Oracle® Argus Insight

Extensibility Guide

Release 8.2

E96558-01

August 2018

ORACLE®

Oracle Argus Insight Extensibility Guide, Release 8.2

E96558-01

Contents

Preface

The *Oracle Argus Insight Extensibility Guide* describes the steps to extend Argus Insight 8.2.

Argus Insight extensibility allows you to expand the application's functionality in various areas in order to meet your specific needs.

Where to Find More Information

Oracle Help Center

The latest user documentation for Oracle Health Sciences products is available at http://docs.oracle.com/en/industries/health-sciences/.

My Oracle Support

The latest release notes, patches and white papers are on My Oracle Support (MOS) at https://support.oracle.com. For help with using MOS, see https://docs.oracle.com/cd/E74665_01/MOSHP/toc.htm.

Documentation Accessibility

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

Access to Oracle Support

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

1

Advanced Conditions Extensibility

You can extend the feature of performing the search using the advanced conditions by creating queries on new fields defined with custom configurations.

This chapter covers the steps involved in creating and configuring the new custom fields, and writing custom SQL for both Insight Mart and Argus Mart.

To configure the new fields, you may need to do some configuration in the following tables:

- CMN_FIELDS — See Section 1.1, "Configure CMN_FIELDS Table."
- CMN_FIELD_CONFIGURATION — See Section 1.2, "Configure CMN_FIELD_CONFIGURATION Table."
- CMN_COMPLEXFIELD_CONFIGURATION — See Section 1.3, "Configure CMN_COMPLEXFIELD_CONFIGURATION Table."

To write custom SQL, see Section 1.4, "Write Custom SQL in Advance Condition."

1.1 Configure CMN_FIELDS Table

You can configure the advance condition extensibility columns for the new field ID in the database table CMN_FIELDS as given below:

Table 1–1 CMN _FIELDS Column Details

Column	Sample Value for Insight Mart	Sample Value for Argus Mart	Description
ENTERPRISE_ID	3	3	Defines the current Enterprise ID. This is a mandatory column.
FIELD_ID	30000000	30000000	Defines the Argus Insight new field ID that must be unique and must be in the following range: ▪ For customers: 30000000 - 39999999 ▪ For partners: 40000000 - 49999999 All other IDs are reserved for Oracle. This is a mandatory column.

Table 1–1 (Cont.) CMN _FIELDS Column Details

Column	Sample Value for Insight Mart	Sample Value for Argus Mart	Description
FIELD_LABEL	Custom Product Country	Custom Product Country	Defined the field label having maximum length of 200 characters. This is a mandatory column.
TABLE_NAME	V_RPT_PRODUCT	CASE_PRODUCT	Defines the table name that contains the column for search criteria. The maximum length for the table name is 50 characters. This is a mandatory column.
COLUMN_ NAME	COUNTRY_ID	COUNTRY_ID	Defines the column name for the search criteria. This column name must exist in table populated in TABLE_NAME. The maximum length of this column is 50 characters. This is a mandatory column.
JOIN_FIELD	COUNTRY_ID	COUNTRY_ID	Defines the column name if this field is of drop-down type on UI. This column contains the name of column that you want to use for join condition between the tables populated in TABLE_NAME and SELECT_TABLE. This is an optional column.
SELECT_TABLE	LM_COUNTRIES	LM_COUNTRIES	Defines the table name if this field is of drop-down type on UI. This column contains the name of table that you want to use to populate the drop-down values. This is an optional column.
SELECT_ COLUMN	COUNTRY	COUNTRY	Defines the column name if this field is of drop-down type on UI. This column contains the name of column that you want to use to populate for the drop-down values. This is an optional column.
ADV_COND_ FIELD	1	1	Contains the value for the new field ID as 1. This is a mandatory column.
TREE_VIEW	PRODUCTS:Product Information	PRODUCTS:Product Information	Defines the hierarchical structure of field in advance condition tree on Advance Condition Editor page. The first level and second level node of the tree must be separated by character "**:**". For example, **First Level Tree Node: Second Level Tree Node** This is a mandatory column.

Table 1–1 (Cont.) CMN _FIELDS Column Details

Column	Sample Value for Insight Mart	Sample Value for Argus Mart	Description
SQL_SELECT	SELECT 1 ID, 'UNITED STATE' STATUS FROM DUAL UNION SELECT 2, 'UNITED KINGDOM' FROM DUAL UNION SELECT 3, 'INDIA' FROM DUAL	SELECT 1 ID, 'UNITED STATE' STATUS FROM DUAL UNION SELECT 2, 'UNITED KINGDOM' FROM DUAL UNION SELECT 3, 'INDIA' FROM DUAL	Defines the SQL query if this field is of drop-down type on UI. This column contains the selected query that you want to use to populate the drop-down values. This select query must contain the columns ID and STATUS. **Note:** If this column is configured then the values configured in columns SELECT_COLUMN, SELECT_TABLE and JOIN_FIELD will be ignored. This is an optional column.
HIDDEN	0	0	Contains the value for the new field ID as 0. This is a mandatory column.
CONTROL_ TYPE_ID	2	2	Defines the ID of the control that you want to display on UI. Refer to the table CMN_ CONTROL_TYPE for supported Control Type ID. 1 - Textbox 2 - Dropdown 3 - DatePicker 4 - DateTimePicker 5 - Numeric Control Type This is a mandatory column.
ADDITIONAL_ TABLE_LIST	RPT_EVENT	CASE_EVENT	Defines the comma separated table list that is to be added in From clause of final SQL query except table name entered in the column Table_Name, and: ▪ V_RPT_CASE (in case of Insight Mart) ▪ CASE_MASTER (in case of Argus Mart) This field is required only if any additional join tables are required. This is an optional column.
ADDITIONAL_ WHERE	V_RPT_ PRODUCT.SEQ_ NUM = RPT_ EVENT.SEQ_NUM AND V_RPT_ PRODUCT. COUNTRY_ID > 0	CASE_ PRODUCT.SEQ_ NUM = CASE_ EVENT.SEQ_NUM AND CASE_ PRODUCT. COUNTRY_ID > 0	Defines the additional Where clause that you want to add in the final SQL query of advance condition. This is an optional column.
DATA_ SOURCE_ID	1	2	Defines the value of the target data source (Insight Mart/Argus Mart). This is a mandatory column.

To configure remaining columns of the database table CMN_FIELDS, refer to the DBA Guide.

1.2 Configure CMN_FIELD_CONFIGURATION Table

The Argus Insight supports different field types. The field ID that belongs to one or more field types must be configured in the database table CMN_FIELD_CONFIGURATION.

Note: One field can belong to one or more field types.

Table 1–2 Argus Insight Supported Field Types

Field Type ID	Field Type	Description
1	SMQ_NARROW	The field configured as SMQ_NARROW field type identifies cases that are highly likely to represent the condition of interest. Narrow search consists of all PTs that indicate the condition with great certainty. To configure this field type, see Section 1.2.1, Configure SMQ_NARROW Field Type.
2	SMQ_BROAD	The field configured as SMQ_BROAD field type identifies all possible cases, including some that may prove to be of little or no interest on closer inspection. Those are highly likely to represent the condition of interest. Field as SMQ_BROAD field type search includes both the **narrow** terms and the additional **broad** terms, often of less-specific nature. To configure this field type, see Section 1.2.2, Configure SMQ_BROAD Field Type.
3	MEDDRA	The field configured as MEDDRA field type enables the MedDRA menu to open MedDRA browser. To configure this field type, see Section 1.2.3, Configure MEDDRA Field Type.
4	WHO	The field configured as WHO field type enables the WHO menu to open WHO Drug browser. To configure this field type, see Section 1.2.4, Configure WHO Field Type.
5	COMPANY_DRUG	The field configured as COMPANY_DRUG field type enables the Company Drug menu to open Product browser. To configure this field type, see Section 1.2.5, Configure COMPANY_DRUG Field Type.
6	INGREDIENT	The field configured as INGREDIENT field type enables the Ingredient menu to open Ingredient browser. To configure this field type, see Section 1.2.6, Configure INGREDIENT Field Type.
7	MINUTES_CALCULATOR	The field configured as MINUTES_CALCULATOR field type enables the Minutes Calculator menu to open Minutes Calculator browser. To configure this field type, see Section 1.2.7, Configure MINUTES_CALCULATOR Field Type.

Table 1–2 (Cont.) Argus Insight Supported Field Types

Field Type ID	Field Type	Description
8	LITERATURE	The field configured as LITERATURE field type enables the Literature menu to open Literature browser. To configure this field type, see Section 1.2.8, Configure LITERATURE Field Type.
9	EVENT_LICENSE	The field configured as EVENT_LICENSE field type enables the Event License menu to open Event License browser. To configure this field type, see Section 1.2.9, Configure EVENT_LICENSE Field Type.
10	STUDY_DRUG	The field configured as STUDY_DRUG field type enables the Study Drug menu to open Product browser. To configure this field type, see Section 1.2.10, Configure STUDY_DRUG Field Type.
11	CLINICAL_STUDY_LOOKUP	The field configured as CLINICAL_STUDY_LOOKUP field type enables the Literature menu to open Clinical Study Lookup browser. To configure this field type, see Section 1.2.11, Configure CLINICAL_STUDY_LOOKUP Field Type.
12	BATCH_LOT_NO	The field configured as BATCH_LOT_NO field type enables the Batch Lot # menu to open Batch/Lot Number browser. To configure this field type, see Section 1.2.12, Configure BATCH_LOT_NO Field Type.
13	INVESTIGATIONAL_DRUG	The field configured as INVESTIGATIONAL_DRUG field type enables the Investigational Drug menu to open Investigational Drug browser. To configure this field type, see Section 1.2.13, Configure INVESTIGATIONAL_DRUG Field Type.
14	CO_DRUG_CODE_WITH_STUDY	The field configured as CO_DRUG_CODE_WITH_STUDY field type enables the Co-Drug Code menu to open Batch/Lot Number browser. To configure this field type, see Section 1.2.14, Configure CO_DRUG_CODE_WITH_STUDY Field Type.
15	DVB	The field configured as DVB field type enables the specific range called Duration Value Bands. To configure this field type, see Section 1.2.15, Configure DVB Field Type.
16	GENERIC	The field configured as GENERIC field type enables the Generic Name menu to open Generic Name browser. To configure this field type, see Section 1.2.16, Configure GENERIC Field Type.
17	PATIENT_HISTORY	The field configured as PATIENT_HISTORY field type is considered as information of the patient. To configure this field type, see Section 1.2.17, Configure PATIENT_HISTORY Field Type.

Table 1–2 (Cont.) Argus Insight Supported Field Types

Field Type ID	Field Type	Description
18	PARTIAL_DATE	The field configured as PARTIAL_DATE field type allows the user to enter the partial date. To configure this field type, see Section 1.2.18, Configure PARTIAL_DATE Field Type.
19	CLOB	The field configured as COLB field type is considered as field with data type CLOB of column configured in CMN_FIELDS.COLUMN_NAME. To configure this field type, see Section 1.2.19, Configure CLOB Field Type.
20	PARENT_HISTORY	The field configured as PARENT_HISTORY field type is considered as information about patient's parent. To configure this field type, see Section 1.2.20, Configure PARENT_HISTORY Field Type.

1.2.1 Configure SMQ_NARROW Field Type

You can configure a field id as SMQ_NARROW field type. This field type identifies cases that are highly likely to represent the condition of interest. Narrow search consists of all PTs that indicate the condition with great certainly.

To configure the new field ID as SMQ_NARROW field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–3 Configurations for Field Type SMQ_NARROW

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	1	SMQ_NARROW
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	PT/LLT/ADDITIONALWHERE	This column contains the key as PT, LLT, or ADDITIONALWHERE. One row for each PT (Preffered Term) and LLT (Low Level Term) is mandatory while row with key as ADDITIONALWHERE is optional.

VALUE	RPT_EVENT.ART_CODE/
	RPT_EVENT.INC_CODE/
	RPT_EVENT.ISPRIMARY = 1
		If KEY= PT then
		Add <<Table Name>>.<<Column name>> which contains PT code.
		If KEY= LLT then
		Add <<Table Name>>.<<Column>> name which contains LLT code.
		If KEY= ADDITIONALWHERE then
		If any additional WHERE condition is required.

Note: You can also refer to existing field EVENTS >Primary Event > Event SMQ (Narrow), (CMN_FIELDS.Field_ID - 201760627) of SMQ_ NARROW field type.

1.2.2 Configure SMQ_BROAD Field Type

You can configure a field ID as SMQ_BROAD field type. This field type identifies all possible cases, including some that may prove to be of little or no interest on closer inspection. Those are highly likely to represent the condition of interest. This field type search includes both the **narrow** terms and additional **broad** terms, often of less-specific nature.

To configure the new field ID as SMQ_BROAD field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–4 Configurations for Field Type SMQ_BROAD

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	2	SMQ_BROAD
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	PT/LLT/ADDITIONALWHERE	This column contains the key as PT, LLT or ADDITIONALWHERE. One row for each PT and LLT is mandatory while row with key as ADDITIONALWHERE is optional.

VALUE	RPT_EVENT.ART_CODE/
	RPT_EVENT.INC_COD/
	RPT_EVENT.ISPRIMARY = 1
		If KEY= PT then
		Add <<Table Name>>.<<Column name>> which contains PT code.
		If KEY= LLT then
		Add <<Table Name>>.<<Column>> name which contains LLT code.
		If KEY= ADDITIONALWHERE then
		If any additional WHERE condition is required.

Note: You can also refer to existing field EVENTS > Primary Event > Event SMQ (Broad), (CMN_FIELDS.Field_ID - 201760628) of SMQ_ NARROW field type.

1.2.3 Configure MEDDRA Field Type

You can configure a field id as MedDRA field type. This field type enables you to open the MedDRA browser from menu. With this browser you can search the following:

- HLGT — High Level Group Term
- HLT — High Level Term
- LLT — Low Level Term
- PT — Preferred Term
- SOC — System Organ Class

To configure the new field ID as MEDDRA field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–5 Configurations for Field Type MEDDRA

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	3	MEDDRA
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	SOC_CODE	Enter the Return Type text. See Table 1–6, " Supported Return Type Key/Value for Field Type MEDDRA". This is a mandatory column.
VALUE	1	Enter the Return Type ID. See Table 1–6, " Supported Return Type Key/Value for Field Type MEDDRA". This is a mandatory column.

Note: You can also refer to existing field EVENTS > Primary Event > Event SMQ (Narrow), (CMN_FIELDS.Field_ID - 201760627) of SMQ_NARROW field type.

Supported Return Type Texts and IDs (Key/Value):

The MedDRA browser returns one the following texts as per the return type configured against the new field ID to the Advance Condition Editor page.

Table 1–6 Supported Return Type Key/Value for Field Type MEDDRA

	Return Type ID
SOC_CODE	1
SOC_NAME	2
HLGT_CODE	3
HLGT_NAME	4
HLT_CODE	5
HLT_NAME	6
PT_CODE	7
PT_NAME	8
LLT_CODE	9
LLT_NAME	10

Note: You can also refer to existing field EVENTS > Event Information > Event Body System Code, (CMN_FIELDS.Field_ID - 201450542) of MEDDRA field type.

1.2.4 Configure WHO Field Type

You can configure a field id as WHO field type. This field type enables you to open the WHO browser from menu. This browser searches the product from WHO Drug Dictionary. With this browser you can search for the following:

- ATC Code/Description
- Country
- Formation
- Ingredient
- Medical Prod ID
- Trade Name

To configure the new field ID as WHO field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–7 Configurations for Field Type WHO

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	4	WHO
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	PROD_NAME	Enter the Return Type text. See Table 1–8, " Supported Return Type Key/Value for Field Type WHO". This is a mandatory column.
VALUE	1	Enter the Return Type ID. See Table 1–8, " Supported Return Type Key/Value for Field Type WHO". This is a mandatory column.

Supported Return Type Texts and IDs (Key/Value):

The WHO browser returns one the following texts as per the return type configured against the new field ID to the Advance Condition Editor page.

Table 1–8 Supported Return Type Key/Value for Field Type WHO

Return Type Text	Return Type ID
PROD_NAME	1
ATC_DESC	2
ATC_CODE	3
DRUG_CODE	4
MED_PROD_ID	5

Note: You can also refer to existing field Products > Product Drug/Vaccine > Drug Code, (CMN_FIELDS.Field_ID - 203650840) of WHO field type.

1.2.5 Configure COMPANY_DRUG Field Type

You can configure a field ID as COMPANY_DRUG field type. This field type enables the Company Drug menu to open the Product browser. With this browser you can search the following:

- Ingredient
- Product Family
- Product Name
- Trade Name

To configure the new field ID as COMPANY_DRUG field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–9 Configurations for Field Type COMPANY_DRUG

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	5	COMPANY_DRUG
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	PRODUCT_NAME	Enter the Return Type text. See Table 1–10, " Supported Return Type Key/Value for Field Type COMPANY_DRUG". This is a mandatory column.
VALUE	1	Enter the Return Type ID. See Table 1–10, " Supported Return Type Key/Value for Field Type COMPANY_DRUG". This is a mandatory column.

Supported Return Type Texts and IDs (Key/Value):

The Company Drug browser returns one the following texts as per the return type configured against the new field ID to the Advance Condition Editor page.

Table 1–10 Supported Return Type Key/Value for Field Type COMPANY_DRUG

Return Type Text	Return Type ID
PROD_NAME	1
PRODUCT_ID	2
INGREDIENT_NAME	3
TRADE_NAME	4

Note: You can also refer to existing field Products > Product Information > Company Product, (CMN_FIELDS.Field_ID - 203650960) of COMPANY_DRUG field type.

1.2.6 Configure INGREDIENT Field Type

You can configure a field ID as INGREDIENT field type. This field type enables the Ingredient menu to open Ingredient Browser. With this browser you can search ingredient.

To configure the new field ID as INGREDIENT field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–11 Configurations for Field Type INGREDIENT

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	6	INGREDIENT
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	INGREDIENT	Enter the Return Type text. This is an optional column.
VALUE	1	Enter the Return Type ID as 1. This is a mandatory column.

Note: You can also refer to existing field PRODUCTS > Product Information > Ingredient, (CMN_FIELDS.Field_ID - 203810990) of INGREDIENT field type.

1.2.7 Configure MINUTES_CALCULATOR Field Type

You can configure a field ID as MINUTES_CALCULATOR field type. This field type enables the Minutes Calculator menu to open the Duration Calculator Browser from menu. This browser allows you enter the time in hours, day, weeks, months, or year, and then converts the time to minutes/seconds. Alternatively, you can select duration band and value, if available.

To configure the new field ID as MINUTES_CALCULATOR field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–12 Configurations for Field Type MINUTES_CALCULATOR

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	7	MINUTES_CALCULATOR
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	DVB_SEC	Enter the Return Type text. See Table 1–13, " Supported Return Type Key/Value for Field Type MINUTES_CALCULATOR". This is a mandatory column.
VALUE	1	Enter the Return Type ID. See Table 1–13, " Supported Return Type Key/Value for Field Type MINUTES_CALCULATOR". This is a mandatory column.

Supported Return Type Texts and IDs (Key/Value):

The Duration Calculator browser returns one the following texts as per the return type configured against the new field ID to the Advance Condition Editor page.

Table 1–13 Supported Return Type Key/Value for Field Type MINUTES_CALCULATOR

Return Type Text	Return Type ID
DVB_MIN	0
DVB_SEC	1
NOTDVB_SEC	3

Note: You can also refer to existing field PRODUCTS > Dosage Regimen > Duration of Regimen, (CMN_FIELDS.Field_ID - 201311457) of INGREDIENT field type.

1.2.8 Configure LITERATURE Field Type

You can configure a field ID as LITERATURE field type. This field type enables the Literature menu to open the Literature browser from menu. With this browser you can search literature.

To configure the new field ID as LITERATURE field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–14 Configurations for Field Type LITERATURE

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	8	LITERATURE
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter NULL.
VALUE	1	Enter the Return Type ID as 1. This is a mandatory column.

Note: You can also refer to existing field GENERAL > Literature > Literature, (CMN_FIELDS.Field_ID - 202810741) of INGREDIENT field type.

1.2.9 Configure EVENT_LICENSE Field Type

You can configure a field ID as EVENT_LICENSE field type. This field type enables the Event License menu to open the Event License browser from menu. With this browser you can search events.

To configure the new field ID as EVENT_LICENSE field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–15 Configurations for Field Type EVENT_LICENSE

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID

Table 1–15 (Cont.) Configurations for Field Type EVENT_LICENSE

Column	Sample Value	Description
FIELD_TYPE_ID	9	EVENT_LICENSE
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter NULL.
VALUE	1	Enter the Return Type ID as 1. This is a mandatory column.

Note: You can also refer to existing field EVENTS > Event Assessment > Event Assessment License, (CMN_FIELDS.Field_ID - 201510613) of EVENT_LICENSE field type.

1.2.10 Configure STUDY_DRUG Field Type

You can configure a field ID as STUDY_DRUG field type. This field type enables the Study Drug menu to open the Study Drug Lookup browser from menu. With this browser you can search study drugs.

To configure the new field ID as STUDY_DRUG field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–16 Configurations for Field Type STUDY_DRUG

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	10	STUDY_DRUG
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	PROD_NAME	Enter the Return Type text as PROD_NAME. This is a mandatory column.
VALUE	NOR	Enter the Return Type ID as NOR. This is a mandatory column.

Note: You can also refer to existing field PRODUCTS > Product Information > Study Drug, (CMN_FIELDS.Field_ID - 203650965) of STUDY_DRUG field type.

1.2.11 Configure CLINICAL_STUDY_LOOKUP Field Type

You can configure a field ID as CLINICAL_STUDY_LOOKUP field type. This field type enables the clinical study lookup menu to open the Clinical Study Lookup browser from menu. With this browser you can search study information for clinical studies based on the following:

- Center ID
- Project ID
- Study ID

To configure the new field ID as CLINICAL_STUDY_LOOKUP field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–17 Configurations for Field Type CLINICAL_STUDY_LOOKUP

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	11	CLINICAL_STUDY_LOOKUP
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	CENTERID	Enter the Return Type text. See Table 1–18, " Supported Return Type Key/Value for Field Type CLINICAL_STUDY_LOOKUP". This is a mandatory column.
VALUE	1	Enter the Return Type ID. See Table 1–18, " Supported Return Type Key/Value for Field Type CLINICAL_STUDY_LOOKUP". This is a mandatory column.

Supported Return Type Texts and IDs (Key/Value):

The Clinical Study Lookup browser returns one the following texts as per the return type configured against the new field ID to the Advance Condition Editor page.

Table 1–18 Supported Return Type Key/Value for Field Type CLINICAL_STUDY_LOOKUP

Return Type Text	Return Type ID
CENTERID	1
STUDYID	2
PROJECTID	3

Note: You can also refer to existing field GENERAL > Case Study > Center ID, (CMN_FIELDS.Field_ID - 200650348) of CLINICAL_STUDY_LOOKUP field type.

1.2.12 Configure BATCH_LOT_NO Field Type

You can configure a field ID as BATCH_LOT_NO field type. This field type enables the Batch Lot # menu to open the Batch Lot # Lookup browser from menu. With this browser you can search batch or lot number.

To configure the new field ID as BATCH_LOT_NO field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–19 Configurations for Field Type BATCH_LOT_NO

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	12	BATCH_LOT_NO

Table 1–19 (Cont.) Configurations for Field Type BATCH_LOT_NO

Column	Sample Value	Description
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter Return type text as NULL.
VALUE	NULL	Enter the Return Type ID as NULL.

Note: You can also refer to existing field PRODUCTS > Dosage Regimen > Batch/Lot #, (CMN_FIELDS.Field_ID - 201350479) of BATCH_LOT_NO field type.

1.2.13 Configure INVESTIGATIONAL_DRUG Field Type

You can configure a field ID as INVESTIGATIONAL_DRUG field type. This field type enables the Investigational Drug menu to open the Investigational Drug browser from menu. With this browser you can search and select investigational drug.

To configure the new field ID as INVESTIGATIONAL_DRUG field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–20 Configurations for Field Type INVESTIGATIONAL_DRUG

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	13	INVESTIGATIONAL_DRUG
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter Return type text as NULL.
VALUE	NULL	Enter the Return Type ID as NULL.

Note: You can also refer to existing field PRODUCTS > Product Information > Investigational Drug, (CMN_FIELDS.Field_ID - 203610883) of INVESTIGATIONAL_DRUG field type.

1.2.14 Configure CO_DRUG_CODE_WITH_STUDY Field Type

You can configure a field ID as CO_DRUG_CODE_WITH_STUDY field type. This field type enables the Co-Drug Code w Study menu to open the Co-Drug Code w Study browser from menu. With this browser you can search and select co-drug code with study.

To configure the new field ID as CO_DRUG_CODE_WITH_STUDY field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–21 Configurations for Field Type CO_DRUG_CODE_WITH_STUDY

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	14	CO_DRUG_CODE_WITH_STUDY
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.

Table 1–21 (Cont.) Configurations for Field Type CO_DRUG_CODE_WITH_STUDY

Column	Sample Value	Description
KEY	NULL	Enter key as NULL
VALUE	NULL	Enter the value as NULL

Note: You can also refer to existing field PRODUCTS > Product Drug/Vaccine > Co-Drug Code w Study, (CMN_FIELDS.Field_ID - 203650861) of CO_DRUG_CODE_WITH_STUDY field type.

1.2.15 Configure DVB Field Type

You can configure a field ID as DVB field type. This field type enables the specific range called the Duration Value Bands (DVB). With this field type, you can specify query criteria for the configured field based on ranges instead of specific values.

Note: All the field IDs configured as DVB field type must also be configured as MINUTES_CALCULATOR field type to open the Minutes Calculator browser.

Table 1–22 Configurations for Field Type DVB

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	15	DVB
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	HOURS	Enter the Return Type text. See Table 1–23, " Supported Return Type Key/Value for Field Type DVB". This is a mandatory column.
VALUE	DUR_HR_BAND	Enter the Return Type ID. See Table 1–23, " Supported Return Type Key/Value for Field Type DVB". This is a mandatory column.

Supported Return Type Texts and IDs (Key/Value):

The following keys must be configured for a field ID of field type as DVB. In the data table CMN_FIELD_CONFIGURATION, one row must be configured for each KEY . Value against all the keys should be a database column name. The database column name should exist in data table configured in CMN_FIELD.TABLE_NAME against the field ID. The following are the available keys for configuration:

Table 1–23 Supported Return Type Key/Value for Field Type DVB

Key	Sample Value
HOURS	ONSET_LATENCY_ HRS _BAND
DAYS	ONSET_LATENCY_DAYS_BAND
WEEKS	ONSET_DELAY_ WEEKS _BAND

Table 1–23 (Cont.) Supported Return Type Key/Value for Field Type DVB

Key	Sample Value
MONTHS	ONSET_LATENCY_MONTHS_BAND
YEARS	ONSET_DELAY_YEARS_BAND

Note: You can also refer to existing field EVENTS > Time to Onset from First Dose, (CMN_FIELDS.Field_ID - 201610626) of DVB field type.

1.2.16 Configure GENERIC Field Type

You can configure a field ID as GENERIC field type. This field type enables the Generic Name menu to open the Generic Name browser from menu. With this browser you can search and select generic name of a product.

To configure the new field ID as GENERIC field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–24 Configurations for Field Type GENERIC

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	16	GENERIC
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter key as NUL
VALUE	NULL	Enter value as NULL

Note: You can also refer to existing field PRODUCTS > Product Information > Generic Name, (CMN_FIELDS.Field_ID - 203650842) of GENERIC field type.

1.2.17 Configure PATIENT_HISTORY Field Type

You can configure a field ID as PATIENT_HISTORY field type, if the field is based on information about the patient. This field type adds an additional condition as PARENT = 0 in the WHERE clause of final SQL query for the field.

To configure the new field ID as PATIENT_HISTORY field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–25 Configurations for Field Type PATIENT_HISTORY

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	17	PATIENT_HISTORY
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter key as NULL.
VALUE	NULL	Enter value as NULL.

Note: You can also refer to existing field PATIENT > Patient History > Relevant History Parent Information, (CMN_FIELDS.Field_ID - 203410798) of PATIENT_HISTORY.

1.2.18 Configure PARTIAL_DATE Field Type

You can configure a field ID as PARTIAL_DATE field type. This field type displays the value "**??-??-0000**" in the control on UI. This field type allows the user to enter the partial date. A valid partial date must comprise either a year, or a year and a month.

To configure the new field ID as PARTIAL_DATE field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–26 Configurations for Field Type PARTIAL_DATE

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	18	PARTIAL_DATE
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	START_DATE	Enter Key as column name configured in CMN_FIELDS.COLUMN_NAME. This is a mandatory column.
VALUE	START_DATE_RES	Enter the column name as replacement of columns name configured in CMN_ FIELDS.COLUMN_NAME if partial date is entered by the user. This is a mandatory column.

Note: PATIENT > Parent History > Stop Date is an existing field of PARTIAL_DATE type in CMN_FIELD_CONFIGURATION table.

1.2.19 Configure CLOB Field Type

You can configure a field ID as CLOB field type, if the data type of column configured in CMN_FIELDS.COLUMN_NAME is CLOB. This field type supports the following advanced conditions:

- Begins with
- Contains
- Does not contains
- Missing
- Exists

To configure the new field ID as CLOB field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–27 Configurations for Field Type CLOB

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID

Table 1–27 (Cont.) Configurations for Field Type CLOB

Column	Sample Value	Description
FIELD_TYPE_ID	19	CLOB
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter Key as NULL.
VALUE	NULL	Enter value as NULL.

Note: : You can also refer to existing field ANALYSIS > Case Narrative > Narrative, (CMN_FIELDS.Field_ID - 203050754) of CLOB field type.

1.2.20 Configure PARENT_HISTORY Field Type

You can configure a field ID as PARENT_HISTORY field type, if the field is based on information about the patient's parent. This field type adds an additional condition as PARENT = 1 in the WHERE clause of final SQL query for the field.

To configure the new field ID as PARENT_HISTORY field type, the following configurations are required in the database table CMN_FIELD_CONFIGURATION:

Table 1–28 Configurations for Field Type PARENT_HISTORY

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_TYPE_ID	20	PARENT_HISTORY
FIELD_ID	30000000	Field ID entered in the table CMN_FIELDS.
KEY	NULL	Enter Key as NULL.
VALUE	NULL	Enter value as NULL.

Note: You can also refer to existing field PATIENT > Parent History > Relevant History Parent Information, (CMN_FIELDS.Field_ID - 205050009) of PARENT_HISTORY.

1.3 Configure CMN_COMPLEXFIELD_CONFIGURATION Table

The table CMN_COMPLEXFIELD_CONFIGURATION is used to configure fields that have very complex business logic. Beside, you can also use this table if you want to specify different condition for different operators in WHERE clause. You should define WHERE condition against each operator.

Table 1–29 CMN_COMPLEXFIELD_CONFIGURATION Column Details

Column	Sample Value	Description
ENTERPRISE_ID	3	Current Enterprise ID
FIELD_ID	30000000	New Field ID

Table 1–29 (Cont.) CMN_COMPLEXFIELD_CONFIGURATION Column Details

Column	Sample Value	Description
OPERATOR	contains	Enter the desired operator to support the new Field ID. See Table 1–30, " Supported Operator List" for configuration. This is a mandatory column.
SORT_ORDER	6	Enter the sorting order of operator. This is a mandatory column.
REQ_TABLE_LIST		Add the common separated table list in FROM Clause of final SQL query except V_RPT_CASE, and table name entered in Table_Name Column. This is an optional column.
WHERE_QUERY	(UPPER(V_RPT_PRODUCT.PRODUCT_NAME) NOT LIKE UPPER('%PARAM_VALUE%') AND V_RPT_PRODUCT.pat_exposure > 0)	Define the WHERE clause for the new field ID against the operator entered in Operator Column. This is a mandatory column. **Note:** Use the Place holder <<PARAM_VALUE>> in WHERE clause of SQL query, where selected value is to be placed.

The following are the supported operators for the new field IDs:

Table 1–30 Supported Operator List

Operator	Description
equal to	Retrieves cases where the selected attribute's value is equal to what the Value field specifies.
not equal to	Retrieves cases where the selected attribute's value is not equal to what the Value field specifies.
greater than	Retrieves cases where the selected attribute's value is greater than what the Value field specifies.
greater than or equal to	Retrieves cases where the selected attribute's value is greater than or equal to what the Value field specifies.
less than	Retrieves cases where the selected attribute's value is less than what the Value field specifies.
less than or equal to	Retrieves cases where the selected attribute's value is less than or equal to the Value that the field specifies.
missing	Retrieves cases where the selected attribute's value has not been specified.
exists	Retrieves cases where the selected attribute has any value.
begins with	Retrieves cases where the selected attribute's value begins with what the Value field specifies.
contains	Retrieves cases where the selected attribute's value contains what the Value field specifies.
does not contain	Retrieves cases where the selected attribute's value does not contain what the Value field specifies.
in	Retrieves cases where the selected attribute's value exists in what the Value field specifies.

Table 1–30 (Cont.) Supported Operator List

Operator	Description
not in	Retrieves cases where the selected attribute's value does not exist in what the Value field specifies.

Note: You can also refer to existing field PRODUCTS > Study Drug, (CMN_FIELDS.Field_ID - 203650965).

1.4 Write Custom SQL in Advance Condition

You may write custom SQL for advanced conditions.

- Write Custom SQL for Argus Insight Advance Condition
- Write Custom SQL for Argus Mart Advance Condition

1.4.1 Write Custom SQL for Argus Insight Advance Condition

The following are the steps to create custom SQL for Argus Insight Advanced Condition:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Insight Mart).
3. Add a field, and save the advance condition.
4. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
5. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT V_RPT_CASE.CASE_ID
FROM V_RPT_CASE, <additionaltable(s)>
WHERE <filter clause(s)>
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT V_RPT_CASE.CASE_ID FROM V_RPT_CASE WHERE ((UPPER(V_RPT_
CASE.CASE_NUM)=UPPER('CASE001')))
```

   Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT V_RPT_CASE.CASE_ID FROM V_RPT_CASE, V_RPT_PRODUCT WHERE  (V_
RPT_CASE.CASE_ID = V_RPT_PRODUCT.CASE_ID  AND  ( (UPPER(V_RPT_CASE.CASE_NUM)
=UPPER('CASE001')) AND  (V_RPT_PRODUCT.COUNTRY_ID=223)))
```

Note:

- Make sure the query begins with *SELECT DISTINCT V_RPT_CASE.CASE_ID FROM V_RPT_CASE*.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2 Write Custom SQL for Argus Mart Advance Condition

Argus Insight provides different type of point-in-time queries. You may create custom SQL for any of these point-in-time queries.

The following sections comprise the procedures to create custom query for each type of point-in-time query:

- Current Data Point-in-Time Query
- As of Date Point-in-Time Query
- At Lock Point-in-Time Query
- Last Locked Revision as of a Point in Time Query
- Last Locked Revision for a Version in a Period (Case Receipt Date) Point-in-Time Query
- Last Locked Revision for a Version in a Period (Case Locked Date) Point-in-Time Query
- Last Locked Revision for a Version in a Period (Case Creation Date) Point-in-Time Query
- Aggregate Queries

1.4.2.1 Current Data Point-in-Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Current Data**.
4. Add a field, and save the advance condition.
5. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
6. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER, <additional table(s)>
WHERE <filter clause(s)> AND CASE_MASTER.EFFECTIVE_END_DATE = '31-DEC-9999'
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_
MASTER WHERE ((UPPER(case_master.case_num) =UPPER('CASE100'))) AND CASE_
```

```
MASTER.EFFECTIVE_END_DATE = '31-DEC-9999'
```

Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_
MASTER, (SELECT * FROM CASE_PARENT_INFO WHERE CASE_PARENT_INFO.EFFECTIVE_END_
DATE = '31-DEC-9999') CASE_PARENT_INFO WHERE (CASE_MASTER.CASE_ID = CASE_
PARENT_INFO.CASE_ID  AND  ( (UPPER(case_master.case_num) =UPPER('CASE100')) AND
(case_parent_info.gender_id=1))) AND CASE_MASTER.EFFECTIVE_END_DATE =
'31-DEC-9999'
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All the tables other than CASE_MASTER should be in format **(SELECT * FROM <TABLE_NAME> WHERE <TABLE_NAME>.EFFECTIVE_END_DATE = '31-DEC-9999') <TABLE_NAME>** to execute query as **Current Data**.

  If the table does not have EFFECTIVE_START_DATE column then no inner view is required.
- If you do not include EFFECTIVE_END_DATE = '31-DEC-9999' clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Current Data** type.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.2 As of Date Point-in-Time Query

The following are the steps to create custom SQL for As of Date point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **As of Date**.
4. Add a field, and save the advance condition.
5. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
6. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER, <additional table(s)>
WHERE <filter clause(s)>
AND CASE_MASTER.EFFECTIVE_START_DATE <= To_Date ('<DATE_FOR_AS_OF_
DATE>','DD-MON-YYYY HH24:MI:SS')
AND CASE_MASTER.EFFECTIVE_END_DATE > To_Date (<DATE_FOR_AS_OF_
DATE>,'DD-MON-YYYY HH24:MI:SS')
```

Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_
MASTER
WHERE ((UPPER(case_master.case_num)=UPPER('CASE100')))
AND  CASE_MASTER.EFFECTIVE_START_DATE <= To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS')
AND CASE_MASTER.EFFECTIVE_END_DATE > To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS')
```

Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_
MASTER,
(SELECT * FROM CASE_PARENT_INFO  WHERE  CASE_PARENT_INFO.EFFECTIVE_START_DATE
<= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY HH24:MI:SS')
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS') ) CASE_PARENT_INFO WHERE(CASE_MASTER.CASE_
ID = CASE_PARENT_INFO.CASE_ID  AND ((UPPER(case_master.case_num)
=UPPER('CASE100')) AND  (case_parent_info.gender_id=1)))
AND  CASE_MASTER.EFFECTIVE_START_DATE <= To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS')
AND CASE_MASTER.EFFECTIVE_END_DATE > To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS'
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All the tables other than CASE_MASTER should be in format **(SELECT * FROM <TABLE_NAME> WHERE <TABLE_NAME>.EFFECTIVE_START_DATE <= To_Date ('< DATE_FOR_AS_OF_DATE >','DD-MON-YYYY HH24:MI:SS') AND <TABLE_NAME>.EFFECTIVE_END_DATE > To_Date ('<DATE_FOR_AS_OF_DATE> ','DD-MON-YYYY HH24:MI:SS')) <TABLE_NAME>** to execute query as **As of Date**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all tables, then the query will execute and case series will be generated, but the result may not be of **As of Date** type.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.3 At Lock Point-in-Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).

3. From **Query Type** drop-down list, select **At Lock**.
4. Add a field, and save the advance condition.
5. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
6. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE, DATE_LOCKED FROM CASE_ALL_
LOCKED_REV WHERE  USER_LOCKED = 1  ) X , <additional table(s)>
WHERE <filter clause(s)>
AND  CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
AND X.CASE_ID = CASE_MASTER.CASE_ID
AND X.DATE_LOCKED <= To_Date ('<DATE_FOR_LOCKED_DATE>','DD-MON-YYYY
HH24:MI:SS')
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE, DATE_LOCKED FROM CASE_ALL_
LOCKED_REV WHERE  USER_LOCKED = 1) X
WHERE  ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND  CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
AND X.CASE_ID = CASE_MASTER.CASE_ID
AND X.DATE_LOCKED <= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY HH24:MI:SS')
```

   Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE, DATE_LOCKED FROM CASE_ALL_
LOCKED_REV WHERE  USER_LOCKED = 1) X,
(SELECT CASE_PARENT_INFO.*
FROM CASE_PARENT_INFO, (SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE,
DATE_LOCKED FROM CASE_ALL_LOCKED_REV WHERE  USER_LOCKED = 1) X
WHERE  CASE_PARENT_INFO.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
AND X.CASE_ID = CASE_PARENT_INFO.CASE_ID
AND X.DATE_LOCKED <= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY HH24:MI:SS'))
CASE_PARENT_INFO
WHERE  (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID  AND  ( (UPPER(case_
master.case_num) =UPPER('CASE100')) AND  (case_parent_info.gender_id=1)))
AND  CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
AND X.CASE_ID = CASE_MASTER.CASE_ID
AND X.DATE_LOCKED <= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY HH24:MI:SS')
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE, DATE_LOCKED FROM CASE_ALL_LOCKED_REV WHERE USER_LOCKED = 1) X WHERE <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE AND X.CASE_ID = <TABLE_NAME>.CASE_ID AND X.DATE_LOCKED <= To_Date ('<DATE_FOR_LOCKED_DATE>','DD-MON-YYYY HH24:MI:SS')) <TABLE_NAME>** to execute query as **At Lock**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **At Lock** type.
- Join with (SELECT CASE_ID, VALIDSTART AS EFFECTIVE_START_DATE, DATE_LOCKED FROM CASE_ALL_LOCKED_REV WHERE USER_LOCKED = 1)X is required to get valid revision for table <TABLE_NAME> which is user locked.
- CASE_ALL_LOCKED_REV table contains all locked revisions (user locked as well as post locked).
- CASE_ALL_LOCKED_REV.USER_LOCKED = 1 will give only user locked revisions.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.4 Last Locked Revision as of a Point in Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Last Locked Revision as of a Point in Time**.
4. Add a field, and save the advance condition.
5. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
6. Write the custom SQL as per the format given below:

Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED <= To_Date ('<DATE_FOR_LAST_LOCKEDREVISION>','DD-MON-YYYY
HH24:MI:SS') GROUP BY CASE_ID ) X , <additional table(s)>
WHERE <filter clause(s)>
AND  CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED <= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY
HH24:MI:SS')  GROUP BY CASE_ID) X
WHERE  ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED <= To_Date ('22-DEC-2015 14:12:07','DD-MON-YYYY
HH24:MI:SS')  GROUP BY CASE_ID) X,
(SELECT CASE_PARENT_INFO.*
FROM CASE_PARENT_INFO, (SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE
FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED <= To_Date ('22-DEC-2015
14:12:07','DD-MON-YYYY HH24:MI:SS')  GROUP BY CASE_ID) X
WHERE  CASE_PARENT_INFO.CASE_ID = X.CASE_ID
AND CASE_PARENT_INFO.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE)
CASE_PARENT_INFO
WHERE  (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID  AND  ( (UPPER(case_
master.case_num) =UPPER('CASE100')) AND  (case_parent_info.gender_id=1)))
AND  CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_ MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_ START_DATE FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED <= To_Date ('<DATE_FOR_LAST_LOCKED_ REVISION>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X WHERE <TABLE_NAME>.CASE_ID = X.CASE_ID AND <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_ NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_ DATE) <TABLE_NAME>** to execute query as **Last Locked Revision as of a Point in Time**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Last Locked Revision as of a Point in Time** type.
- Join with (SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED <= To_Date ('<DATE_FOR_LAST_ LOCKED_REVISION>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X is required to get all user locked revisions of cases.
- CASE_ALL_LOCKED_REV table contains all locked revisions (user locked as well as post locked).
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.5 Last Locked Revision for a Version in a Period (Case Receipt Date) Point-in-Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Last Locked Revision for a Version in a Period**.

   The Last Locked Revision for a Version In a Period dialog box appears.
4. Select **Case Receipt Date** option, enter the date range in **From** and **To** fields, and click **Save**.
5. Add a field, and save the advance condition.

6. Click **View SQL**.

   The Advanced Conditions SQL screen appears.

7. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_
DATE>','DD-MON-YYYY HH24:MI:SS')  AND RECEIPT_DATE < To_Date ('<TO_
DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID ) X , <additional table(s)>
WHERE <filter clause(s)>
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID ) X
WHERE  ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID ) X,
(SELECT CASE_PARENT_INFO.* FROM CASE_PARENT_INFO,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS')
AND RECEIPT_DATE < To_Date ('22-DEC-2015 23:59:59','DD-MON-YYYY HH24:MI:SS')
GROUP BY CASE_ID ) X
WHERE  CASE_PARENT_INFO.CASE_ID = X.CASE_ID and CASE_PARENT_INFO.EFFECTIVE_
START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE)
CASE_PARENT_INFO
WHERE  (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID
AND ((UPPER(case_master.case_num) =UPPER('CASE100'))
AND  (case_parent_info.gender_id=1)))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X WHERE <TABLE_NAME>.CASE_ID = X.CASE_ID and <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE) <TABLE_NAME>** to execute query as **Last Locked Revision for a Version in a Period (Case Receipt Date)**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Last Locked Revision for a Version in a Period (Case Receipt Date)** type.
- Join with (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X is required to get all post locked revisions of cases for each Receipt Date.
- ALL_CASES_BY_RECEIPT_DATE table contains Receipt Date and corresponding post lock revision effective start date.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.6 Last Locked Revision for a Version in a Period (Case Locked Date) Point-in-Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Last Locked Revision for a Version in a Period**.

   The Last Locked Revision for a Version In a Period dialog box appears.

4. Select **Case Locked Date** option, enter the date range in **From** and **To** fields, and click **Save**.
5. Add a field, and save the advance condition.
6. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
7. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS')
AND DATE_LOCKED < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_
ID) X, <additional table(s)>
WHERE <filter clause(s)>
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED >= To_Date ('01-JAN-2014 00:00:00','DD-MON-YYYY
HH24:MI:SS') AND DATE_LOCKED < To_Date ('22-DEC-2015 23:59:59','DD-MON-YYYY
HH24:MI:SS') GROUP BY CASE_ID) X
WHERE ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE  FROM CASE_ALL_LOCKED_
REV WHERE DATE_LOCKED >= To_Date ('01-JAN-2014 00:00:00','DD-MON-YYYY
HH24:MI:SS') AND DATE_LOCKED < To_Date ('22-DEC-2015 23:59:59','DD-MON-YYYY
HH24:MI:SS') GROUP BY CASE_ID) X,
(SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE
FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND DATE_LOCKED < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X
WHERE  CASE_PARENT_INFO.CASE_ID = X.CASE_ID and CASE_PARENT_INFO.EFFECTIVE_
START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE)
CASE_PARENT_INFO
WHERE  (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID
AND ((UPPER(case_master.case_num) =UPPER('CASE100'))
AND  (case_parent_info.gender_id=1)))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND DATE_LOCKED < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X WHERE <TABLE_NAME>.CASE_ID = X.CASE_ID and <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE) <TABLE_NAME>** to execute query as **Last Locked Revision for a Version in a Period (Case Locked Date)**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Last Locked Revision for a Version in a Period (Case Locked Date)** type.
- Join with (SELECT CASE_ID, MAX(VALIDSTART) AS EFFECTIVE_START_DATE FROM CASE_ALL_LOCKED_REV WHERE DATE_LOCKED >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND DATE_LOCKED < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X is required to get all user locked revisions of cases.
- CASE_ALL_LOCKED_REV table contains all locked revisions (user locked as well as post locked.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.7 Last Locked Revision for a Version in a Period (Case Creation Date) Point-in-Time Query

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Last Locked Revision for a Version in a Period**.

   The Last Locked Revision for a Version In a Period dialog box appears.

4. Select **Case Creation Date** option, enter the date range in **From** and **To** fields, and click **Save**.
5. Add a field, and save the advance condition.
6. Click **View SQL**.

   The Advanced Conditions SQL screen appears.
7. Write the custom SQL as per the format given below:

   Query Format:

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('<FROM_
DATE>','DD-MON-YYYY HH24:MI:SS') AND CREATE_TIME < To_Date ('<TO_
DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X , <additional table(s)>
WHERE <filter clause(s)>
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE
FROM ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND CREATE_TIME < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X
WHERE  ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

   Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND CREATE_TIME < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS')
AND CREATE_TIME < To_Date ('22-DEC-2015 23:59:59','DD-MON-YYYY HH24:MI:SS')
GROUP BY CASE_ID )X
WHERE  CASE_PARENT_INFO.CASE_ID = X.CASE_ID and CASE_PARENT_INFO.EFFECTIVE_
START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_PARENT_INFO.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE)
CASE_PARENT_INFO
WHERE  (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID
AND ((UPPER(case_master.case_num) =UPPER('CASE100'))
AND  (case_parent_info.gender_id=1)))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND CREATE_TIME < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X WHERE <TABLE_NAME>.CASE_ID = X.CASE_ID and <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE) <TABLE_NAME>** to execute query as **Last Locked Revision for a Version in a Period (Case Creation Date)**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Last Locked Revision for a Version in a Period (Case Creation Date)** type.
- Join with (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE CREATE_TIME >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND CREATE_TIME < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X is required to get all post locked revisions of each cases.
- ALL_CASES_BY_RECEIPT_DATE table contains Create Time and corresponding post lock revision effective start date.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

1.4.2.8 Aggregate Queries

The following are the steps to create custom SQL for Current Data point-in-time query:

1. Login to Argus Insight.
2. Navigate to Queries > Advance Condition > New (Argus Mart).
3. From **Query Type** drop-down list, select **Aggregate Queries**.
4. Add a field, and save the advance condition.
5. Click **View SQL**.

The Advanced Conditions SQL screen appears.

6. Write the custom SQL as per the format given below:

Query Format:

- When unlocked revisions are not required.

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE
FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_
DATE>','DD-MON-YYYY HH24:MI:SS')  AND RECEIPT_DATE < To_Date ('<TO_
DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X, <additional table(s)>
WHERE  <filter clause(s)>
AND  CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

- When unlocked revisions are required.

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, NVL(MAX(LOCKED_EFFECTIVE_START_DATE), MAX(UNLOCKED_
EFFECTIVE_START_DATE)) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_
DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS')
AND RECEIPT_DATE < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS')
GROUP BY CASE_ID )X , <additional table(s)>
WHERE  <filter clause(s)>
AND  CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Example 1: Custom SQL using a single table

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X
WHERE  ((UPPER(case_master.case_num) =UPPER('CASE100')))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Example 2: Custom SQL using two or more tables

```
SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE
FROM CASE_MASTER,
(SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM
ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('01-JAN-2014
00:00:00','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('22-DEC-2015
23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X,
(SELECT CASE_PARENT_INFO.*
FROM CASE_PARENT_INFO, (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS
EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_
Date ('01-JAN-2014 00:00:00','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_
Date ('22-DEC-2015 23:59:59','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID ) X
WHERE  CASE_PARENT_INFO.CASE_ID = X.CASE_ID and CASE_PARENT_INFO.EFFECTIVE_
START_DATE <= X.EFFECTIVE_START_DATE AND CASE_PARENT_INFO.EFFECTIVE_END_DATE >
X.EFFECTIVE_START_DATE)
CASE_PARENT_INFO
```

```
WHERE (CASE_MASTER.CASE_ID = CASE_PARENT_INFO.CASE_ID
AND ((UPPER(case_master.case_num) =UPPER('CASE100'))
AND  (case_parent_info.gender_id=1)))
AND CASE_MASTER.CASE_ID = X.CASE_ID
AND CASE_MASTER.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE
AND CASE_MASTER.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE
```

Note:

- Make sure the query begins with *SELECT DISTINCT CASE_MASTER.CASE_ID,CASE_MASTER.EFFECTIVE_START_DATE FROM CASE_MASTER*.
- All tables other than CASE_MASTER should be in format **(SELECT <TABLE_NAME>.* FROM <TABLE_NAME>, (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X WHERE <TABLE_NAME>.CASE_ID = X.CASE_ID and <TABLE_NAME>.EFFECTIVE_START_DATE <= X.EFFECTIVE_START_DATE AND <TABLE_NAME>.EFFECTIVE_END_DATE > X.EFFECTIVE_START_DATE) <TABLE_NAME>** to execute query as **Aggregate Queries**.

  If the table does not have EFFECTIVE_START_DATE and EFFECTIVE_END_DATE columns then no inner view is required.
- If you do not include EFFECTIVE_START_DATE and EFFECTIVE_END_DATE clause with all the tables, then the query will execute and case series will be generated, but the result may not be of **Aggregate Queries** type.
- Join with (SELECT CASE_ID, MAX(LOCKED_EFFECTIVE_START_DATE) AS EFFECTIVE_START_DATE FROM ALL_CASES_BY_RECEIPT_DATE WHERE RECEIPT_DATE >= To_Date ('<FROM_DATE>','DD-MON-YYYY HH24:MI:SS') AND RECEIPT_DATE < To_Date ('<TO_DATE>','DD-MON-YYYY HH24:MI:SS') GROUP BY CASE_ID) X is required to get all post locked revisions of cases for each Receipt Date.
- ALL_CASES_BY_RECEIPT_DATE table contains Receipt Date and corresponding post lock revision effective start date.
- **NVL(MAX(LOCKED_EFFECTIVE_START_DATE), MAX(UNLOCKED_EFFECTIVE_START_DATE)) AS EFFECTIVE_START_DATE** provides latest unlocked version when there is not locked version available for the selected date range.
- Make sure the query is well formatted and executable without any parameters.
- Do not use "**;**" at the end of the query.
- Do not use comments in the query.

2

Case Series Extensibility

You can extend the feature of merging the case series by customizing new operations or creating new merge options.

Argus Mart, by default, provides the following merge options:

- Current Data
- Latest revision
- All revisions

2.1 Create New Merge Option

The following are the steps to create a new merge option:

1. Connect to Argus Insight Schema with APR_MART (Mart user).
2. Make new entry in **cfg_merge_type_master**, and enter all the following mandatory fields:
   - TYPE_ID = 4 (next available number)
   - Display_Name = '<New Option Name>'
   - Target_Function = Name of function that contains the complete logic of the merge operation for Intersect, Union, and Minus.

     This function must be of public type.

     For example: **F_MERGE_NEW_OPTION**

     Make sure the Target_Function is accessible from the schema AM_APP_OWNER.
   - Enabled = 1
3. Create new **Target_Function** for Merge in the package **pkg_sm_case_series** by using the following template:
   - **Declaration for package specification**

```
FUNCTION F_MERGE_NEW_OPTION (
    pi_merge_seriesid IN NUMBER,
    pi_left_seriesid  IN NUMBER,
    pi_right_seriesid IN NUMBER,
    pi_merge_type     IN NUMBER,
    pi_user_id        IN NUMBER)
    RETURN VARCHAR2 ;
```

   - **Function for package body**

```
FUNCTION F_MERGE_NEW_OPTION (
    pi_merge_seriesid IN NUMBER,
    pi_left_seriesid  IN NUMBER,
    pi_right_seriesid IN NUMBER,
    pi_merge_type     IN NUMBER,
    pi_user_id        IN NUMBER)
    RETURN VARCHAR2 IS
    ln_set_env_var NUMBER;
    PRAGMA AUTONOMOUS_TRANSACTION;
  BEGIN

--------------------------------------------------------------------------
    -- for minus, choose security information from left case series --
    -- study_unblind_ok code broken formula values(for union choose max
value and for intersection choose least value) --
    --                                                --
    -- null    null       20             20    --
    -- 1   1,2,3,4  10+code_broken 10,11,12,13,14  --
    -- 0   1,2,3,5  code_broken 0,1,2,3,4        --
--------------------------------------------------------------------------
    -- SET USER SECURITY
    ln_set_env_var := pkg_sm_data_security.f_set_env_var (pi_user_id);
    IF pi_merge_type = 1 THEN   --UNION
      INSERT INTO case_detail
                  (enterprise_id, seriesid, case_num, case_id, study_
unblind_ok, code_broken, effective_start_date)
        SELECT distinct enterprise_id, pi_merge_seriesid, case_num, case_
id, study_unblind_ok, code_broken, effective_start_date
                   FROM ( <your Selection Logic> );
    ELSIF pi_merge_type = 2 THEN   -- INTERSECT
      INSERT INTO case_detail
                          (enterprise_id,
seriesid, case_num, case_id, study_unblind_ok, code_broken, effective_
start_date
       SELECT DISTINCT enterprise_id, pi_merge_seriesid, case_num, case_id,
study_unblind_ok, code_broken, effective_start_date
                   FROM ( <your Selection Logic> );
    ELSIF pi_merge_type = 3 THEN   --MINUS
      INSERT INTO case_detail
                  (enterprise_id, seriesid, case_num, case_id, study_
unblind_ok, code_broken, effective_start_date)
        SELECT DISTINCT enterprise_id, pi_merge_seriesid seriesid, case_
num, case_id, study_unblind_ok, code_broken, effective_start_date
                   FROM ( <your Selection Logic> );
    END IF;
    COMMIT;
    RETURN 1;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      RETURN SUBSTR (SQLERRM, 1, 3999);
  END F_MERGE_NEW_OPTION;
```

- To create <selection logic> refer to the following existing functions:
  - f_merge_curr_data
  - f_merge_latest_rev
  - f_merge_all_rev

Table 2–1 Template Details

Parameter	Description
pi_merge_seriesid	Defines the output case series ID.
pi_left_seriesid	Defines the left side selected case series.
pi_right_seriesid	Defines the right side selected case series.
pi_merge_type	Defines the flag to contain operation type. ■ 1—Union ■ 2—Intersect ■ 3—Minus
pi_user_id	Defines the User ID of the logged-in application user.

Note:

- Always use v_case_series to fetch complete data from pi_left_seriesid and pi_right_seriesid. For example:

```
Select  enterprise_id, seriesid, case_num, case_id, study_
unblind_ok, code_broken, effective_start_date from v_case_
detail where seriesid = pi_left_seriesid
minus
Select  enterprise_id, seriesid, case_num, case_id, study_
unblind_ok, code_broken, effective_start_date from v_case_
detail where seriesid = pi_right_seriesid
```

- Make sure all the required parameters are available in the Target_Function, and in the same order as that of the template.

3

Code List Extensibility

Flexible Data Recategorization is an Argus Safety functionality through which users can define code list display values in different languages, whether natural human languages like English or artificial ones like E2B.

Argus Safety maintains the data for supported languages and Argus Insight ETL populates this code list data in the corresponding tables as listed below:

Table 3–1 Code List Data Tables

Argus Safety Table	Argus Insight Table
CODE_LIST_MASTER	DM_CODE_LIST_MASTER
CODE_LIST_CODE_ATTRIBUTES	DM_CODE_LIST_CODE_ATTRIBUTES
CODE_LIST_DETAIL_DISCRETE	DM_CODE_LIST_DETAIL_DISCRETE

Customer-specific changes, such as new values for the existing code lists as well as completely new code lists, are made in Argus Safety. These values are then fetched into Argus Insight through the ETL. Users can then create advanced condition queries in Argus Insight that reference the fields in the Flexible Data Recategorization Code List.

The following sections explain how to configure a code list display value in a new language for an already existing code in Argus Safety.

3.1 Configure Flexible Data Recategorization with a New Natural Language

You can configure a code list display value in a new Natural language for an already existing code in Argus Safety.

For example, assume that for the code list GENDER, data in the table CODE_LIST_DETAIL_DISCRETE for code 1 is available in the following three decode contexts (languages):

Figure 3–1 Original Decode Contexts (Languages)

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
GENDER	en	1	Male	0	(null)	05-FEB-13	1
GENDER	E2B	1	1	0	(null)	05-FEB-13	1
GENDER	SM	1	M	0	(null)	05-FEB-13	1

To configure the same code 1 in the code list GENDER for a new language such as GERMAN (decode context '**ge**'):

1. Populate the table CODE_LIST_DETAIL_DISCRETE in Argus Safety with required values in the GERMAN language

```
INSERT INTO CODE_LIST_DETAIL_DISCRETE (CODE_LIST_ID, DECODE_CONTEXT, CODE,
DISPLAY_VALUE, PREFERRED, SORT, LAST_UPDATE_TIME, ENTERPRISE_ID) VALUES
('GENDER', 'ge', 1, 'männlich', 0, null, sysdate, 1);
```

Figure 3–2 New Decode Contexts (Languages)

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
GENDER	en	1	Male	0	(null)	05-FEB-13	1
GENDER	E2B	1	1	0	(null)	05-FEB-13	1
GENDER	SM	1	M	0	(null)	05-FEB-13	1
GENDER	ge	1	männlich	0	(null)	20-FEB-13	1

2. After the Argus Insight ETL runs, to create an Advanced Condition field which displays the GENDER value in the GERMAN language, add a new row in the CMN_FIELDS table in Argus Insight with values similar to the example shown below:

Column	Value
ENTERPRISE_ID	1
FIELD_ID	New field ID that must be unique and must be in the following range: ■ For customers: 30000000 - 39999999 ■ For partners: 40000000 - 49999999 All other IDs are reserved for Oracle.
FIELD_LABEL	Gender German
TABLE_NAME	V_RPT_CASE
COLUMN_NAME	GENDER_ID
JOIN_FIELD	
SELECT_TABLE	
SELECT_COLUMN	
ADV_COND_FIELD	1
TREE_VIEW	PATIENT:Patient Information
UNIQUE_FIELD_LABEL	Gender German
SQL_SELECT	SELECT CODE ID, DISPLAY_VALUE STATUS from DM_CODE_LIST_DETAIL_ DISCRETE WHERE CODE_LIST_ID = GENDER AND DECODE_CONTEXT = 'ge'
FIELD_TYPE	1
HIDDEN	0
TYPE_AHEAD	
BLINDED_FIELD	
CONTROL_TYPE_ID	2
FIELD_LENGTH	255
ADDITIONAL_TABLE_LIST	
ADDITIONAL_WHERE	

3.2 Configure Flexible Data Recategorization with a New Custom Language

You can configure a code list display value in a new Custom language for an already existing code in Argus Safety.

For example, assume that for the code list CAUSALITY, the following data is available in the table CODE_LIST_DETAIL_DISCRETE for '**en**' decode context (English language):

Table 3–2 Original Display Values

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
CAUSALITY	en	1	Definitely Not	0	(null)	9-Jul-13	1
CAUSALITY	en	2	Unlikely	0	(null)	9-Jul-13	1
CAUSALITY	en	3	Possible	0	(null)	9-Jul-13	1
CAUSALITY	en	4	Probable	0	(null)	9-Jul-13	1
CAUSALITY	en	5	Highly Probable	0	(null)	9-Jul-13	1
CAUSALITY	en	6	Definite	0	(null)	9-Jul-13	1

To configure the same code list CAUSALITY for the custom values **Related** and **Unrelated**, which are used as buckets or categories to group the already existing values:

1. Add a new language such as CUSTOM (decode context CUSTOM) by populating the table CODE_LIST_DETAIL_DISCRETE in Argus Safety with required values in the CUSTOM language.

```
INSERT INTO CODE_LIST_DETAIL_DISCRETE (CODE_LIST_ID, DECODE_CONTEXT, CODE,
DISPLAY_VALUE, PREFERRED, SORT, LAST_UPDATE_TIME, ENTERPRISE_ID) VALUES
('CAUSALITY', 'CUSTOM', 1, 'Related', 0, null, sysdate, 1);
```

Table 3–3 New Display Values

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
CAUSALITY	en	1	Definitely Not	0	(null)	9-Jul-13	1
CAUSALITY	en	2	Unlikely	0	(null)	9-Jul-13	1
CAUSALITY	en	3	Possible	0	(null)	9-Jul-13	1
CAUSALITY	en	4	Probable	0	(null)	9-Jul-13	1
CAUSALITY	en	5	Highly Probable	0	(null)	9-Jul-13	1
CAUSALITY	en	6	Definite	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	1	Unrelated	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	2	Unrelated	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	3	Related	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	4	Related	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	5	Related	0	(null)	9-Jul-13	1
CAUSALITY	CUSTOM	6	Related	0	(null)	9-Jul-13	1

2. After the Argus Insight ETL runs, to create an Advanced Condition field which displays custom CAUSALITY values, add a new row in the CMN_FIELDS table in Argus Insight with values similar to the example shown below:

Column	Value
ENTERPRISE_ID	1
FIELD_ID	New field ID that must be unique and must be in the following range: ■ For customers: 30000000 - 39999999 ■ For partners: 40000000 - 49999999 All other IDs are reserved for Oracle.
FIELD_LABEL	Custom Reported Causality
TABLE_NAME	RPT_EVENT_ASSESS
COLUMN_NAME	RPT_CAUSALITY_ID
JOIN_FIELD	
SELECT_TABLE	
SELECT_COLUMN	
ADV_COND_FIELD	1
TREE_VIEW	ANALYSIS:Case Assessment
UNIQUE_FIELD_LABEL	Custom Reported Causality
SQL_SELECT	SELECT DISTINCT DISPLAY_VALUE ID, DISPLAY_VALUE STATUS FROM DM_ CODE_LIST_DETAIL_DISCRETE WHERE CODE_LIST_ID = 'CAUSALITY' AND DECODE_CONTEXT = ' CUSTOM'
FIELD_TYPE	1
HIDDEN	0
TYPE_AHEAD	
BLINDED_FIELD	
CONTROL_TYPE_ID	2
FIELD_LENGTH	255
ADDITIONAL_TABLE_LIST	
ADDITIONAL_WHERE	

3. Now, insert a new row to the table CMN_COMPLEXFIELDS_CONFIGURATION.

Column	Value
ENTERPRISE_ID	3
FIELD_ID	<Same Field ID as in the CMN_FIELDS table>
OPERATOR	equal to
SORT_ORDER	1
REQ_TABLE_LIST	

Column	Value
WHERE_QUERY	RPT_EVENT_ASSESS.RPT_CAUSALITY_ID IN (SELECT CODE FROM DM_CODE_LIST_DETAIL_DISCRETE WHERE DISPLAY_VALUE = 'PARAM_VALUE' AND CODE_LIST_ID = 'CAUSALITY' AND DECODE_CONTEXT = 'CUSTOM')

4

ETL Extensibility

Custom Routines are the configured procedures that are executed during Argus Insight Incremental ETL to perform custom actions.

Argus Insight supports the following custom routines:

- PRE_INCREMENTAL_ETL_TASK - Executes the configured routine during incremental ETL before population of Argus Insight staging tables.
- POST_INCREMENTAL_ETL_TASK - Executes the configured routine during incremental ETL after population of Argus Insight mart tables.

These custom routines are useful in the following scenarios:

- Populating custom tables or new columns based on the business needs.
- Analyzing tables with huge data.
- Triggering an event based on ETL completion for the use with other custom products.

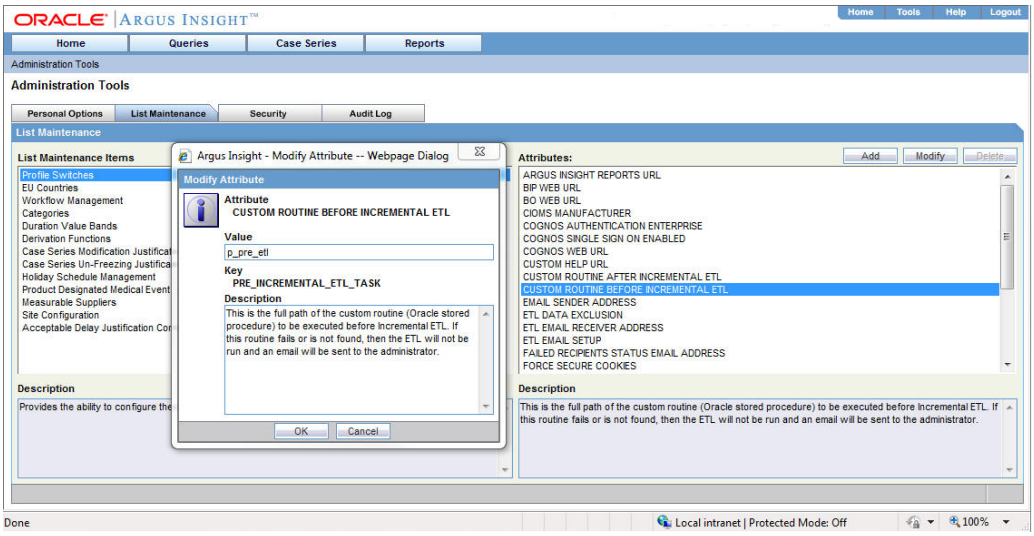
4.1 View Argus Insight Custom Routines

> **Note:** These routines are Global-level switches, visible in Argus Insight Administration Tools.
>
> In a multi-tenant environment, these switches are visible only when you login through Default enterprise.

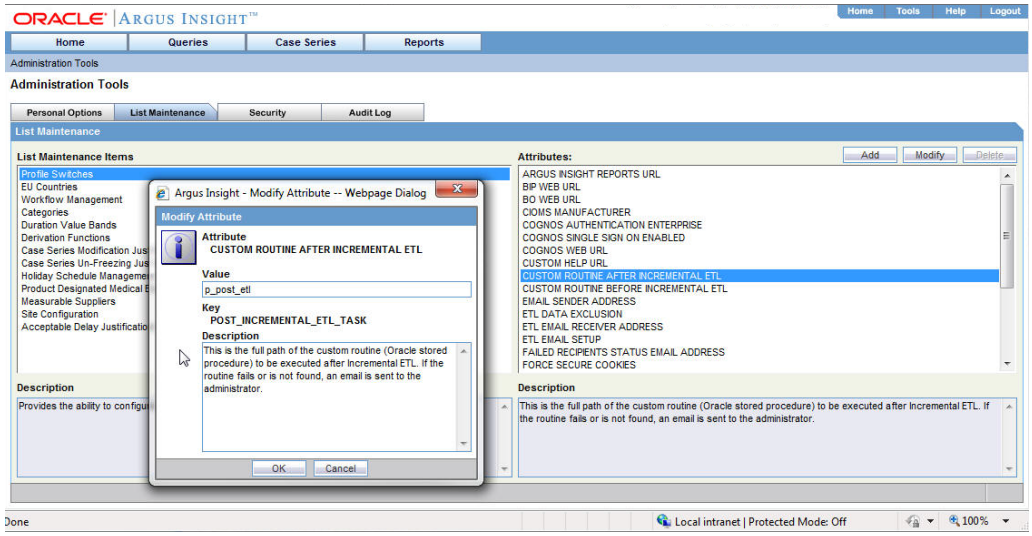
The following are the steps to view the custom routines:

1. Log in to the Argus Insight Application as Admin user.

   Alternatively, in a multi-tenant environment log in to the Default enterprise.
2. From the menu bar, click **Tools**.
3. Click the **List Maintenance** tab to open the List Maintenance Items.
4. Select **Profile Switches** from the List Maintenance Items.
5. From the **Attributes** section, select CUSTOM ROUTINE BEFORE INCREMENTAL ETL, and click **Modify** to see the Value for this routine.



6. Similarly, select CUSTOM ROUTINE AFTER INCREMENTAL ETL, and click **Modify** to see the Value for this routine.



4.2 Execute Argus Insight Custom Routines

The ETL Routines can be executed at two levels:

- Before starting the incremental ETL.
- After executing the incremental ETL.

The following are the steps to execute the custom routine:

1. Select the custom routine (PRE or POST), and enter an Oracle stored procedure name in the **Value** text box relevant to that custom routine. This Routine searches the database object that matches the procedure name in the schema APR_MART during Incremental ETL execution.

> **Note:** To view or modify the Value of a custom routine, refer to the Section 4.1, View Argus Insight Custom Routines.

2. If the procedure is found, the application executes the ETL.
   a. If the custom routine executes without any errors, then the application moves to the next step of the Incremental ETL process.
   b. If the Custom routine executes with errors, then the application logs the error(s) in the table MART_DATA_INSERT_LOG and exits.

Note:

- ETL does not process any data before execution of the pre-incremental ETL custom routine.
- ETL commits the data before the execution of the post-incremental ETL routine.
- You should manually execute the post-incremental ETL routine, if it fails as it cannot be resumed.

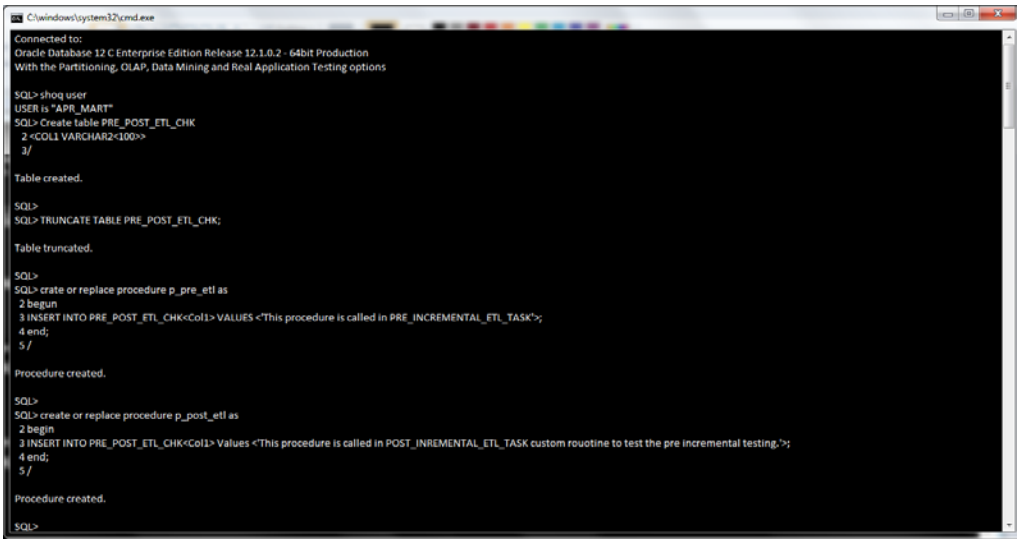
---

3. To track the error when a custom routine fails, refer to the example as explained below:
   a. Connect to the schema APR_MART, and create:

      A table PRE_POST_ETL_CHK with one column col1 varchar2(100).

      A procedure P_PRE_ETL to insert a row in the table with less than 100 characters.

      A procedure P_POST_ETL to insert a row with more than 100 characters.



   b. Update the Values of the custom routines.

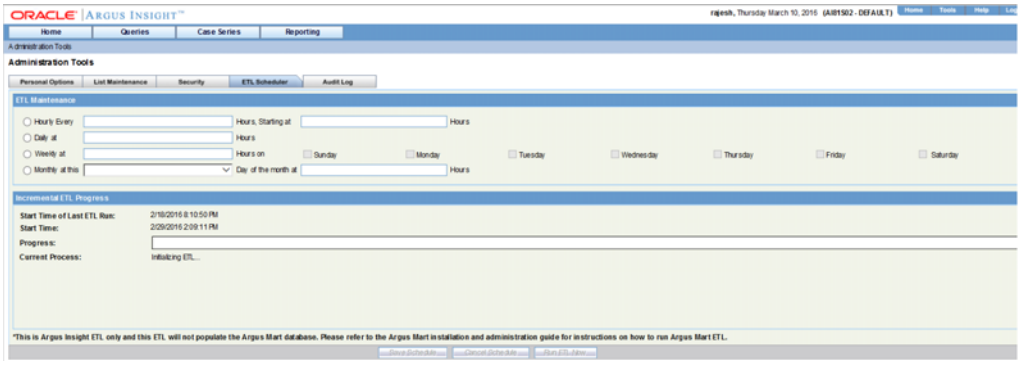
      For key PRE_INCREMENTAL_ETL_TASK, set the value to P_PRE_ETL.

      For key POST_INCREMENTAL_ETL_TASK, set the value to P_POST_ETL.

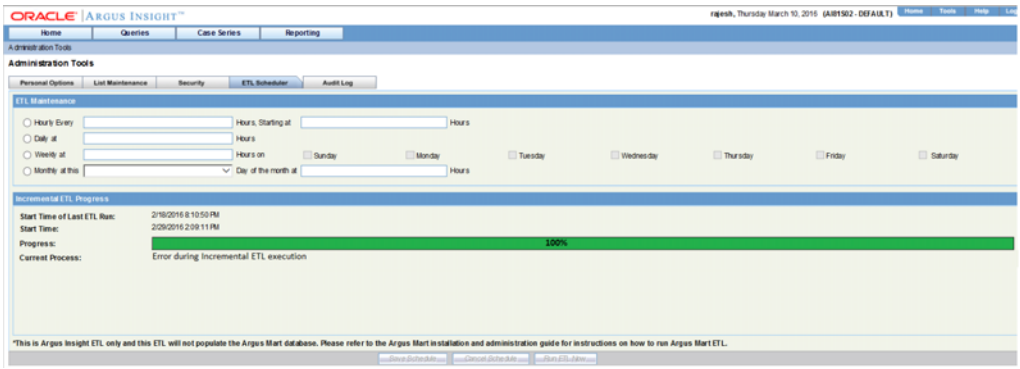
      The ETL will show an error while executing the post-incremental custom procedure as we try to insert large value than the column's length.

Note: To view or modify the Value of a custom routine, refer to the Section 4.1, View Argus Insight Custom Routines.

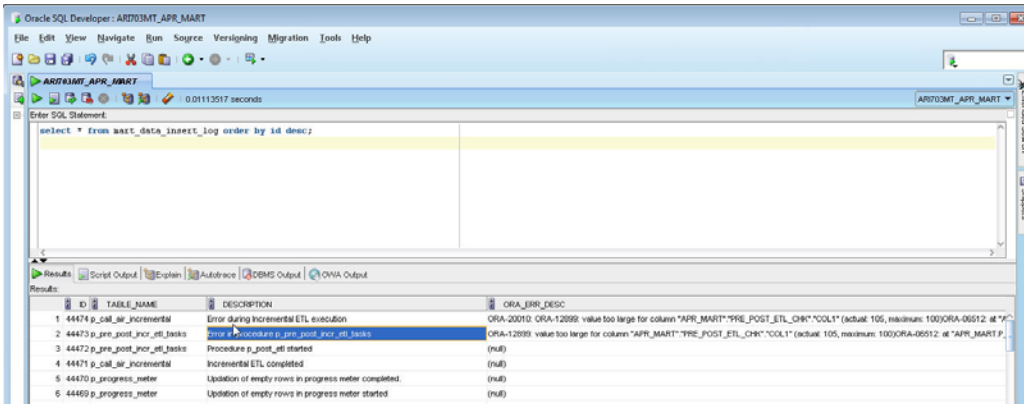
c. Run the incremental ETL.



d. Since P_POST_ETL procedure fails to insert a row, error occurs at the end of the ETL execution.



e. To verify the error, view the table MART_DATA_INSERT_LOG.

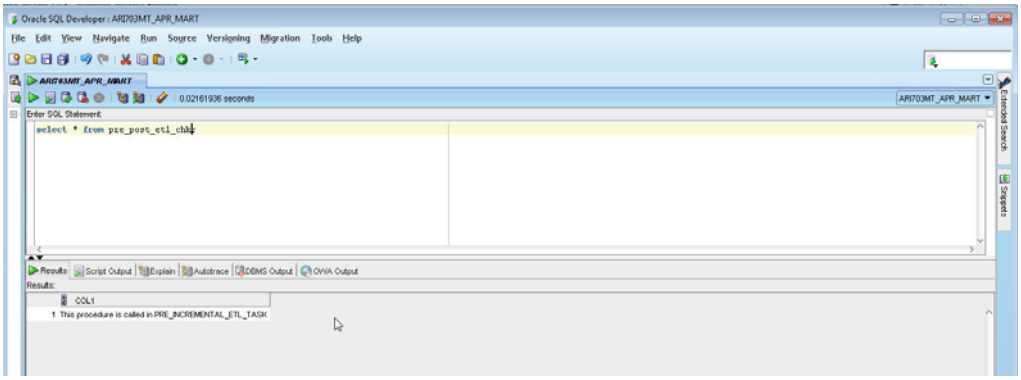


The actual error text that is displayed in the column ORA_ERR_DESC is as below:

```
"ORA-20010: ORA-12899: value too large for column "APR_MART"."PRE_POST_ETL_
CHK"."COL1" (actual: 105, maximum: 100)
ORA-06512: at "APR_MART.P_POST_ETL", line 3
ORA-06512: at line 1
```

```
ORA-06512: at "APR_MART.PKG_PWR_UTIL", line 3306
 -- ERROR while processing p_pre_post_incr_etl_tasks at 25-jun-2013 12:
ORA-06512: at "APR_MART.PKG_AIR_STOM", line 313
ORA-06512: at "APR_MART.PKG_PWR_UTIL", line 3323
ORA-06512: at "APR_MART.PKG_DBMS_JOB", line 1659
```

f. To ensure that a row is inserted from the custom routine before incremental ETL - P_PRE_ETL, view table PRE_POST_ETL_CHK.



4. If the procedure (or database object) is not found, then the application logs the error(s), and fails the ETL.

In this case, the ETL may be executed if you have explicitly created an exception-handling for such cases to absorb any exceptions, and go to the next step in the ETL process.

Optionally, to resolve this issue, create a procedure of that name, provide an existing procedure name, or remove the configuration.

Oracle Recommends:

- You should not modify the existing names of the database objects of Argus Insight, though additional objects can be created as part of customization as per your business needs.
- The changes related to custom routines should be tested in a test environment before implementing in the production environment.

5
Report Extensibility

This chapter provides the information on the following:

- Business Intelligence Publisher (BIP/ BI Publisher) report and the report framework.
- The OBIEE Argus Insight RPD architecture and how to use flex bucketing in the RPD. The querying is done on the data from Argus Safety BIP temporary tables that are brought into Argus Mart tables (information about corresponding report configuration and value of report parameters used for report execution).

5.1 Business Intelligence Publisher Extensibility

Note: The appearance of the user interface that you see in the application may vary from the figures displayed in the subsequent sections.

5.1.1 Assumptions

The Business Intelligence Publisher (BI Publisher) extensibility assumes that the user has a working knowledge of report creation in BI Publisher.

See Also:

Oracle FMW - Administrator Guide for Oracle Business Intelligence Publisher > Configuring the Catalog

5.1.2 Business Purpose

This report is a generic listing of cases with key *Pharmacovigilance* data elements. This framework can be used for custom reporting.

5.1.3 Global Temporary Tables

Global Temporary Tables (GTTs) are the Oracle tables, having data type as *private*; such that data inserted by a session can be accessed by that session only.

The session-specific rows in a GTT can be preserved for the entire session, as AI report tables are created using *ON COMMIT PRESERVE ROWS* clause.

The report specific package *pkg_rep_linelisting*, populates the following report GTTs:

- rep_case_tmp
- rep_event_tmp

- rep_prod_dose_tmp
- rep_evt_assess_tmp
- rep_case_detail_tmp - The Case Detail GTT is populated with user accessible cases in the generic package after applying user data security.

Extend Global Temporary Tables

The following are the steps to extend GTTs:

1. Alter the GTT, to add a new column.
2. Write population logic for the new column in User Exit package. For example, to populate case level table *rep_case_tmp* the following User Exit package - procedure can be used: *pkg_rep_linelisting_user_exit.p_modify_case_tmp*
3. Modify the User Exit package to append case number with ABC, such as:

   PROCEDURE p_modify_case_tmp IS

   BEGIN

   UPDATE REP_CASE_TMP SET CASE_NUM = 'ABC'||CASE_NUM;

   END p_modify_case_tmp;

   Note: Any DML statement or complex PL/SQL logic can be implemented in the User Exit packages.)
4. Compile the User Exit package and run the report.

In the report, you will find case number prefixed with ABC.

5.1.4 Report Package Features

A package is a namespace that organizes a set of related classes and interfaces.

The types of packages used in BI Publisher report are:

- Generic Package
- Line Listing Package

5.1.4.1 Generic Package

BI Publisher report has *pkg_rep_generic* as the generic package that will be used to create/modify all future BI Publisher reports.

This package performs the following functions:

- User Context is set, so that the user can view data only as per user data access rights.
- Global table *rep_case_detail_tmp* is populated with cases after applying data security.
- Log tables population logic is created within the generic package.

This package contains following procedures/functions:

Table 5–1 Generic Package - Procedures and Functions

S.No.	Procedure/Function Name	Parameter/ Argument Used	Description
1.	p_set_user_context	▪ pi_enterprise_id: Enterprise ID ▪ pi_user_name: Report User Name (the user who has logged in to BI Publisher)	This procedure is used to set user context (for multi-tenancy) and data security variables. Using the package *pkg_rls.set_context*, user context will be set, by passing enterprise ID, user name and application name to the package.
2.	p_pop_case_detail	▪ pi_querytype: Q - Query, A-Advance Condition, F - Filter, and C - Case Series ▪ pi_id: CASESERIES_ ID/QUERY_ ID/AC_ ID/Filter_ID to get data for cases	This procedure populates case series in global table *rep_case_detail_tmp*, used in BI Publisher reports. For *p_querytype* = **C**, cases are inserted in global table *rep_case_ detail_tmp.* from the table *case_detail*. For *p_querytype* IN ('**Q**', '**F**', '**A**'), the global table *rep_case_detail_tmp* gets populated in the procedure *p_ caseseries_from_query.*
3.	p_rep_execution_log	▪ pi_ora_err_desc: Oracle-defined error code and description ▪ pi_table_name: Table/Module name ▪ pi_description: User-defined descriptive error message	This procedure is used to log status of table population and SQL exceptions in table *rep_execution_log*. **Routine Call**: PKG_REP_ GENERIC.P_REP_EXECUTION_ LOG (NULL, 'p_pop_case_tmp', 'Data population for table REP_ CASE_TMP started.'); Before populating the table *rep_case_ tmp*, this procedure logs a message that '*data population for table <rep_ case_tmp> started*'. After successful completion of the process, it logs a message that '*data population for table <rep_case_tmp> completed'*. Besides, in each population routine section in the SQL exceptions; this procedure is called to log SQL error messages. **See Also:** Section 5.1.4.2.3, Populate Data for Generic Line Listing Report

Table 5–1 (Cont.) Generic Package - Procedures and Functions

S.No.	Procedure/Function Name	Parameter/ Argument Used	Description
4.	p_rep_sql_log	▪ pi_module_name: identifier to various calling modules ▪ pi_sql_text: Dynamic SQL created	This procedure logs dynamic SQL queries created in the generic package. The following SQL statements are logged in this package: **1.** Insert statements in the table *rep_case_detail_tmp*. **2.** Update *study_unblind_ok*, *code_broken* statement in the table *rep_case_detail_tmp.* **3.** Insert statements in the report log tables. For example: pkg_rep_generic.p_rep_sql_log (pi_module_name, lvc_sql); --lvc_sql Once report is executed, you can copy the query from column *sql_text* of the table *rep_sql_log* where all queries exist. Execute the desired query in the database. **Example Routine Call:** pkg_rep_generic.p_rep_sql_log ('p_caseseries_from_query', lclb_sql); where lclb_sql := 'INSERT INTO rep_case_detail_tmp (case_id) ' || lclb_rpt_sql; Besides, *lclb_rpt_sql* > *sql_for_report* column value from the table *cfg_adv_cond*.
5.	p_keep_report_data	▪ pi_module_name: Calling module name ▪ pi_src_table: Source table name ▪ pi_tgt_table: Target table name	This procedure maintains session data in the report log tables. It is called in the report specific package *pkg_rep_linelisting*. For example: PKG_REP_GENERIC.P_KEEP_REPORT_DATA ('p_pop_case_tmp', 'REP_CASE_TMP', 'REP_CASE_LOG'); In the above example, if the profile switch *KEEP_REPORT_DATA* value is yes, then the table *rep_case_log* will be populated with the session data *rep_case_tmp*. **See Also:** Log Audit Tables, explained later in this chapter

Table 5–1 (Cont.) Generic Package - Procedures and Functions

S.No.	Procedure/Function Name	Parameter/ Argument Used	Description
6.	f_get_insert_sql	■ pi_src_table: Source table name ■ pi_tgt_table: Target table name ■ pi_append_flag: Append hint	This internal function generates dynamic SQL to insert data from the report GTT into the report log tables. It also returns the generated SQL. **Example Routine Call**: pkg_rep_generic.f_get_insert_sql (pi_src_table, pi_tgt_table The data from source table is inserted into the target table.
7.	p_caseseries_from_query	■ pi_ac_id: Query ID to get SQLs for case detail and blinded security ■ pi_querytype: Q - Query, and F - Filter	This procedure inserts cases into the table rep_case_detail_tmp, when the Query/Case parameter is passed a value as Q/F: ■ For Query type - **Q**, the SQL query is fetched from the table *cfg_adv_cond*. ■ For Query type - **F,** the SQL query is fetched from the table *filter_valuesets*. This procedure is called in the procedure *p_pop_case_detail* to populate cases for Query or Filters.
8.	f_get_query_details	■ xdo_user_name: Report User Name (the user who has logged in the BI Publisher) ■ pi_enterprise_id: Enterprise ID ■ pi_querytype: C - Case Series, Q - QBE, A - Advanced Condition, or F- Filter	This function populates the Case Series/Query/Advanced Condition/Filter Name as per the user access rights. The parameter *pi_id* for Case/Query Name prompt, populates with the Case/Query/AC/Filter names based on the selected Enterprise ID. And parameter *pi_querytype* for Case Series/Query prompt, populates as per the logged-in user.

5.1.4.1.1 Context Setting

The context settings for multi tenancy are described in this section.

The procedure *p_set_user_context*, sets enterprise, user name (*username*), and application name (*app_name*) context for Oracle Virtual Private Database policy (VPD).

See Also:

Oracle Technical Reference documents for more information on Oracle VPD.

5.1.4.1.2 Case Series Data Population

The cases in the table *rep_case_detail_tmp* are populated as follows:

- For Case Series/Query Type - **C**: Cases from the table *case_detail* are populated.
- For Case Series/Query Type - **Q** or **A**: Execute the SQL command on the column *sql_for_report* from the table *cfg_adv_cond.*

- For Case Series/Query Type - **F**: Execute the SQL command on the column *sql_for_report* from the table *cfg_adv_cond* and also join another table *filter_valuesets*.

5.1.4.2 Line Listing Package

The BI Publisher report has *pkg_rep_linelisting* as a Generic Line Listing Report specific package.

In this package the report GTTs are populated.

See Also:

Section 5.1.3, Global Temporary Tables

5.1.4.2.1 Generic Parameters

For generic parameters, it is mandatory to declare these parameters in the package that are used in the BI Publisher report. Henceforth, if any new parameter is required to be included in the report then it (new parameter) must be declared in the report specific package.

See Also:

Section 5.1.5.2, Report Parameters for more information about the parameter variables usage in data model.

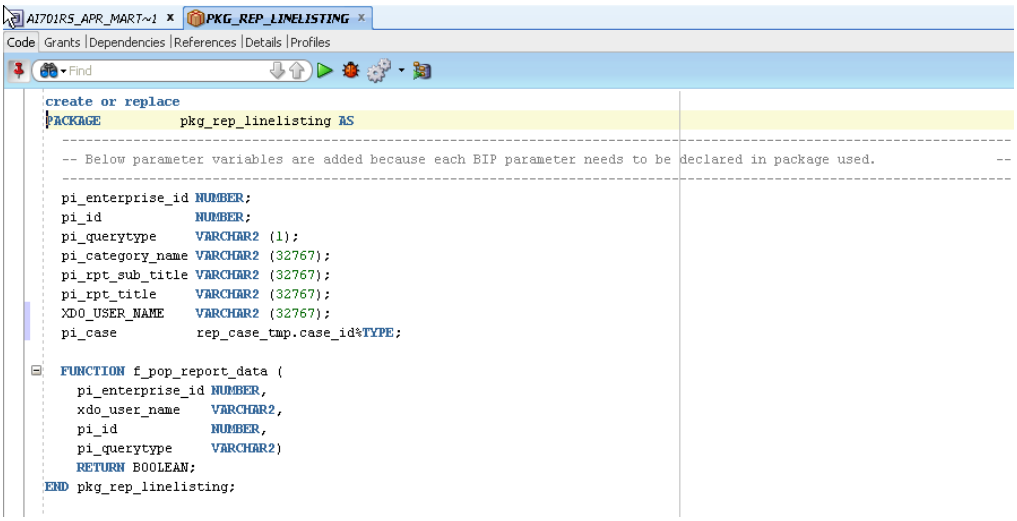
The following report parameters are declared in the report package *pkg_rep_linelisting:*

Table 5–2 Report Parameters

S.No.	Parameter Name	Mandatory/ Optional	Description
1.	pi_enterprise_id: Enterprise ID	Mandatory	A user specific Enterprise ID is passed from BI Publisher to the package, where Enterprise ID is fetched from the table *cfg_user_enterprise_apps.*
2.	pi_querytype: Case Series or Query	Mandatory	A Case Series (C), Query/QBE (Q), Advanced Condition (A) or Filter (F) is passed from BI Publisher based on the user selection.
3.	pi_id: CASESERIES_ID/QUERY_ID/AC_ID/Filter_ID to get data for cases	Mandatory	A user specific case series ID, query ID or filter ID is passed to the package based on the user selection. But in the report, Case series or Query Name is displayed for the enterprise ID and query type selected.
4.	pi_category_name: Category Name	Optional	This is an optional free text parameter, where a user can enter report category name.
5.	pi_rpt_sub_title: Report Sub-heading	Optional	This is an optional free text parameter, where report sub-title is entered.
6.	pi_rpt_title: Report Name	Optional	This is an optional free text parameter, where report name is entered.
7.	xdo_user_name	Optional	A BI Publisher login user name is passed to this parameter. This is BI Publisher system parameter. **See Also:** *BI Publisher Technical Reference* document.

5.1.4.2.2 Add New Parameter in Package

This section is explained with the help of an example. Let us say, you want to add a new parameter *pi_case* and restrict the data model based on the Case ID input. To do so, declare the new parameter in the package as shown below:



See Also:

Section 5.1.5.2, Report Parameters > Add New Parameter in Data Model

5.1.4.2.3 Populate Data for Generic Line Listing Report

The list of routines/functions that are used to populate data for the Generic Line Listing Report is as follows:

Table 5–3 List of Routine/Function used for Generic Line Listing Report Data

S.No.	Routine/Function Name	Parameter Used	Description
1.	f_pop_report_data	pi_enterprise_id, xdo_user_name, pi_id, pi_querytype See Also: Report Parameters Generic Parameters	In this function, the following procedures are called in the same order as listed: 1. To set user context call the procedure as: pkg_rep_generic.p_set_user_context (pi_enterprise_id, xdo_user_name); 2. To populate the cases in GTT *rep_case_detail_tmp* after applying user security, call the routine as: pkg_rep_generic.p_pop_case_detail (pi_id,pi_querytype); 3. *p_pop_case_tmp* - This routine is explained later in the table. 4. *p_pop_event_tmp* - This routine is explained later in the table. 5. *p_pop_prod_dose_tmp* - This routine is explained later in the table. 6. *p_pop_evt_assess_tmp* - This routine is explained later in the table.

Table 5–3 (Cont.) List of Routine/Function used for Generic Line Listing Report Data

S.No.	Routine/Function Name	Parameter Used	Description
2.	p_pop_case_tmp	Not applicable	This Procedure populates data in the GTT *rep_case_tmp*. Before inserting data in the table *rep_case_tmp*, log table *rep_execution_log* is populated with the message as: *PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_case_tmp', 'Data population for table REP_CASE_TMP started.');* **See Also:** Section 5.1.4.2.1, Generic Parameters Once the processing is completed for all the rows in the table *rep_case_tmp*, log the completion details as: *PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_case_tmp', 'Data population for table REP_CASE_TMP completed successfully. ' || SQL%ROWCOUNT || ' row(s) processed.')* **Calling User Exit procedure:** You can write your own logic to update case data in the User Exit procedure *PKG_REP_LINELISTING_USER_EXIT.P_MODIFY_CASE_TMP;* Any exception/errors while populating the table *rep_case_tmp* are handled in WHEN OTHERS exception as: *pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'p_pop_case_tmp', 'Error during data population for table REP_CASE_TMP.')*

Table 5–3 (Cont.) List of Routine/Function used for Generic Line Listing Report Data

S.No.	Routine/Function Name	Parameter Used	Description
3.	p_pop_event_tmp	Not applicable	This procedure populates data in the GTT *rep_event_tmp*. Before inserting data in the table *rep_event_tmp*, log table *rep_execution_log* is populated with the message as: *PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_event_tmp', 'Data population for table REP_EVENT_TMP started.');* **See Also:** Section 5.1.4.2.1, Generic Parameters Once the processing is completed for all the rows in the table *rep_event_tmp*, log the completion details as: *PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_event_tmp', 'Data population for table REP_EVENT_TMP completed successfully. ' || SQL%ROWCOUNT || ' row(s) processed.');* **Calling User Exit procedure:** You can write your own logic to update the event data in the User Exit procedure: *PKG_REP_LINELISTING_USER_EXIT.P_MODIFY_EVENT_TMP;* Any exception/errors while populating the table *rep_event_tmp* are handled in WHEN OTHERS exception as *pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'p_pop_event_tmp', 'Error during data population for table REP_EVENT_TMP.'*)

Table 5–3 (Cont.) List of Routine/Function used for Generic Line Listing Report Data

S.No.	Routine/Function Name	Parameter Used	Description
4.	p_pop_prod_dose_tmp	Not applicable	This procedure populates data in the GTT *rep_prod_dose_tmp*. Before inserting data in the table *rep_prod_dose_tmp*, log table *rep_execution_log* is populated with the message as: *PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_prod_dose_tmp', 'Data population for table REP_PROD_DOSE_TMP started.');* **See Also:** Section 5.1.4.2.1, Generic Parameters Once the processing is completed for all the rows in the table *rep_prod_dose_tmp*, log the completion details as: PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_prod_dose_tmp', 'Data population for table REP_PROD_DOSE_TMP completed successfully. ' || SQL%ROWCOUNT || ' row(s) processed.'); **Calling User Exit procedure:** You can write your own logic to update the product related data in the User Exit procedure: PKG_REP_LINELISTING_USER_EXIT.P_MODIFY_PROD_DOSE_TMP; Any exception/errors while populating the table *rep_prod_dose_tmp* are handled in WHEN OTHERS exception as: pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'p_pop_prod_dose_tmp', 'Error during data population for table REP_PROD_DOSE_TMP.')

Table 5–3 (Cont.) List of Routine/Function used for Generic Line Listing Report Data

S.No.	Routine/Function Name	Parameter Used	Description
5.	p_pop_evt_assess_tmp	Not applicable	This procedure populates data in the GTT *rep_evt_assess_tmp*. Before inserting data in the table *rep_evt_assess_tmp*, log table *rep_execution_log* is populated with the message as: PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_evt_assess_tmp', 'Data population for table REP_EVT_ASSESS_TMP started.'); **See Also:** Section 5.1.4.2.1, Generic Parameters Once the processing is completed for all the rows in the table *rep_evt_assess_tmp*, log the completion details as: PKG_REP_GENERIC.P_REP_EXECUTION_LOG (NULL, 'p_pop_evt_assess_tmp', 'Data population for table REP_EVT_ASSESS_TMP completed successfully. ' || SQL%ROWCOUNT || ' row(s) processed.'); **Calling User Exit procedure:** You can write your own logic to update the event assessment data in the User Exit procedure: PKG_REP_LINELISTING_USER_EXIT.P_MODIFY_EVT_ASSESS_TMP; Any exception/errors while populating the table *rep_evt_assess_tmp* are handled in WHEN OTHERS exception as: pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'p_pop_evt_assess_tmp', 'Error during data population for table REP_EVT_ASSESS_TMP.'); Any error exception in the function *f_pop_report_data*, is handled with message as: pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'f_pop_report_data', 'Error during execution of f_pop_report_data for ENTERPRISE ID - ' || pi_enterprise_id || ', USER NAME - ' || xdo_user_name || '.')

5.1.4.2.4 Log (Audit) Table

The log tables are divided into three categories as follows:

- **Session Details** - There are four report log tables to hold the session data, namely:
  - rep_case_log
  - rep_prod_dose_log
  - rep_event_log
  - rep_evt_assess_log

These tables are populated only if the BI Publisher profile switch **KEEP_REPORT_DATA** is '**Y**' that is, populate the report log tables. By default it is set as '**N**' that is, do not populate the report log tables. This is an enterprise specific switch.

The profile switch are available in the *Argus Insight List Maintenance* section, where you can set it to 'Y' or 'N'.

See Also:

Admin Guide > *<section - TBD>* for the profile switch information.

The procedure *p_keep_report_data*, in generic package is used to populate data for the Report Log tables.

See Also:

Section 5.1.4.1, Generic Package

- **Process Details** - The log table *rep_execution_log*, records the entire report table process details. At each temporary table population procedures the log table will be populated. In all exceptions, this log table is populated with Oracle SQL errors.

  See Also:

  Section 5.1.4.1, Generic Package

- **Dynamic SQL Details** - The log table *rep_sql_log*, is populated with the dynamic SQLs generated in the generic package, only if the database profile switch **LOG_REPORT_SQL** value is '**1**' that is, yes. This is a global switch to identify, if report SQL is to be logged or not. The default value of this switch is '**0**' that is, no.

  This database switch is not available in the Argus Insight UI List maintenance section. It is required to be set in the database only.

  See Also:

  – Section 5.1.4.1, Generic Package

  – Section 5.1.4.2.3, Populate Data for Generic Line Listing Report

5.1.4.2.5 User Exits

A User Exit is a package, which provides a way to pass control from reports specific package to a User Exit package that performs some function (more appropriately data manipulation function), and then return control to main report specific package.

User Exit is used for data manipulations that need extended procedural capabilities.

In section *Populate Data for Generic Line Listing Report*, under each report table population, corresponding User Exit tables are mentioned.

See Also:

- Section 5.1.3, Global Temporary Tables > Extend Global Temporary Tables
- Section 5.1.4.2.3, Populate Data for Generic Line Listing Report

5.1.4.2.6 Lexical Parameters

A Lexical Parameter is a placeholder column containing the actual text to be used in a query. At runtime report query can be modified using lexical parameters.

Modify the Report Package specification to add Lexical Parameters as shown below:

```
create or replace
PACKAGE          pkg_rep_linelisting AS
  ------------------------------------------------------------------------------------------------------------------------
  -- Below parameter variables are added because each BIP parameter needs to be declared in package used. --
  ------------------------------------------------------------------------------------------------------------------------
 pi_enterprise_id NUMBER;
  pi_id            NUMBER;
  pi_querytype     VARCHAR2 (1);
  pi_category_name VARCHAR2 (32767);
  pi_rpt_sub_title VARCHAR2 (32767);
  pi_rpt_title     VARCHAR2 (32767);
  xdo_user_name    VARCHAR2 (32767);
  pi_case          VARCHAR2 (32767);

  --[Lexical parameter Variables]--
  pi_orderby       VARCHAR2 (32767);
  gl_orderby       VARCHAR2 (32767);

  FUNCTION f_pop_report_data (
    pi_enterprise_id NUMBER,
    xdo_user_name    VARCHAR2,
    pi_id            NUMBER,
    pi_querytype     VARCHAR2)
    RETURN BOOLEAN;
END pkg_rep_linelisting;
```

In the above figure, two Lexical Parameters *pi_orderby* and *gl_orderby* are added to the Report Package.

pi_orderby is the parameter in the Data Model based on the value selected in this parameter, the parameter *gl_orderby* will be selected.

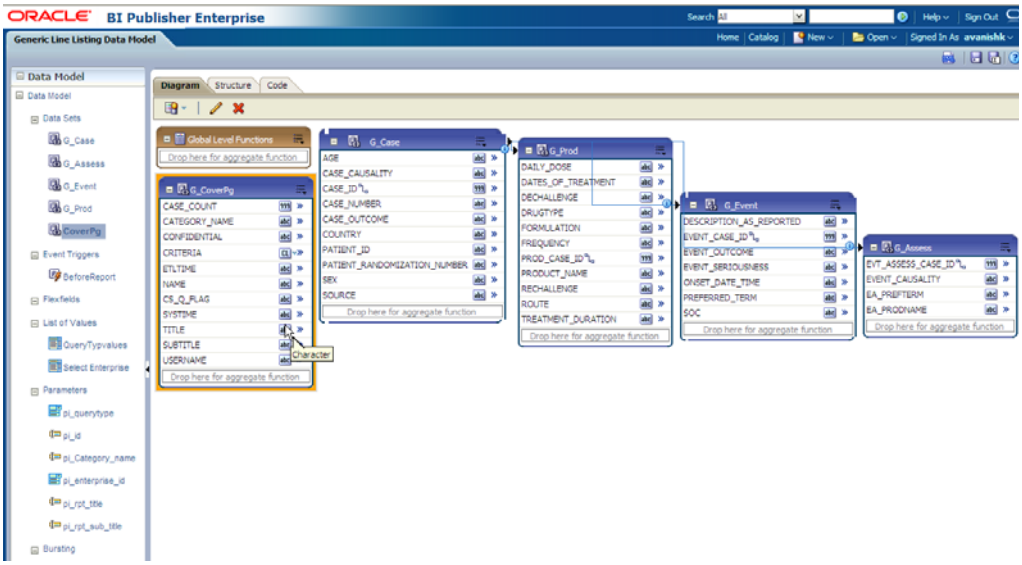
Now, add code in the Report Package body that is, in the function *f_pop_report_data*, the parameter *pi_orderby* is included as shown below:

```
  ------------------------------------------------------------------------------------------------------------------------
  -- FUNCTION : F_POP_REPORT_DATA - function to populate data for Generic Line Listing report. --
  ------------------------------------------------------------------------------------------------------------------------
  -- Returns       : PL/SQL BOOLEAN
  -- Parameter (s) :
  --   1) pi_enterprise_id : Enterprise_ID
  --   2) xdo_user_name    : Report user Name
  --   3) pi_id            : Advanced Condition ID
  --   4) pi_querytype     : Query Type. C = Case Series, Q = Custom Query
  ------------------------------------------------------------------------------------------------------------------------
 FUNCTION f_pop_report_data (
    pi_enterprise_id NUMBER,
    xdo_user_name    VARCHAR2,
    PI_ID            NUMBER,
    PI_QUERYTYPE     VARCHAR2,
    pi_orderby       VARCHAR2)
    RETURN BOOLEAN AS
  BEGIN
    pkg_rep_generic.p_rep_execution_log (NULL, 'f_pop_report_data', 'Data population for ENTERPRISE ID - ' || pi_enterprise_id || ', USER NAME - ' || :
    pkg_rep_generic.p_set_user_context (pi_enterprise_id, xdo_user_name);
    pkg_rep_generic.p_pop_case_detail (pi_id, pi_querytype);
    p_pop_case_tmp;
    p_pop_event_tmp;
    p_pop_prod_dose_tmp;
    p_pop_evt_assess_tmp;

    --[Start Lexical Parameters]--
    IF pi_orderby = '1' THEN
       gl_orderby := ' ORDER BY case_num ';
    ELSIF pi_orderby = '2' THEN
       gl_orderby := ' ORDER BY case_id ';
    ELSE
       GL_ORDERBY := '';
    END IF;
    --[End Lexical Parameters]--

    pkg_rep_generic.p_rep_execution_log (NULL, 'f_pop_report_data', 'Data population for ENTERPRISE ID - ' || pi_enterprise_id || ', USER NAME - ' || :
    RETURN TRUE;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      pkg_rep_generic.p_rep_execution_log (SUBSTR (SQLERRM, 1, 300), 'f_pop_report_data', 'Error during execution of f_pop_report_data for ENTERPRISE :
  END f_pop_report_data;
END pkg_rep_linelisting;
```

Once the package is compiled without any errors, refer to Section 5.1.5.4, Add Lexical Parameter in Data Model, to add the lexical parameters in the BI Publisher.

5.1.5 Data Model

In Argus Insight Generic Line Listing Report, there are five data sets, where *G_Case* is the master data set from which *case_id* column is linked to all other data sets, such as *G_Prod*, *G_Event* and *G_Assess*. So, for each *case_id* all the child data values will be fetched.



Example 5–1 Generate sample XML Data Structure with our Data Model

```
<G_CASE>
<CASE_ID>10031422</CASE_ID>
<CASE_NUMBER>BIPLLREPORT2</CASE_NUMBER>

<G_PROD>
<DAILY_DOSE>3.333 ml</DAILY_DOSE>
<DRUGTYPE>S</DRUGTYPE>
<PROD_CASE_ID>10031422</PROD_CASE_ID>
<PRODUCT_NAME>MMR StudyDB Name Comp</PRODUCT_NAME>
</G_PROD>

<G_EVENT>
<DESCRIPTION_AS_REPORTED>yellow fever</DESCRIPTION_AS_REPORTED>
<EVENT_CASE_ID>10031422</EVENT_CASE_ID>
<PREFERRED_TERM>Yellow fever</PREFERRED_TERM>
<SOC>Infections and infestations</SOC>
</G_EVENT>

<G_EVENT>
<DESCRIPTION_AS_REPORTED>rash</DESCRIPTION_AS_REPORTED>
<EVENT_CASE_ID>10031422</EVENT_CASE_ID>
<PREFERRED_TERM>Rash</PREFERRED_TERM>
<SOC>Skin and subcutaneous tissue disorders</SOC>
</G_EVENT>

<G_ASSESS>
...
</G_ASSESS>
</G_CASE>
```

See Also:

Oracle Fusion Middleware - Report Designer Guide > Chapter 9

5.1.5.1 Data Sets

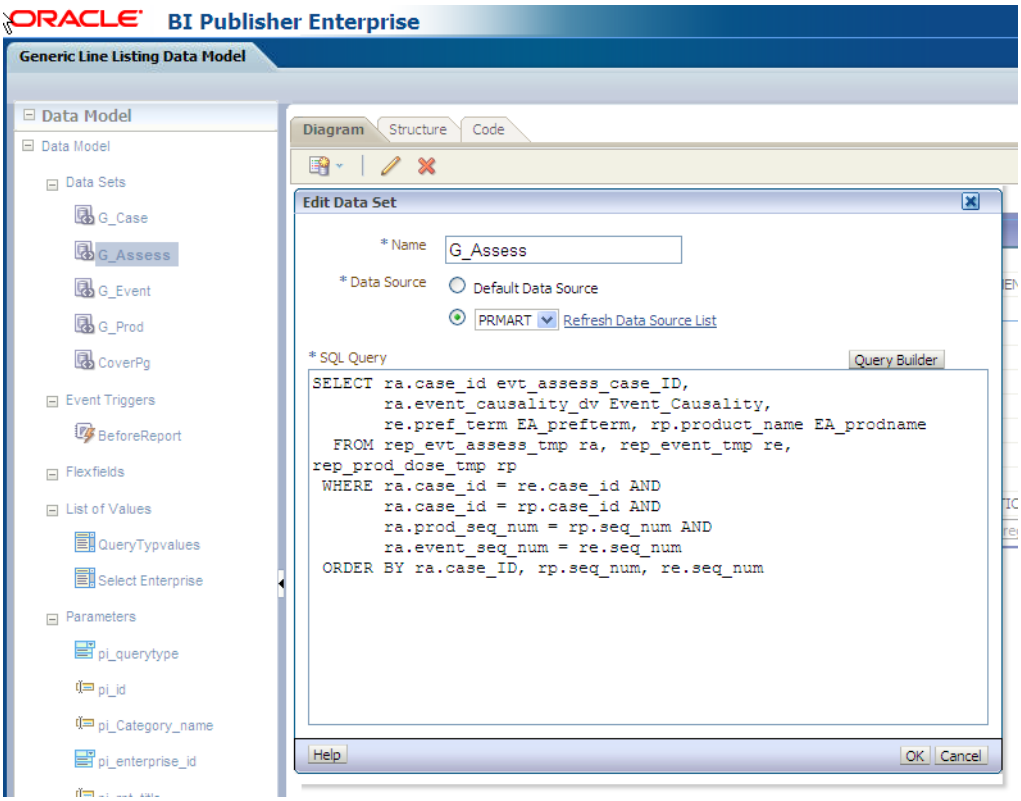
This section contains the information of the following actions:

- Add New Column in Existing Data Set
- Add New Data Set

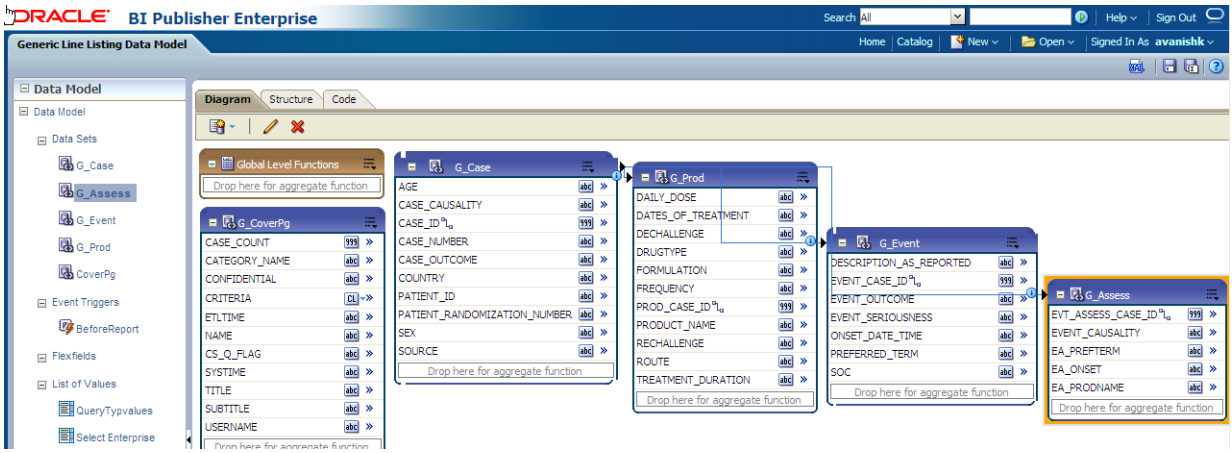
5.1.5.1.1 Add New Column in Existing Data Set

The following are the steps to add a new column in a data set:

1. Click on the data set in which you need to add a column and edit using icons below **Diagram** tab.
2. Let us edit data set *G_Assess*. Click on *G_Assess* and edit the Data Set as shown below:



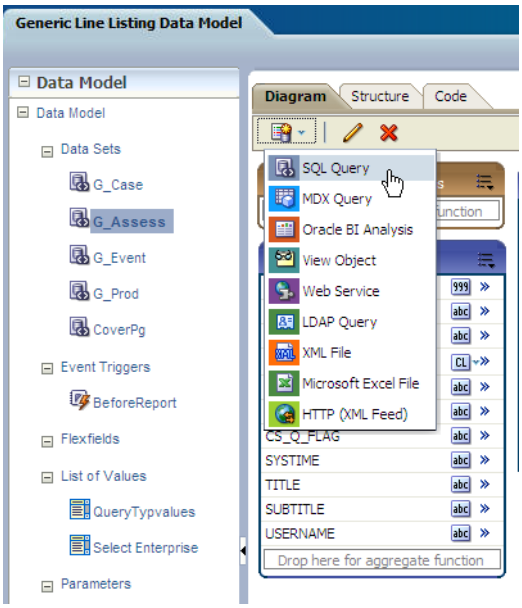
3. In the SQL Query, add any column from the available tables and click **Query Builder**. For example, *re.onset_ve EA_onset*. Once query is built successfully, the column is added to the data set *G_Assess*.



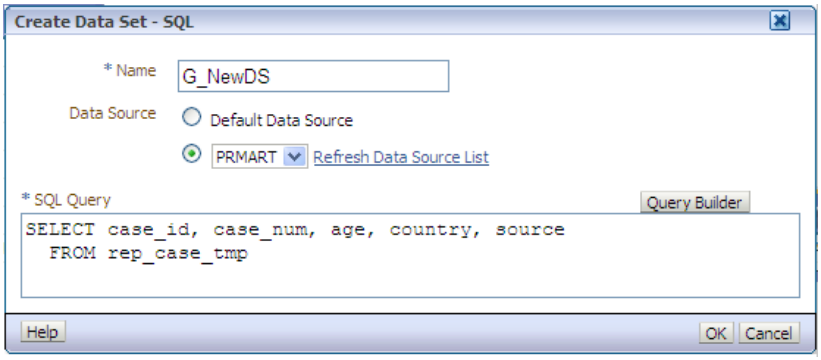
5.1.5.1.2 Add New Data Set

The following are the steps to add a new data set:

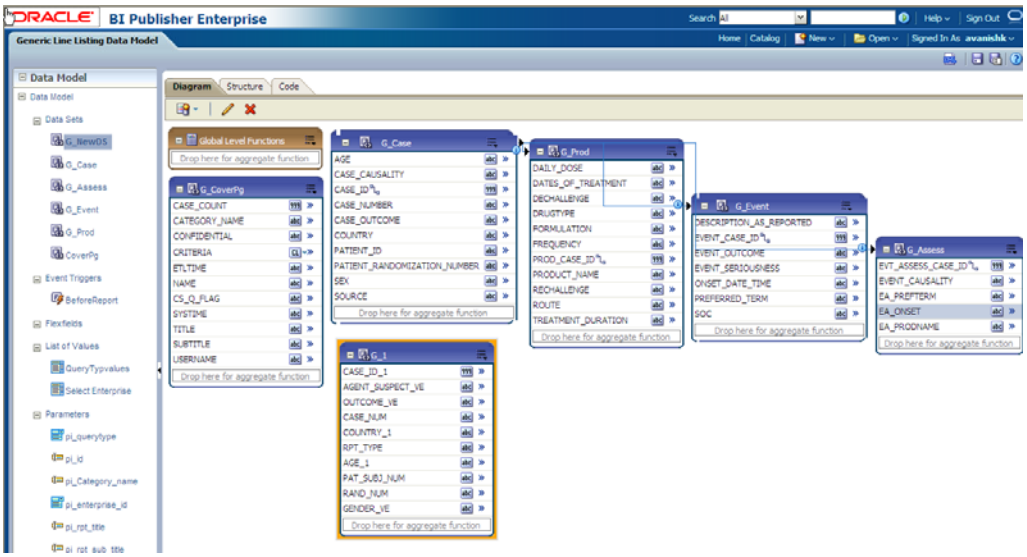
1. Click on **New Data Set** icon and select **SQL Query** as shown below:



2. Write a valid SQL statement to fetch values from the report GTTs. Enter a data set name, such as *G_NewDS* and select proper *Data Source* from the list box. Click **OK**.



3. You can see that new data set *G_NewDS* is created.



4. Save the new Data Model and verify that new data set and columns are available in the data model. Click **Structure** tab to give proper business names for the newly added columns. You can see new data set *G_NEWDS* is available. Modify the business name to *G_MyDS*.

G_Event	G_Event			Event	
DESCRIPTION_AS_REPORTED	DESCRIPTION_AS_REPORTED			Description as Reported	
EVENT_CASE_ID	EVENT_CASE_ID			Event Case ID	
EVENT_OUTCOME	EVENT_OUTCOME			Event Outcome	
EVENT_SERIOUSNESS	EVENT_SERIOUSNESS			Event Seriousness	
ONSET_DATE_TIME	ONSET_DATE_TIME			Onset Date/Time	
PREFERRED_TERM	PREFERRED_TERM			Preferred Term	
SOC	SOC			SOC	
G_Assess	G_Assess			Event_Assessment	
EVT_ASSESS_CASE_ID	EVT_ASSESS_CASE_ID			EA Case ID	
EVENT_CAUSALITY	EVENT_CAUSALITY			Event Causality	
EA_PREFTERM	EA_PREFTERM			Preferred Term	
EA_ONSET	EA_ONSET			EA_ONSET	
EA_PRODNAME	EA_PRODNAME			Product Name	
G_NEWDS	G_MyDS			G_MyDS	
CASE_ID	CASE_ID_1			CASE_ID	
AGENT_SUSPECT_VE	AGENT_SUSPECT_VE			AGENT_SUSPECT_VE	
OUTCOME_VE	OUTCOME_VE			OUTCOME_VE	
CASE_NUM	CASE_NUM			CASE_NUM	
COUNTRY	COUNTRY_1			COUNTRY	
RPT_TYPE	RPT_TYPE			RPT_TYPE	
AGE	AGE_1			AGE	
PAT_SUBJ_NUM	PAT_SUBJ_NUM			PAT_SUBJ_NUM	
RAND_NUM	RAND_NUM			RAND_NUM	
GENDER_VE	GENDER_VE			GENDER_VE	

5.1.5.2 Report Parameters

Report parameters are used to specify the data to use in a report, connect related reports together, and vary report presentation.

The following report parameters are used in BI Publisher:

Note: All the below mentioned parameters, which are used in the report data model must be declared in the report specific package.

If any of the parameters are not declared in the package, those parameters cannot be used in the data model.

Table 5–4 Report Parameters

S.No.	Parameter Name	Label/ Display Name	Parameter Type	Data Type	Description
1.	pi_ enterprise_id	Enterprise ID	Drop-down list	Integer	This prompt lists the Enterprise ID of all the enterprises as per your login credentials (that is, to which logged in user belongs). You are required to select an enterprise for which you want to run the report. For the menu type, parameter list of values object needs to be selected. The List of Value *Select Enterprise* is selected for this parameter. In the list of values any valid SQL query can be provided. In this parameter Enterprise ID is listed.
2.	pi_querytype	Case Series or Query	Fixed drop-down list	String	Generic Line Listing Report can be run on a Case Series, QBE, Advanced Condition or Filter. This is a drop-down (single select) list that allows user to select one of these type on which you want to run the report. The default value selected for this parameter is *Case Series*.
3.	pi_id	Case Series/Query Name	Drop-down list	Integer	An Enterprise ID is passed to get the correct Case Series/QBE/Advanced Condition/Filter names as per the login credentials. Case series, QBE, Advanced Condition or Filter name will be listed based on the Case Series or Query parameter selected by you. You will be allowed to select any one option from the drop-down list. In the report, Case Series or Query name is shown in the drop-down list, but Case Series ID or Query/Filter ID will be passed to the database packages.

Table 5–4 (Cont.) Report Parameters

S.No.	Parameter Name	Label/ Display Name	Parameter Type	Data Type	Description
4.	pi_category_ name	Category Name	User Input	String	This is optional text prompt where you can enter the name of report category (or BI Publisher folder where report is saved). This will be printed in report header box of *Cover Page* section.
5.	pi_rpt_title	Report Name	User Input	String	This is an optional text prompt where you can enter a report title. This will be printed on each page of the report.
6.	pi_rpt_sub_ title	Report Sub-Heading	User Input	String	This is an optional text prompt where you can enter report sub-heading. This will be printed on each page of the report.

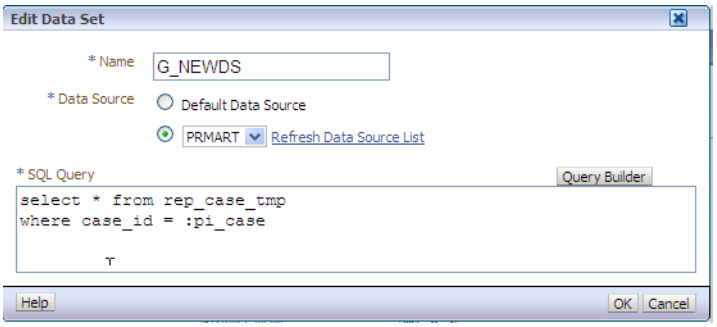
See Also:

Report Mapping Specification Document > 2.1.6. Report Prompts

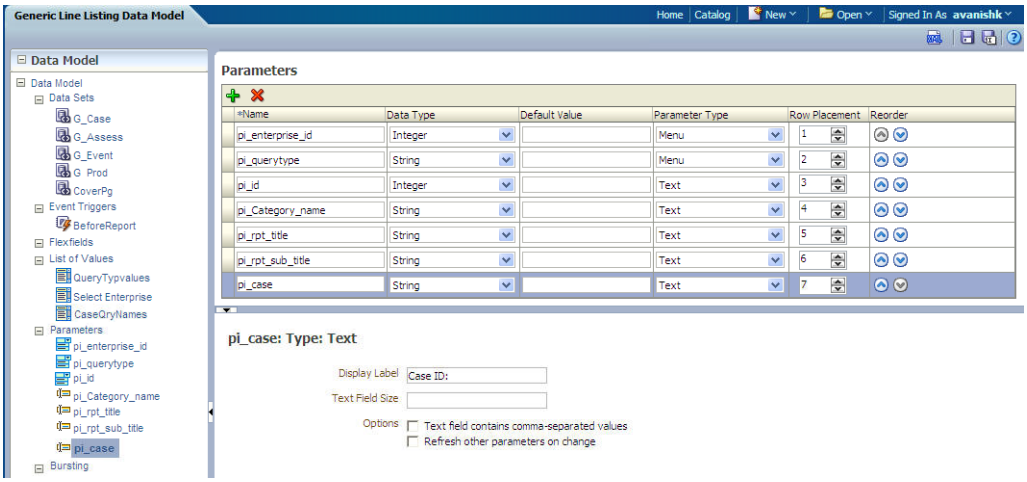
Add New Parameter in Data Model

The following are the steps to add new parameter in the data model:

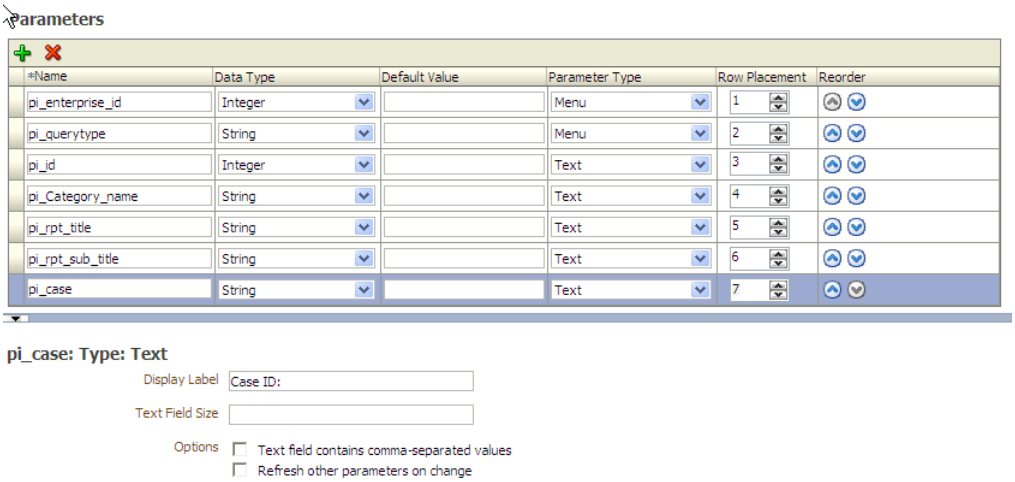
1. Include the parameter in the data set. For example, you want to see data for a *Case ID*. Add **where** condition with a parameter *pi_case* in the data set *G_NEWDS.*



2. Click **Query Builder** and new parameter is created. Click **OK** to confirm.
3. The parameter *pi_case* is now available in the parameter section of the Data Model.



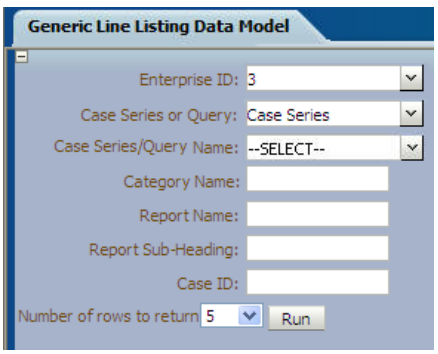
4. Add the display label for the new added parameter, which will be shown at the time of report execution.



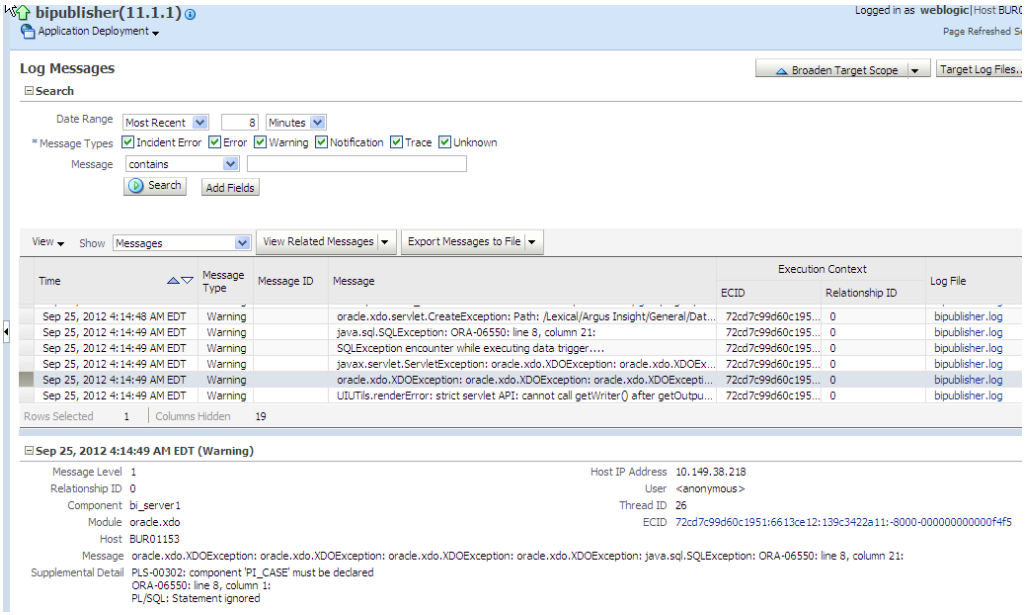
5. Declare the parameter *pi_case* in the Generic Report Line Listing Package *pkg_rep_linelisting*. It is mandatory to declare the parameter in the report package. If the parameter is not declared, the report will not execute. Execute the report and you will be able to search data based on the newly added parameter *Case ID*.

See Also:

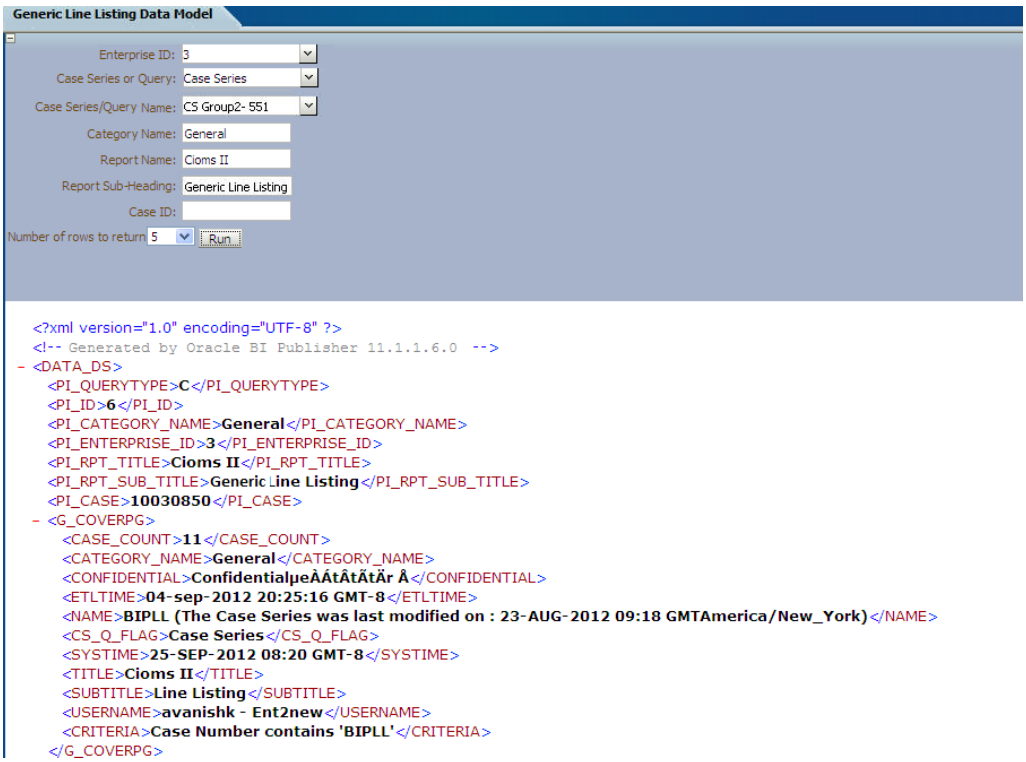
Section 5.1.4.2.2, Add New Parameter in Package



6. If the parameter is not declared in the package, the error message *Component PI_CASE must be declared as shown in the enterprise manager bipublisher logs* displays as shown below:



7. Once the parameter *pi_case* is declared in the package, the report is executed successfully.



5.1.5.3 Event Triggers

The following are the steps to view event triggers:

1. In BI Publisher report, there are three different types of event trigger: *Before Data*, *After Data* and *Schedule*.
2. In the Event Triggers, for the Generic Line Listing Report you will create *Before Data* trigger, which will set the user context before populating all the reporting GTTs. The function called in the Event Trigger as shown in the above picture is:

   pkg_rep_linelisting.f_pop_report_data(:pi_enterprise_id,:xdo_user_name,:pi_id,:pi_querytype)
3. In case, you want to delete some customized tables after data is generated, you can create Event Trigger of type *After Data* and call package with delete statements.

   See Also:

   Report Designer's Guide for Oracle Business Intelligence Publisher

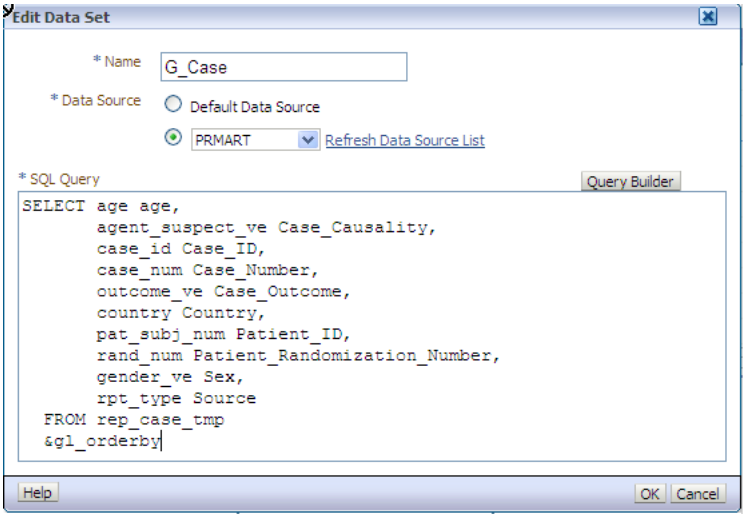
5.1.5.4 Add Lexical Parameter in Data Model

The following are the steps to add lexical parameter in the data model:

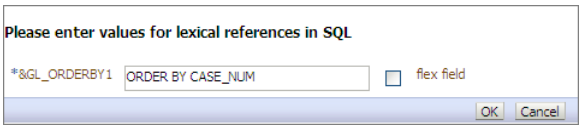
1. Edit the data set *G_Case*. Add Lexical Parameter *&gl_orderby*, as declared in the package.

   See Also:

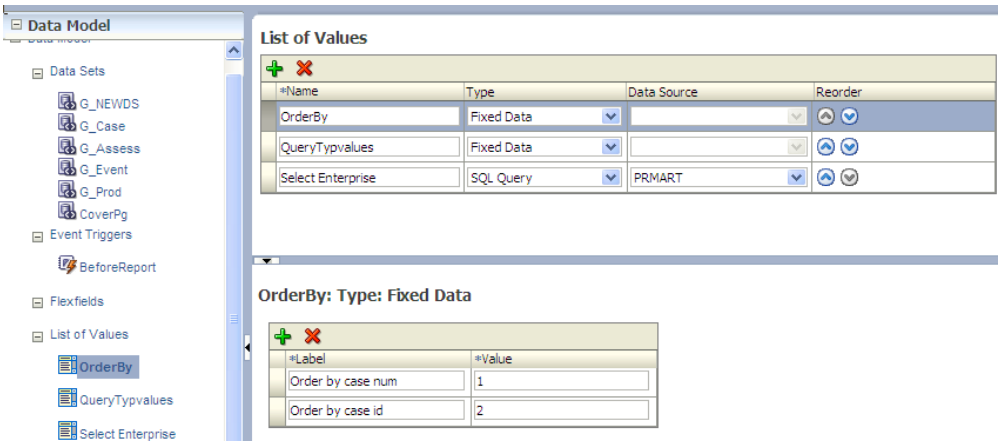
   Section 5.1.4.2.6, Lexical Parameters



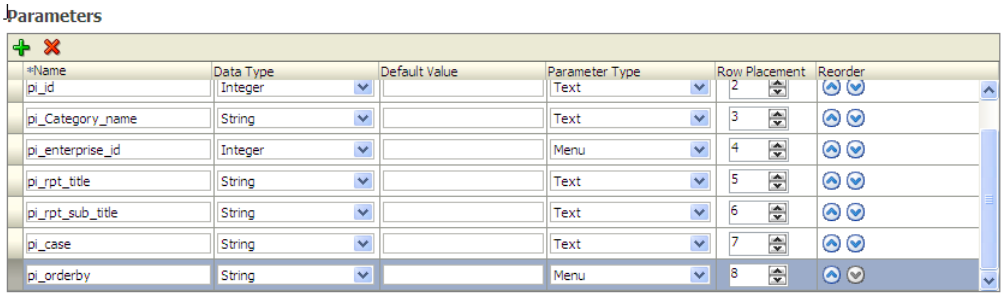
2. When Lexical Parameters are added for the first time in the Data Model, BI Publisher will ask for lexical references in SQL that is, Default Value for the Lexical Parameter.



3. Create a List of Values, **Order By** as shown below:



4. Create the parameter *pi_orderby* in the Data Model and assign the *LOV-OrderBy* as shown below:

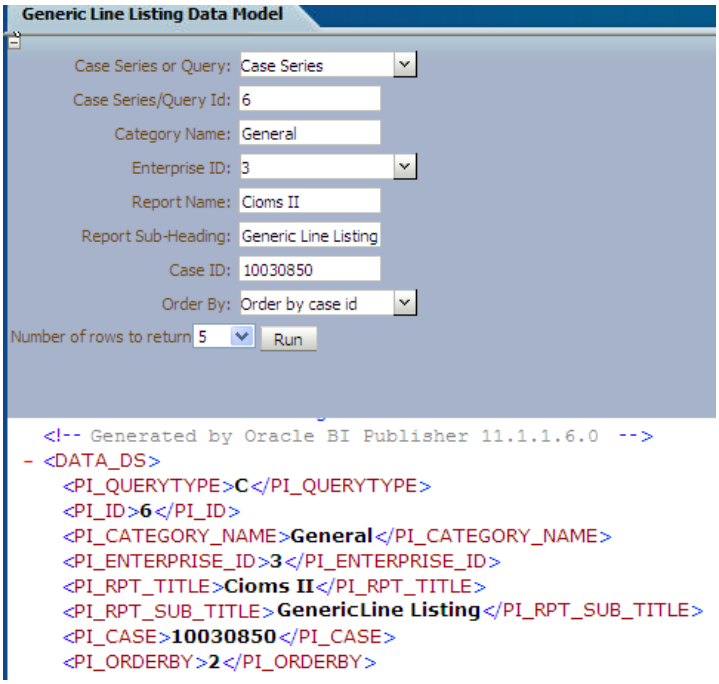


5. View the Report by selecting the parameter *OrderBy*.

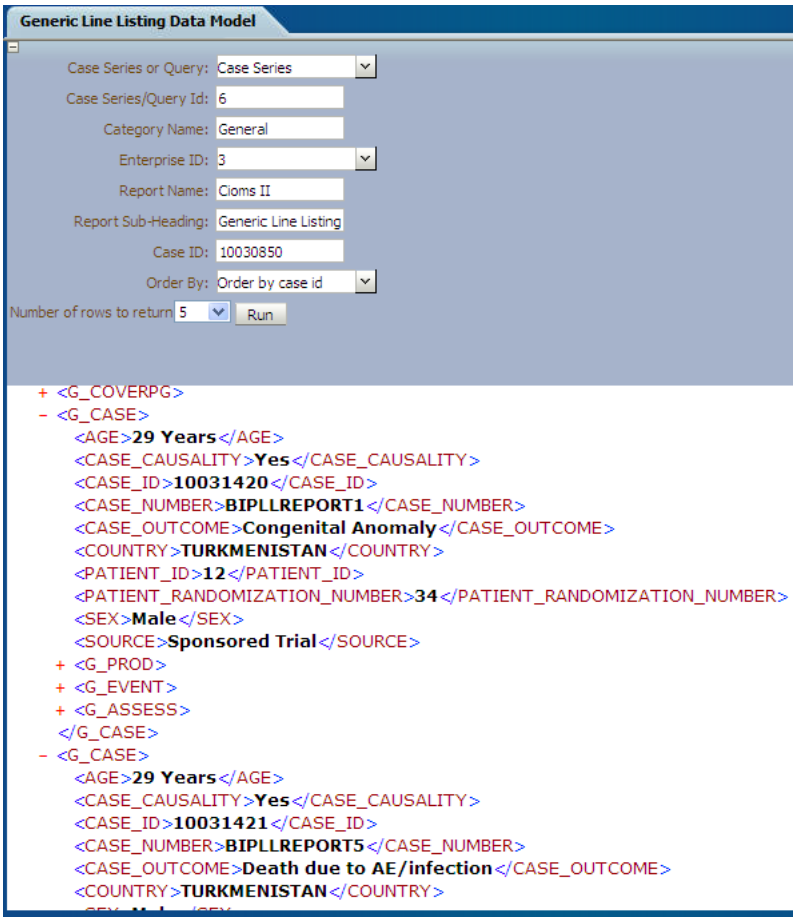
6. Execute the Report and verify that data is in order by Case ID as per the selected option. You can find that the XML value of *pi_orderby* is '**2**'. In the package *pi_orderby* value '**2**' means Order By *case_id*.

See Also:

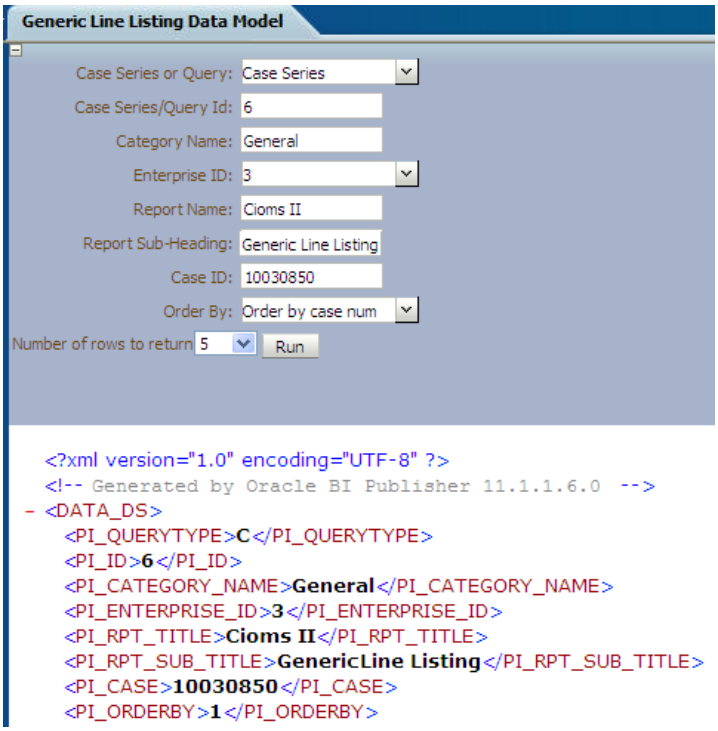
Section 5.1.4.2.6, Lexical Parameters



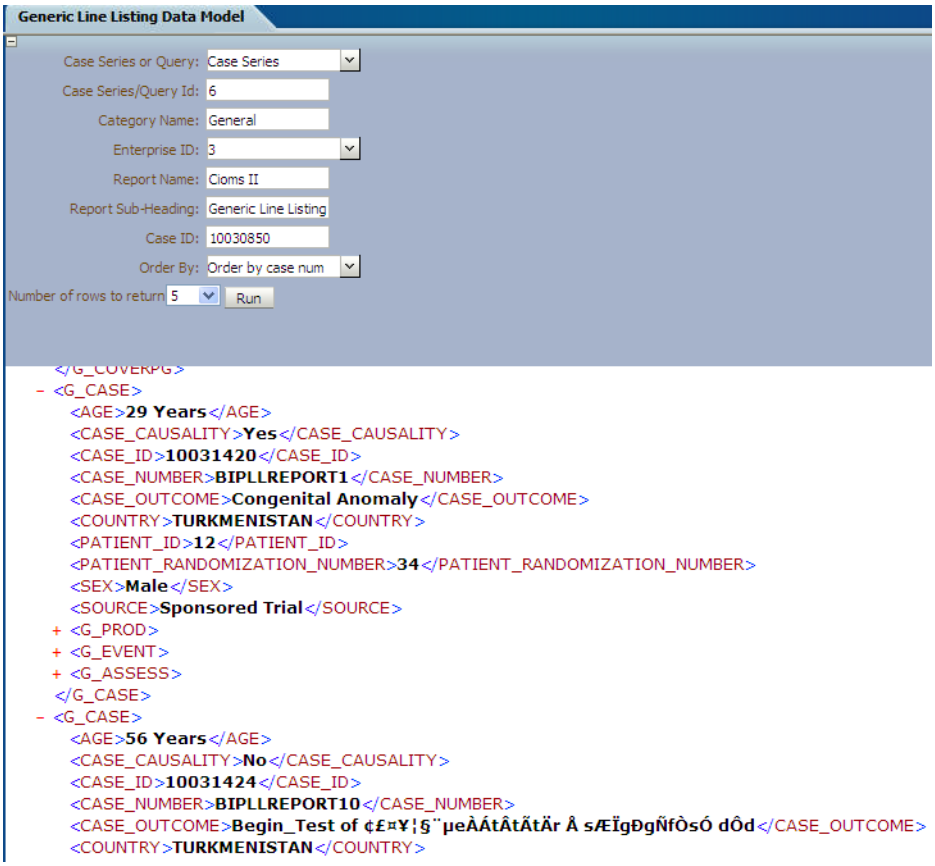
7. Check the case data for the order of cases by *case_id: 10031420* and *10031421* in figure shown below:



8. Now, select the *Order By case_num* option in the Data Model.



9. Verify the case data for order of cases by *case_num: 10031420* and *10031424*, in the figure shown below:



5.1.6 BI Publisher Report Templates

This section explains the types of report template used in BI Publisher Report as follows:

- Layout Editor
- Rich Text File Template

To view Event Assessment Data in the reports, you should create Event Assessment as a separate block in both Layout Editor and Rich Text File (RTF) template; Product and Event details should be fetched from the Event Assessment Level only to see Event Assessment Data.

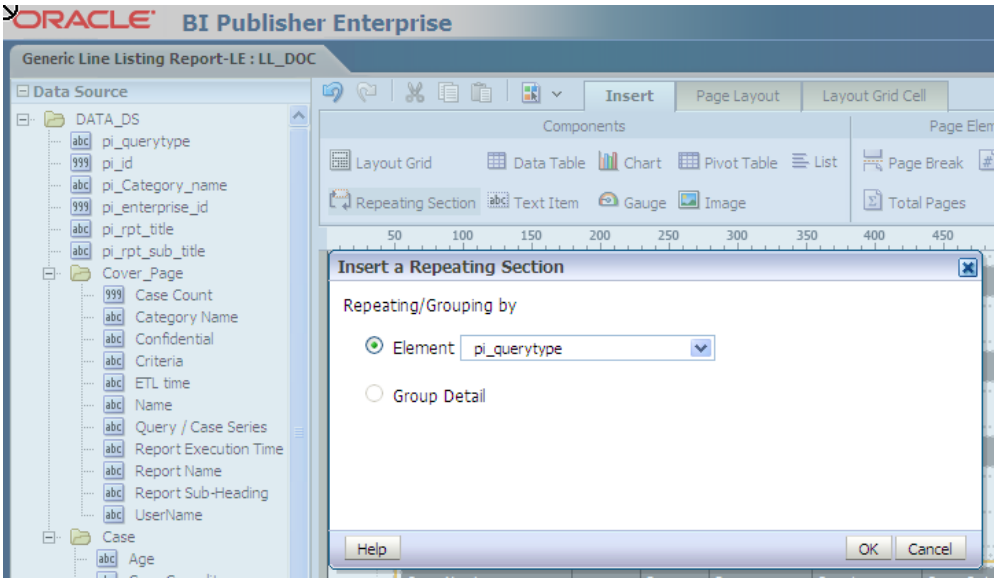
This section also explains:

- BI Publisher Logs

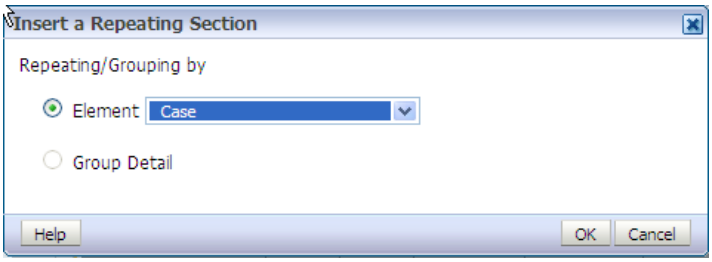
5.1.6.1 Layout Editor

The following are the steps to edit/modify an existing report layout:

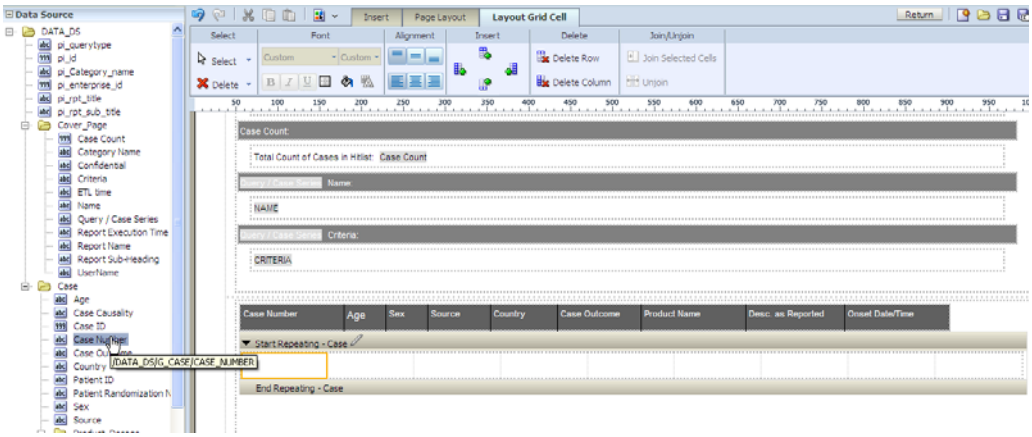
1. Create a Repeating section as shown below:



2. Select a valid *Group Name* that is, **Element** from the element drop-down list.

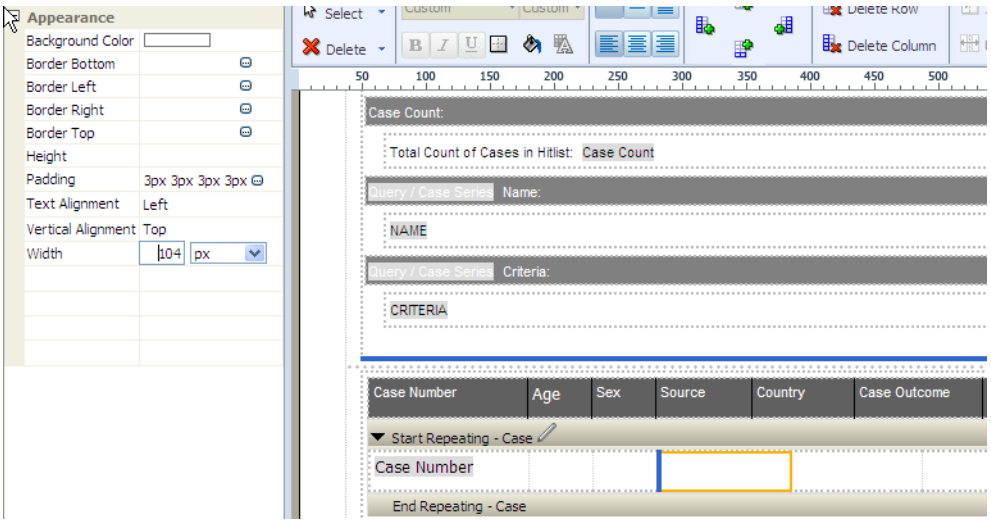


3. A Repeating section is created.
4. Add columns in the Repeating section. For example, click **Case Number** and drag it to the Report Layout section.

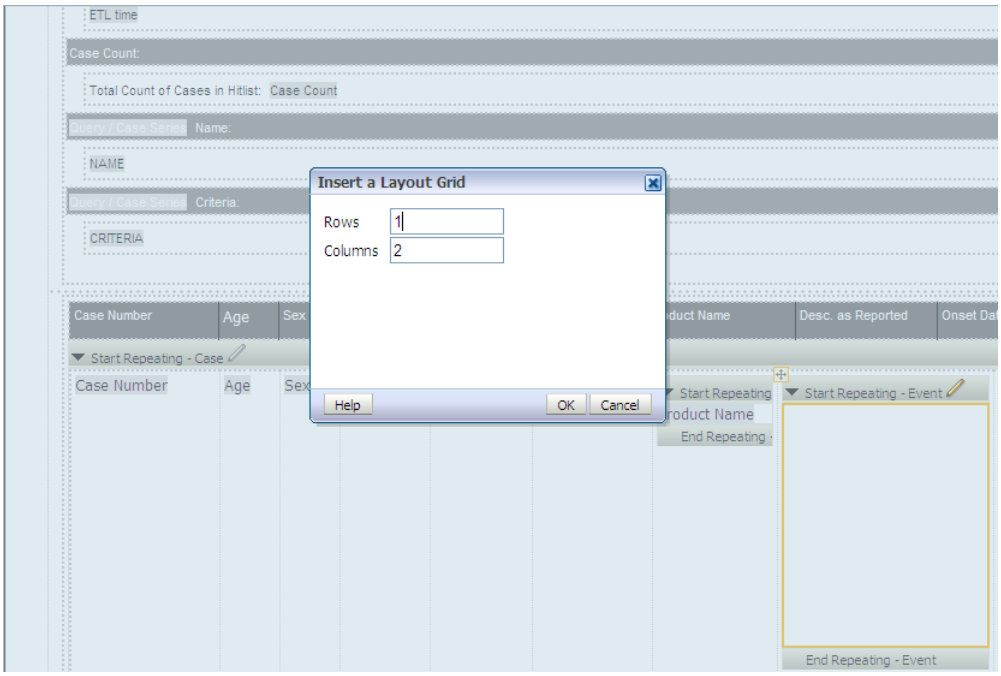


5. Drag Case Level columns only in the above Repeating section. Columns from other groups, such as **Product** or **Event** should not be included here.

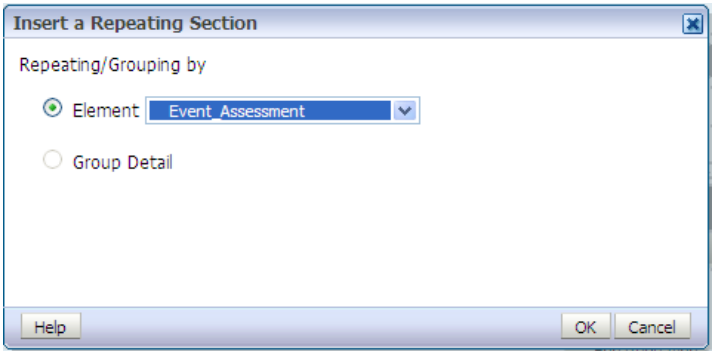
6. Add a child Repeating section for the Product.



7. In the Repeating section, you can add **Layout Grid** with as many required columns as you want to include in the report.



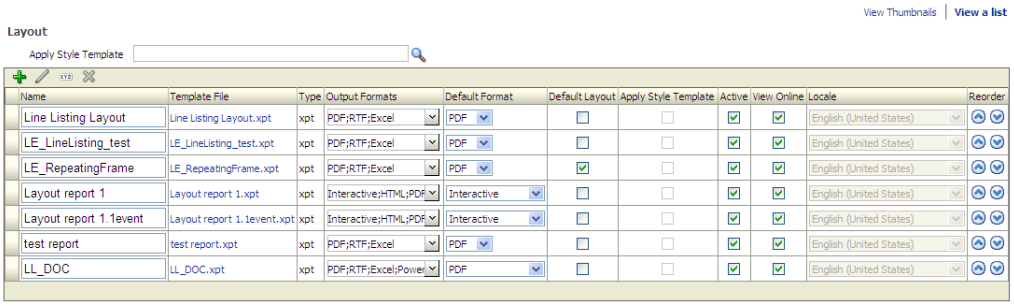
8. Add Repeating section for child group *Event Assessment*. Once added, save the report and click **Return**.



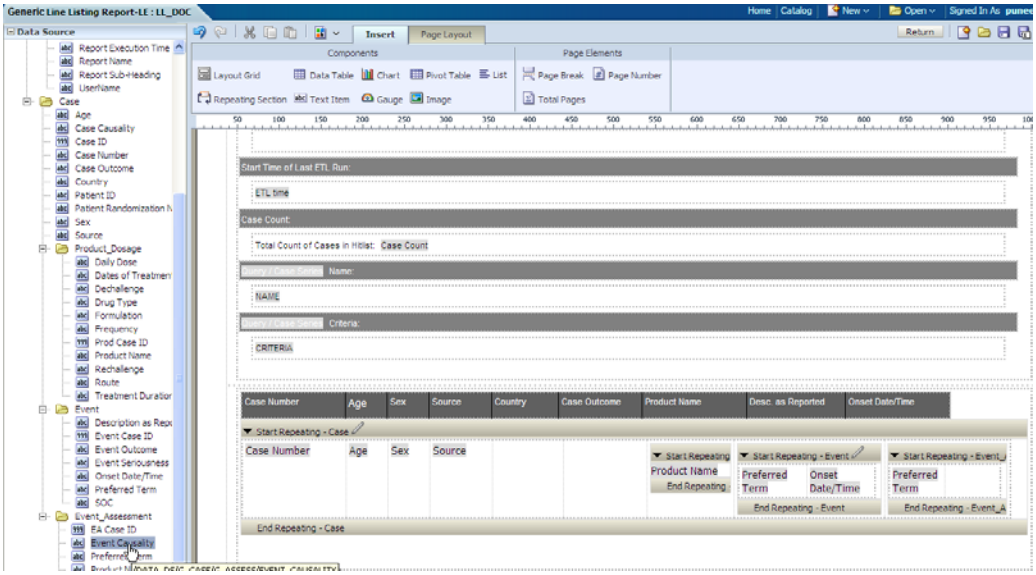
9. The Report is displayed as shown below:

LL_DOC
Edit | Properties | Delete

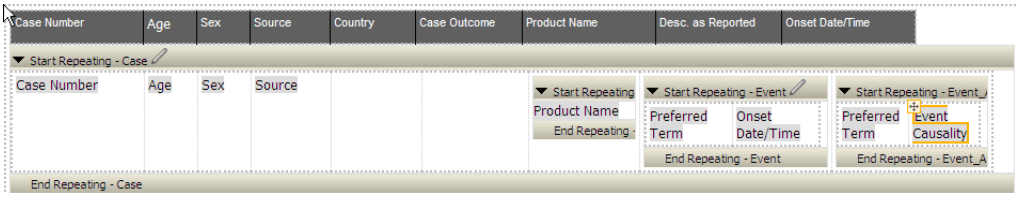
10. Click **View a list** to select Default Format, Default Report and etc.



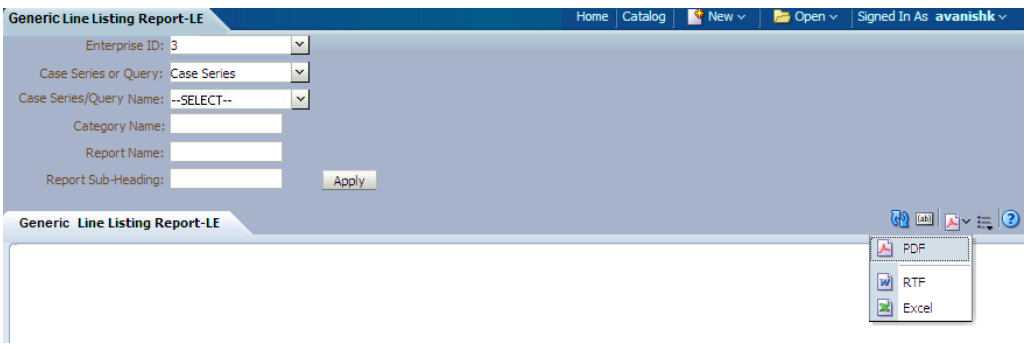
11. To add more columns in a Repeating section, go to Data Source panel and select the required column from the appropriate group. Drag the selected column into the Repeating section.



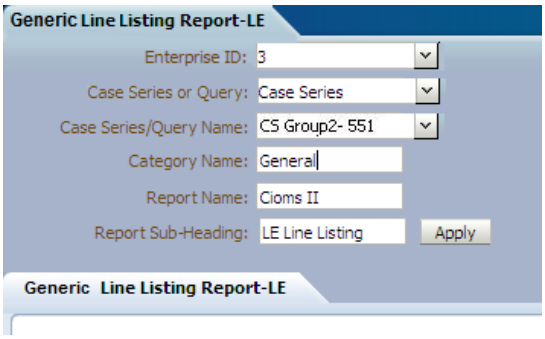
12. The column *Event Causality* is added in the **Event Assessment** section.



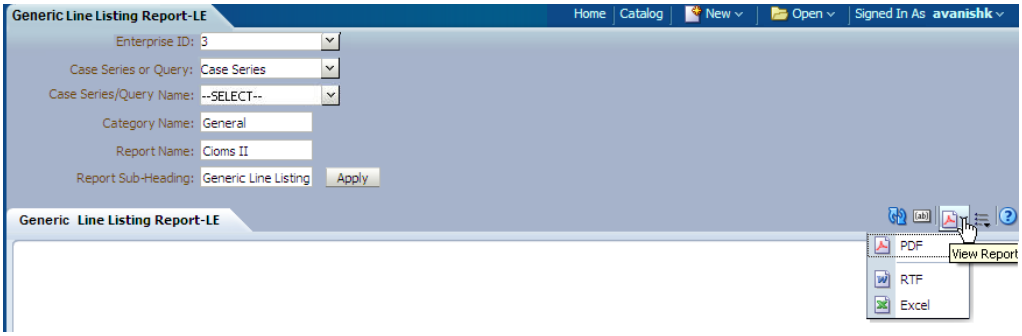
13. To execute the report, click **Report Link** or **Open** the report. The following screen displays:



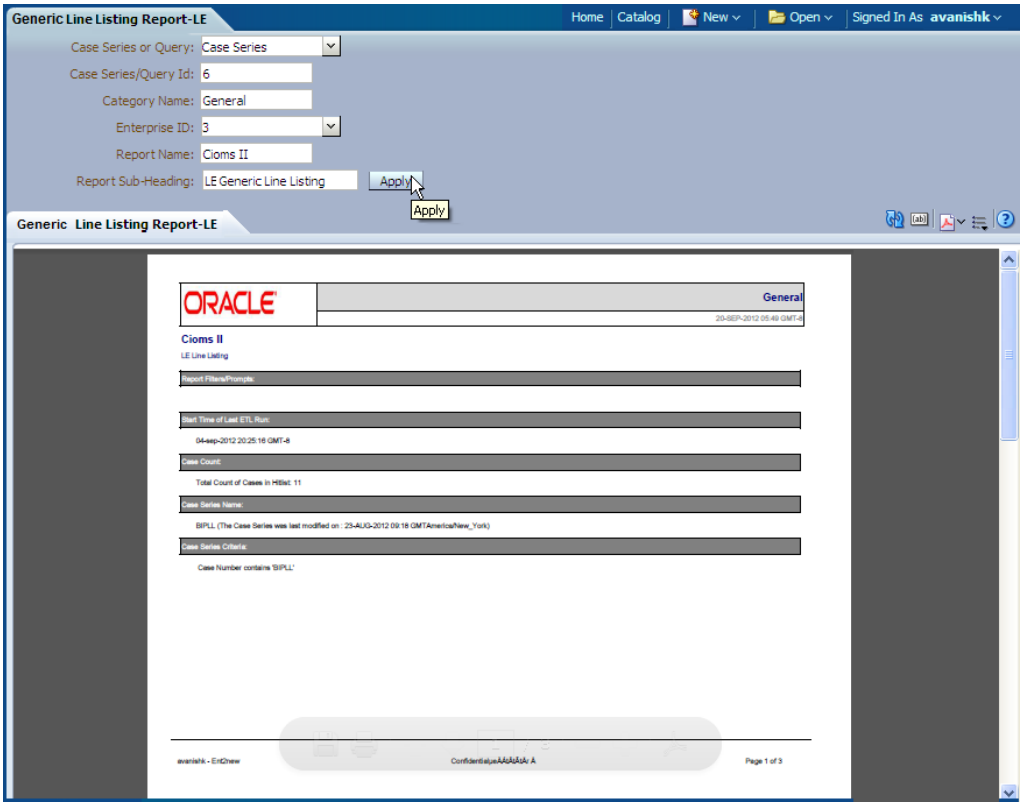
14. Enter the appropriate parameters.



15. Select a report output type, like *PDF*.



16. The report is generated in PDF format.



5.1.6.2 Rich Text File Template

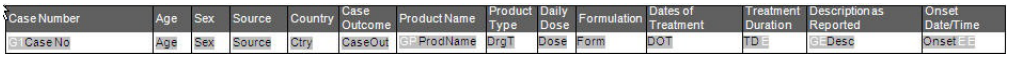
The RTF template has a main template and one sub-template. You can use the sub-template in any future reports.

- **Sub-template:** The sub-template cover page details are as shown below:

  The sub-template is divided into three categories:

  - **Template- Header:** It contains Company Logo, Report Run Date, Report Category, Report Title, and Report Sub-heading.
  - **Template- CovPG:** It contains Report Prompts, Start ETL Time, Case Count, Query/Case Criteria and Name.
  - **Template- Footer:** It contains Login User, Confidentiality and Page Number.

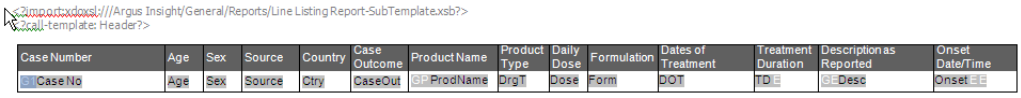
- **Main Template:** In this template the report columns are created in different tables for different groups. Besides, sub-template should be called in the Main Template as shown below:



Adding New Column in RTF

The following are the steps to add a new column in RTF:

1. Remove any existing column from the specific group, like Product or Event and add a new column from the same group. Or, reduce the width of the column to add a new column without removing an existing column.
2. To view **Event Assessment** values, **Product** and **Event** information should be fetched from the Event Assessment Level only. You should not compare Event Assessment Data with Product and Event level columns given in the Default Report template.
3. Click **Edit** in the RTF template report and save the RTF template at your local machine.



The File Download dialog box appears.

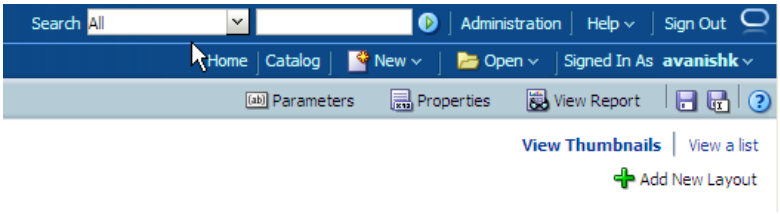
4. Click **Open** to display the RTF template document. Double-click on any existing column of BI Publisher. The BI Publisher **Properties** displays. Enter any valid XML tag for BI Publisher columns.

   See Also:

   Section 5.1.5.1.2, Add New Data Set for XML tags available under the Data Sets **Structure** tab.

5. Modify the column *Country* to display *Patient Random Number* column and save the RTF.

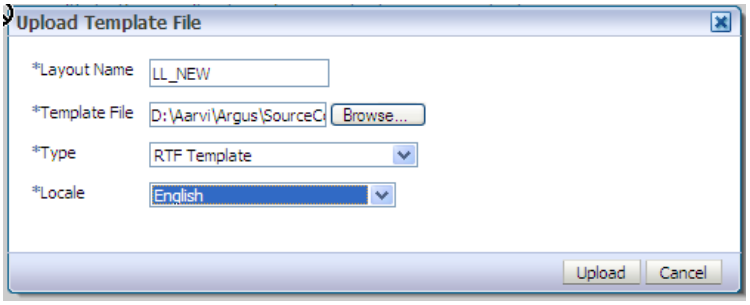
6. Upload RTF to the report. Click **Add New Layout** option as shown below:



The Upload Template File dialog box appears.

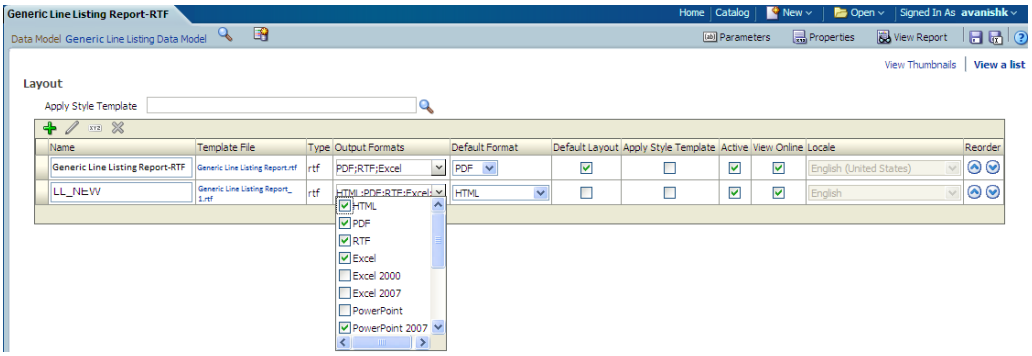
7. Click **Upload**.

8. Select the new **RTF template**.

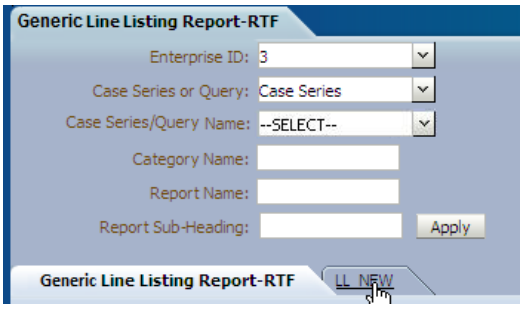


9. Once uploaded, you can find two layouts in Thumbnail format as shown below:

10. Click **View a list** option to select Default Report and Output Format options. Once you have saved the changes, click on view report option to execute the report.



11. You can find both the Layouts and can view any Report Template Output by selecting the appropriate tab. After passing correct parameters click **Apply**.



See Also:

Oracle Business Intelligence Publisher Technical Reference Manual > Report Designer's Guide > Oracle Fusion >Creating an RTF template section.

5.1.6.3 BI Publisher Logs

While running BI Publisher report, by passing incorrect/invalid parameters, sometimes you may get the following error messages:

File does not begin with '%PDF-'. Local\EWHa4ipsm8u

Or,

Verify the BI Publisher logs from the Enterprise Manager.

You can verify the AI log tables or login to enterprise manager to check the BI Publisher server logs.

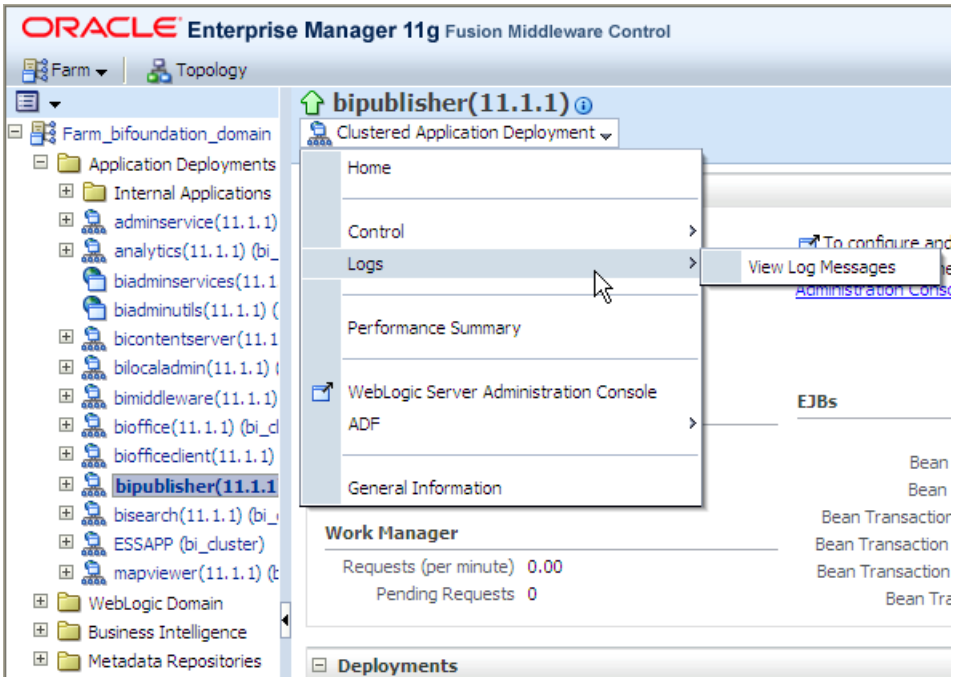
See Also:

Section 5.1.4.2.4, Log (Audit) Table

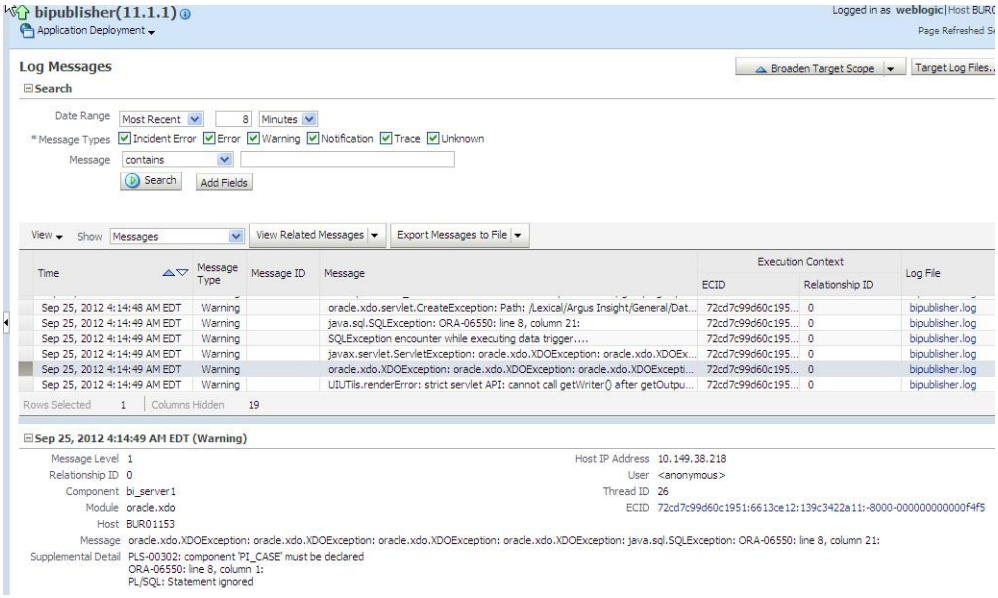
The following are the steps to check BI Publisher server logs:

1. Login to **Enterprise Manager**.

2. Click **Applications > BI Publisher**.
3. Click **Clustered Application Deployment > Logs and View Log messages** as shown below:



4. Select the *Date Range* or *Message Type* and click **Search**. The BI Publisher logs displays as the search result.



5.1.7 BI Publisher Reporting Tips

You can extend our existing report model using the following actions:

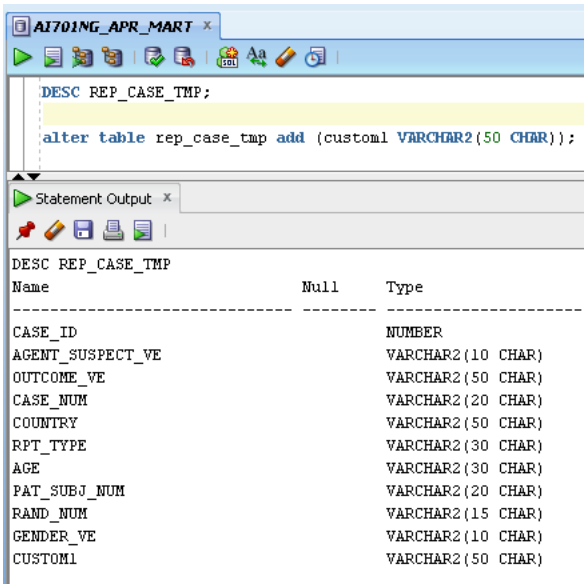
- Add Column in Global Temporary Tables
- Populate New Column in User Exit Package

- Add New Column in Data Set
- Add New Column in Layout Report

5.1.7.1 Add Column in Global Temporary Tables

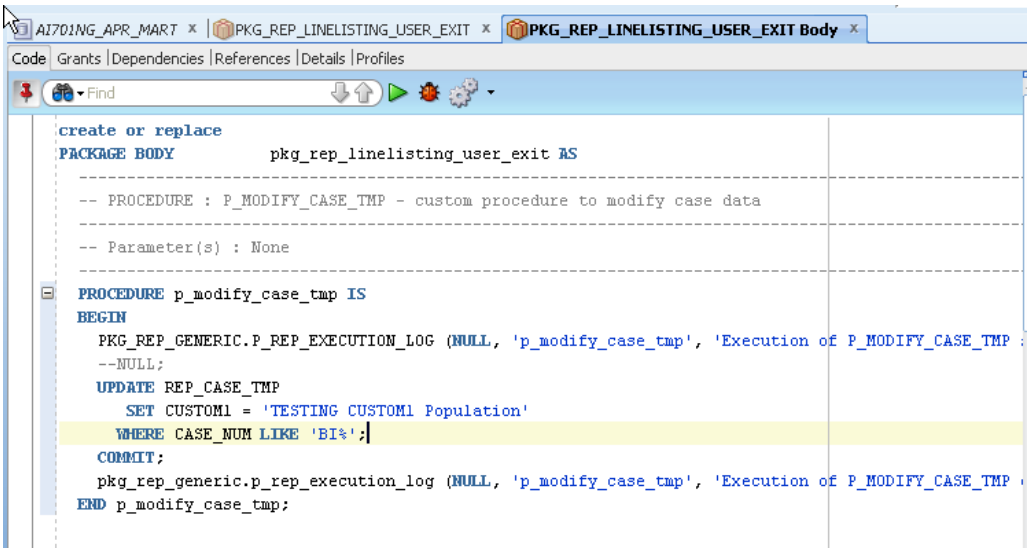
The GTTs are created in the MART database.

To add new column in a GTT, login to the **Mart schema** and add a new column *CUSTOM* in the *GTT rep_case_tmp* as shown below:



5.1.7.2 Populate New Column in User Exit Package

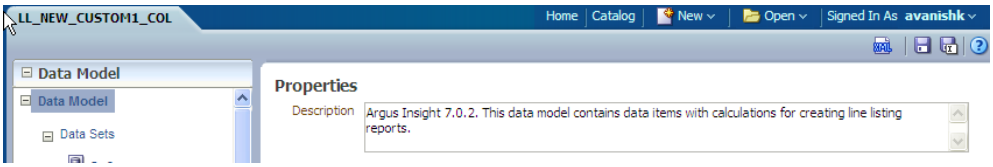
You can populate the column *CUSTOM* in User Exit package by modifying the package to include your DML statements and compile the package as shown below:



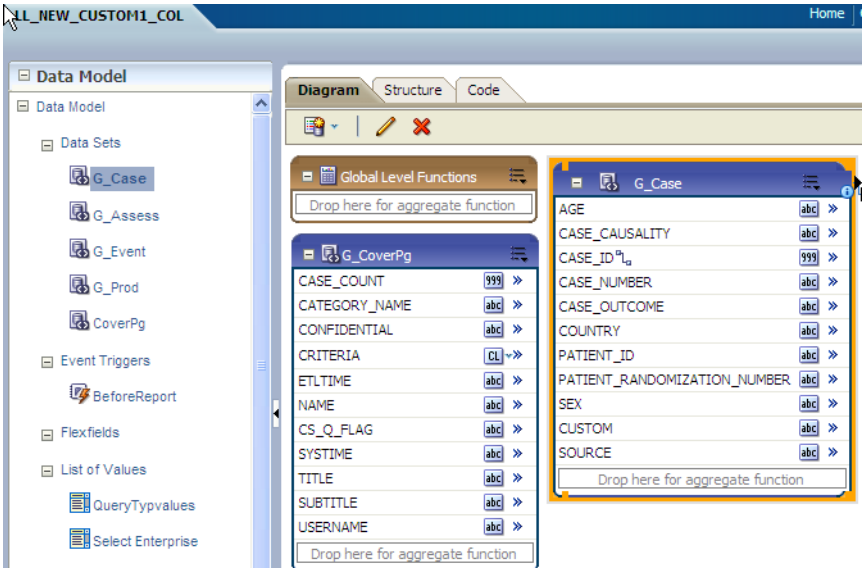
5.1.7.3 Add New Column in Data Set

The following are the steps to add a new column in the data set:

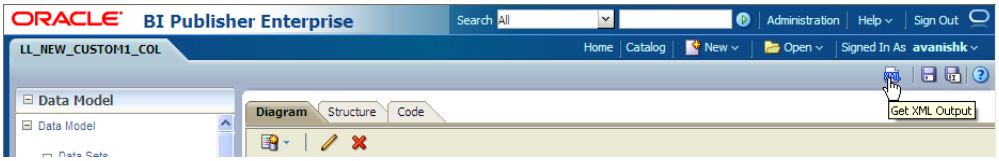
1. Edit the existing Data Model and save the new Data Model with appropriate name, such as **LL_NEW_CUSTOM1_COL**.



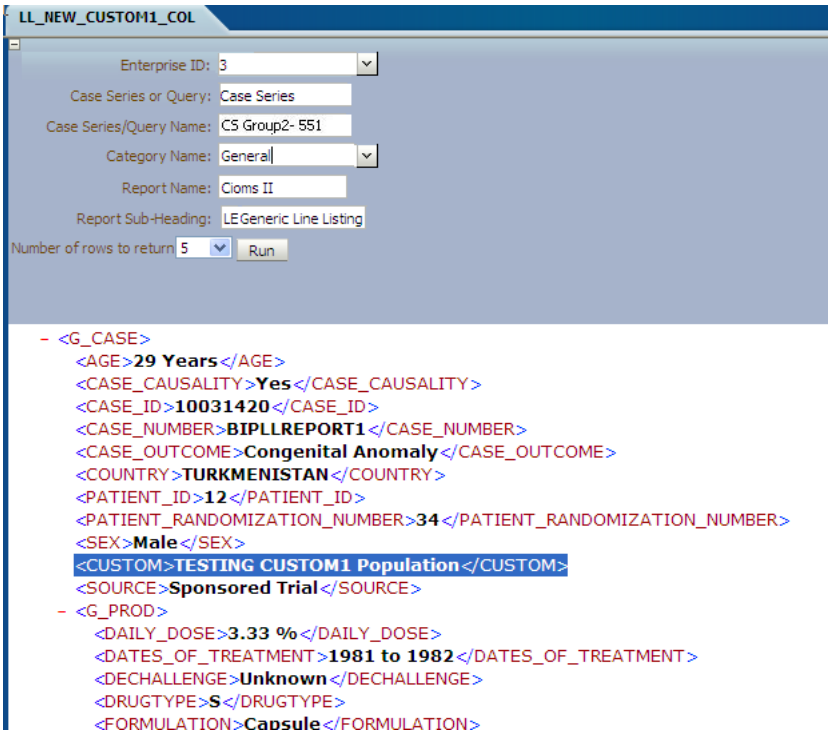
2. Edit the data set *G_Case*, include the new column and save the Data Model. The column *CUSTOM* is added to the data set as shown below:



3. Click **Get XML Output** to view the XML output of the new data model.



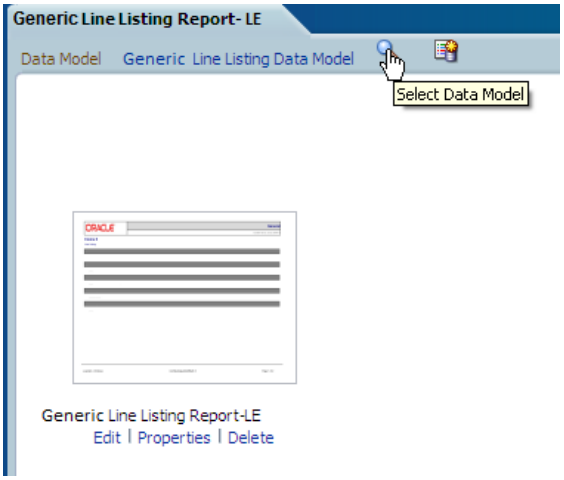
4. In the above generated XML output, verify the column *CUSTOM* that is populated with the value as per the logic written in the *User Exit* package.



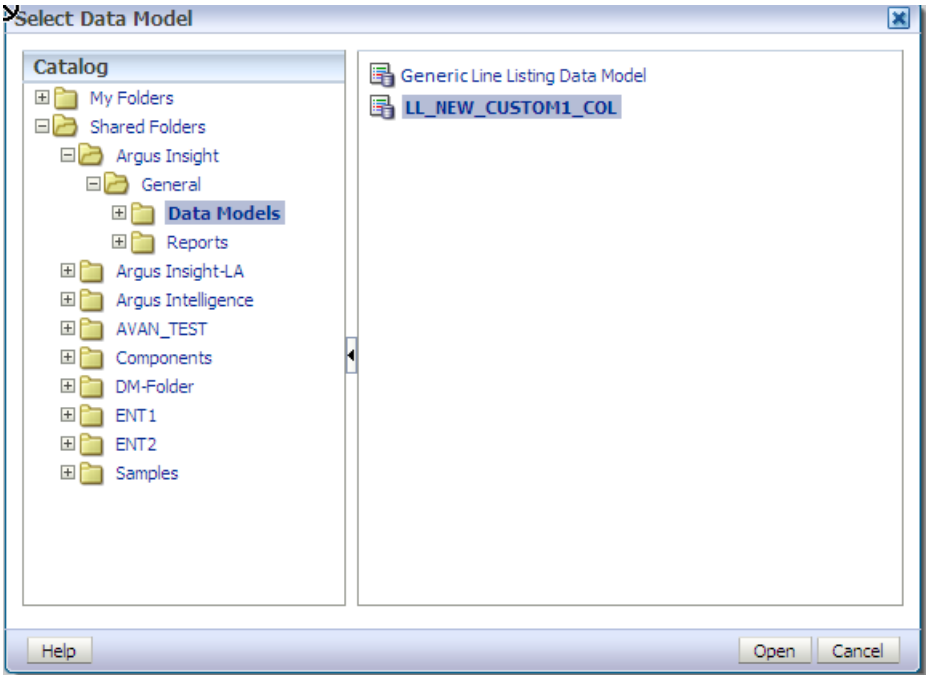
5.1.7.4 Add New Column in Layout Report

The following are the steps to add a new column in the Layout Report:

1. Edit the existing Layout Report and save as **LL_NEW_CUSTOM_LE**. Check that new data model is selected for the new Layout Report.

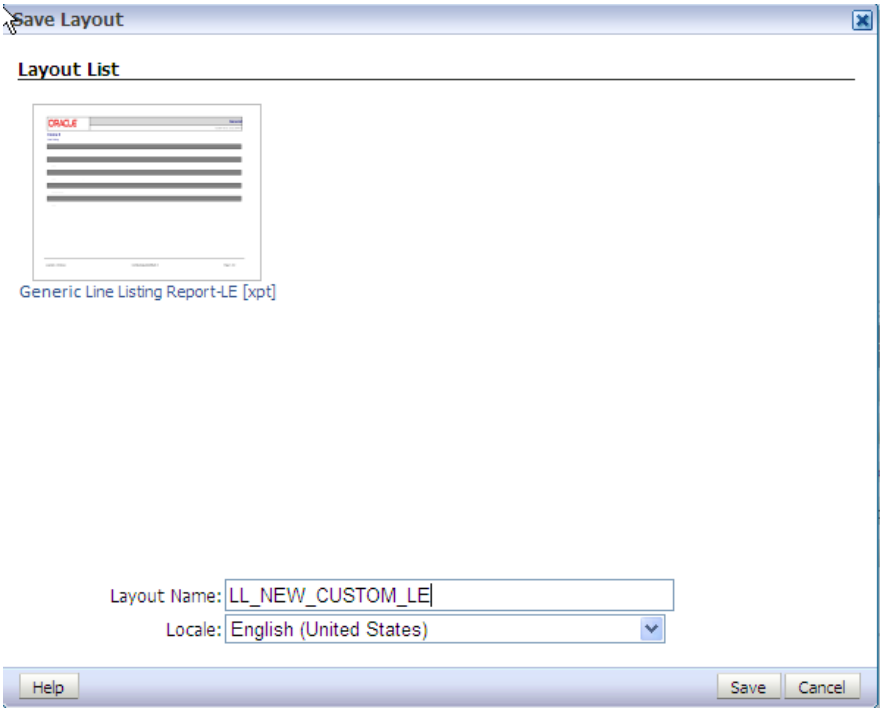


2. Select the Data Model **LL_CUSTOM1_COL**

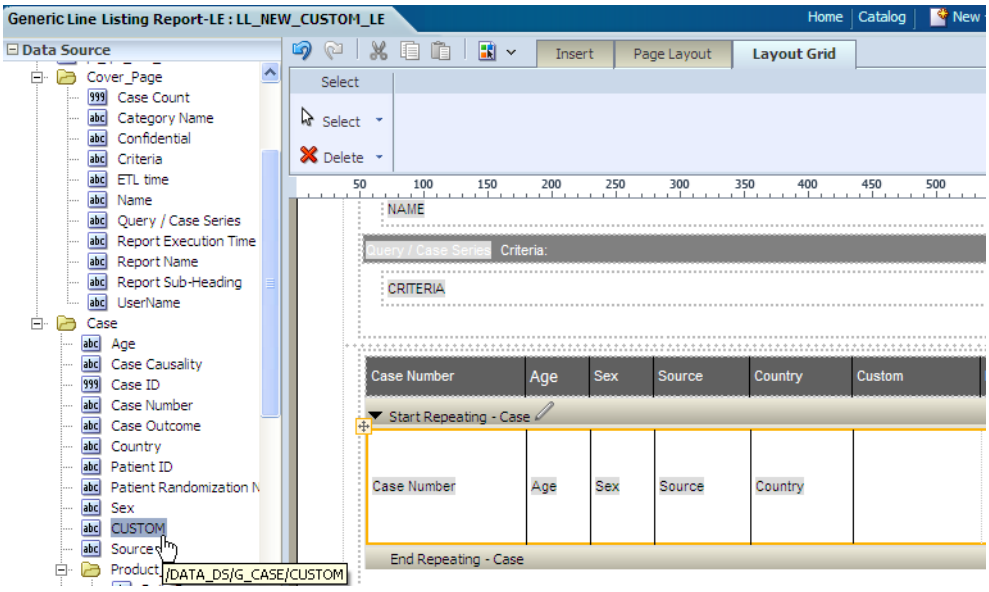


3. At the top-left corner, you can see the new data model as selected for the Layout Report.

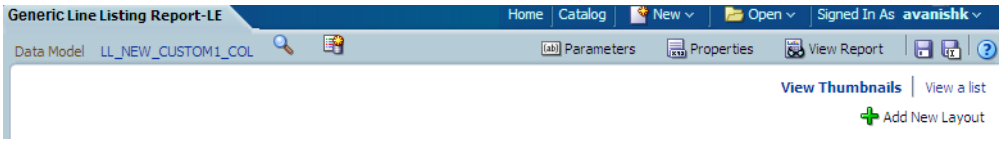
4. Save the Layout Report as **LL_NEW_CUSTOM_LE.**



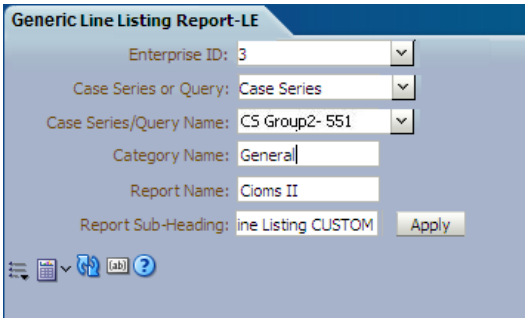
5. In the Data Source panel you can view the column *CUSTOM*.



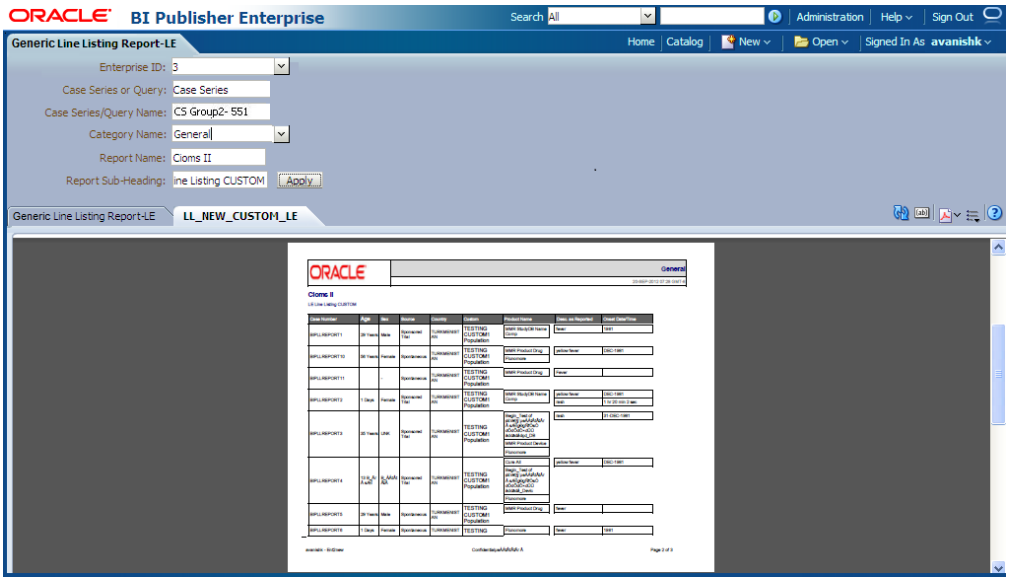
6. Drag the column and include in the **Case Repeating** section only. Save the Layout Report. Click **Return** and then click **View Report**.



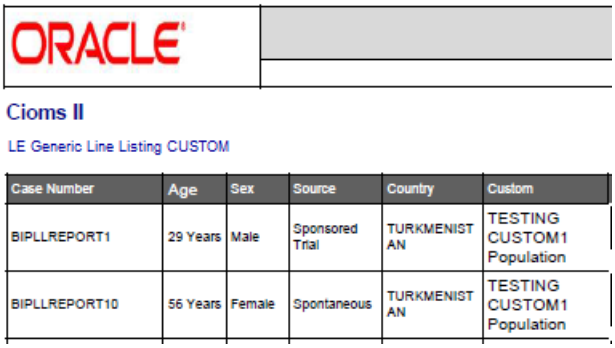
7. Enter the appropriate values to the *Report Parameters* and click **Apply**.



8. Check that the report is executed successfully with CUSTOM value populated as per the logic.



9. You can see that the column *CUSTOM* is populated.



ORACLE

Cioms II

LE Generic Line Listing CUSTOM

Case Number	Age	Sex	Source	Country	Custom
BIPLLREPORT1	29 Years	Male	Sponsored Trial	TURKMENISTAN	TESTING CUSTOM1 Population
BIPLLREPORT10	56 Years	Female	Spontaneous	TURKMENISTAN	TESTING CUSTOM1 Population

5.2 OBIEE Extensibility

Argus Insight provides an out of the box RPD for analyzing the aggregate reporting data which is generated by Argus Safety/BI Publisher. As part of BIP aggregate reporting generation, Argus Safety system populates log tables. These tables are used in the RPD for further analysis by creating OBIEE Answers and Dashboards.

The BIP tables in Argus Mart are populated from Argus Safety (BIP enabled) through Argus Mart Initial/Incremental ETL. The Initial ETL will fetches all the data, whereas the Incremental ETL fetches only the updated data between the last ETL execution time and the current execution time.

Incremental ETL will not fetch the purged data from Argus Safety.

For more details on data purging, refer to *Oracle Argus Safety 8.2 BIP Extensibility Guide > Section 4.1.7*.

This section comprises the following topics:

- Assumptions
- RPD Architecture
- Add New Dimension Using Flex Bucketing
- Create Custom Dashboards and Prompts

5.2.1 Assumptions

The OBIEE extensibility has the following assumptions:

- The user has a working knowledge of Dashboard/BI Answers and RPD in OBIEE.
- The RPD and Catalog are deployed as per the *Oracle Argus Insight 8.2 Installation Guide*.

5.2.2 RPD Architecture

The RPD architecture comprises the following layers:

- Physical Layer
- BMM Layer
- Presentation Layer

5.2.2.1 Physical Layer

The following tables are fetched into the physical layer of the RPD as Facts:

- Case (RM_RPT_AGG_CASE)
- Drug (RM_RPT_AGG_DRUG)
- Event (RM_RPT_AGG_EVENT)
- Event To Drug (RM_RPT_AGG_EV2DRUG)

The various tables used in Physical Layer are:

- Code List Discrete Table
- Dimension Tables
- Prompts
- Connection Pool
- User Security Table
- Event Polling Table
- Facts
- Measure

Code List Discrete Table

Most of the dimensions are based on the Code List Discrete table (RM_CODE_LIST_DETAIL_DISCRETE_D). It contains all the code list IDs like COUNTRY, DOSE_UNITS etc, and their display value.

Few tables such as Drug names, Event reactions are from the Actual tables.

See Section 5.2.3, Add New Dimension Using Flex Bucketing, for details on how the Code List table is used as a Dimension.

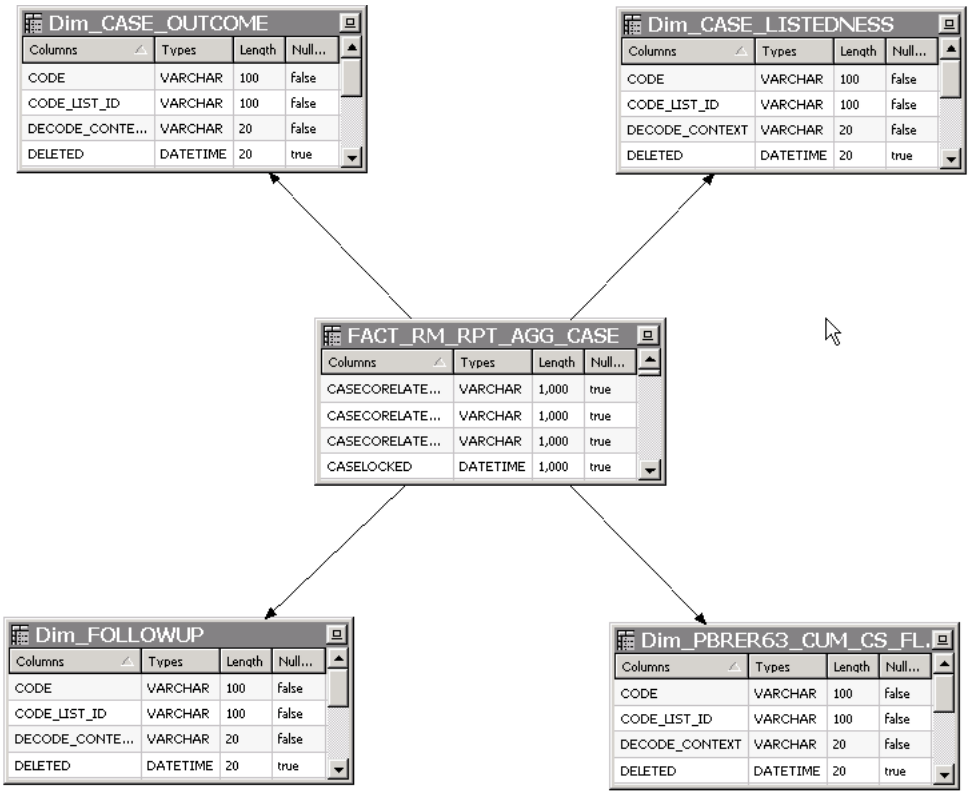
In physical layer of the RPD aliases for all the dimensions and facts are available. For the Code List Discrete table multiple aliases for different dimension attributes are available.

For example:

Case Seriousness, Case Listedness, and Event Outcome are from the Code List Discrete table, so for each code list ID an alias and a dimension is made available.

The following screen displays the joins of these dimensions with the respective Fact table:

Figure 5–1 Joins of Dimensions with the Fact table



See Section 5.2.3, Add New Dimension Using Flex Bucketing, for details on how the Join conditions are applied.

Dimension Tables

Other Dimension tables are from direct tables such as Drug names and Reactions etc.

Few dimensions are derived from the select statements. The following tables are created:

- RM_RPT_AGG_CLINICALDRUGROLE_D
- RM_RPT_AGG_DIAG_SYMPT_FLAG_D
- RM_RPT_AGG_TREATMENT_LIST_D
- RM_RPT_AGG_REACTION_D
- RM_LM_CLINICAL_REF_TYPES_D
- RM_LM_REF_TYPES_D
- RM_RPT_AGG_PERIOD_D
- RM_RPT_AGG_PRIM_STUDY_PROD_D
- RM_RPT_AGG_PROD_NAMES_D
- RM_RPT_AGG_STUDY_ID_D
- RM_RPT_AGG_STUDY_NAMES_D

Prompts

The Dashboard and Page prompts dimension available are:

- Dim_Enterprise_Id
- Dim_Report_Form_Id
- Dim_Report_Template
- Dim_Report_Type

These prompts are created from the following tables:

- RM_RPT_AGG_ENTERPRISE_ID_D
- RM_RPT_AGG_PROMPTS_D
- REPORT_FORM_ID_D

For more information on these dimensions, see Appendix: Dimensions and their Mapping.

Connection Pool

The connection to Argus Mart is established using the AM_BI user, which is a Read-only user created during Argus Mart schema creation.

To display the enterprises along with their data as per the user access rights, set context as 0 (zero) in Connection Pool.

Other security settings are taken care by the User Security table.

User Security Table

A periodic report configuration that is created in Argus Safety can be shared across multiple user groups. The users under these user groups will have access to Modify and Execute the Report Configuration. This information is saved in the security table RM_RPT_AGG_USER_ACCESS_S.

This security table is joined to all the Facts, so that for the logged in user, only those reports information is available which he has access to. Other data security (blinding etc) settings are taken care by BIP tables in Argus Safety.

Event Polling Table

An Event Polling table RM_BI_S_NQ_EPT is created to handle event polling.

Refer to the *Oracle OBIEE Guide* for more information on Event Polling.

5.2.2.2 BMM Layer

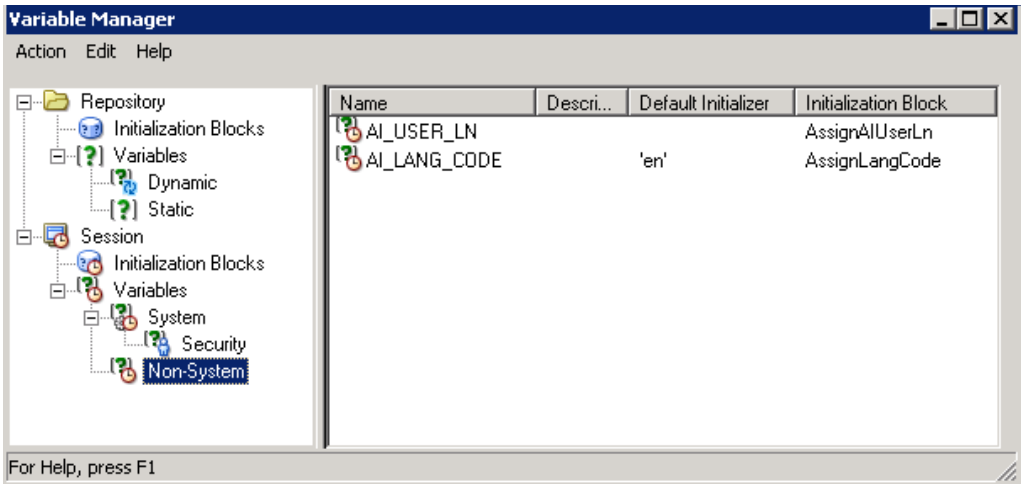
For all the dimensions, logical hierarchies are created at this layer and WHERE clause is added.

See Section 5.2.3, Add New Dimension Using Flex Bucketing, for an example of setting the WHERE clause.

In the Argus Insight RPD, two session variables are created:

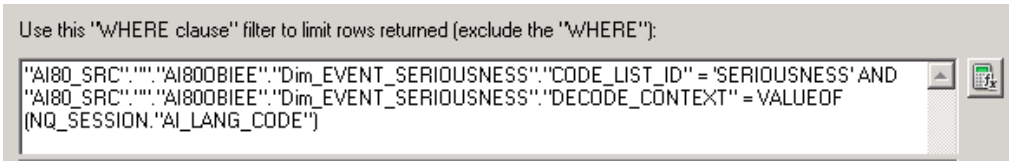
- AI_USER_LN — Validates the logged in user name.
- AI_LANG_CODE — Contains the value **en**. Avoids hard coding of the value in the WHERE clause in the BMM layer at various places.

Figure 5–2 Variable Manager in RPD



For example:

Figure 5–3 BMM layer — WHERE clause using AI_LANG_CODE



Facts

The following are the logical combination of fact tables that are created in the RPD:

- Case Fact
- Drug Fact
- Event Fact
- Event to Drug Fact
- Case Event Fact
- Case Drug Fact
- Case Event to Drug Fact
- Consolidated Fact

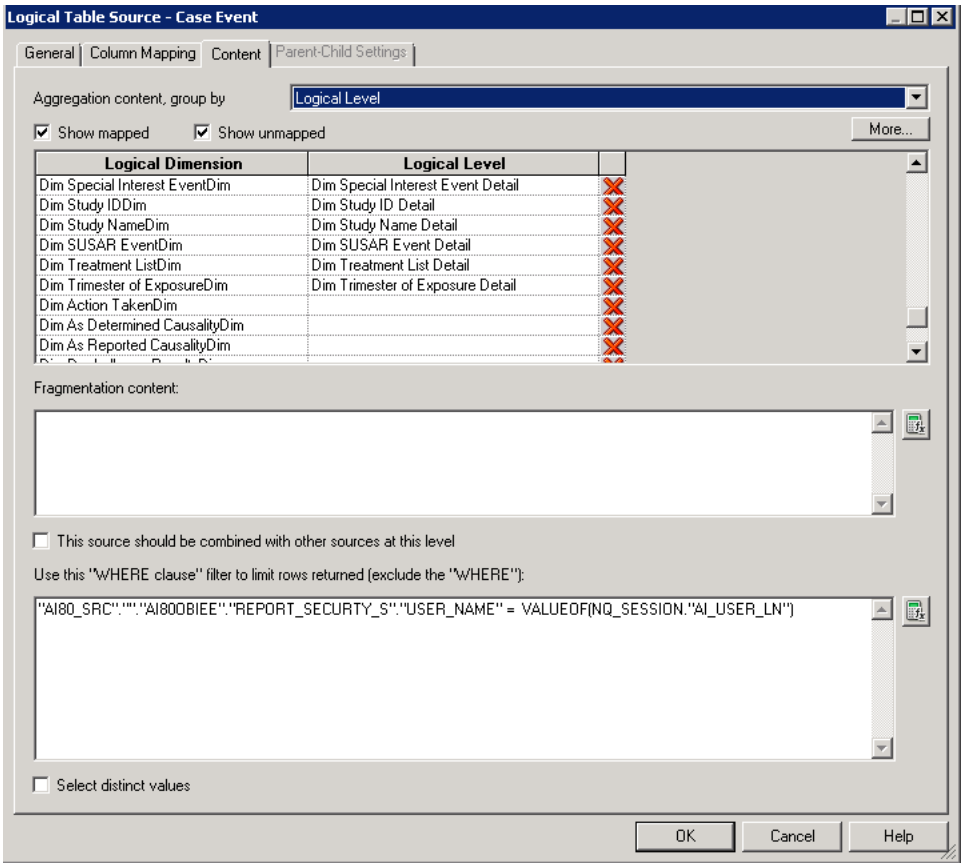
For example:

- Case Fact in physical table is FACT_RM_RPT_AGG_CASE.
- Case Event Fact is a combination of tables FACT_RM_RPT_AGG_CASE and FACT_RM_RPT_AGG_EVENT.

See Appendix: Dimensions and their Mapping, for details of RPD including dimensions, Fact tables and their joins.

The logical level should be set for each dimension (based on the access of each dimension) for all the logical Facts properly.

Figure 5–4 Logical Table Source

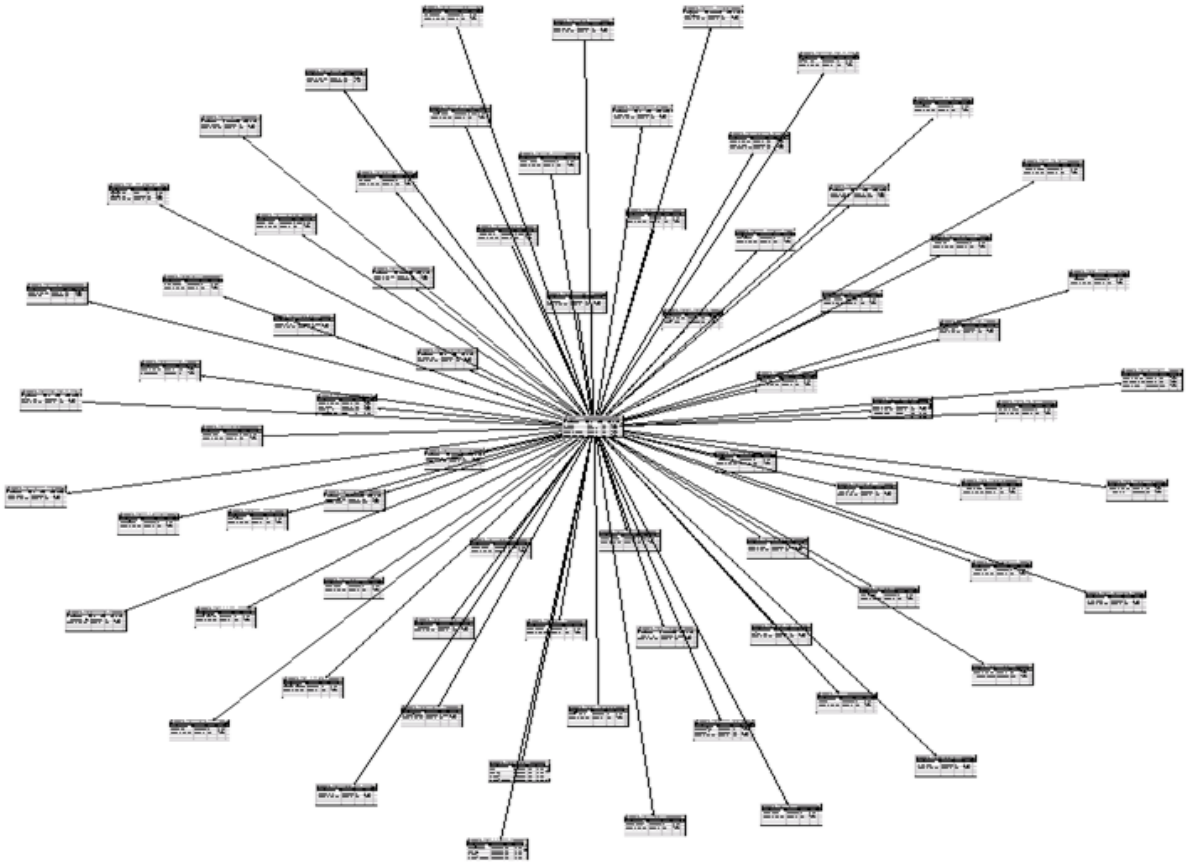


For Example:

The logical fact Case Event will have the dimensions that are applicable to Case and Event tables only.

The BMM layer should be a perfect star schema as shown below:

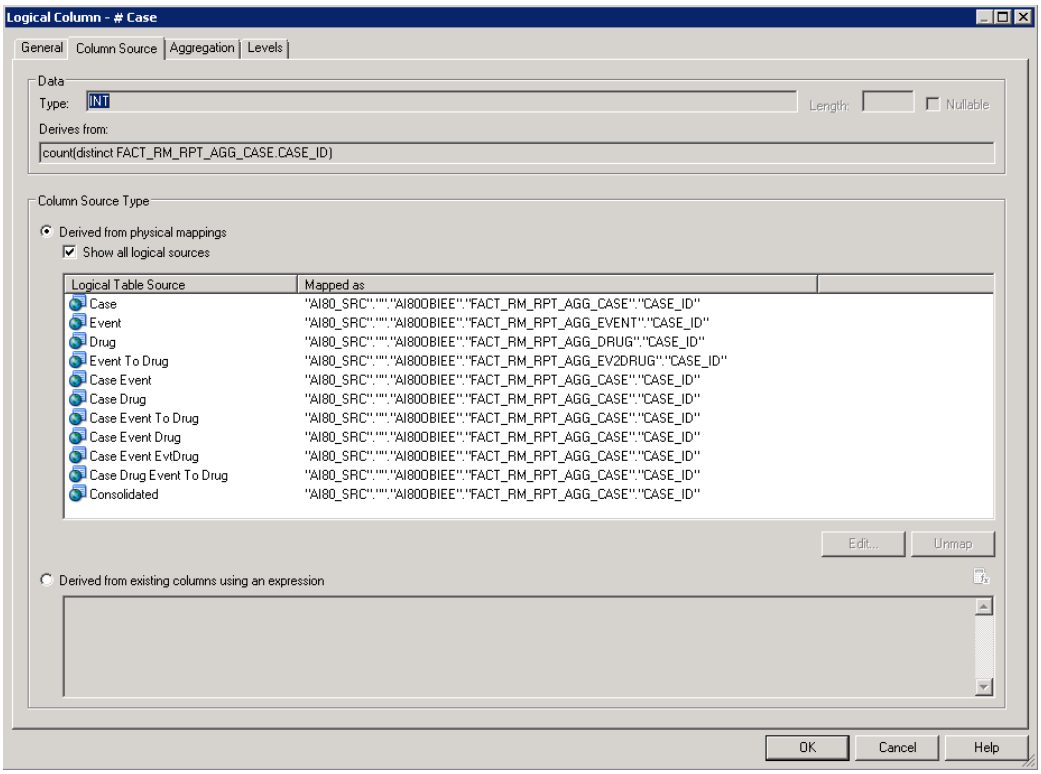
Figure 5–5 Business Model Diagram



Measure

In the Argus Insight RPD, only one measure Case Count is derived from the Fact tables.

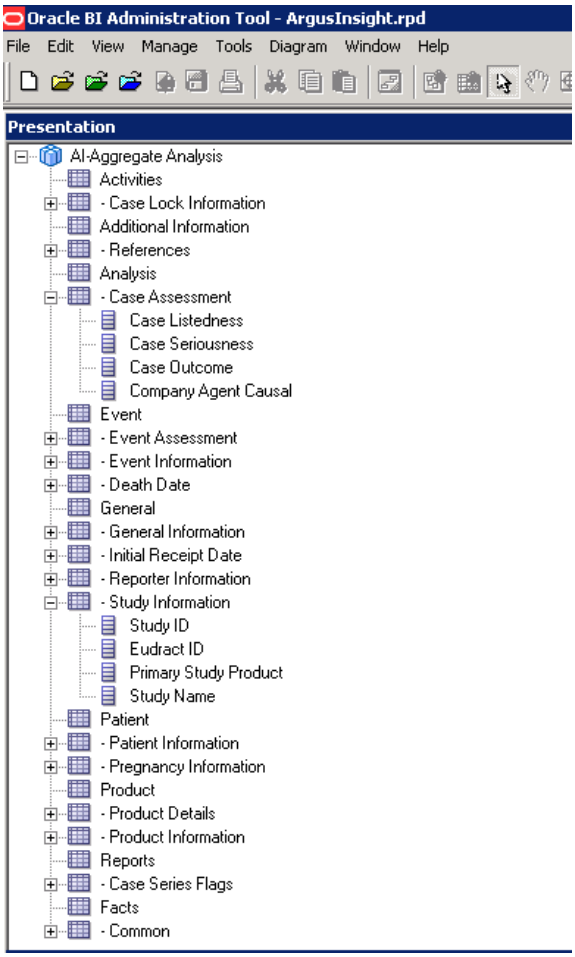
Figure 5–6 Case Count Measure Properties



5.2.2.3 Presentation Layer

The dimensions created are renamed and arranged in a tree view in the presentation layer.

Figure 5–7 Presentation Layer Tree View



5.2.3 Add New Dimension Using Flex Bucketing

Note: In the Argus Safety Aggregate Reporting Data Model, you may update any column value. For more information, refer to *Oracle Argus Safety BIP Extensibility Guide > Section 6.2.2 Extending with User Exits*.

For Example:

PROLONGED EXPOSURE column which exists in the OBIEE RPD can be updated in the Aggregate Reporting Data Model and it can be used for analysis in the OBIEE Answers/Dashboards.

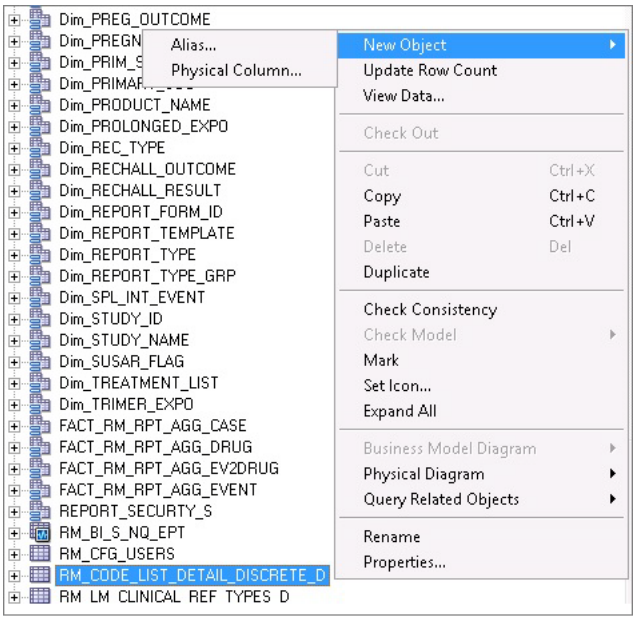
New dimensions can be created on the existing RPD.

The following are the steps of creating a dimension from the source RM_CODE_LIST_DETAIL_DISCRETE_D, explained with the help of an example:

1. Open the Argus Insight RPD using the default password (insight 123), or the password changed using the steps mentioned in the *Oracle Argus Insight Installation Guide*.

2. At the Physical Layer, right-click on RM_CODE_LIST_DETAIL_DISCRETE_D, and create an alias.

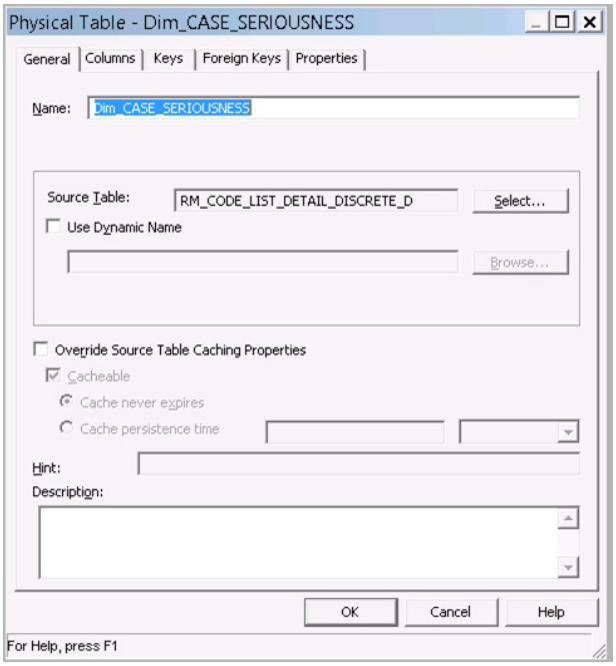
Figure 5–8 Physical Layer — Creating Alias



3. Enter the dimension name for the alias.

   For example: Dim_CASE_SERIOUSNESS

Figure 5–9 Dimension Properties



4. Create join with the corresponding Fact table in this case FACT_RM_RPT_AGG_CASE, as shown below:

Figure 5–10 Join with the Fact table

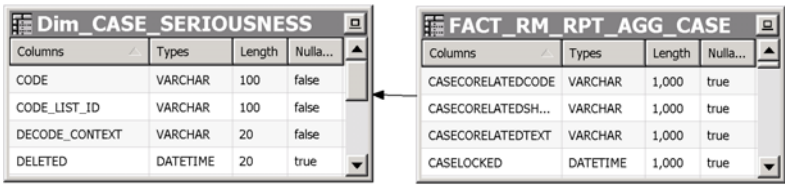
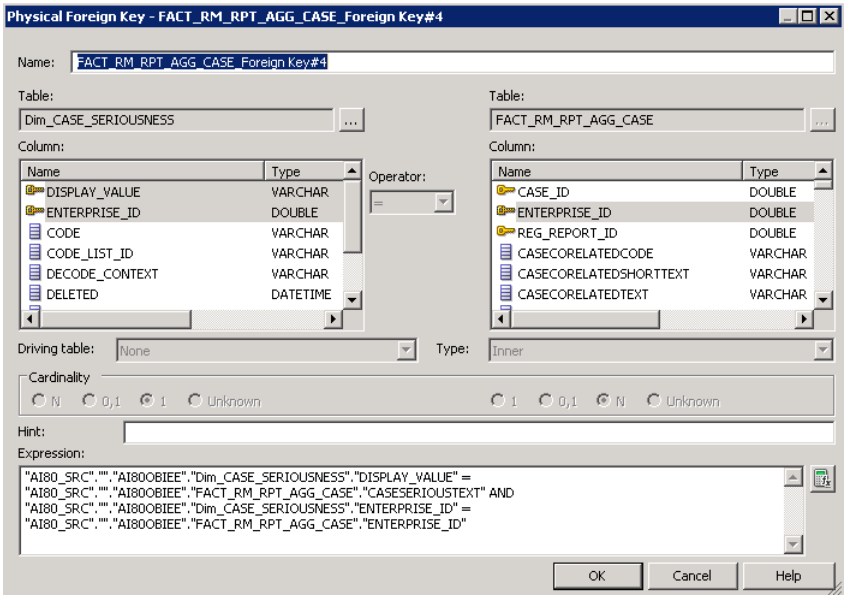
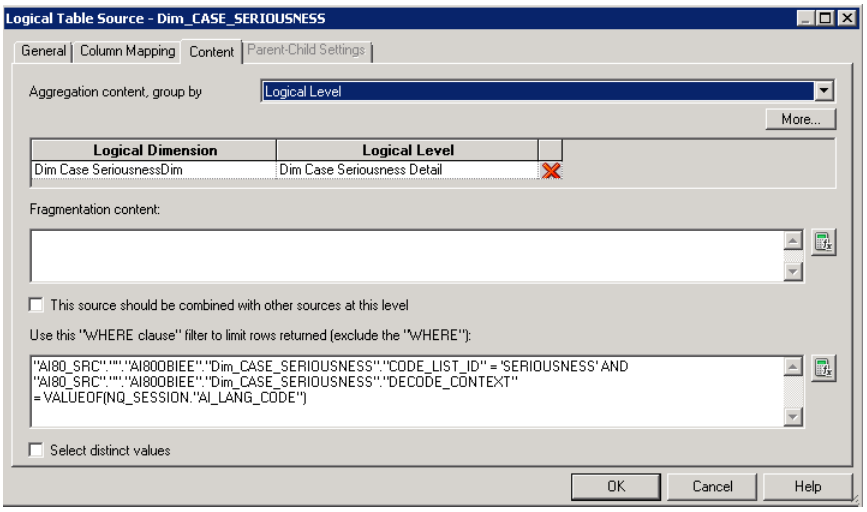


Figure 5–11 Join Definition



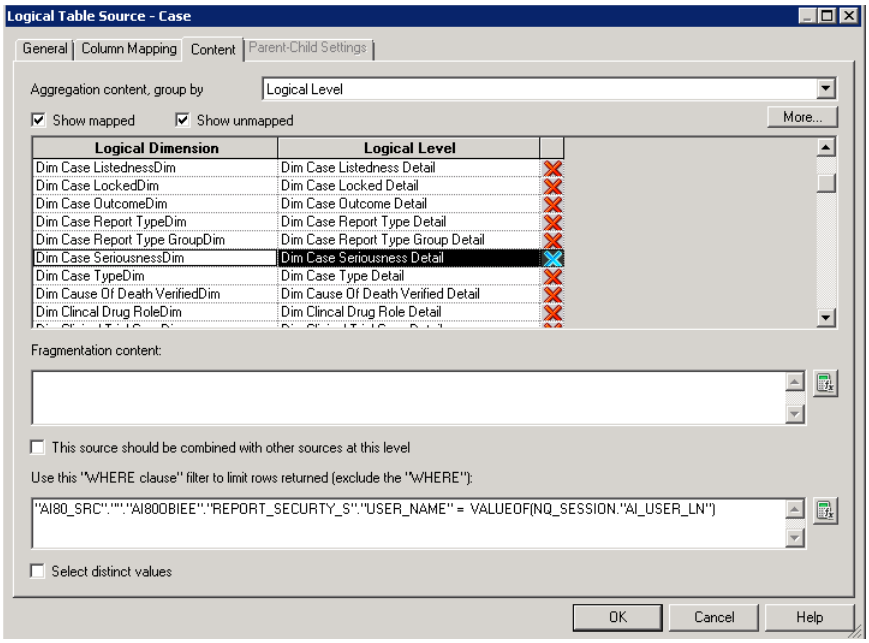
5. Drag this dimension into the Business Layer and set the WHERE clause at the business layer.

Figure 5–12 Business Layer — WHERE clause



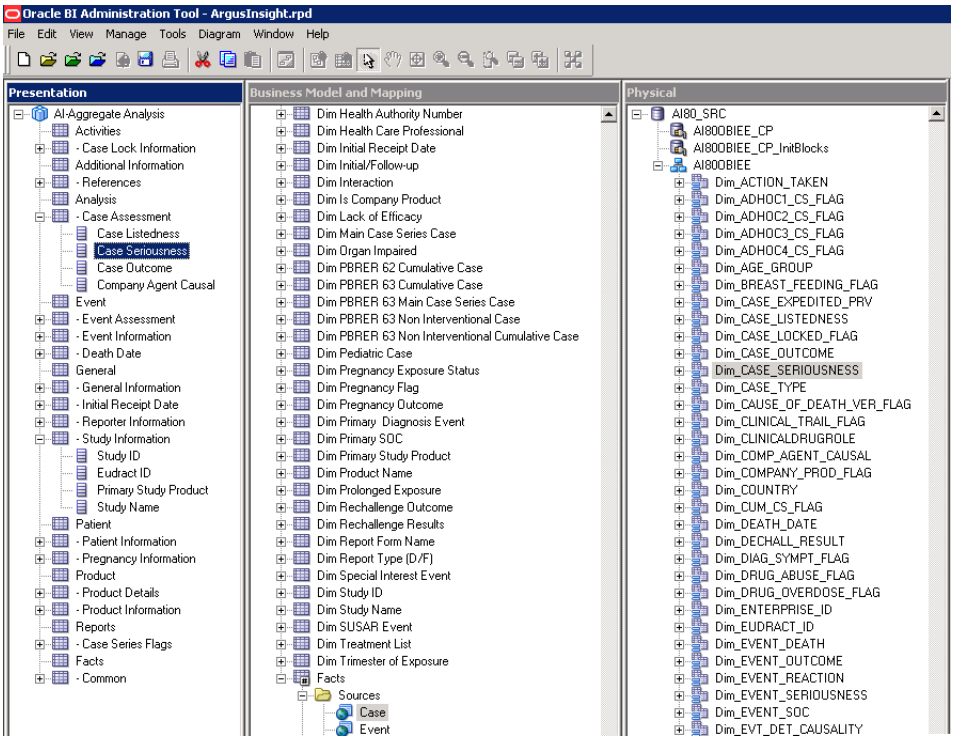
6. Right-click and create a logical dimension.
7. Go to Facts > Sources, and add the dimension to the corresponding logical table source.

Figure 5–13 Logical Table Source



8. Drag the dimension to the presentation layer in the corresponding tree level.

Figure 5–14 RPD — Presentation Layer



9. When the RPD is deployed, the new dimension can be used in the BI Answers/Dashboards.

5.2.4 Create Custom Dashboards and Prompts

Refer to *Oracle Business Intelligence Enterprise Edition > Fusion Middleware User's Guide,* available in Oracle Technology Network.

A

Appendix: Dimensions and their Mapping

The following table lists the details of RPD including dimensions, Fact tables and their joins:

Table 5–5 Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_ACTION_TAKEN	Product > Product Information > Action Taken	"Dim_ACTION_TAKEN"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_DRUG"."ACTIONDRUG" AND "Dim_ACTION_TAKEN"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID"	Code_list_id = 'ACTION_TAKEN' and decode_context = <lang_code>	RM_RPT_AGG_DRUG	ACTIONDRUG
Dim_ADHOC1_CS_FLAG	Reports > Case Series Flags > Adhoc Line Listing 1	"Dim_ADHOC1_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9ADHOC1FLAG" AND "Dim_ADHOC1_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9ADHOC1FLAG
Dim_ADHOC2_CS_FLAG	Reports > Case Series Flags > Adhoc Line Listing 2	"Dim_ADHOC2_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9ADHOC2FLAG" AND "Dim_ADHOC2_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9ADHOC2FLAG
Dim_ADHOC3_CS_FLAG	Reports > Case Series Flags > Adhoc Line Listing 3	"Dim_ADHOC3_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9ADHOC3FLAG" AND "Dim_ADHOC3_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9ADHOC3FLAG

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_ADHOC4_CS_FLAG	Reports > Case Series Flags > Adhoc Line Listing 4	"Dim_ADHOC4_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9ADHOC4FLAG" AND "Dim_ADHOC4_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9ADHOC4FLAG
Dim_AGE_GROUP	Patient > Patient Information > Age Group	" Dim_AGE_GROUP""."DISPLAY_VALUE"" = FACT_RM_RPT_AGG_CASE""."PATIENTAGEGROUPTEXT"" AND Dim_AGE_GROUP""."ENTERPRISE_ID"" = ""FACT_RM_RPT_AGG_CASE""."ENTERPRISE_ID"""	"Code_list_id = 'AGE_GROUPS' and decode_context = <lang_code> "	RM_RPT_AGG_CASE	PATIENTAGEGROUPTEXT
Dim_BREAST_FEEDING_FLAG	Patient > Patient Information > Breastfeeding	"Dim_BREAST_FEEDING_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9LACTATIONFLAG" AND "Dim_BREAST_FEEDING_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9LACTATIONFLAG
Dim_CASE_EXPEDITED_PRV	Reports > Case Series Flags > Case Expedited Previously	"Dim_CASE_EXPEDITED_PRV"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."EXPEDITEDFLAG" AND "Dim_CASE_EXPEDITED_PRV"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	EXPEDITEDFLAG
Dim_CASE_LISTEDNESS	Analysis > Case Assessment > Case Listedness	"Dim_CASE_LISTEDNESS"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CASEUNLABELEDNESSTEXT" AND "Dim_CASE_LISTEDNESS"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'LISTEDNESS' and decode_context = <lang_code> "	RM_RPT_AGG_CASE	CASEUNLABELEDNESSTEXT
Dim_CASE_LOCKED_FLAG	Activities > Case Lock Information > Case Locked	"Dim_CASE_LOCKED_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CASE_LOCKED_FLAG" AND "Dim_CASE_LOCKED_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	CASE_LOCKED_FLAG

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_CASE_OUTCOME	Analysis > Case Assessment > Case Outcome	"Dim_CASE_OUTCOME"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."OUTCOMETEXT" AND "Dim_CASE_OUTCOME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'EVENT_OUTCOME' and decode_context = <lang_code>	RM_RPT_AGG_CASE	OUTCOMETEXT
Dim_CASE_SERIOUSNESS	Analysis > Case Assessment > Case Seriousness	Dim_CASE_SERIOUSNESS."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CASESERIOUSTEXT" AND "Dim_CASE_SERIOUSNESS"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'SERIOUSNESS' and decode_context = <lang_code>	RM_RPT_AGG_CASE	CASESERIOUSTEXT
Dim_CASE_TYPE	General > General Information > Case Type	"Dim_CASE_TYPE. ENTERPRISE_ID = FACT_RM_RPT_AGG_CASE.ENTERPRISE_ID AND Dim_CASE_TYPE. DISPLAY_VALUE = FACT_RM_RPT_AGG_CASE.CASETYPETEXT "	Code_list_id = 'REPORT_TYPE' and decode_context = 'CASETYPETEXT'	RM_RPT_AGG_CASE	CASETYPETEXT
Dim_CAUSE_OF_DEATH_VER_FLAG	Event > Event Information > Cause of Death Verified	"Dim_CAUSE_OF_DEATH_VER_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CAUSEOFDEATHVERIFIED" AND "Dim_CAUSE_OF_DEATH_VER_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	CAUSEOFDEATHVERIFIED
Dim_CLINICAL_TRAIL_FLAG	Reports > Case Series Flags > Clinical Trial Case	"Dim_CLINICAL_TRAIL_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC7FLAG" AND "Dim_CLINICAL_TRAIL_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC7FLAG
Dim_CLINICALDRUGROLE	Event > Event Information > Clinical Drug Role	"Dim_CLINICALDRUGROLE"."DRUG_ROLE_NUM" = "FACT_RM_RPT_AGG_CASE"."CLINICALDRUGROLE" AND "Dim_CLINICALDRUGROLE"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	N/A	RM_RPT_AGG_CASE	CLINICALDRUGROLE

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_COMP_AGENT_CAUSAL	Analysis > Case Assessment > Company Agent Causal	"Dim_COMP_AGENT_CAUSAL"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CASECORELATEDTEXT" AND "Dim_COMP_AGENT_CAUSAL"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_3' and decode_context = <lang_code> "	RM_RPT_AGG_CASE	CASECORELATEDTEXT
Dim_COMPANY_PROD_FLAG	Product > Product Information > Is Company Product	"Dim_COMPANY_PROD_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_DRUG"."COMPANYDRUGFLAG" AND "Dim_COMPANY_PROD_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_DRUG	COMPANYDRUGFLAG
Dim_COUNTRY	General > General Information > Country of Incidence	Dim_COUNTRY"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."OCCURCOUNTRYTEXT" AND "Dim_COUNTRY"."ENTERPRISE_ID" = .FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID	"Code_list_id = 'COUNTRY' and decode_context = <lang_code> "	RM_RPT_AGG_CASE	OCCURCOUNTRYTEXT
Dim_CUM_CS_FLAG	Reports > Case Series Flags > Cumulative Case	"Dim_CUM_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC6CUMMFLAG" AND "Dim_CUM_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC6CUMMFLAG
Dim_DEATH_DATE	Event > Event Information > Death Date	"Dim_DEATH_DATE"."ROW_WID" = "FACT_RM_RPT_AGG_CASE"."PATIENTDEATHDATE_WID"	N/A	RM_RPT_AGG_CASE	PATIENTDEATHDATE_WID
Dim_DECHALL_RESULT	Product > Product Details > Dechallenge Results	"Dim_DECHALL_RESULT"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_DRUG"."DECHALLENGETEXT" AND "Dim_DECHALL_RESULT"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID"	"Code_list_id = 'STATE_POS_NEG' and decode_context = <lang_code> "	RM_RPT_AGG_DRUG	DECHALLENGETEXT
Dim_DIAG_SYMPT_FLAG	Event > Event Information > Diagnosis/Symptoms	"Dim_DIAG_SYMPT_FLAG"."DIAG_SYMPT_FLAG" = "FACT_RM_RPT_AGG_EVENT"."TERMTYPEFLAG" AND "Dim_DIAG_SYMPT_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	N/A	RM_RPT_AGG_EVENT	TERMTYPEFLAG

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_DRUG_ABUSE_FLAG	Product > Product Information > Drug Abuse	"Dim_DRUG_ABUSE_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9DRUGABUSEFLAG" AND "Dim_DRUG_ABUSE_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9DRUGABUSEFLAG
Dim_DRUG_OVERDOSE_FLAG	Product > Product Information > Drug Overdose	"Dim_DRUG_OVERDOSE_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9OVERDOSEFLAG" AND "Dim_DRUG_OVERDOSE_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9OVERDOSEFLAG
Dim_ENTERPRISE_ID	Facts > Common > Enterprise_Id	"""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_ENTERPRISE_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_CASE"".""ENTERPRISE_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_ENTERPRISE_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EVENT"".""ENTERPRISE_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_ENTERPRISE_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_DRUG"".""ENTERPRISE_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_ENTERPRISE_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EV2DRUG"".""ENTERPRISE_ID"""	N/A		
Dim_EUDRACT_ID	General > Study Information > EUDRACT ID	"Dim_EUDRACT_ID"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_EUDRACT_ID"."REF_TYPE_DESC" = "FACT_RM_RPT_AGG_CASE"."EUDRACTID"	ref_type_id = 4 and deleted IS NULL	RM_RPT_AGG_CASE	EUDRACTID

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_EVENT_DEATH	Event > Event Information > Event Death	"Dim_EVENT_DEATH"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."DIEDFLAG" AND "Dim_EVENT_DEATH"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_EVENT	DIEDFLAG
Dim_EVENT_LISTEDNESS	Event > Event Assessment > Event Listedness	"Dim_EVENT_LISTEDNESS"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTUNLABELEDNESSTEXT" AND "Dim_EVENT_LISTEDNESS"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'LISTEDNESS' and decode_context = <lang_code> "	RM_RPT_AGG_EVENT	EVENTUNLABELEDNESSTEXT
Dim_EVENT_OUTCOME	Event > Event Information > Event Outcome	"Dim_EVENT_OUTCOME"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTOUTCOMELIST" AND "Dim_EVENT_OUTCOME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	Code_list_id = 'EVENT_OUTCOME' and decode_context = <lang_code>	RM_RPT_AGG_EVENT	EVENTOUTCOMELIST
Dim_EVENT_REACTION	Event > Event Information > Event Reported	"Dim_EVENT_REACTION"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID" AND "Dim_EVENT_REACTION"."REACTION" = "FACT_RM_RPT_AGG_EVENT"."REACTION"	N/A	RM_RPT_AGG_EVENT	REACTION
Dim_EVENT_SERIOUSNESS	Event > Event Information > Event Seriousness	"Dim_EVENT_SERIOUSNESS"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTSERIOUSTEXT" AND "Dim_EVENT_SERIOUSNESS"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	Code_list_id = 'SERIOUSNESS' and decode_context = <lang_code>	RM_RPT_AGG_EVENT	EVENTSERIOUSTEXT
Dim_EVENT_SOC	Event > Event Information > Event SOC	"Dim_EVENT_SOC"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."SOC" AND "Dim_EVENT_SOC"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'SOC_DISPLAY_ORDER' and decode_context = 'SOC' "	RM_RPT_AGG_EVENT	SOC

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_EVT_DET_CAUSALITY	Event > Event Assessment > As Determined Causality	"Dim_EVT_DET_CAUSALITY"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTCORELATEDTEXT" AND "Dim_EVT_DET_CAUSALITY"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'CAUSAL' "	RM_RPT_AGG_EVENT	EVENTCORELATEDTEXT
Dim_EVT_PRIM_DIAG	Event > Event Information > Primary Diagnosis Event	"Dim_EVT_PRIM_DIAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."PRIMARYDIAGNOSISFLAG" AND "Dim_EVT_PRIM_DIAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_EVENT	PRIMARYDIAGNOSISFLAG
Dim_EVT_RPT_CAUSALITY	Event > Event Assessment > As Reported Causality	"Dim_EVT_RPT_CAUSALITY"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTRPTRELATEDTEXT" AND "Dim_EVT_RPT_CAUSALITY"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'CAUSALITY' and decode_context = <lang_code> "	RM_RPT_AGG_EVENT	EVENTRPTRELATEDTEXT
Dim_FATAL_LIST_FLAG	Reports > Case Series Flags > Fatal Listing Case	"Dim_FATAL_LIST_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC6FATALFLAG" AND "Dim_FATAL_LIST_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC6FATALFLAG
Dim_FOLLOWUP	Reports > Case Series Flags > Initial/Follow-up	"Dim_FOLLOWUP"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PSURFOLLOWUPTEXT" AND "Dim_FOLLOWUP"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'STATE_2' and decode_context = 'FOLLOWUPTEXT'	RM_RPT_AGG_CASE	PSURFOLLOWUPTEXT
Dim_Gender	Patient > Patient Information > Gender	Dim_GENDER"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PATIENTSEXTEXT" AND "Dim_GENDER"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'GENDER' and Decode_context = <lang_code> "	RM_RPT_AGG_CASE	PATIENTSEXTEXT

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_GERIATRIC_CASE_FLAG	Reports > Case Series Flags > Geriatric Case	"Dim_GERIATRIC_CASE_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9SPLGERIATICFLAG" AND "Dim_GERIATRIC_CASE_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9SPLGERIATICFLAG
Dim_HEALTH_AUTH_NUM	Additional Information > References > Health Authority Number	"Dim_HEALTH_AUTH_NUM"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_HEALTH_AUTH_NUM"."TYPE_DESC" = "FACT_RM_RPT_AGG_CASE"."HEALTHAUTHORITYNBRLIST"	Deleted IS NULL	RM_RPT_AGG_CASE	HEALTHAUTHORITYNBRLIST
Dim_HEALTH_CARE_PROF	General > Reporter Information > Health Care Professional	"Dim_HEALTH_CARE_PROF"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."CASEMEDICALLYCONFIRMFLAG" AND "Dim_HEALTH_CARE_PROF"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	CASEMEDICALLYCONFIRMFLAG
Dim_INIT_RCPT_DATE	General > General Information > Initial Receipt Date	"Dim_INIT_RCPT_DATE"."ROW_WID" = "FACT_RM_RPT_AGG_CASE"."INITRCPTDATE_WID"	N/A	RM_RPT_AGG_CASE	INITRCPTDATE_WID
Dim_INTERACTION_FLAG	Product > Product Information > Interaction	"Dim_INTERACTION_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9INTERACTIONSFLAG" AND "Dim_INTERACTION_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9INTERACTIONSFLAG
Dim_LACK_EFFICACY_FLAG	Product > Product Information > Lack of Efficacy	"Dim_LACK_EFFICACY_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC8LACKOFEFFICACYFLAG" AND "Dim_LACK_EFFICACY_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC8LACKOFEFFICACYFLAG
Dim_MAIN_CS_FLAG	Reports > Case Series Flags > Main Case Series Case	"Dim_MAIN_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC61FLAG" AND "Dim_MAIN_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC61FLAG

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_ORGAN_IMPAIRED_FLAG	Reports > Case Series Flags > Organ Impaired	"Dim_ORGAN_IMPAIRED_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9SPLIMPAIREDFLAG" AND "Dim_ORGAN_IMPAIRED_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9SPLIMPAIREDFLAG
Dim_PBRER62_CUM_CS_FLAG	Reports > Case Series Flags > PBRER 62 Cumulative Case	"Dim_PBRER62_CUM_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC62CUMFLAG" AND "Dim_PBRER62_CUM_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC62CUMFLAG
Dim_PBRER63_CUM_CS_FLAG	Reports > Case Series Flags > PBRER 63 Cumulative Case	"Dim_PBRER63_CUM_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC63CUMFLAG" AND "Dim_PBRER63_CUM_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC63CUMFLAG
Dim_PBRER63_MAIN_CS_FLAG	Reports > Case Series Flags > PBRER 63 Main Case Series Case	"Dim_PBRER63_MAIN_CS_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC63MAINFLAG" AND "Dim_PBRER63_MAIN_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC63MAINFLAG
Dim_PBRER63_NONINT_CS_FLAG	Reports > Case Series Flags > PBRER 63 Non Interventional Case	"Dim_PBRER63_NONINT_CS_FLAG"."DISPLAY_VALUE" ="FACT_RM_RPT_AGG_CASE"."SEC63NONINTMAINFLAG" AND "Dim_PBRER63_NONINT_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC63NONINTMAINFLAG
Dim_PBRER63_NONINT_MAIN_CS_FLAG	Reports > Case Series Flags > PBRER 63 Non Interventional Cumulative Case	"Dim_PBRER63_NONINT_CUM_CS_FLAG"."DISPLAY_VALUE" ="FACT_RM_RPT_AGG_CASE"."SEC63NONINTCUMFLAG" AND "Dim_PBRER63_NONINT_CUM_CS_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC63NONINTCUMFLAG

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_PEDIATRIC_CASE_FLAG	Reports > Case Series Flags > Pediatric Case	"Dim_PEDIATRIC_CASE_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9SPLPEDFLAG" AND "Dim_PEDIATRIC_CASE_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9SPLPEDFLAG
Dim_PREG_EXPO	Patient > Pregnancy Information > Pregnancy Exposure Status	"Dim_PREG_EXPO"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PREGEXPOSURECASESTATUSTEXT" AND "Dim_PREG_EXPO"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id= 'PROSPECTIVE_STATUS' and decode_context = <lang_code> "	RM_RPT_AGG_CASE	PREGEXPOSURECASESTATUSTEXT
Dim_PREG_OUTCOME	Patient > Pregnancy Information > Pregnancy Outcome	"Dim_PREG_OUTCOME"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PREGNANCYOUTCOMETEXT" AND "Dim_PREG_OUTCOME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'FETAL_OUTCOME' and decode_context = <lang_code>	RM_RPT_AGG_CASE	PREGNANCYOUTCOMETEXT
Dim_PREGNANCY_FLAG	Patient > Patient Information > Pregnancy Flag	"Dim_PREGNANCY_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9PREGNANCYFLAG" AND "Dim_PREGNANCY_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9PREGNANCYFLAG
Dim_PRIM_STUDY_PROD	General > Study Information > Primary Study Product	"Dim_PRIM_STUDY_PROD"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_PRIM_STUDY_PROD"."PROJECT_DRUG" = "FACT_RM_RPT_AGG_CASE"."PROJECTDRUG"	N/A	RM_RPT_AGG_CASE	PROJECTDRUG
Dim_PRIMARY_SOC	Event > Event Information > Primary SOC	"Dim_PRIMARY_SOC"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PRIMARYCASESOC" AND "Dim_PRIMARY_SOC"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'SOC_DISPLAY_ORDER' and decode_context = 'SOC' "	RM_RPT_AGG_CASE	PRIMARYCASESOC

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_PRODUCT_NAME	Product > Product Information > Product Name	"Dim_PRODUCT_NAME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID" AND "Dim_PRODUCT_NAME"."PRODUCT_NAME" = "FACT_RM_RPT_AGG_DRUG"."DRUGNAME"	N/A	RM_RPT_AGG_DRUG	DRUGNAME
Dim_PROLONGED_EXPO	Reports > Case Series Flags > Prolonged Exposure	"Dim_PROLONGED_EXPO"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."SEC9PROLONGFLAG" AND "Dim_PROLONGED_EXPO"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_CASE	SEC9PROLONGFLAG
Dim_RECHALL_OUTCOME	Product > Product Details > Rechallenge Outcome	"Dim_RECHALL_OUTCOME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID" AND "Dim_RECHALL_OUTCOME"."CODE" = "FACT_RM_RPT_AGG_DRUG"."RECHALLENGEOUTCOME"	"Code_list_id = 'RECHALLENGE_OUTCOME' and decode_context = <lang_code> "	RM_RPT_AGG_DRUG	RECHALLENGEOUTCOME
Dim_RECHALL_RESULT	Product > Product Details > Rechallenge Results	"Dim_RECHALL_RESULT"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_DRUG"."RECHALLENGETEXT" AND "Dim_RECHALL_RESULT"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_DRUG"."ENTERPRISE_ID"	"Code_list_id = 'STATE_POS_NEG' and decode_context = <lang_code> "	RM_RPT_AGG_DRUG	RECHALLENGETEXT

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_REPORT_FORM_ID	Facts > Common > Aggregate Configuration Name	"""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EVENT"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EVENT"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_CASE"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_CASE"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_DRUG"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_DRUG"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EV2DRUG"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_FORM_ID"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EV2DRUG"".""REG_REPORT_ID"""	N/A		

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_REPORT_TEMPLATE	Facts > Common > Report Form Name	"""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EVENT"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EVENT"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_CASE"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_CASE"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_DRUG"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_DRUG"".""REG_REPORT_ID"" --- ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""ENTERPRISE_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EV2DRUG"".""ENTERPRISE_ID"" AND ""AI81_SRC"".""""."".""AI81OBIEE"".""Dim_REPORT_TEMPLATE"".""REG_REPORT_ID"" = ""AI81_SRC"".""""."".""AI81OBIEE"".""FACT_RM_RPT_AGG_EV2DRUG"".""REG_REPORT_ID"""	N/A		

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_REPORT_TYPE	General > General Information > ReportType	"Dim_REPORT_TYPE"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."REPORTTYPE" AND "Dim_REPORT_TYPE"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'REPORT_TYPE' and decode_context = 'REPTYPECODE' "	RM_RPT_AGG_CASE	REPORTTYPE
Dim_REPORT_TYPE	General > General Information > Report Type	"Dim_REPORT_TYPE"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."REPORTTYPE" AND "Dim_REPORT_TYPE"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	"Code_list_id = 'REPORT_TYPE' and decode_context = 'REPTYPECODE' "	RM_RPT_AGG_CASE	REPORTTYPE
Dim_REPORT_TYPE_GRP	General > General Information > ReportType Group	"Dim_REPORT_TYPE_GRP"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."REPORTTYPETEXT" AND "Dim_REPORT_TYPE_GRP"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'REPORT_TYPE' and decode_context = 'REPTYPEGRP'	RM_RPT_AGG_CASE	REPORTTYPETEXT
Dim_SPL_INT_EVENT	Event > Event Information > Special Interest Event	"Dim_SPL_INT_EVENT"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTSPLINTRSTSYMBOL" AND "Dim_SPL_INT_EVENT"."ENTERPRISE_ID" = "AI81_SRC".""."AI81OBIEE"."FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_EVENT	EVENTSPLINTRSTSYMBOL
Dim_STUDY_ID	General > Study Information > Study ID	"Dim_STUDY_ID"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_STUDY_ID"."STUDY_NUM" = "FACT_RM_RPT_AGG_CASE"."SPONSORSTUDYNUMB"	N/A	RM_RPT_AGG_CASE	SPONSORSTUDYNUMB
Dim_STUDY_NAME	General > Study Information > Study Name	"Dim_STUDY_NAME"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_STUDY_NAME"."STUDY_NAME" = "FACT_RM_RPT_AGG_CASE"."STUDYNAME"	N/A	RM_RPT_AGG_CASE	STUDYNAME

Table 5–5 (Cont.) Dimensions and their Mapping

Dimension	Presentation Layer Tree View	JOIN in Physical Layer	WHERE clause to be used in BMM Layer	Join Table Name	Join Column Name
Dim_SUSAR_FLAG	Event > Event Information > SUSAR Event	"Dim_SUSAR_FLAG"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_EVENT"."EVENTSUSARSYMBOL" AND "Dim_SUSAR_FLAG"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_EVENT"."ENTERPRISE_ID"	"Code_list_id = 'STATE_2' and decode_context = 'EN_ABBRV' "	RM_RPT_AGG_EVENT	EVENTSUSARSYMBOL
Dim_TREATMENT_LIST	Product > Product Information > Treatment list	"Dim_TREATMENT_LIST"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID" AND "Dim_TREATMENT_LIST"."TREATMENT_LIST" = "FACT_RM_RPT_AGG_CASE"."TREATMENTLIST"	N/A	RM_RPT_AGG_CASE	TREATMENTLIST
Dim_TRIMER_EXPO	Patient > Pregnancy Information > Trimester of Exposure	"Dim_TRIMER_EXPO"."DISPLAY_VALUE" = "FACT_RM_RPT_AGG_CASE"."PREGDRUGEXPOSURECODE" AND "Dim_TRIMER_EXPO"."ENTERPRISE_ID" = "FACT_RM_RPT_AGG_CASE"."ENTERPRISE_ID"	Code_list_id = 'TRIMESTER_STATUS' and decode_context = <lang_code>	RM_RPT_AGG_CASE	PREGDRUGEXPOSURECODE