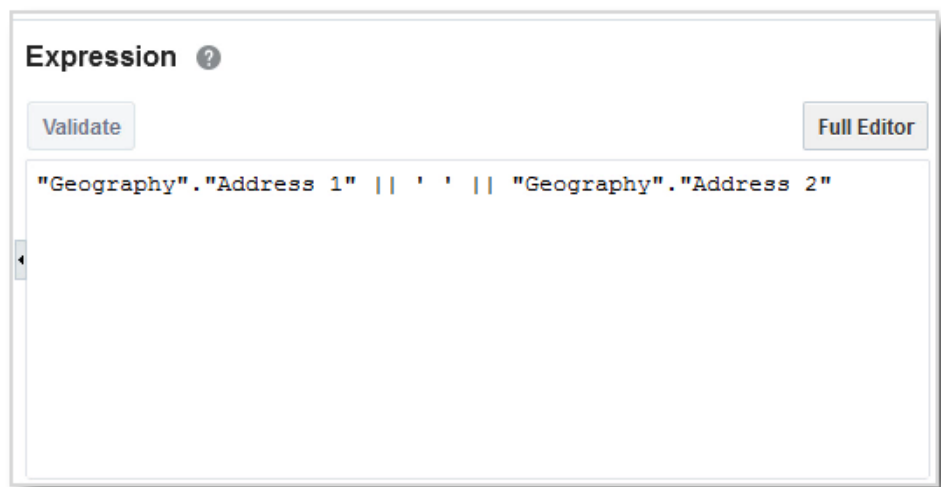


Create Derived Attributes

You can create custom or derived attributes for dimension tables that are based on an expression. For example, you can use an expression to concatenate multiple address columns into a single Full Address column.

1. In Data Modeler, lock the model for editing.
2. In the Dimension Tables area, click the dimension table for which you want to create derived attributes.
3. In the Columns area, click **Add Column**.
4. In the New Column editor, enter a name and description for the column. Then, enter an expression directly in the Expression box, or click **Full Editor** to display the Expression editor.

You can use a variable in a column expression.



5. Click **Done** to return to the table editor.

Create Expressions in the Expression Editor

You can use the Expression Editor to create constraints, aggregations, and other transformations on columns.

Topics:

- [About the Expression Editor](#)
- [Create an Expression](#)

About the Expression Editor

When modeling data, you can use the Expression Editor to create constraints, aggregations, and other transformations on columns. For example, you can use the Expression Editor to change the data type of a column from date to character. You can also use the Expression Editor to create expressions for data filters.

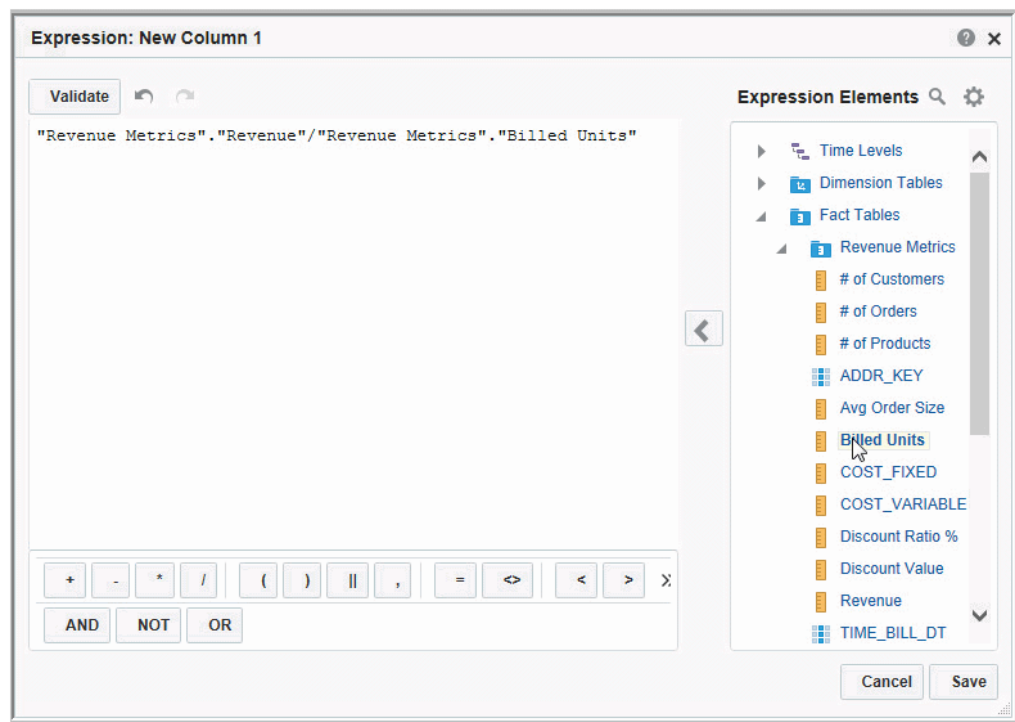
The Expression Editor contains the following sections:

- The Expression box on the left-hand side enables you to edit the current expression.
- The toolbar at the bottom contains commonly used expression operators, such as a plus sign, equals sign, or comma to separate items.
- The Expression Elements section on the right-hand side provides building blocks that you can use in your expression. Examples of elements are tables, columns, functions, and types.

The Expression Elements section only includes items that are relevant for your task. For example, if you open the Expression Editor to define a calculated measure, the Expression Elements section only includes the current fact table, any dimension tables joined to that table, plus any fact tables indirectly joined through a dimension table. Similarly, when you define a derived attribute, you see the current dimension table, any fact tables joined to that table, and any dimension table joined to those fact tables.

Another example is that time hierarchies are only included if the Time fact table is joined to the current table.

See [Expression Editor Reference](#).



Create an Expression

You can use the Expression Editor to create constraints, aggregations, and other transformations on columns.

1. Add or edit a column from the Table editor.
2. Enter an expression in the Expression box and click **Done**. Or, click **Full Editor** to launch the Expression Editor.

3. Use the Expression Elements menus to locate the building blocks you want to use to create your expression.

Drag and drop an element to add it to your expression. You can also double-click an element to insert it, or you can select the element and click the arrow icon.

When you add a function, brackets indicate text that needs to be replaced. Select the text, then type, or use the Expression Elements menus to add the appropriate element.

See [Expression Editor Reference](#).

4. Click **Filter** and then enter text in the search box to filter the available elements. Remove the text to revert to the full list of elements.
5. Click **Actions** to show or hide menus under Expression Elements, or to expand or collapse all menus.
6. Click an item on the toolbar to insert an operator.
7. Click **Undo** or **Redo** as needed when building your expression.
8. Click **Validate** to check your work.
9. Click **Save** when you're finished.

Copy Measures and Attributes

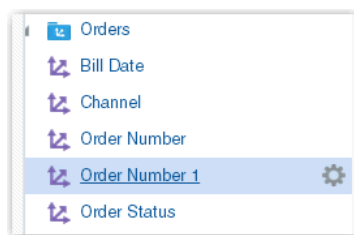
You can copy measures and attributes in your semantic model.

- From the Data Model menu in the left pane, right-click the column that you want to copy and select **Copy**.

To copy multiple columns, Shift + click or Ctrl + click all the rows that you want and right-click to select **Copy**.

- From the Data Model menu in the left pane, click **Column Actions** for the column that you want to copy and select **Copy**.

The copy is displayed with a number added to the name.



Copy Model Objects

Sometimes it's quicker to copy objects rather than starting from scratch.

In Data Modeler you can copy fact tables, dimension tables, database tables, and database views:

- **Fact tables**

To copy an existing fact table, select **Duplicate** from the **Fact Table Actions** menu. When you copy a fact table, Data Modeler includes joins by default. See [Create Fact Tables Individually](#).

Aggregation level settings for measures aren't copied as, in most cases, level settings in the original fact table and the copied version differ. After copying a fact table, review and set the aggregation levels for measures as required.

- **Dimension tables**

To copy an existing dimension table, select **Duplicate** from the **Dimension Table Actions** menu. When you copy a dimension table, Data Modeler excludes joins default. See [Create Dimension Tables Individually](#).

- **Database tables and views**

To copy an existing database object, select **Duplicate** from the **Actions** menu. When you copy a table or view, Data Modeler creates a view based on the table or view you copy. See [Add Your Own Source Views](#).

5

Define Hierarchies and Levels to Drill and Aggregate

You can define hierarchies and levels in Data Modeler.

Topics:

- [Typical Workflow to Define Hierarchies and Levels](#)
- [About Hierarchies and Levels](#)
- [Edit Hierarchies and Levels](#)
- [Set Aggregation Levels for Measures](#)

Typical Workflow to Define Hierarchies and Levels

Here are the common tasks to add hierarchies and levels to your semantic model.

Task	Description	More Information
Add hierarchies and levels	Create hierarchies and levels for your dimension tables	Edit Hierarchies and Levels
Set aggregation levels for measures	Set custom aggregation levels for measures that are different from the default level	Set Aggregation Levels for Measures

About Hierarchies and Levels

A hierarchy shows relationships among groups of columns in a dimension table. For example, quarters contain months and months contain days. Hierarchies enable drilling in reports.

A dimension table can have one or more hierarchies. A hierarchy typically begins with a total level, then has child levels, working down to the lowest detail level.

All hierarchies for a given dimension must have a common lowest level. For example, a time dimension might contain a fiscal hierarchy and a calendar hierarchy, with Day as the common lowest level. Day has two named parent levels called Fiscal Year and Calendar Year, which are both children of the All root level.

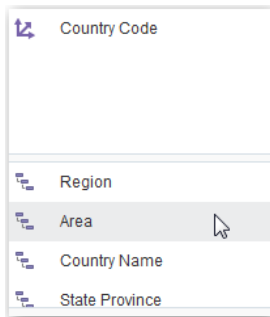
All levels, except the total level, must have at least one column specified as the key or display column. However, it's not necessary to explicitly associate all of the columns from a table with levels. Any column that you don't associate with a level is automatically associated with the lowest level in the hierarchy that corresponds to that dimension table.

There's no limit to the number of levels you can have in a hierarchy. The total number of levels isn't by itself a determining factor in query performance. However, be aware that for extremely complex queries, even a few levels can impact performance.

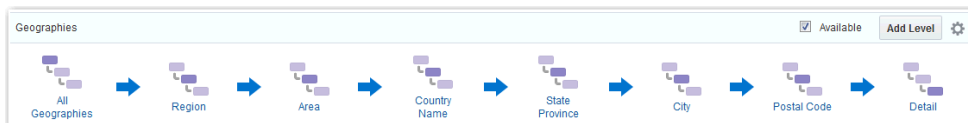
Edit Hierarchies and Levels

When fact tables and dimension tables are joined, a default hierarchy is created but you can also add hierarchies and levels to those tables. For example, a Geography hierarchy might include levels for Country, State, and City.

1. In Data Modeler, lock the model for editing.
2. In the Dimension Tables area, click the dimension table for which you want to add a hierarchy. The dimension table must have at least one join to a fact table.
3. In the Dimension editor, click the Hierarchies tab.
4. In the Hierarchies area, click **Add Level**, and select the dimension columns or shared levels that you want to use.



5. Drag and drop levels to a different location in the order, as appropriate. You can also right-click a level and select **Move left** or **Move right**.



6. Click a level to display a dialog in which you can specify the level name, the key column, and the display column for the level.
7. Deselect **Available** if you don't want the hierarchy visible in analyses.
8. Click **Done** when you're finished.

Set Dimension Table Properties for Hierarchies

From the Overview tab for a particular dimension table, you can set properties that apply to all hierarchies for that table.

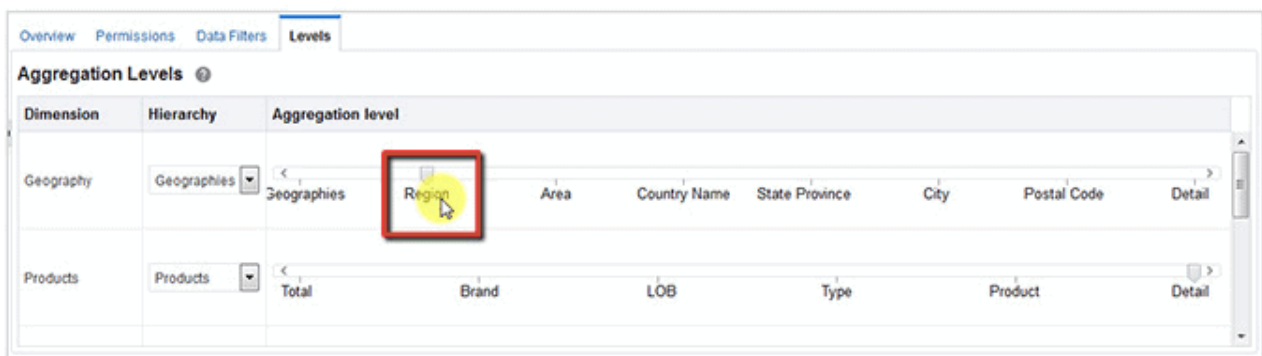
1. In Data Modeler, lock the model for editing.
2. Click the dimension table that you want to edit.
3. On the Overview tab, set properties as required.
 - **Time dimension** — Specifies that hierarchies for this dimension table support a time dimension. Hierarchies for time dimensions can't include skip levels or be unbalanced.

- **Enable skipped levels** — Specifies that this dimension table supports hierarchies with skipped levels. A skip-level hierarchy is a hierarchy where there are members that do not have a value for a particular ancestor level. For example, in a Country-State-City-District hierarchy, the city "Washington, D.C." does not belong to a State. In this case, you can drill down from the Country level (USA) to the City level (Washington, D.C.) and below. In a query, skipped levels aren't displayed, and don't affect computations. When sorted hierarchically, members appear under their nearest ancestors.
- **Enable unbalanced hierarchies** — Specifies that this dimension table supports unbalanced hierarchies. An unbalanced (or ragged) hierarchy is a hierarchy where the leaves (members with no children) don't necessarily have the same depth. For example, a site can choose to have data for the current month at the day level, previous months data at the month level, and the previous 5 years data at the quarter level.

Set Aggregation Levels for Measures

When fact tables and dimension tables are joined, you can set custom aggregation levels for a measure.

1. In Data Modeler, lock the model for editing.
2. In the Fact Tables area, click the fact table in which the measure is located.
3. Specify the aggregation rule for the new column that you want to become the level-based measure.
4. Click the column name, then click **Levels**.
5. In the Levels tab, for one or more hierarchies, use the slider to select the aggregation level for the measure.



6. Click **Done** to return to the table editor.

About Setting Aggregation Levels for Measures

By default, measures are aggregated at the level of the dimension attributes that are selected in an analysis. For example, in an analysis that includes Sales Person and Revenue columns, the Revenue is aggregated at the level of a Sales Person.

To calculate ratios, you often need measures that are aggregated at a level that is different than the grain of the analysis. For example, to calculate the Revenue Percent Contribution for a Sales Person with respect to his department, you need Department Revenue at the Sales

Person level in an analysis (Sales Person, Revenue, Revenue *100 / Revenue@Dept). In this example, Revenue@Dept has a custom aggregation level that is different from the default level.

6

Secure Your Semantic Model

You can define object-level permissions and row-level security data filters for your semantic model.

Topics:

- [Typical Workflow to Secure Model Data](#)
- [Create Variables to Use in Expressions](#)
- [Secure Access to Objects in the Model](#)
- [Secure Access to Data](#)

Typical Workflow to Secure Model Data

Here are the common tasks to secure your semantic model.

Task	Description	More Information
Define variables for data filters, if needed	Optionally, create variables that dynamically calculate and store values for use in column expressions and data filters.	Create Variables to Use in Expressions
Set permissions on model objects	Object permissions control visibility for your entire model, or individual fact tables, dimension tables, and columns.	Secure Access to Objects in the Model
Define row-level security filters	Data filters limit results returned for fact tables, dimension tables, and columns.	Secure Access to Data

Create Variables to Use in Expressions

In Data Modeler, you can define variables that dynamically calculate and store values so that you can use those values in column expressions or data filters.

Topics:

- [About Variables](#)
- [Define Variables](#)

About Variables

Variables dynamically calculate and store values so that you can use those values in expressions. You can use variables in column expressions, or in data filters.

For example, suppose User1 belongs to Department1 and User2 belongs to Department2. Each user must access only the data that is specific to his department. You can use the DEPARTMENT_NUMBER variable to store the appropriate values for User1 and User2. You can use this variable in a data filter in which the data is filtered by Department1 for User1 and Department2 for User2. In other words, variables dynamically modify metadata content to adjust to a changing data environment.

Values in variables aren't secure, because object permissions don't apply to variables. Anybody who knows or can guess the name of the variable can use it in an expression. Because of this, don't put sensitive data like passwords in variables.

You can't use a variable in an expression that defines another variable.

Define Variables

You can create a variable for use in column expressions and data filters. For example, a variable called SalesRegion might use a SQL query to retrieve the name of the sales region of the user.



Tip:

Only reference source database objects in the SQL query for a variable. Don't include names of semantic model objects in the query.

1. In Data Modeler, lock the model for editing.
2. In the Variables menu in the left pane, click the **Plus** icon.
3. Enter a SQL query to populate the value of the variable:
 - a. Specify whether the variable returns **A single value** or **Multiple values**.
 - b. Enter a SQL query to populate the value or values of the variable. For example:
 - Return a single value with the query like: `SELECT prod-name FROM products`
 - Return multiple values with a query like: `SELECT 'MyVariable', prod-name FROM products`For multiple values, always use the format: `SELECT 'VariableName', VariableValue FROM Table`
 - c. Provide a default starting value if needed.
 - d. Click **Test** to validate that the query returns an appropriate value

4. To create a variable that refreshes its value at the start of each user session, select **On sign in** for **Update Value**.
5. To create a variable that refreshes its value on a schedule that you set, select **On a schedule** for **Update Value**.

In the **Run SQL Query** area, select the frequency and start date for refreshing the variable.

6. To create a variable with a static value that never changes, select **Never** for **Update Value** and provide a value for the variable in the **Value** field.
7. Click **Done** to return to the semantic model.

 **Tip:**

To edit an existing variable, right-click it in the Variables list and select **Inspect**. To delete a variable, right-click it and select **Delete**.

After you have defined a variable, you can use it in a data filter or in a column expression.

Secure Access to Objects in the Model

It's important to keep sensitive information secure. Everyone has access to the data in your model by default. To avoid exposing sensitive data, set show and hide permissions for your entire model or for individual fact tables, dimension tables, and columns.

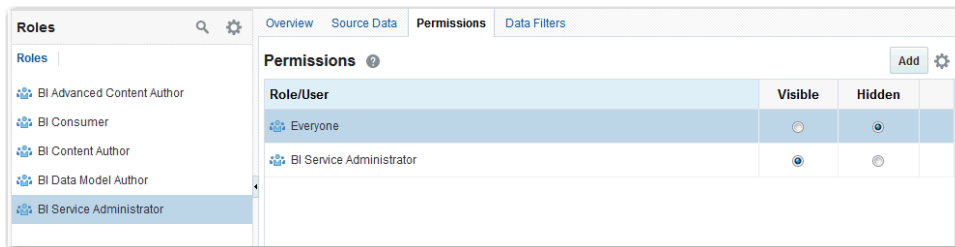
For example, you can restrict access to certain Revenue columns to ensure only authorized users can view them. Or you can restrict access to an entire model to stop people opening the model or accessing its subject area.

1. In Data Modeler, lock the model for editing.
2. To restrict access to the whole model, select the **Permissions** tab.

To restrict access to a specific item in the model, edit the fact table, dimension table, or column whose access you want to secure, then select the **Permissions** tab.

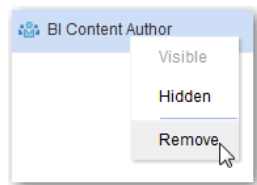
3. To control access, click **Add** and select the appropriate role.

Alternatively, in the left pane, click **Roles**. Then, drag and drop a role to the Permissions list. To add multiple roles, use Shift + click or Ctrl + click to make your selections before you drag and drop.



4. Specify whether or not this object is visible to users with that role by selecting either **Visible** or **Hidden**.
 - **Models** — If you hide a model, users with that role can't open the model or its subject area.
 - **Model objects** — If you hide a fact table, dimension table, or column, users with that role can't see the object in reports.

The same users will see the object in Data Modeler if they have the BI Data Model Author role and have access to the model.
5. To remove roles from the Permissions list (you can't remove the Everyone role), do one of the following:
 - Right-click a role and select **Remove**.



- Select **Remove** from the Actions menu for that role.
- Select multiple roles using Shift + click or Ctrl + click, then select **Remove Selected** from the Permissions Action menu.
- Remove all roles by selecting **Remove All** from the Permissions Action menu.

About Permission Inheritance

When multiple application roles act on a user or role with conflicting security attributes, the user or role is granted the least restrictive security attribute. Also, any explicit permissions acting on a user take precedence over any permissions on the same objects granted to that user through application roles.

Tip:

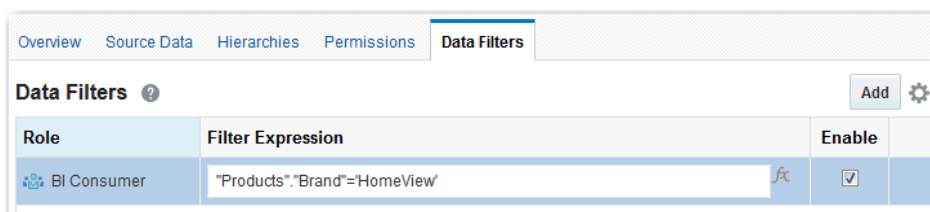
If you deny access to a table, access to all columns in that table is implicitly denied as well.

Secure Access to Data

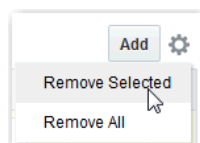
You can define data filters for fact tables, dimension tables, and columns that provide row-level security for data model objects. For example, you can create a filter that restricts access to the Products table so that only certain brands are visible to users assigned to a particular role.

1. In Data Modeler, lock the model for editing.
2. Edit the fact table, dimension table, or column you want to secure.
3. Select the **Data Filters** tab.
4. Add a role to the Data Filters list by doing one of the following:
 - Click **Add** and select the appropriate role.
 - In the left pane, click **Roles**. Then, drag and drop a role to the Data Filters list.
5. Enter an expression to specify which data is accessible for that role. Either enter the expression directly, or click **Full Editor** to display the Expression Editor.

You can use a variable in a data filter expression.



6. Select **Enable** to specify whether the filter is enabled for that role.
7. To remove filters from the Data Filters list, do one of the following:
 - Right-click a filter and select **Remove**.
 - Select **Remove** from the Actions menu for that filter.
 - Select multiple filters using Shift-click or Ctrl-click, then select **Remove Selected** from the Data Filters Action menu.



- Remove all filters by selecting **Remove All** from the Data Filters Action menu.
8. Click **Done**.

7

Expression Editor Reference

This section describes the expression elements that you can use in the Expression Editor.

Topics:

- [Semantic Model Objects](#)
- [SQL Operators](#)
- [Conditional Expressions](#)
- [Functions](#)
- [Constants](#)
- [Types](#)
- [Variables](#)

Semantic Model Objects

You can use semantic model objects in expressions, like time levels, dimension columns, and fact columns.

To reference a semantic model object, use the syntax:

"Fact/Dimension Table Name"."Column Name"

For example: "Order Metrics"."Booked Amount"-"Order Metrics"."Fulfilled Amount"

The Expression Elements section includes only items that are relevant for your task, so not all fact tables and dimension tables might be listed. Similarly, time hierarchies are included only if the Time fact table is joined to the current table.

SQL Operators

SQL operators are used to specify comparisons between expressions.

You can use various types of SQL operators.

Operator	Example	Description	Syntax
BETWEEN	"COSTS"."UNIT_COST" BETWEEN 100.0 AND 5000.0	Determines if a value is between two non-inclusive bounds. BETWEEN can be preceded with NOT to negate the condition.	BETWEEN [LowerBound] AND [UpperBound]
IN	"COSTS"."UNIT_COST" IN(200, 600, 'A')	Determines if a value is present in a set of values.	IN ([Comma Separated List])
IS NULL	"PRODUCTS"."PRODUCT_NAME" IS NULL	Determines if a value is null.	IS NULL

Operator	Example	Description	Syntax
LIKE	"PRODUCTS"."PRO D_NAME" LIKE 'prod%'	Determines if a value matches all or part of a string. Often used with wildcard characters to indicate any character string match of zero or more characters (%) or any single character match (_).	LIKE
+	(FEDERAL_REVENUE + LOCAL_REVENUE) - TOTAL_EXPENDITURE	Plus sign for addition.	+
-	(FEDERAL_REVENUE + LOCAL_REVENUE) - TOTAL_EXPENDITURE	Minus sign for subtraction.	-
* or X	SUPPORT_SERVICE * S_EXPENDITURE * 1.5	Multiply sign for multiplication.	* X
/	CAPITAL_OUTLAY_ EXPENDITURE/ 1.05	Divide by sign for division.	/
%		Percentage	%
	STATE CAST (YEAR AS CHAR (4))	Character string concatenation.	
((FEDERAL_REVENUE + LOCAL_REVENUE) - TOTAL_EXPENDITURE	Open parenthesis.	(
)	(FEDERAL_REVENUE + LOCAL_REVENUE) - TOTAL_EXPENDITURE	Close parenthesis.)
>	YEAR > 2000 and YEAR < 2016 and YEAR <> 2013	Greater than sign, indicating values higher than the comparison.	>
<	YEAR > 2000 and YEAR < 2016 and YEAR <> 2013	Less than sign, indicating values lower than the comparison.	<
=		Equal sign, indicating the same value.	=

Operator	Example	Description	Syntax
>=		Greater than or equal to sign, indicating values the same or higher than the comparison.	>=
<=		Less than or equal to sign, indicating values the same or lower than the comparison.	<=
<>	YEAR > 2000 and YEAR < 2016 and YEAR <> 2013	Not equal to, indicating values higher or lower, but different.	<>
,	STATE in ('ALABAMA', 'CAL IFORNIA')	Comma, used to separate elements in a list.	,

Conditional Expressions

You use conditional expressions to create expressions that convert values.

The conditional expressions described in this section are building blocks for creating expressions that convert a value from one form to another.

Follow these rules:

- In **CASE** statements, **AND** has precedence over **OR**.
- Strings must be in single quotes.

Expression	Example	Description	Syntax
CASE (If)	CASE WHEN score-par < 0 THEN 'Under Par' WHEN score-par = 0 THEN 'Par' WHEN score-par = 1 THEN 'Bogey' WHEN score-par = 2 THEN 'Double Bogey' ELSE 'Triple Bogey or Worse' END	Evaluates each WHEN condition and if satisfied, assigns the value in the corresponding THEN expression. If none of the WHEN conditions are satisfied, it assigns the default value specified in the ELSE expression. If no ELSE expression is specified, the system automatically adds an ELSE NULL . Note: See <i>Best Practices for using CASE statements in Analyses and Visualizations</i> .	CASE WHEN request_condition1 THEN expr1 ELSE expr2 END

Expression	Example	Description	Syntax
CASE (Switch)	<pre> CASE Score-par WHEN -5 THEN 'Birdie on Par 6' WHEN -4 THEN 'Must be Tiger' WHEN -3 THEN 'Three under par' WHEN -2 THEN 'Two under par' WHEN -1 THEN 'Birdie' WHEN 0 THEN 'Par' WHEN 1 THEN 'Bogey' WHEN 2 THEN 'Double Bogey' ELSE 'Triple Bogey or Worse' END </pre>	<p>Also referred to as CASE (Lookup). The value of the first expression is examined, then the WHEN expressions. If the first expression matches any WHEN expression, it assigns the value in the corresponding THEN expression.</p> <p>If none of the WHEN expressions match, it assigns the default value specified in the ELSE expression. If no ELSE expression is specified, the system automatically adds an ELSE NULL.</p> <p>If the first expression matches an expression in multiple WHEN clauses, only the expression following the first match is assigned.</p> <p>Note See <i>Best Practices for using CASE statements in Analyses and Visualizations</i>.</p>	<pre> CASE expr1 WHEN expr2 THEN expr3 ELSE expr4 END </pre>
IfCase > ELSE	-	-	ELSE [expr]
IfCase > IFNULL	-	-	IFNULL([expr], [value])
IfCase > NULLIF	-	-	NULLIF([expr], [expr])
IfCase > WHEN	-	-	WHEN [Condition] THEN [expr]
IfCase > CASE	-	-	CASE WHEN [Condition] THEN [expr] END
SwitchCase > ELSE	-	-	ELSE [expr]
SwitchCase > >IFNULL	-	-	IFNULL([expr], [value])
SwitchCase > NULLIF	-	-	NULLIF([expr], [expr])
SwitchCase > WHEN	-	-	WHEN [Condition] THEN [expr]

Functions

There are various types of functions that you can use in expressions.

Topics:

- [Aggregate Functions](#)
- [Analytics Functions](#)
- [Conversion Functions](#)
- [Date and Time Functions](#)
- [Date Extraction Functions](#)
- [Display Functions](#)
- [Evaluate Functions](#)
- [Mathematical Functions](#)
- [Running Aggregate Functions](#)
- [Spatial Functions](#)
- [String Functions](#)
- [System Functions](#)
- [Time Series Functions](#)

Aggregate Functions

Aggregate functions perform operations on multiple values to create summary results.

The following list describes the aggregation rules that are available for columns and measure columns. The list also includes functions that you can use when creating calculated items for analyses.

- **Default** — Applies the default aggregation rule as in the semantic model or by the original author of the analysis. Not available for calculated items in analyses.
- **Server Determined** — Applies the aggregation rule that's determined by the Oracle Analytics (such as the rule that is defined in the semantic model). The aggregation is performed within Oracle Analytics for simple rules such as Sum, Min, and Max. Not available for measure columns in the Layout pane or for calculated items in analyses.
- **Sum** — Calculates the sum obtained by adding up all values in the result set. Use this for items that have numeric values.
- **Min** — Calculates the minimum value (lowest numeric value) of the rows in the result set. Use this for items that have numeric values.
- **Max** — Calculates the maximum value (highest numeric value) of the rows in the result set. Use this for items that have numeric values.
- **Average** — Calculates the average (mean) value of an item in the result set. Use this for items that have numeric values. Averages on tables and pivot tables are rounded to the nearest whole number.

- **First** — In the result set, selects the first occurrence of the item for measures. For calculated items, selects the first member according to the display in the Selected list. Not available in the Edit Column Formula dialog box.
- **Last** — In the result set, selects the last occurrence of the item. For calculated items, selects the last member according to the display in the Selected list. Not available in the Edit Column Formula dialog box.
- **Count** — Calculates the number of rows in the result set that have a non-null value for the item. The item is typically a column name, in which case the number of rows with non-null values for that column are returned.
- **Count Distinct** — Adds distinct processing to the Count function, which means that each distinct occurrence of the item is counted only once.
- **None** — Applies no aggregation. Not available for calculated items in analyses.
- **Report-Based Total (when applicable)** — If not selected, specifies that the Oracle Analytics should calculate the total based on the entire result set, before applying any filters to the measures. Not available in the Edit Column Formula dialog box or for calculated items in analyses. Only available for attribute columns.

Function	Example	Description	Syntax
AGGREGATE AT	AGGREGATE(sales AT year)	Aggregates columns based on the level or levels in the data model hierarchy you specify. <ul style="list-style-type: none"> • <i>measure</i> is the name of a measure column. • <i>level</i> is the level at which you want to aggregate. <p>You can optionally specify more than one level. You can't specify a level from a dimension that contains levels that are being used as the measure level for the measure you specified in the first argument. For example, you can't write the function as AGGREGATE(yearly_sales AT month) if <i>month</i> is from the same time dimension used as the measure level for <i>yearly_sales</i>.</p>	AGGREGATE(measure AT level [, level1, levelN])
AGGREGATE BY	AGGREGATE(sales BY month, region)	Aggregates a measure based on one or more dimension columns. <ul style="list-style-type: none"> • <i>measure</i> is the name of a measure column to aggregate. • <i>column</i> is the dimension column at which you want to aggregate. <p>You can aggregate measures based more than one column.</p>	AGGREGATE(measure BY column [, column1, columnN])
AVG	Avg(Sales)	Calculates the average (mean) of a numeric set of values.	AVG(expr)
AVGDISTINCT		Calculates the average (mean) of all distinct values of an expression.	AVG(DISTINCT expr)

Function	Example	Description	Syntax
BIN	<pre>BIN(revenue BY productid, year WHERE productid > 2 INTO 4 BINS RETURNING RANGE_LOW)</pre>	Classifies a given numeric expression into a specified number of equal width buckets. The function can return either the bin number or one of the two end points of the bin interval. <i>numeric_expr</i> is the measure or numeric attribute to bin. BY <i>grain_expr1</i> ,..., <i>grain_exprN</i> is a list of expressions that define the grain at which the <i>numeric_expr</i> is calculated. BY is required for measure expressions and is optional for attribute expressions. WHERE a filter to apply to the <i>numeric_expr</i> before the numeric values are assigned to bins INTO <i>number_of_bins</i> BINS is the number of bins to return BETWEEN <i>min_value</i> AND <i>max_value</i> is the min and max values used for the end points of the outermost bins RETURNING NUMBER indicates that the return value should be the bin number (1, 2, 3, 4, etc.). This is the default. RETURNING RANGE_LOW indicates the lower value of the bin interval RETURNING RANGE_HIGH indicates the higher value of the bin interval	<pre>BIN(numeric_expr [BY grain_expr1, ..., grain_exprN] [WHERE condition] INTO number_of_bins BINS [BETWEEN min_value AND max_value] [RETURNING {NUMBER RANGE_LOW RANGE_HIGH}])</pre>
BottomN		Ranks the lowest n values of the expression argument from 1 to n, 1 corresponding to the lowest numerical value. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer. Represents the bottom number of rankings displayed in the result set, 1 being the lowest rank.	<pre>BottomN(expr, integer)</pre>
COUNT	<pre>COUNT(Products)</pre>	Determines the number of items with a non-null value.	<pre>COUNT(expr)</pre>
COUNTDISTINCT		Adds distinct processing to the COUNT function. <i>expr</i> is any expression.	<pre>COUNT(DISTINCT expr)</pre>
COUNT*	<pre>SELECT COUNT(*) FROM Facts</pre>	Counts the number of rows.	<pre>COUNT(*)</pre>
First	<pre>First(Sales)</pre>	Selects the first non-null returned value of the expression argument. The <i>First</i> function operates at the most detailed level specified in your explicitly defined dimension.	<pre>First([NumericExpression])</pre>
Last	<pre>Last(Sales)</pre>	Selects the last non-null returned value of the expression.	<pre>Last([NumericExpression])</pre>
MAVG		Calculates a moving average (mean) for the last n rows of data in the result set, inclusive of the current row. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer. Represents the average of the last n rows of data.	<pre>MAVG(expr, integer)</pre>

Function	Example	Description	Syntax
MAX	MAX (Revenue)	Calculates the maximum value (highest numeric value) of the rows satisfying the numeric expression argument.	MAX (expr)
MEDIAN	MEDIAN (Sales)	Calculates the median (middle) value of the rows satisfying the numeric expression argument. When there are an even number of rows, the median is the mean of the two middle rows. This function always returns a double.	MEDIAN (expr)
MIN	MIN (Revenue)	Calculates the minimum value (lowest numeric value) of the rows satisfying the numeric expression argument.	MIN (expr)
NTILE		Determines the rank of a value in terms of a user-specified range. It returns integers to represent any range of ranks. NTILE with numTiles=100 returns what is commonly called the "percentile" (with numbers ranging from 1 to 100, with 100 representing the high end of the sort). <i>expr</i> is any expression that evaluates to a numerical value. numTiles is a positive, nonnull integer that represents the number of tiles.	NTILE (expr, numTiles)
PERCENTILE		Calculates a percentile rank for each value satisfying the numeric expression argument. The percentile rank ranges are between 0 (0th percentile) to 1 (100th percentile). <i>expr</i> is any expression that evaluates to a numerical value.	PERCENTILE (expr)
RANK	RANK (chronological_key, null, year_key_columns)	Calculates the rank for each value satisfying the numeric expression argument. The highest number is assigned a rank of 1, and each successive rank is assigned the next consecutive integer (2, 3, 4,...). If certain values are equal, they're assigned the same rank (for example, 1, 1, 1, 4, 5, 5, 7...). <i>expr</i> is any expression that evaluates to a numerical value.	RANK (expr)
STDDEV	STDDEV (Sales) STDDEV (DISTINCT Sales)	Returns the standard deviation for a set of values. The return type is always a double.	STDDEV (expr)
STDDEV_POP	STDDEV_POP (Sales) STDDEV_POP (DISTINCT Sales)	Returns the standard deviation for a set of values using the computational formula for population variance and standard deviation.	STDDEV_POP ([NumericExpression])
SUM	SUM (Revenue)	Calculates the sum obtained by adding up all values satisfying the numeric expression argument.	SUM (expr)

Function	Example	Description	Syntax
SUMDISTINCT		Calculates the sum obtained by adding all of the distinct values satisfying the numeric expression argument. <i>expr</i> is any expression that evaluates to a numerical value.	SUM(DISTINCT <i>expr</i>)
TOPN		Ranks the highest n values of the expression argument from 1 to n, 1 corresponding to the highest numerical value. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer. Represents the top number of rankings displayed in the result set, 1 being the highest rank.	TOPN(<i>expr</i> , <i>integer</i>)

Analytics Functions

Analytics functions allow you to explore data using models such as trendline and cluster.

Function	Example	Description	Syntax
TRENDLINE	TRENDLINE(<i>revenue</i> , (<i>calendar_year</i> , <i>calendar_quarter</i> , <i>calendar_month</i>) BY (<i>product</i>), 'LINEAR', 'VALUE')	Oracle recommends that you apply a Trendline using the Add Statistics property when viewing a visualization. See Adjust Visualization Properties. Fits a linear, polynomial, or exponential model, and returns the fitted values or model. The <i>numeric_expr</i> represents the Y value for the trend and the <i>series</i> (time columns) represent the X value.	TRENDLINE(<i>numeric_expr</i> , ([<i>series</i>]) BY ([<i>partitionBy</i>]), <i>model_type</i> , <i>result_type</i>)
CLUSTER	CLUSTER((<i>product</i> , <i>company</i>), (<i>billed_quantity</i> , <i>revenue</i>), ' <i>clusterName</i> ', 'algorithm=k- means;numClusters=%1;maxI ter=%2;useRandomSeed=FALSE; enablePartitioning=TRUE' , 5, 10)	Collects a set of records into groups based on one or more input expressions using K-Means or Hierarchical Clustering.	CLUSTER((<i>dimension_expr1</i> , ... <i>dimension_exprN</i>), (<i>expr1</i> , ... <i>exprN</i>), <i>output_column_name</i> , <i>options</i> , [<i>runtime_binded_options</i>])
OUTLIER	OUTLIER((<i>product</i> , <i>company</i>), (<i>billed_quantity</i> , <i>revenue</i>), ' <i>isOutlier</i> ', 'algorithm=kmeans')	Classifies a record as Outlier based on one or more input expressions using K-Means or Hierarchical Clustering or Multi-Variate Outlier detection Algorithms.	OUTLIER((<i>dimension_expr1</i> , ... <i>dimension_exprN</i>), (<i>expr1</i> , ... <i>exprN</i>), <i>output_column_name</i> , <i>options</i> , [<i>runtime_binded_options</i>])

Function	Example	Description	Syntax
REGR	REGR(revenue, (discount_amount), (product_type, brand), 'fitted', '')	Fits a linear model and returns the fitted values or model. This function can be used to fit a linear curve on two measures.	REGR(y_axis_measure_expr, (x_axis_expr), (category_expr1, ..., category_exprN), output_column_name, options, [runtime_binded_options])

Date and Time Functions

Date and time functions manipulate data based on DATE and DATETIME.

Function	Example	Description	Syntax
CURRENT_Date	CURRENT_DATE	Returns the current date. The date is determined by the system in which the Oracle BI is running.	CURRENT_DATE
CURRENT_TIME	CURRENT_TIME(3)	Returns the current time to the specified number of digits of precision, for example: HH:MM:SS.SSS If no argument is specified, the function returns the default precision.	CURRENT_TIME(expr)
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP(3)	Returns the current date/timestamp to the specified number of digits of precision.	CURRENT_TIMESTAMP(expr)
DAYNAME	DAYNAME(Order_Date)	Returns the name of the day of the week for a specified date expression.	DAYNAME(expr)
DAYOFMONTH	DAYOFMONTH(Order_Date)	Returns the number corresponding to the day of the month for a specified date expression.	DAYOFMONTH(expr)
DAYOFWEEK	DAYOFWEEK(Order_Date)	Returns a number between 1 and 7 corresponding to the day of the week for a specified date expression. For example, 1 always corresponds to Sunday, 2 corresponds to Monday, and so on through to Saturday which returns 7.	DAYOFWEEK(expr)
DAYOFYEAR	DAYOFYEAR(Order_Date)	Returns the number (between 1 and 366) corresponding to the day of the year for a specified date expression.	DAYOFYEAR(expr)
DAY_OF_QUARTER	DAY_OF_QUARTER(Order_Date)	Returns a number (between 1 and 92) corresponding to the day of the quarter for the specified date expression.	DAY_OF_QUARTER(expr)
HOUR	HOUR(Order_Time)	Returns a number (between 0 and 23) corresponding to the hour for a specified time expression. For example, 0 corresponds to 12 a.m. and 23 corresponds to 11 p.m.	HOUR(expr)
MINUTE	MINUTE(Order_Time)	Returns a number (between 0 and 59) corresponding to the minute for a specified time expression.	MINUTE(expr)

Function	Example	Description	Syntax
MONTH	MONTH(Order_Time)	Returns the number (between 1 and 12) corresponding to the month for a specified date expression.	MONTH(expr)
MONTHNAME	MONTHNAME(Order_Time)	Returns the name of the month for a specified date expression.	MONTHNAME(expr)
MONTH_OF_QUARTER	MONTH_OF_QUARTER(Order_Date)	Returns the number (between 1 and 3) corresponding to the month in the quarter for a specified date expression.	MONTH_OF_QUARTER(expr)
NOW	NOW()	Returns the current timestamp. The NOW function is equivalent to the CURRENT_TIMESTAMP function.	NOW()
QUARTER_OF_YEAR	QUARTER_OF_YEAR(Order_Date)	Returns the number (between 1 and 4) corresponding to the quarter of the year for a specified date expression.	QUARTER_OF_YEAR(expr)
SECOND	SECOND(Order_Time)	Returns the number (between 0 and 59) corresponding to the seconds for a specified time expression.	SECOND(expr)
TIMESTAMPADD	TIMESTAMPADD(SQL_TSI_MONTH, 12, Time."Order Date")	Adds a specified number of intervals to a timestamp, and returns a single timestamp. Interval options are: <i>SQL_TSI_SECOND</i> , <i>SQL_TSI_MINUTE</i> , <i>SQL_TSI_HOUR</i> , <i>SQL_TSI_DAY</i> , <i>SQL_TSI_WEEK</i> , <i>SQL_TSI_MONTH</i> , <i>SQL_TSI_QUARTER</i> , <i>SQL_TSI_YEAR</i>	TIMESTAMPADD(interval, expr, timestamp)
TIMESTAMPDIFF	TIMESTAMPDIFF(SQL_TSI_MONTH, Time."Order Date", CURRENT_DATE)	Returns the total number of specified intervals between two timestamps. Use the same intervals as <i>TIMESTAMPADD</i> .	TIMESTAMPDIFF(interval, expr, timestamp2)
WEEK_OF_QUARTER	WEEK_OF_QUARTER(Order_Date)	Returns a number (between 1 and 13) corresponding to the week of the quarter for the specified date expression.	WEEK_OF_QUARTER(expr)
WEEK_OF_YEAR	WEEK_OF_YEAR(Order_Date)	Returns a number (between 1 and 53) corresponding to the week of the year for the specified date expression.	WEEK_OF_YEAR(expr)
YEAR	YEAR(Order_Date)	Returns the year for the specified date expression.	YEAR(expr)

Date Extraction Functions

These functions calculate or round-down timestamp values to the nearest specified time period, such as hour, day, week, month, and quarter.

You can use the calculated timestamps to aggregate data using a different grain. For example, you might apply the `EXTRACTDAY()` function to sales order dates to calculate a timestamp for midnight on the day that orders occur, so that you can aggregate the data by day.

Function	Example	Description	Syntax
Extract Day	<pre>EXTRACTDAY("Order Date")</pre> <ul style="list-style-type: none"> 2/22/1967 3:02:01 AM returns 2/22/1967 12:00:00 AM. 9/2/2022 10:38:21 AM returns 9/2/2022 12:00:00 AM. 	Returns a timestamp for midnight (12 AM) on the day in which the input value occurs. For example, if the input timestamp is for 3:02:01 AM on February 22nd, the function returns the timestamp for 12:00:00 AM on February 22nd.	<code>EXTRACTDAY(expr)</code>
Extract Hour	<pre>EXTRACTHOUR("Order Date")</pre> <ul style="list-style-type: none"> 2/22/1967 3:02:01 AM returns 2/22/1967 3:00:00 AM. 6/17/1999 11:18:30 PM returns 6/17/1999 11:00:00 PM. 	Returns a timestamp for the start of the hour in which the input value occurs. For example, if the input timestamp is for 11:18:30 PM, the function returns the timestamp for 11:00:00 PM.	<code>EXTRACTHOUR (expr)</code>
Extract Hour of Day	<pre>EXTRACTHOUROFDAY("Order Date")</pre> <ul style="list-style-type: none"> 2014/09/24 10:58:00 returns 2000/01/01 10:00:00. 2014/08/13 11:10:00 returns 2000/01/01 11:00:00 	Returns a timestamp where the hour equals the hour of the input value with default values for year, month, day, minutes, and seconds.	<code>EXTRACTHOUROFDAY (expr)</code>
Extract Millisecond	<pre>EXTRACTMILLISECOND("Order Date")</pre> <ul style="list-style-type: none"> 1997/01/07 15:32:02.150 returns 1997/01/07 15:32:02.150. 1997/01/07 18:42:01.265 returns 1997/01/07 18:42:01.265. 	Returns a timestamp containing milliseconds for the input value. For example, if the input timestamp is for 15:32:02.150, the function returns the timestamp for 15:32:02.150.	<code>EXTRACTMILLISECOND (expr)</code>
Extract Minute	<pre>EXTRACTMINUTE("Order Date")</pre> <ul style="list-style-type: none"> 6/17/1999 11:18:00 PM returns 6/17/1999 11:18:00 PM. 9/2/2022 10:38:21 AM returns 9/2/2022 10:38:00 AM. 	Returns a timestamp for the start of the minute in which the input value occurs. For example, if the input timestamp is for 11:38:21 AM, the function returns the timestamp for 11:38:00 AM.	<code>EXTRACTMINUTE from (expr)</code>
Extract Month	<pre>EXTRACTMONTH("Order Date")</pre> <ul style="list-style-type: none"> 2/22/1967 3:02:01 AM returns 2/1/1967 12:00:00 AM. 6/17/1999 11:18:00 PM returns 6/1/1999 12:00:00 AM. 	Returns a timestamp for the first day in the month in which the input value occurs. For example, if the input timestamp is for February 22nd, the function returns the timestamp for February 1st.	<code>EXTRACTMONTH (expr)</code>

Function	Example	Description	Syntax
Extract Quarter	<p>EXTRACTQUARTER("Order Date")</p> <ul style="list-style-type: none"> 2/22/1967 3:02:01 AM returns 1/1/1967 12:00:00 AM, the first day of the first fiscal quarter. 6/17/1999 11:18:00 PM returns 4/1/1999 12:00:00 AM, the first day of the second fiscal quarter. 9/2/2022 10:38:21 AM returns 7/1/2022 12:00:00 AM, the first day of the third fiscal quarter. <p>Tip: Use QUARTER (expr) to calculate just the ordinal quarter from the returned timestamp.</p>	Returns a timestamp for the first day in the quarter in which the input value occurs. For example, if the input timestamp occurs in the third fiscal quarter, the function returns the timestamp for July 1st.	EXTRACTQUARTER (expr)
Extract Second	<p>EXTRACTSECOND("Order Date")</p> <ul style="list-style-type: none"> 1997/01/07 15:32:02.150 returns 1997/01/07 15:32:02. 1997/01/07 20:44:18.163 returns 1997/01/07 20:44:18. 	Returns a timestamp for the input value. For example, if the input timestamp is for 15:32:02.150, the function returns the timestamp for 15:32:02.	EXTRACTSECOND (expr)
Extract Week	<p>EXTRACTWEEK("Order Date")</p> <ul style="list-style-type: none"> 2014/09/24 10:58:00 returns 2014/09/21. 2014/08/13 11:10:00 returns 2014/08/10. 	Returns the date of the first day of the week (Sunday) in which the input value occurs. For example, if the input timestamp is for Wednesday, September 24th, the function returns the timestamp for Sunday, September 21st.	EXTRACTWEEK (expr)
Extract Year	<p>EXTRACTYEAR("Order Date")</p> <ul style="list-style-type: none"> 1967/02/22 03:02:01 returns 1967/01/01 00:00:00. 1999/06/17 23:18:00 returns 1999/01/01 00:00:00. 	Returns a timestamp for January 1st for the year in which the input value occurs. For example, if the input timestamp occurs in 1967, the function returns the timestamp for January 1st, 1967.	EXTRACTYEAR from (expr)

Conversion Functions

Conversion functions convert a value from one form to another.

Function	Example	Description	Syntax
CAST	CAST(hiredate AS CHAR(40)) FROM employee	Changes the data type of an expression or a null literal to another data type. For example, you can cast a <i>customer_name</i> (a data type of CHAR or VARCHAR) or <i>birthdate</i> (a datetime literal). Use CAST to change to a <i>Date</i> data type. Don't use TODATE.	CAST(expr AS type)
IFNULL	IFNULL(Sales, 0)	Tests if an expression evaluates to a null value, and if it does, assigns the specified value to the expression.	IFNULL(expr, value)
INDEXCOL	SELECT INDEXCOL(VALUEOF(NQ_SESSION.GEOGRAPHY_LEVEL), Country, State, City), Revenue FROM Sales	Uses external information to return the appropriate column for the signed-in user to see.	INDEXCOL([integer literal], [expr1] [, [expr2], ?-])
NULLIF	SELECT e.last_name, NULLIF(e.job_id, j.job_id) "Old Job ID" FROM employees e, job_history j WHERE e.employee_id = j.employee_id ORDER BY last_name, "Old Job ID";	Compares two expressions. If they're equal, then the function returns NULL. If they're not equal, then the function returns the first expression. You can't specify the literal NULL for the first expression.	NULLIF([expression], [expression])
To_DateTime	SELECT To_DateTime('2009-03-03 01:01:00', 'yyyy-mm-dd hh:mi:ss') FROM sales	Converts string literals of <i>DateTime</i> format to a <i>DateTime</i> data type.	To_DateTime([expression], [literal])
VALUEOF	SalesSubjectArea.Customer.Region = VALUEOF("Region Security"."REGION")	References the value of a semantic model variable in a filter. Use <i>expr</i> variables as arguments of the VALUEOF function. Refer to static semantic model variables by name.	VALUEOF(expr)

Display Functions

Display functions operate on the result set of a query.

Function	Example	Description	Syntax
BottomN	BottomN(Sales, 10)	Returns the <i>n</i> lowest values of expression, ranked from lowest to highest.	BottomN([NumericExpression], [integer])
FILTER	FILTER(Sales USING Product = 'widget')	Computes the expression using the given preaggregate filter.	FILTER(measure USING filter_expr)

Function	Example	Description	Syntax
MAVG	MAVG(Sales, 10)	Calculates a moving average (mean) for the last <i>n</i> rows of data in the result set, inclusive of the current row.	MAVG([NumericExpression], [integer])
MSUM	SELECT Month, Revenue, MSUM(Revenue, 3) as 3_MO_SUM FROM Sales	Calculates a moving sum for the last <i>n</i> rows of data, inclusive of the current row. The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data, and so on. When the <i>n</i> th row is reached, the sum is calculated based on the last <i>n</i> rows of data.	MSUM([NumericExpression], [integer])
NTILE	NTILE(Sales, 100)	Determines the rank of a value in terms of a user-specified range. It returns integers to represent any range of ranks. The example shows a range from 1 to 100, with the lowest sale = 1 and the highest sale = 100.	NTILE([NumericExpression], [integer])
PERCENTILE	PERCENTILE(Sales)	Calculates a percent rank for each value satisfying the numeric expression argument. The percentile rank ranges are from 0 (1st percentile) to 1 (100th percentile), inclusive.	PERCENTILE([NumericExpression])
RANK	RANK(Sales)	Calculates the rank for each value satisfying the numeric expression argument. The highest number is assigned a rank of 1, and each successive rank is assigned the next consecutive integer (2, 3, 4,...). If certain values are equal, they're assigned the same rank (for example, 1, 1, 1, 4, 5, 5, 7...).	RANK([NumericExpression])
RCOUNT	SELECT month, profit, RCOUNT(profit) FROM sales WHERE profit > 200	Takes a set of records as input and counts the number of records encountered so far.	RCOUNT([NumericExpression])
RMAX	SELECT month, profit, RMAX(profit) FROM sales	Takes a set of records as input and shows the maximum value based on records encountered so far. The specified data type must be one that can be ordered.	RMAX([NumericExpression])
RMIN	SELECT month, profit, RMIN(profit) FROM sales	Takes a set of records as input and shows the minimum value based on records encountered so far. The specified data type must be one that can be ordered.	RMIN([NumericExpression])
RSUM	SELECT month, revenue, RSUM(revenue) as RUNNING_SUM FROM sales	Calculates a running sum based on records encountered so far. The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data, and so on.	RSUM([NumericExpression])
TOPN	TOPN(Sales, 10)	Returns the <i>n</i> highest values of expression, ranked from highest to lowest.	TOPN([NumericExpression], [integer])

Evaluate Functions

Evaluate functions are database functions that can be used to pass through expressions to get advanced calculations.

Embedded database functions can require one or more columns. These columns are referenced by %1 ... %N within the function. The actual columns must be listed after the function.

Function	Example	Description	Syntax
EVALUATE	SELECT EVALUATE('instr(%1, %2)', address, 'Foster City') FROM employees	Passes the specified database function with optional referenced columns as parameters to the database for evaluation.	EVALUATE([string expression], [comma separated expressions])
EVALUATE_AGGR	EVALUATE_AGGR('R EGR_SLOPE(%1, %2)', sales.quantity, market.marketkey)	Passes the specified database function with optional referenced columns as parameters to the database for evaluation. This function is intended for aggregate functions with a GROUP BY clause.	EVALUATE_AGGR('db_agg _function(%1...%N)' [AS datatype] [, column1, columnN])

Mathematical Functions

The mathematical functions described in this section perform mathematical operations.

Function	Example	Description	Syntax
ABS	ABS(Profit)	Calculates the absolute value of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	ABS(<i>expr</i>)
ACOS	ACOS(1)	Calculates the arc cosine of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	ACOS(<i>expr</i>)
ASIN	ASIN(1)	Calculates the arc sine of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	ASIN(<i>expr</i>)
ATAN	ATAN(1)	Calculates the arc tangent of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	ATAN(<i>expr</i>)
ATAN2	ATAN2(1, 2)	Calculates the arc tangent of y/x , where y is the first numeric expression and x is the second numeric expression.	ATAN2(<i>expr1</i> , <i>expr2</i>)

Function	Example	Description	Syntax
CEILING	CEILING(Profit)	Rounds a non-integer numeric expression to the next highest integer. If the numeric expression evaluates to an integer, the CEILING function returns that integer.	CEILING(expr)
COS	COS(1)	Calculates the cosine of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	COS(expr)
COT	COT(1)	Calculates the cotangent of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	COT(expr)
DEGREES	DEGREES(1)	Converts an expression from radians to degrees. <i>expr</i> is any expression that evaluates to a numerical value.	DEGREES(expr)
EXP	EXP(4)	Sends the value to the power specified. Calculates <i>e</i> raised to the <i>n</i> -th power, where <i>e</i> is the base of the natural logarithm.	EXP(expr)
ExtractBit	Int ExtractBit(1, 5)	Retrieves a bit at a particular position in an integer. It returns an integer of either 0 or 1 corresponding to the position of the bit.	ExtractBit([Source Number], [Digits])
FLOOR	FLOOR(Profit)	Rounds a non-integer numeric expression to the next lowest integer. If the numeric expression evaluates to an integer, the FLOOR function returns that integer.	FLOOR(expr)
LOG	LOG(1)	Calculates the natural logarithm of an expression. <i>expr</i> is any expression that evaluates to a numerical value.	LOG(expr)
LOG10	LOG10(1)	Calculates the base 10 logarithm of an expression. <i>expr</i> is any expression that evaluates to a numerical value.	LOG10(expr)
MOD	MOD(10, 3)	Divides the first numeric expression by the second numeric expression and returns the remainder portion of the quotient.	MOD(expr1, expr2)
PI	PI()	Returns the constant value of pi.	PI()
POWER	POWER(Profit, 2)	Takes the first numeric expression and raises it to the power specified in the second numeric expression.	POWER(expr1, expr2)
RADIANS	RADIANS(30)	Converts an expression from degrees to radians. <i>expr</i> is any expression that evaluates to a numerical value.	RADIANS(expr)
RAND	RAND()	Returns a pseudo-random number between 0 and 1.	RAND()

Function	Example	Description	Syntax
RANDFromSeed	RAND(2)	Returns a pseudo-random number based on a seed value. For a given seed value, the same set of random numbers are generated.	RAND(<i>expr</i>)
ROUND	ROUND(2.166000, 2)	Rounds a numeric expression to <i>n</i> digits of precision. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer that represents the number of digits of precision.	ROUND(<i>expr</i> , <i>integer</i>)
SIGN	SIGN(Profit)	Returns the following: <ul style="list-style-type: none"> • 1 if the numeric expression evaluates to a positive number • -1 if the numeric expression evaluates to a negative number • 0 if the numeric expression evaluates to zero 	SIGN(<i>expr</i>)
SIN	SIN(1)	Calculates the sine of a numeric expression.	SIN(<i>expr</i>)
SQRT	SQRT(7)	Calculates the square root of the numeric expression argument. The numeric expression must evaluate to a nonnegative number.	SQRT(<i>expr</i>)
TAN	TAN(1)	Calculates the tangent of a numeric expression. <i>expr</i> is any expression that evaluates to a numerical value.	TAN(<i>expr</i>)
TRUNCATE	TRUNCATE(45.1234, 2)	Truncates a decimal number to return a specified number of places from the decimal point. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer that represents the number of characters to the right of the decimal place to return.	TRUNCATE(<i>expr</i> , <i>integer</i>)

Running Aggregate Functions

Running aggregate functions perform operations on multiple values to create summary results.

Function	Example	Description	Syntax
MAVG		Calculates a moving average (mean) for the last <i>n</i> rows of data in the result set, inclusive of the current row. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer. Represents the average of the last <i>n</i> rows of data.	MAVG(<i>expr</i> , <i>integer</i>)

Function	Example	Description	Syntax
MSUM	select month, revenue, MSUM(revenue, 3) as 3_MO_SUM from sales_subject_ar ea	Calculates a moving sum for the last n rows of data, inclusive of the current row. <i>expr</i> is any expression that evaluates to a numerical value. <i>integer</i> is any positive integer. Represents the sum of the last n rows of data.	MSUM(<i>expr</i> , <i>integer</i>)
RSUM	SELECT month, revenue, RSUM(revenue) as RUNNING_SUM from sales_subject_ar ea	Calculates a running sum based on records encountered so far. <i>expr</i> is any expression that evaluates to a numerical value.	RSUM(<i>expr</i>)
RCOUNT	select month, profit, RCOUNT(profit) from sales_subject_ar ea where profit > 200	Takes a set of records as input and counts the number of records encountered so far. <i>expr</i> is an expression of any datatype.	RCOUNT(<i>expr</i>)
RMAX	SELECT month, profit,RMAX(prof it) from sales_subject_ar ea	Takes a set of records as input and shows the maximum value based on records encountered so far. <i>expr</i> is an expression of any datatype.	RMAX(<i>expr</i>)
RMIN	select month, profit,RMIN(prof it) from sales_subject_ar ea	Takes a set of records as input and shows the minimum value based on records encountered so far. <i>expr</i> is an expression of any datatype.	RMIN(<i>expr</i>)

Spatial Functions

Spatial functions enable you to perform geographical analysis when you model data. For example, you might calculate the distance between two geographical areas (known as shapes or polygons).



Note:

You can't use these spatial functions in custom calculations for visualization workbooks.

Function	Example	Description	Syntax
GeometryArea	GeometryArea(Shape)	Calculates the area that a shape occupies.	GeometryArea(Shape)
GeometryDistance	GeometryDistance(TRIP_START, TRIP_END)	Calculates the distance between two shapes.	GeometryDistance(Shape 1, Shape 2)

Function	Example	Description	Syntax
GeometryLength	GeometryLength(Shape)	Calculates the circumference of a shape.	GeometryLength(Shape)
GeometryRelate	GeometryRelate(TRIP_START, TRIP_END)	Determines whether one shape is inside another shape. Returns TRUE or FALSE as a string (varchar).	GeometryRelate(Shape 1, Shape 2)
GeometryWithinDistance	GeometryWithinDistance(TRIP_START, TRIP_END, 500)	Determines whether two shapes are within a specified distance of each other. Returns TRUE or FALSE as a string (varchar).	GeometryWithinDistance(Shape1, Shape2, DistanceInFloat)

String Functions

String functions perform various character manipulations. They operate on character strings.

Function	Example	Description	Syntax
ASCII	ASCII('a')	Converts a single character string to its corresponding ASCII code, between 0 and 255. If the character expression evaluates to multiple characters, the ASCII code corresponding to the first character in the expression is returned. <i>expr</i> is any expression that evaluates to a character string.	ASCII(<i>expr</i>)
BIT_LENGTH	BIT_LENGTH('abcdef')	Returns the length, in bits, of a specified string. Each Unicode character is 2 bytes in length (equal to 16 bits). <i>expr</i> is any expression that evaluates to a character string.	BIT_LENGTH(<i>expr</i>)
CHAR	CHAR(35)	Converts a numeric value between 0 and 255 to the character value corresponding to the ASCII code. <i>expr</i> is any expression that evaluates to a numerical value between 0 and 255.	CHAR(<i>expr</i>)
CHAR_LENGTH	CHAR_LENGTH(Customer_Name)	Returns the length, in number of characters, of a specified string. Leading and trailing blanks aren't counted in the length of the string. <i>expr</i> is any expression that evaluates to a character string.	CHAR_LENGTH(<i>expr</i>)
CONCAT	SELECT DISTINCT CONCAT('abc', 'def') FROM employee	Concatenates two character strings. <i>exprs</i> are expressions that evaluate to character strings, separated by commas. You must use raw data, not formatted data, with CONCAT.	CONCAT(<i>expr1</i> , <i>expr2</i>)

Function	Example	Description	Syntax
INSERT	SELECT INSERT('123456', 2, 3, 'abcd') FROM table	<p>Inserts a specified character string into a specified location in another character string. <i>expr1</i> is any expression that evaluates to a character string. Identifies the target character string.</p> <p><i>integer1</i> is any positive integer that represents the number of characters from the beginning of the target string where the second string is to be inserted.</p> <p><i>integer2</i> is any positive integer that represents the number of characters in the target string to be replaced by the second string.</p> <p><i>expr2</i> is any expression that evaluates to a character string. Identifies the character string to be inserted into the target string.</p>	INSERT(<i>expr1</i> , <i>integer1</i> , <i>integer2</i> , <i>expr2</i>)
LEFT	SELECT LEFT('123456', 3) FROM table	<p>Returns a specified number of characters from the left of a string.</p> <p><i>expr</i> is any expression that evaluates to a character string</p> <p><i>integer</i> is any positive integer that represents the number of characters from the left of the string to return.</p>	LEFT(<i>expr</i> , <i>integer</i>)
LENGTH	LENGTH(Customer_ Name)	<p>Returns the length, in number of characters, of a specified string. The length is returned excluding any trailing blank characters.</p> <p><i>expr</i> is any expression that evaluates to a character string.</p>	LENGTH(<i>expr</i>)
LOCATE	LOCATE('d' 'abcdef')	<p>Returns the numeric position of a character string in another character string. If the character string isn't found in the string being searched, the function returns a value of 0.</p> <p><i>expr1</i> is any expression that evaluates to a character string. Identifies the string for which to search.</p> <p><i>expr2</i> is any expression that evaluates to a character string.</p> <p>Identifies the string to be searched.</p>	LOCATE(<i>expr1</i> , <i>expr2</i>)
LOCATEN	LOCATEN('d' 'abcdef', 3)	<p>Like LOCATE, returns the numeric position of a character string in another character string. LOCATEN includes an integer argument that enables you to specify a starting position to begin the search.</p> <p><i>expr1</i> is any expression that evaluates to a character string. Identifies the string for which to search.</p> <p><i>expr2</i> is any expression that evaluates to a character string. Identifies the string to be searched.</p> <p><i>integer</i> is any positive (nonzero) integer that represents the starting position to begin to look for the character string.</p>	LOCATEN(<i>expr1</i> , <i>expr2</i> , <i>integer</i>)

Function	Example	Description	Syntax
LOWER	LOWER(Customer_Name)	Converts a character string to lowercase. <i>expr</i> is any expression that evaluates to a character string.	LOWER(<i>expr</i>)
OCTET_LENGTH	OCTET_LENGTH('abcdef')	Returns the number of bytes of a specified string. <i>expr</i> is any expression that evaluates to a character string.	OCTET_LENGTH(<i>expr</i>)
POSITION	POSITION('d', 'abcdef')	Returns the numeric position of <i>strExpr1</i> in a character expression. If <i>strExpr1</i> isn't found, the function returns 0. <i>expr1</i> is any expression that evaluates to a character string. Identifies the string to search for in the target string. <i>expr2</i> is any expression that evaluates to a character string. Identifies the target string to be searched.	POSITION(<i>expr1</i> IN <i>expr2</i>)
REPEAT	REPEAT('abc', 4)	Repeats a specified expression <i>n</i> times. <i>expr</i> is any expression that evaluates to a character string <i>integer</i> is any positive integer that represents the number of times to repeat the character string.	REPEAT(<i>expr</i> , <i>integer</i>)
REPLACE	REPLACE('abcd1234', '123', 'zz')	Replaces one or more characters from a specified character expression with one or more other characters. <i>expr1</i> is any expression that evaluates to a character string. This is the string in which characters are to be replaced. <i>expr2</i> is any expression that evaluates to a character string. This second string identifies the characters from the first string that are to be replaced. <i>expr3</i> is any expression that evaluates to a character string. This third string specifies the characters to substitute into the first string.	REPLACE(<i>expr1</i> , <i>expr2</i> , <i>expr3</i>)
RIGHT	SELECT RIGHT('123456', 3) FROM table	Returns a specified number of characters from the right of a string. <i>expr</i> is any expression that evaluates to a character string. <i>integer</i> is any positive integer that represents the number of characters from the right of the string to return.	RIGHT(<i>expr</i> , <i>integer</i>)
SPACE	SPACE(2)	Inserts blank spaces. <i>integer</i> is any positive integer that indicates the number of spaces to insert.	SPACE(<i>expr</i>)

Function	Example	Description	Syntax
SUBSTRING	<code>SUBSTRING('abcdef' FROM 2)</code>	Creates a new string starting from a fixed number of characters into the original string. <i>expr</i> is any expression that evaluates to a character string. <i>startPos</i> is any positive integer that represents the number of characters from the start of the left side of the string where the result is to begin.	<code>SUBSTRING([SourceString] FROM [StartPosition])</code>
SUBSTRINGN	<code>SUBSTRING('abcdef' FROM 2 FOR 3)</code>	Like SUBSTRING, creates a new string starting from a fixed number of characters into the original string. <i>SUBSTRINGN</i> includes an integer argument that enables you to specify the length of the new string, in number of characters. <i>expr</i> is any expression that evaluates to a character string. <i>startPos</i> is any positive integer that represents the number of characters from the start of the left side of the string where the result is to begin.	<code>SUBSTRING(expr FROM startPos FOR length)</code>
TrimBoth	<code>Trim(BOTH '_' FROM '_abcdef_')</code>	Strips specified leading and trailing characters from a character string. <i>char</i> is any single character. If you omit this specification (and the required single quotes), a blank character is used as the default. <i>expr</i> is any expression that evaluates to a character string.	<code>TRIM(BOTH char FROM expr)</code>
TRIMLEADING	<code>TRIM(LEADING '_' FROM '_abcdef')</code>	Strips specified leading characters from a character string. <i>char</i> is any single character. If you omit this specification (and the required single quotes), a blank character is used as the default. <i>expr</i> is any expression that evaluates to a character string.	<code>TRIM(LEADING char FROM expr)</code>
TRIMTRAILING	<code>TRIM(TRAILING '_' FROM 'abcdef_')</code>	Strips specified trailing characters from a character string. <i>char</i> is any single character. If you omit this specification (and the required single quotes), a blank character is used as the default. <i>expr</i> is any expression that evaluates to a character string.	<code>TRIM(TRAILING char FROM expr)</code>
UPPER	<code>UPPER(Customer_Name)</code>	Converts a character string to uppercase. <i>expr</i> is any expression that evaluates to a character string.	<code>UPPER(expr)</code>

System Functions

The `USER` system function returns values relating to the session. For example, the user name you signed in with.

Function	Example	Description	Syntax
DATABASE		Returns the name of the subject area to which you're logged on.	DATABASE()
USER		Returns the user name for the semantic model to which you're logged on.	USER()

Time Series Functions

Time series functions are aggregate functions that operate on time dimensions.

Time dimension members must be at or below the level of the function. Because of this, one or more columns that uniquely identify members at or below the given level must be projected in the query.

Function	Example	Description	Syntax
AGO	SELECT Year_ID, AGO(sales, year, 1)	Calculates the aggregated value of a measure from the current time to a specified time period in the past. For example, AGO can produce sales for every month of the current quarter and the corresponding quarter-ago sales.	AGO(expr, time_level, offset)
PERIODROLLING	SELECT Month_ID, PERIODROLLING (monthly_sales, -1, 1)	Computes the aggregate of a measure over the period starting <i>x</i> units of time and ending <i>y</i> units of time from the current time. For example, PERIODROLLING can compute sales for a period that starts at a quarter before and ends at a quarter after the current quarter. <i>measure</i> is the name of a measure column. <i>x</i> is an integer that specifies the offset from the current time. <i>y</i> specifies the number of time units over which the function computes. <i>hierarchy</i> is an optional argument that specifies the name of a hierarchy in a time dimension, such as <i>yr</i> , <i>mon</i> , <i>day</i> , that you want to use to compute the time window.	PERIODROLLING(measure , x [,y])
TODATE	SELECT Year_ID, Month_ID, TODATE (sales, year)	Aggregates a measure from the beginning of a specified time period to the currently displayed time. For example, this function can calculate Year to Date sales. <i>expr</i> is an expression that references at least one measure column. <i>time_level</i> is the type of time period, such as quarter, month, or year.	TODATE(expr, time_level)

FORECAST Function

Creates a time-series model of the specified measure over the series using Exponential Smoothing (ETS) or Seasonal ARIMA or ARIMA. This function outputs a forecast for a set of periods as specified by *numPeriods* argument.

Syntax FORECAST(*numeric_expr*, ([*series*]), *output_column_name*, *options*, [*runtime_binded_options*]))

Where:

- *numeric_expr* indicates the measure to forecast, for example, revenue data to forecast.
- *series* indicates the time grain used to build the forecast model. The series is a list of one or more time dimension columns. If you omit series, then the time grain is determined from the query.
- *output_column_name* indicates the valid column names of *forecast*, *low*, *high*, and *predictionInterval*.
- *options* indicates a string list of name/value pairs separated by a semi-colon (;). The value can include %1 ... %N specified in *runtime_binded_options*.
- *runtime_binded_options* indicates a comma separated list of columns and options. Values for these columns and options are evaluated and resolved during individual query execution time.

FORECAST Function Options The following table list available options to use with the FORECAST function.

Option Name	Values	Description
numPeriods	Integer	The number of periods to forecast
predictionInterval	0 to 100, where higher values specify higher confidence	The confidence level for the prediction.
modelType	ETS SeasonalArima ARIMA	The model to use for forecasting.
useBoxCox	TRUE FALSE	If <i>TRUE</i> , then use Box-Cox transformation.
lambdaValue	Not applicable	The Box-Cox transformation parameter. Ignore if NULL or when <i>useBoxCox</i> is <i>FALSE</i> . Otherwise the data is transformed before the model is estimated.
trendDamp	TRUE FALSE	This is a parameter for ETS model. If <i>TRUE</i> , then use damped trend. If <i>FALSE</i> or NULL, then use non-damped trend.
errorType	Not applicable	This is a parameter for ETS model.

Option Name	Values	Description
trendType	N (none) A (additive) M (multiplicative) Z (automatically selected)	This is a parameter for ETS model.
seasonType	N (none) A (additive) M (multiplicative) Z (automatically selected)	This is a parameter for ETS model.
modelParamIC	ic_auto ic_aicc ic_bic ic_auto (this is the default)	The information criterion (IC) used in the model selection.

Revenue Forecast by Day Example

This example selects revenue forecast by day.

```
FORECAST("A - Sample Sales"."Base Facts"."1- Revenue" Target,
("A - Sample Sales"."Time"."T00 Calendar Date"), 'forecast',
'numPeriods=30;predictionInterval=70;') ForecastedRevenue
```

Revenue Forecast by Year and Quarter Example

This example selects revenue forecast by year and quarter.

```
FORECAST("A - Sample Sales"."Base Facts"."1- Revenue",
("A - Sample Sales"."Time"."T01 Year" timeYear, "A - Sample Sales"."Time"."T02
Quarter" TimeQuarter), 'forecast', 'numPeriods=30;predictionInterval=70;')
ForecastedRevenue
```

Constants

You can use constants to include specific dates and times in expressions.

Available constants include Date, Time, and Timestamp.

Constant	Example	Description	Syntax
DATE	DATE [2014-04-09]	Inserts a specific date.	DATE [yyyy-mm-dd]
TIME	TIME [12:00:00]	Inserts a specific time.	TIME [hh:mi:ss]
TIMESTAMP	TIMESTAMP [2014-04-09 12:00:00]	Inserts a specific timestamp.	TIMESTAMP [yyyy-mm-dd hh:mi:ss]

Types

You can use data types, such as `CHAR`, `INT`, and `NUMERIC` in expressions.

For example, you use types when creating `CAST` expressions that change the data type of an expression or a null literal to another data type.

Variables

Variables are used in expressions.

You can use a variable in an expression.

See [Advanced Techniques: Reference Stored Values in Variables](#).