# Creating a Repository Using the Oracle BI Administration Tool (12.2.1.0.0)

## Before You Begin

### Purpose

This tutorial covers using the Oracle Business Intelligence (BI) Administration Tool (12.2.1.0.0) to build, modify, enhance, and manage an Oracle BI repository. You learned how to import metadata from a data source, simplify and reorganize the imported metadata into a business model, and then structure the business model for presentation to users who request business intelligence information via Oracle BI user interfaces. You learn to create calculated measures, hierarchies, aggregates, and time series measures.

### Time to Complete

Approximately 3 hours.

### Background

This tutorial shows you how to build an Oracle BI metadata repository using the Oracle BI Administration Tool. You learn how to import metadata from data sources, simplify and reorganize the imported metadata into a business model, and then structure the business model for presentation to users who request business intelligence information via Oracle BI user interfaces.

### Hardware and Software Requirements

The following is a list of hardware and software requirements:

1. Have access to or have Installed Oracle Business Intelligence Enterprise Edition 12c.

   Please note: This tutorial is built using Oracle Business Intelligence Enterprise Edition 12.2.1.0.0.

   When setting up query logging, you must select Action > Set Online User Filter in Identity Manager to view users in the repository.

2. To complete this tutorial you must have access to the BISAMPLE schema that is included with the Sample Application for Oracle Business Intelligence Suite Enterprise Edition Plus. There are two options for accessing the BISAMPLE schema:

   1. You can download the Sample Application virtual box image from here (http://www.oracle.com/technetwork/middleware/bi-foundation/obiee-samples-167534.html).
   2. Click here (files/ForOBE.7z) to access the ForOBE.7z file, which has the BISAMPLE schema. Save

ForOBE.7z, unzip, and begin with README.txt.

### What Do You Need?

Before starting this tutorial, you should:

- Have some familiarity with the Oracle BI 12c Administration Tool
- Have the proper permissions to upload a repository

## Building the Physical Layer of a Repository

In this topic you use the Oracle BI Administration Tool to build the Physical layer of a repository.

The Physical layer defines the data sources to which Oracle BI Server submits queries and the relationships between physical databases and other data sources that are used to process multiple data source queries. The recommended way to populate the Physical layer is by importing metadata from databases and other data sources. The data sources can be of the same or different varieties. You can import schemas or portions of schemas from existing data sources. Additionally, you can create objects in the Physical layer manually.

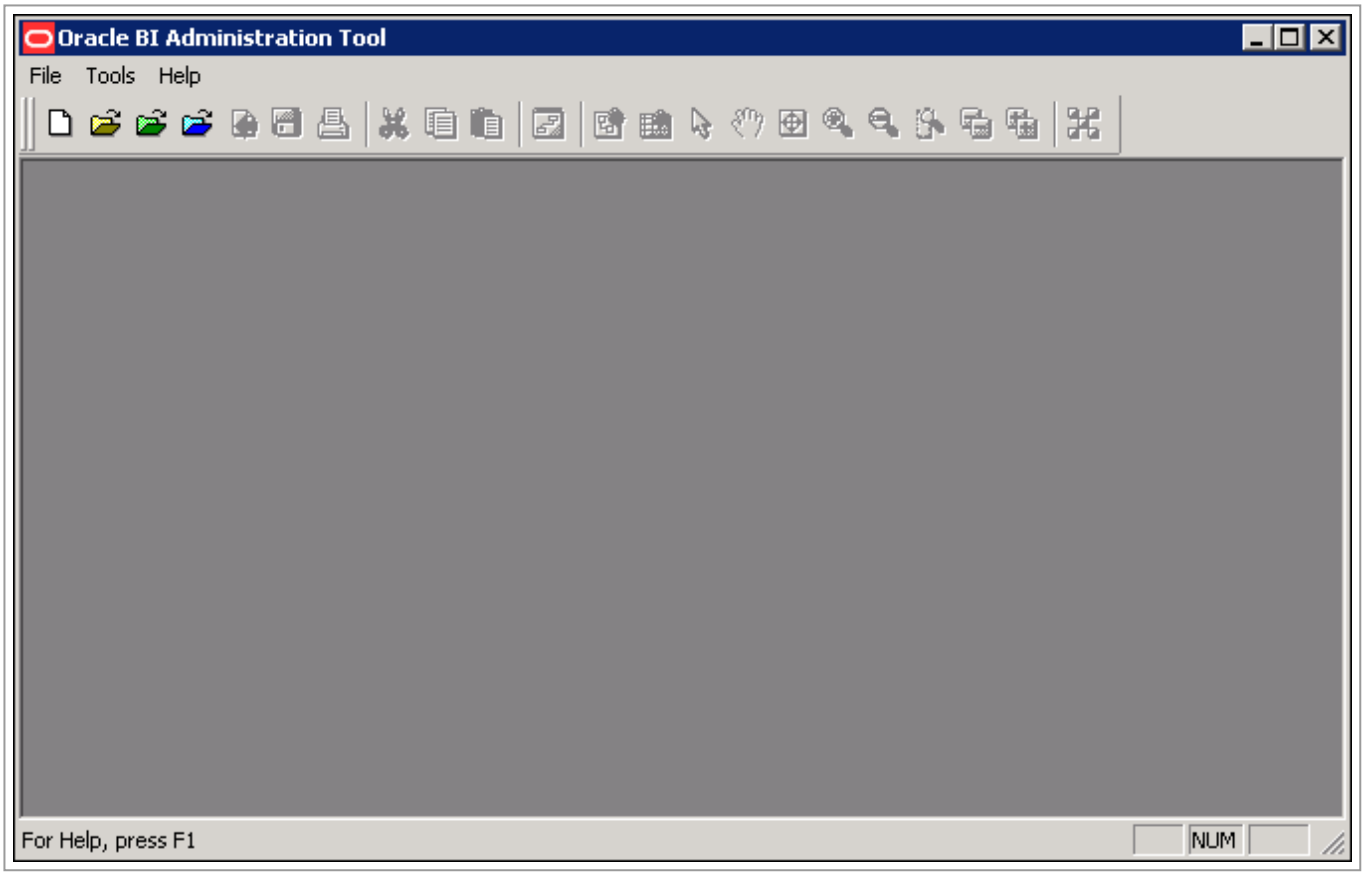
When you import metadata, many of the properties of the data sources are configured automatically based on the information gathered during the import process. After import, you can also define other attributes of the physical data sources, such as join relationships, that might not exist in the data source metadata. There can be one or more data sources in the Physical layer, including databases, flat files, XML documents, and so forth. In this example, you import and configure tables from the BISAMPLE schema.

To build the Physical layer of a repository, you perform the following steps:

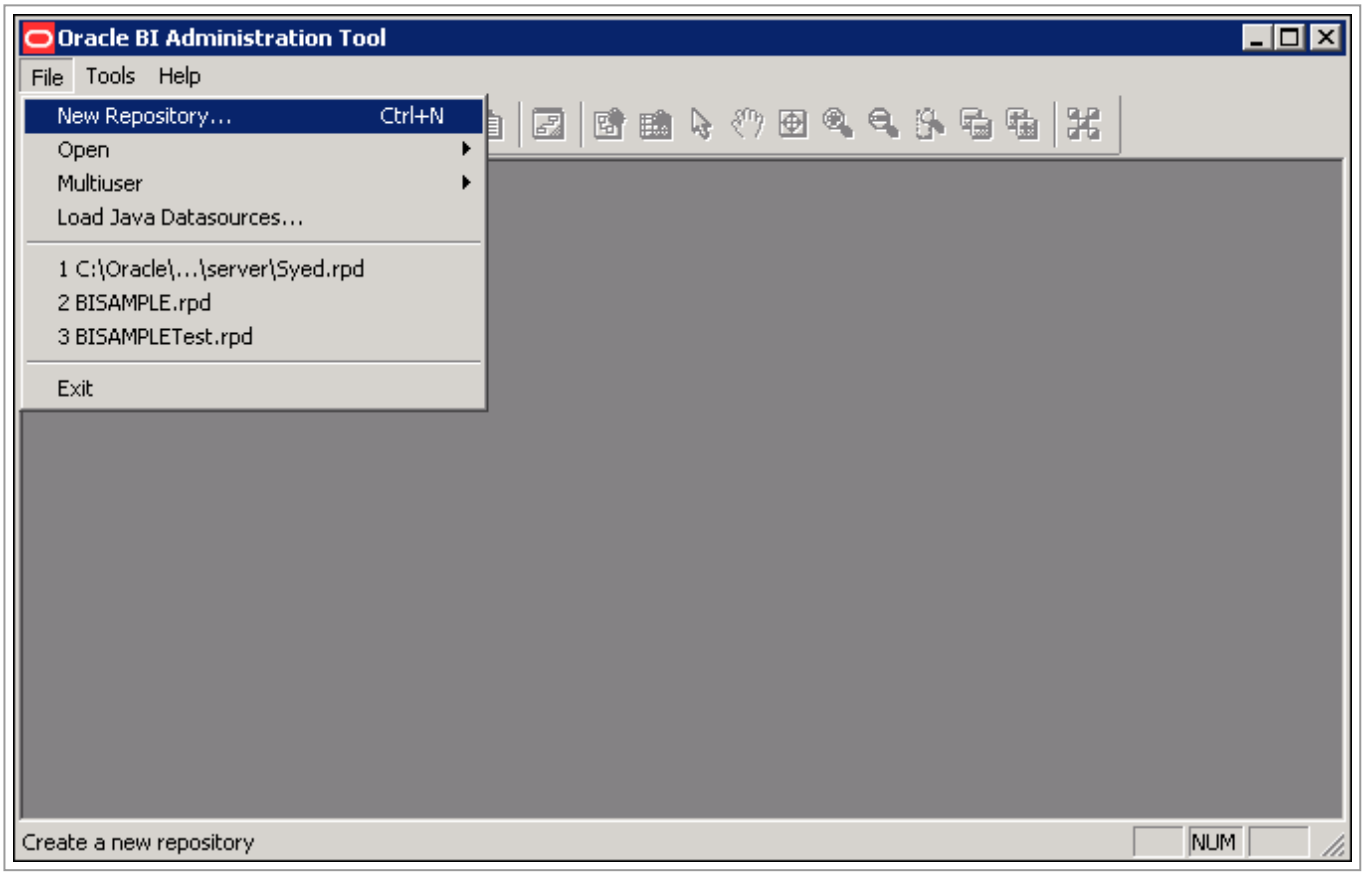
1. Creating a New Repository
2. Importing Metadata
3. Verifying Connection
4. Creating Aliases
5. Creating Physical Keys and Joins

### Creating a New Repository

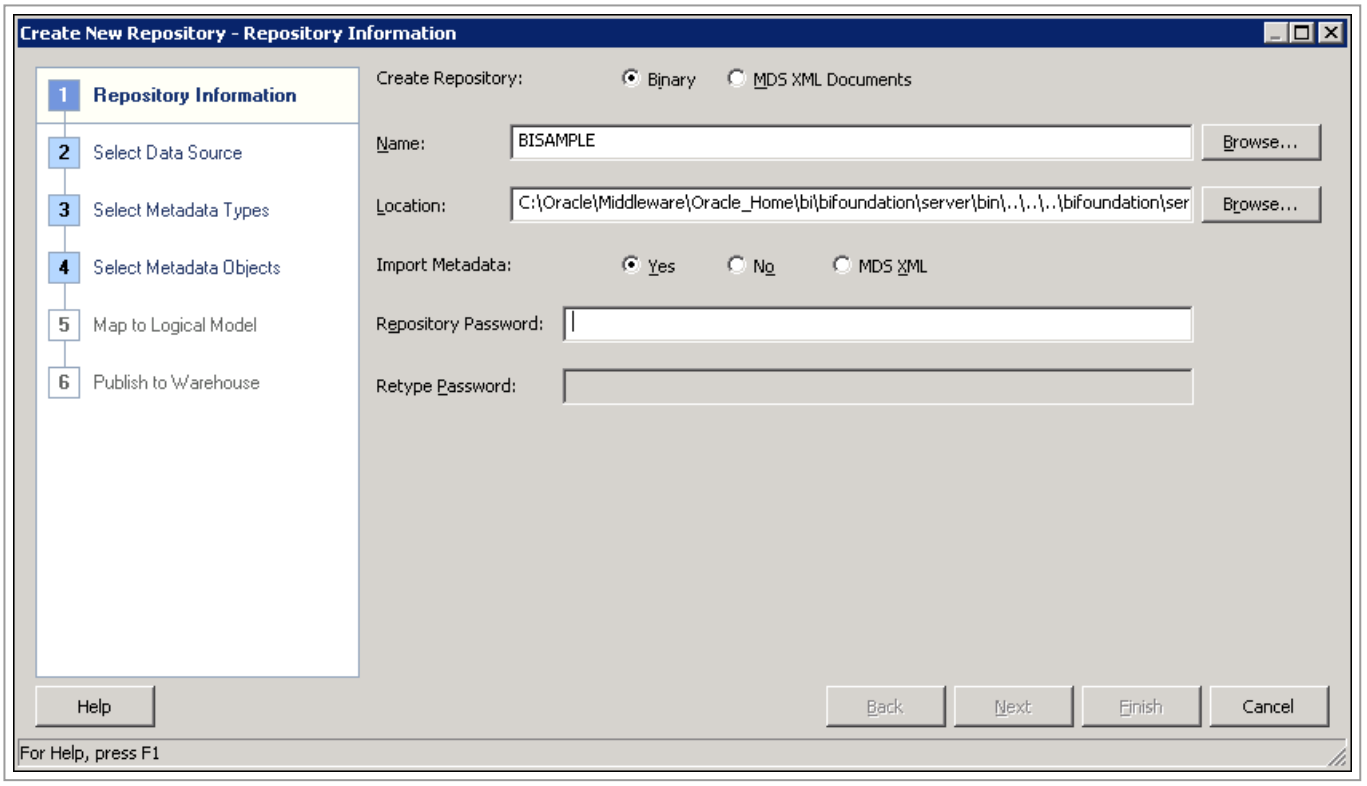
1. Open the Administration Tool.




2. Select **File > New Repository**.



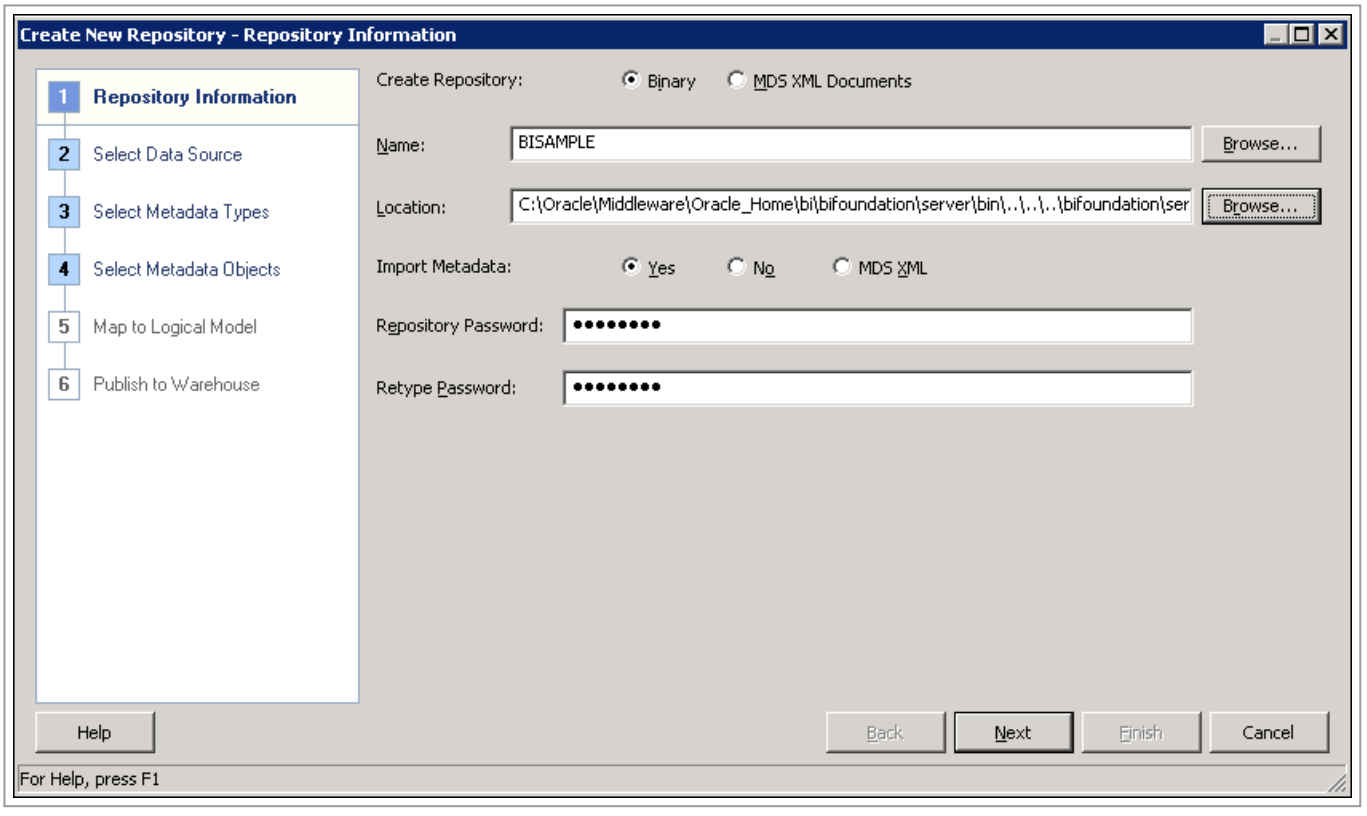

3. Select the **Binary** method and enter a name for the repository. In this tutorial the repository name is **BISAMPLE**.




**4.** Leave the default location as is. It points to the default repository directory.

**5.** Leave Import Metadata set to **Yes**.

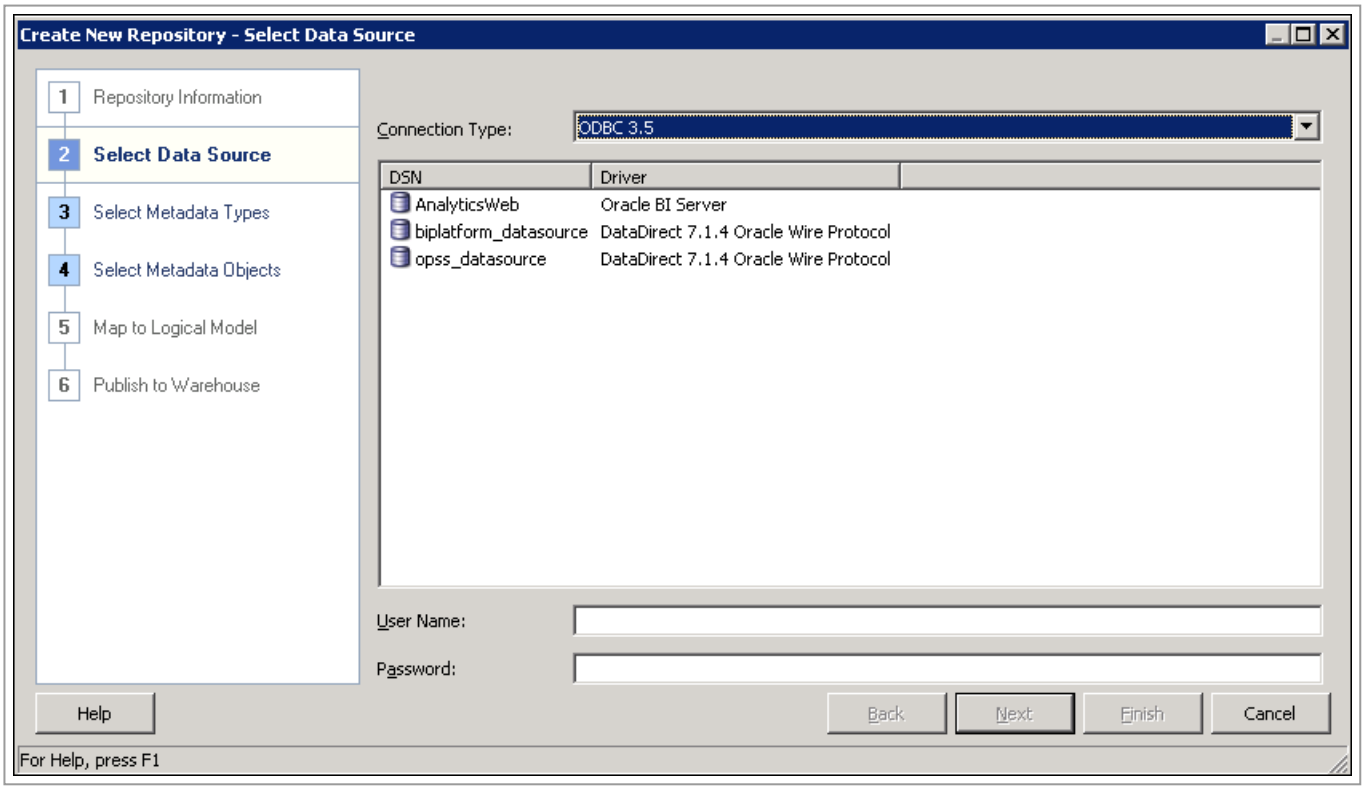
**6.** Enter and retype a password for the repository. In this tutorial **Admin123** is the repository password.



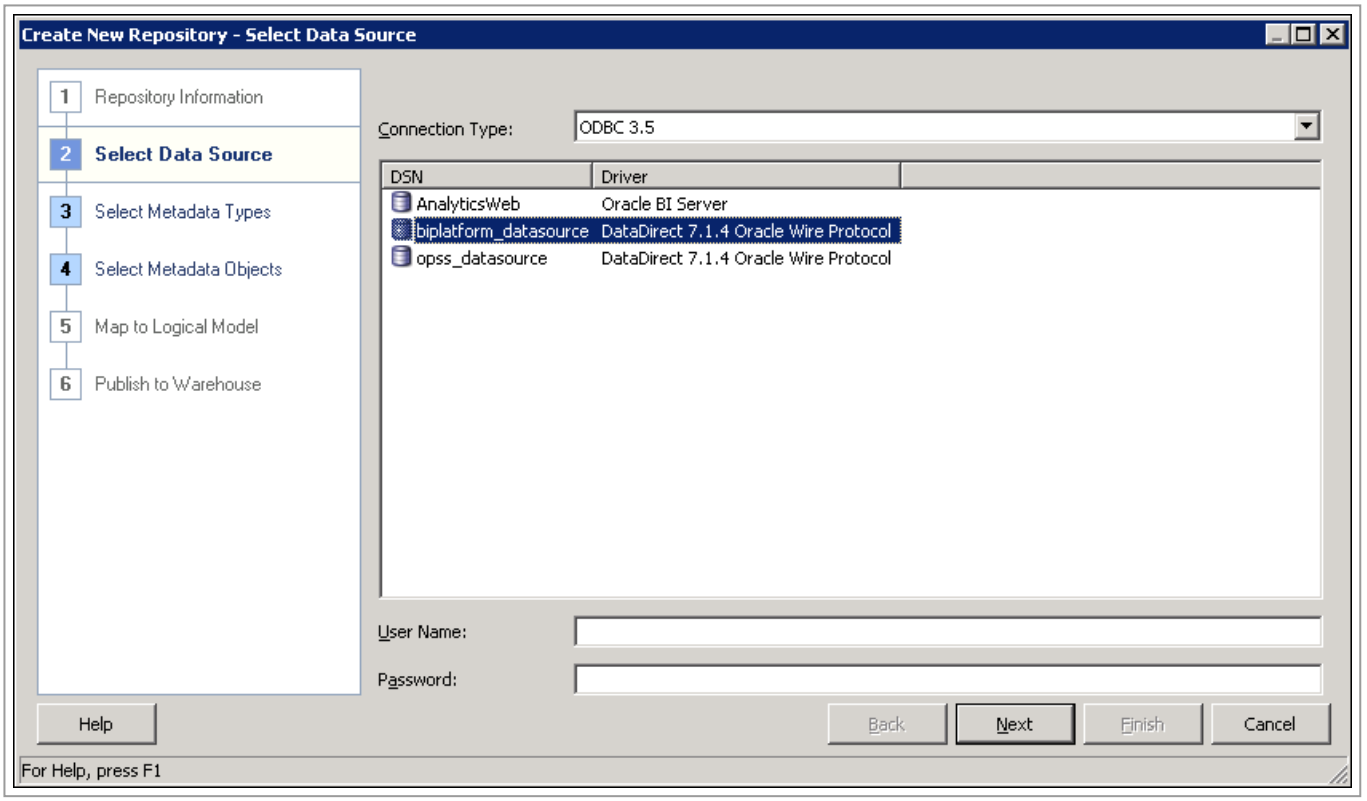

**7.** Click **Next**.

## Importing Metadata

1. Change the Connection Type to **ODBC 3.5**. The screen displays connection fields based on the connection type you selected.

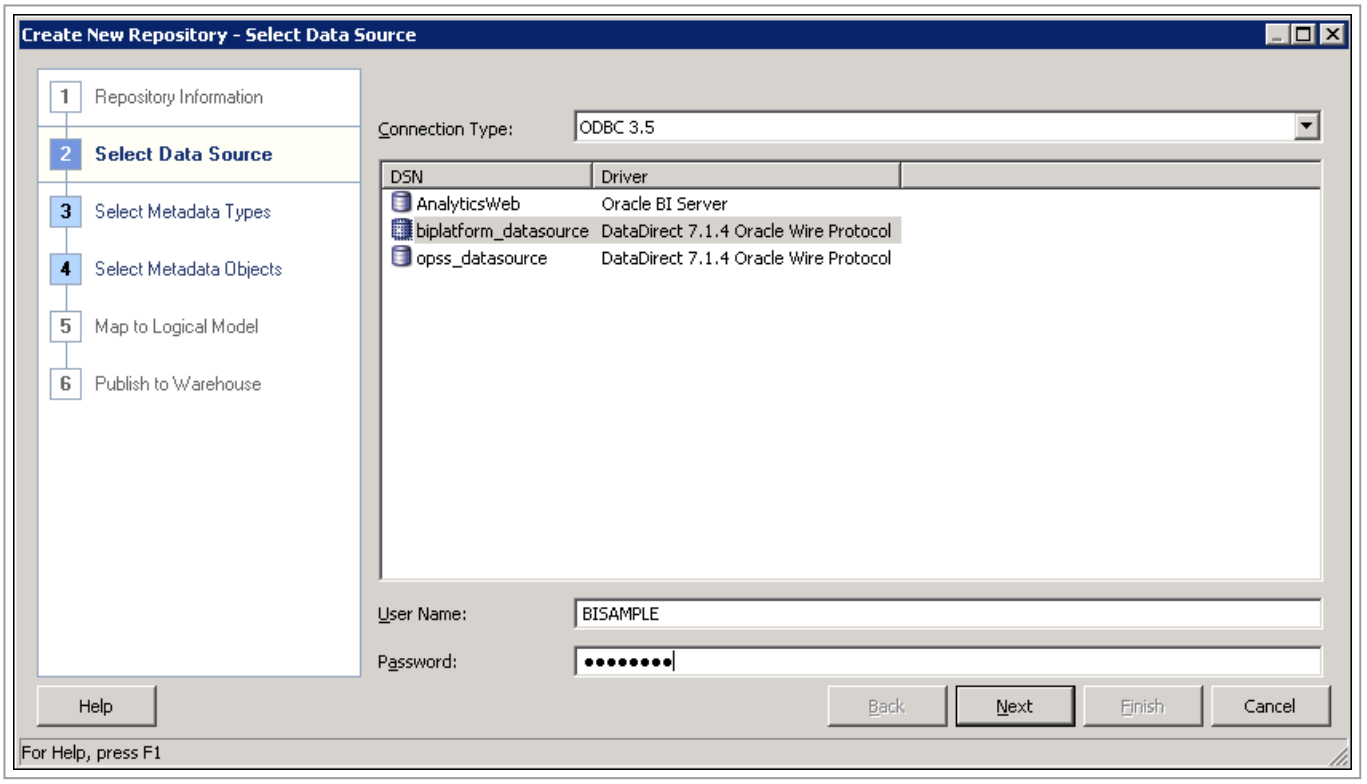



2. Select a data source. In this example the data source name is biplatform_datasource.



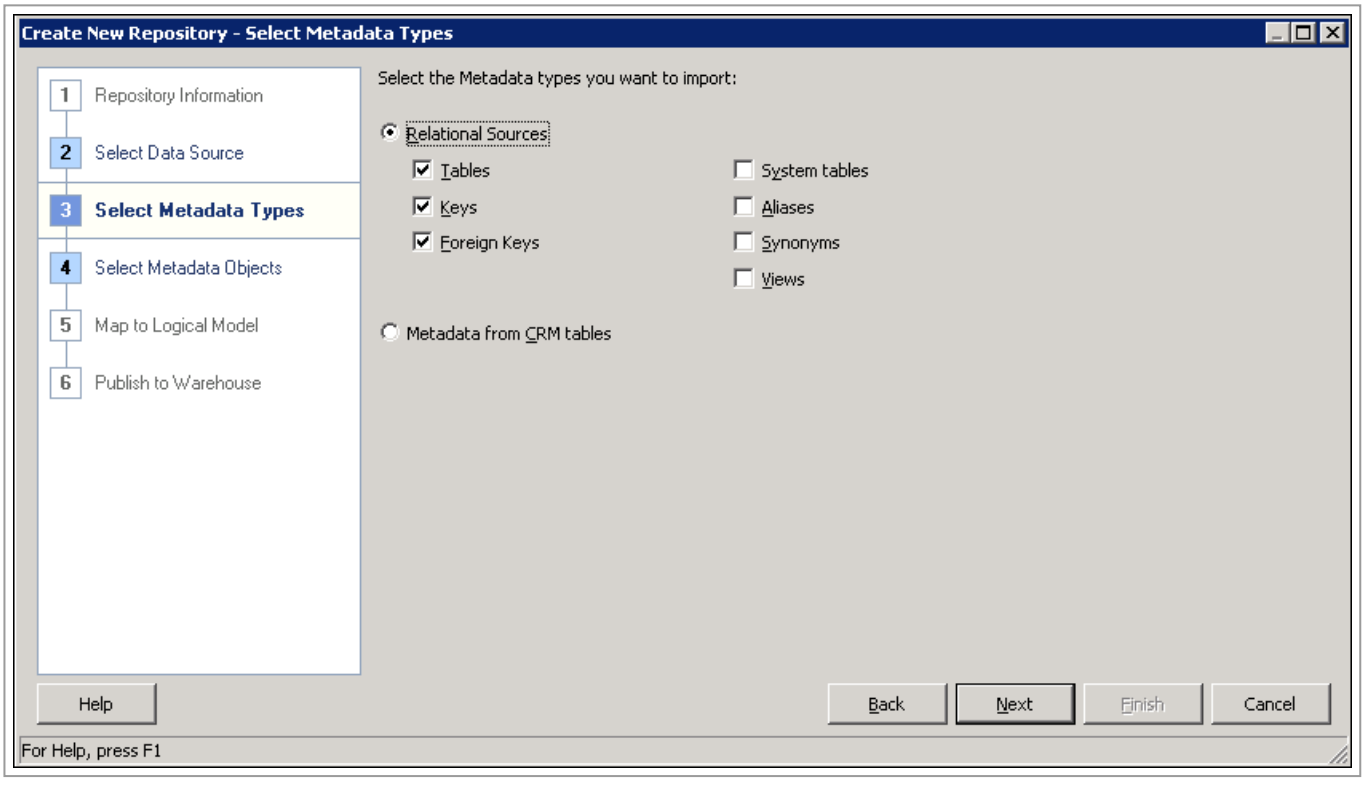

3. Enter user name and password for the data source. In this example the user name and password are both **BISAMPLE**. Recall that BISAMPLE is the name of the user/schema you created in the prerequisite

section.

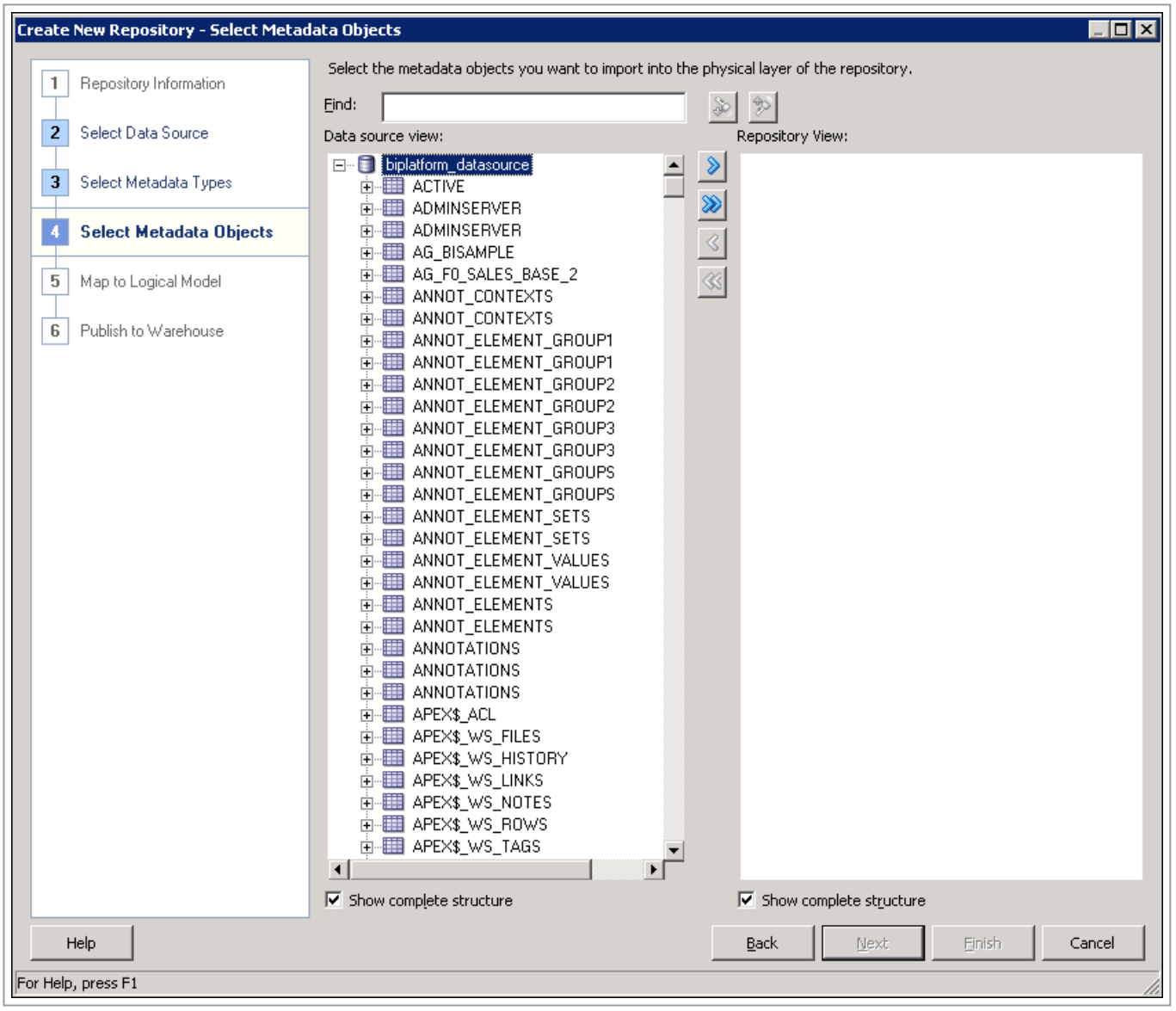



**4.** Click **Next**.

**5.** Accept the default metadata types and click **Next**.

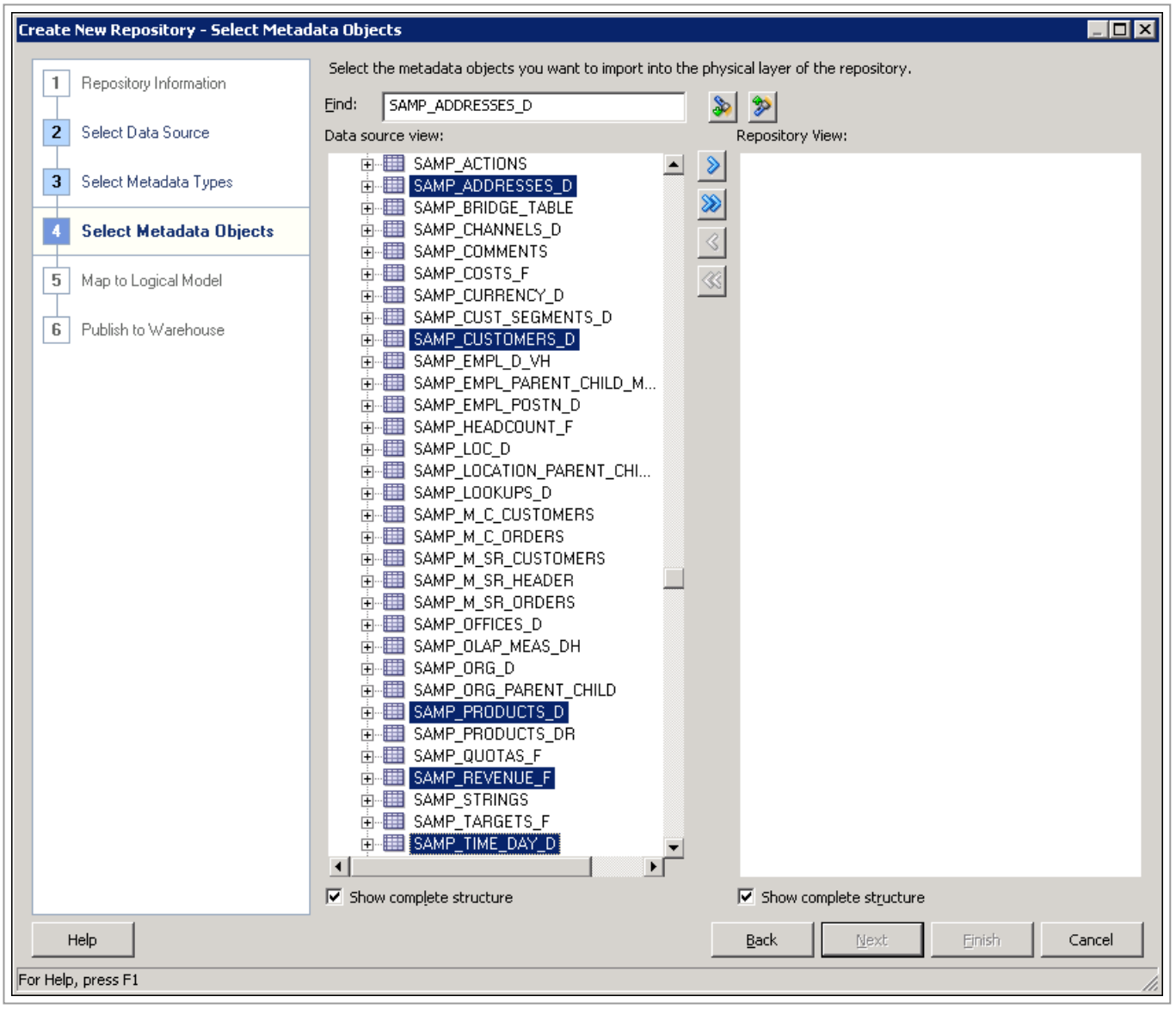



**6.** In the Data source view, expand the **biplatform_datasource** schema.

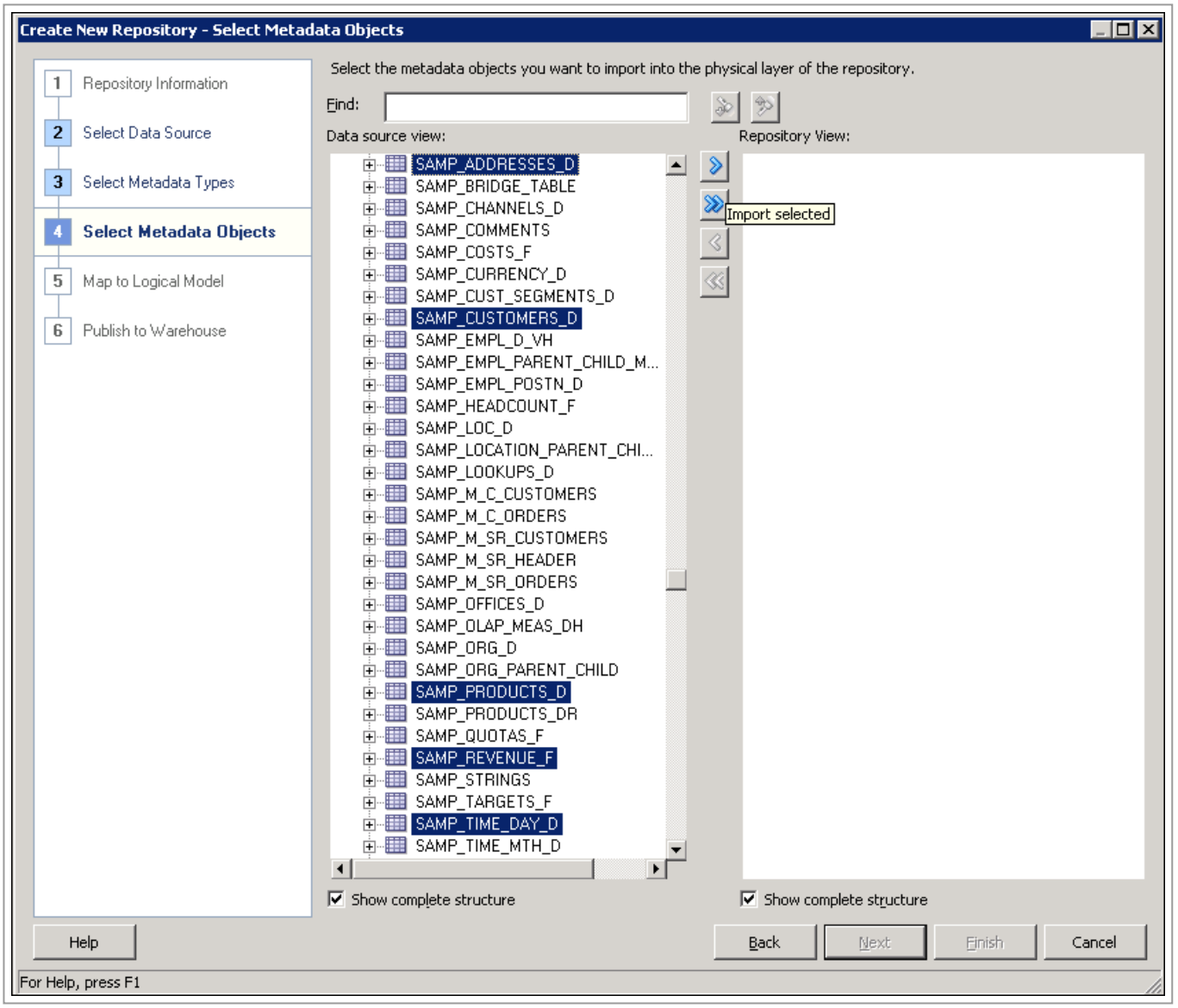



7. Press and hold **Ctrl** and click the following tables to select from the **BISAMPLE** schema:

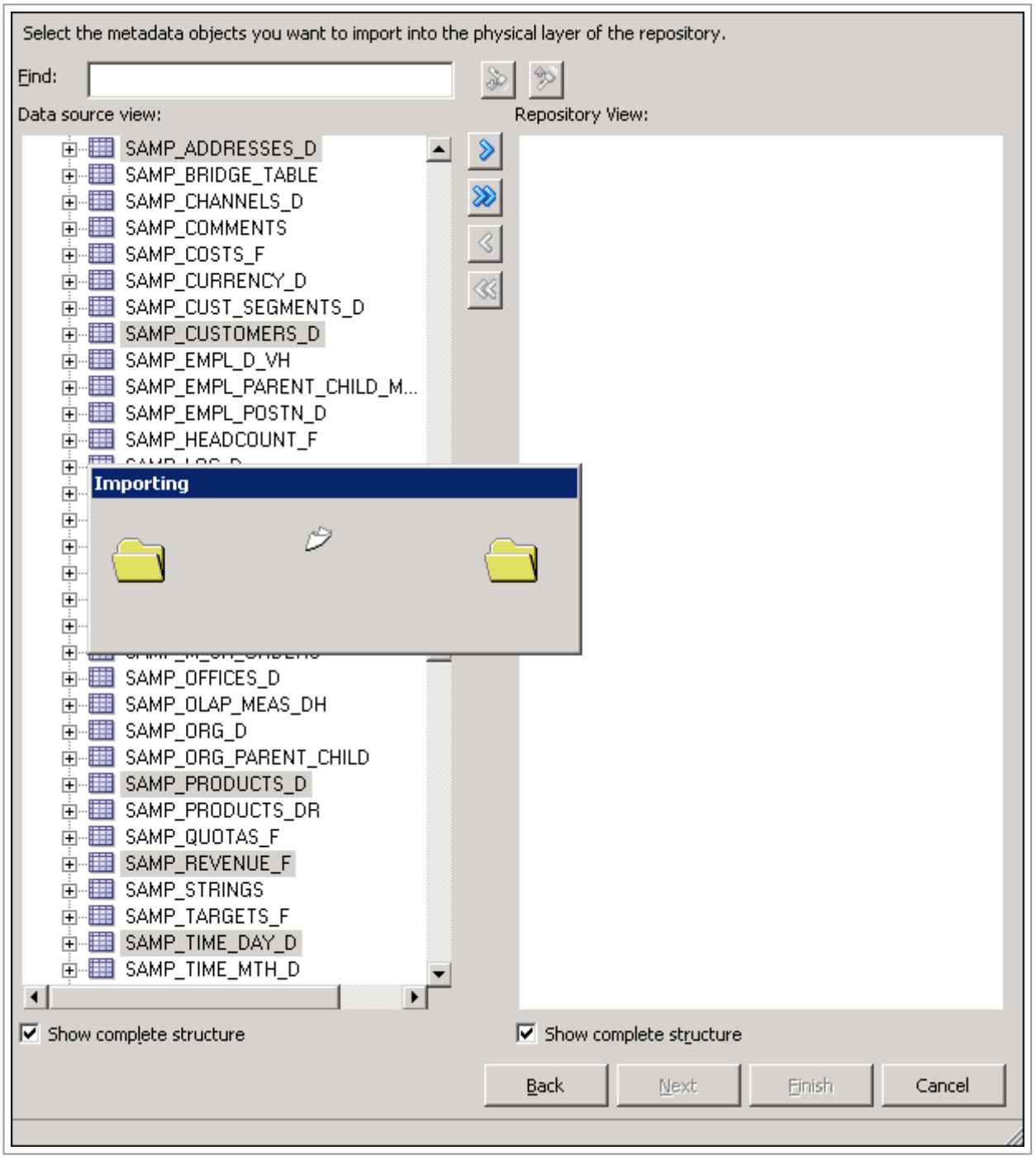
**SAMP_ADDRESSES_D**
**SAMP_CUSTOMERS_D**
**SAMP_PRODUCTS_D**
**SAMP_REVENUE_F**
**SAMP_TIME_DAY_D**



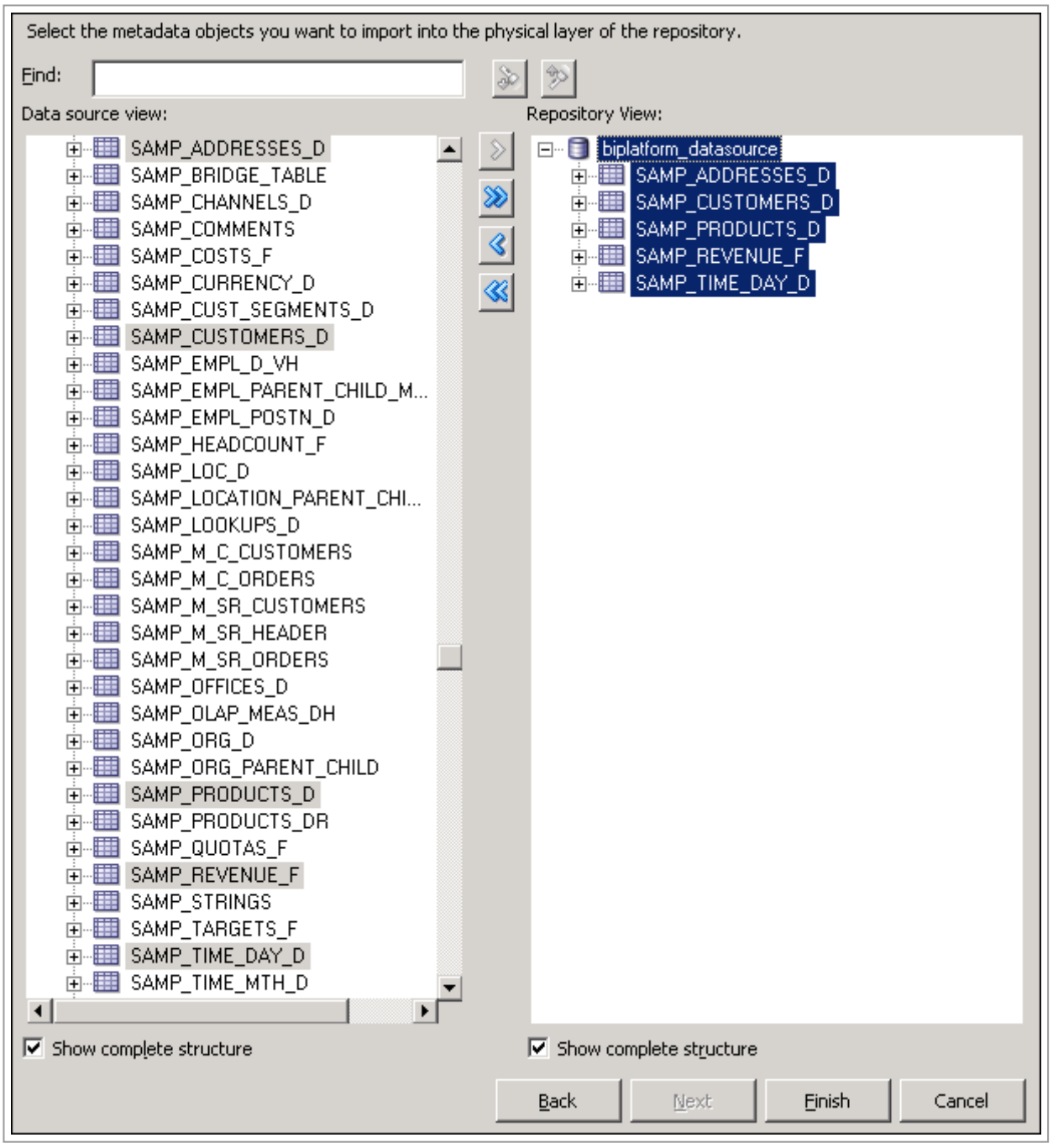

**8.** Click the **Import Selected** button to add the tables to the Repository View.



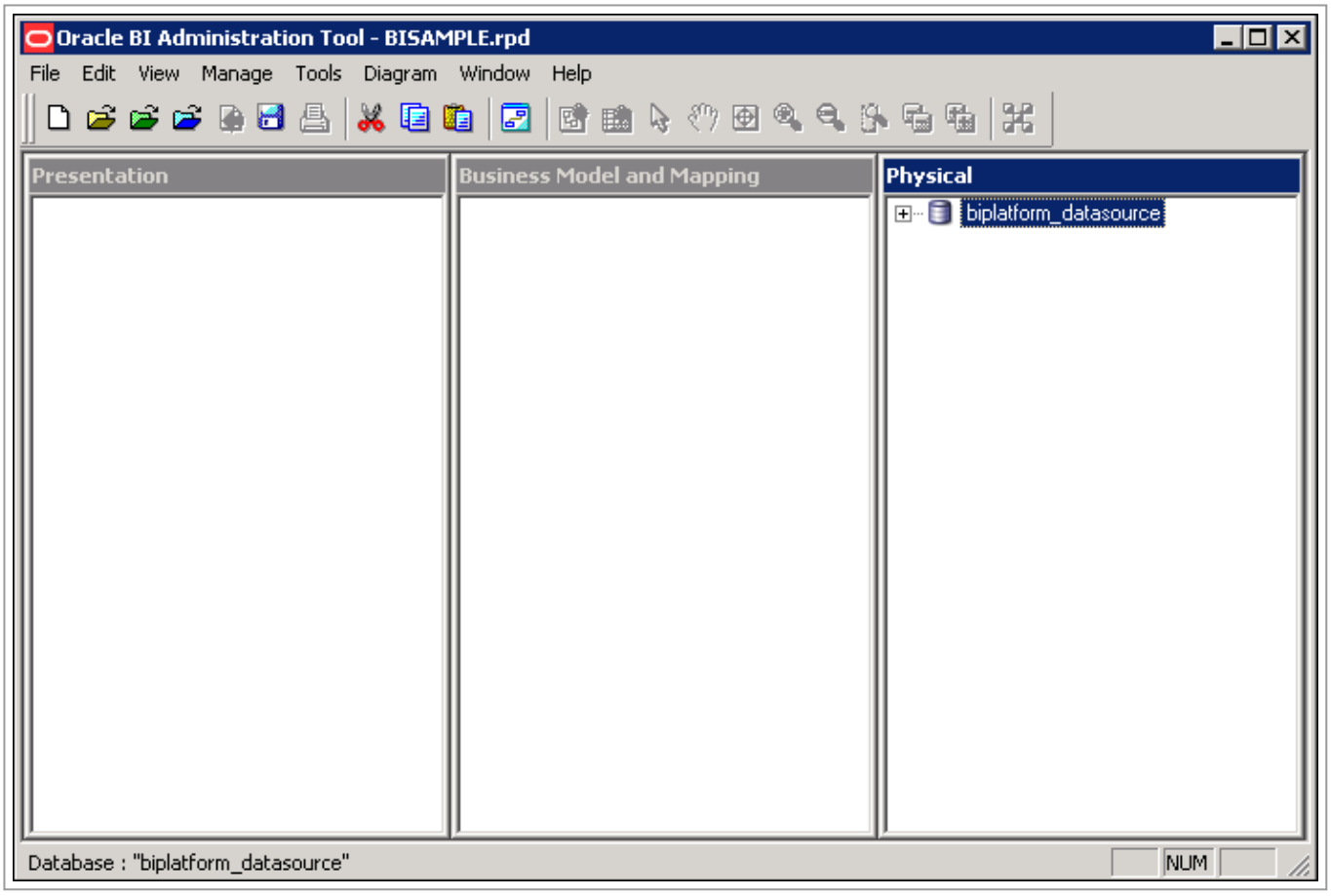

**9.** The **Importing** message appears.



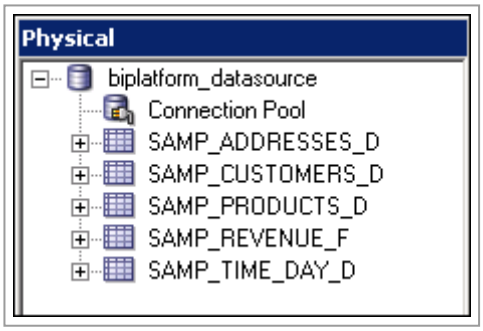

**10.** When import is complete, in the Repository View, expand **biplatform_datasource** and verify that the five tables are listed.




**11.** Click **Finish** to open the repository.

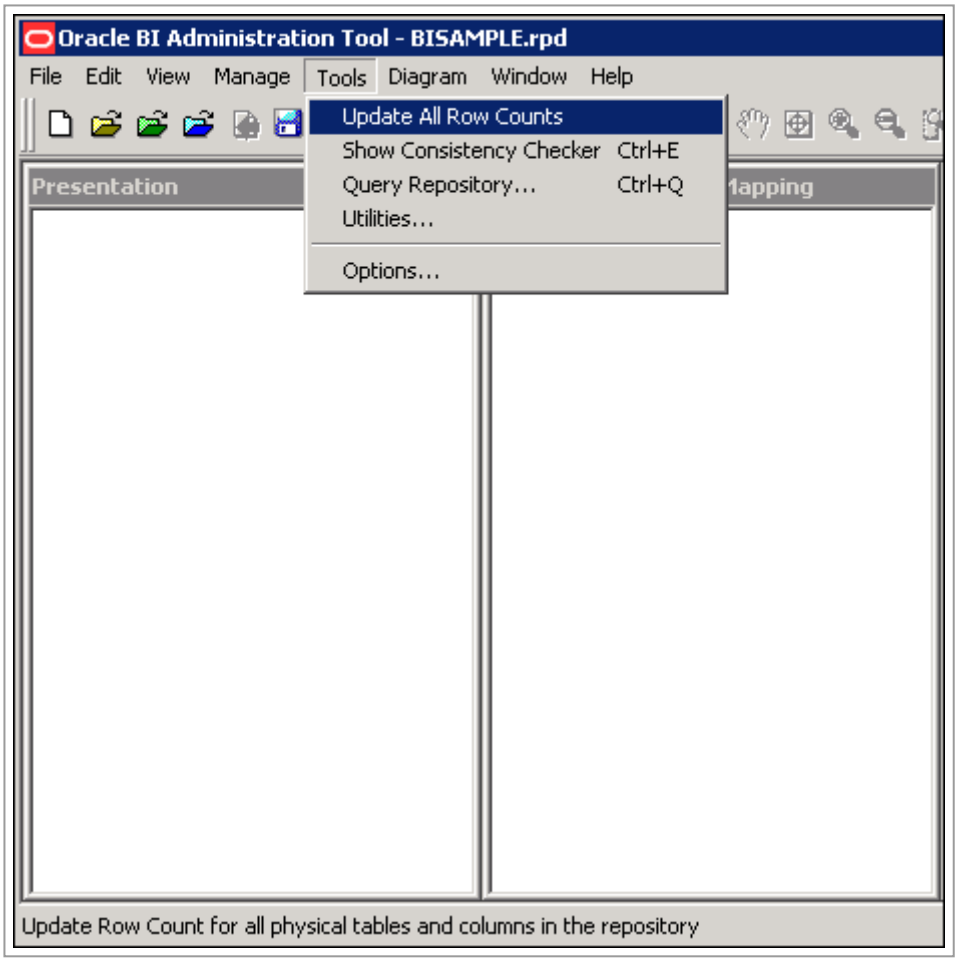



12. Expand **biplatform_datasource** and confirm that the five tables are imported into the **Physical** layer of the repository.

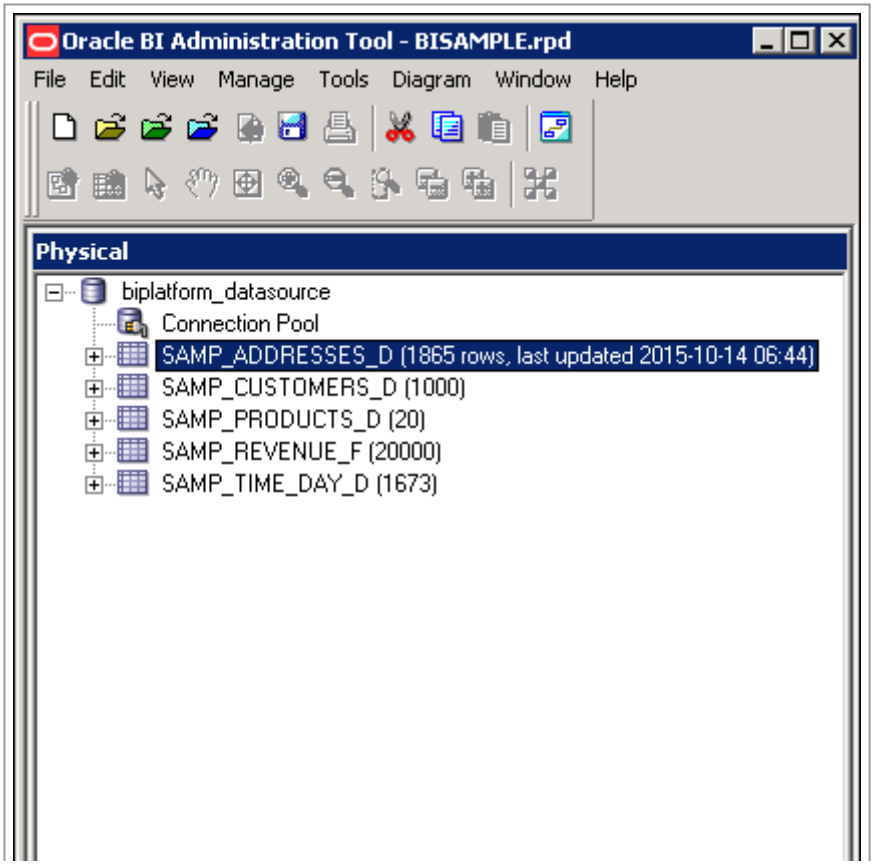


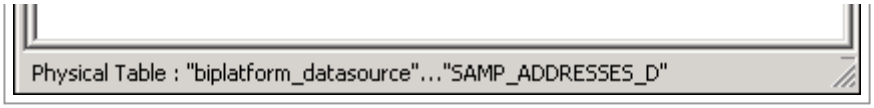

## Verifying Connection

1. Select **Tools > Update All Row Counts**.

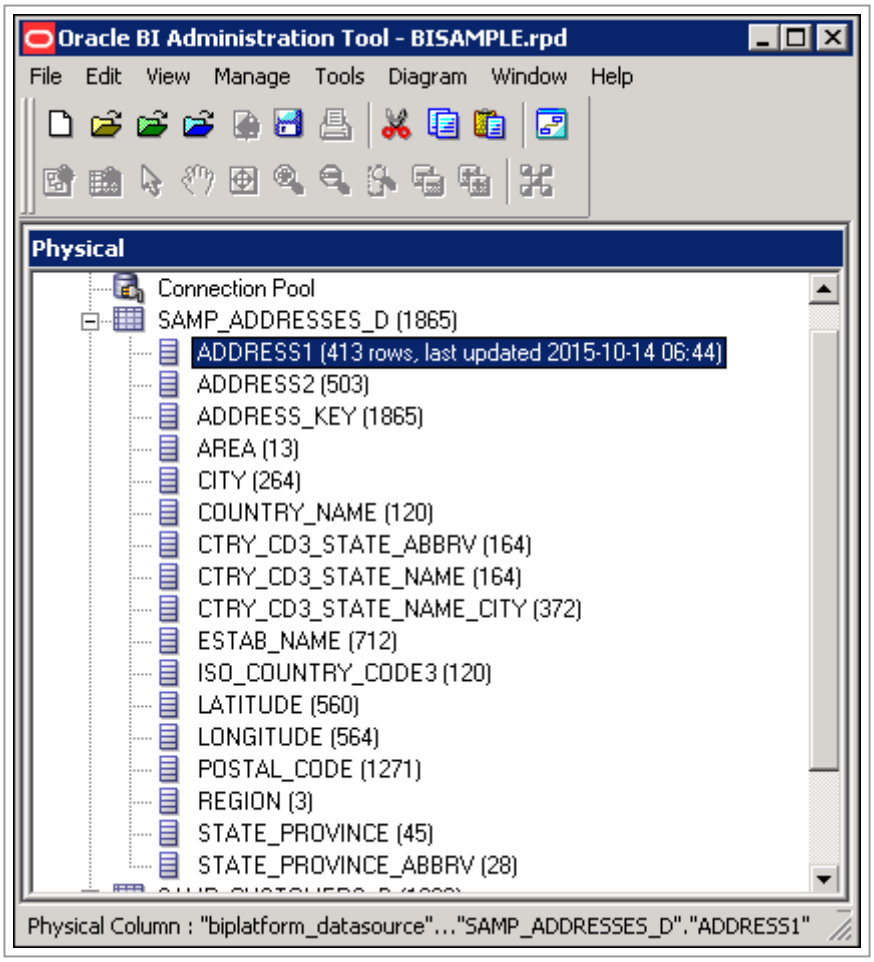



2. When update row counts completes, move the cursor over the tables and observe that row count information is now visible, including when the row count was last updated.

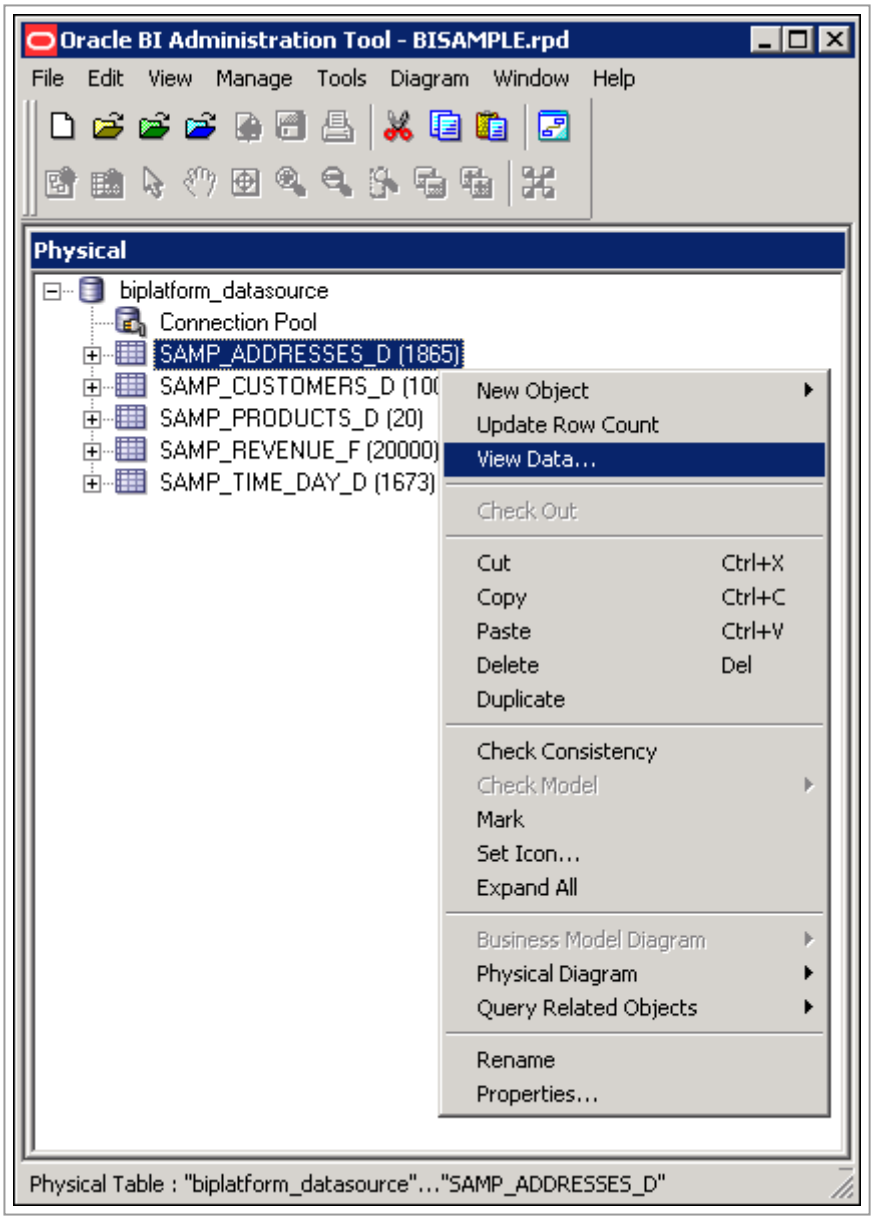




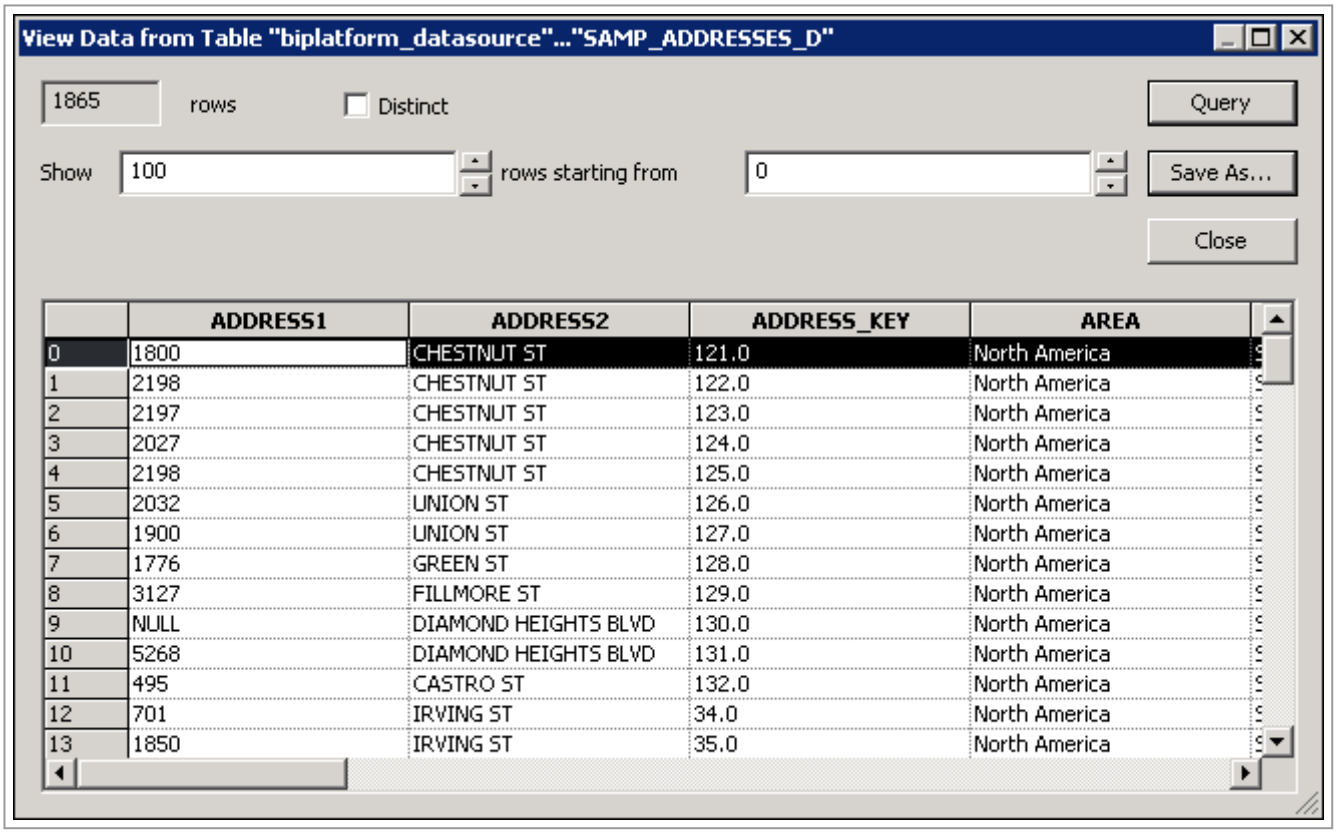

**3.** Expand tables and observe that row count information is also visible for individual columns.




**4.** Right-click a table and select **View Data** to view the data for the table.




5. Close the View Data dialog box when you are done. It is a good idea to update row counts or view data after an import to verify connectivity. Viewing data or updating row count, if successful, tells you that your connection is configured correctly.

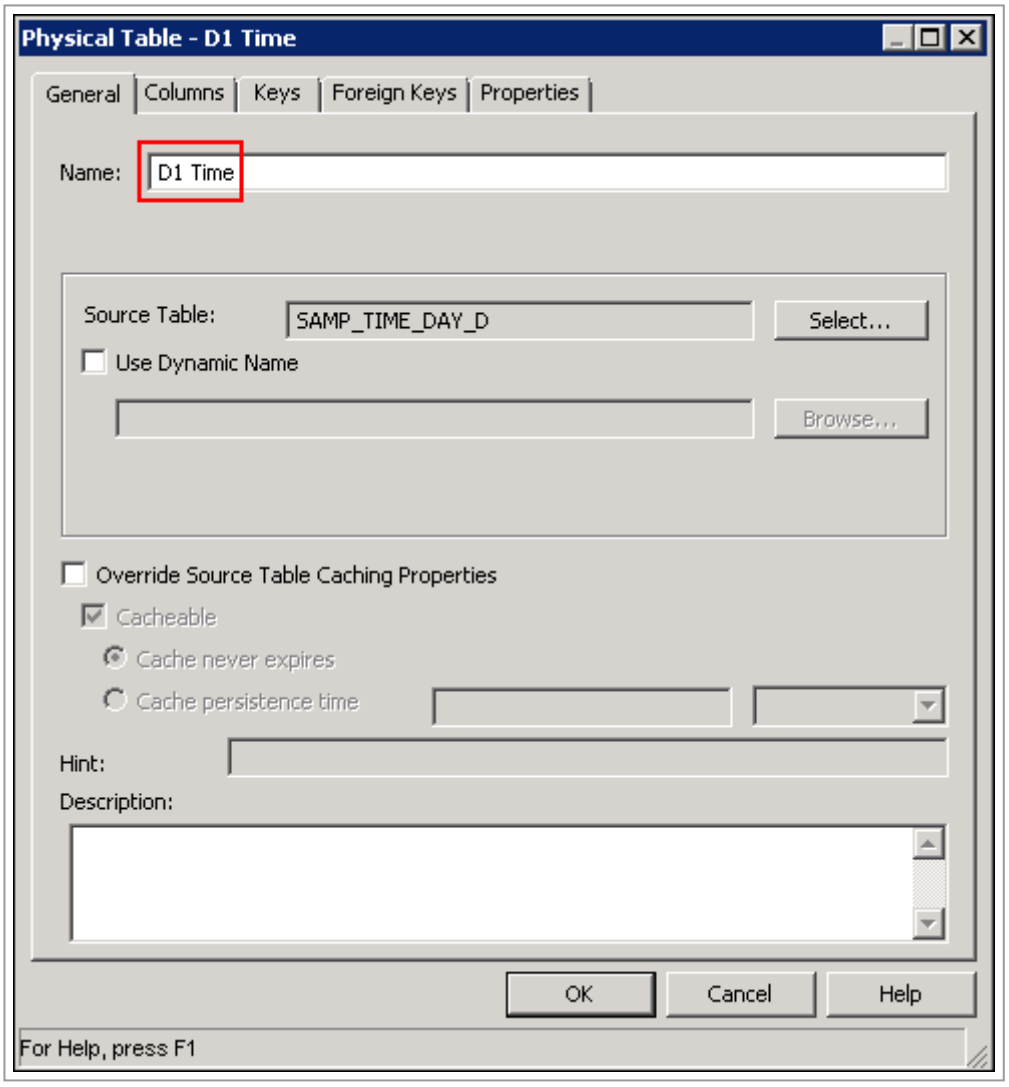



| | ADDRESS1 | ADDRESS2 | ADDRESS_KEY | AREA |
|---|---|---|---|---|
| 0 | 1800 | CHESTNUT ST | 121.0 | North America |
| 1 | 2198 | CHESTNUT ST | 122.0 | North America |
| 2 | 2197 | CHESTNUT ST | 123.0 | North America |
| 3 | 2027 | CHESTNUT ST | 124.0 | North America |
| 4 | 2198 | CHESTNUT ST | 125.0 | North America |
| 5 | 2032 | UNION ST | 126.0 | North America |
| 6 | 1900 | UNION ST | 127.0 | North America |
| 7 | 1776 | GREEN ST | 128.0 | North America |
| 8 | 3127 | FILLMORE ST | 129.0 | North America |
| 9 | NULL | DIAMOND HEIGHTS BLVD | 130.0 | North America |
| 10 | 5268 | DIAMOND HEIGHTS BLVD | 131.0 | North America |
| 11 | 495 | CASTRO ST | 132.0 | North America |
| 12 | 701 | IRVING ST | 34.0 | North America |
| 13 | 1850 | IRVING ST | 35.0 | North America |

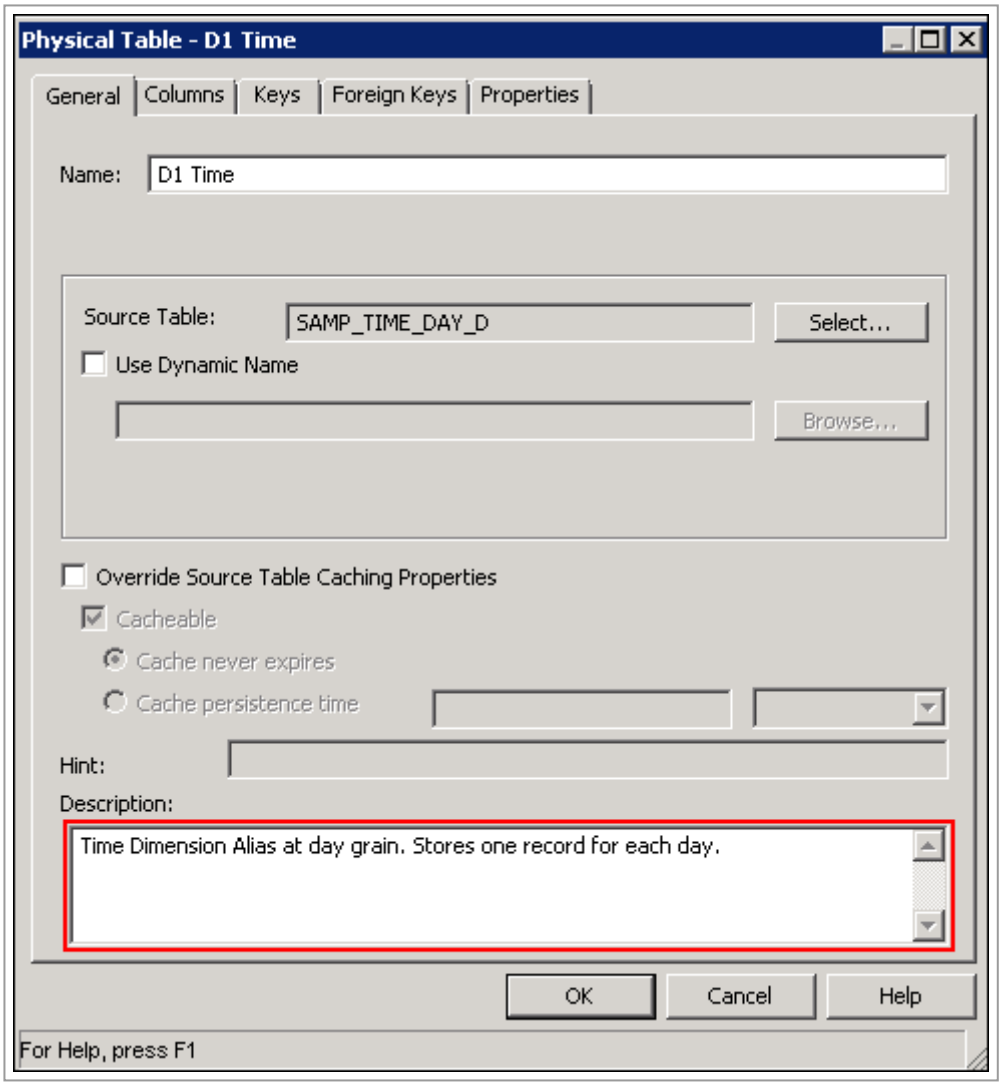
## Creating Aliases

1. It is recommended that you use table aliases frequently in the Physical layer to eliminate extraneous joins and to include best practice naming conventions for physical table names. Right-click **SAMP_TIME_DAY_D** and select **New Object > Alias** to open the **Physical Table** dialog box.

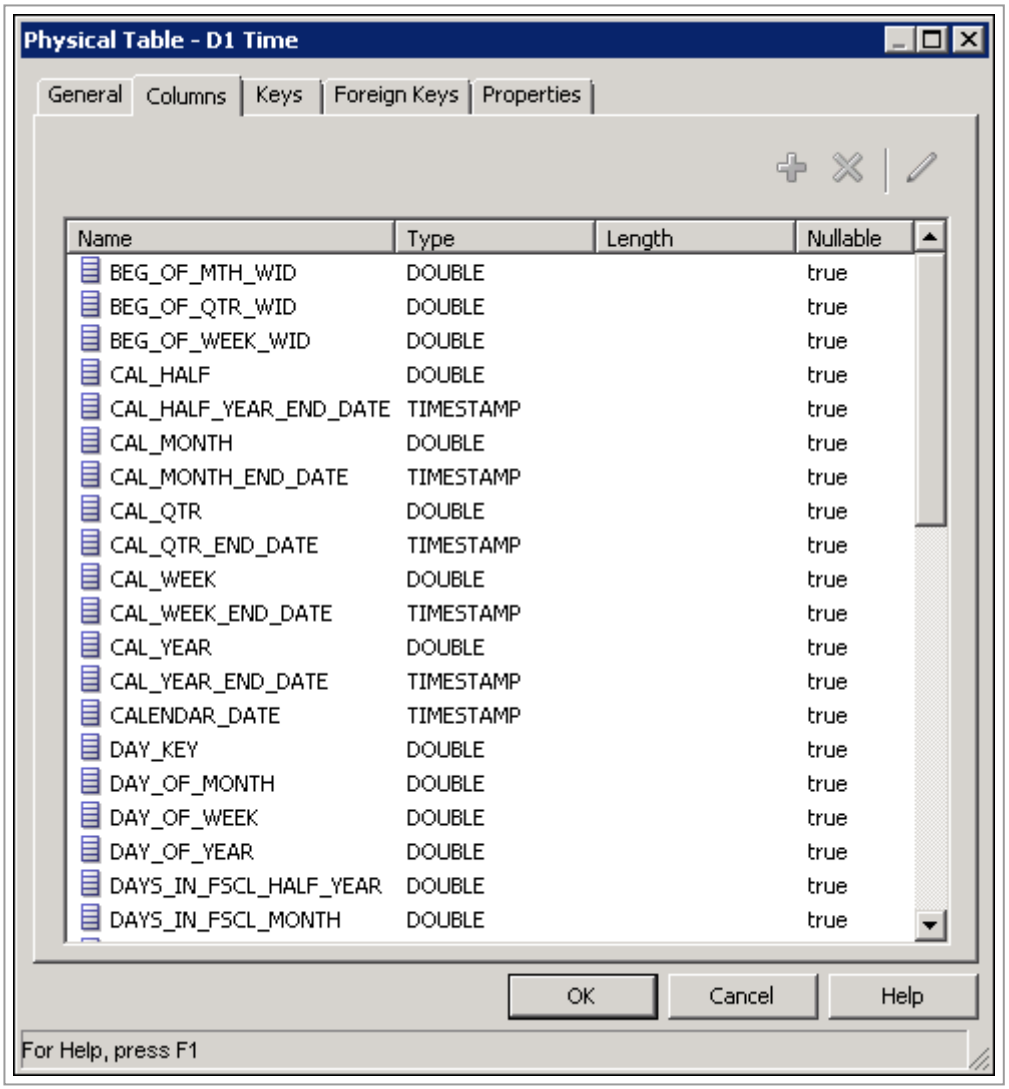
2. Enter **D1 Time** in the Name field.




**3.** In the Description field, enter **Time Dimension Alias at day grain. Stores one record for each day**.



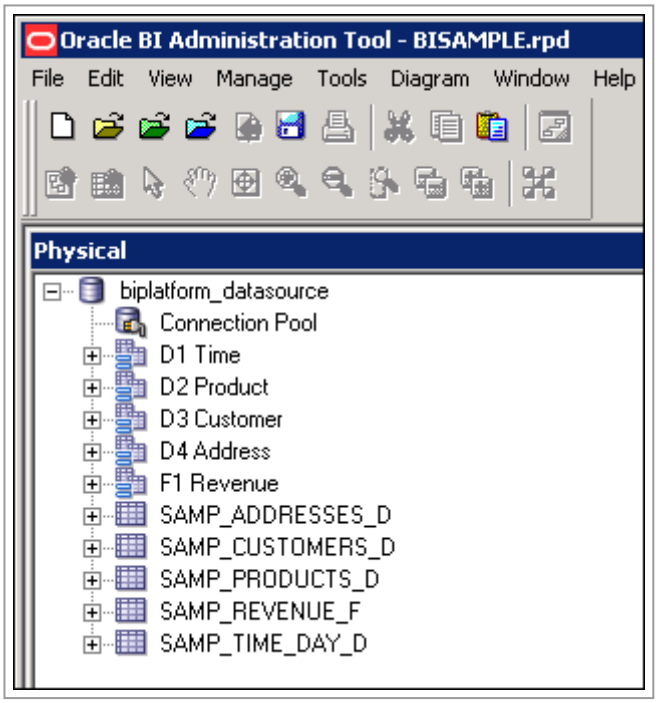

**4.** Click the **Columns** tab. Note that alias tables inherit all column definitions from the source table.




5. Click **OK** to close the **Physical Table** dialog box.

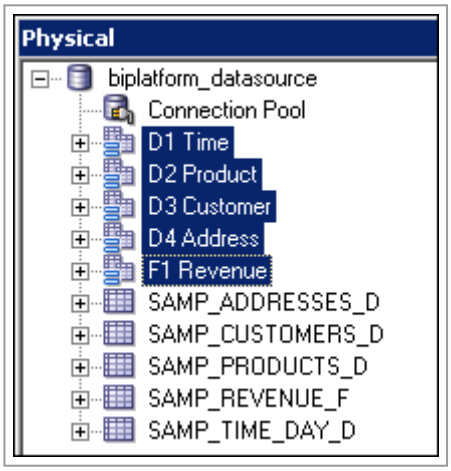
6. Repeat the steps and create the following aliases for the remaining physical tables.

   SAMP_ADDRESSES_D = **D4 Address**
   SAMP_CUSTOMERS_D = **D3 Customer**
   SAMP_PRODUCTS_D = **D2 Product**
   SAMP_REVENUE_F = **F1 Revenue**

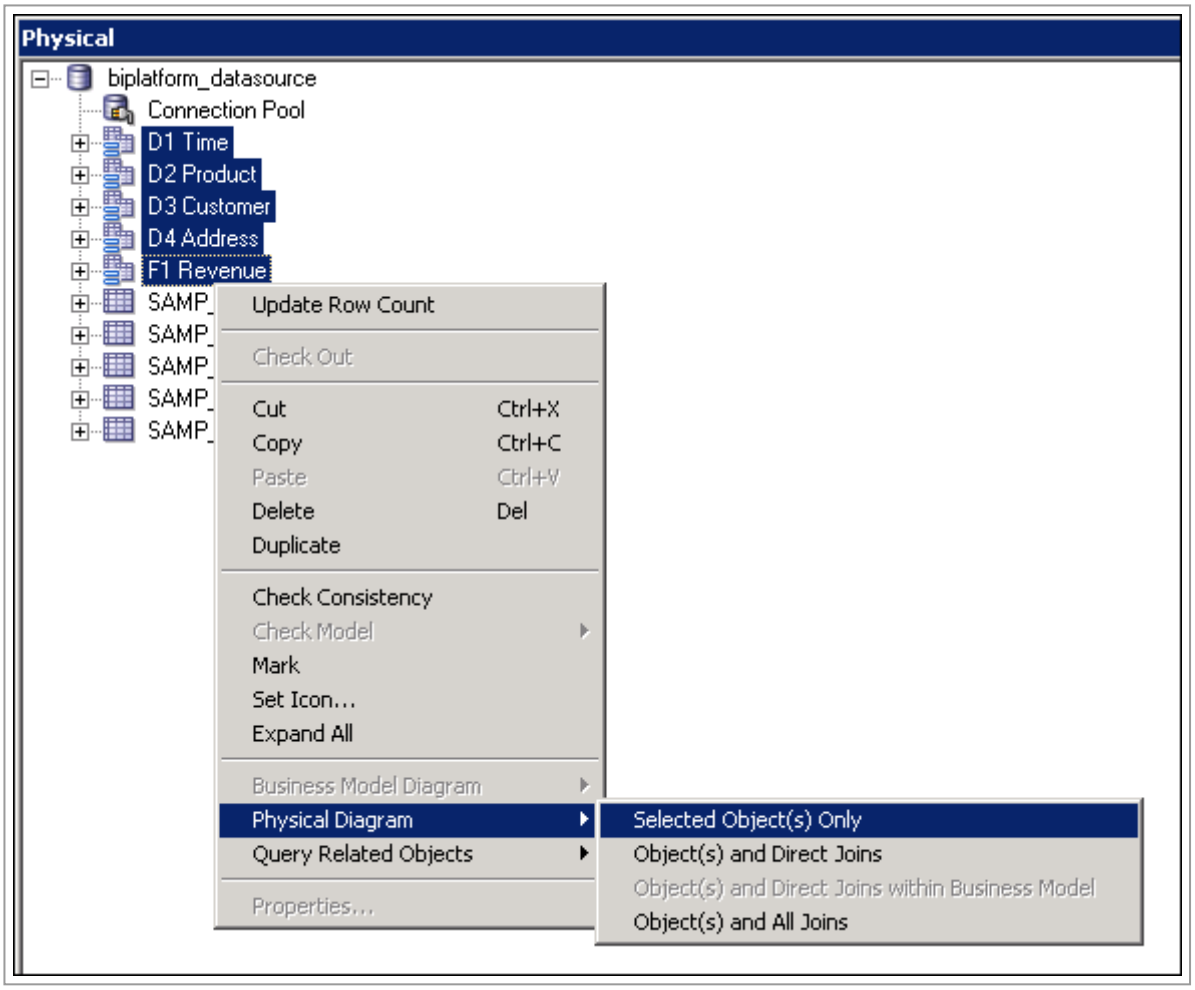



## Creating Keys and Joins

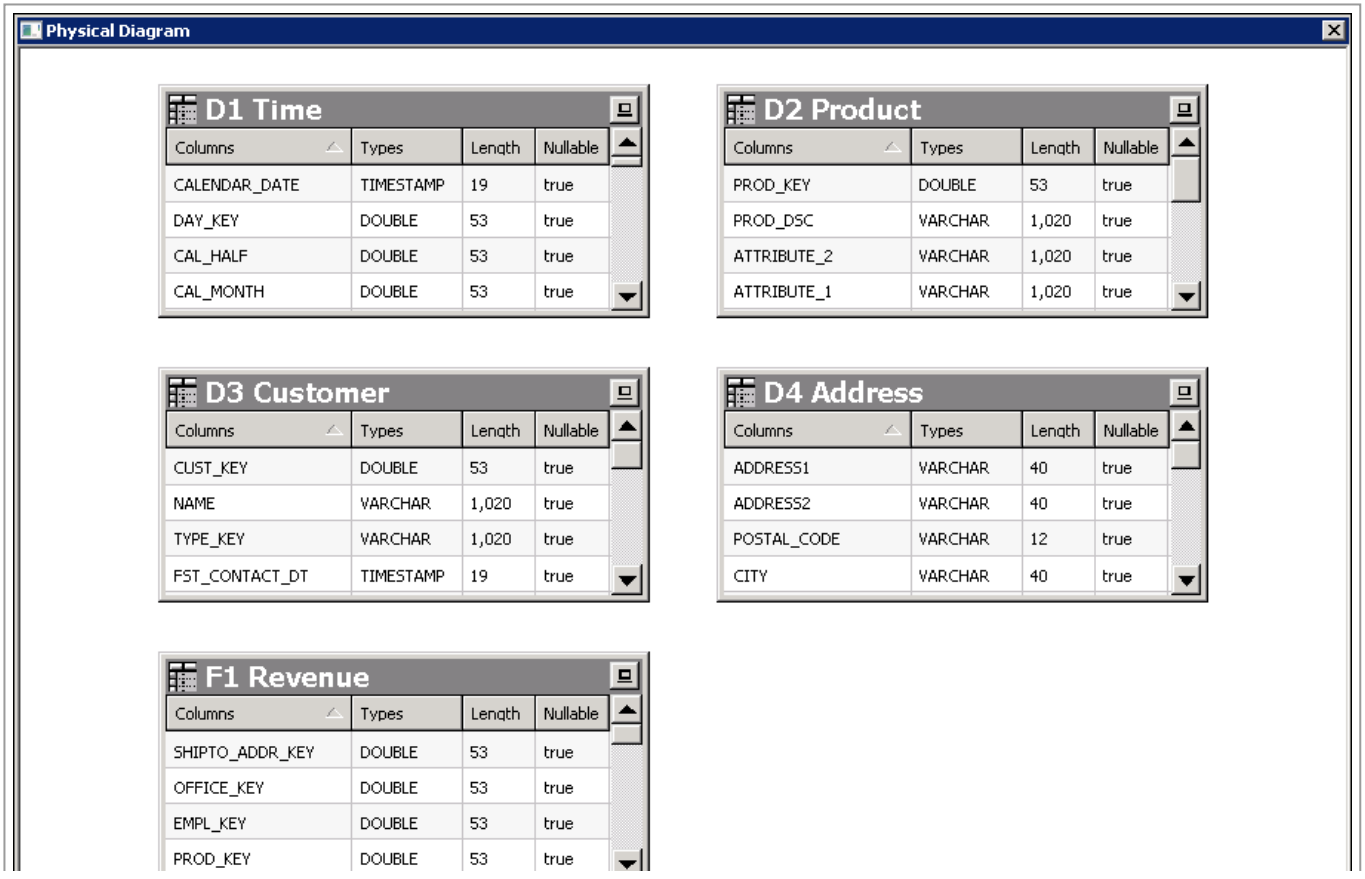
1. Select the five alias tables in the **Physical** layer.




2. Right-click one of the highlighted alias tables and select **Physical Diagram > Selected Object(s) Only** to open the Physical Diagram. Alternatively, you can click the Physical Diagram button on the toolbar.

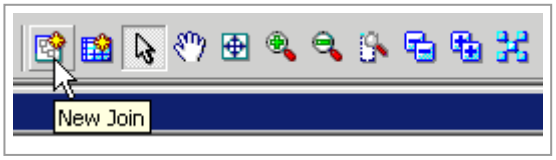



**3.** Rearrange the alias table objects so they are all visible.

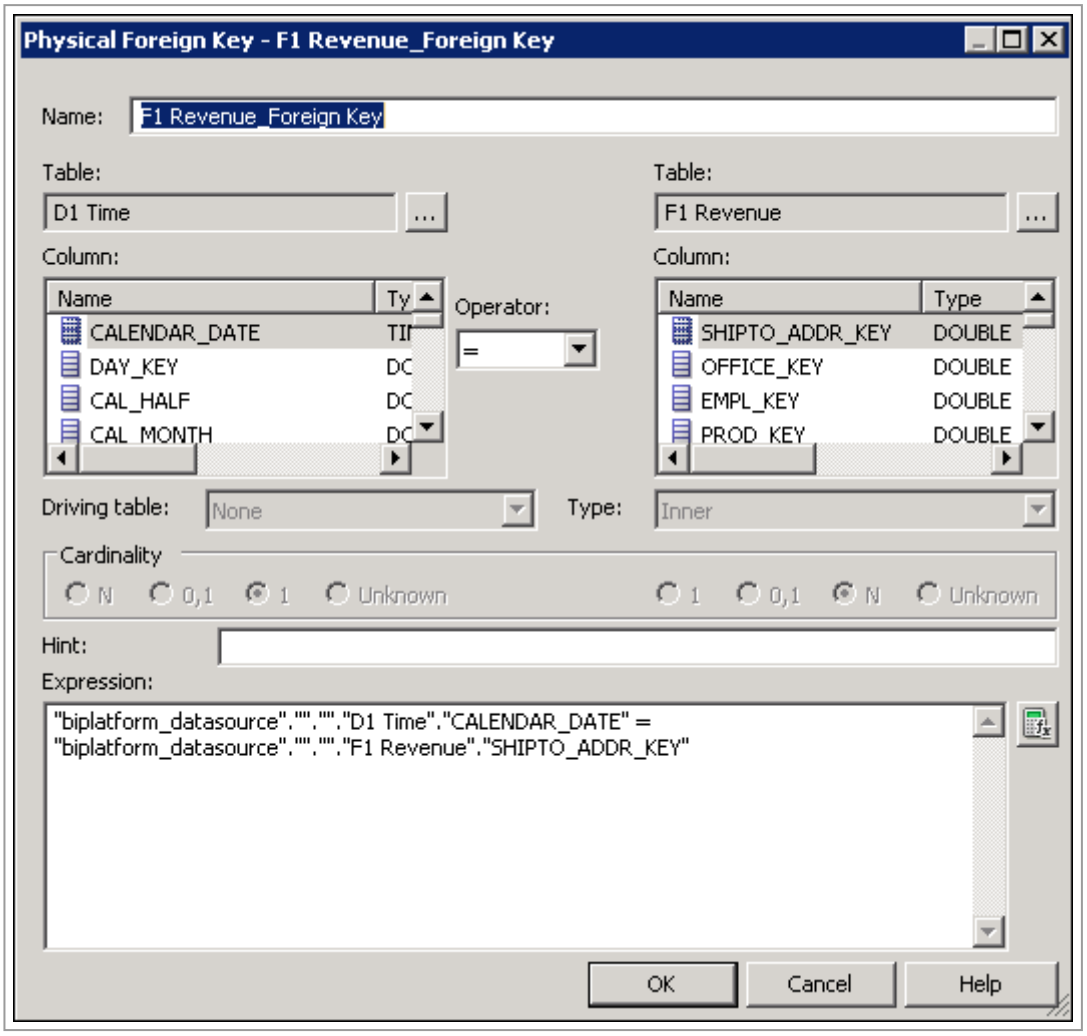


4. You may want to adjust the objects in the Physical Diagram. If so, use the toolbar buttons to zoom in, zoom out, fit the diagram, collapse or expand objects, select objects, and so forth:

5. Click the **New Join** button on the toolbar.



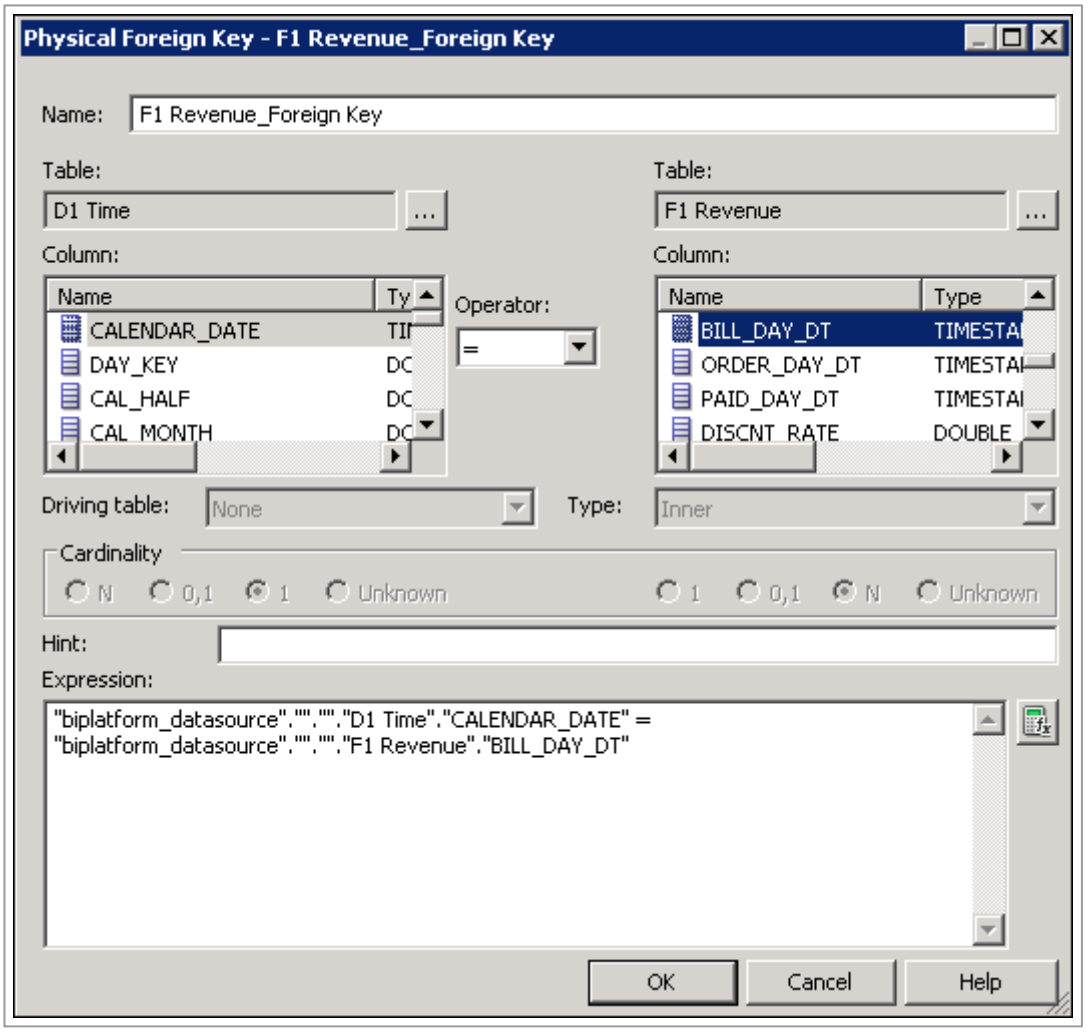

6. Click the **F1 Revenue** table and then the **D1 Time** table. The **Physical Foreign Key** dialog box opens. It matters which table you click first. The join creates a **one-to-many (1:N)** relationship that joins the key column in the first table to a foreign key column in the second table.



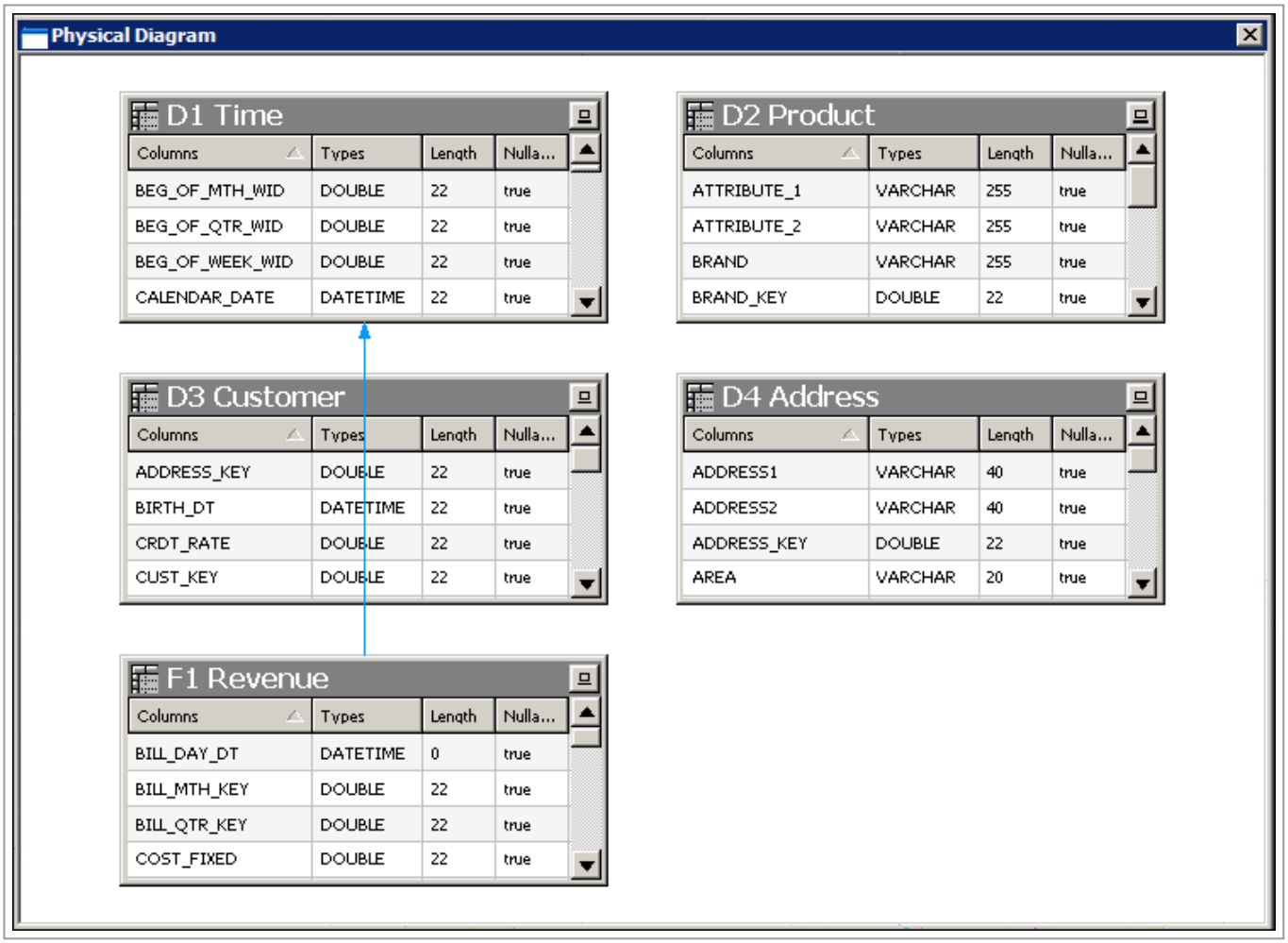

7. Select the **D1 Time. CALENDAR_DATE** column, and then select **F1 Revenue.BILL_DAY_DT** to join the tables. Ensure that the Expression edit box (at the bottom) contains the following expression:

"biplatform_datasource"."".""BISAMPLE"."D1 Time"."CALENDAR_DATE" =
"biplatform_datasource"."".""BISAMPLE"."F1 Revenue"."BILL_DAY_DT"

biplatform_datasource"."."BISAMPLE"."F1 Revenue"."BILL_DAY_DT"




**8.** Click **OK** to close the Physical Foreign Key dialog box. The join is visible in the Physical Diagram.




Please be aware of the following upgrade considerations for Oracle BI EE 11g Release 1 (11.1.1.5): Joins in the Physical and Business Model diagrams are now represented by a line with an arrow at the "one" end of the join, rather than the line with crow's feet at the "many" end of the join that was used in previous releases. When creating joins in the Physical and Business Model Diagrams, you now select the "many" end of the join first, and then select the "one" end of the join. In previous releases, joins in the diagrams were created by selecting the "one" end of the join first.

9. Repeat the steps to create joins for the remaining tables. Use the following expressions as a guide. Please notice that **D4 Address** joins to **D3 Customer**.

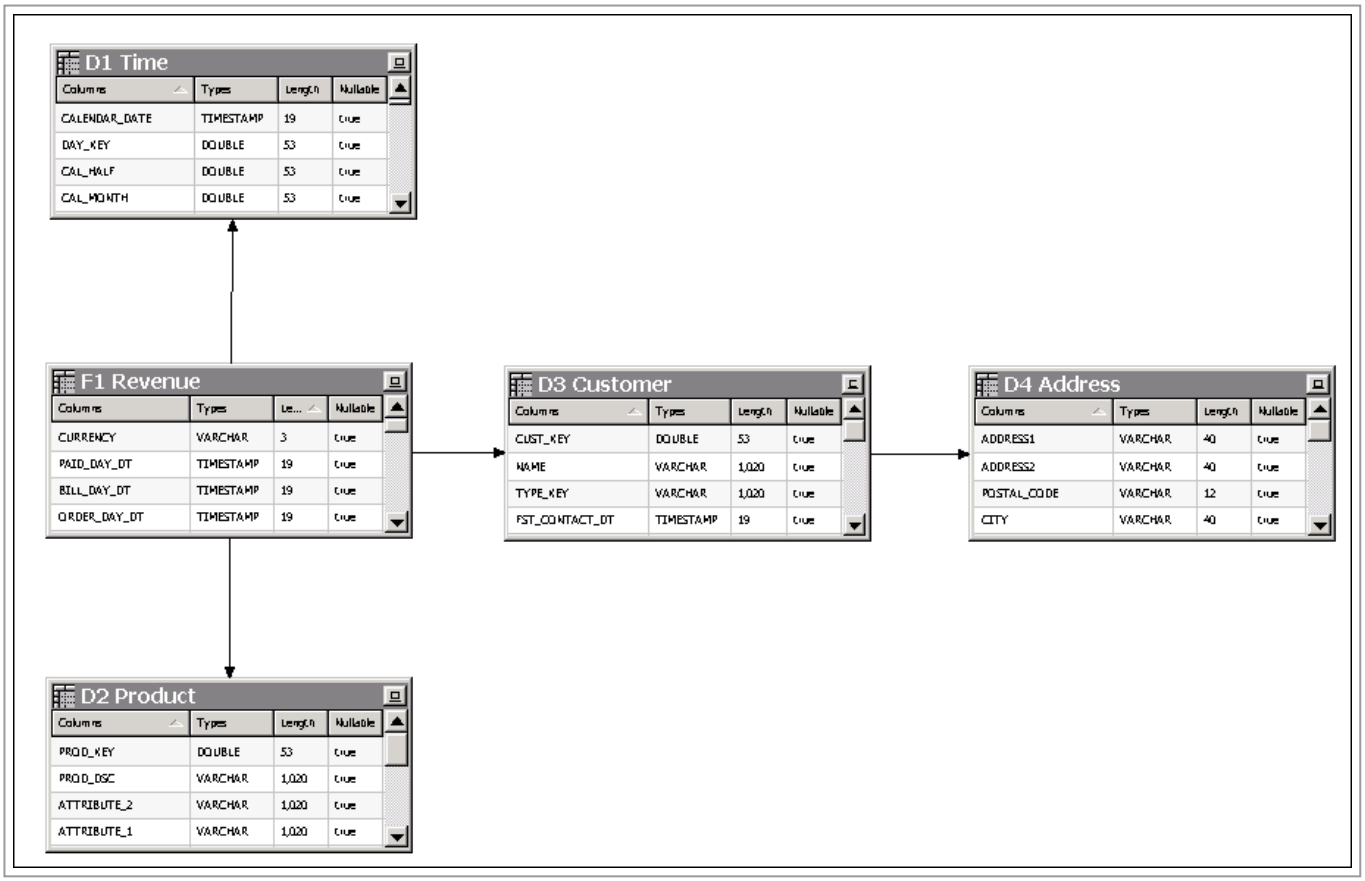
   "biplatform_datasource"."""."BISAMPLE"."D2 Product"."PROD_KEY" =
   "biplatform_datasource"."""."BISAMPLE"."F1 Revenue"."PROD_KEY"

   "biplatform_datasource"."""."BISAMPLE"."D3 Customer"."CUST_KEY" =
   "biplatform_datasource"."""."BISAMPLE"."F1 Revenue"."CUST_KEY"

   "biplatform_datasource"."""."BISAMPLE"."D4 Address"."ADDRESS_KEY" =
   "biplatform_datasource"."""."BISAMPLE"."D3 Customer"."ADDRESS_KEY"

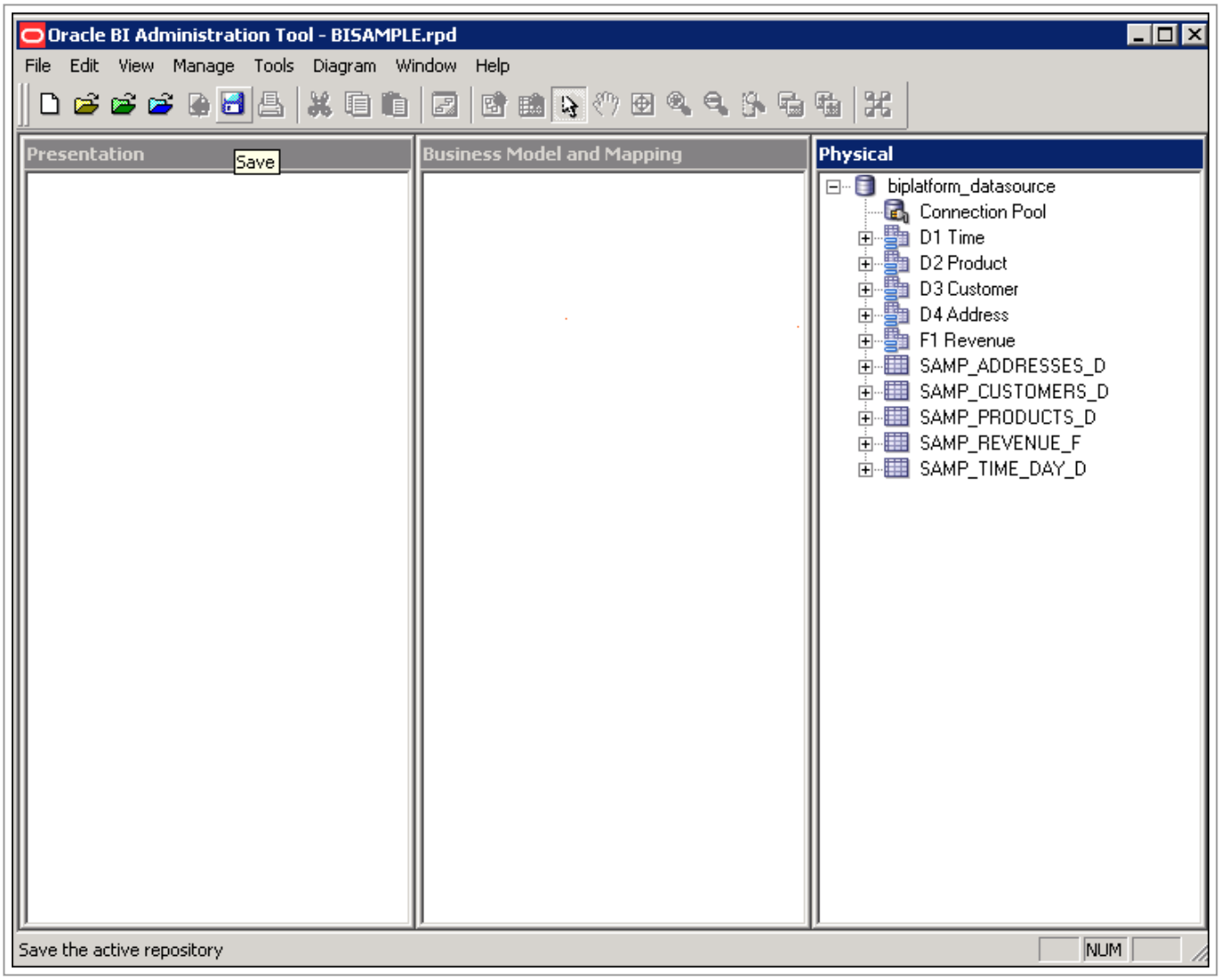
10. Click the **Auto Layout** button on the toolbar.


**11.** Your diagram should look similar to the screenshot:

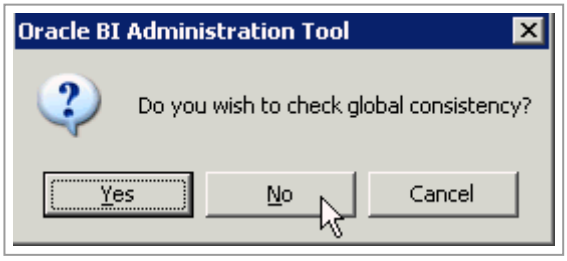



**12.** Click the **X** in the upper right corner to close the Physical Diagram.

**13.** Select **File > Save** or click the **Save** button on the toolbar to save the repository.




14. Click **No** when prompted to check global consistency. Some of the more common checks are done in the **Business Model and Mapping** layer and **Presentation** layer. Since these layers are not defined yet, bypass this check until the other layers in the repository are built.




15. Leave the **Administration Tool** and the repository open for the next topic.

Congratulations! You have successfully created a new repository, imported a table schema from an external data source into the Physical layer, created aliases, and defined keys and joins.

In the next topic you learn how to build the **Business Model and Mapping** layer of a repository.

## Building the Business Model and Mapping Layer of a Repository

In this topic you use the Oracle BI Administration Tool to build the Business Model and Mapping layer of a repository.

The Business Model and Mapping layer of the Administration Tool defines the business, or logical model of the data and specifies the mappings between the business model and the Physical layer schemas. This layer is where the physical schemas are simplified to form the basis for the users' view of the data. The Business Model and Mapping layer of the Administration Tool can contain one or more business model objects. A business model object contains the business model definitions and the mappings from logical to physical tables for the business model.

The main purpose of the business model is to capture how users think about their business using their own vocabulary. The business model simplifies the physical schema and maps the users' business vocabulary to physical sources. Most of the vocabulary translates into logical columns in the business model. Collections of logical columns form logical tables. Each logical column (and hence each logical table) can have one or more physical objects as sources.

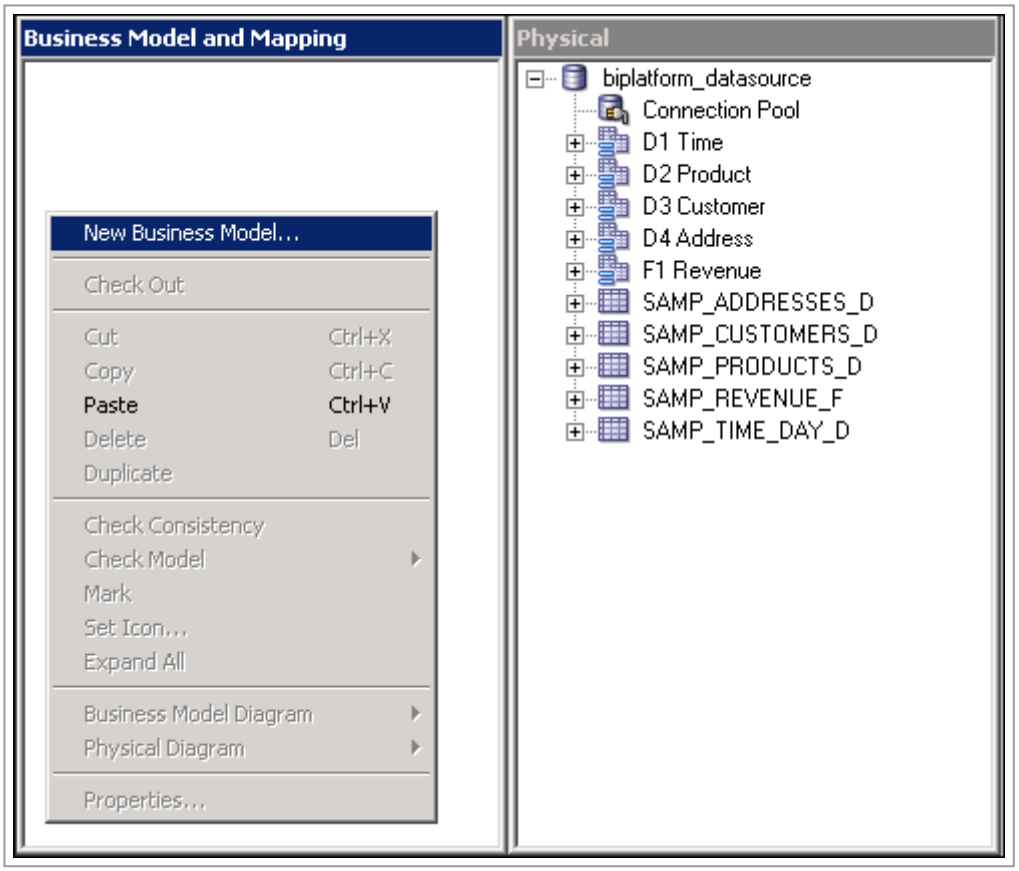
There are two main categories of logical tables: fact and dimension. Logical fact tables contain the measures by which an organization gauges its business operations and performance. Logical dimension tables contain the data used to qualify the facts.

To build the Business Model and Mapping layer of a repository, you perform the following steps:

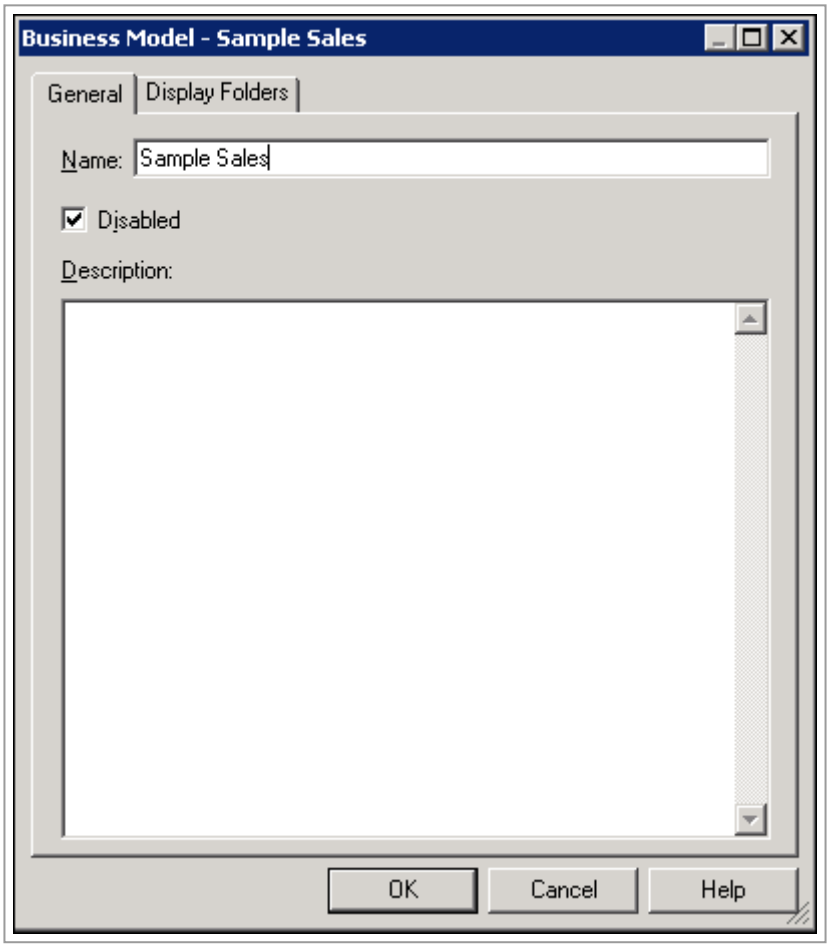
- Creating a Business Model
- Examining Logical Joins
- Examining Logical Columns
- Examining Logical Table Sources
- Renaming Logical Objects Manually
- Renaming Logical Objects Using the Rename Wizard
- Deleting Unnecessary Logical Objects
- Creating Simple Measures

## Creating a Business Model

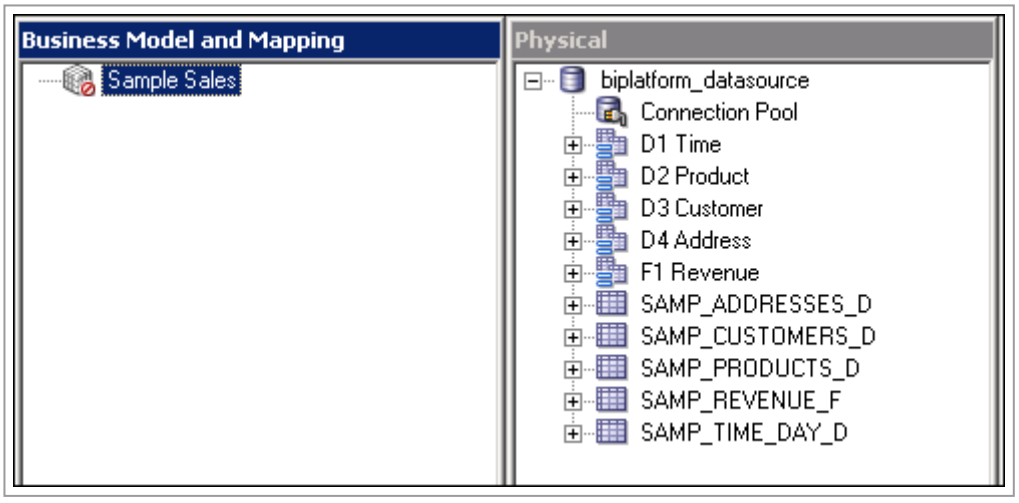
1. Right-click the white space in the **Business Model and Mapping** layer and select **New Business Model** to open the **Business Model** dialog box.




**2.** Enter **Sample Sales** in the **Name** field. Leave **Disabled** checked.



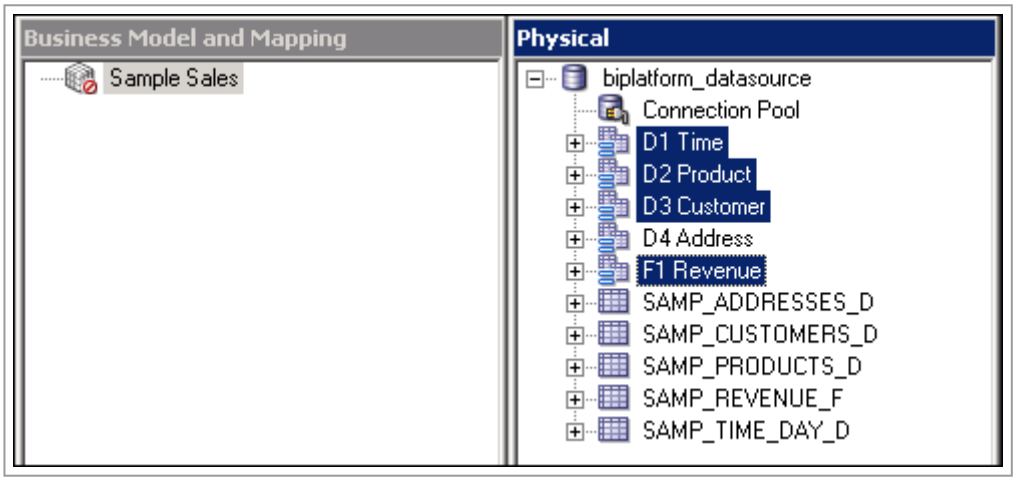
3. Click **OK**. The Sample Sales business model is added to the **Business Model and Mapping** layer.



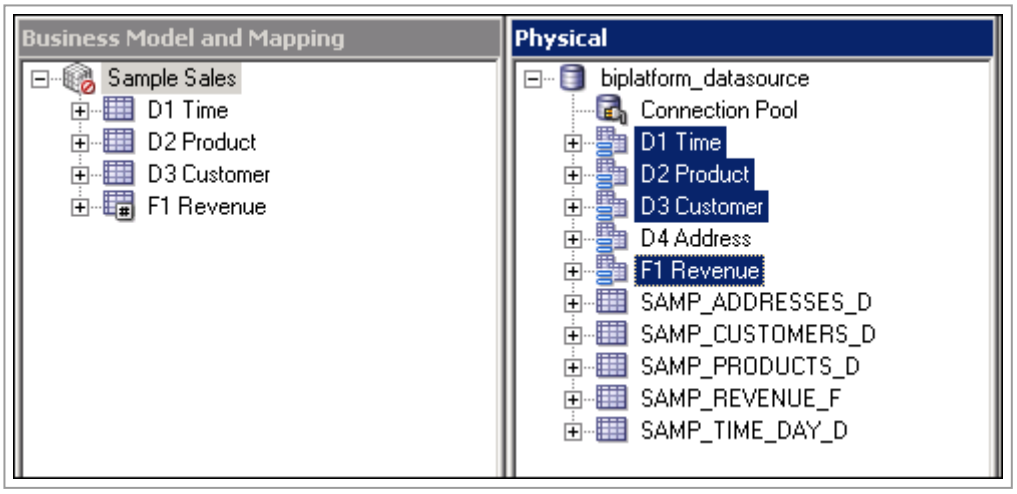

4. In the **Physical** layer, select the following four alias tables:

   **D1 Time**
   **D2 Product**
   **D3 Customer**
   **F1 Revenue**

   Do not select **D4 Address** at this time.

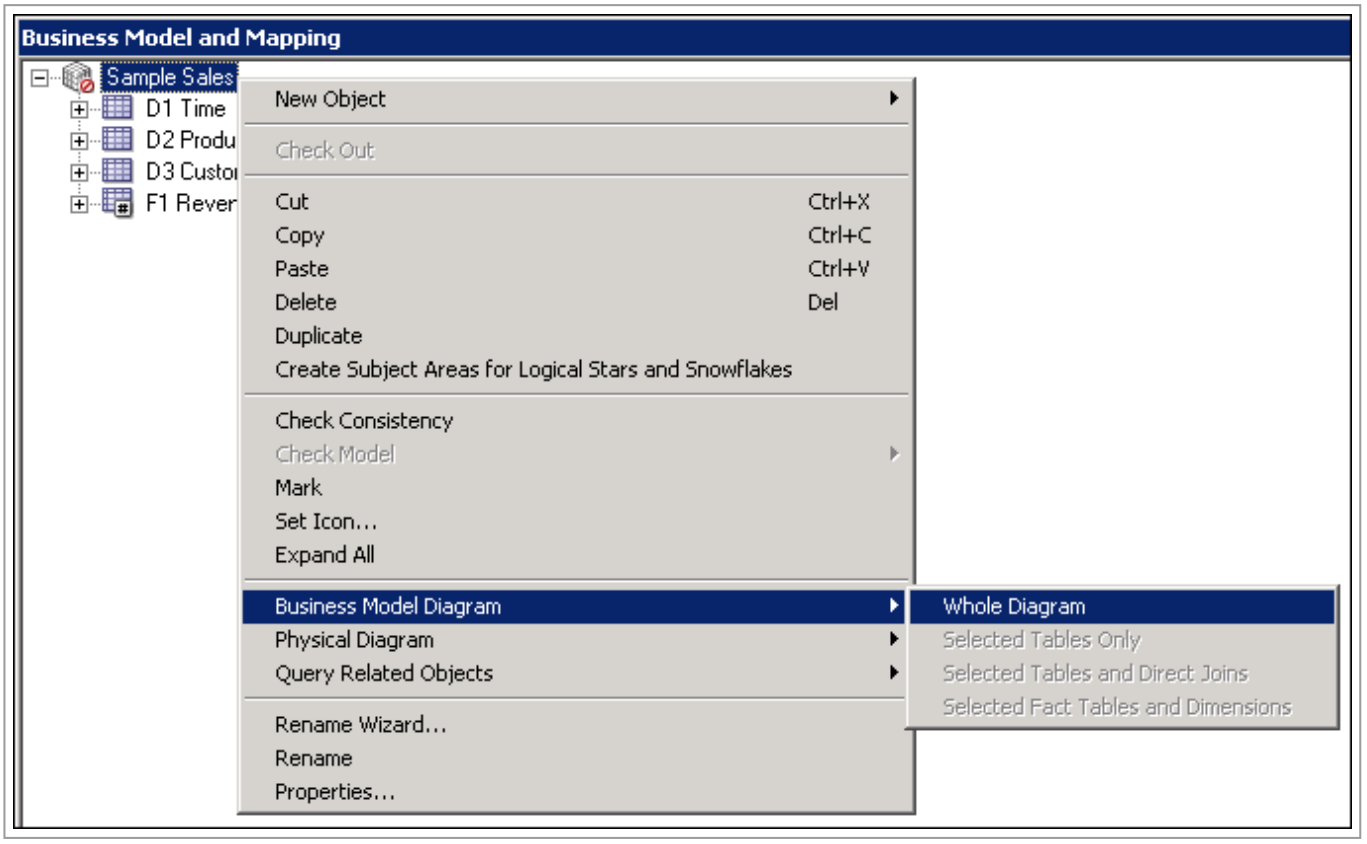



5. **Drag** the four alias table from the **Physical** layer to the **Sample Sales** business model in the **Business Model and Mapping** layer. The tables are added to the **Sample Sales** business model. Notice that the three dimension tables have the same icon, whereas the F1 Revenue table has an icon with a # sign, indicating it is a fact table.

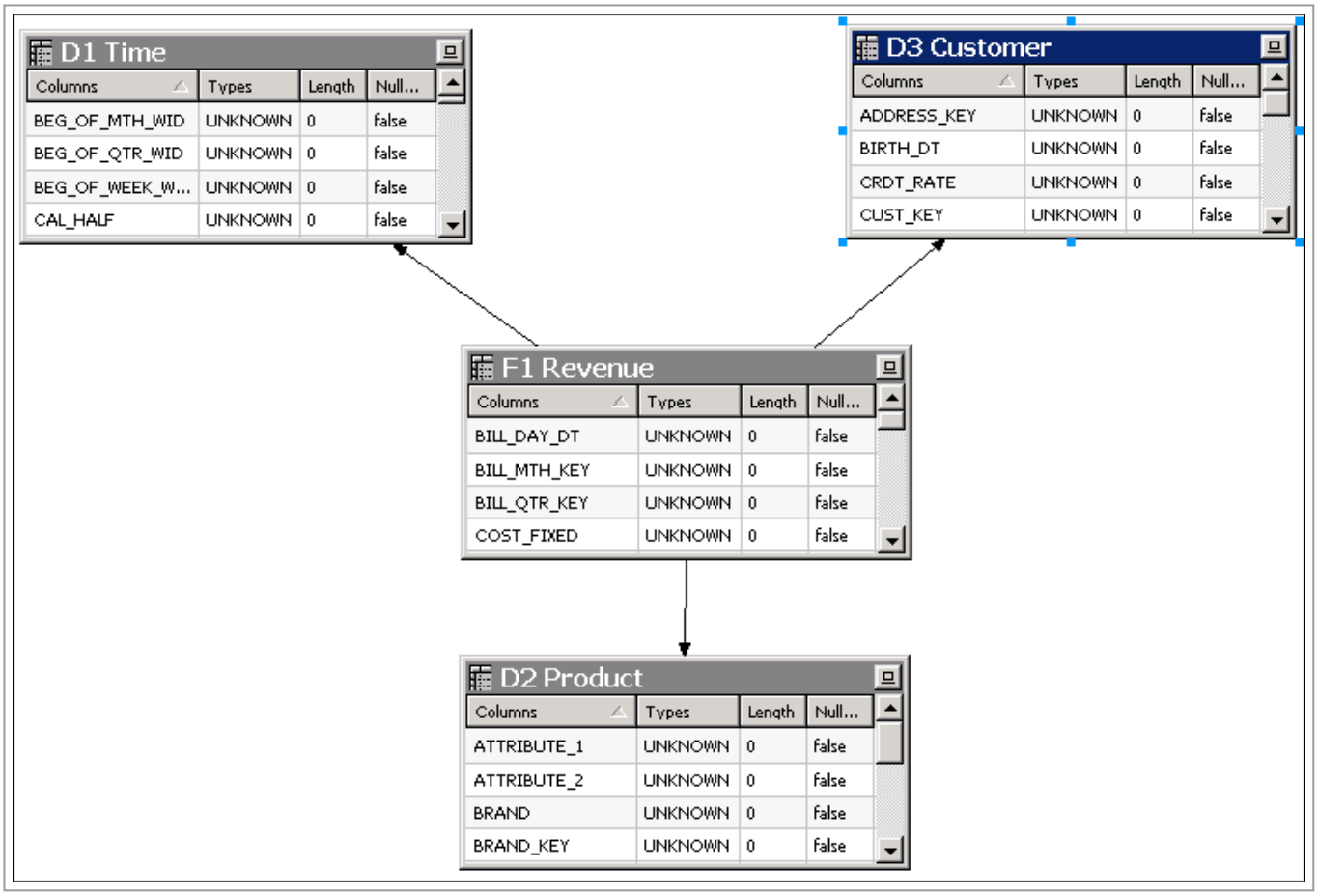



## Examining Logical Joins

1. Right-click the **Sample Sales** business model and select **Business Model Diagram > Whole Diagram** to open the **Business Model Diagram**.

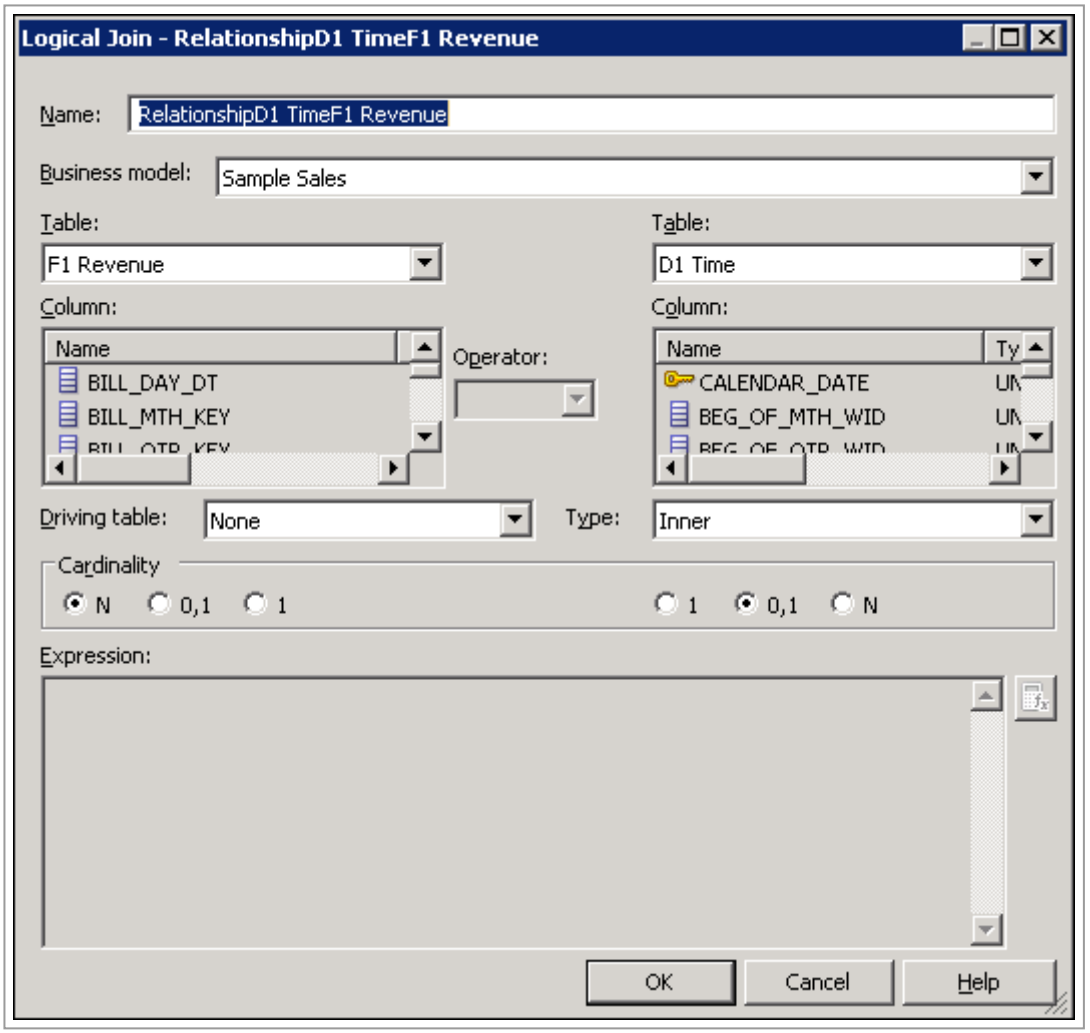



2. If necessary, rearrange the objects so that the join relationships are visible.




Because you dragged all tables simultaneously from the Physical layer onto the business model, the logical keys and joins are created automatically in the business model. This is because the keys and join relationships were already created in the Physical layer. However, you typically do not drag all physical tables simultaneously, except in very simple models. Later in this tutorial, you learn how to manually build logical keys and joins in the Business Model and Mapping layer. The process is very similar to building joins in the Physical layer.

3. Double-click any one of the joins in the diagram to open the **Logical Join** dialog box. In this example the join between **D1 Time** and **F1 Revenue** is selected.



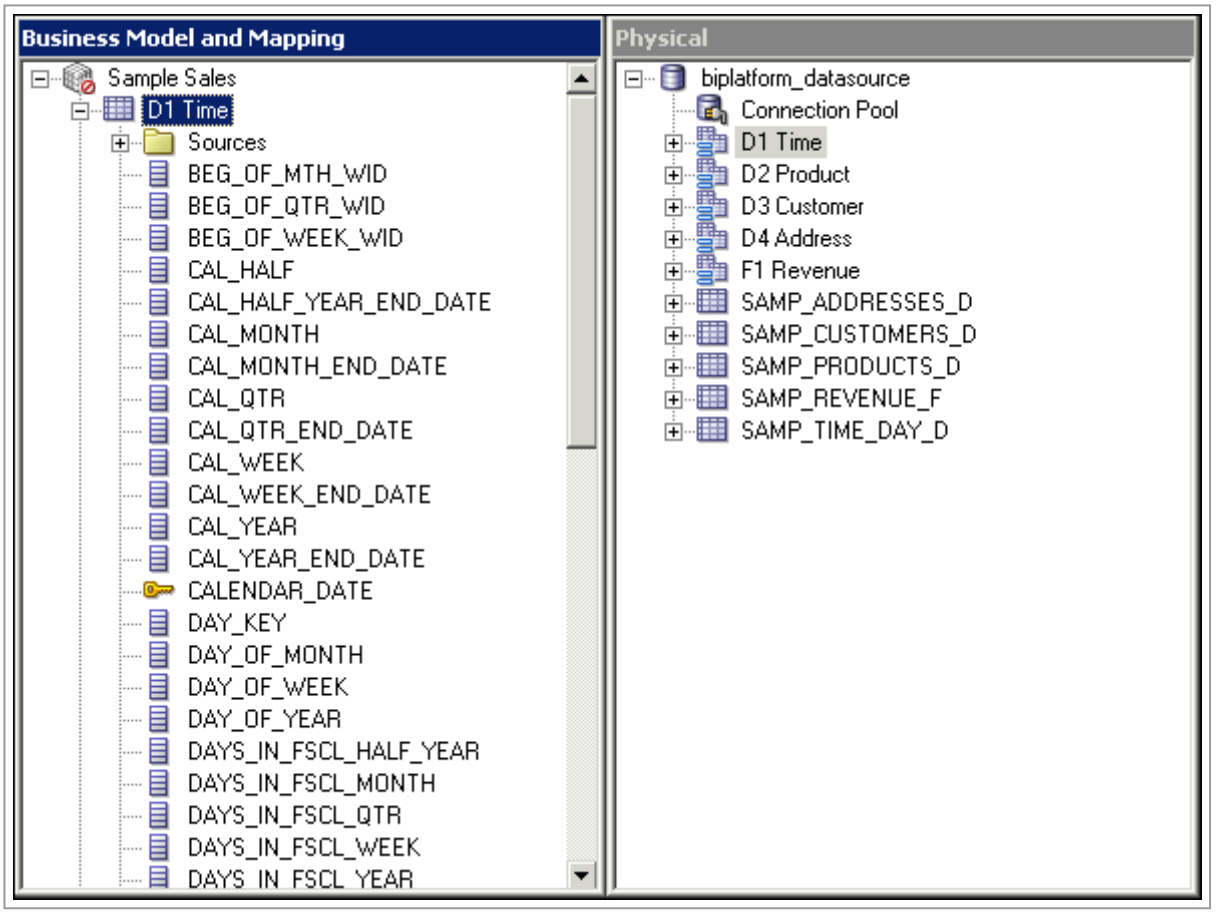

Notice that there is no join expression. Joins in the BMM layer are logical joins. Logical joins express the cardinality relationships between logical tables and are a requirement for a valid business model. Specifying the logical table joins is required so that Oracle BI Server has necessary metadata to translate logical requests against the business model into SQL queries against the physical data sources. Logical joins help Oracle BI Server understand the relationships between the various pieces of the business model. When a query is sent to Oracle BI Server, the server determines how to construct physical queries by examining how the logical model is structured. Examining logical joins is an integral part of this process. The Administration Tool considers a table to be a logical fact table if it is at the “many” end of all logical joins that connect it to other logical tables.

4. Click **OK** to close the Logical Join dialog box.

5. Click the **X** to close the **Business Model Diagram**.

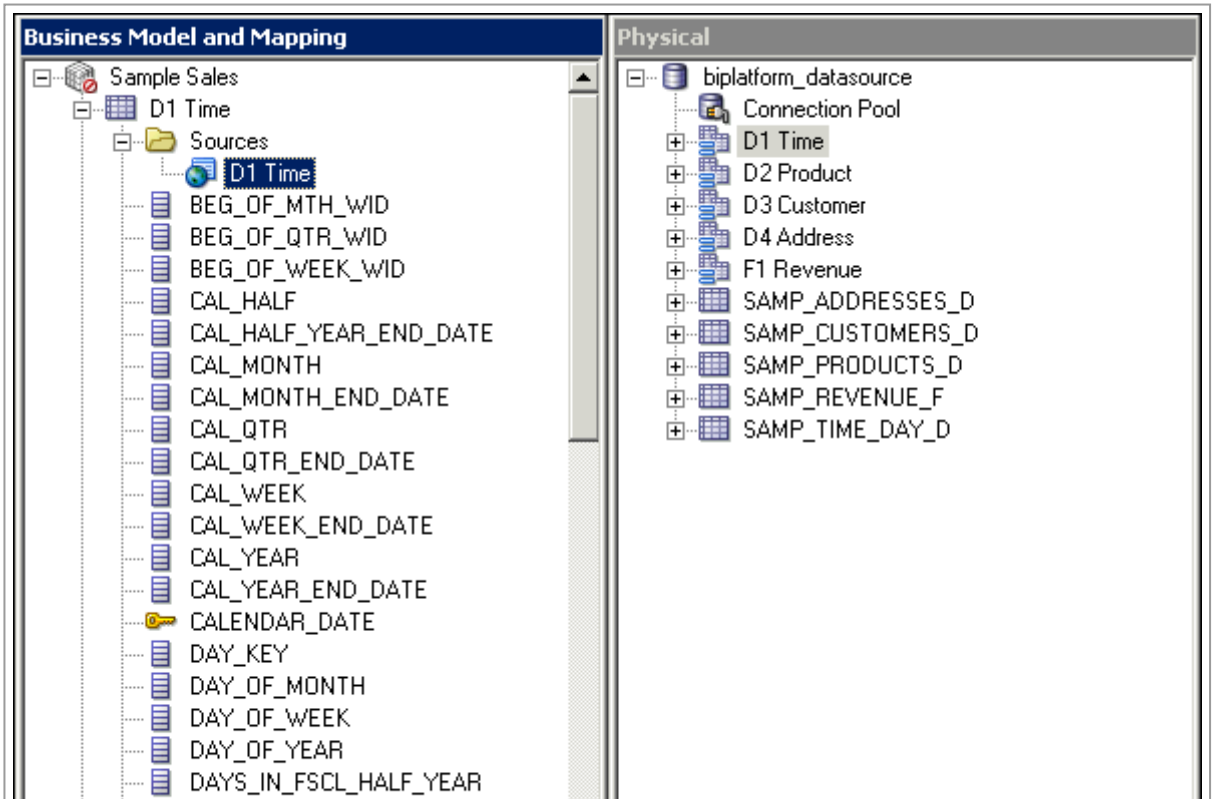
## Examining Logical Columns

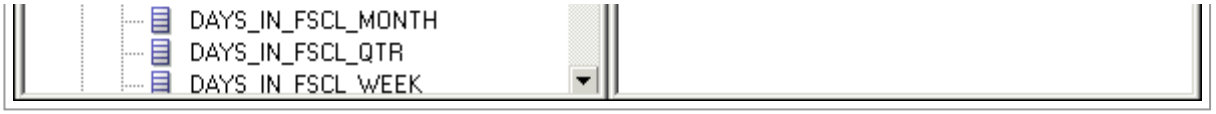
1. Expand the **D1 Time** logical table. Notice that logical columns were created automatically for each table when you dragged the alias tables from the **Physical** layer to the **BMM** layer.



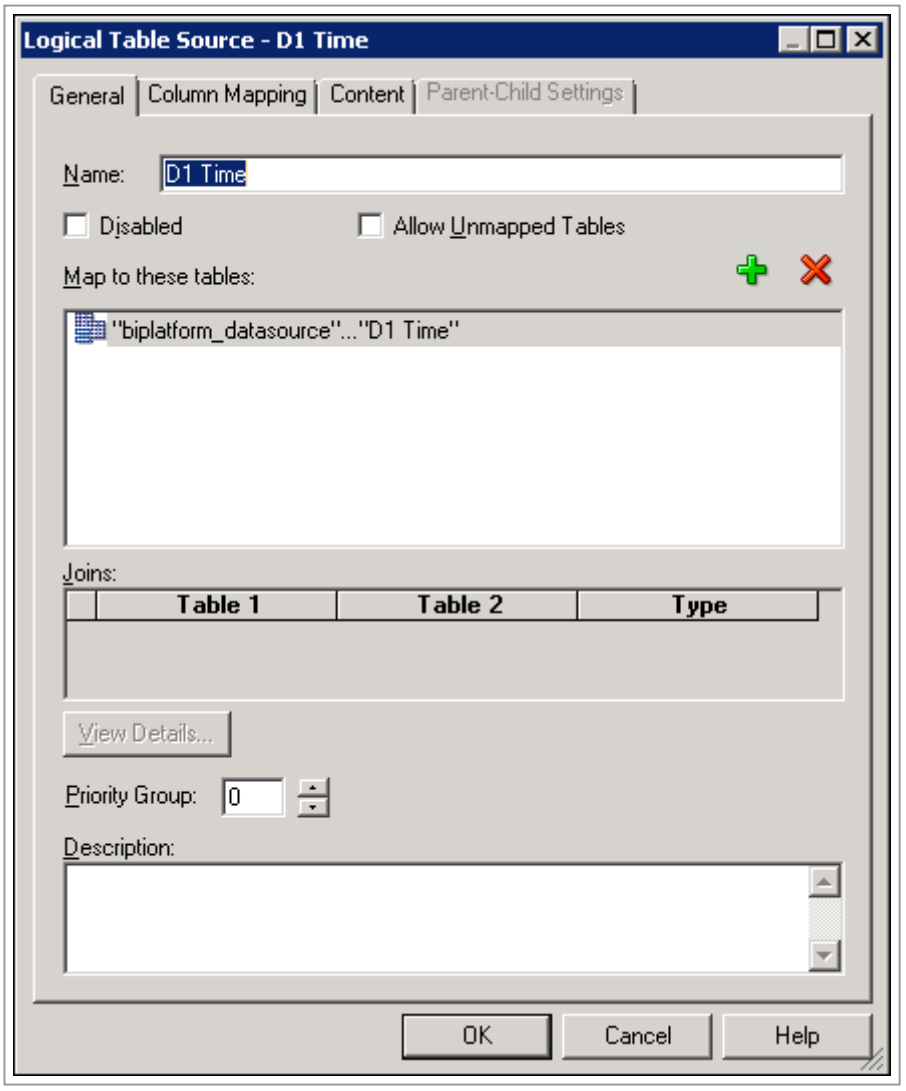

## Examining Logical Table Sources

1. Expand the **Sources** folder for the **D1 Time** logical table. Notice there is a logical table source, **D1 Time**. This logical table source maps to the **D1 Time** alias table in the **Physical** layer.

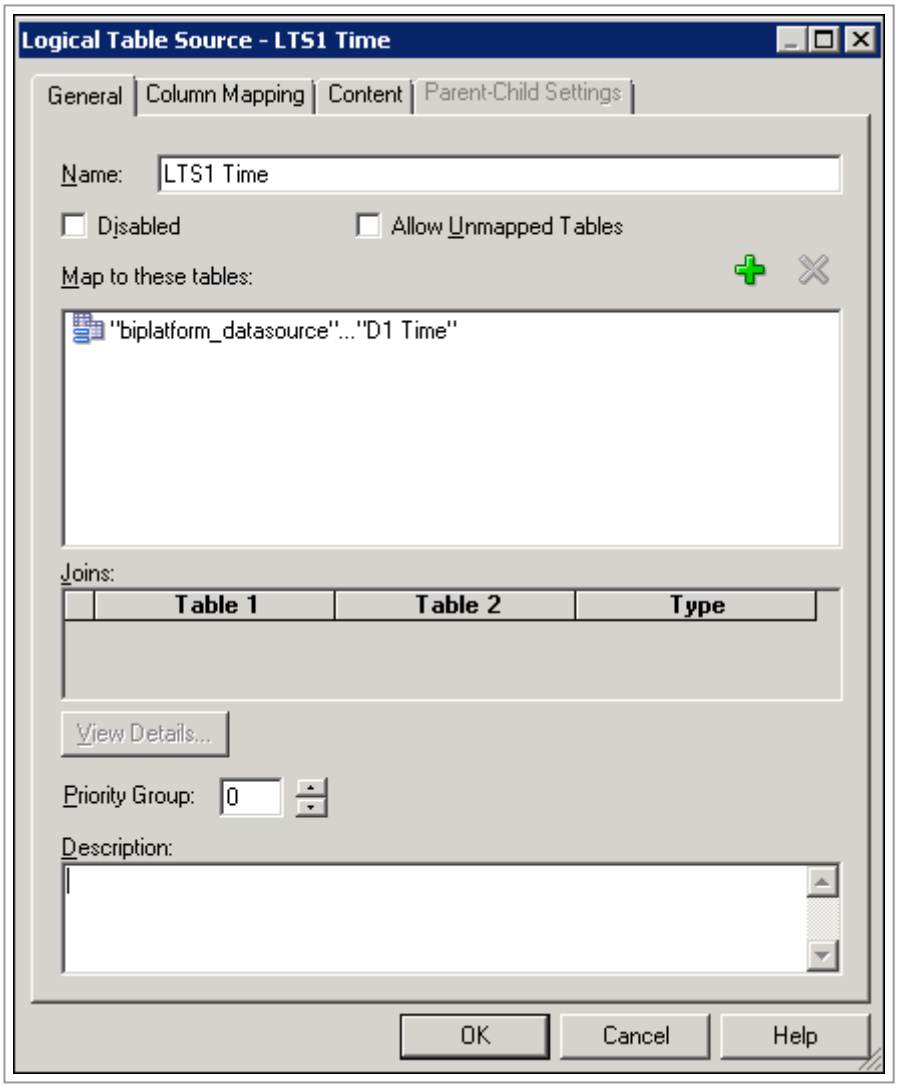




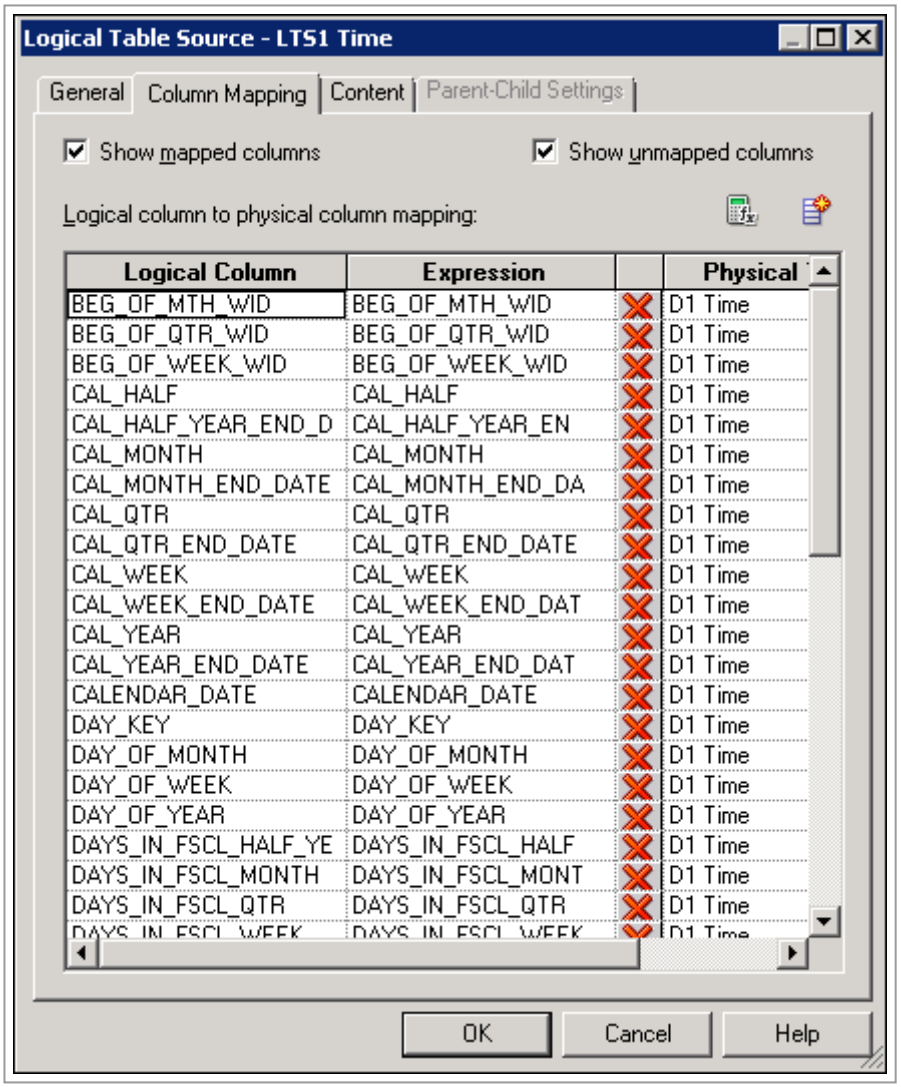

2. Double-click the **D1 Time** logical table source (not the logical table) to open the **Logical Table Source** dialog box.



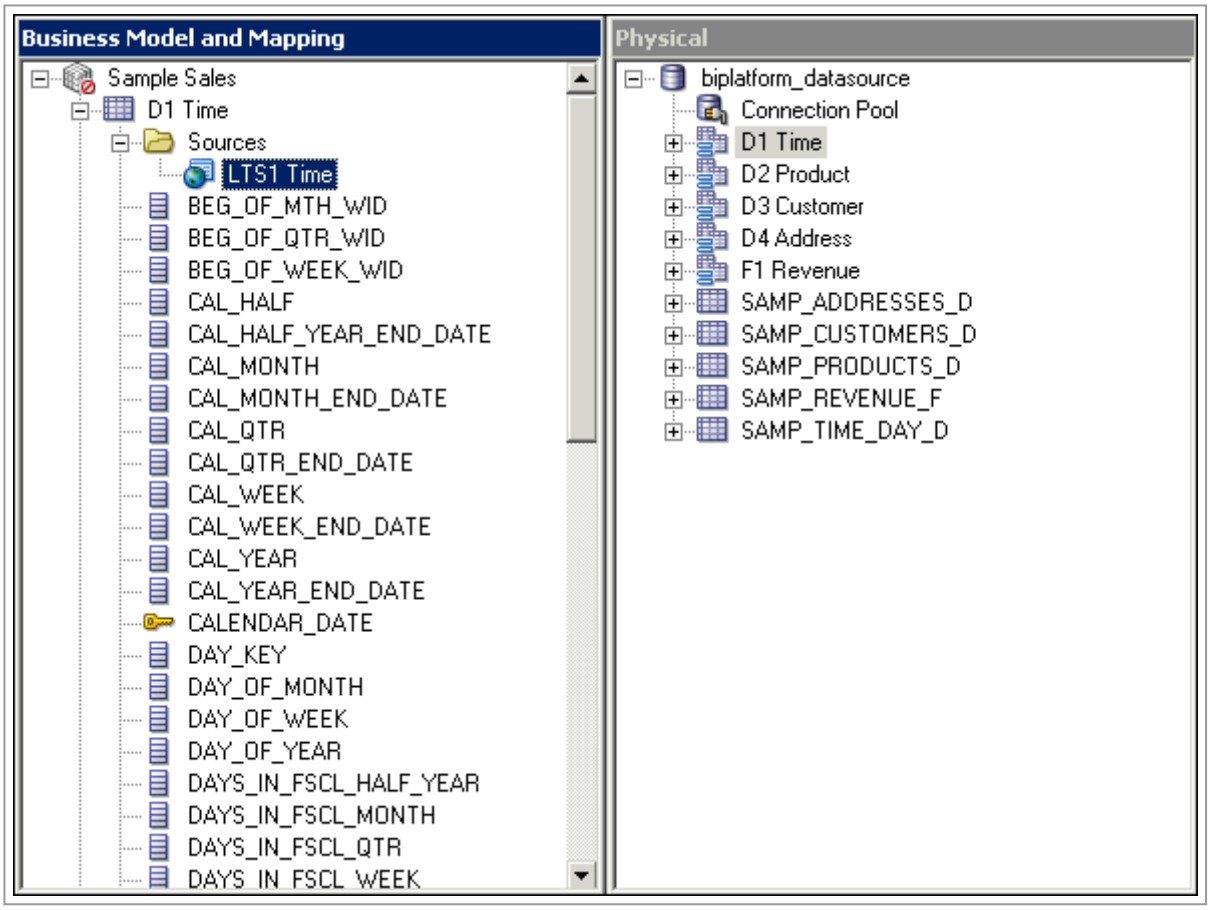

3. On the General tab, rename the **D1 Time** logical table source to **LTS1 Time**. Notice that the logical table to physical table mapping is defined in the "Map to these tables" section.




4. On the Column Mapping tab, notice that logical column to physical column mappings are defined. If mappings are not visible, select **Show mapped columns**.

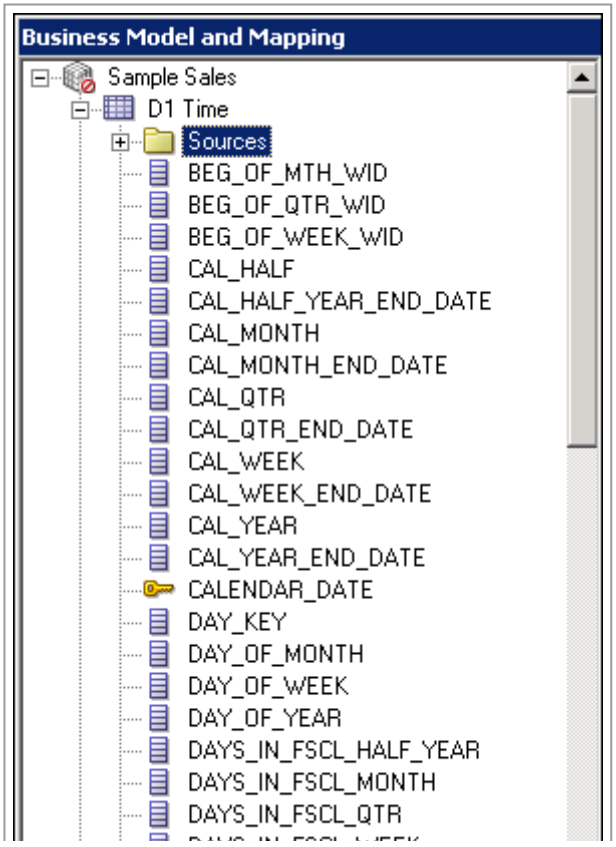


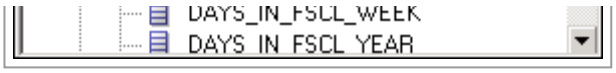

5. Click **OK** to close the Logical Table Source dialog box. If desired, explore logical table sources for the remaining logical tables.



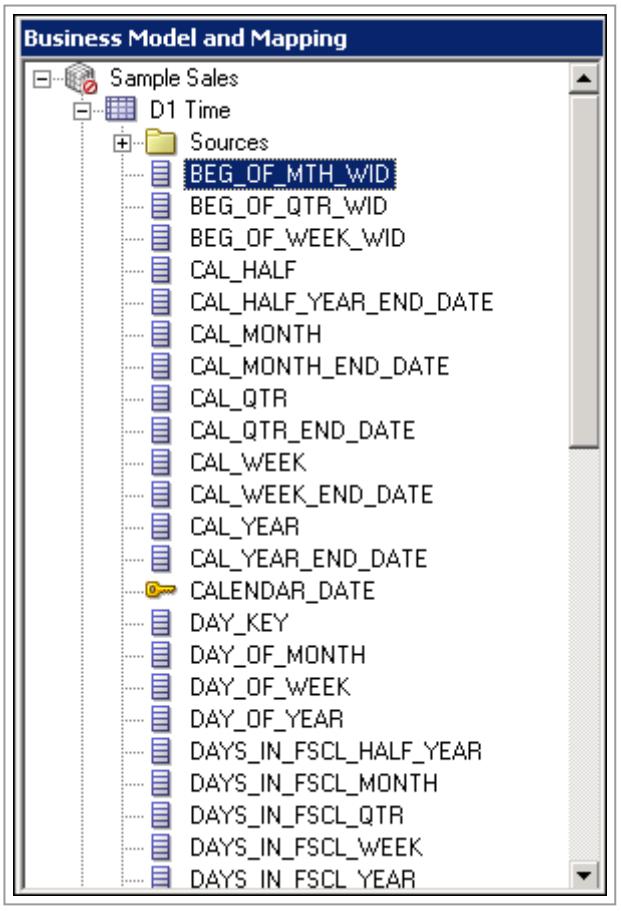

## Renaming Logical Objects Manually

**1.** Expand the **D1 Time** logical table.

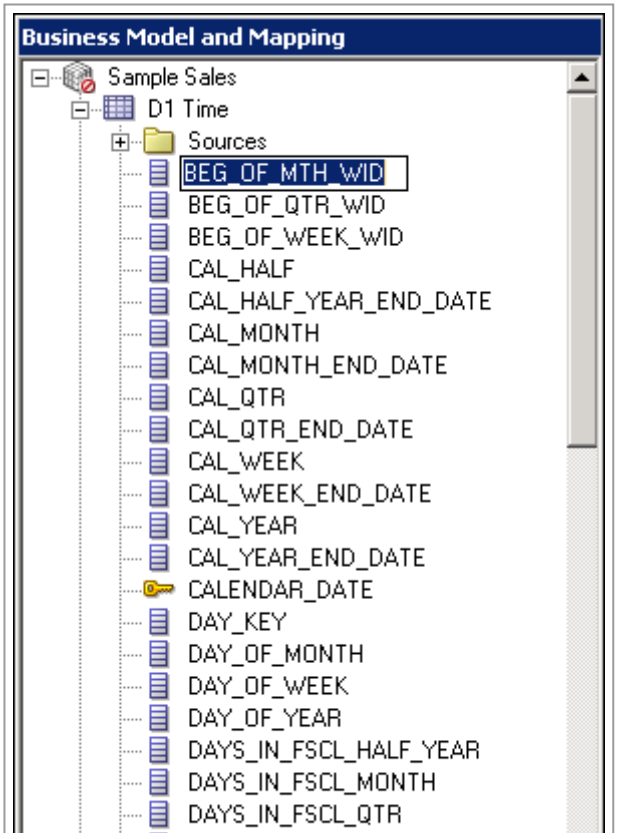


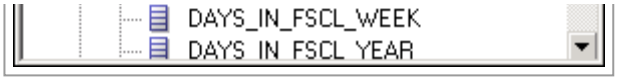



**2.** Click on the first logical column, **BEG_OF_MONTH_WID**, to highlight it.

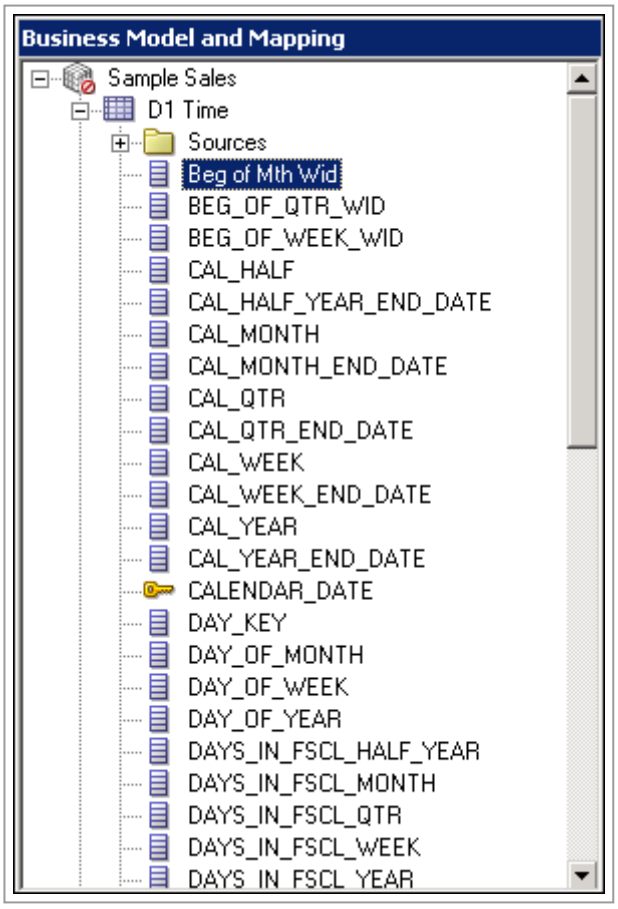



**3.** Click on **BEG_OF_MONTH_WID** again to make it editable.



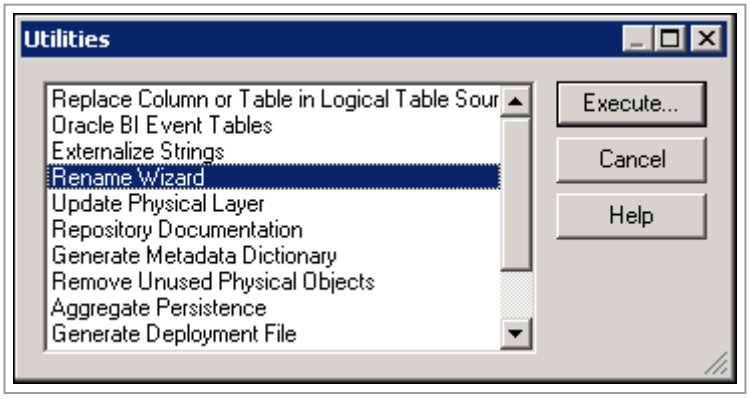



4. Rename **BEG_OF_MONTH_WID** to **Beg of Mth Wid**. This is the manual method for renaming objects. You can also rename an object and select **Rename** to manually rename an object.

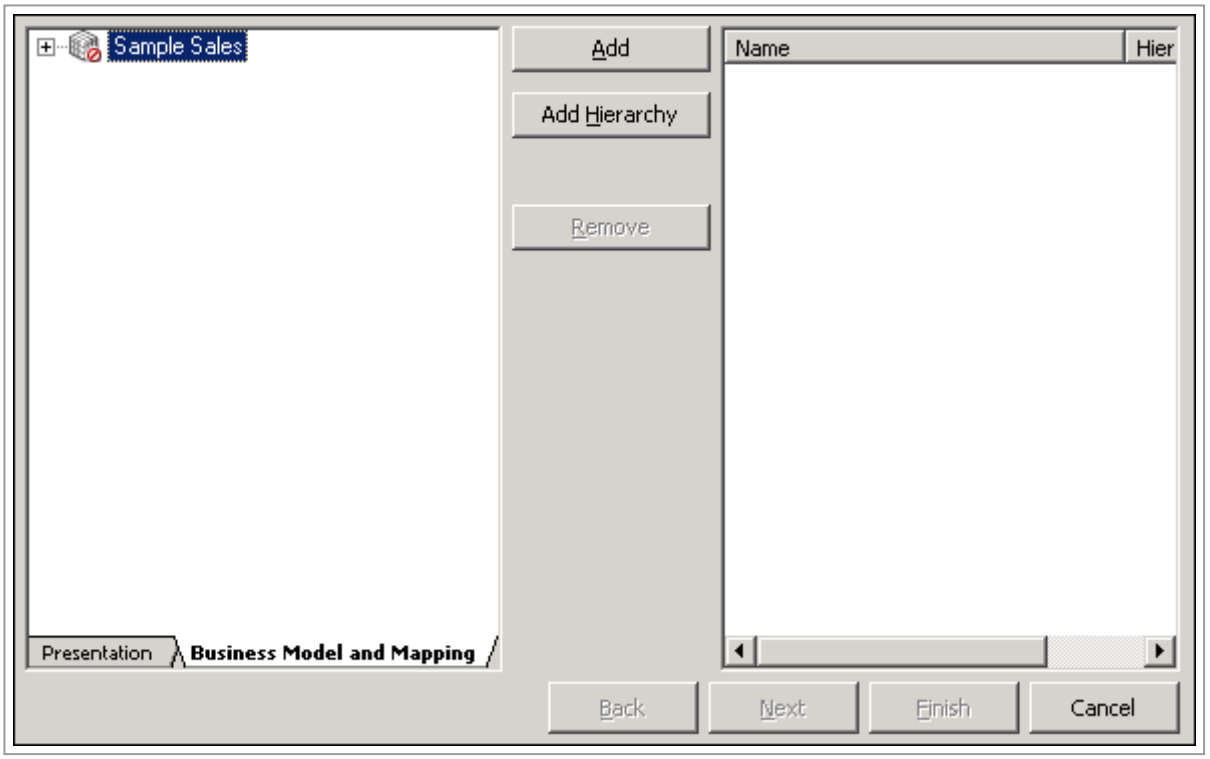



## Renaming Objects Using the Rename Wizard

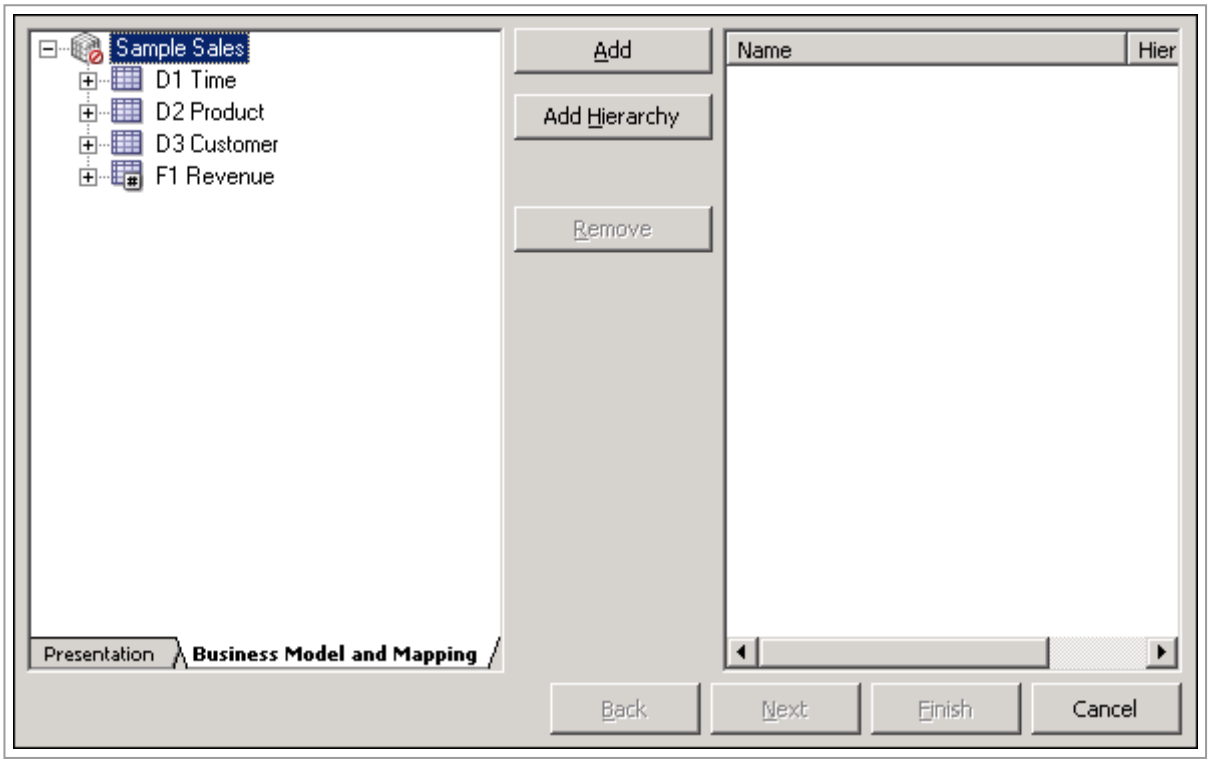
1. Select **Tools > Utilities > Rename Wizard > Execute** to open the Rename Wizard.



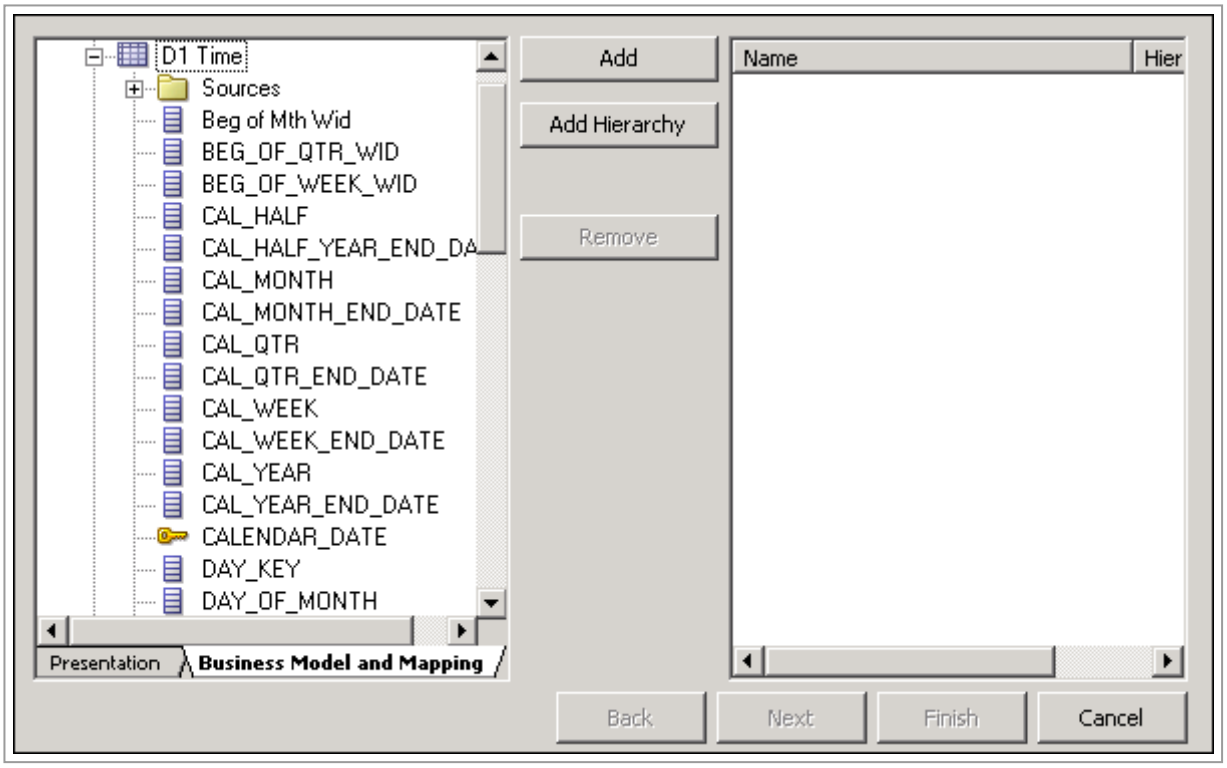

2. In the Select Objects screen, click **Business Model and Mapping** in the middle pane.



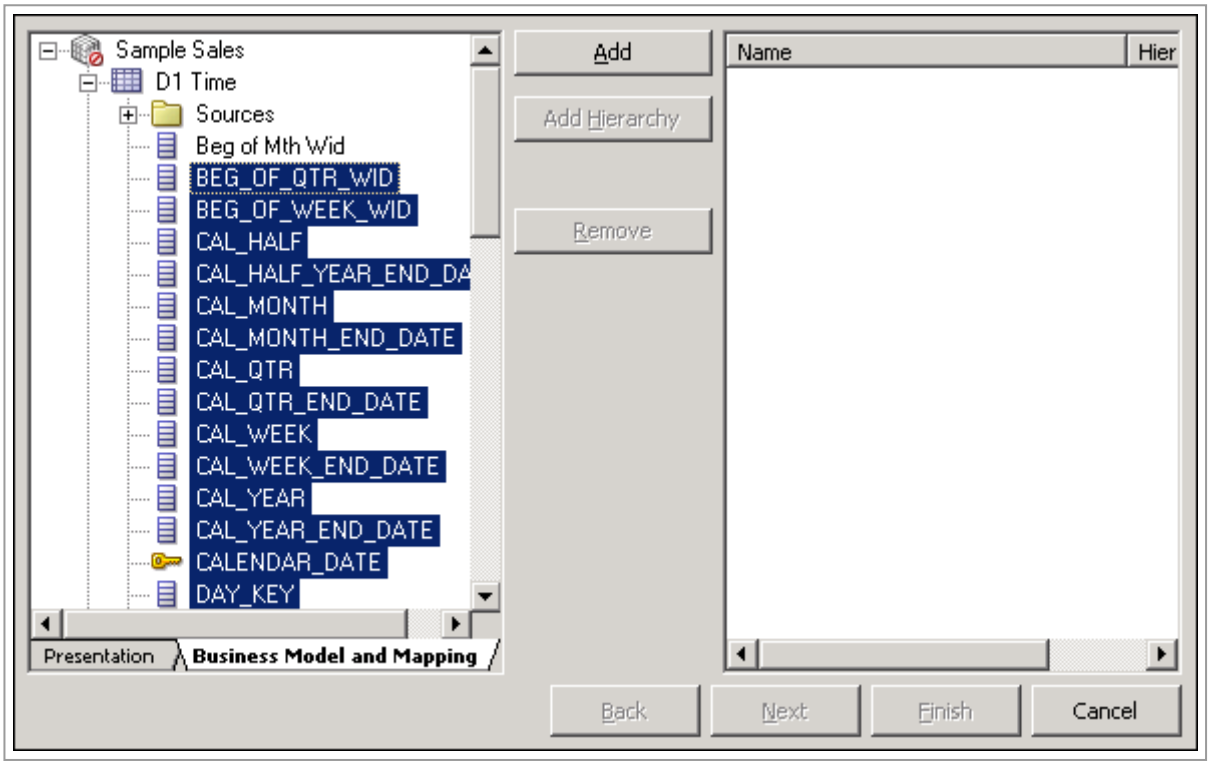

**3.** Expand the **Sample Sales** business model.



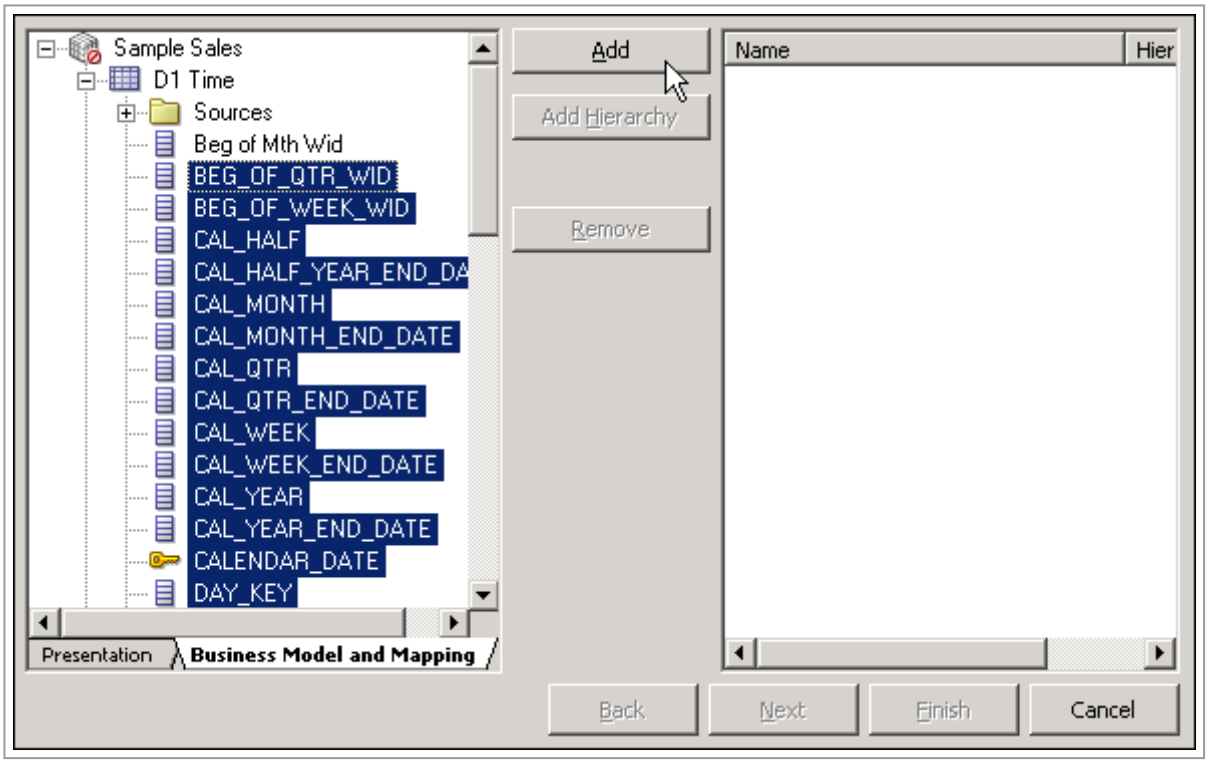

**4.** Expand the **D1 Time** logical table.



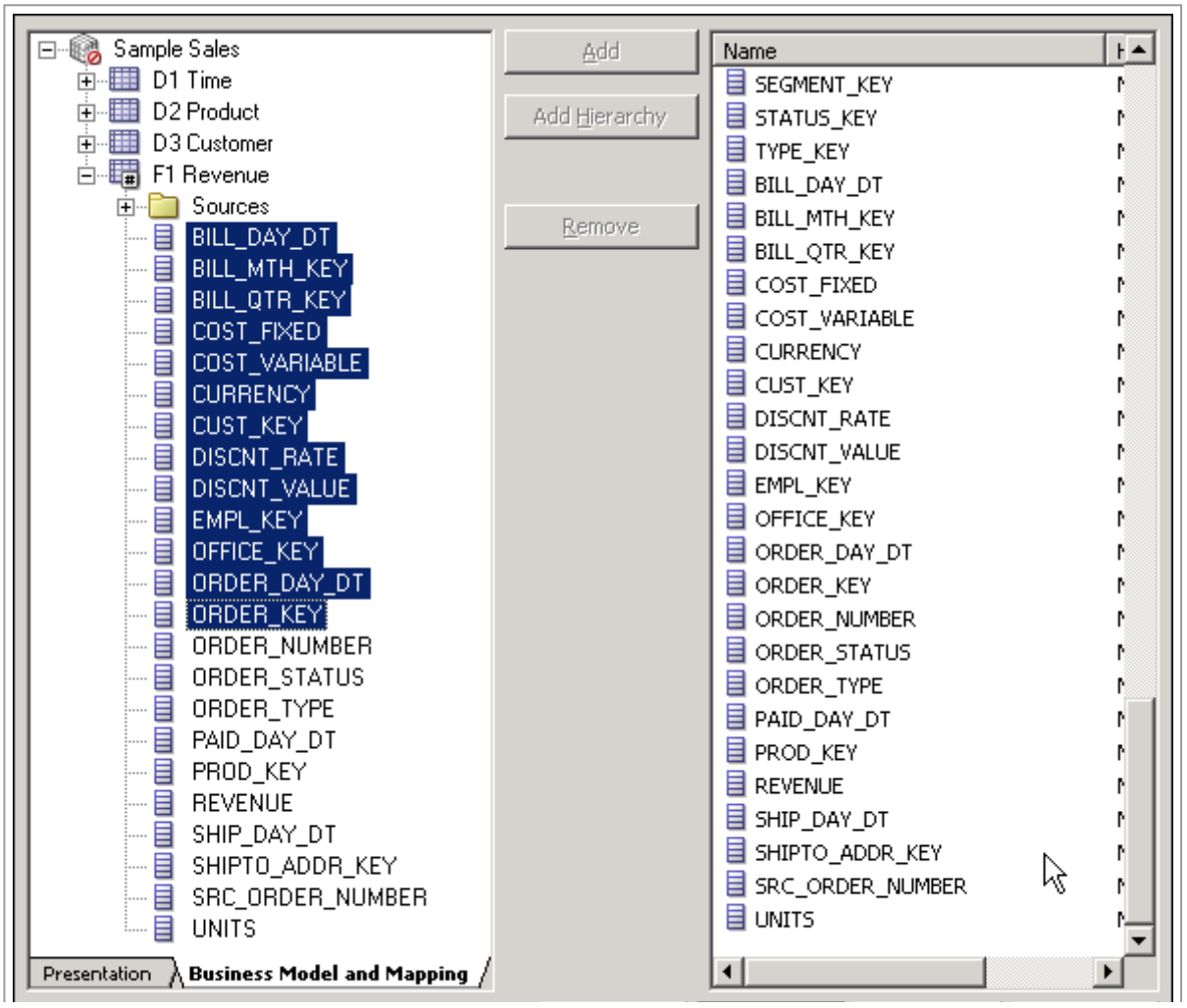

**5.** Select all of the logical columns except for the column you already renamed, **Beg of Mth Wid**.



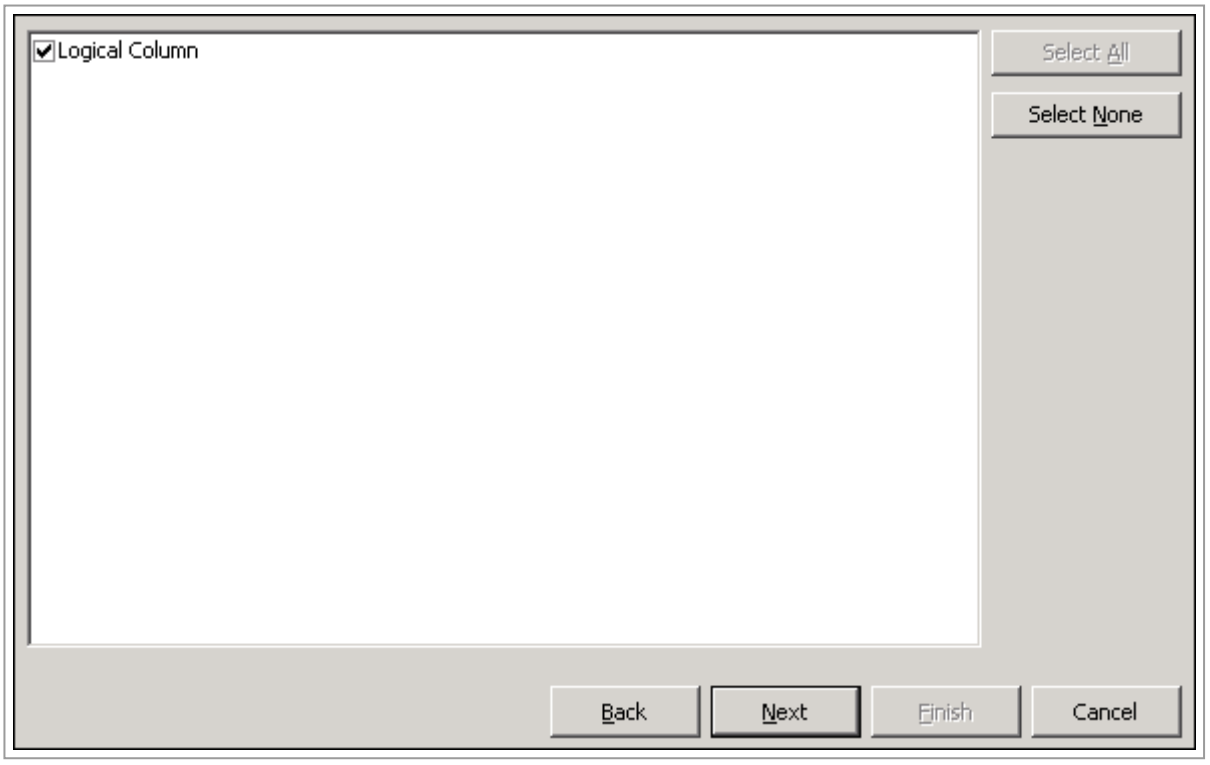

**6.** Click **Add** to add the columns to the right pane.




7. Repeat the steps for the three remaining logical tables so that all logical columns from the **Sample Sales** business model are added to the right pane. Only the columns from **F1 Revenue** are shown in the screenshot.

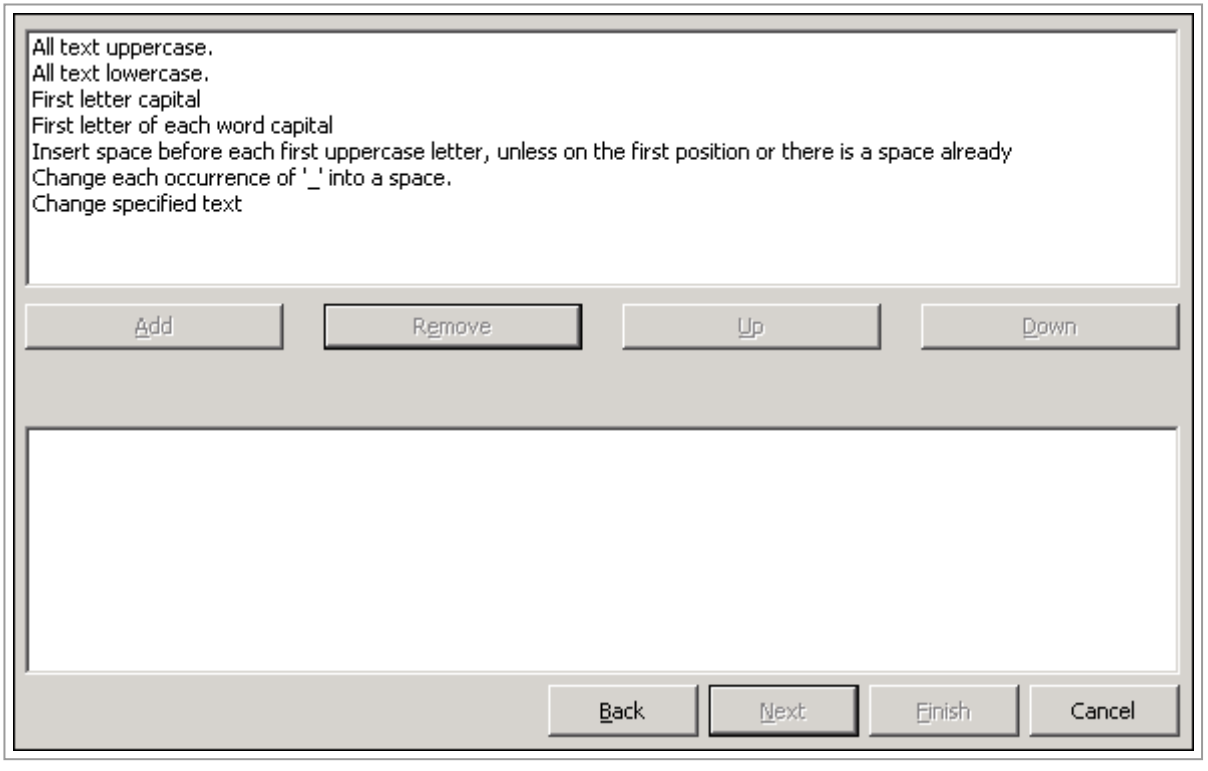



8. Click **Next** to move to the Select Types screen.

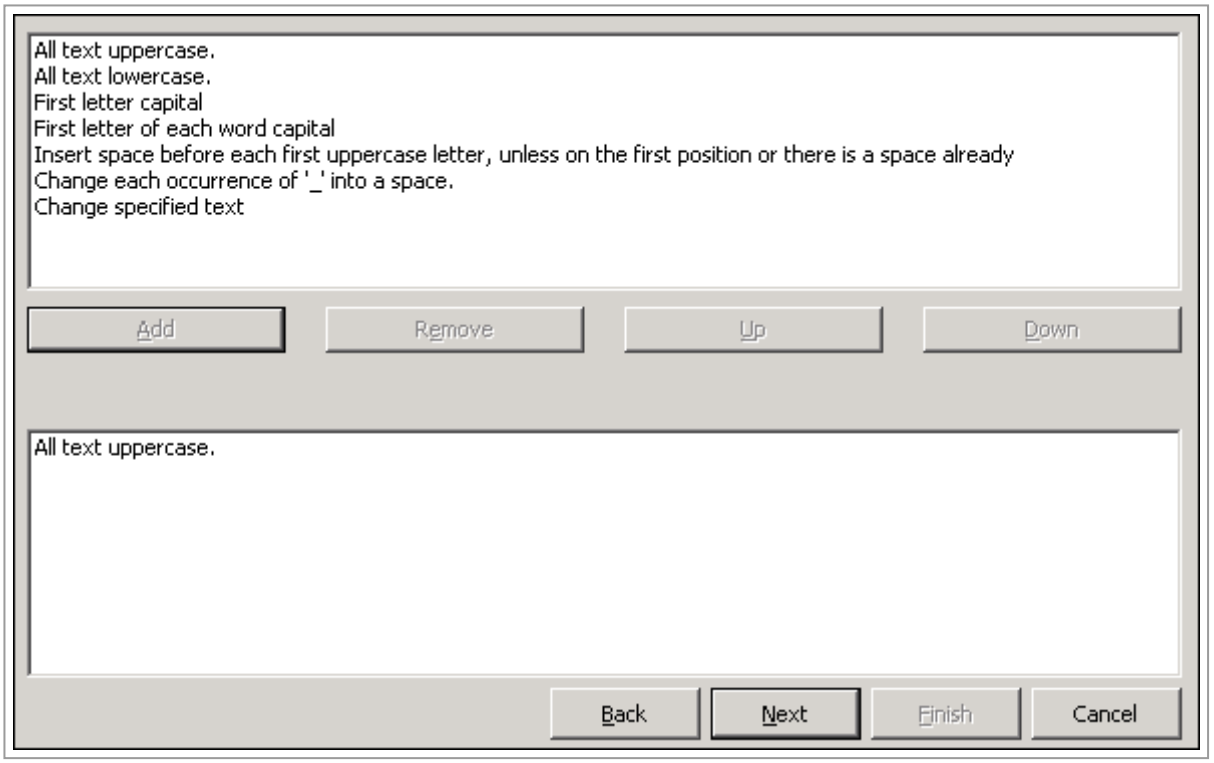



Notice that Logical Column is selected. If you had selected other object types, such as logical tables, the type would have appeared here.

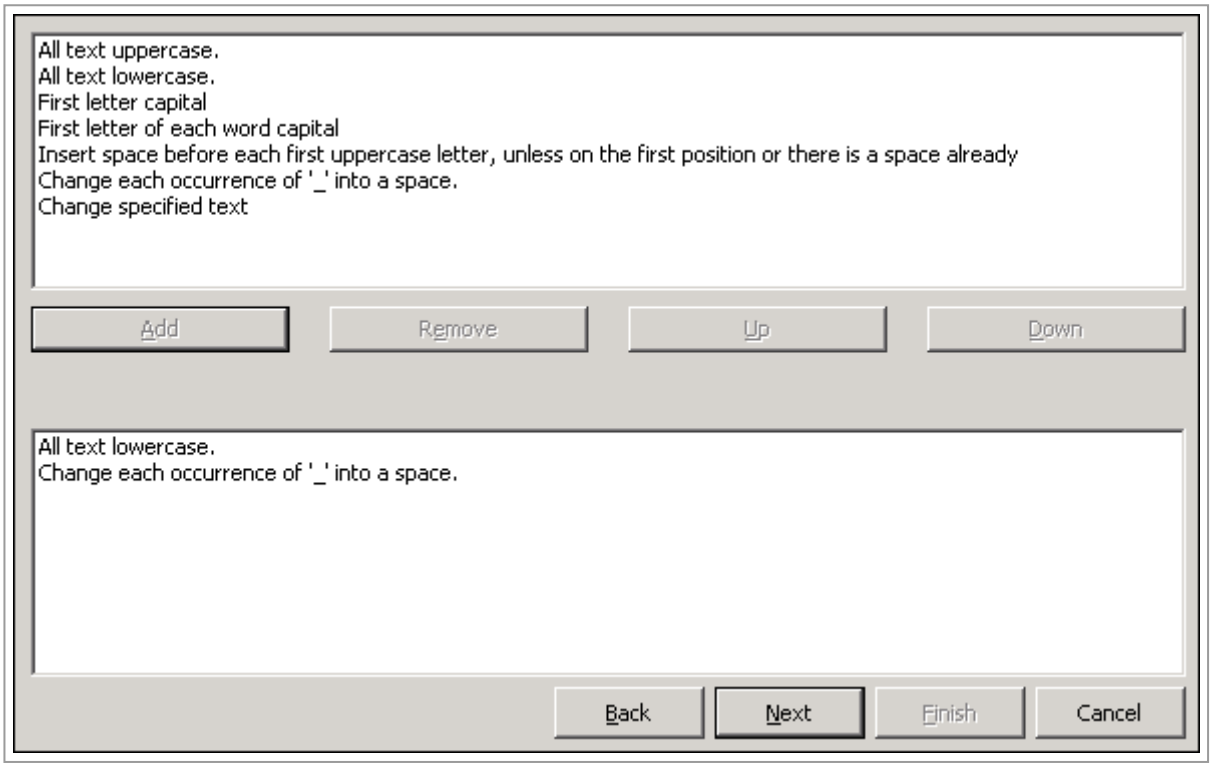
9. Click **Next**.



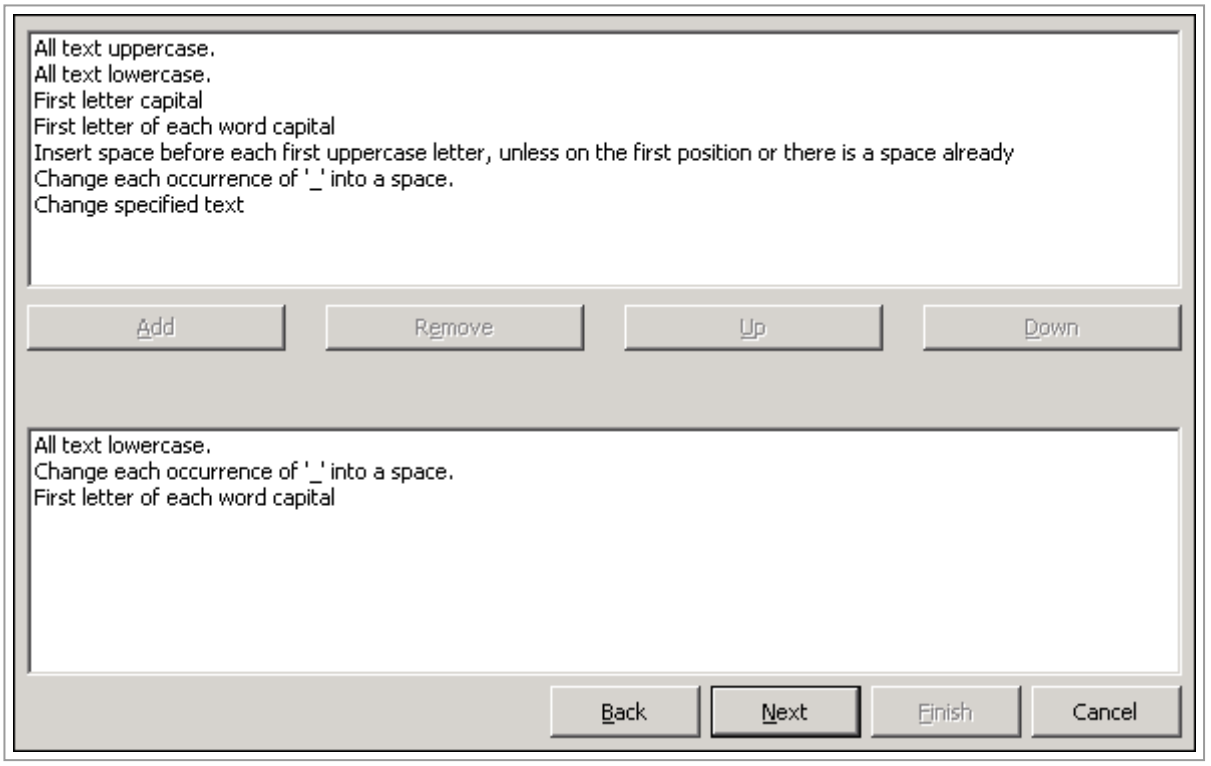

10. In the Select Rules screen, select **All text lowercase** and click **Add** to add the rule to the lower pane.



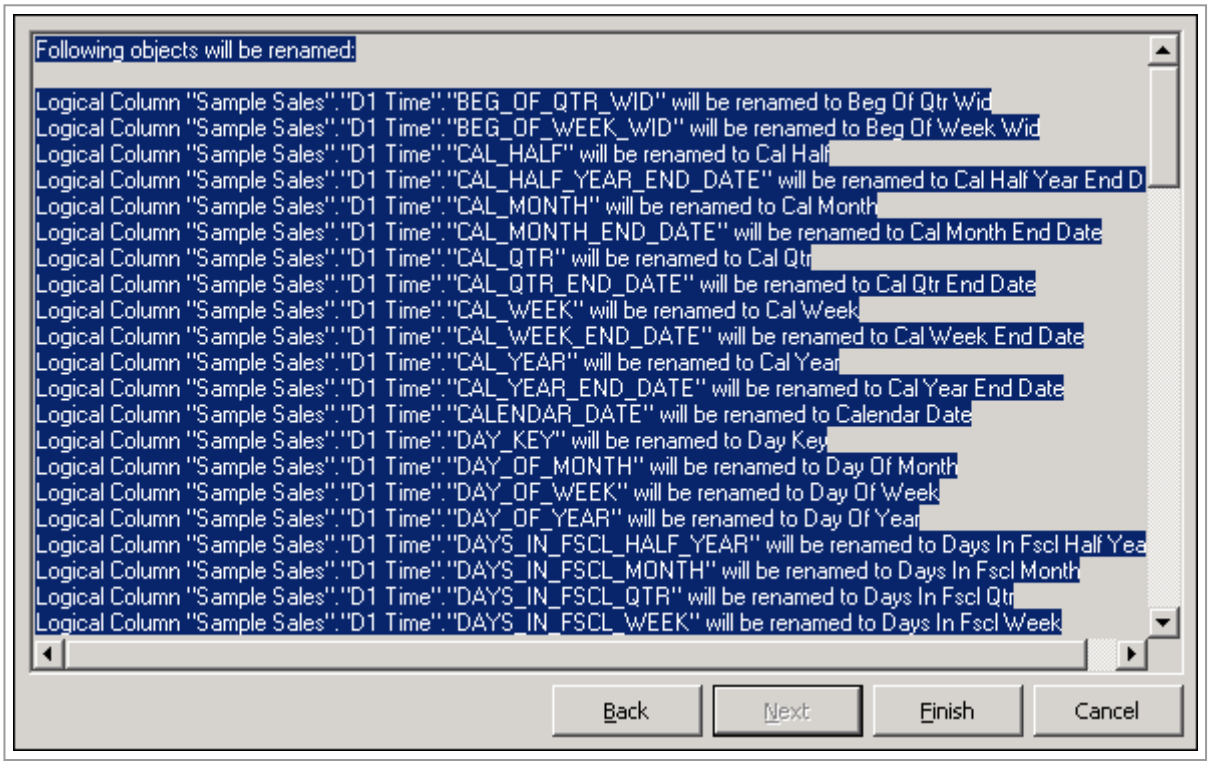

**11.** Add the rule **Change each occurrence of '_' into a space**.




**12.** Add the rule **First letter of each word capital**.

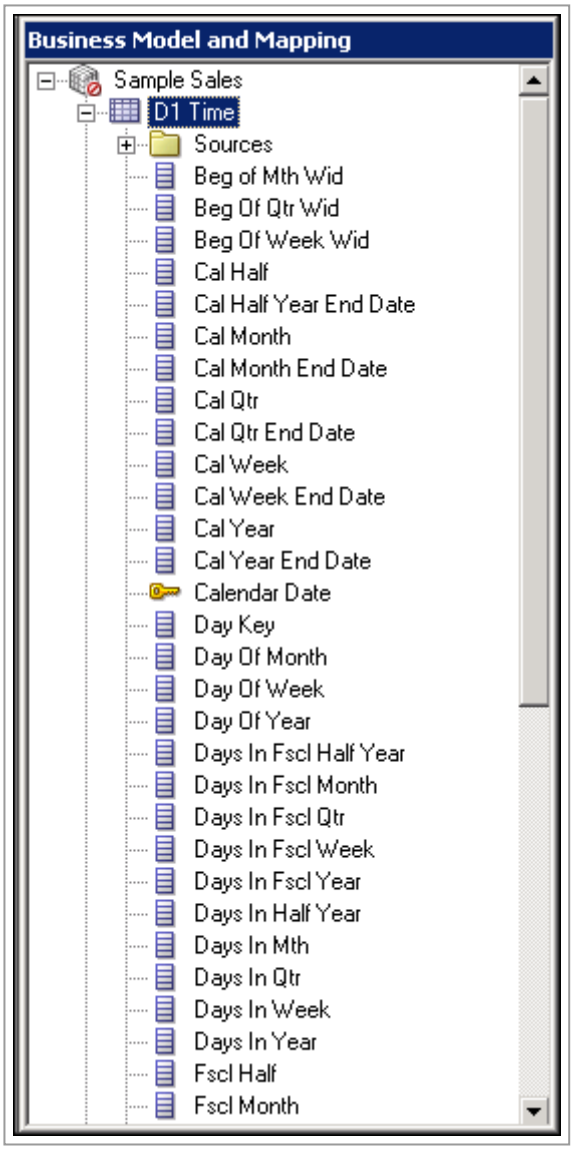



**13.** Click **Next** to open the Finish screen. Verify that all logical columns will be named according to the rename rules you selected.

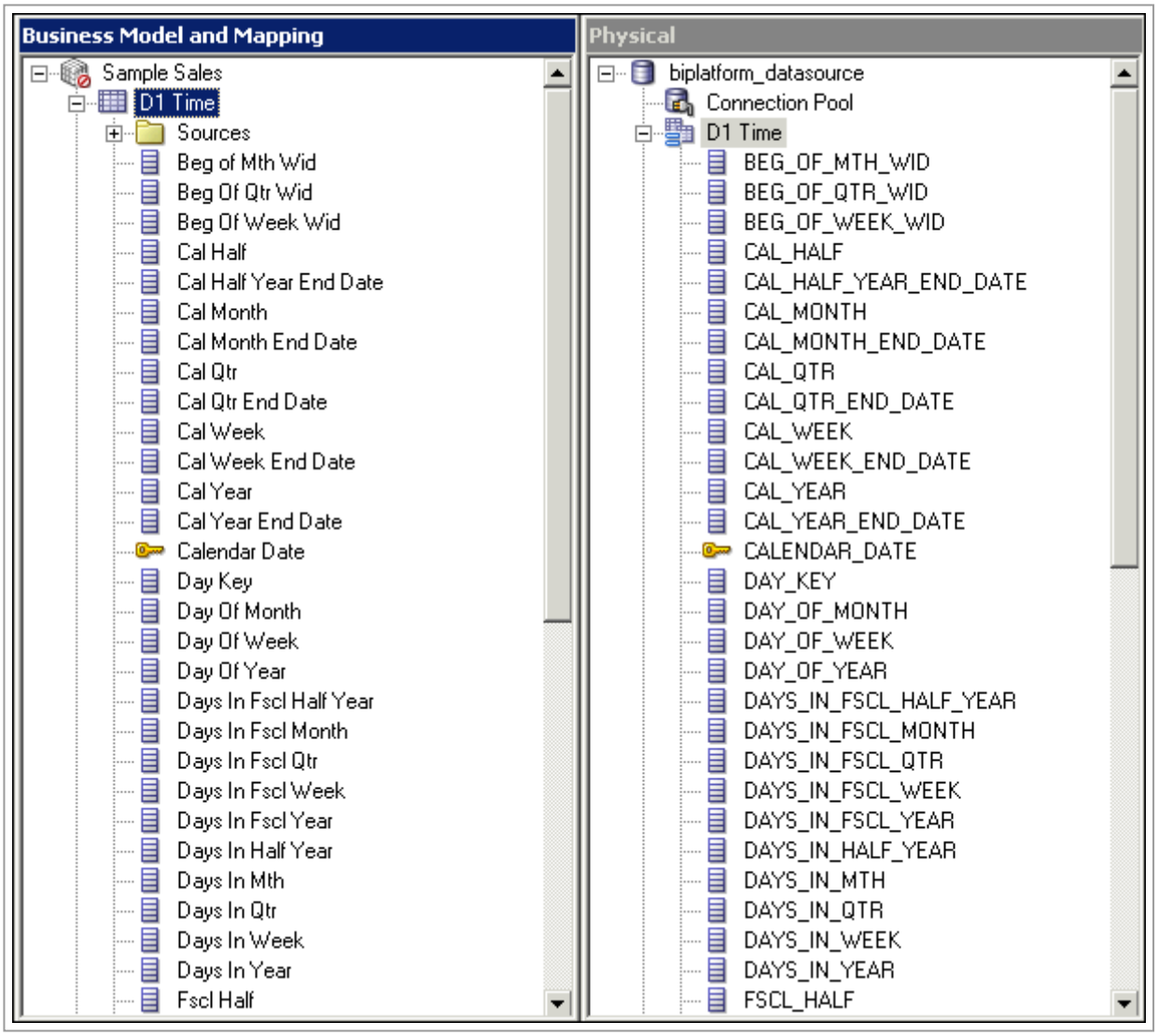



**14.** Click **Finish**.

**15.** In the Business Model and Mapping layer, expand the logical tables and confirm that all logical columns have been renamed as expected. The screenshot shows only the columns in D1 Time.

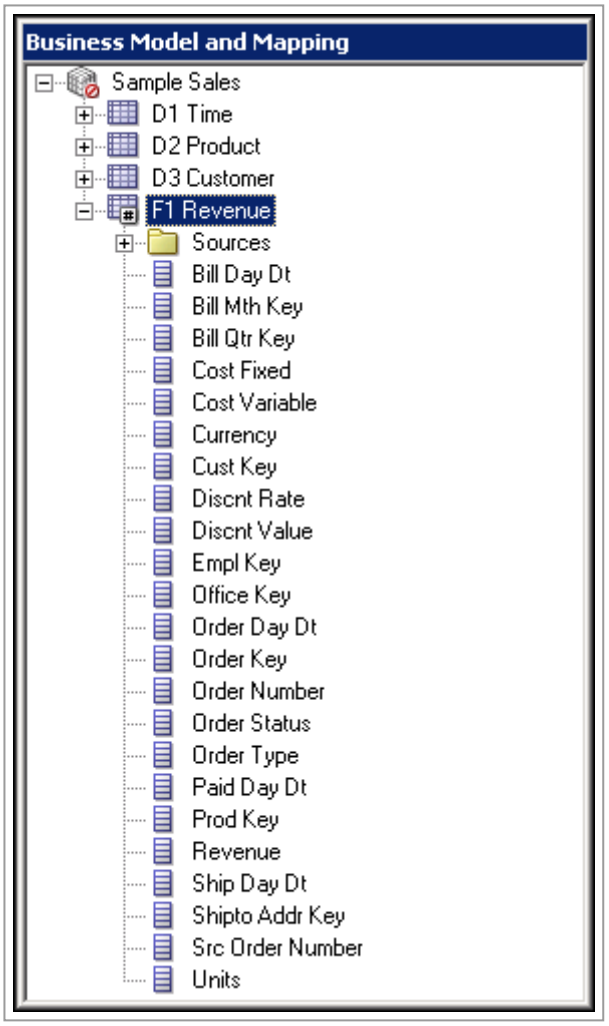



16. In the Physical layer, expand the alias tables and confirm that all physical columns have *not* been renamed. The point here is you can change object names in the **BMM** layer without impacting object names in the **Physical** layer. When logical objects are renamed, the relationships between logical objects and physical objects are maintained by the logical column to physical column mappings.

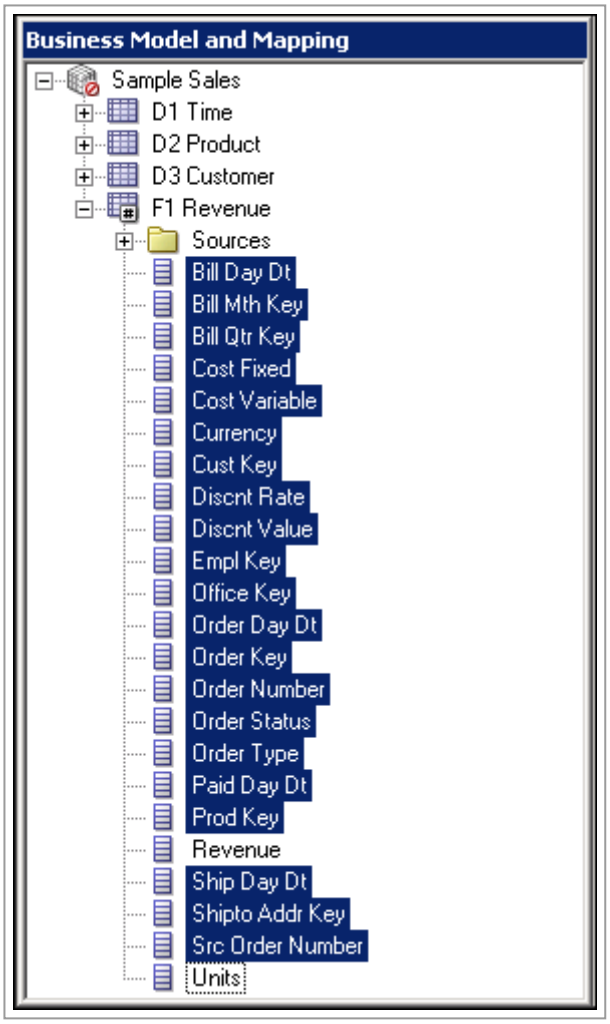



## Deleting Unnecessary Logical Objects

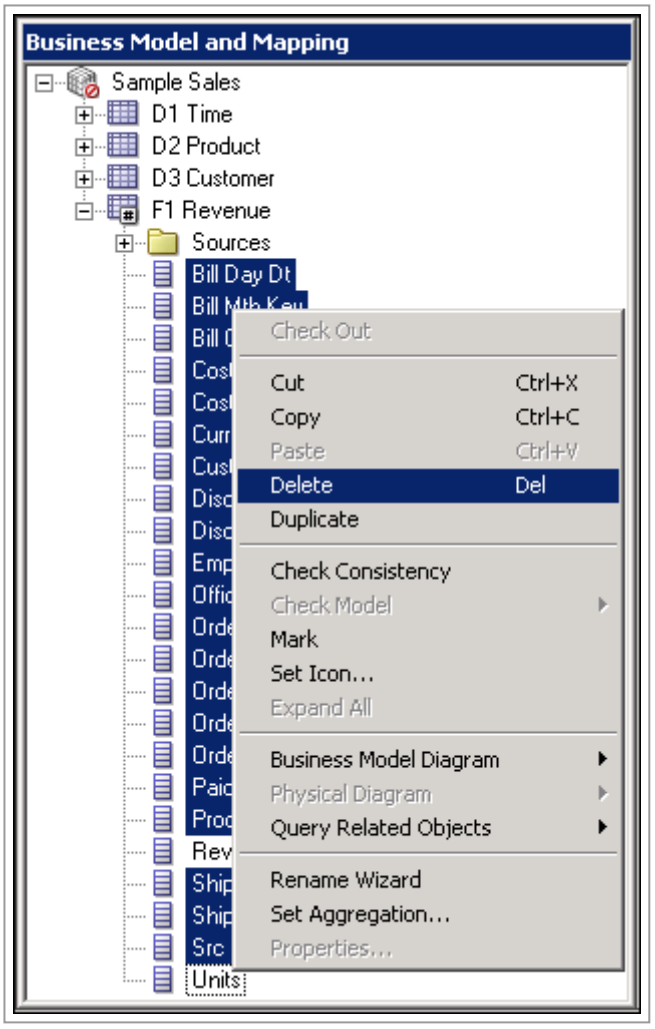
1. In the **BMM** layer, expand **Sample Sales > F1 Revenue**.



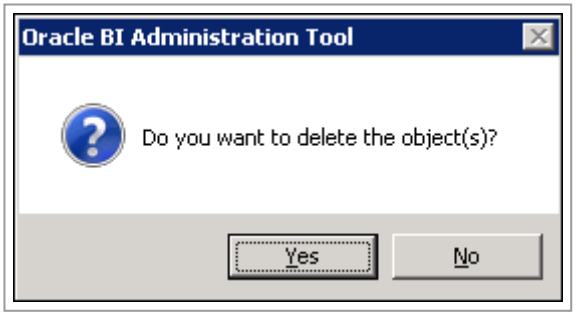

**2.** Select all **F1 Revenue** logical columns except for **Revenue** and **Units**.



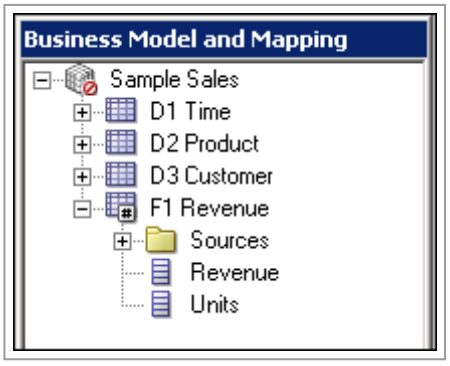

3. Right-click any one of the highlighted logical columns and select **Delete**. Alternatively you can select **Edit > Delete** or press the Delete key on your keyboard.




**4.** Click **Yes** to confirm the delete.

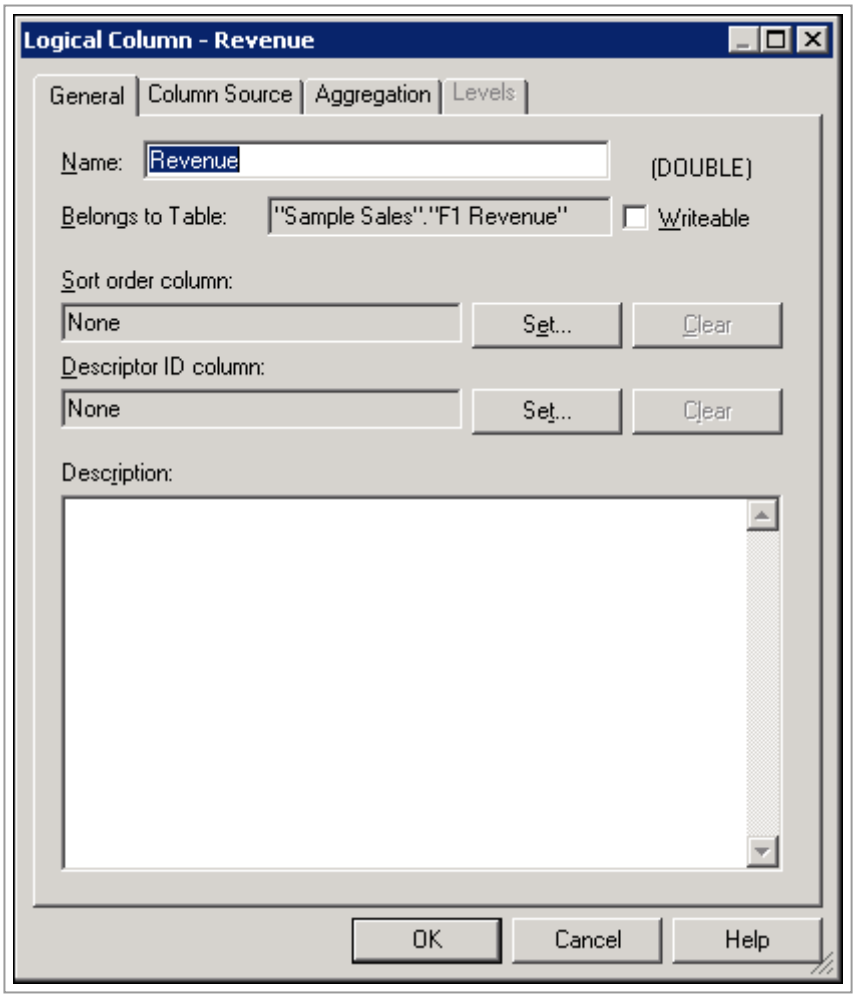



**5.** Confirm that **F1 Revenue** contains only the **Revenue** and **Units** columns.

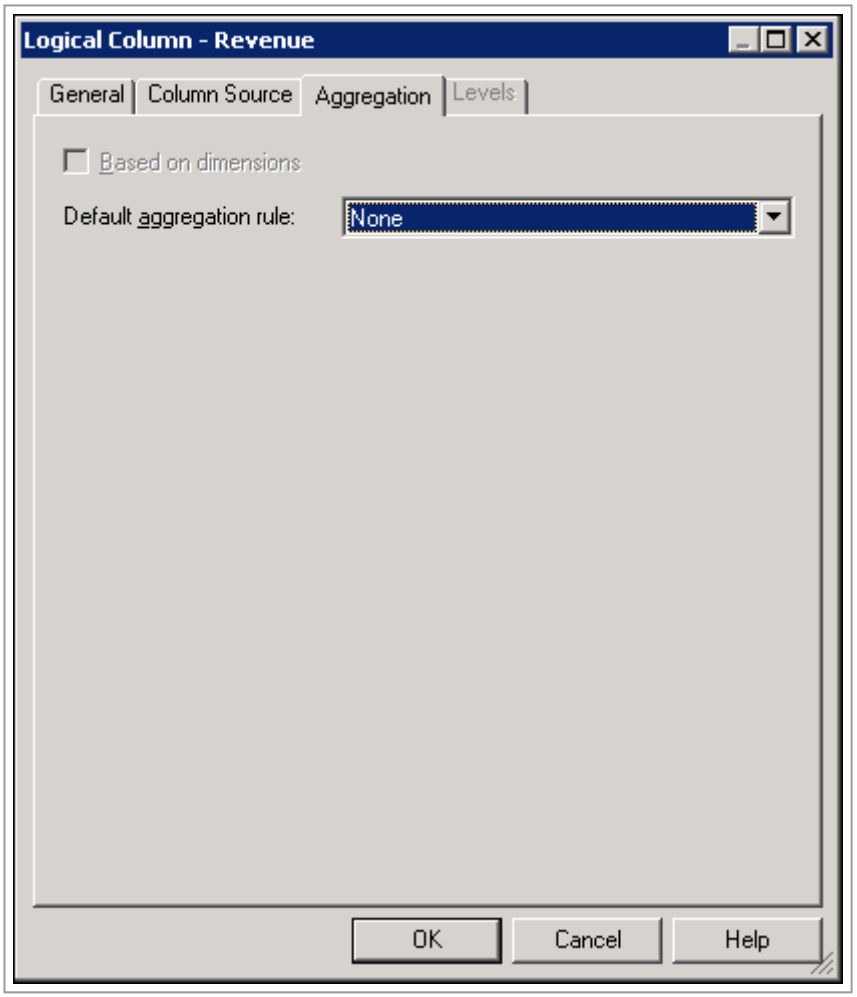


## Creating Simple Measures

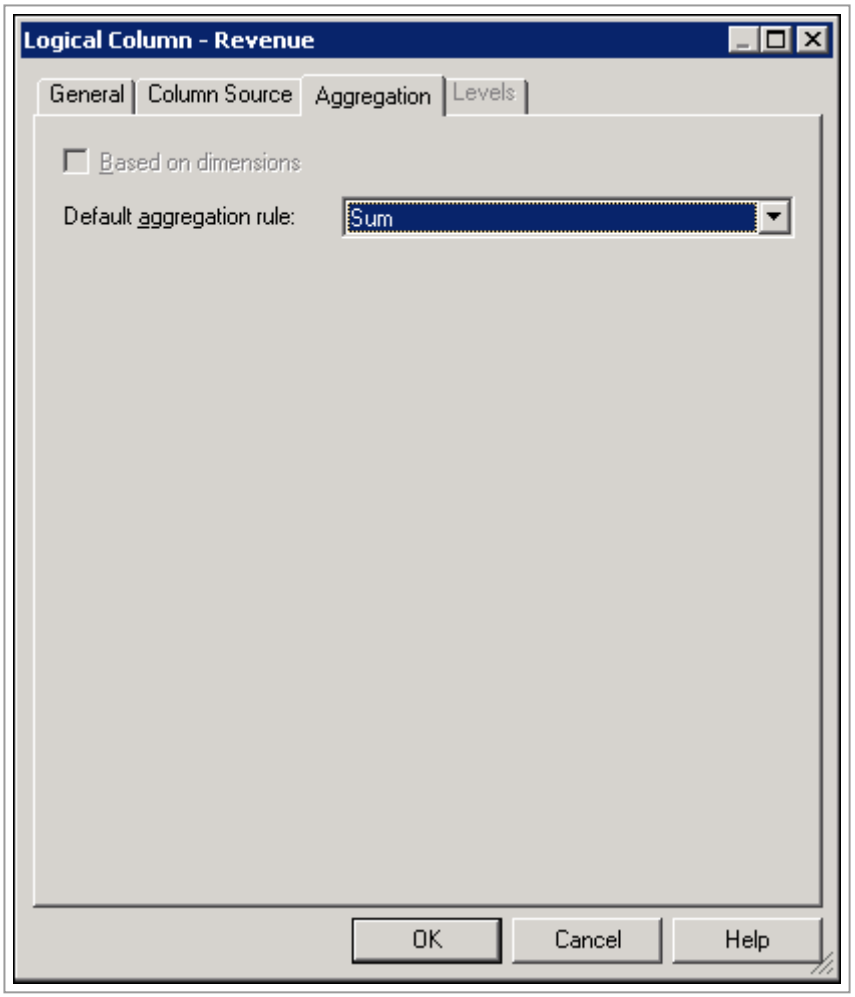
1. Double-click the **Revenue** logical column to open the **Logical Column** dialog box.



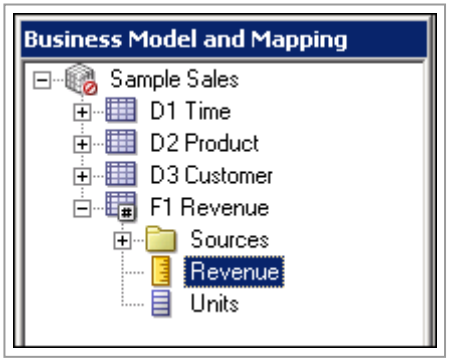

2. Click the **Aggregation** tab.



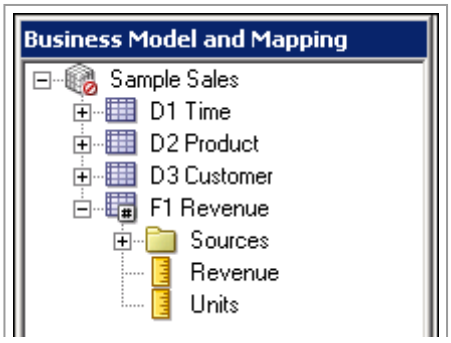

**3.** Change the default aggregation rule to **Sum**.

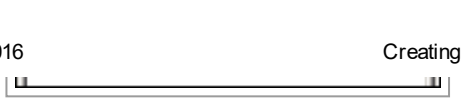



4. Click **OK** to close the Logical Column dialog box. Notice that the icon has changed for the Revenue logical column indicating that the aggregation rule has been applied.




5. Repeat the steps to define the **SUM** aggregation rule for the **Units** logical column.





Measures are typically data that is additive, such as total dollars or total quantities. The F1 Revenue logical fact table contains the measures in your business model. You aggregated two logical columns by summing the column data.

6. Save the repository without checking global consistency.

   Congratulations! You have successfully built a business model in the Business Model and Mapping layer of a repository and created business measures.

## Building the Presentation Layer of a Repository

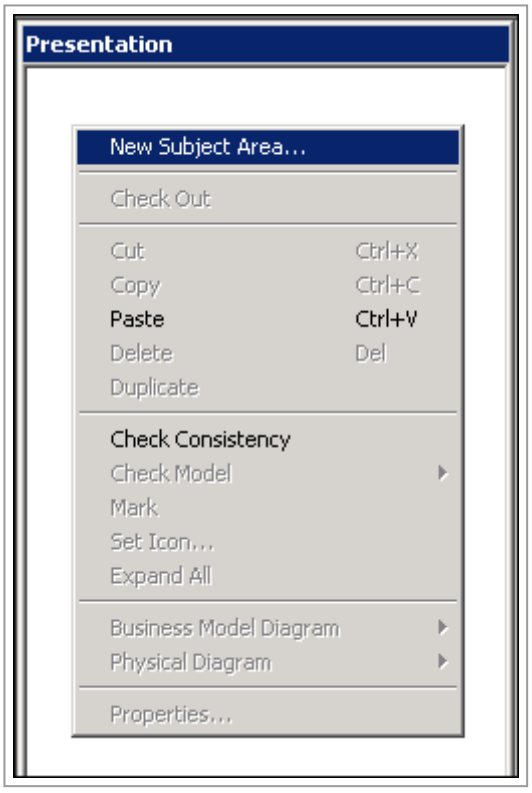
You have created the initial Sample Sales business model in the repository. You now create the Presentation layer of the repository. The Presentation layer exposes the business model objects in Oracle BI user interfaces so that users can build analyses and dashboards to analyze their data.

To build the Presentation layer you perform the following steps:

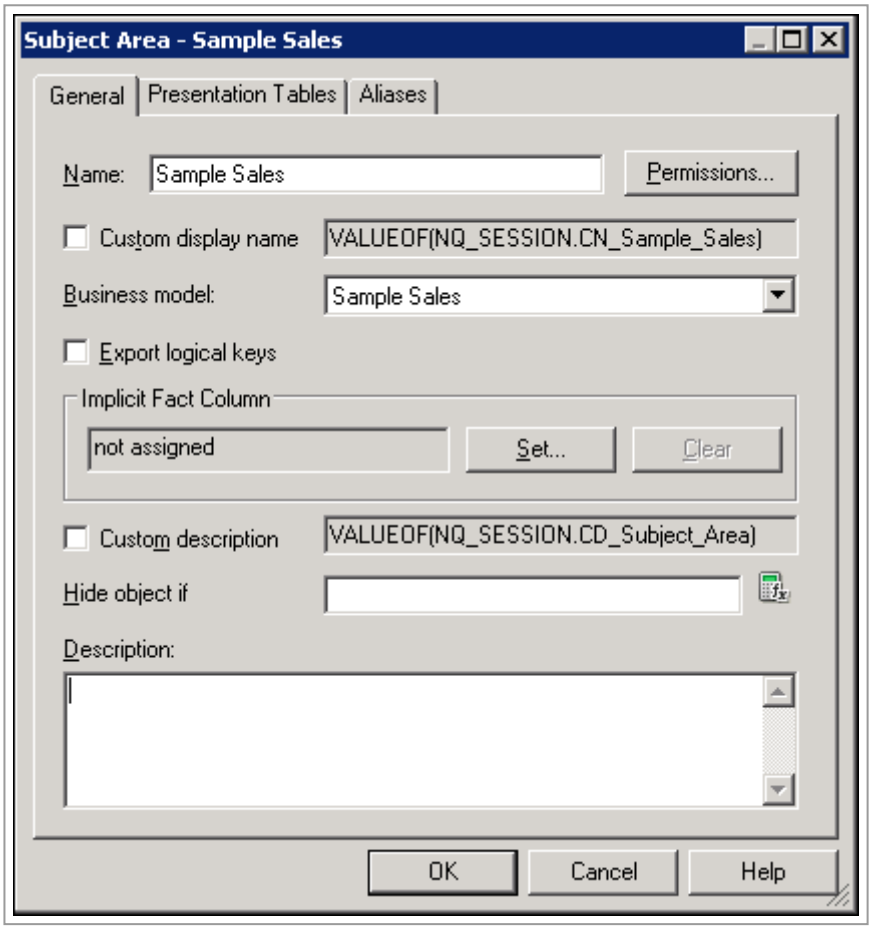
- Creating a Subject Area
- Creating Presentation Tables
- Creating Presentation Columns
- Renaming Presentation Columns
- Reordering Presentation Columns

### Creating a Subject Area

1. Right-click the **white space** in the Presentation layer and select **New Subject Area** to open the Subject Area dialog box.



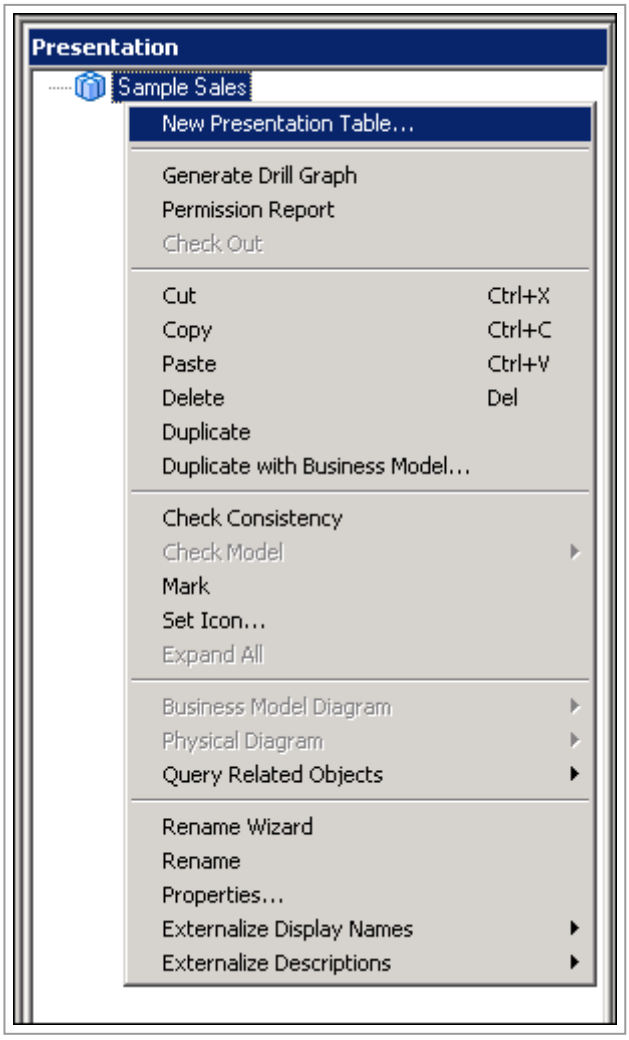

2. On the General tab, enter **Sample Sales** as the name of the subject area.



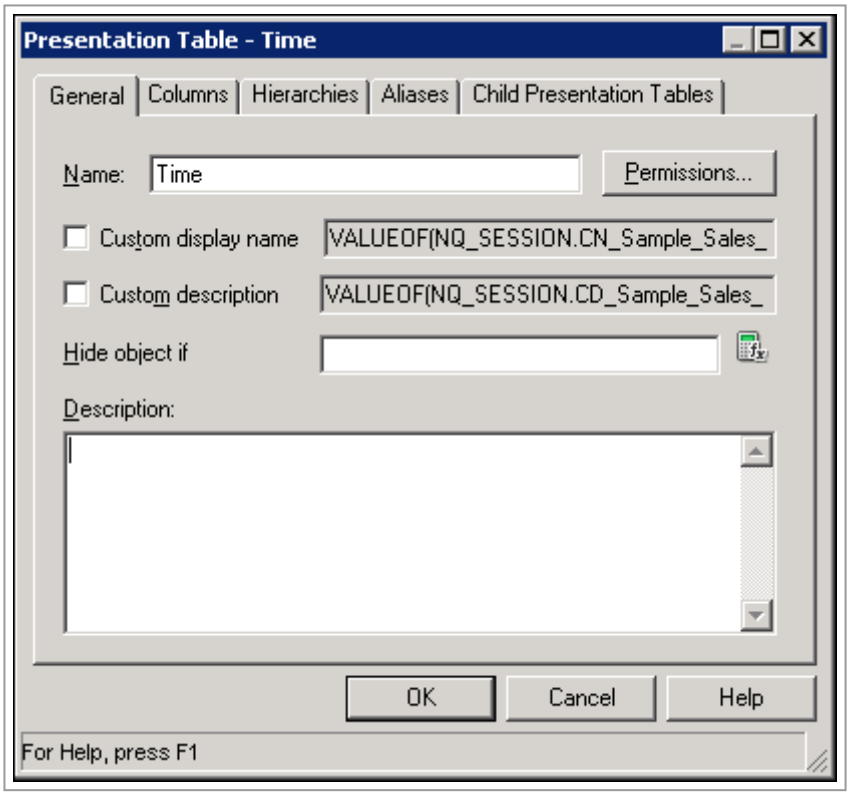

3. Click **OK** to close the Subject Area dialog box. The **Sample Sales** subject area is added to the **Presentation** layer.


## Creating Presentation Tables

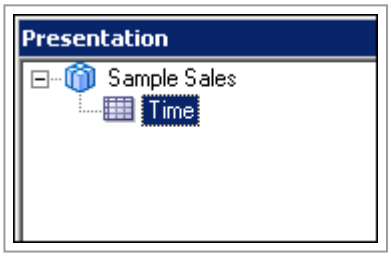
1. Right-click the **Sample Sales** subject area and select **New Presentation Table** to open the Presentation Table dialog box.



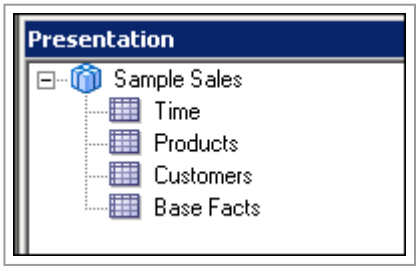

2. On the General tab, enter **Time** as the name of the presentation table.




3. Click **OK** to close the Presentation Table dialog box. The Time presentation table is added to the Sample Sales subject area.



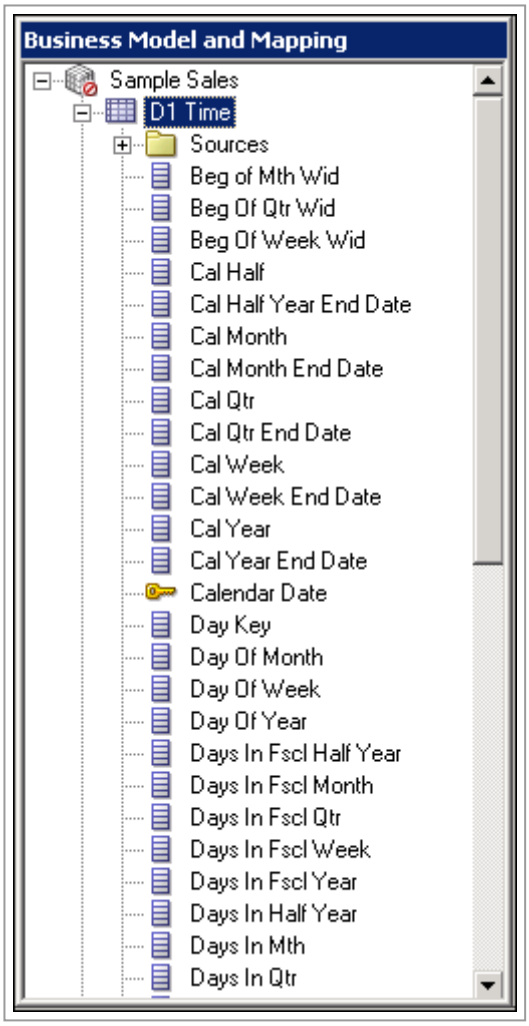

4. Repeat the process and add three more presentation tables: **Products**, **Customers**, and **Base Facts**.




Please note that you are using the manual method for creating Presentation layer objects. For simple models it is also possible to drag objects from the BMM layer to the Presentation layer to create the Presentation layer objects. When you create presentation objects by dragging from the BMM layer, the business model becomes a subject area, the logical tables become presentation tables, and the logical columns become presentation columns. Note that all objects within a subject area must derive from a single business model.

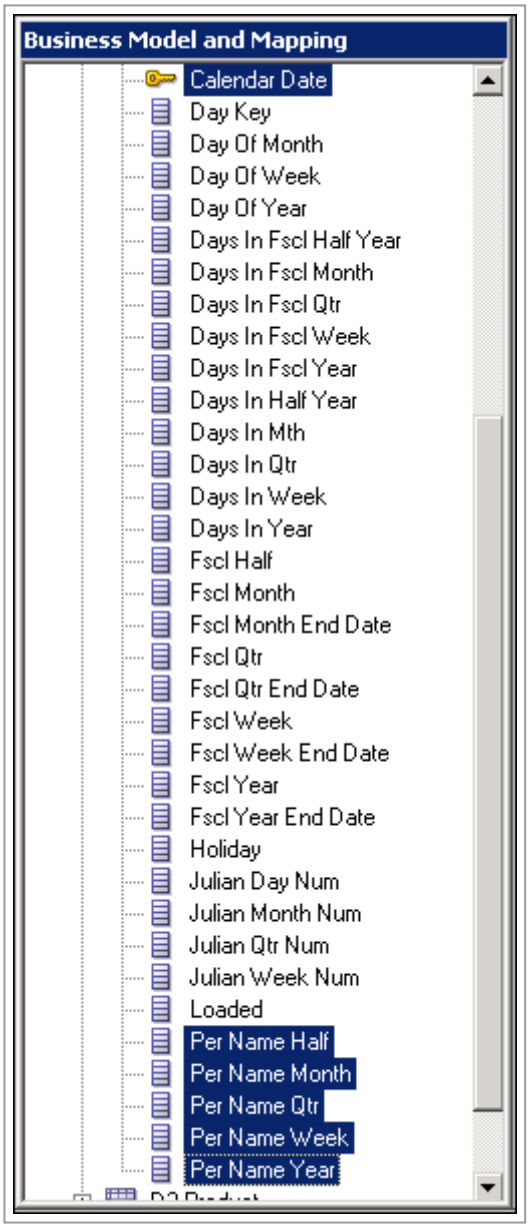
## Creating Presentation Columns

1. In the **BMM** layer, expand the **D1 Time** logical table.

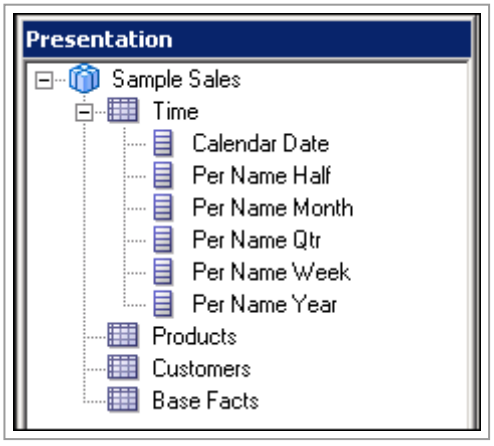



2. Select the following logical columns:

**Calendar Date**
**Per Name Half**
**Per Name Month**
**Per Name Qtr**
**Per Name Week**
**Per Name Year**.



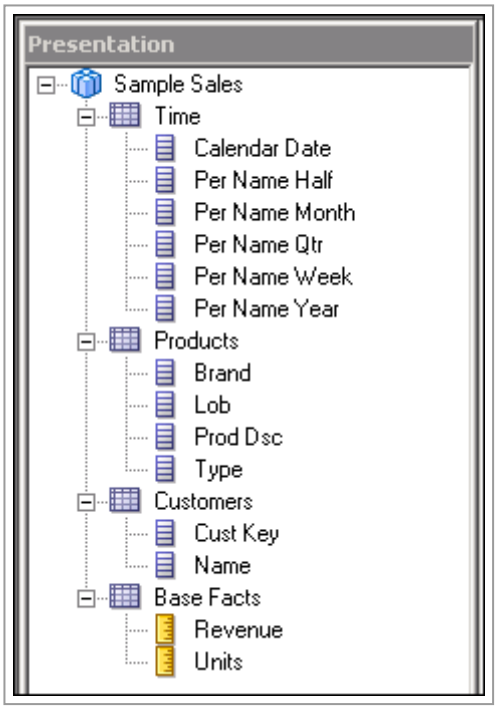

3. Drag the selected logical columns to the **Time** presentation table in the **Presentation** layer.




4. Repeat the process and add the following logical columns to the remaining presentation tables:

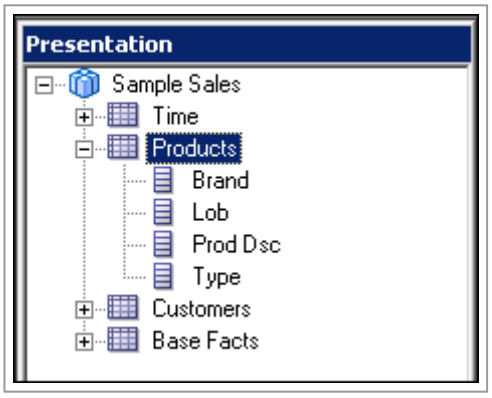
   **Products**: Drag **Brand**, **Lob**, **Prod Dsc**, **Type** from **D2 Product**.
   **Customers**: Drag **Cust Key**, **Name** from **D3 Customer**.

**Base Facts**: Drag **Revenue**, **Units** from **F1 Revenue**.




## Renaming Presentation Columns

1. In the **Presentation** layer, expand the **Products** presentation table.

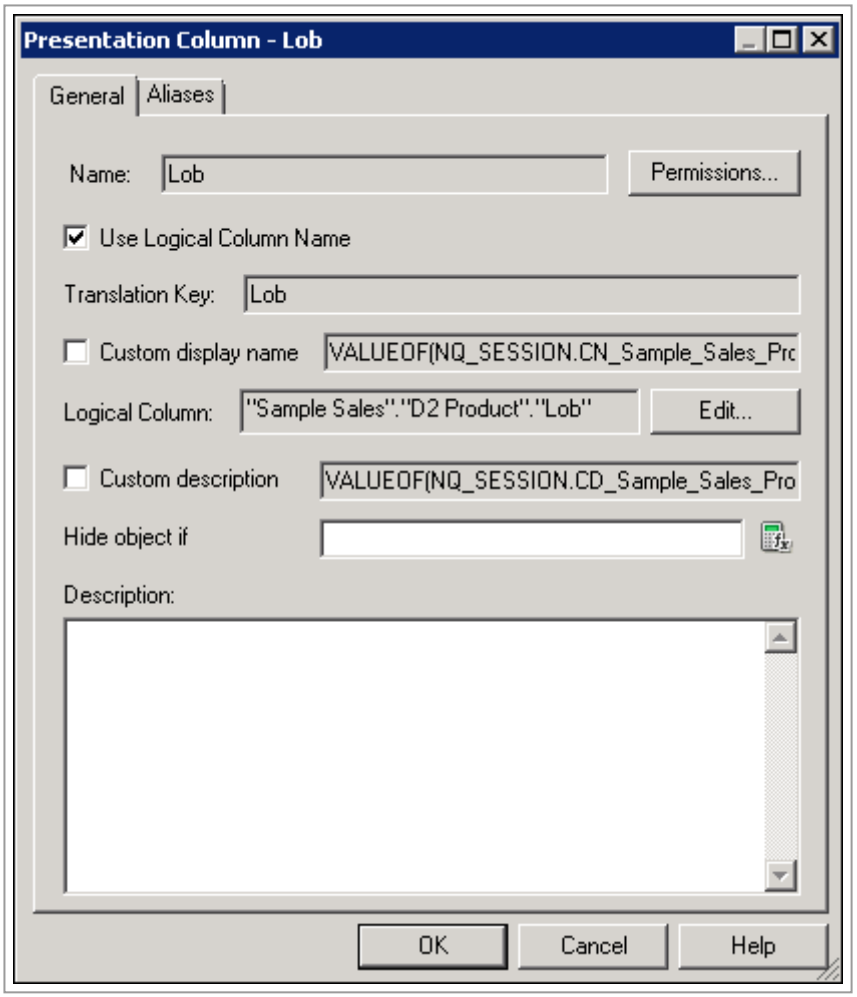



**Notes**: By default, presentation object names are read-only. Select “Edit presentation names” option to allow the names of presentation objects to be modified.

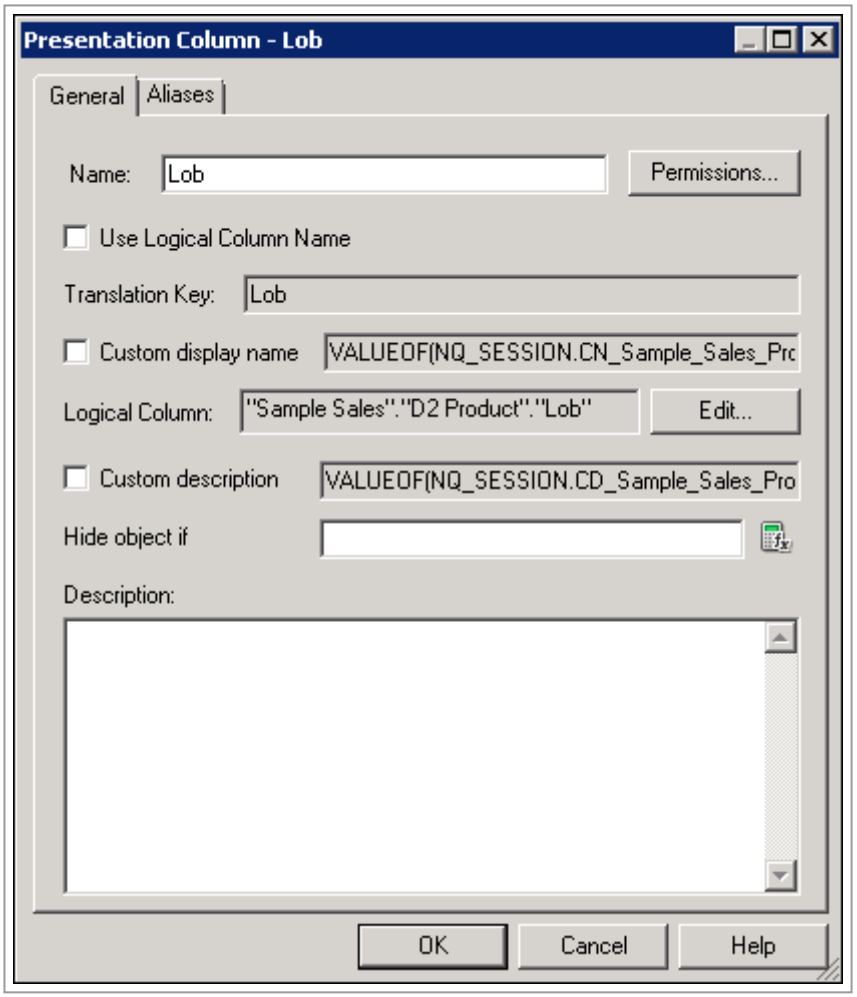
**To set Edit presentation names options:**

- In the Administration Tool, select **Tools**, then select **Options** to display the Options dialog.
- On the General tab, select the **Edit presentation names** options you want to choose.

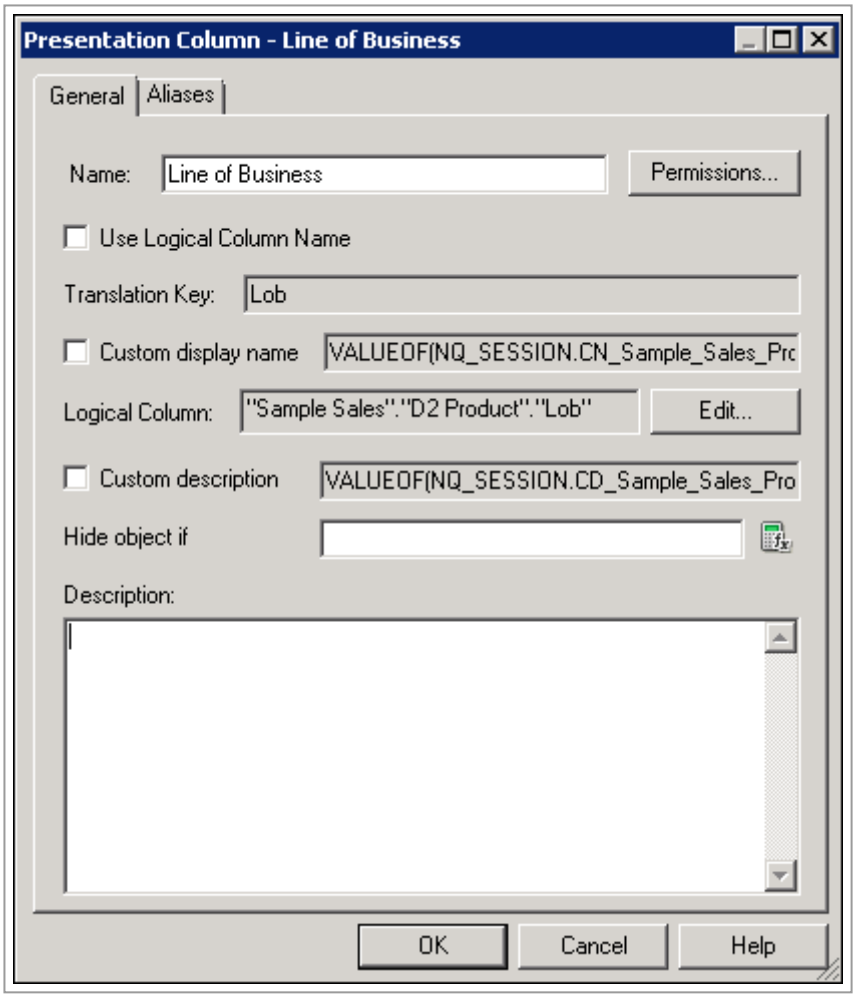
2. Double-click the **Lob** presentation column to open the Presentation Column dialog box. On the General tab notice that **"Use Logical Column Name"** is selected. When you drag a logical column to a presentation table, the resulting presentation column inherits the logical column name by default. In this example the Lob presentation column inherits the name of the logical column **"Sample Sales"."D2 Product"."Lob"**.



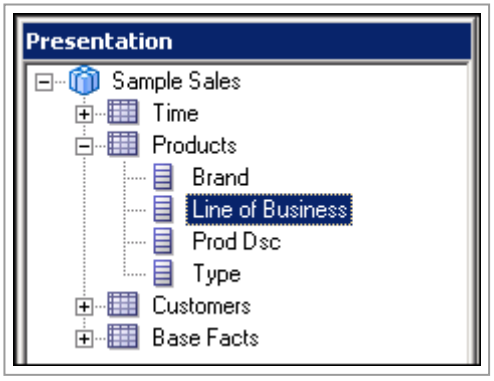

**3.** Deselect **Use Logical Column Name**. The Name field is now editable.



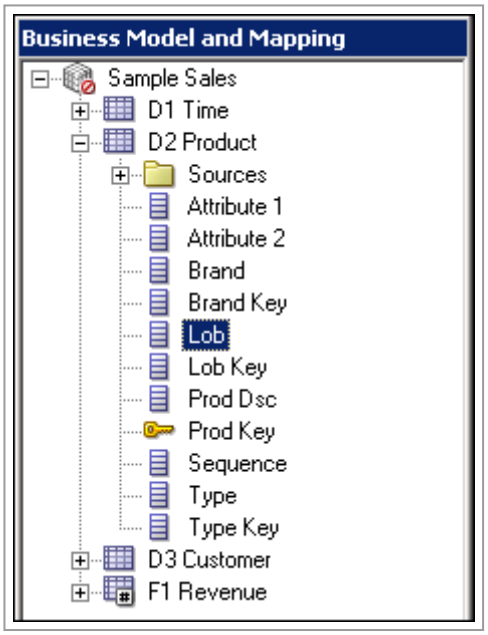

**4.** Enter **Line of Business** in the Name field.



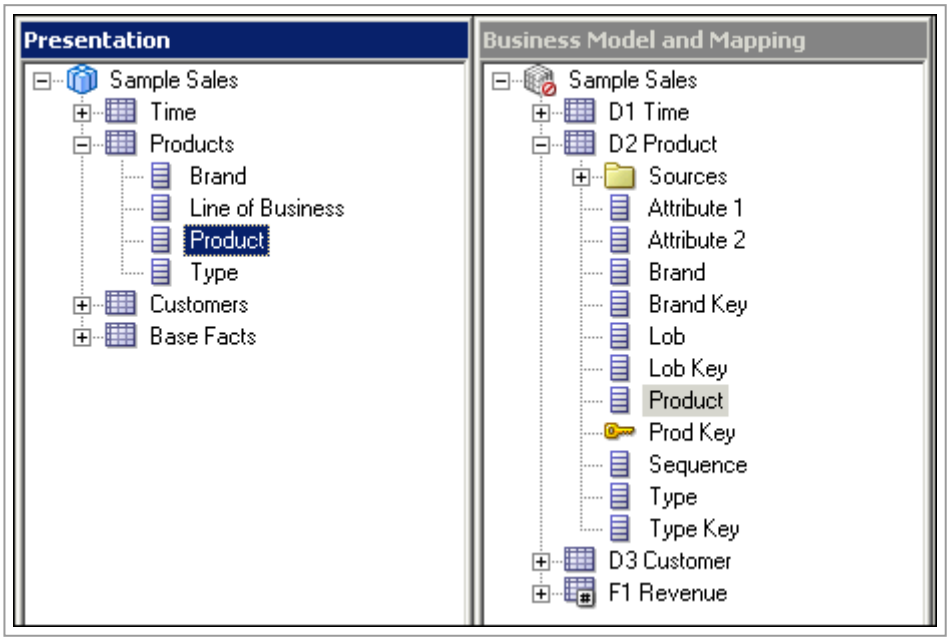

5. Click **OK** to close the Presentation Column dialog box. Notice that the presentation column name is now changed to **Line of Business** in the Presentation layer.




6. In the **BMM** layer, expand **D2 Product**. Notice that the **Lob** logical column name is not changed. The point here is you can change object names in the **Presentation** layer without impacting object names in the **BMM** or **Physical** layers.



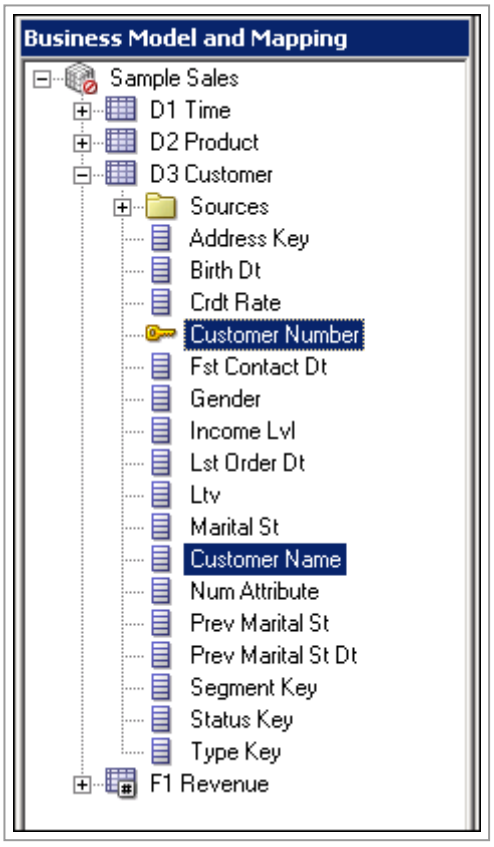

7. In the **BMM** layer, rename the **Prod Dsc** logical column to **Product**. Notice that the name change is inherited by the corresponding presentation column.



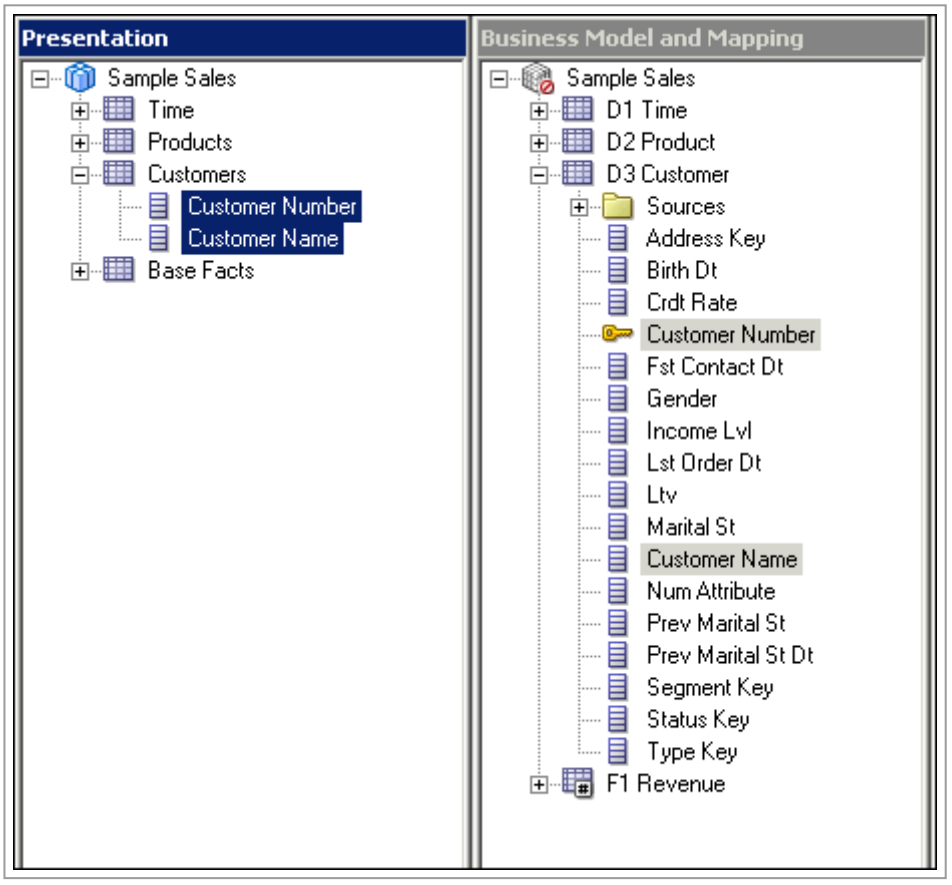

8. Make the following name changes to logical objects in the **BMM** layer so that the names of the corresponding presentation columns are also changed:

   For the **D3 Customer** logical table:

   Change **Cust Key** to **Customer Number**.
   Change **Name** to **Customer Name**.

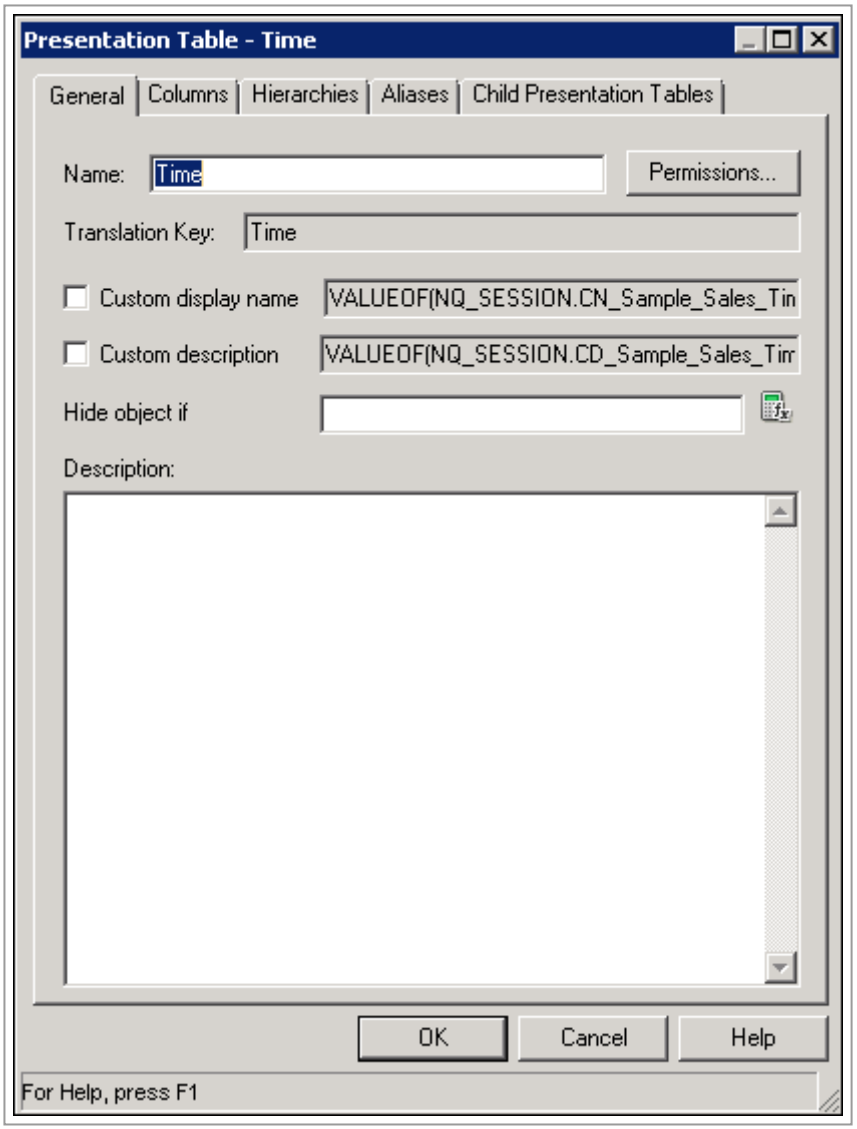



**9.** Confirm that the corresponding presentation column names are changed.

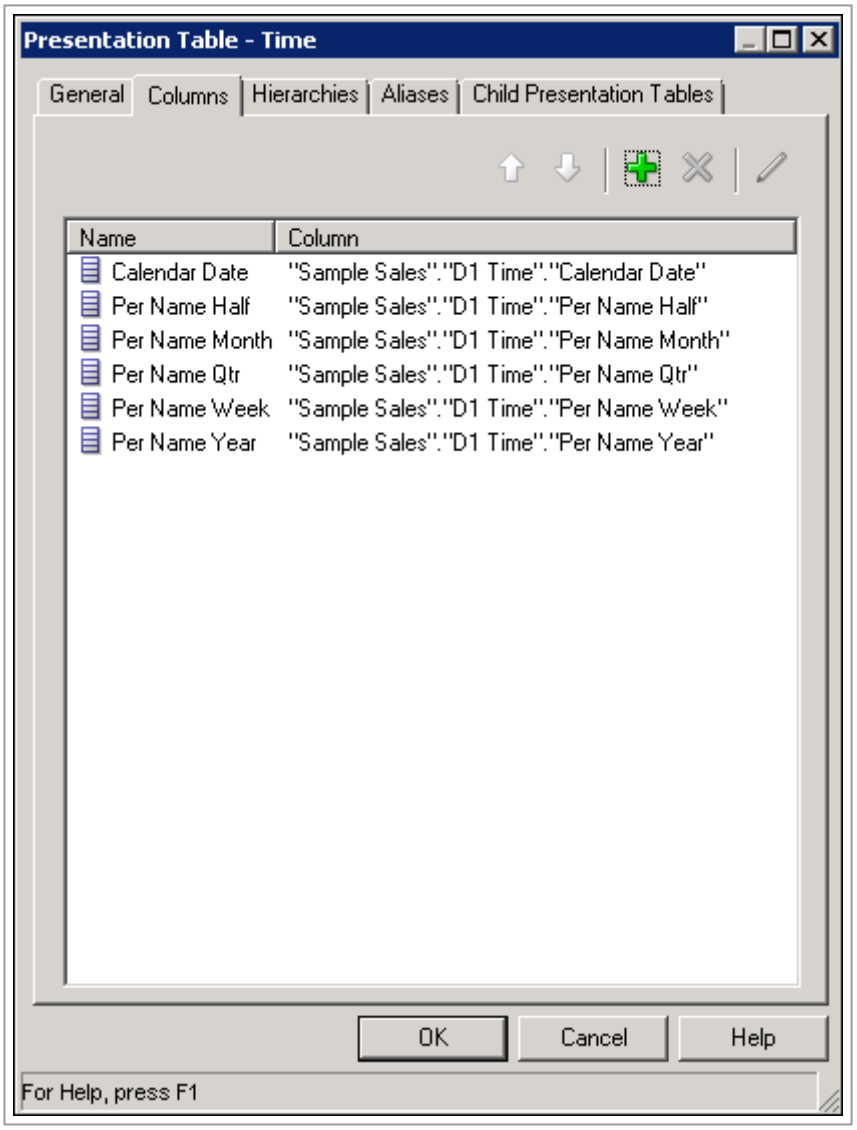



## Reordering Presentation Columns

1. In the **Presentation** layer, double-click the **Time** presentation table to open the **Presentation Table** dialog box.

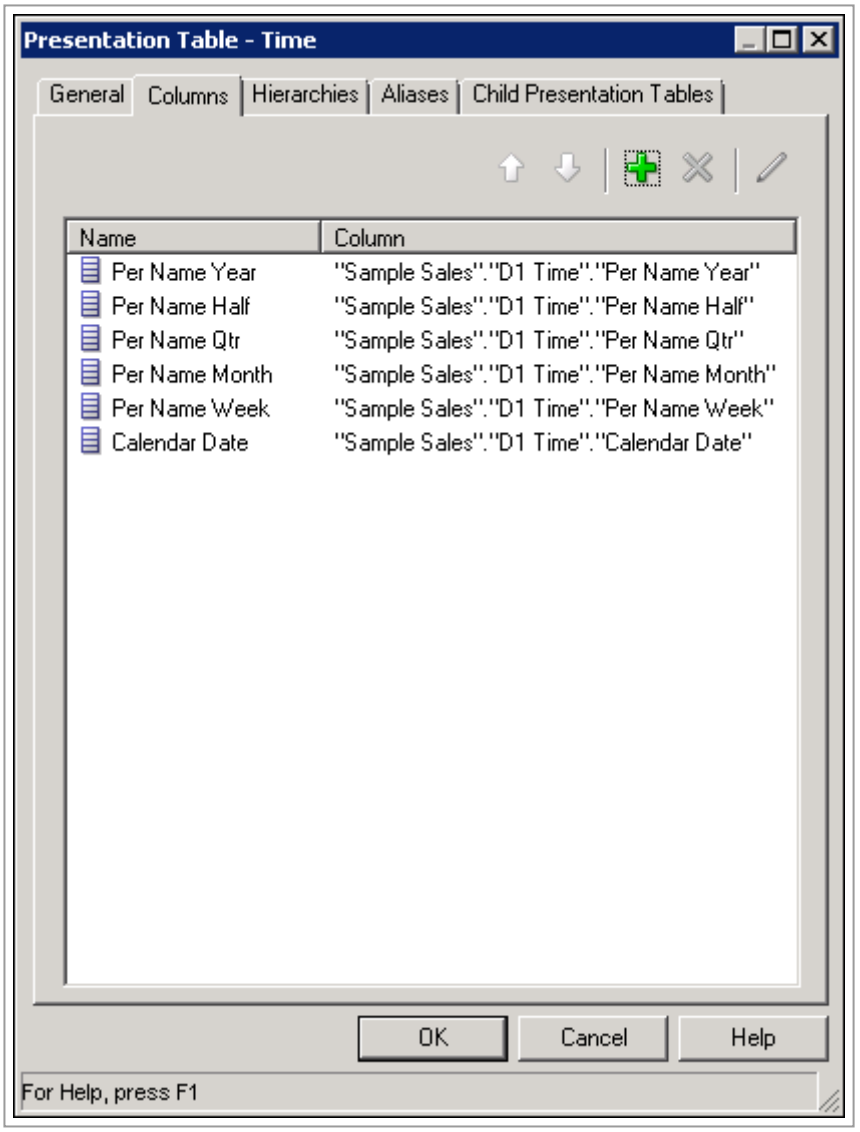



2. Click the **Columns** tab.

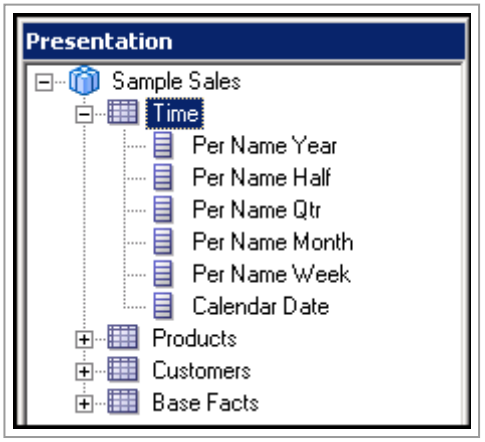



3. Select columns and use the up and down arrows, or drag the columns. to rearrange the presentation columns into the following order from top to bottom:

   **Per Name Year**
   **Per Name Half**
   **Per Name Qtr**
   **Per Name Month**
   **Per Name Week**
   **Calendar Date**



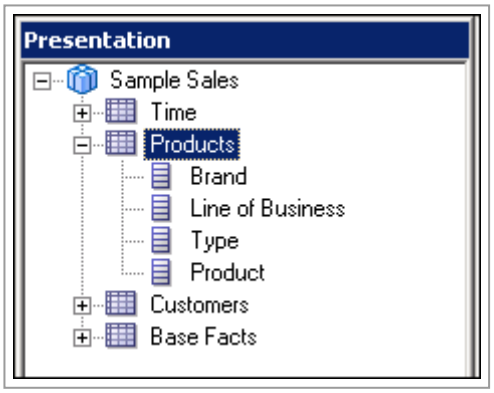

4. Click **OK** to close the **Presentation Table** dialog box and confirm that the presentation column order is changed in the Presentation layer.




5. Repeat the steps to reorder the columns in the **Products** presentation table:

**Brand**
**Line of Business**
**Type**
**Product**

Product




6. **Save** the repository without checking global consistency.

   Congratulations! You have successfully built the Presentation layer of a repository.

## Testing and Validating a Repository

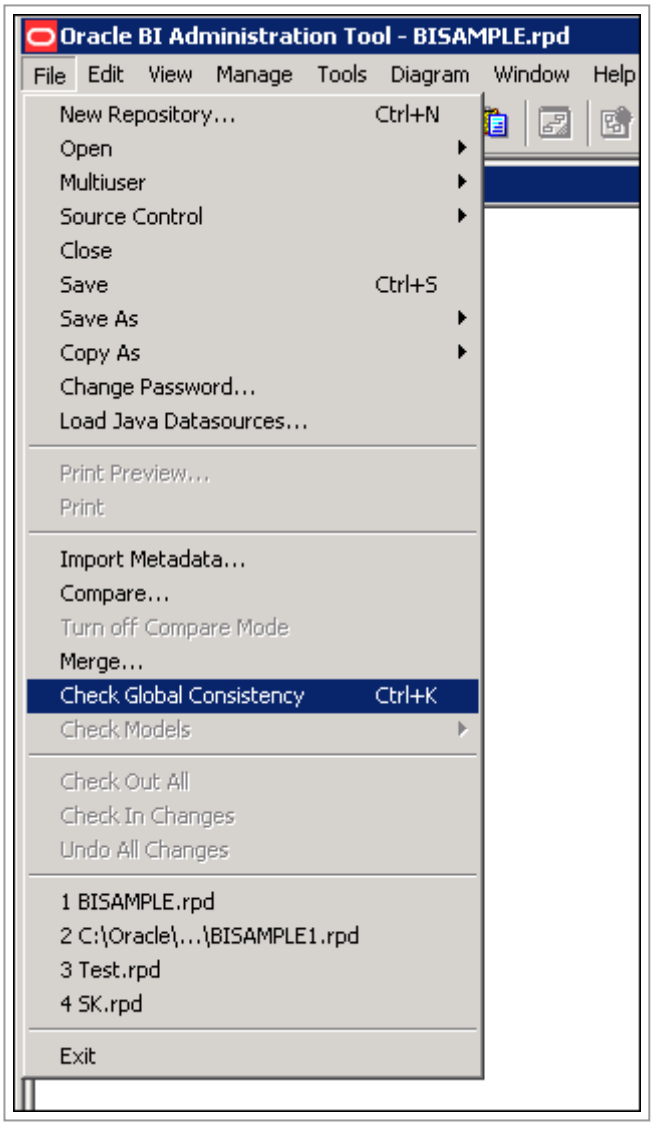
You have finished building an initial business model and now need to test and validate the repository before continuing. You begin by checking the repository for errors using the consistency checking option. Next you load the repository into Oracle BI Server memory. You then test the repository by running an Oracle BI analysis and verifying the results.

To test and validate a repository you perform the following steps:

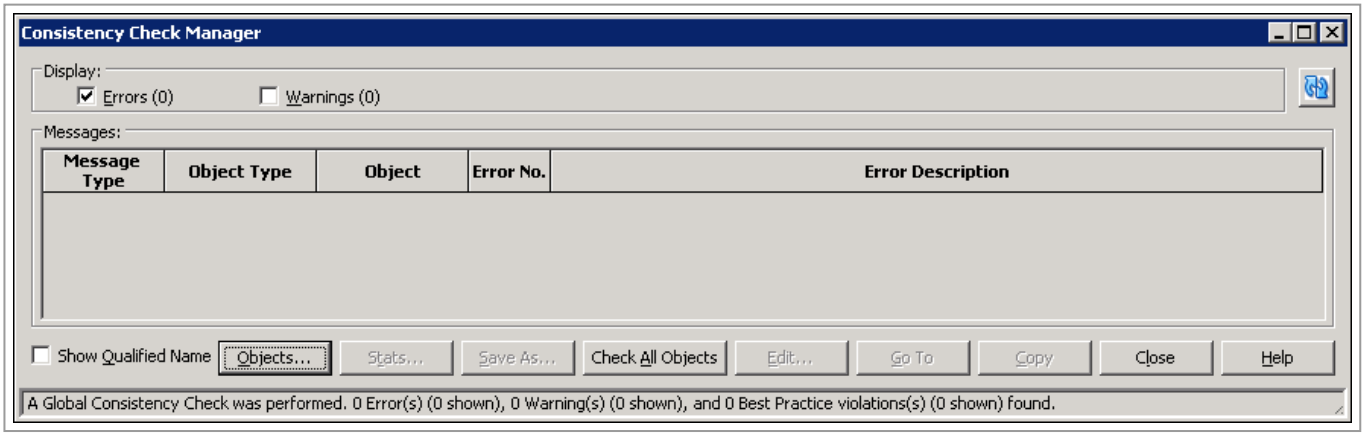
- Checking Consistency
- Disabling Cache
- Loading the Repository
- Creating and Running Analysis

### Checking Consistency

1. Select **File > Check Global Consistency**.

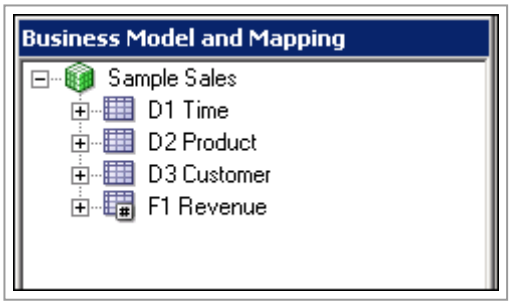



2. You should receive the message: **A Global Consistency Check was performed; 0 Error(s) (0 shown), 0 Warning(s) (0 shown), and 0 Best Practice violation(s) (0 shown) found.**




If you do not receive this message, you must fix any consistency check errors or warnings before proceeding.

3. Click **Close**. Notice that the Sample Sales business model icon in the BMM layer is now green, indicating it is available for queries.




4. Save **File > Save** to save the repository.

5. Select **File > Close** to close the repository. Leave the Administration Tool open.

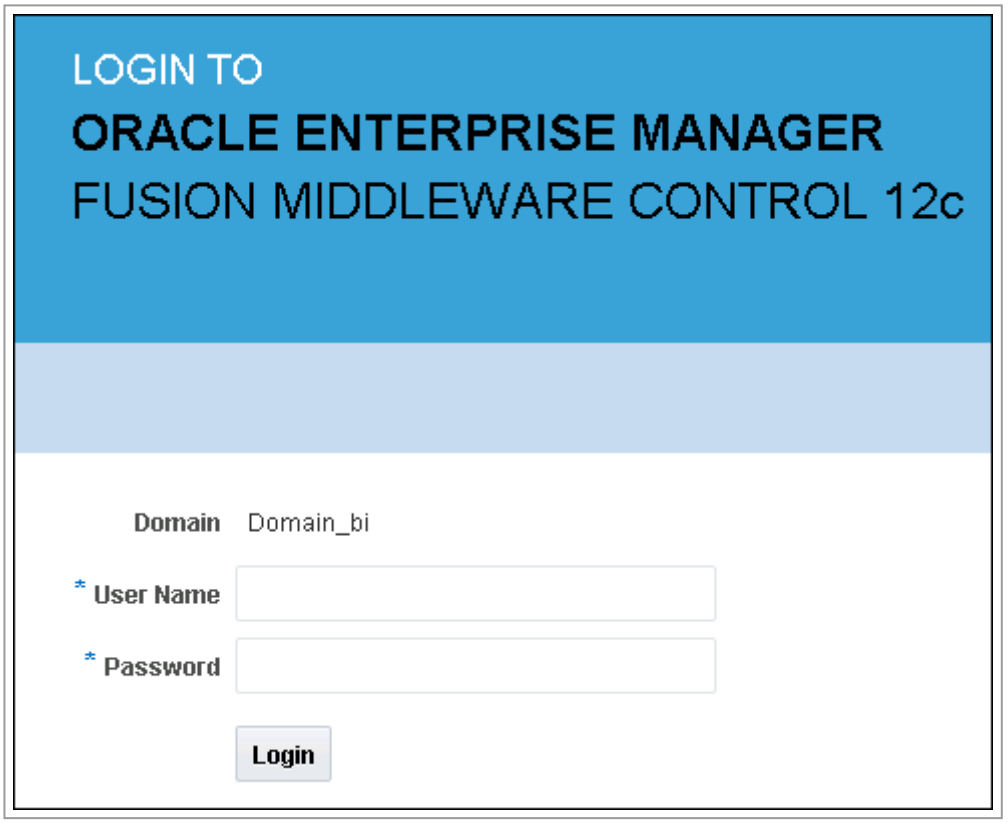
## Disabling Cache

1. Open a browser and enter the following URL to navigate to Oracle Enterprise Manager:

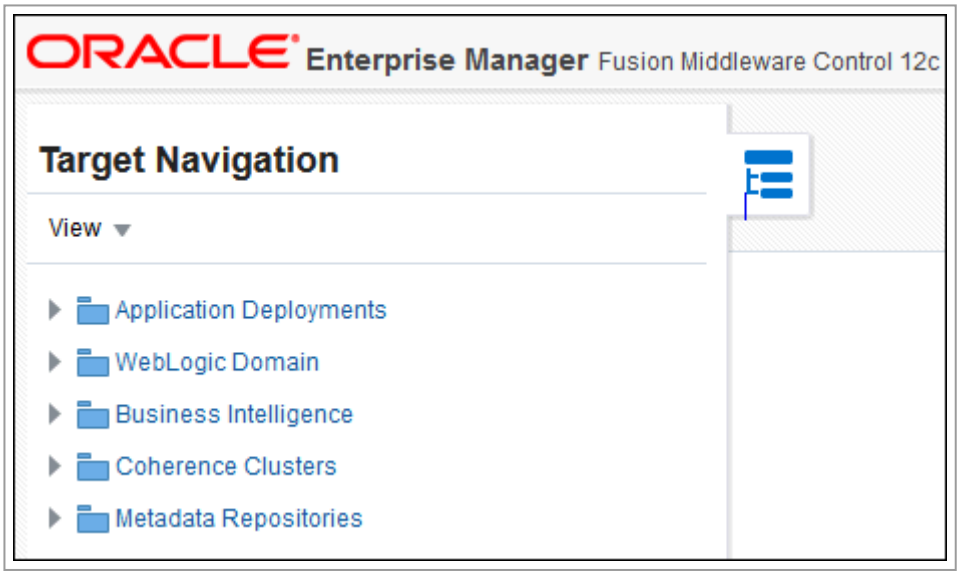
   **http://<machine name>:9500/em**

   In this tutorial the URL is http://localhost:9500/em

2. Log in as an administrative user. Typically you use the administrative user name and password provided during the Oracle BI installation. In this example the user name is **weblogic**.

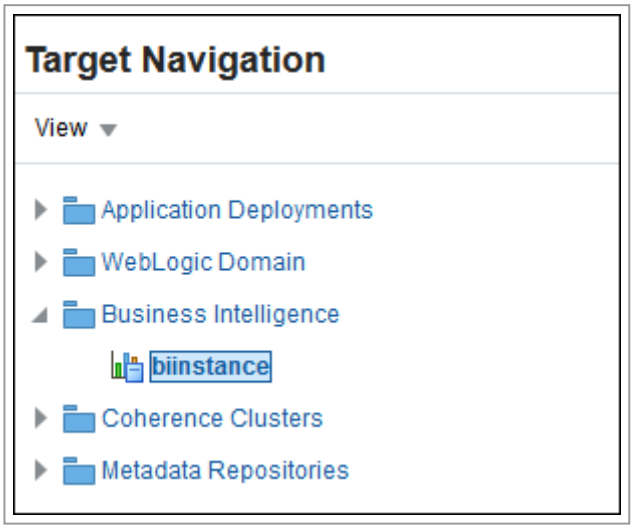



3. On the top left, click **Target Navigation**.

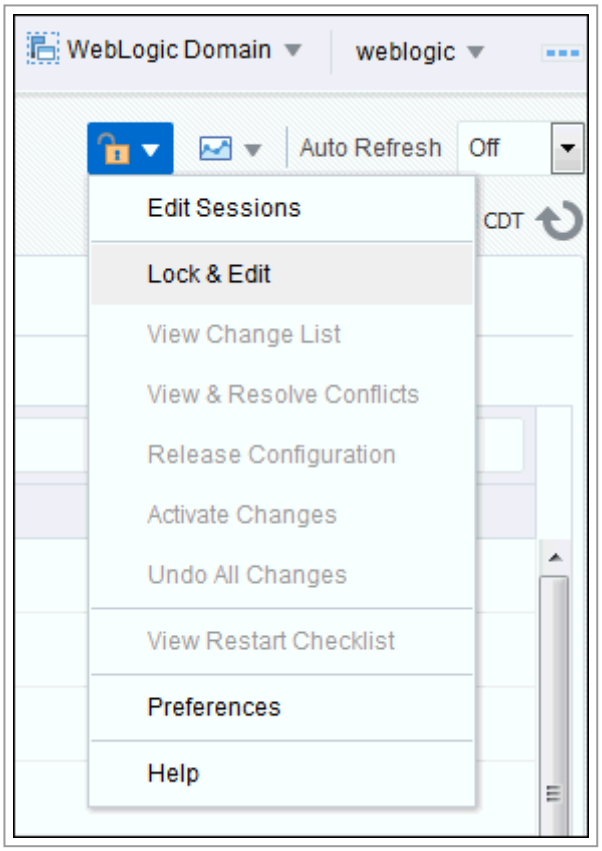



4. Expand **Business Intelligence** and click **biinstance**.

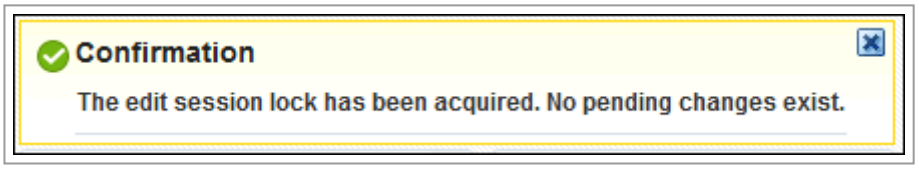
   Click outside of the navigation tree to see the tabs on the page.



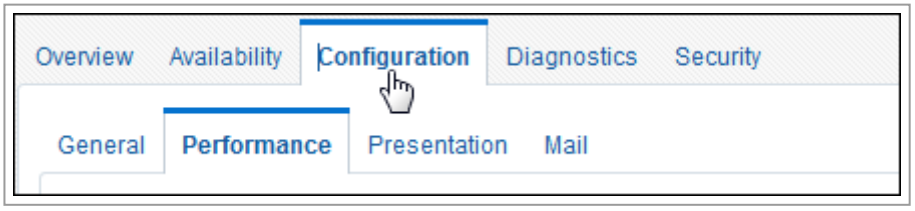

5. On the top right, click **Lock and Edit**.



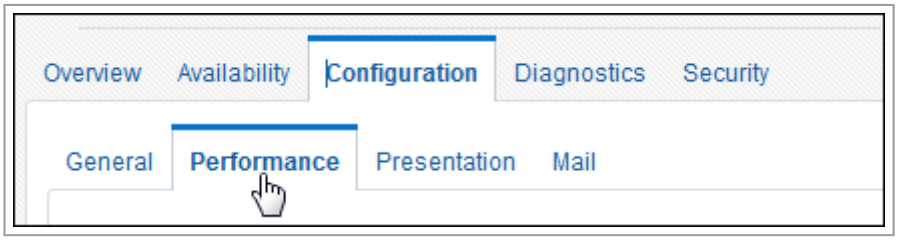

6. You receive the confirmation message "The edit session lock has been acquired. No pending changes exist."




7. Click the **Configuration** tab.

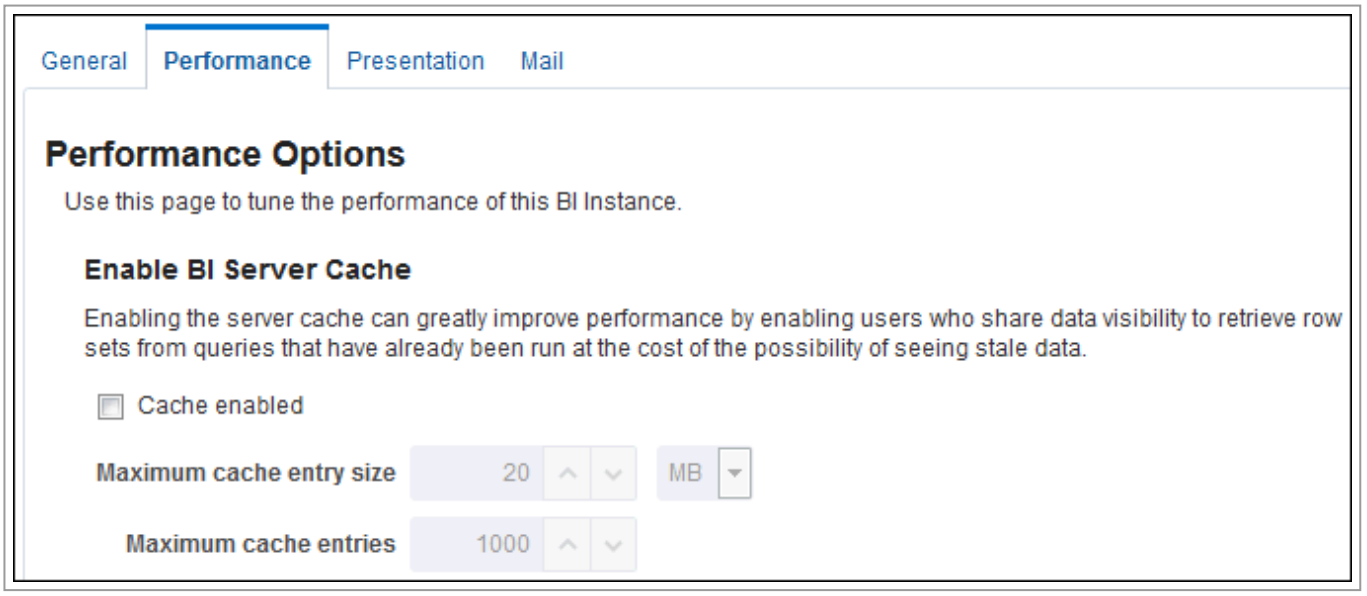



8. Click the **Performance** sub tab.

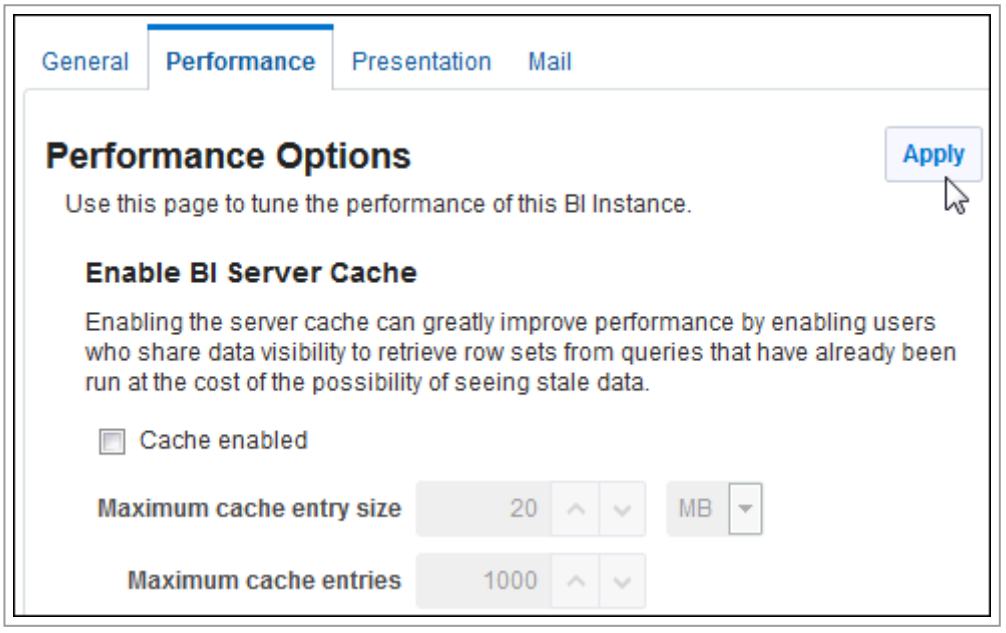



9. Locate the **Enable BI Server Cache** section. Cache is enabled by default.

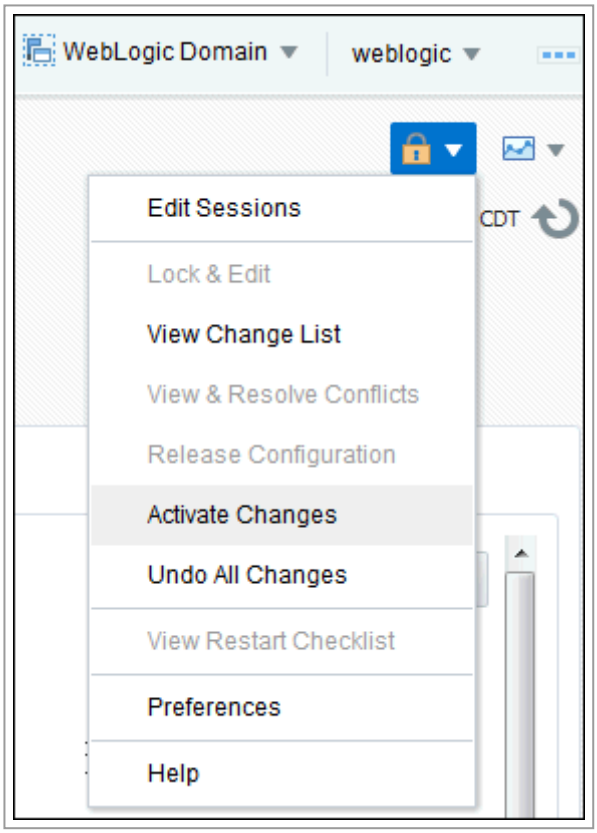
10. Deselect **Cache enabled**. Caching is typically not used during development. Disabling cache improves query performance.



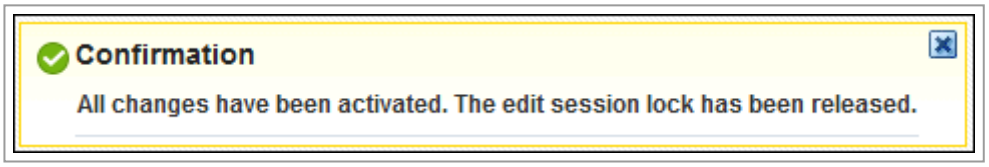

11. Click **Apply**.




12. Click **Activate Changes**.



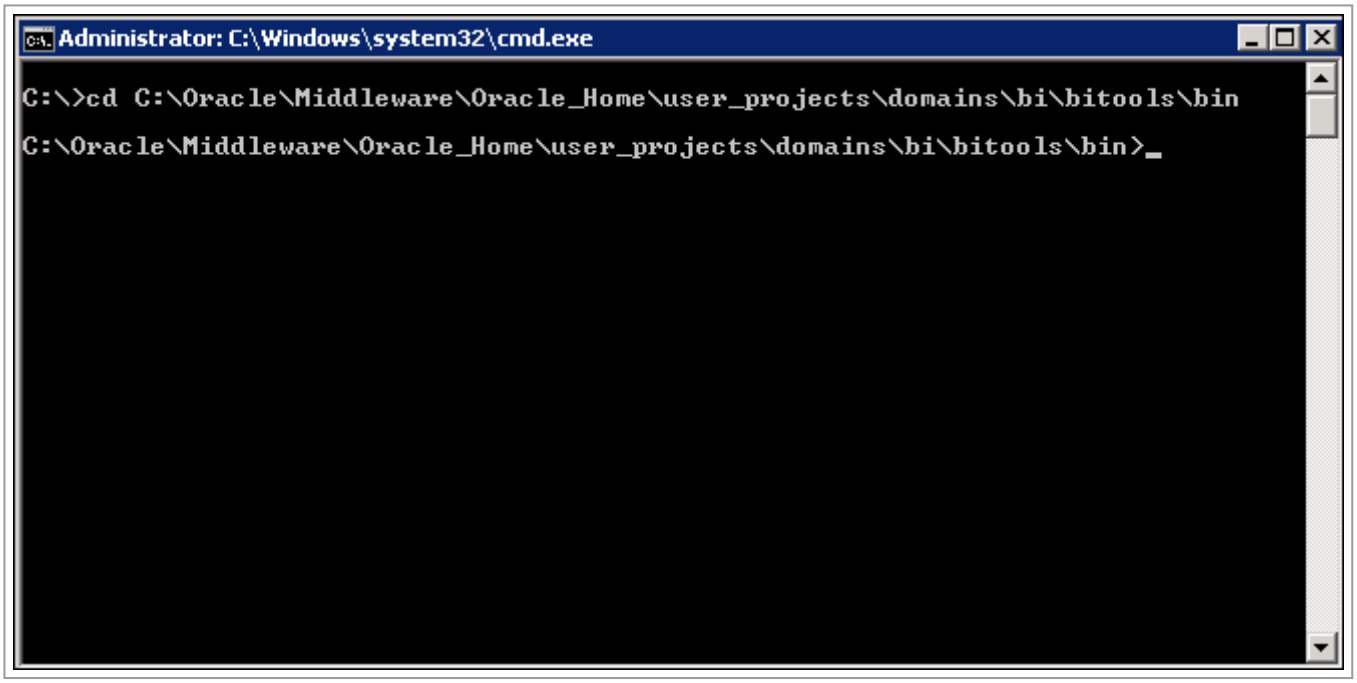

13. You receive the confirmation message " All changes have been activated. The edit session lock has been released."




## Loading the Repository

1. Open the command prompt and type "`cd \`" to change the directory and press the **Enter** key.

2. Type "`cd C:\Oracle\Middleware\Oracle_Home\user_projects\domains\bi\bitools\bin`" and press the **Enter** key.

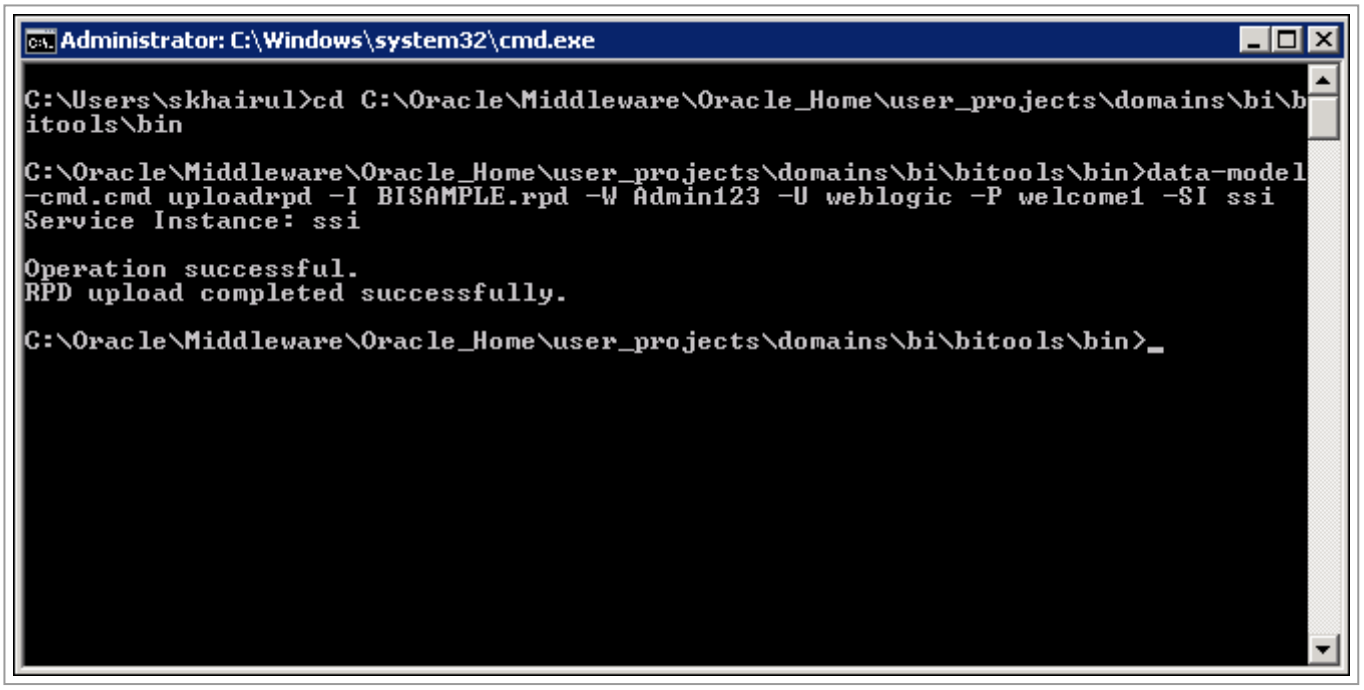



3. Run the **data-model-cmd.cmd** utility with the uploadrpd parameters:

```
uploadrpd -I BISAMPLE.rpd -W Admin123 -U weblogic -P welcome1 -SI ssi
```

If the operation completes successfully, you will see the following message:

"Operation Successful. RPD upload completed successfully. "




**Notes:**

`I` specifies name of the repository.
`W` specifies the repository's password.
`U` specifies a valid user's name to be used for Oracle BI EE authentication.
`P` specifies the password corresponding to the user's name that you specified for U.

`SI` specifies the name of the service instance.

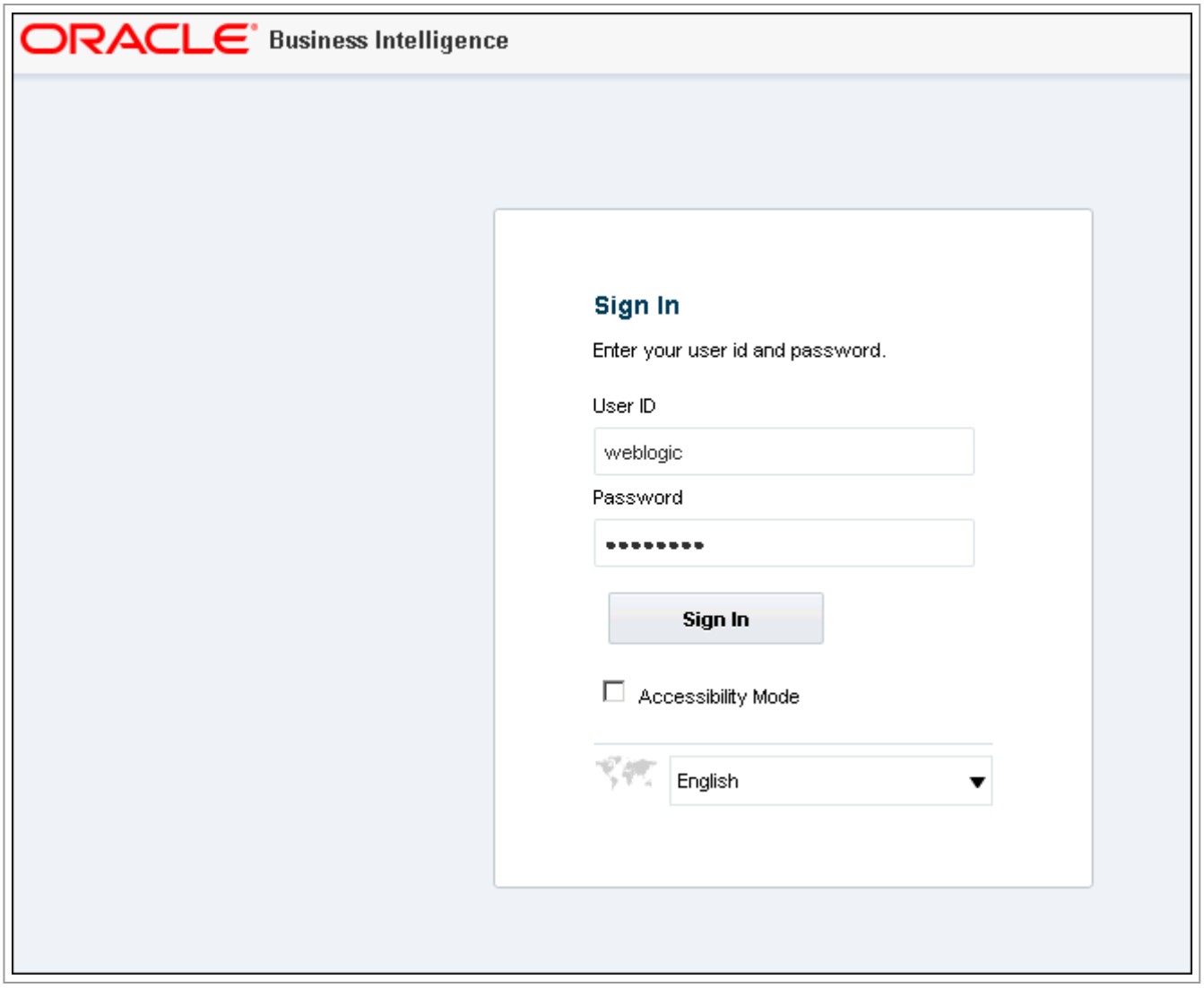
## Creating and Running Analysis

1. Open a browser or a new browser tab and enter the following URL to navigate to Oracle Business Intelligence:

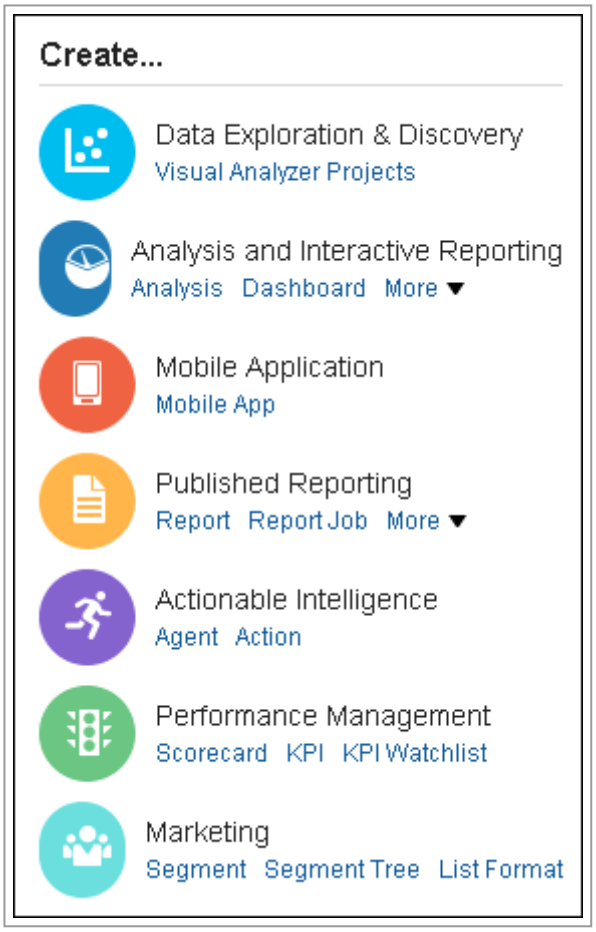
   **http://<machine name>:9502/analytics**

   In this tutorial the URL is http://localhost:9502/analytics

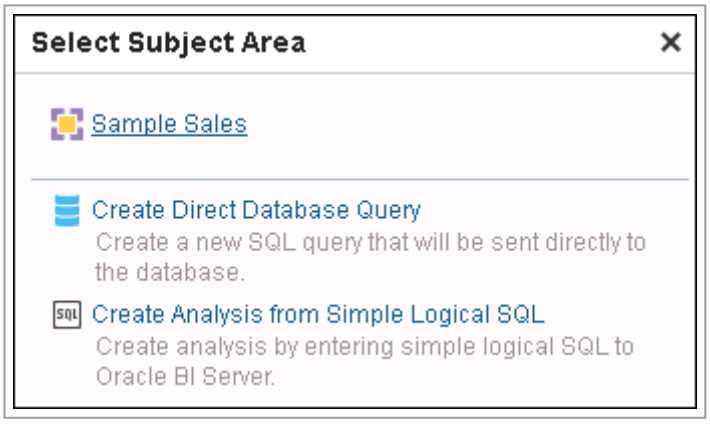
2. Sign in as an administrative user. Typically you use the administrative user name and password provided during the Oracle BI installation. In this example the user name is **weblogic**. If you need help identifying a user name and password, contact your company's Oracle BI Administrator.



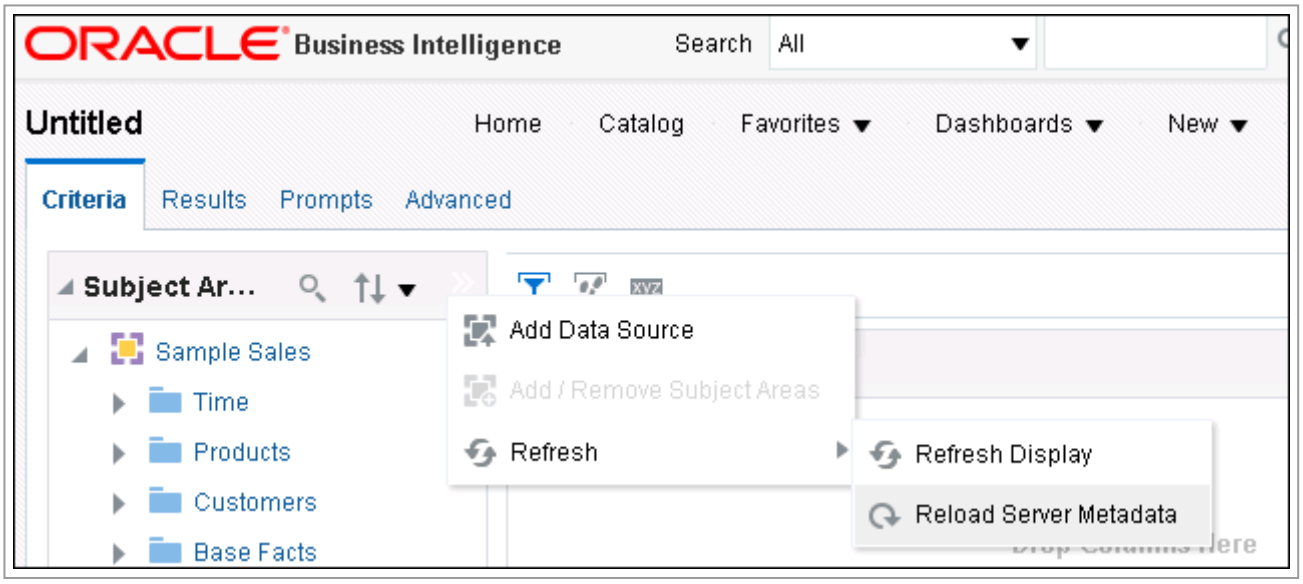

3. In the left navigation pane, under **Create... >Analysis and Interactive Reporting**, select **Analysis**.



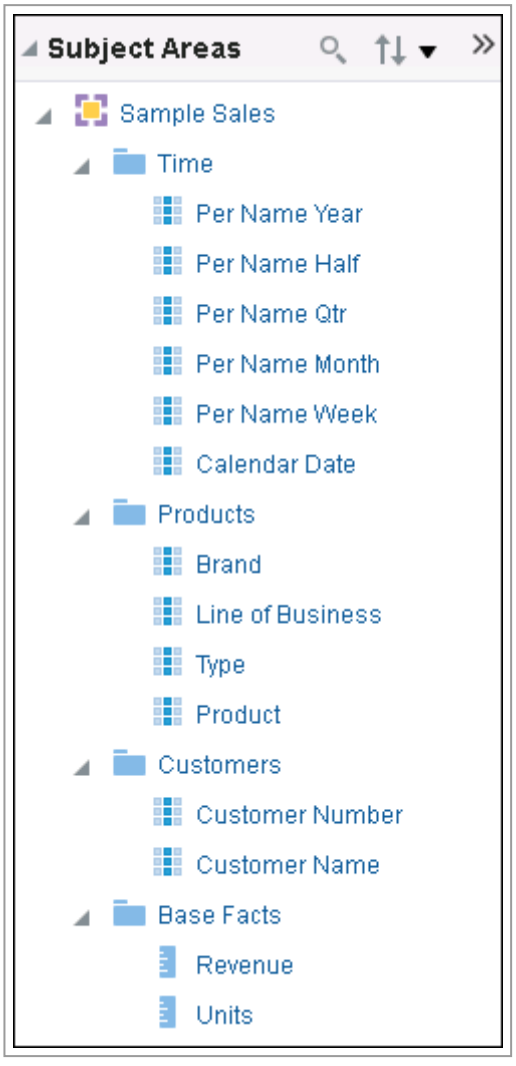

**4.** Select the **Sample Sales** subject area.




**5.** Select **"Reload Server Metadata"**.



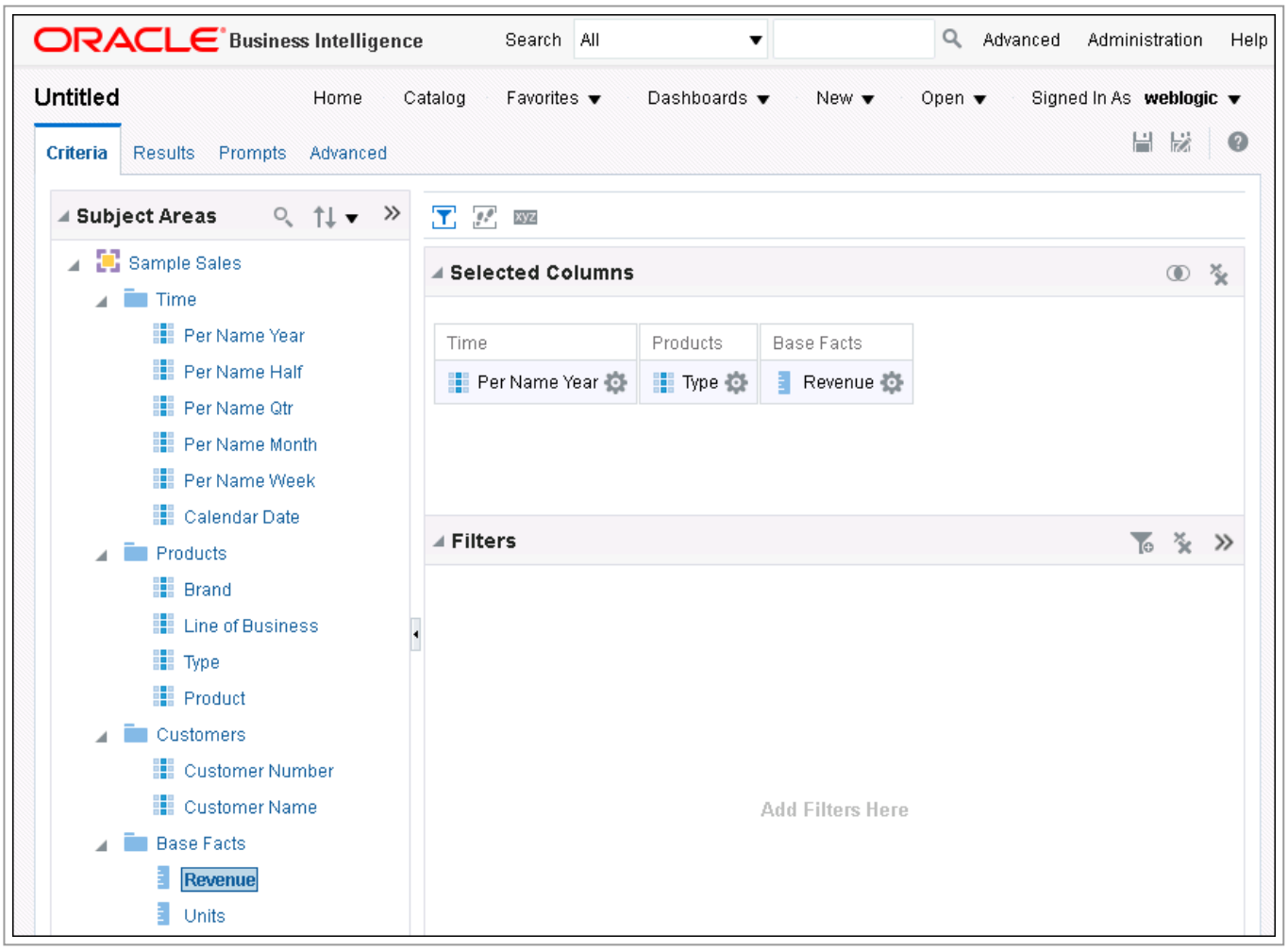

6. In the left navigation pane, expand the folders in the Sample Sales subject area and confirm that the user interface matches the **Presentation** layer of the repository.



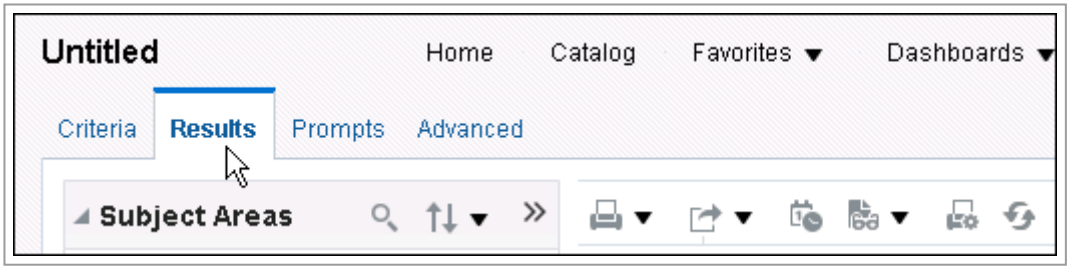

7. Double-click the following column names in the Subject Areas pane to add them to the analysis:

**Time.Per Name Year**
**Products.Type**

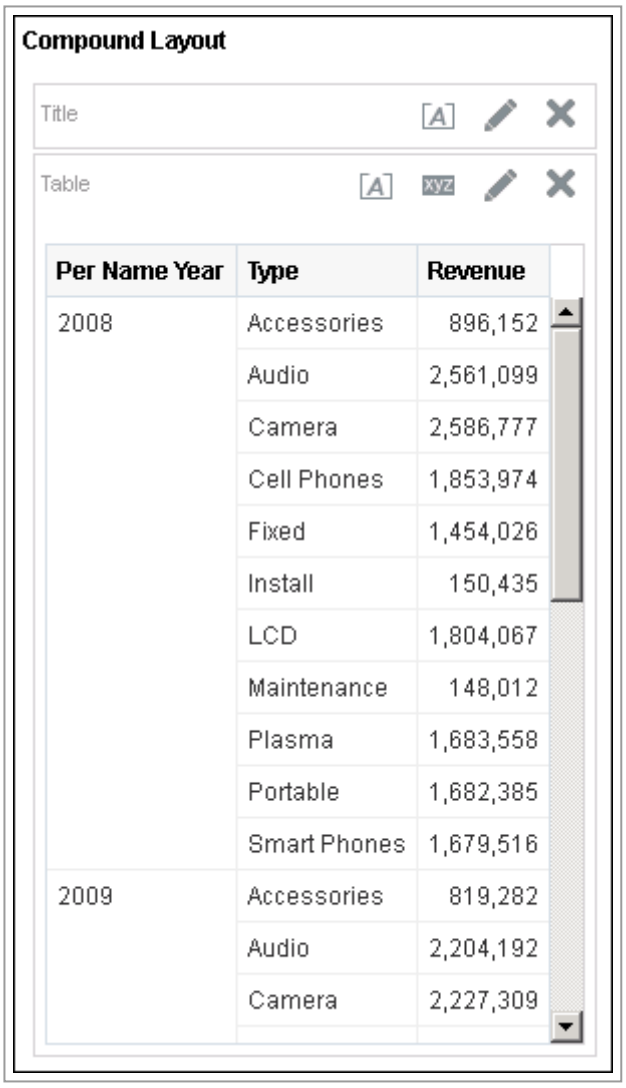
**Base Facts.Revenue**



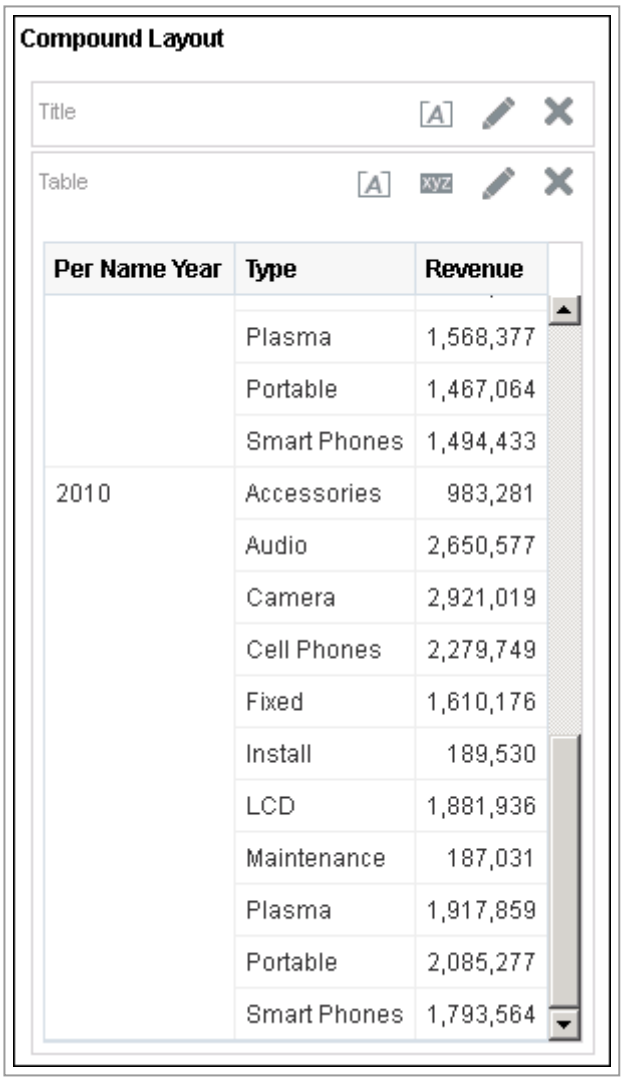

8. Click **Results**.




9. The analysis results are displayed in a compound layout, which includes a Title view and a Table view.




**10.** Use scrollbar slider of the compound layout to view additional rows.




## Managing Logical Table Sources

In this set of steps you create multiple logical table sources for the D3 Customer logical table. To create multiple logical table sources you perform the following steps:

- Opening the Repository in Offline Mode
- Adding a New Logical Table Source
- Creating Presentation Layer Objects
- Loading the Repository
- Creating and Running an Analysis

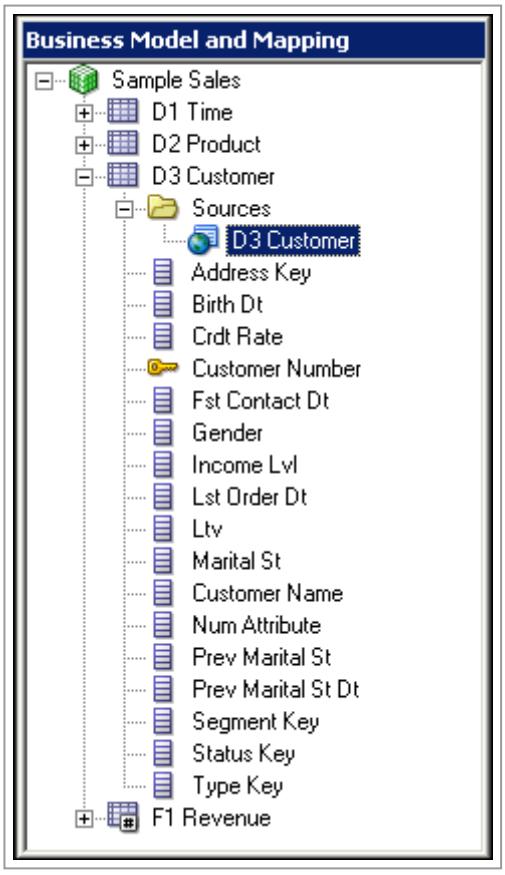
### Opening the Repository in Offline Mode

1. Return to the Administration Tool.

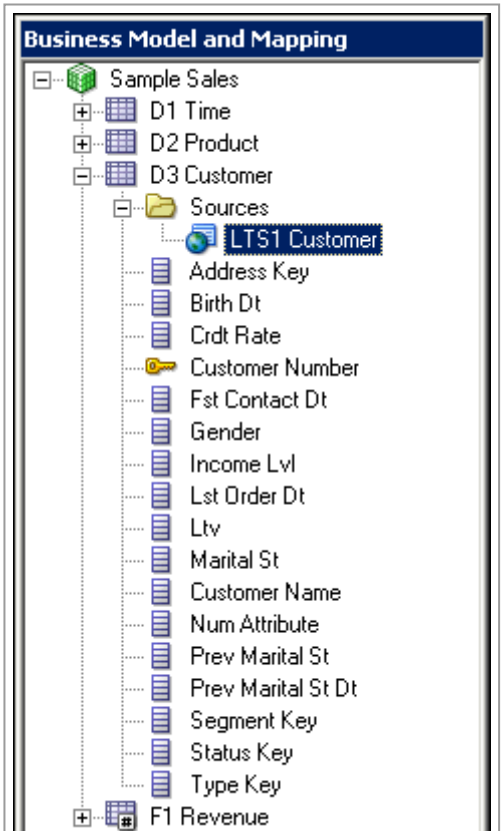
2. Open the **BISAMPLE** repository in offline mode with repository password as **Admin123**. Recall that earlier in this tutorial you created a copy of the online repository and saved it as BISAMPLE.rpd.

## Adding a New Logical Table Source

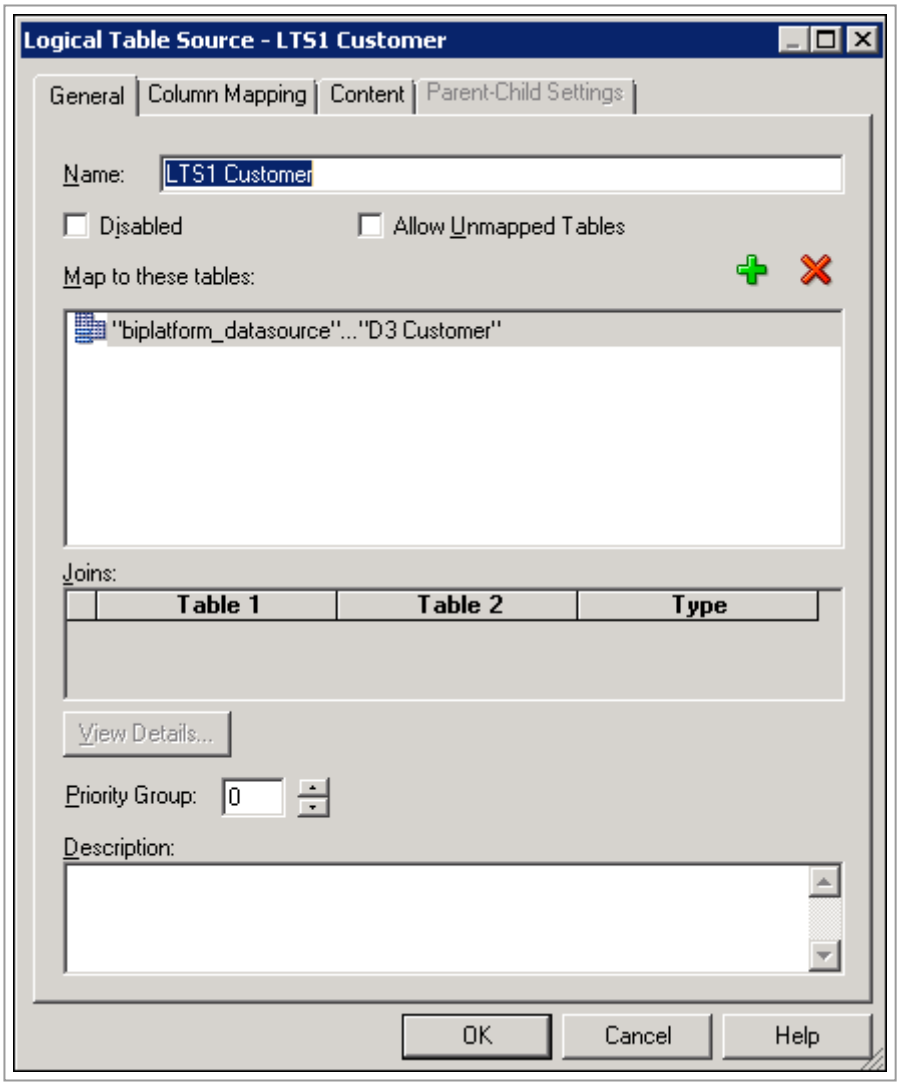
1. In the **BMM** layer, expand **Sample Sales > D3 Customer > Sources**. Notice that the D3 Customer logical table has one logical table source named **D3 Customer**.



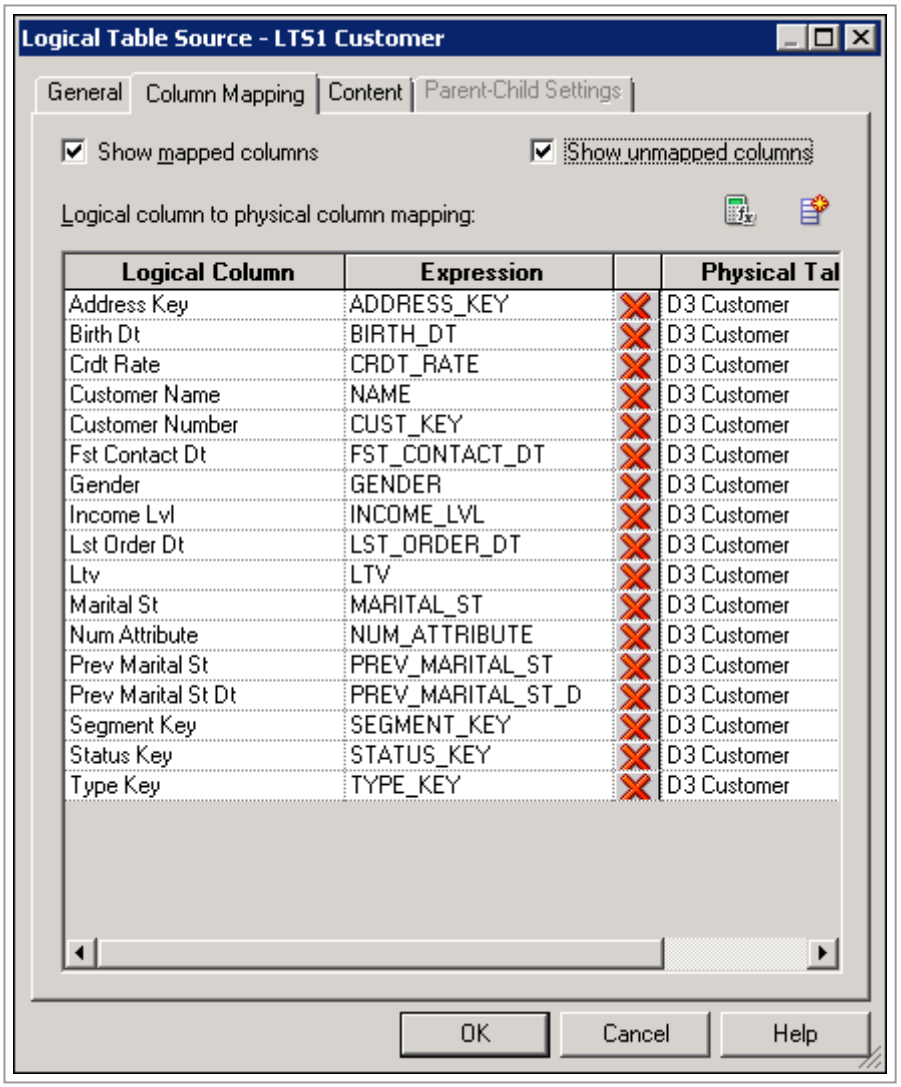

2. Rename the D3 Customer logical table source (not the logical table) to **LTS1 Customer**.



3. Double-click **LTS1 Customer** to open the Logical Table Source dialog box.

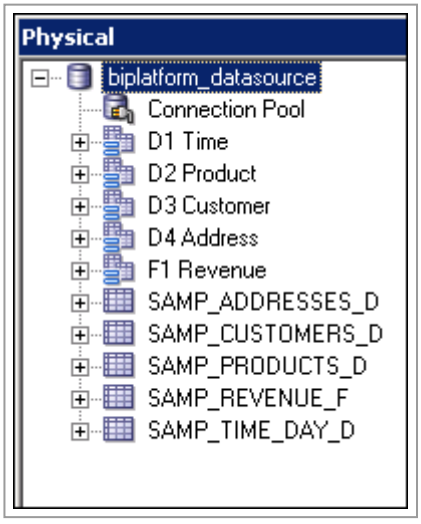



4. Click the **Column Mapping** tab and notice that all logical columns map to physical columns in the same physical table: **D3 Customer**. It may be necessary to scroll to the right to see the Physical Table column. Make sure **"Show mapped columns"** is selected.

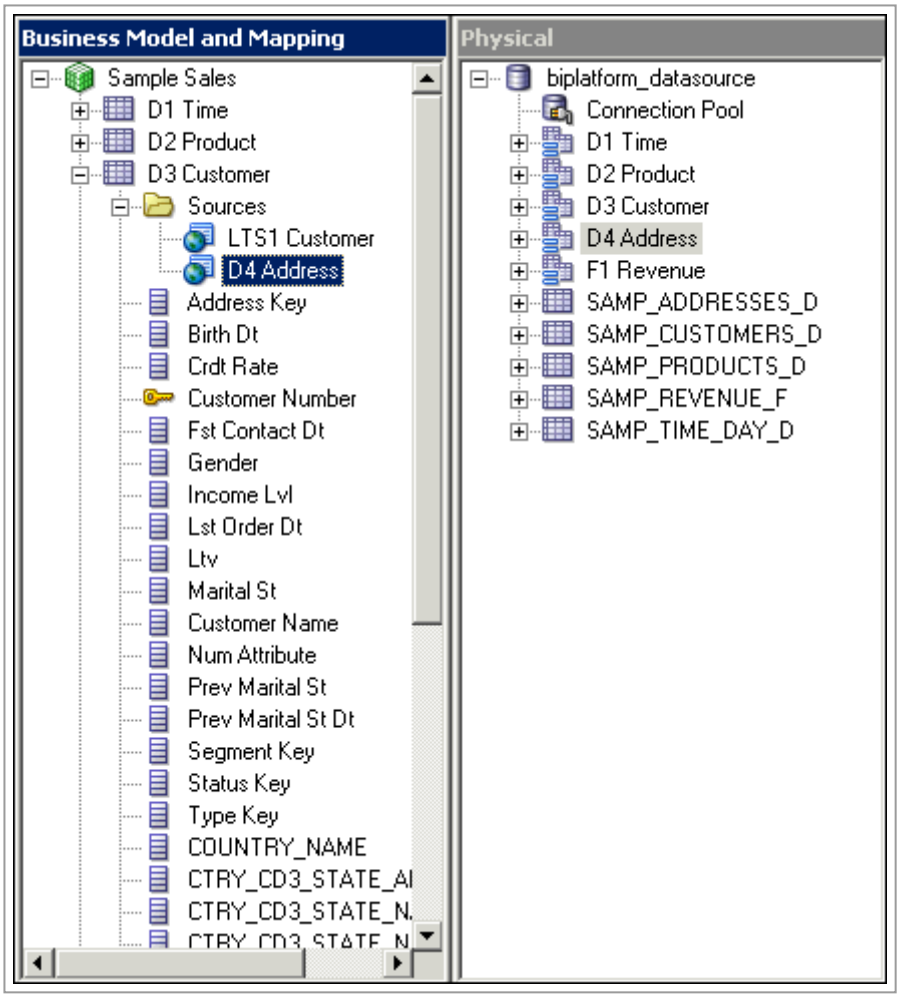



5. Click **OK** to close the Logical Table Source dialog box.

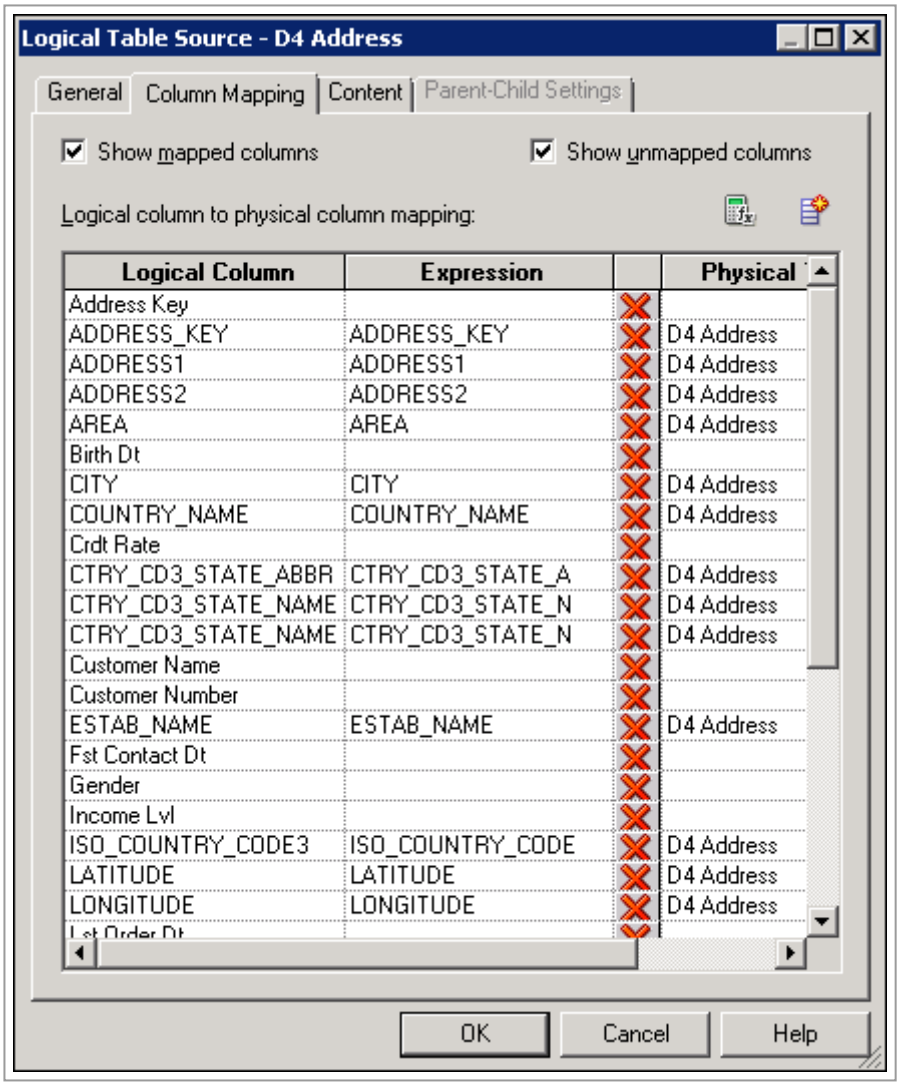
6. In the **Physical** layer, expand **biplatform_datasource**.



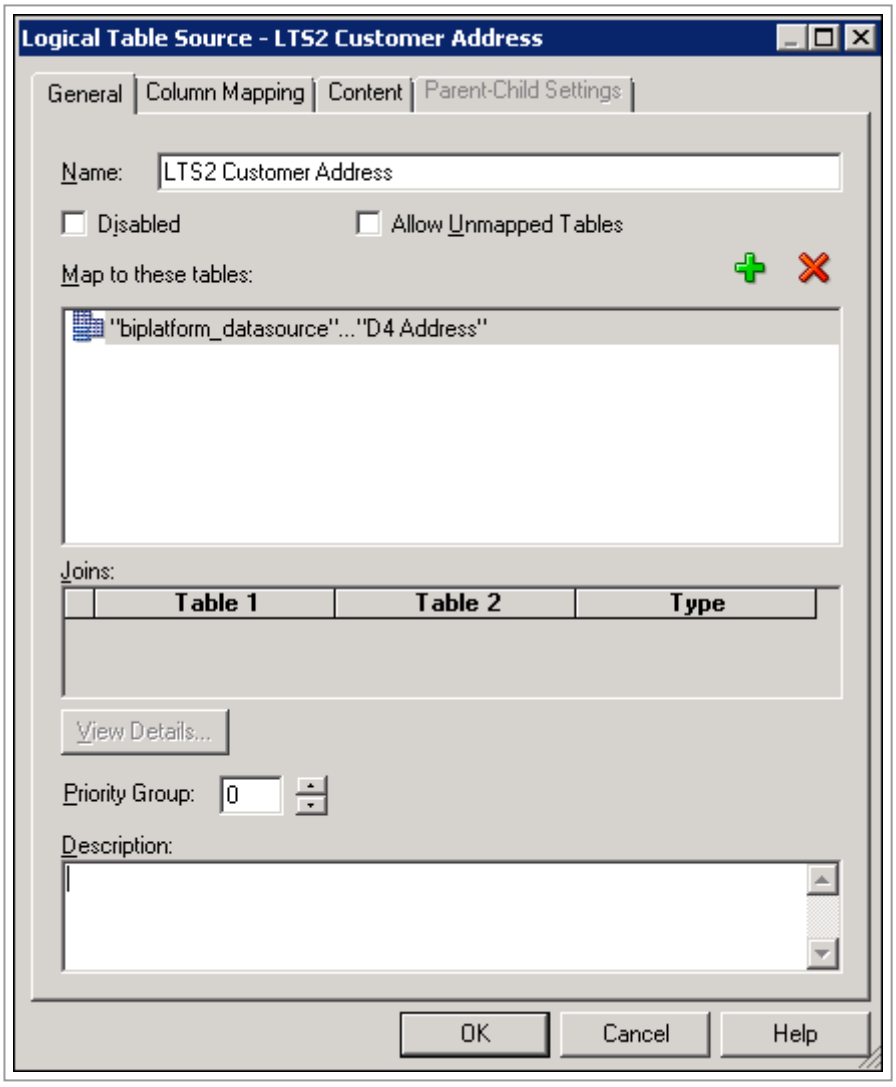

7. Drag **D4 Address** from the Physical layer to the **D3 Customer** logical table in the **BMM** layer. Notice this creates a new logical table source named **D4 Address** for the D3 Customer logical table. It also creates new logical columns that map to the D4 Address physical table.



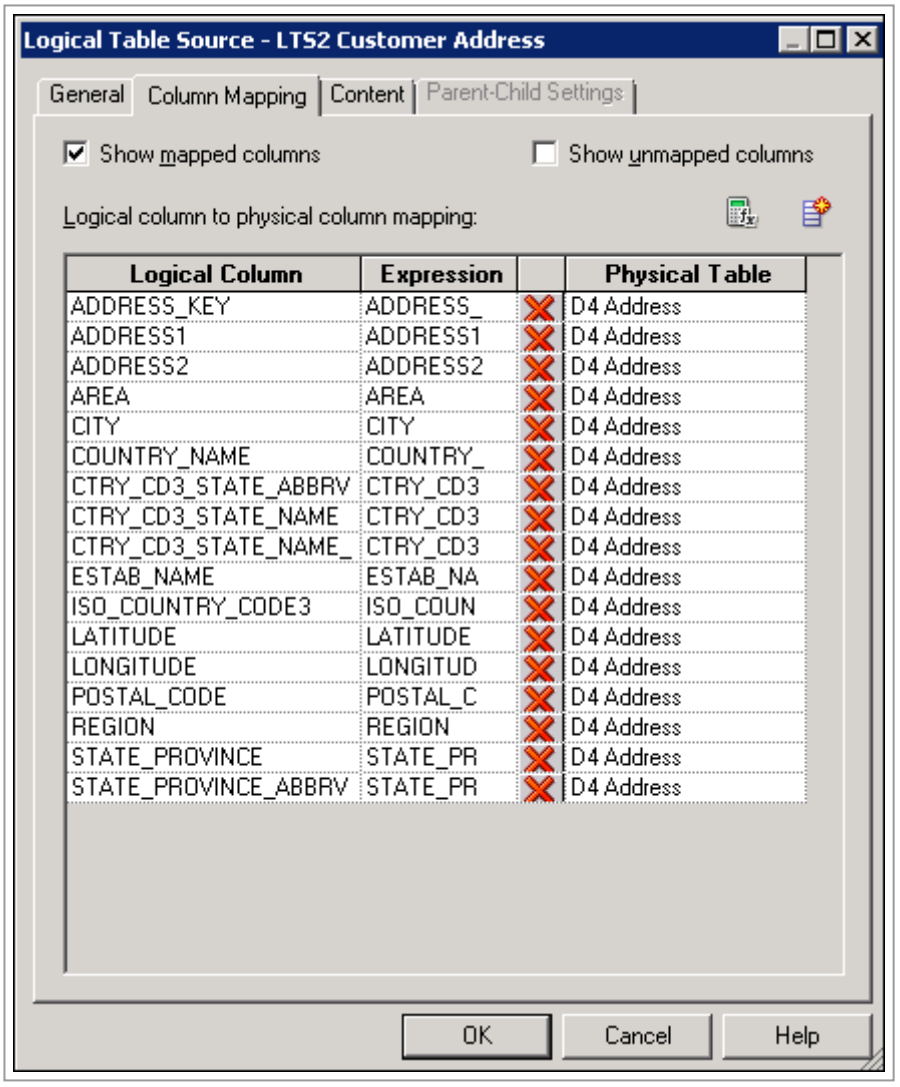

8. In the **BMM** layer, double-click the new **D4 Address** logical table source to open the Logical Table Source dialog box.



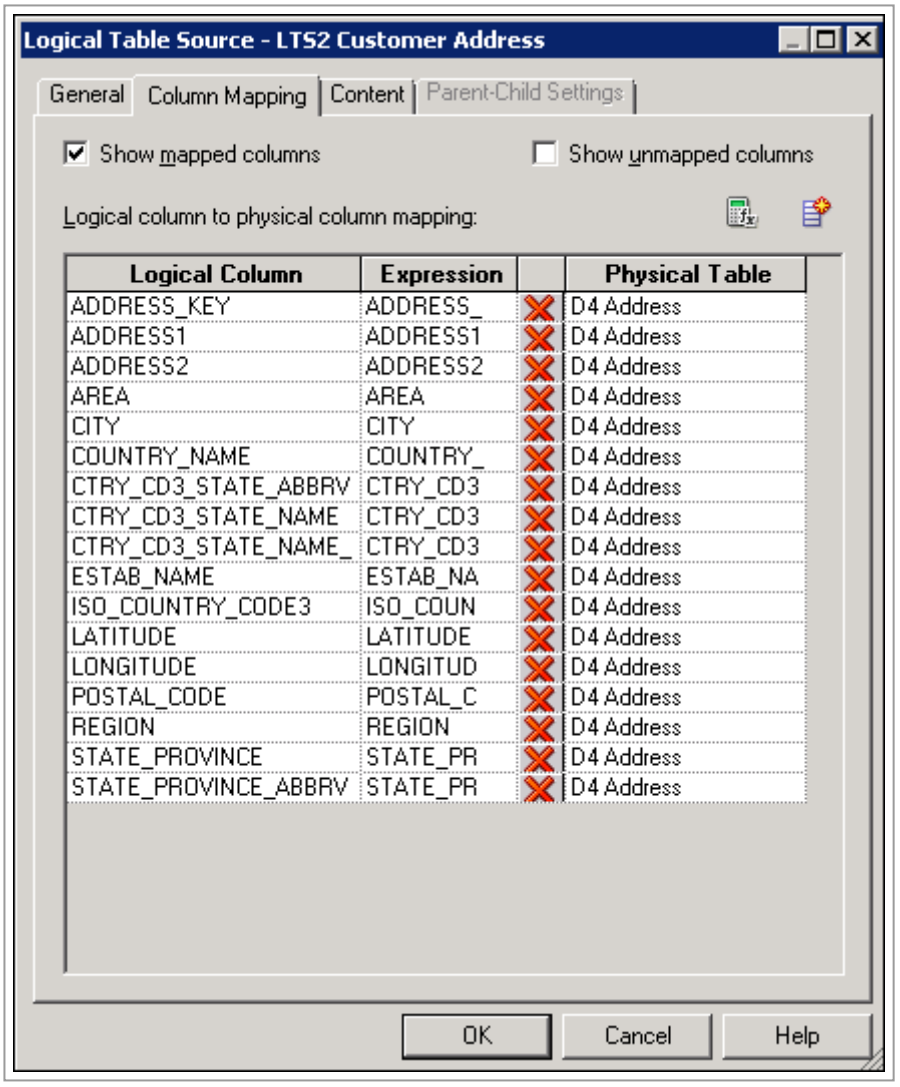

**9.** On the General tab, enter **LTS2 Customer Address** in the Name field.



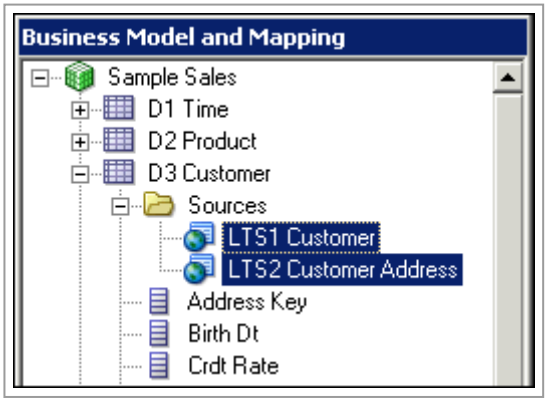

10. Click the **Column Mapping** tab and notice that all logical columns map to physical columns in the same physical table: **D4 Address**. If necessary, select **Show mapped columns** and deselect **Show unmapped columns**.



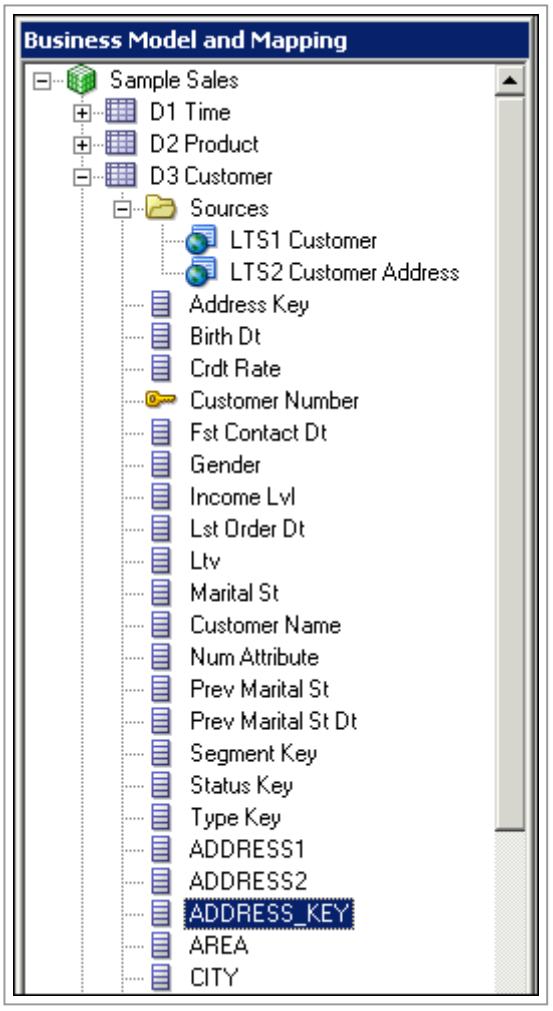

**11.** Click **OK** to close the Logical Table Source dialog box.



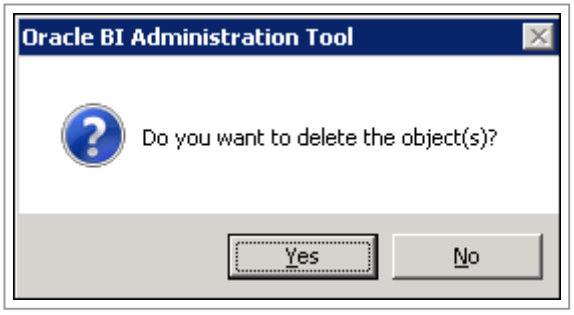

12. Confirm that the **D3 Customer** logical table now has two logical table sources: **LTS1 Customer** and **LTS2 Customer Address**. A single logical table now maps to two physical sources.



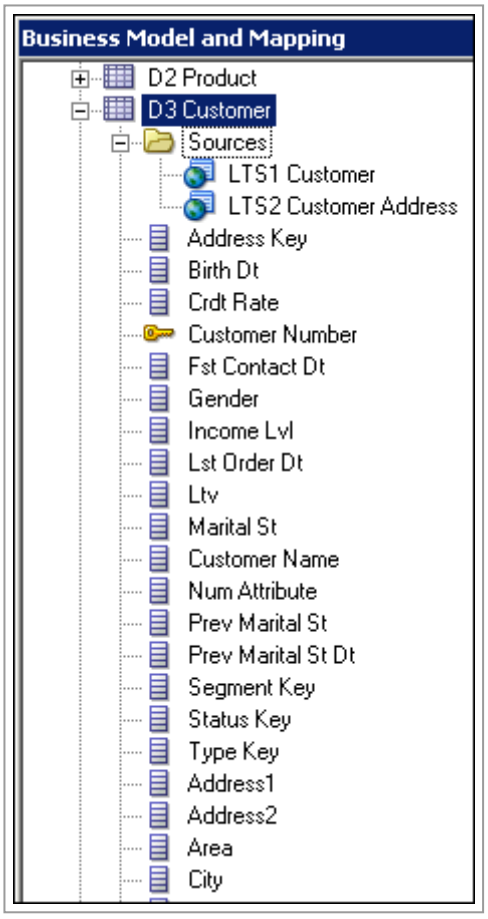

13. Right-click the new **ADDRESS_KEY** column and select **Delete**. This is a duplicate column and is not needed.




**14.** Click **Yes** to confirm the delete.



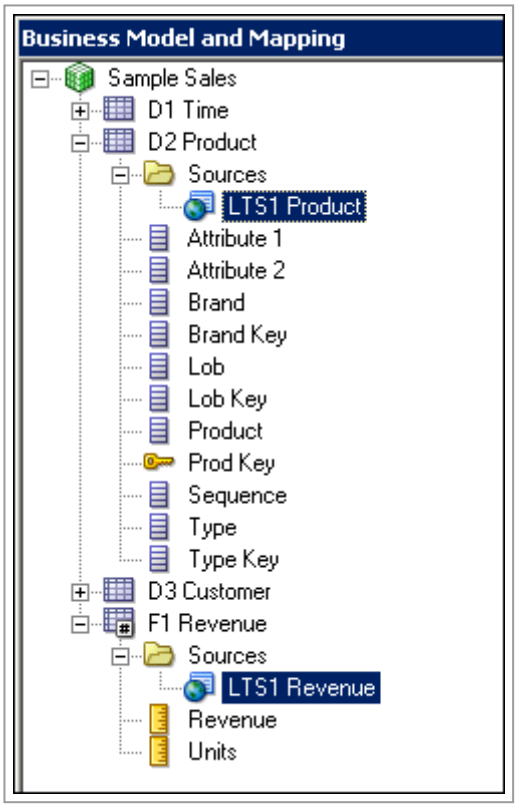

**15.** Use the Rename Wizard or a manual renaming technique to rename the new address logical columns (with uppercase letters) in D3 Customer. Your results should look similar to the screenshot. Hint: To use the Rename Wizard, select all of the new logical columns, then right-click any one of the highlighted columns and select Rename Wizard to launch the wizard. If you need help using the Rename Wizard, refer to **Renaming Objects Using the Rename Wizard** section from earlier in this tutorial.




16. Rename the remaining logical table sources according to the following table. Recall that logical table sources are located in the Sources folder for a logical table. For example: **D2 Product > Sources**.

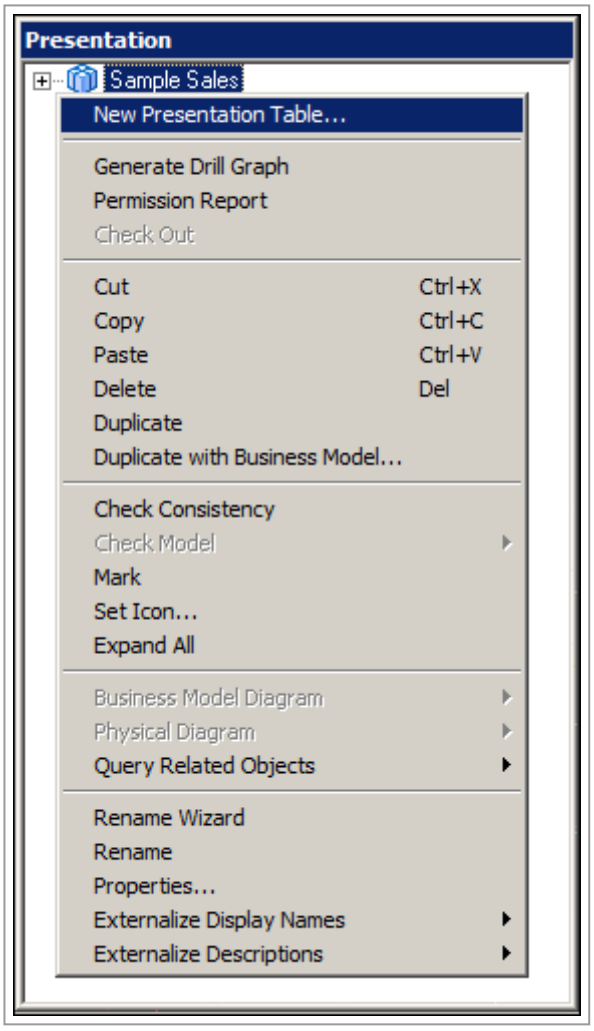
| Logical Table Source | Rename |
|---|---|
| D2 Product | LTS1 Product |
| F1 Revenue | LTS1 Revenue |

Your results should look similar to the screenshot.

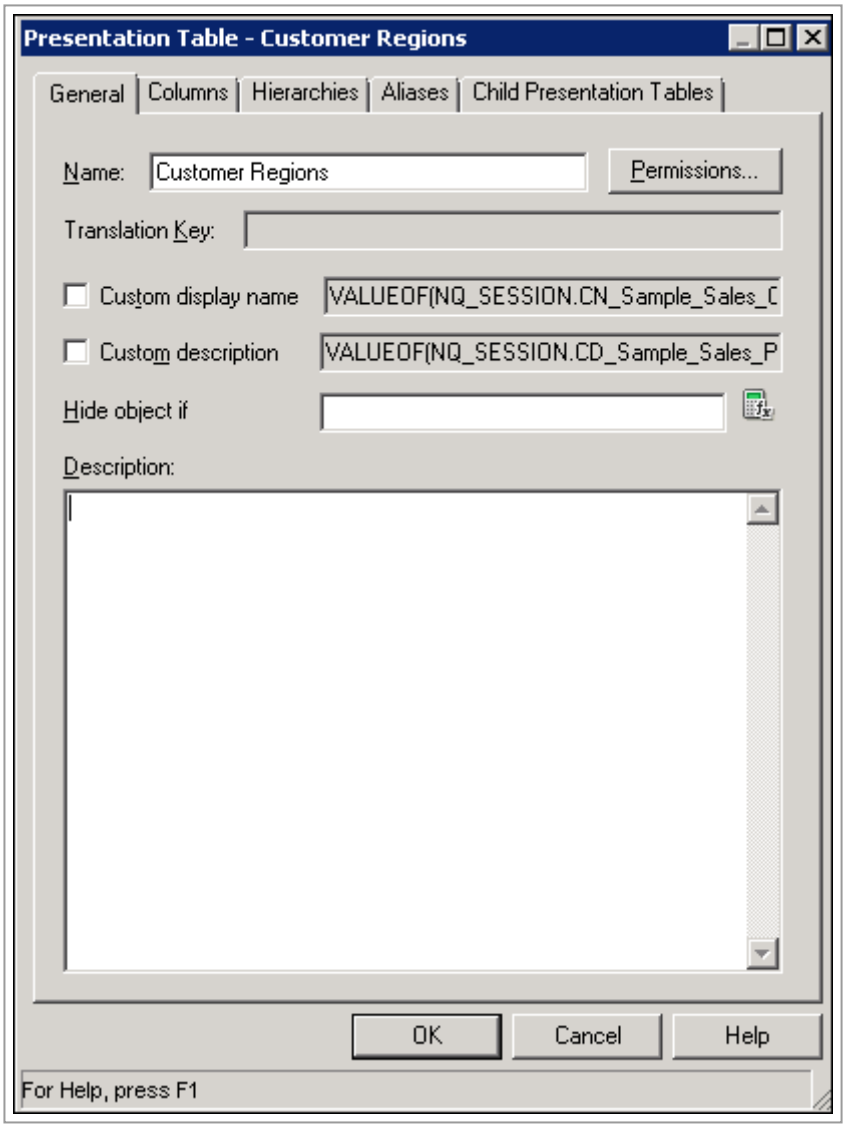



## Creating Presentation Layer Objects

1. In the **Presentation** layer, right-click the **Sample Sales** subject area and select **New Presentation Table** to open the Presentation Table dialog box.

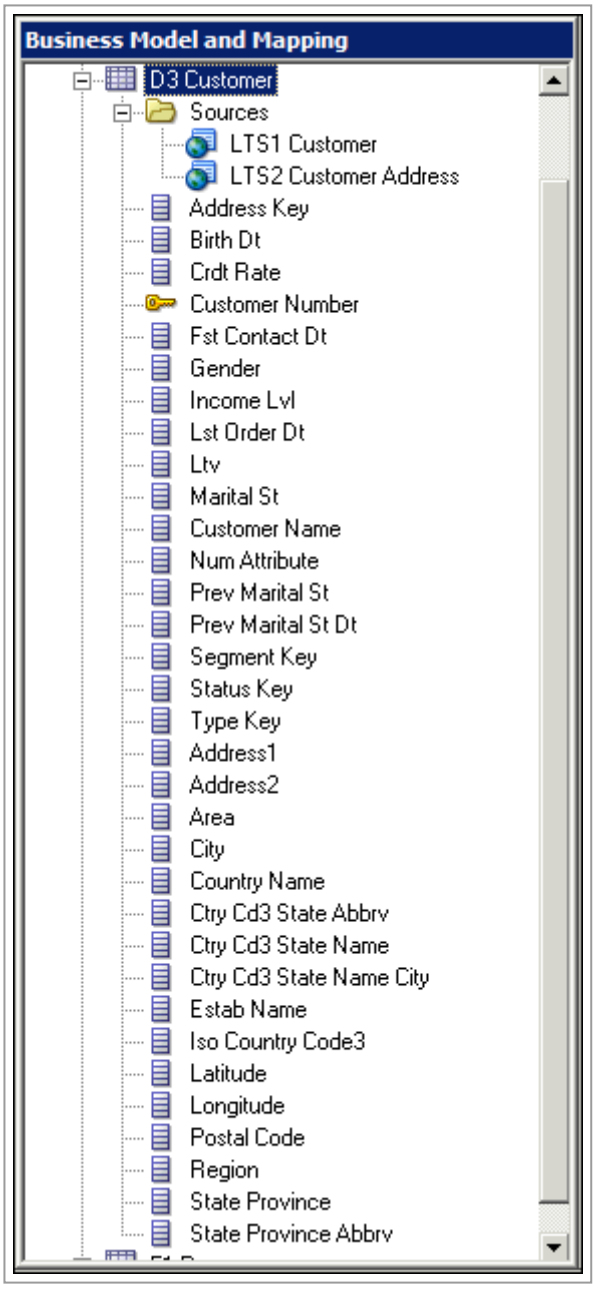



**2.** On the **General** tab, enter **Customer Regions** in the Name field.




3. Click **OK** to close the **Presentation Table** dialog box. Confirm that the **Customer Regions** presentation table is added to the Sample Sales subject area in the **Presentation** layer.

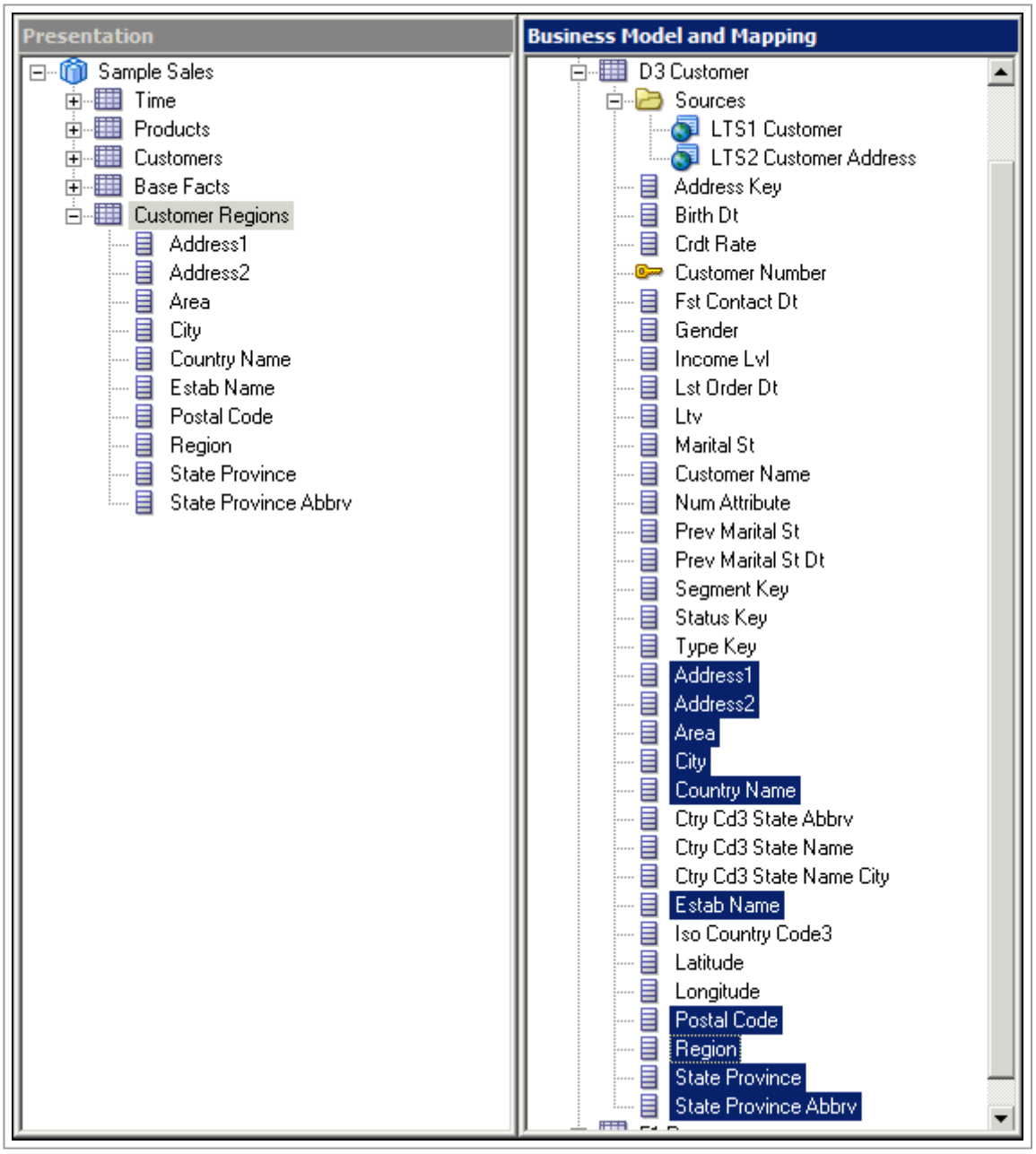

4. In the **BMM** layer, expand **Sample Sales > D3 Customer**.




5. Drag the following logical columns from **D3 Customer** to **Customer Regions** in the **Presentation** layer:

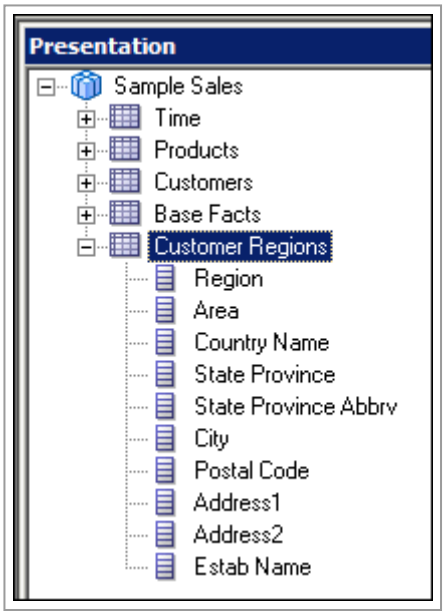
**Address 1**
**Address 2**
**Area**
**City**
**Country Name**
**Estab Name**
**Postal Code**
**Region**
**State Province**
**State Province Abbrv**

Your column names may be slightly different depending on how you renamed them.

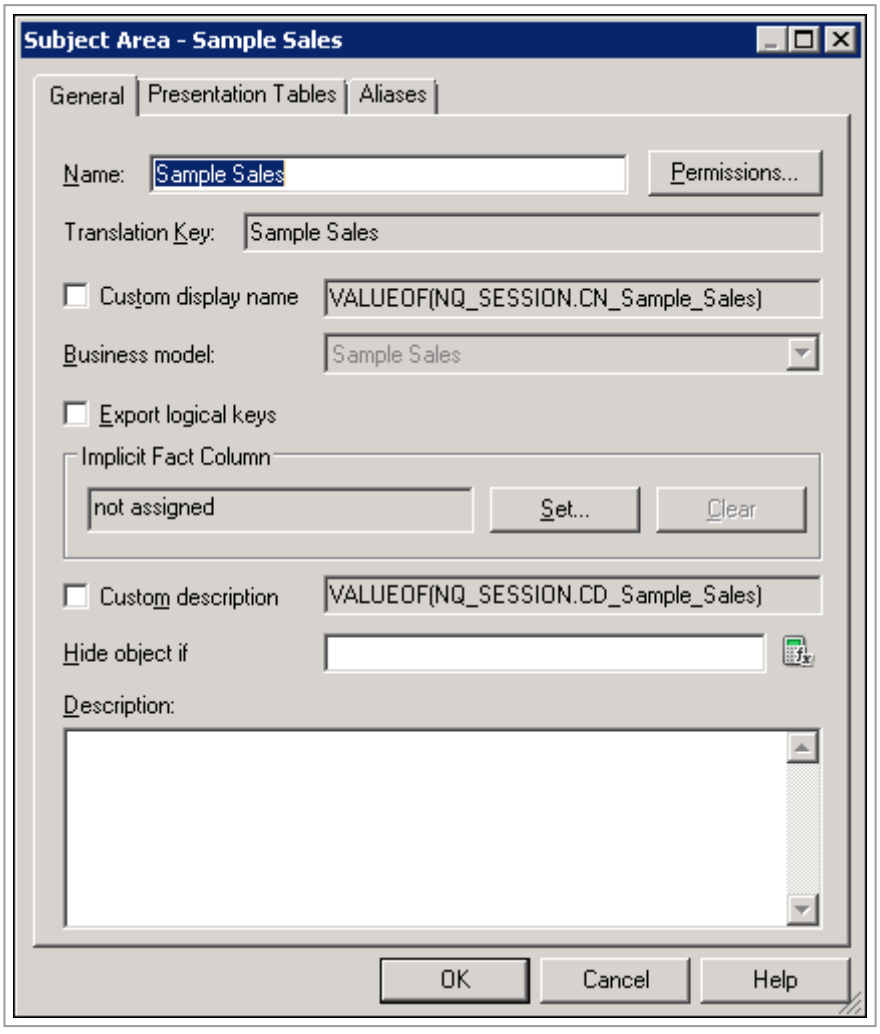



6. Reorder the Customer Regions presentation columns in the following order, from top to bottom:

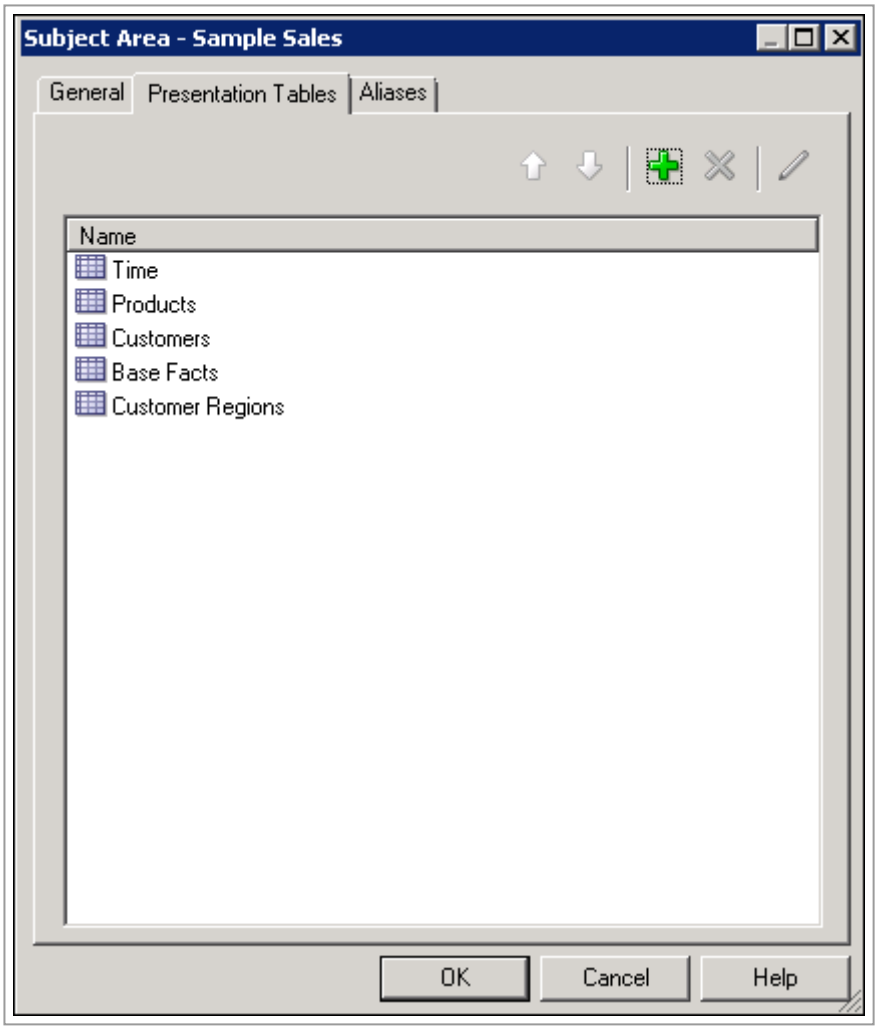
**Region**
**Area**
**Country Name**
**State Province**
**State Province Abbrv**
**City**
**Postal Code**
**Address 1**
**Address 2**
**Estab Name**



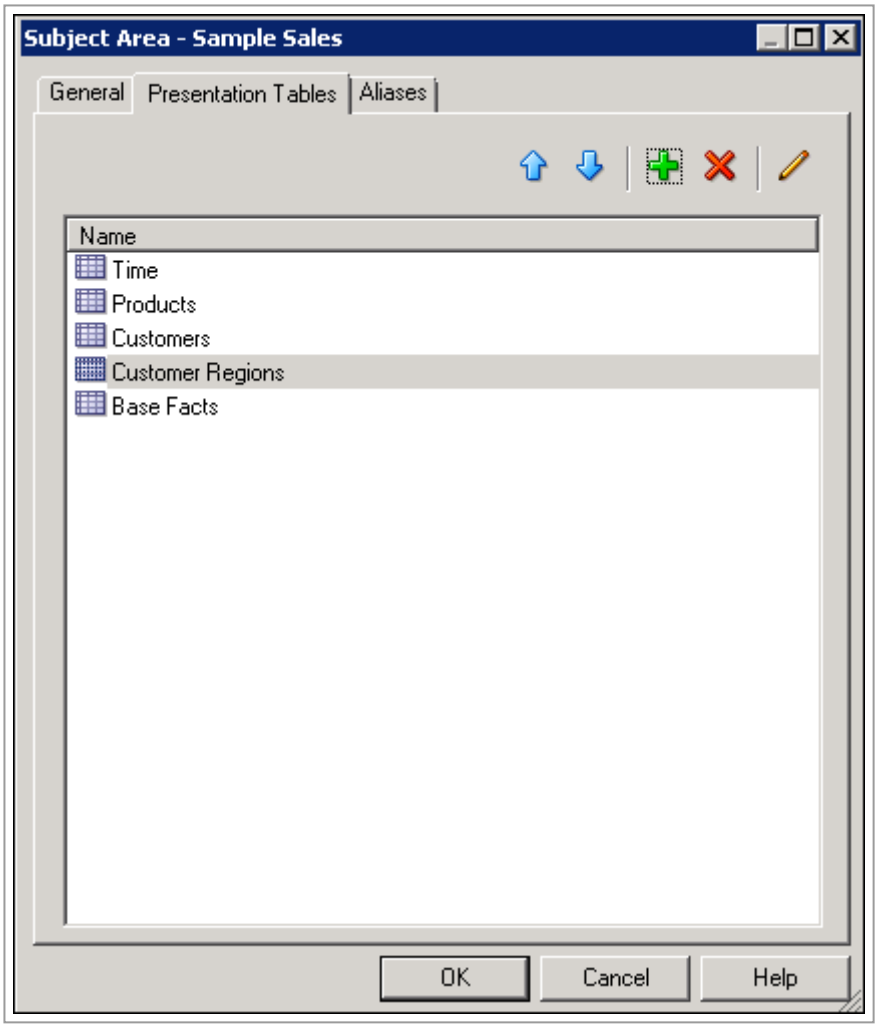

7. Double-click the **Sample Sales** subject area in the **Presentation** layer to open the Subject Area dialog box.



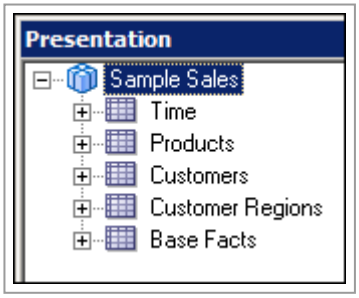

8. Click the **Presentation Tables** tab.




**9.** Reorder the presentation tables so that **Customer Regions** appears after **Customers**.

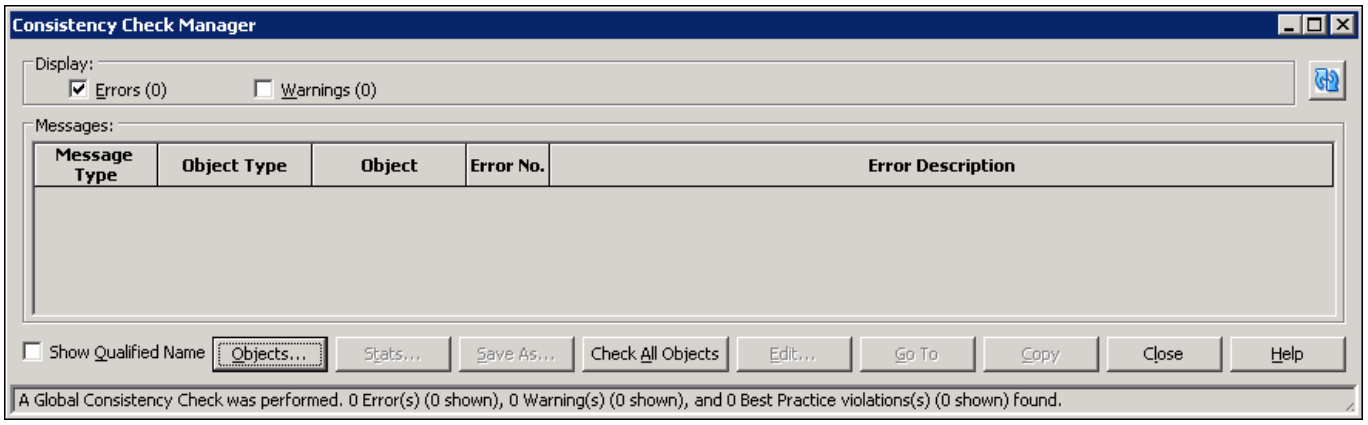



**10.** Click **OK** to close the Subject Area dialog box. Confirm that the presentation tables appear in the expected order.




You now have two presentation tables, Customers and Customer Regions, mapped to the same logical table, D3 Customer. The D3 Customer logical table is mapped to two physical sources: D3 Customer and D4 Address.

**11.** Save the repository and check global consistency when prompted. You should receive a message that there are no errors, warnings, or best practice violations to report.




If you do receive any consistency check errors or warnings, fix them before proceeding.

12. Click **Close** to close the Consistency Check Manager dialog box.

13. Close the repository and keep the Administration Tool open.

## Loading the Repository

1. Repeat the steps described in the Testing and Validating Repostory, Loading the Repository section above to upload the saved repository to the BI Server.

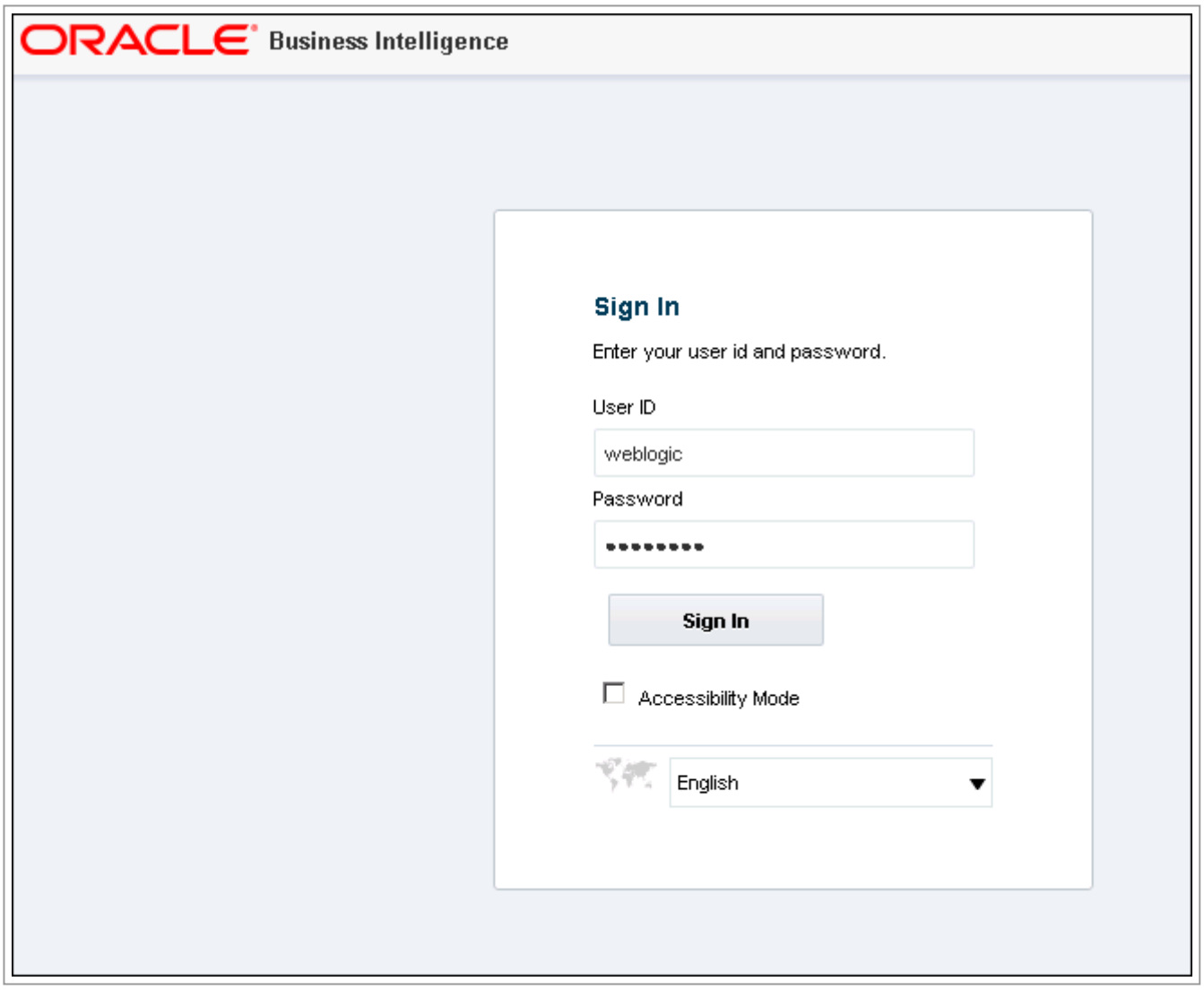
## Creating and Running an Analysis

1. Return to **Oracle BI**, which should still be open. If not, open a browser or browser tab and enter the following URL to navigate to Oracle Business Intelligence:

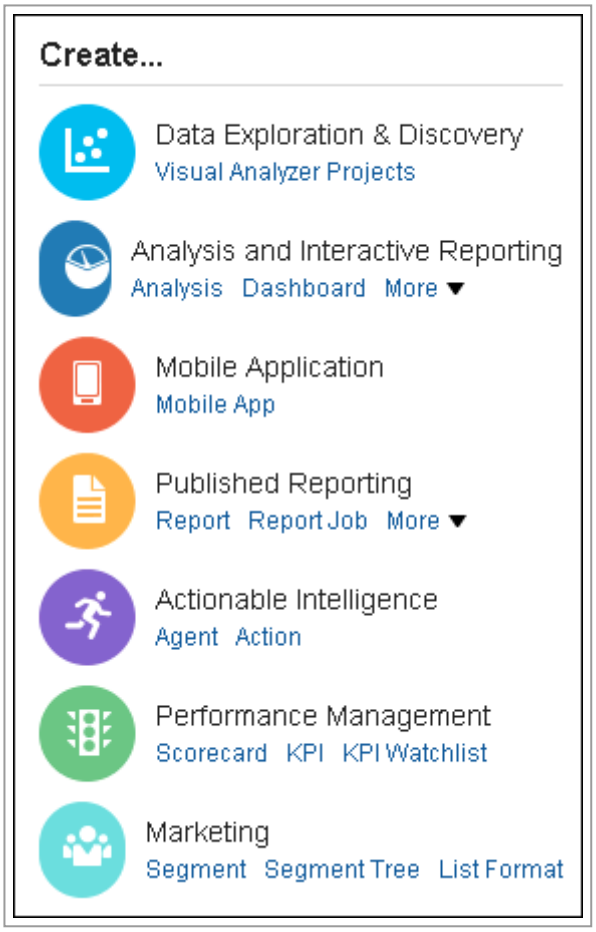
   **http://<machine name>/:9502/analytics**

   In this tutorial the URL is http://localhost:9502/analytics.

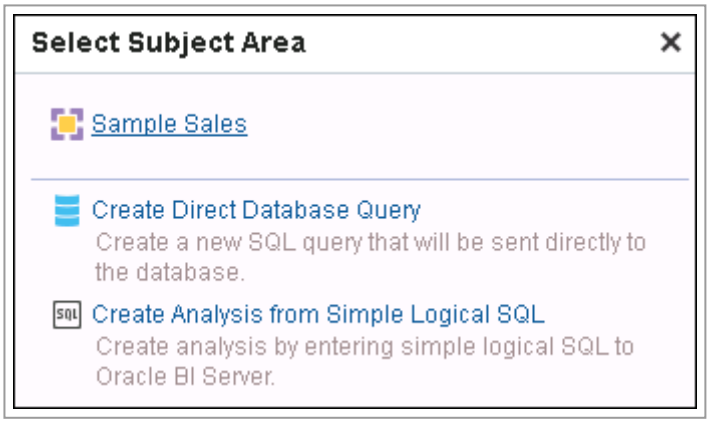
2. If your previous session has timed out, sign in as an administrative user. Typically you use the administrative user name and password provided during the Oracle BI installation. In this example the user name is **weblogic**.



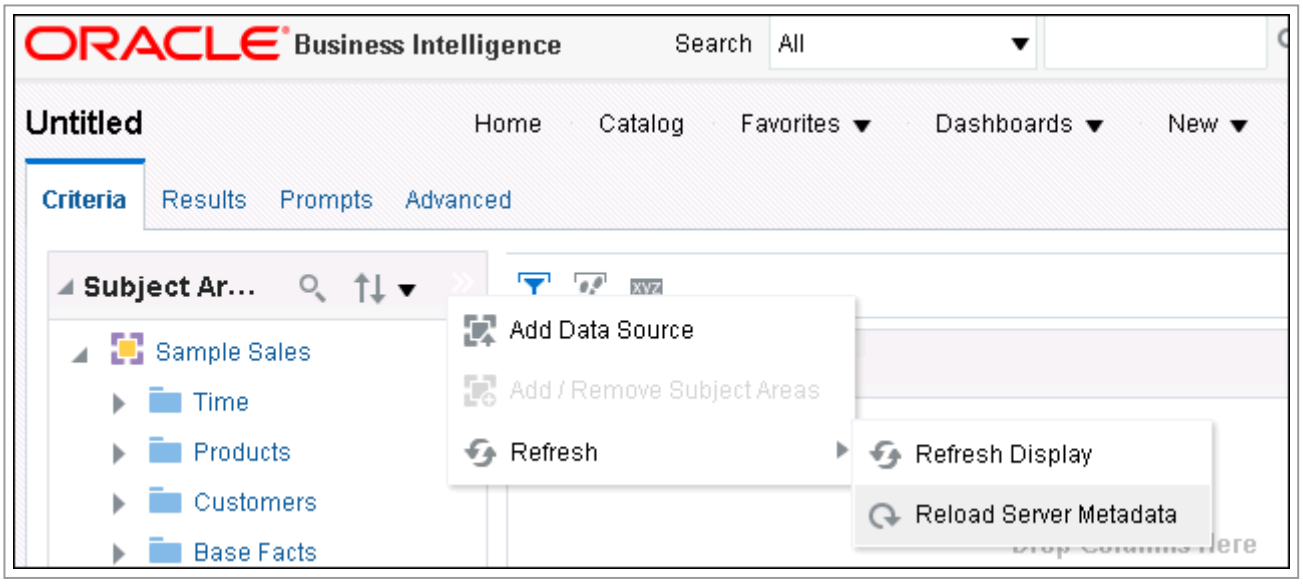

**3.** In the left navigation pane, under Create... Analysis and Interactive Reporting, select **Analysis**.



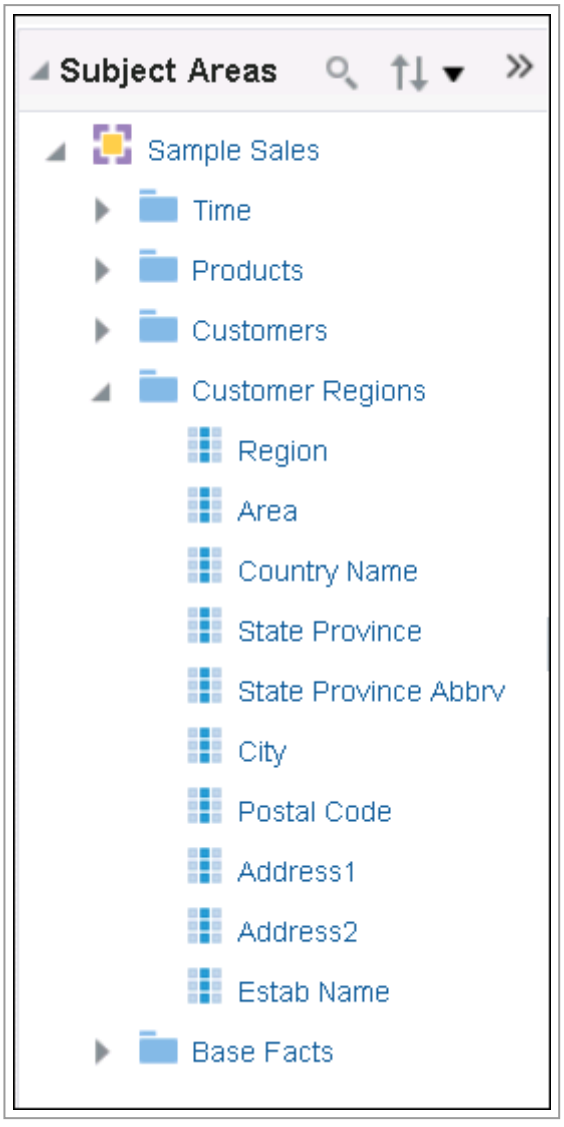

**4.** Select the **Sample Sales** subject area.




**5.** Select **"Reload Server Metadata"**.



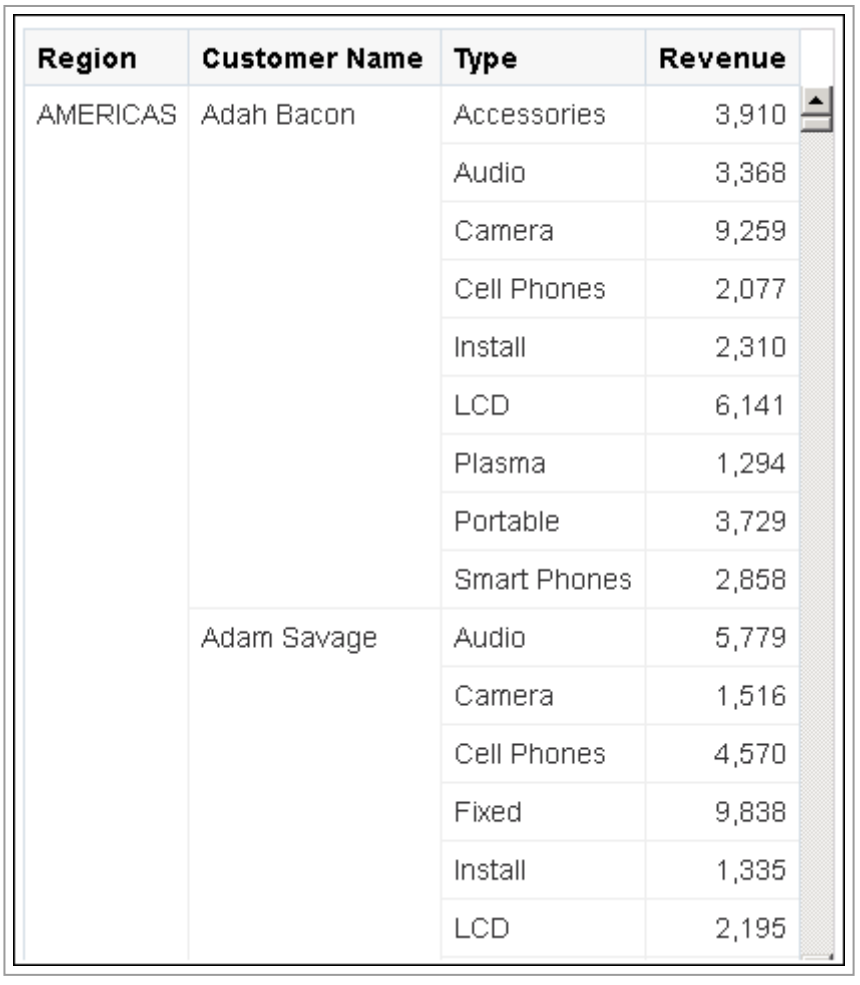

6. In the left navigation pane, expand the folders and confirm that the **Customer Regions** folder and corresponding columns appear.




7. Create the following analysis by double-clicking column names in the Subject Areas pane:

**Customer Regions.Region**
**Customers.Customer Name**
**Products.Type**
**Base Facts.Revenue**


8. Click **Results** to view the analysis results. Use the **Get more rows** button at the bottom of the results screen to see more rows.



| Region | Customer Name | Type | Revenue |
|---|---|---|---|
| AMERICAS | Adah Bacon | Accessories | 3,910 |
| | | Audio | 3,368 |
| | | Camera | 9,259 |
| | | Cell Phones | 2,077 |
| | | Install | 2,310 |
| | | LCD | 6,141 |
| | | Plasma | 1,294 |
| | | Portable | 3,729 |
| | | Smart Phones | 2,858 |
| | Adam Savage | Audio | 5,779 |
| | | Camera | 1,516 |
| | | Cell Phones | 4,570 |
| | | Fixed | 9,838 |
| | | Install | 1,335 |
| | | LCD | 2,195 |

## Creating Calculation Measures

In this set of steps you use existing measures to created a derived calculation measure. To create a derived calculation measure you perform the following steps:

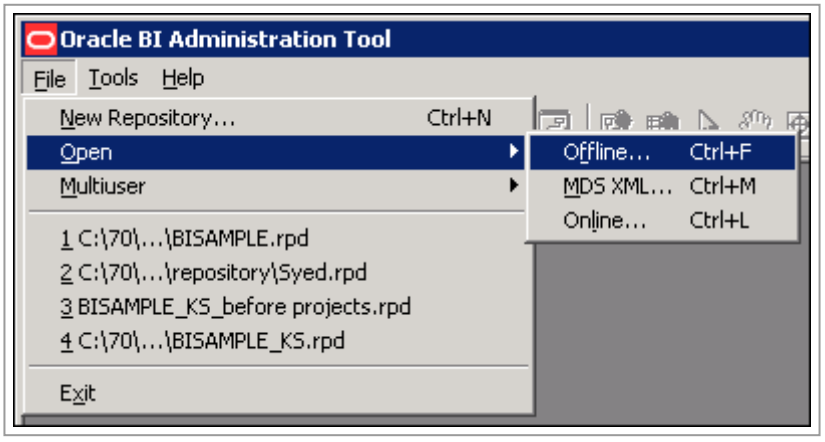
- Opening the Repository in Offline Mode
- Creating a Calculation Measure Derived from Existing Columns

- Creating a Calculation Measure Using a Function
- Loading the Repository
- Creating and Running an Analysis

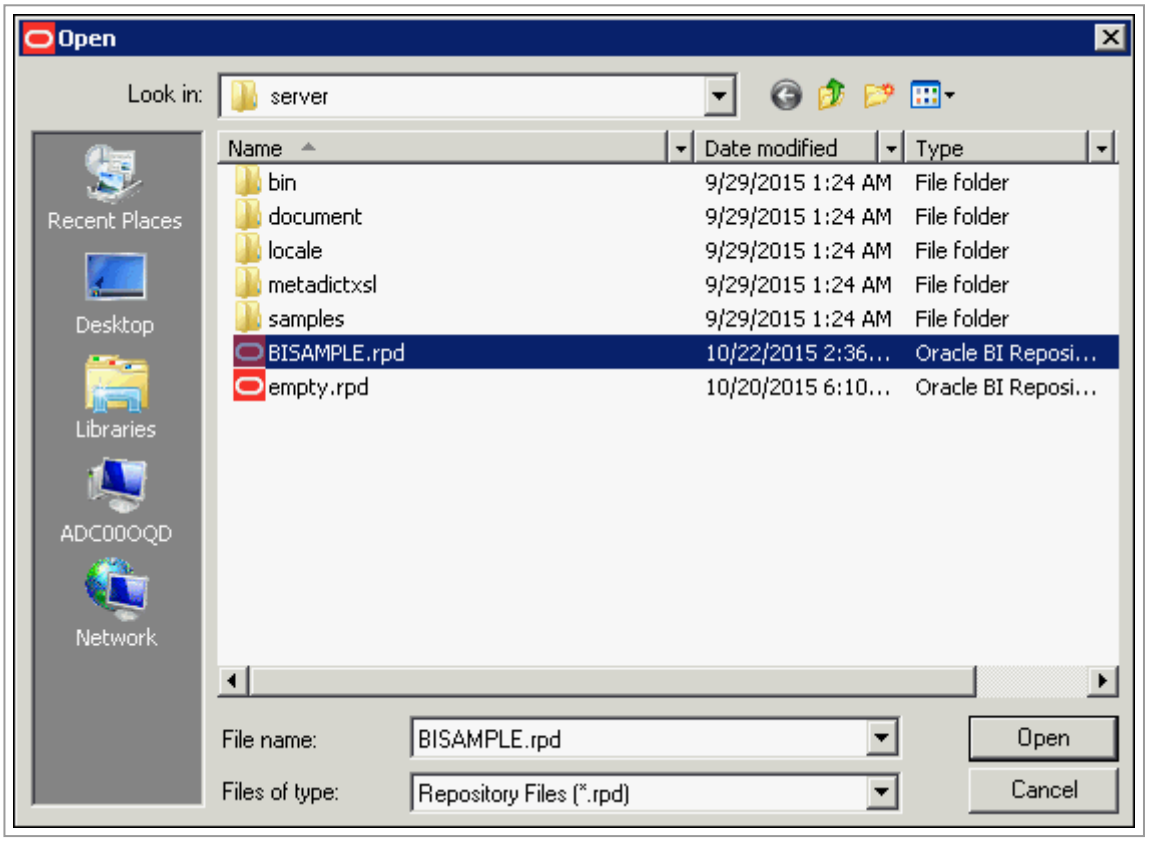
## Opening the Repository in Offline Mode

**1.** Return to the Administration Tool.

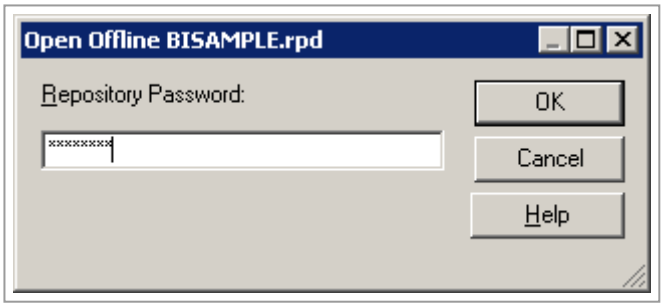
**2.** Select **File > Open > Offline**.




**3.** Select **BISAMPLE.rpd** and click **Open**.

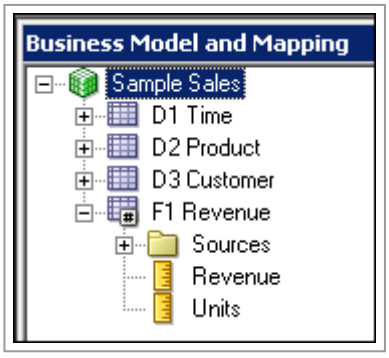



**4.** Enter **Admin123** as the repository password and click **OK** to open the repository.

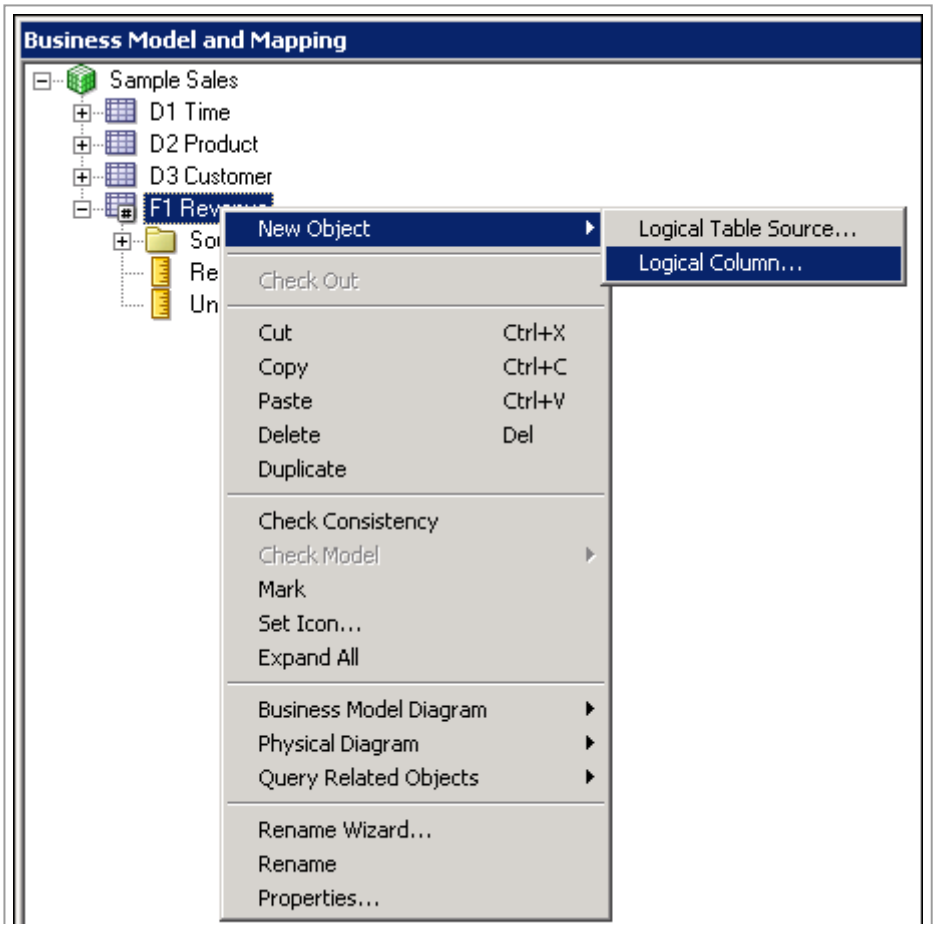



## Creating a Calculation Measure Derived from Existing Columns

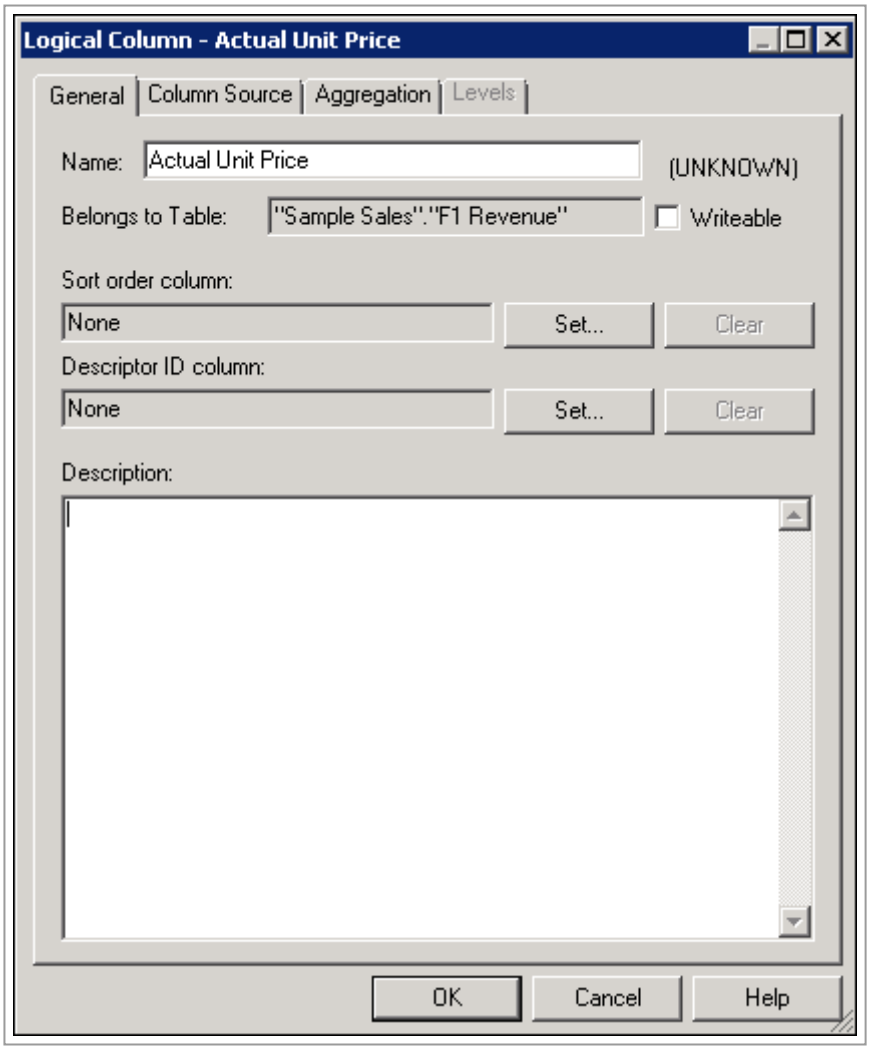
1. In the **BMM** layer, expand **Sample Sales > F1 Revenue**.



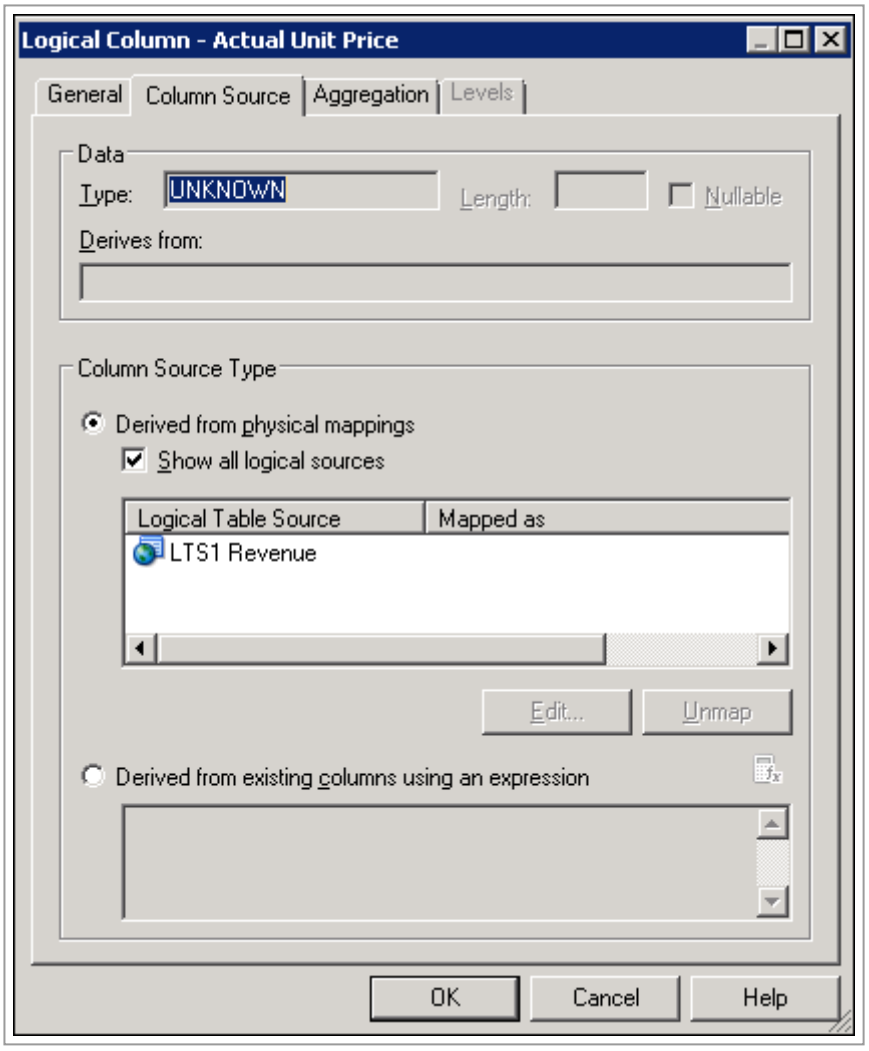

2. Right-click **F1 Revenue** and select **New Object > Logical Column** to open the Logical Column dialog box.



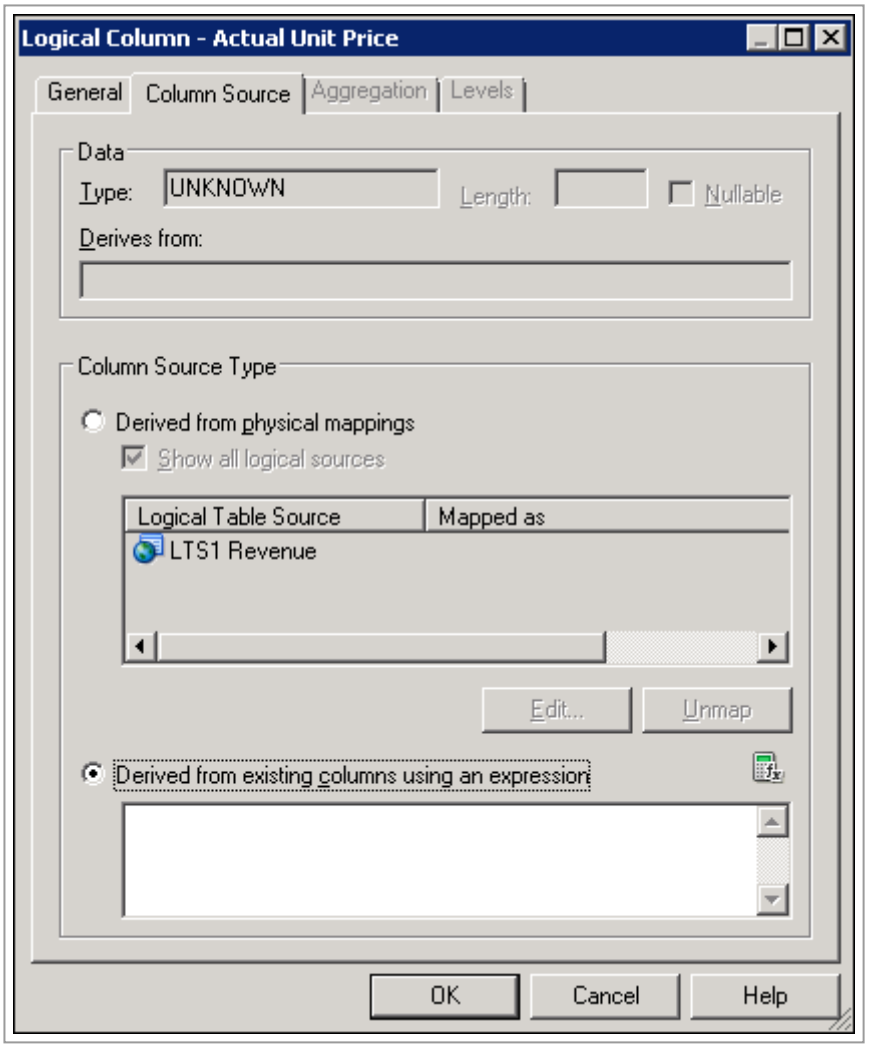
3. On the General tab, enter **Actual Unit Price** in the Name field.



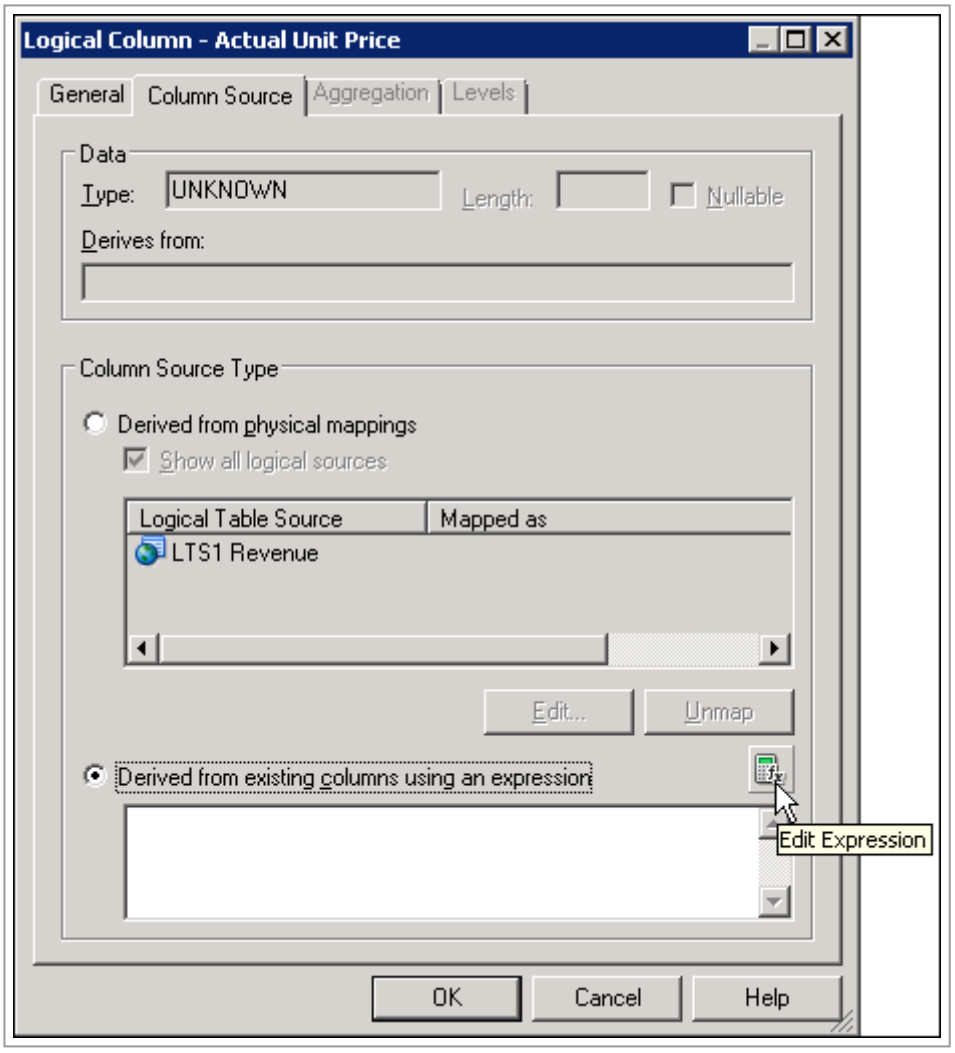

4. Click the **Column Source** tab.



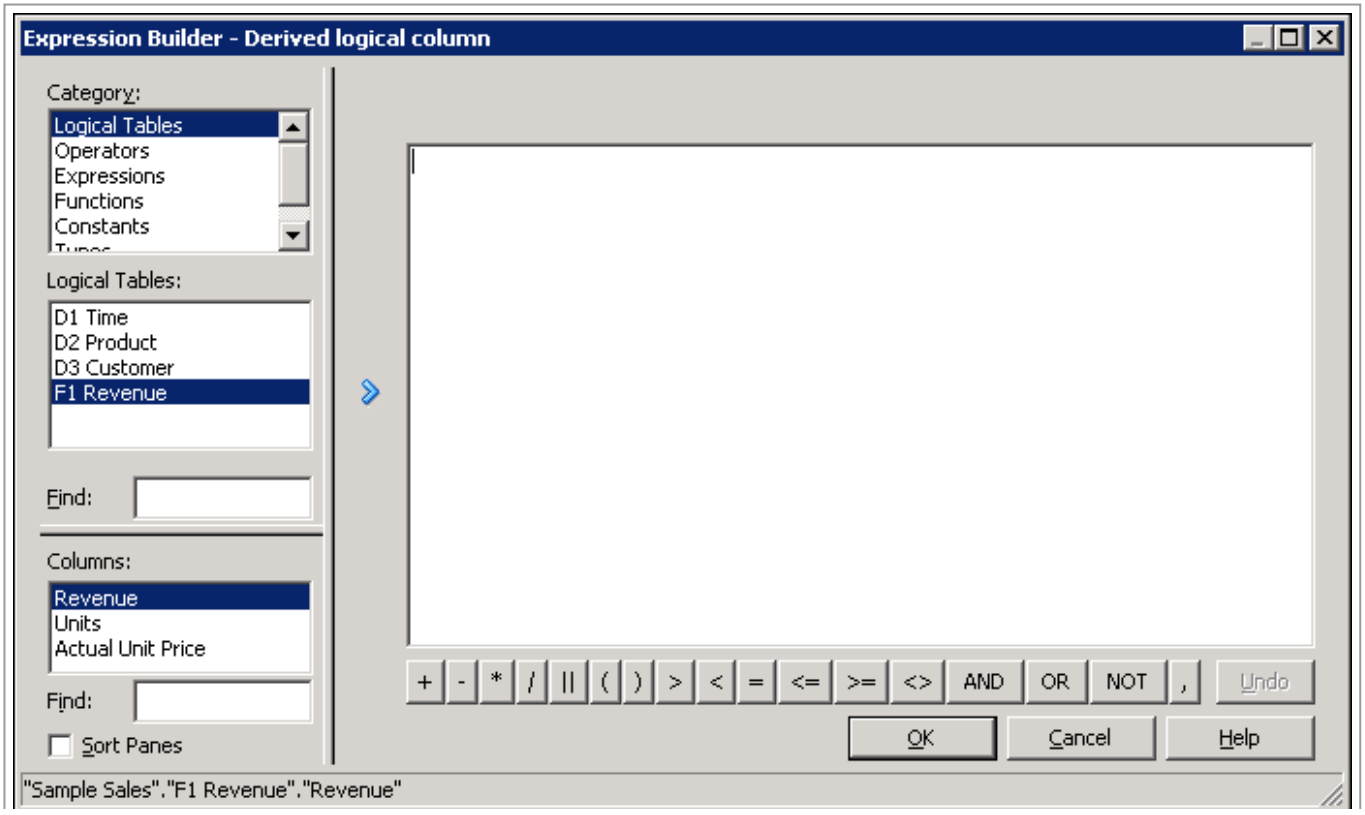

**5.** Select **Derived from existing columns using an expression**.



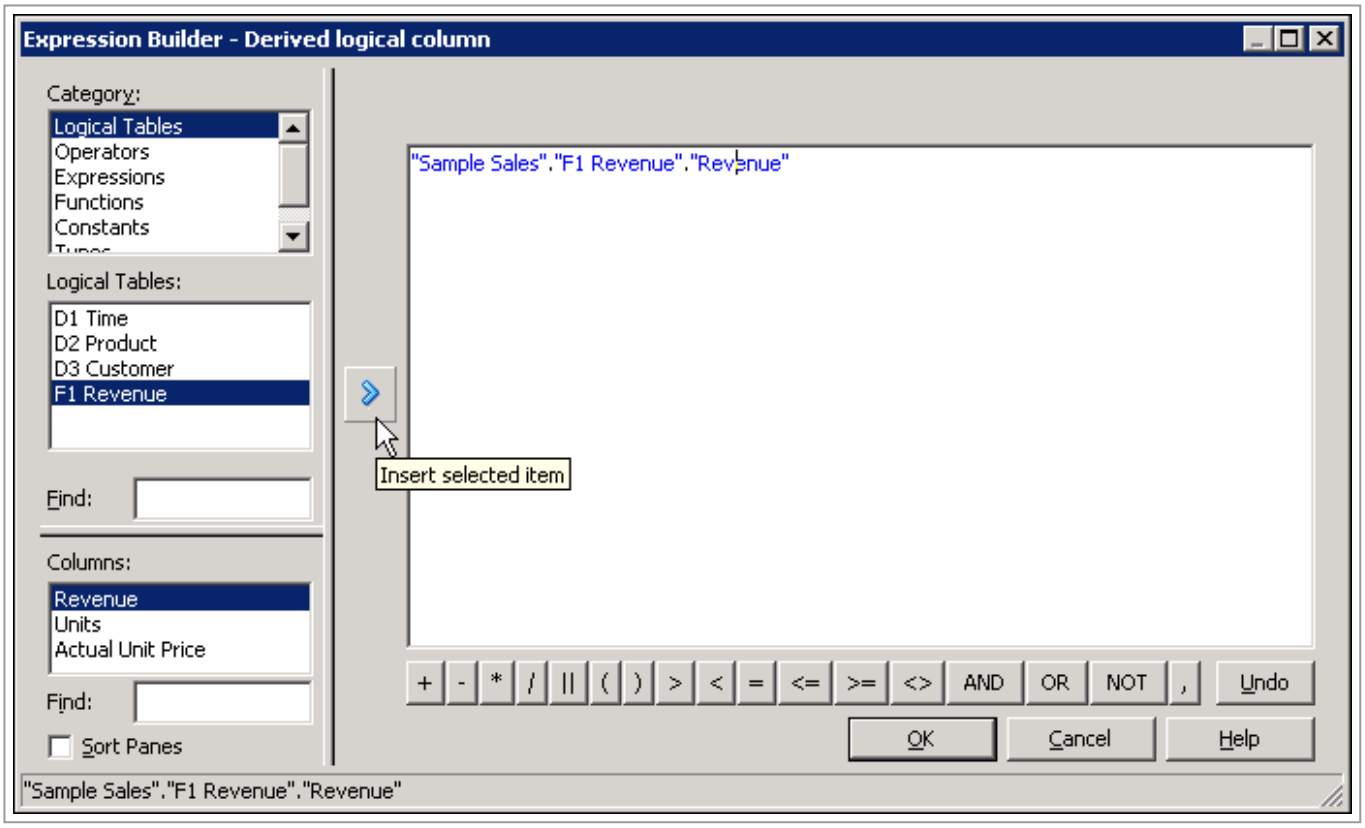

**6.** Click the **Edit Expression** button to open Expression Builder.



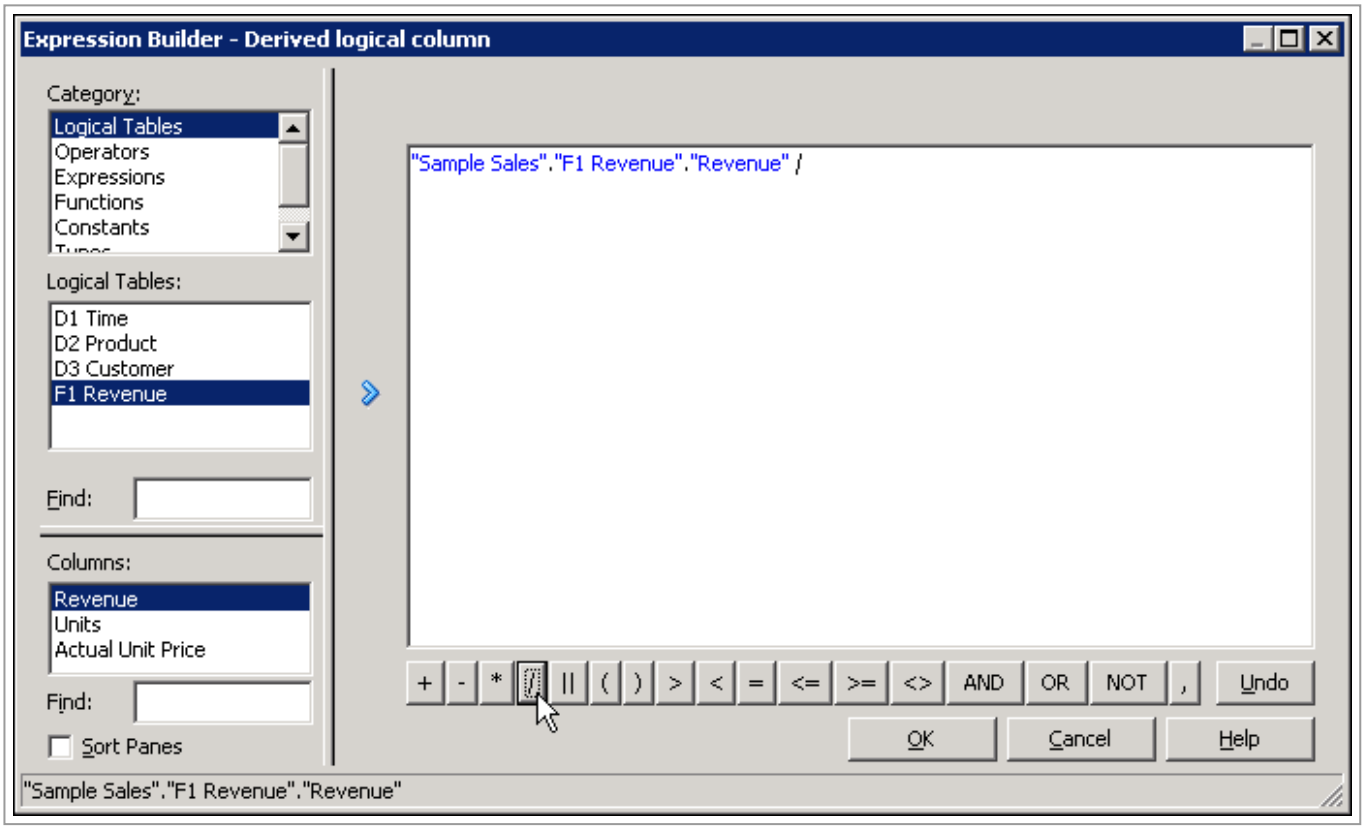

**7.** In the left pane select **Logical Tables > F1 Revenue > Revenue**.



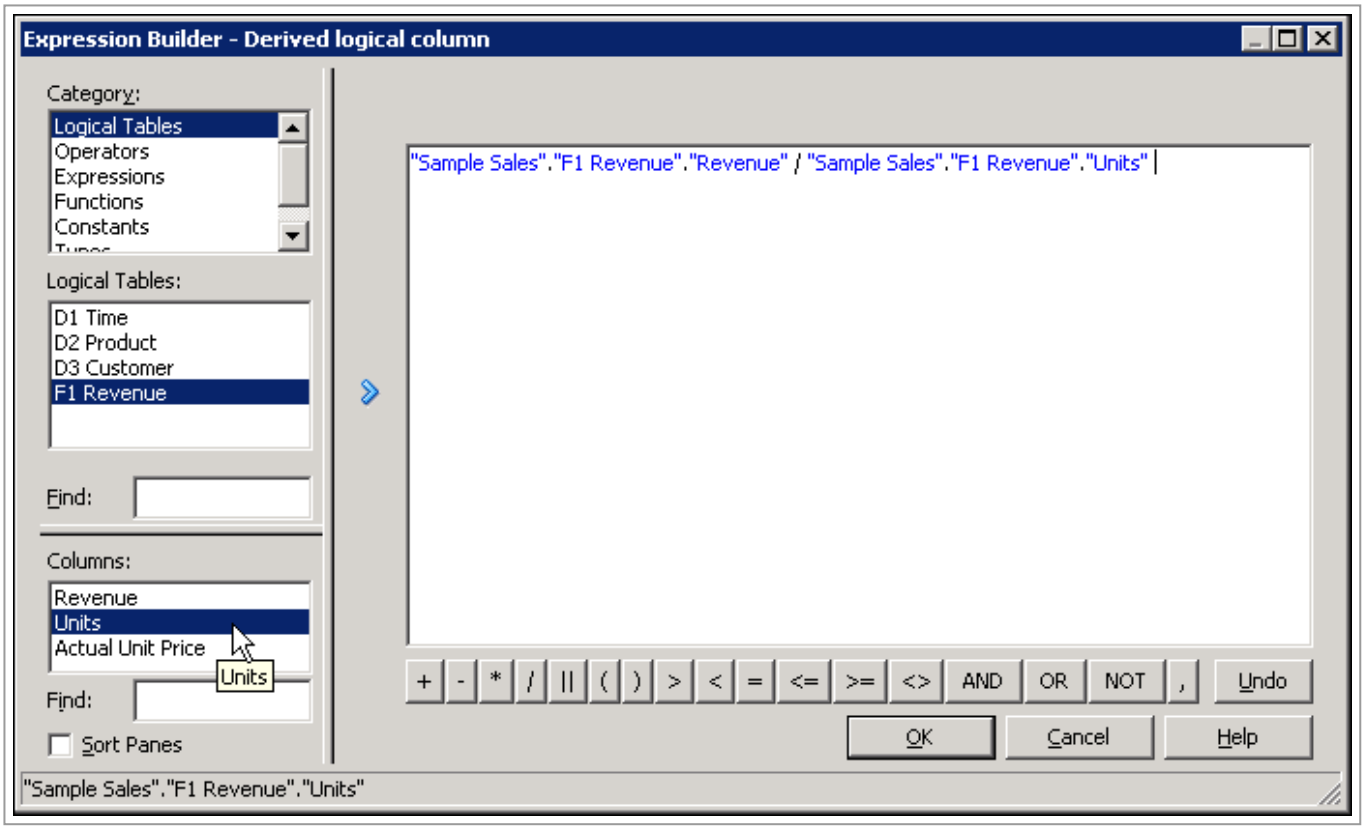
8. Click the **Insert selected item** button to move the Revenue column to the right pane.



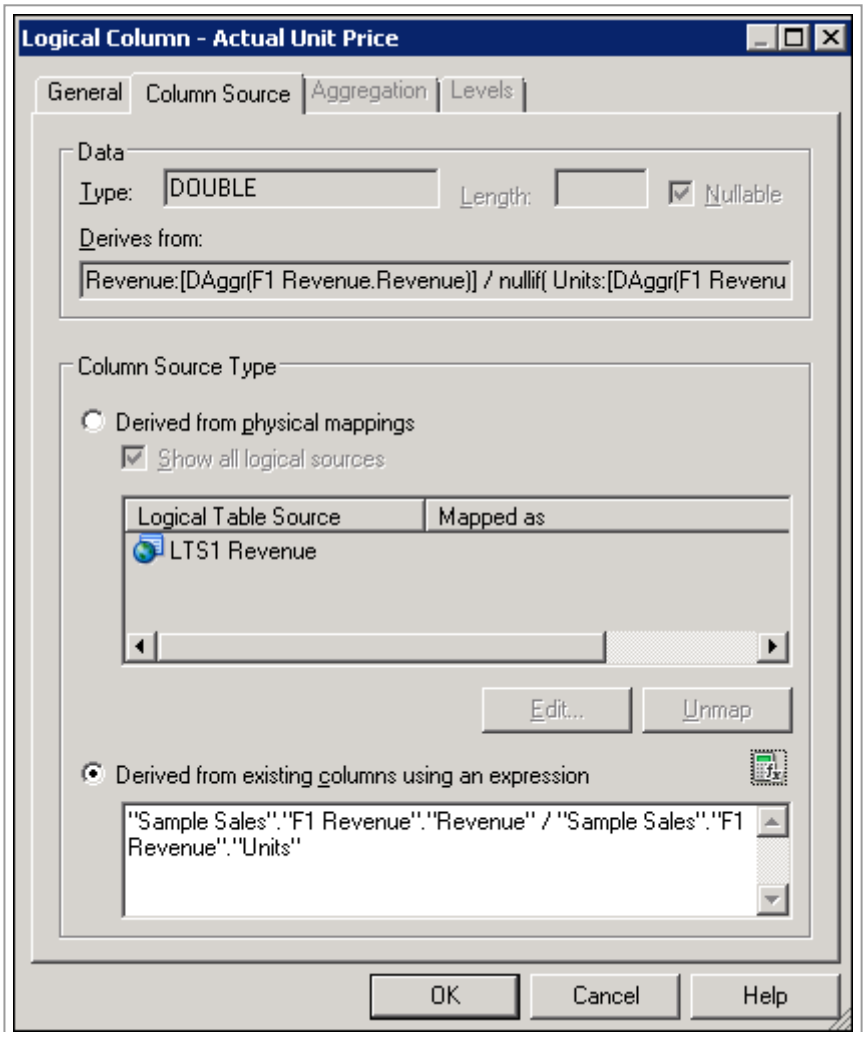

9. Click the **division operator** to add it to the expression.



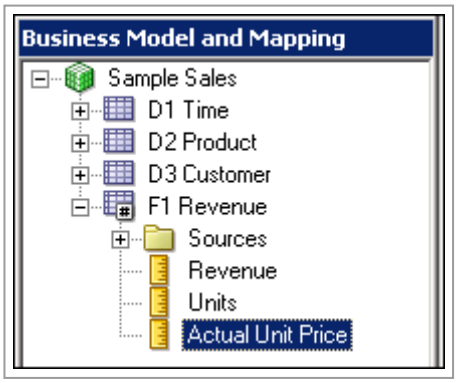

10. In the left pane select **Logical Tables > F1 Revenue** and then double-click **Units** to add it to the expression.



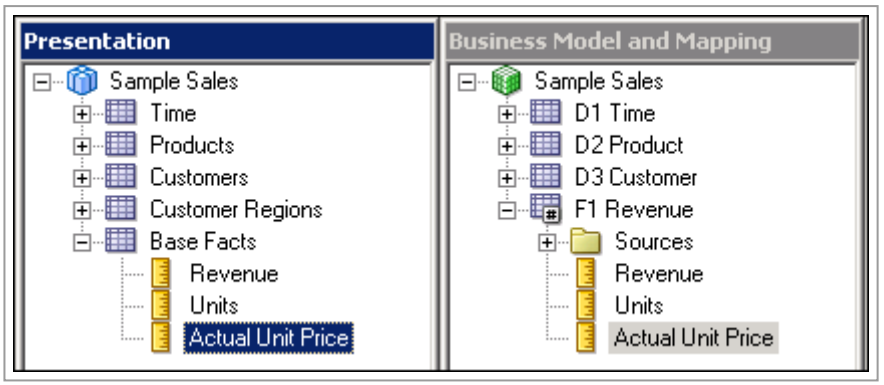

**11.** Click **OK** to close Expression Builder. Notice that the formula is added to the Logical Column dialog box.



12. Click **OK** to close the Logical Column dialog box. The **Actual Unit Price** calculated measure is added to the business model.



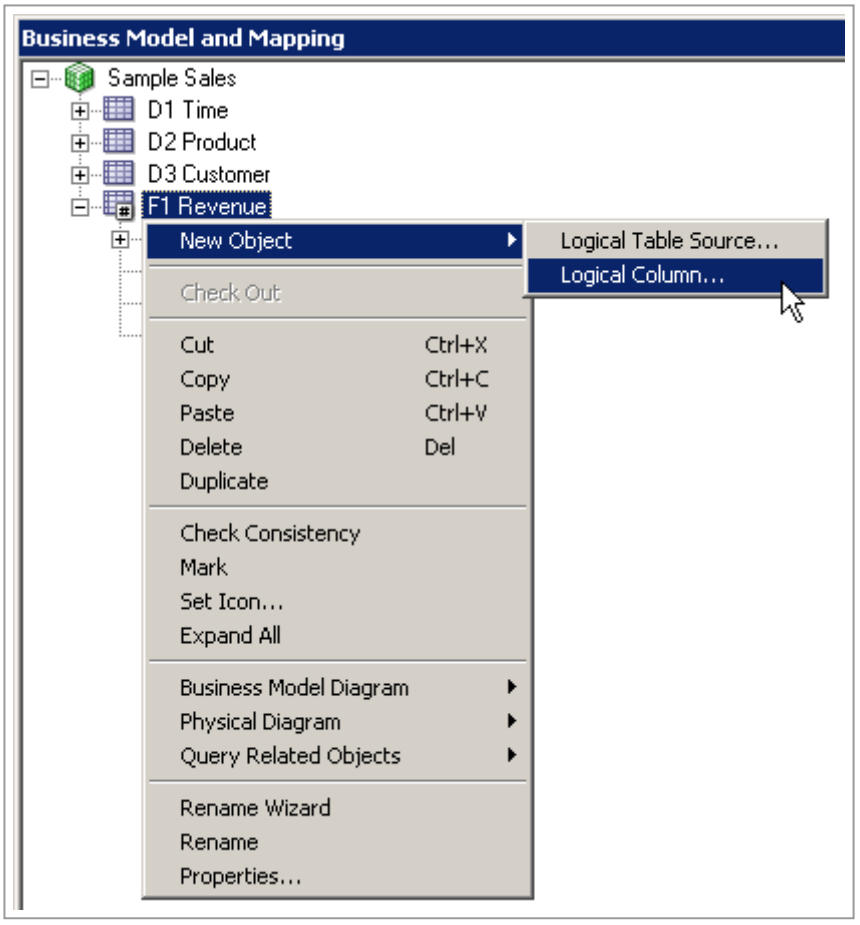

13. Drag **Actual Unit Price** from the **BMM** layer to the **Base Facts** presentation table in the **Presentation** layer.



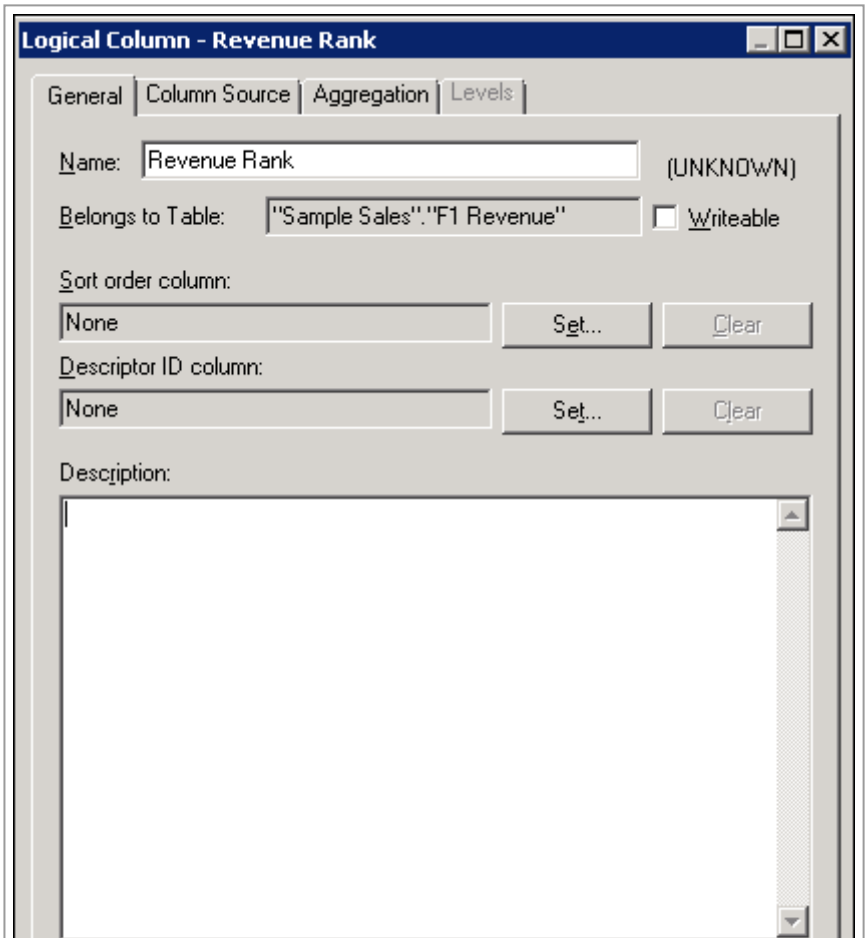

14. Save the repository and check consistency. Fix any errors or warnings before proceeding.

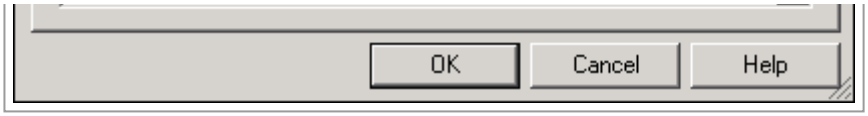
## Creating a Calculation Measure Using a Function

1. In the **BMM** layer, right-click **F1 Revenue** and select **New Object > Logical Column** to open the Logical Column dialog box.

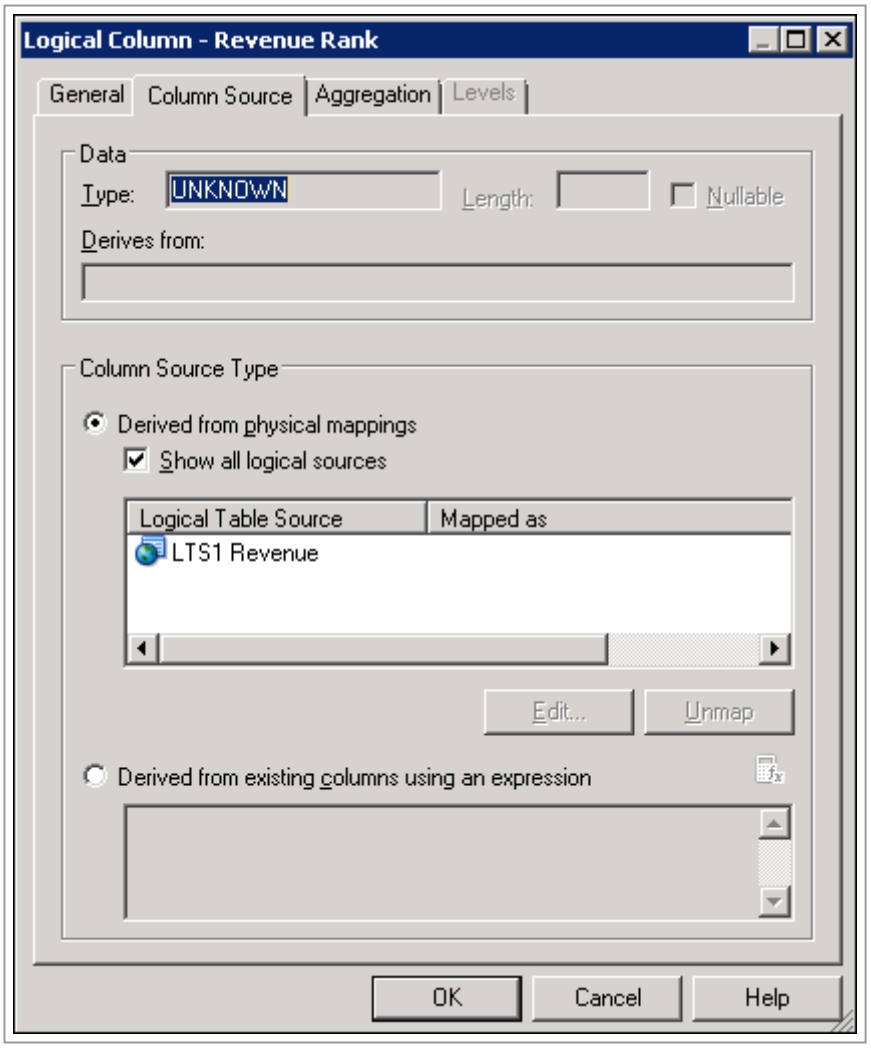



**2.** On the General tab, enter **Revenue Rank** in the Name field.

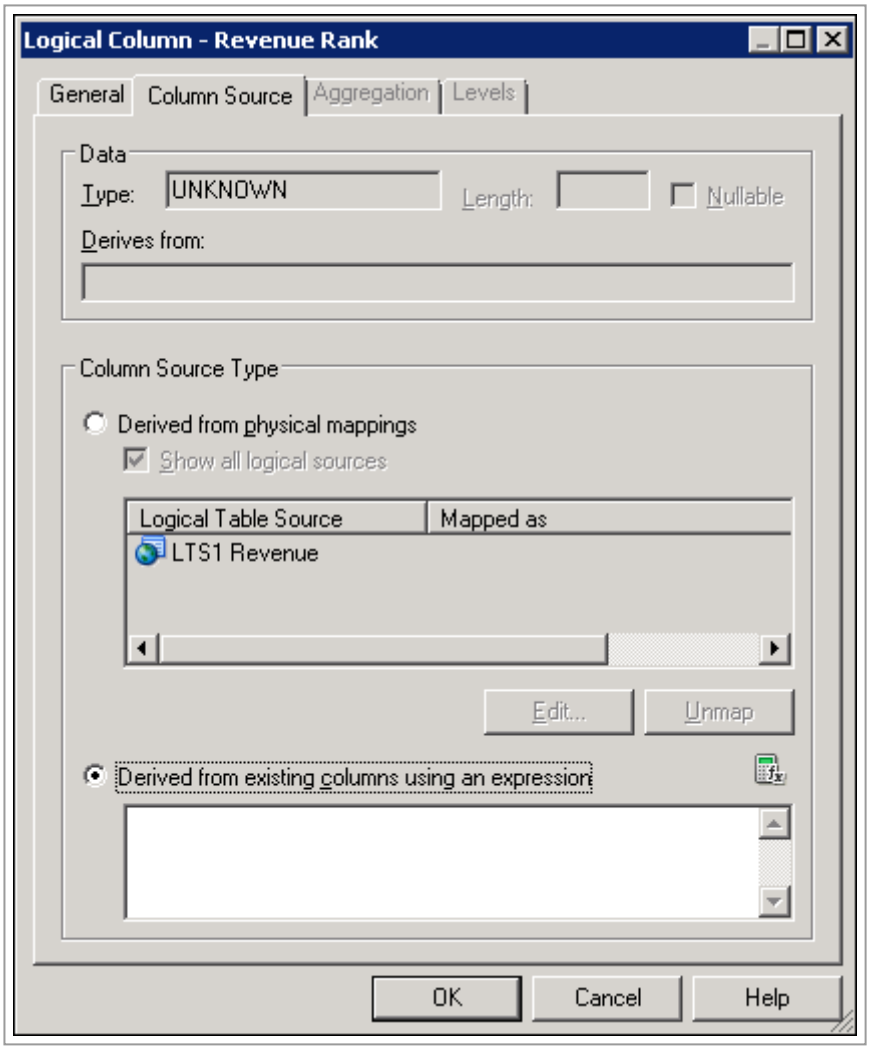




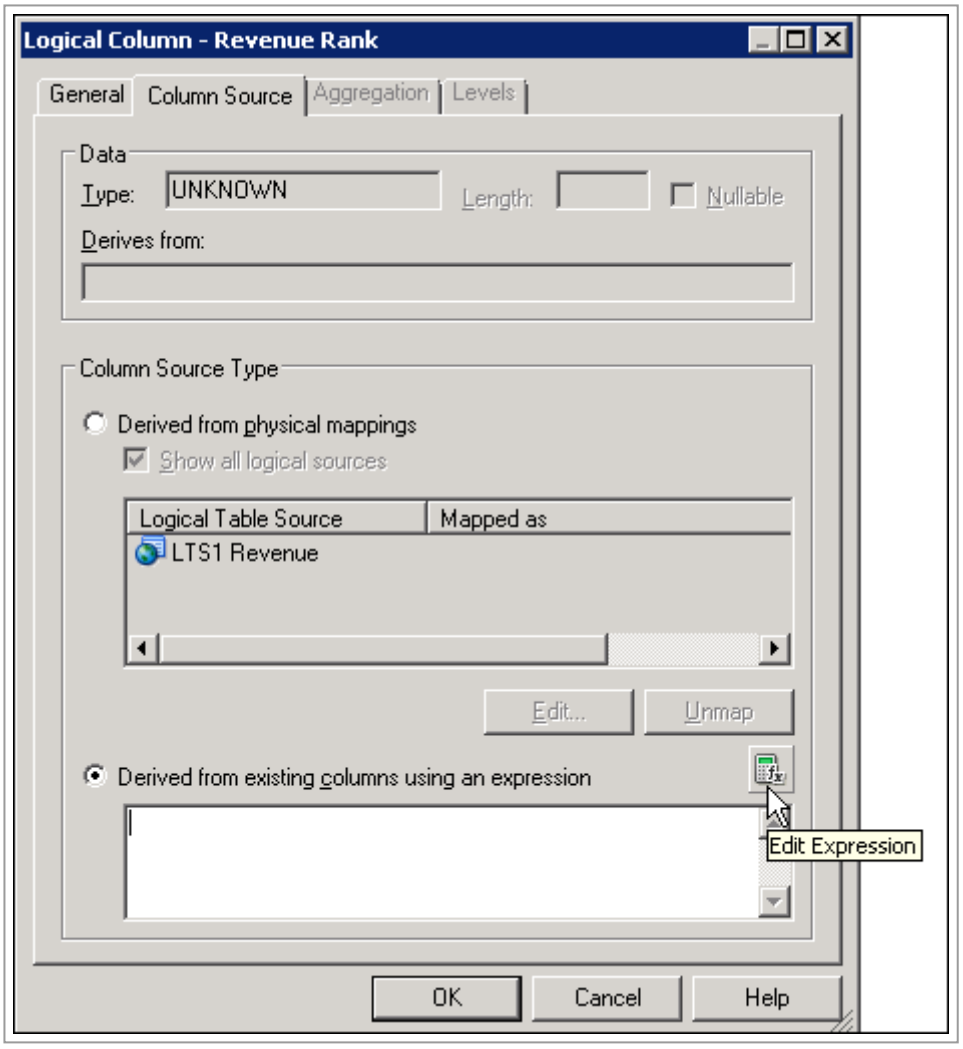

3. Click the **Column Source** tab.



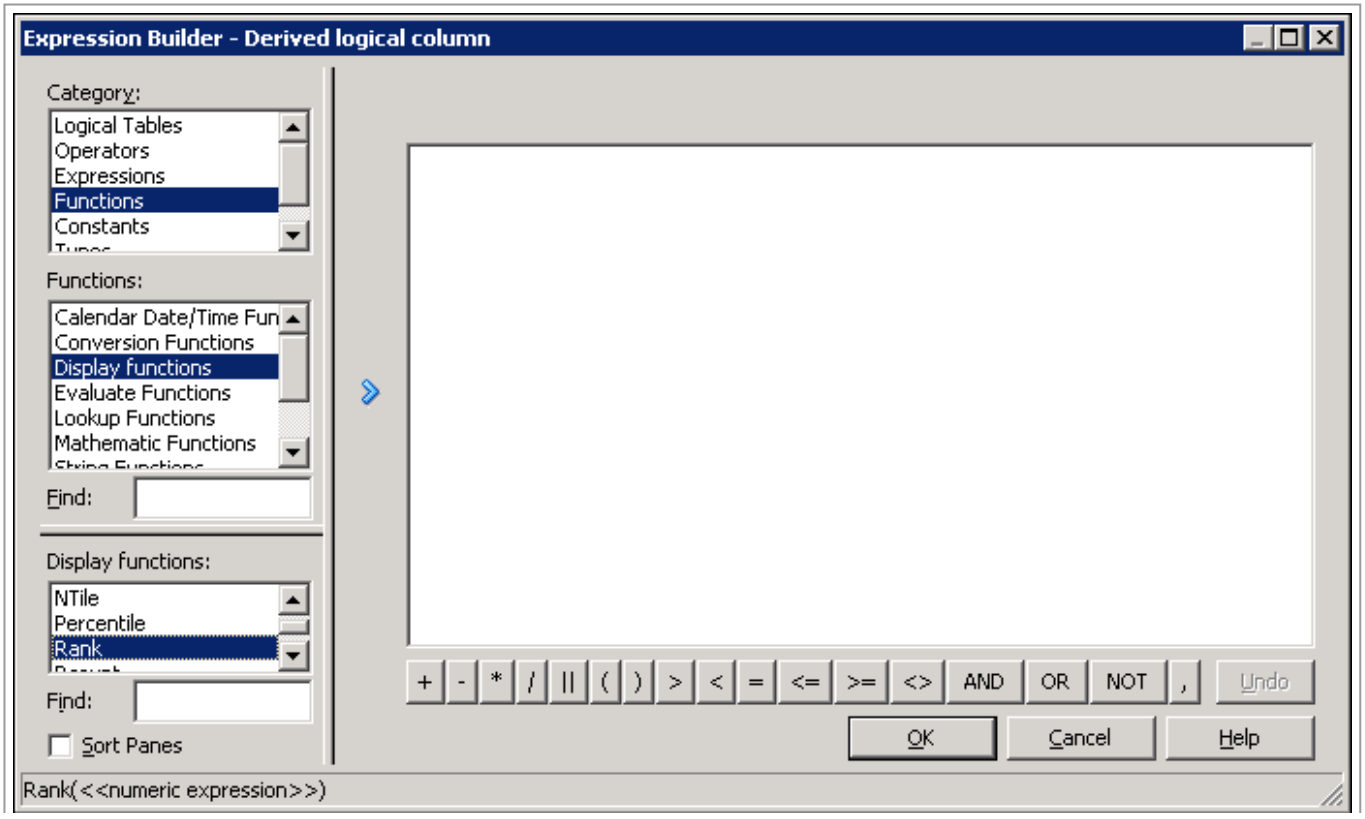

4. Select **Derived from existing columns using an expression**.



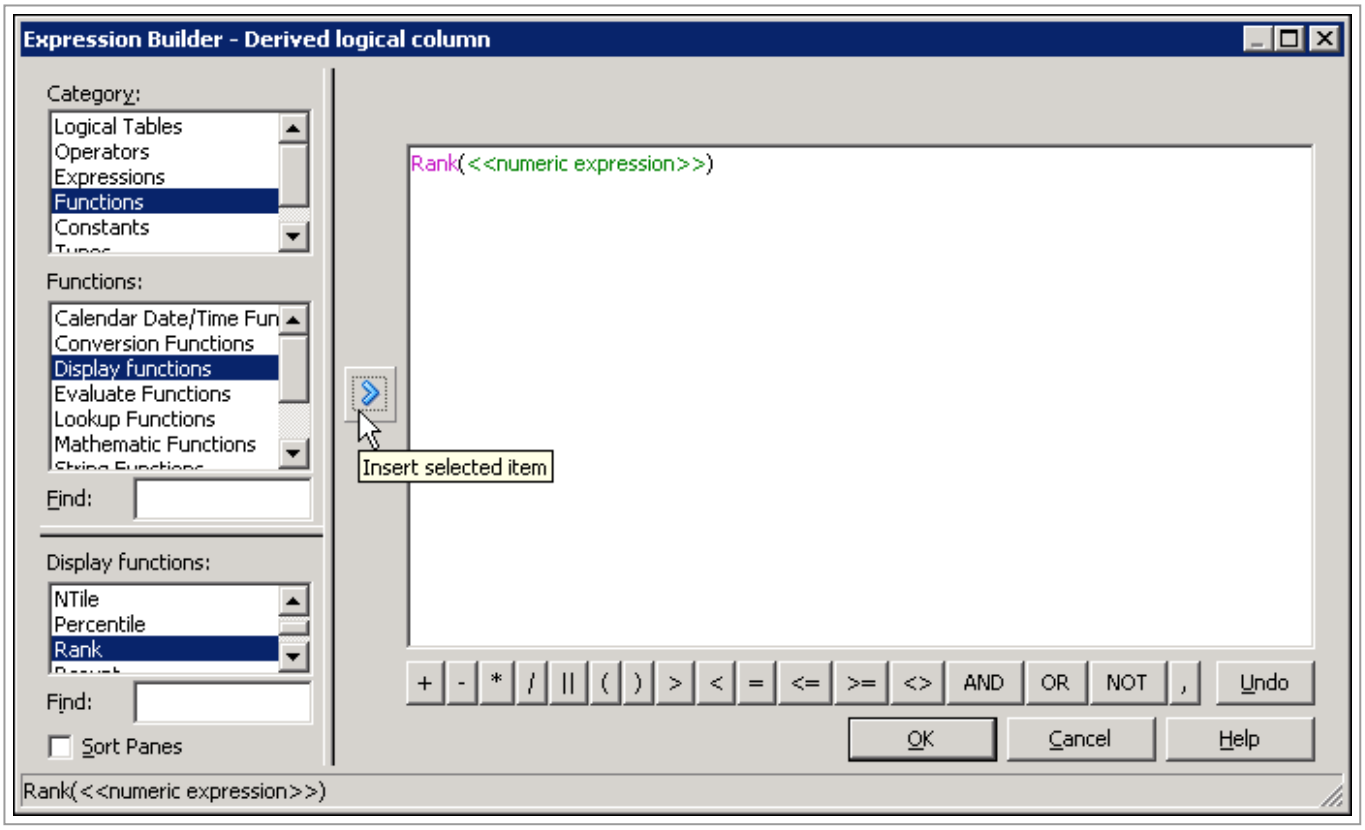

**5.** Click the **Edit Expression** button to open Expression Builder.



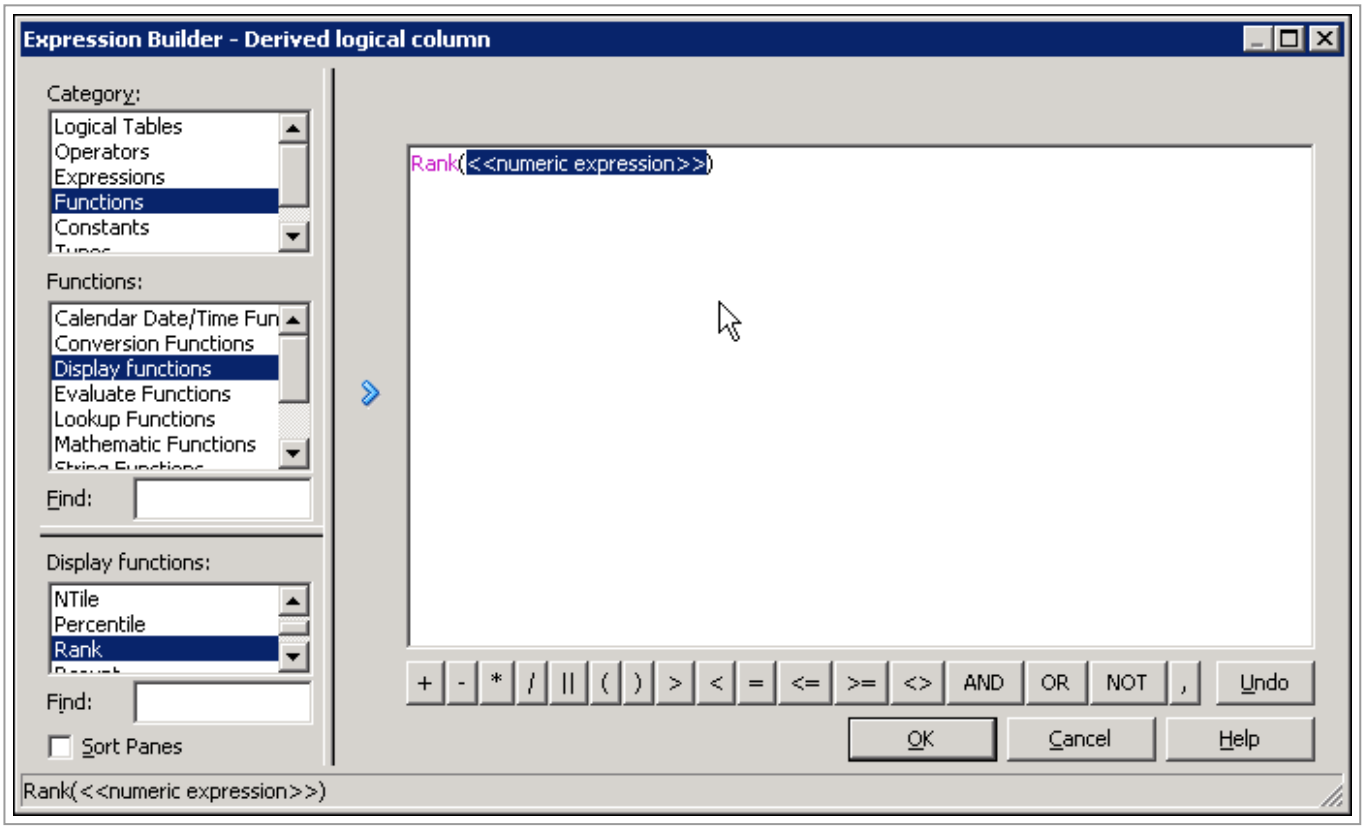

**6.** In the left pane select **Functions > Display functions > Rank.**



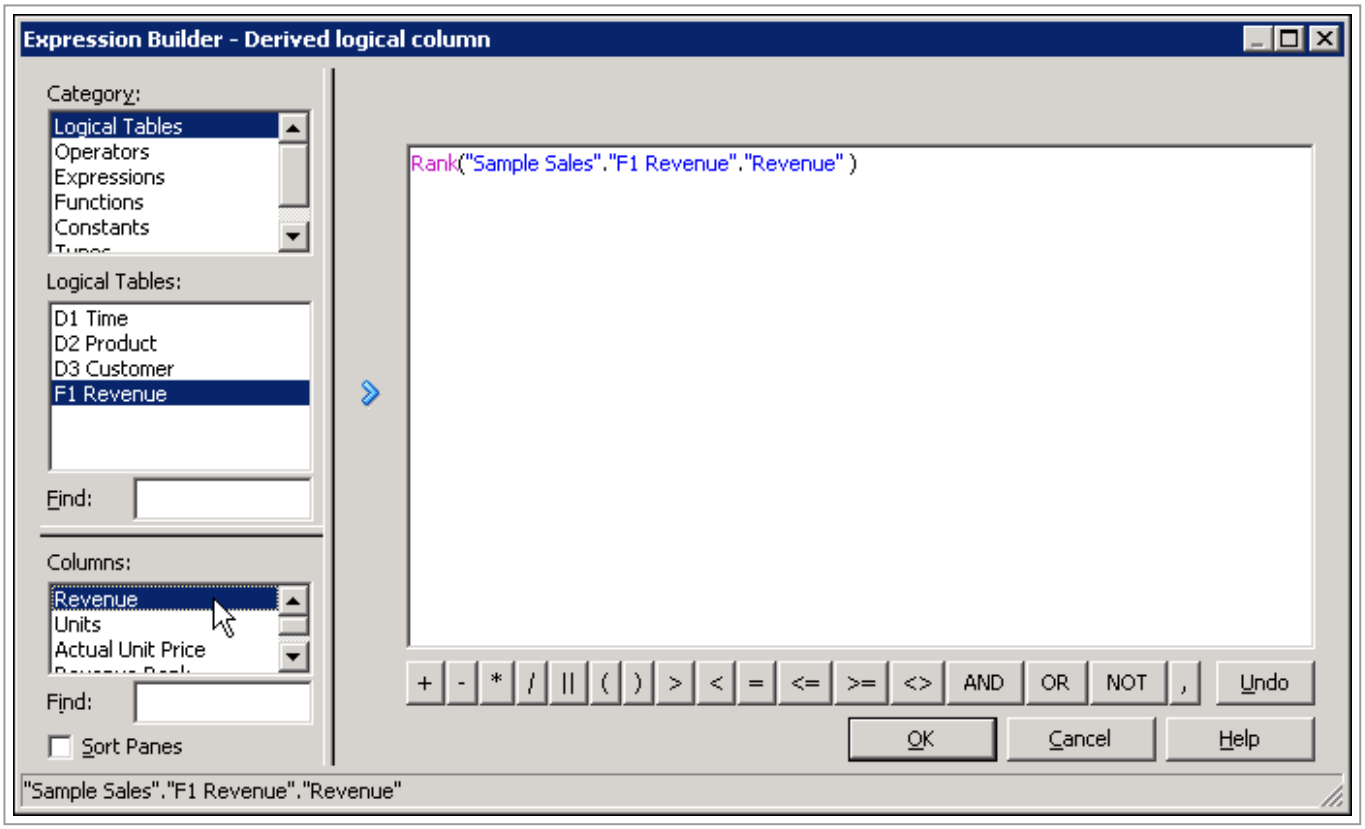
7. Click the **Insert selected item** button to move the Rank function to the right pane.



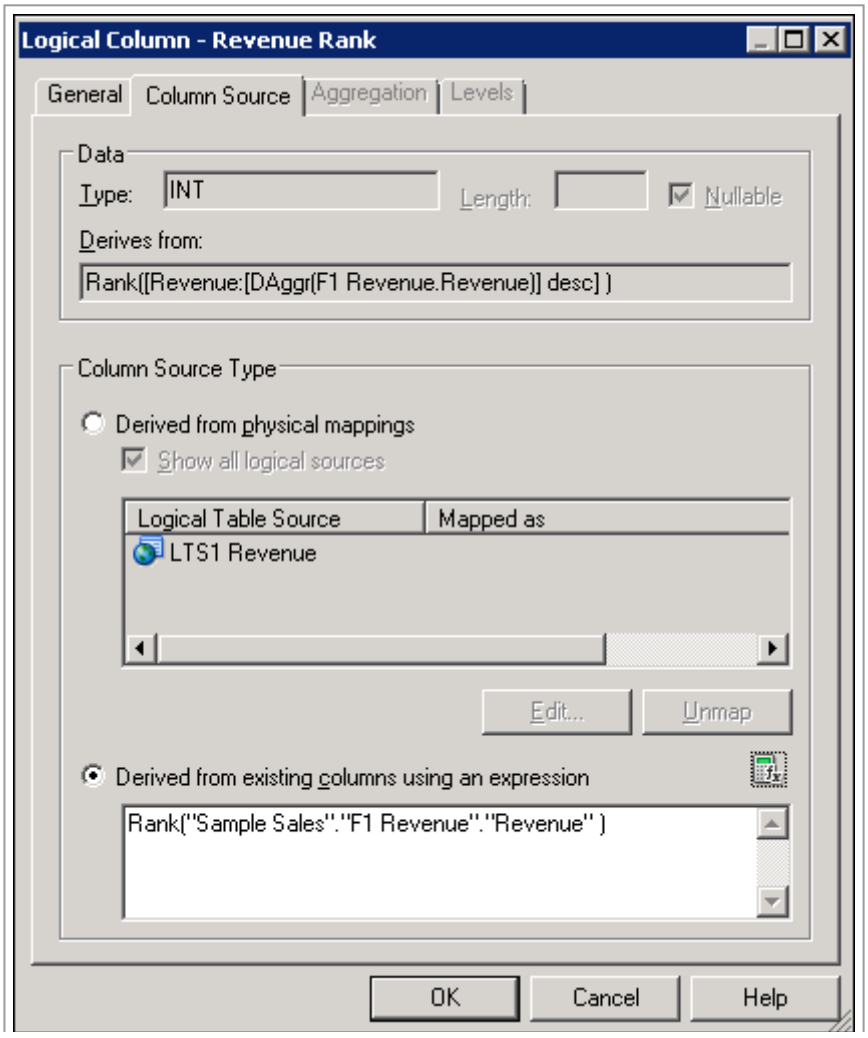

8. Click **<<numeric expression>>** in the expression.




9. In the left pane select **Logical Tables > F1 Revenue** and then double-click **Revenue** to add it to the expression.

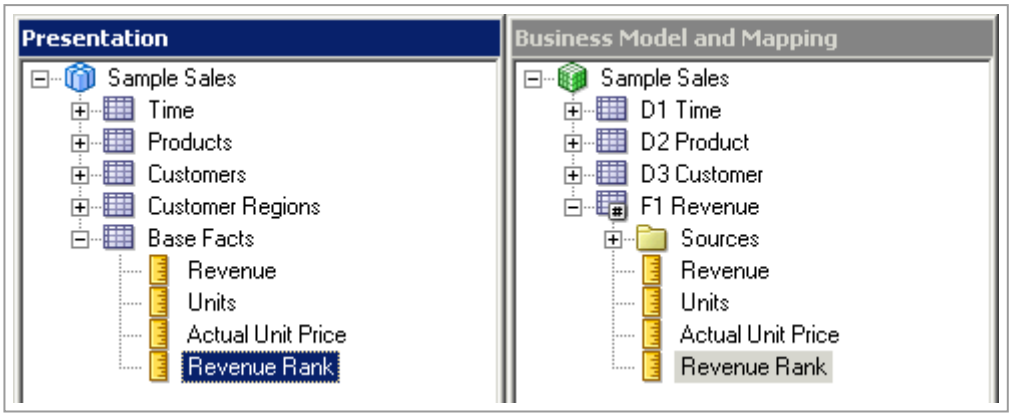



**10.** Click **OK** to close Expression Builder. Notice that the formula is added to the Logical Column dialog box.

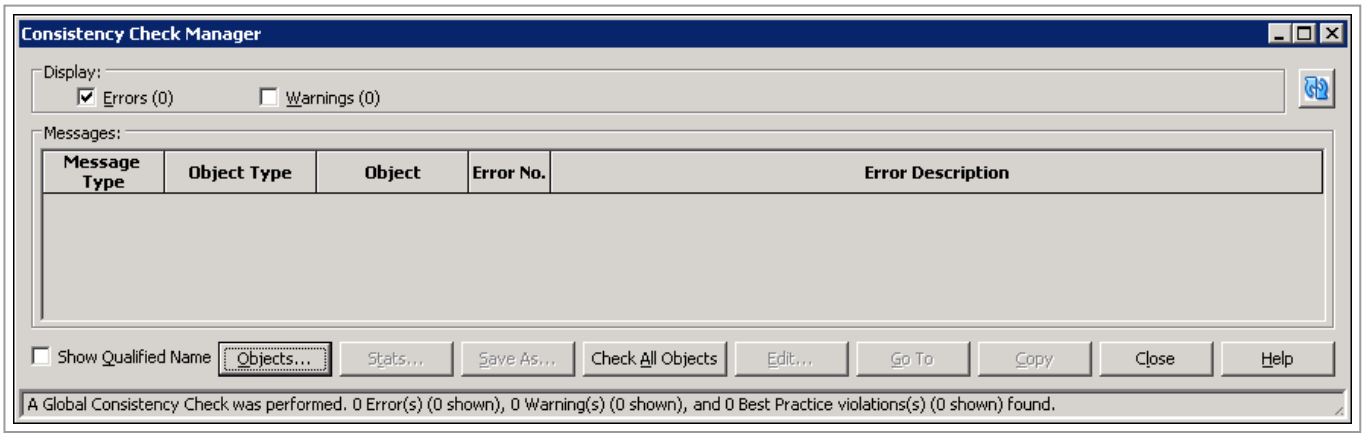


11. Click **OK** to close the Logical Column dialog box. The Revenue Rank calculated measure is added to the business model.


12. Drag **Revenue Rank** from the **BMM** layer to the **Base Facts** presentation table in the **Presentation** layer.




13. Save the repository and check consistency. Fix any errors or warnings before proceeding.




14. Close the repository and keep the Administration Tool open.

## Loading the Repository

1. Repeat the steps described in the Testing and Validating Repostory, Loading the Repository section above to upload the saved repository to the BI Server.

above to upload the saved repository to the BI Server.

## Creating and Running an Analysis

1. Return to Oracle BI, which should still be open. If not, open a browser or browser tab and enter the following URL to navigate to Oracle Business Intelligence:

   **http://<machine name>/:9502/analytics**

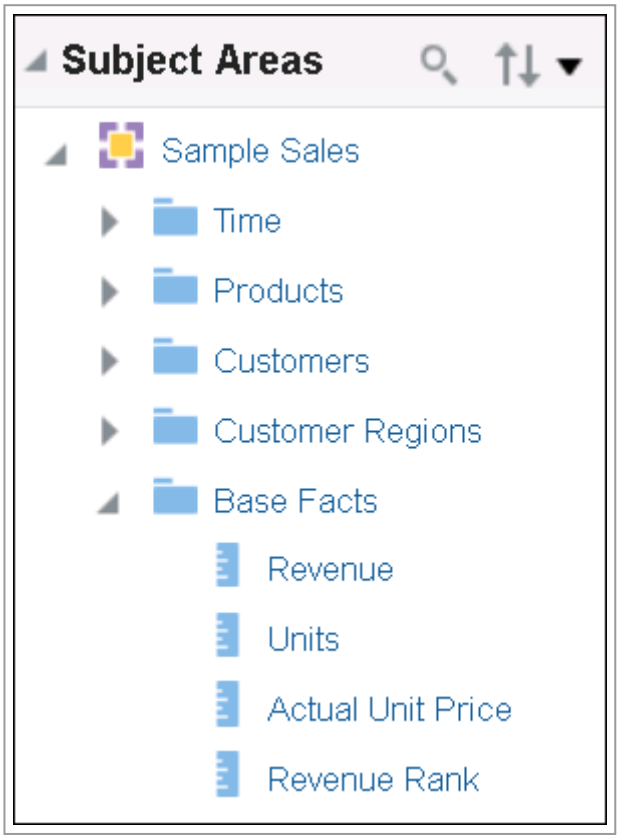
   In this tutorial the URL is http://localhost:9502/analytics.

2. If necessary, log in as an administrative user. Typically you use the administrative user name and password provided during the Oracle BI installation. In this example the user name is weblogic.

3. In the left navigation pane, under Create... Analysis and Interactive Reporting, select **Analysis**.

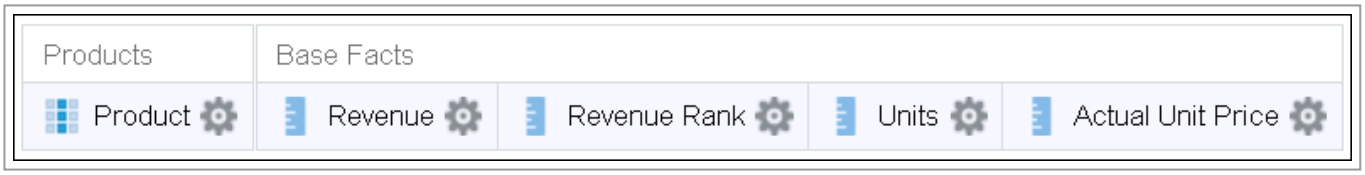
4. Select the **Sample Sales** subject area.

5. In the left navigation pane, expand the **Base Facts** folder and confirm that the **Actual Unit Price** and **Revenue Rank** columns are visible.

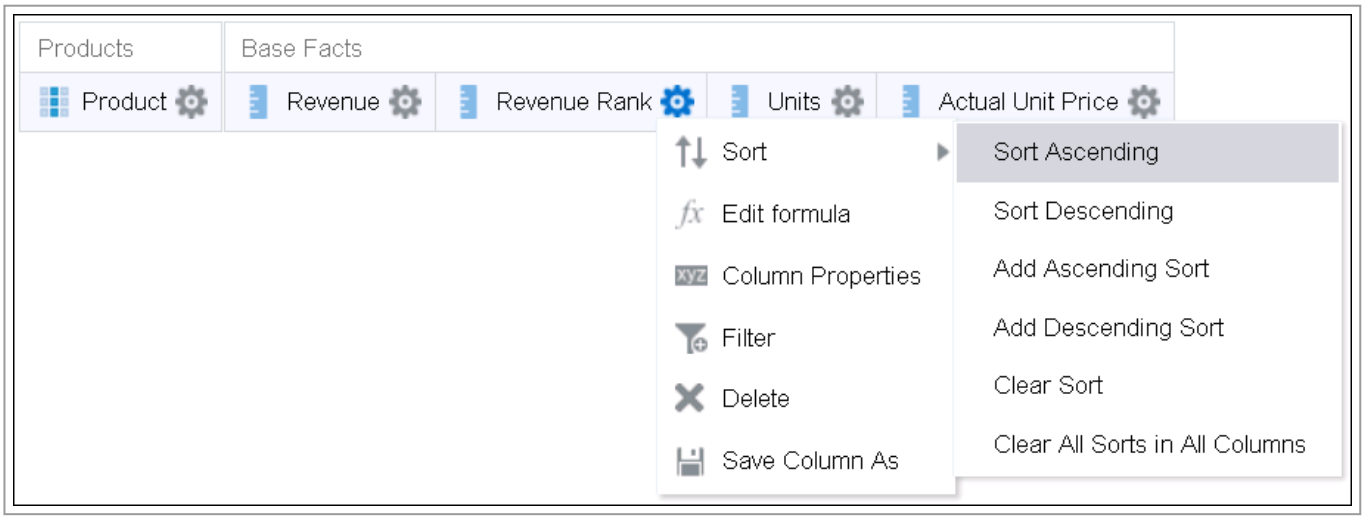



6. Create the following analysis by double-clicking column names in the Subject Areas pane:

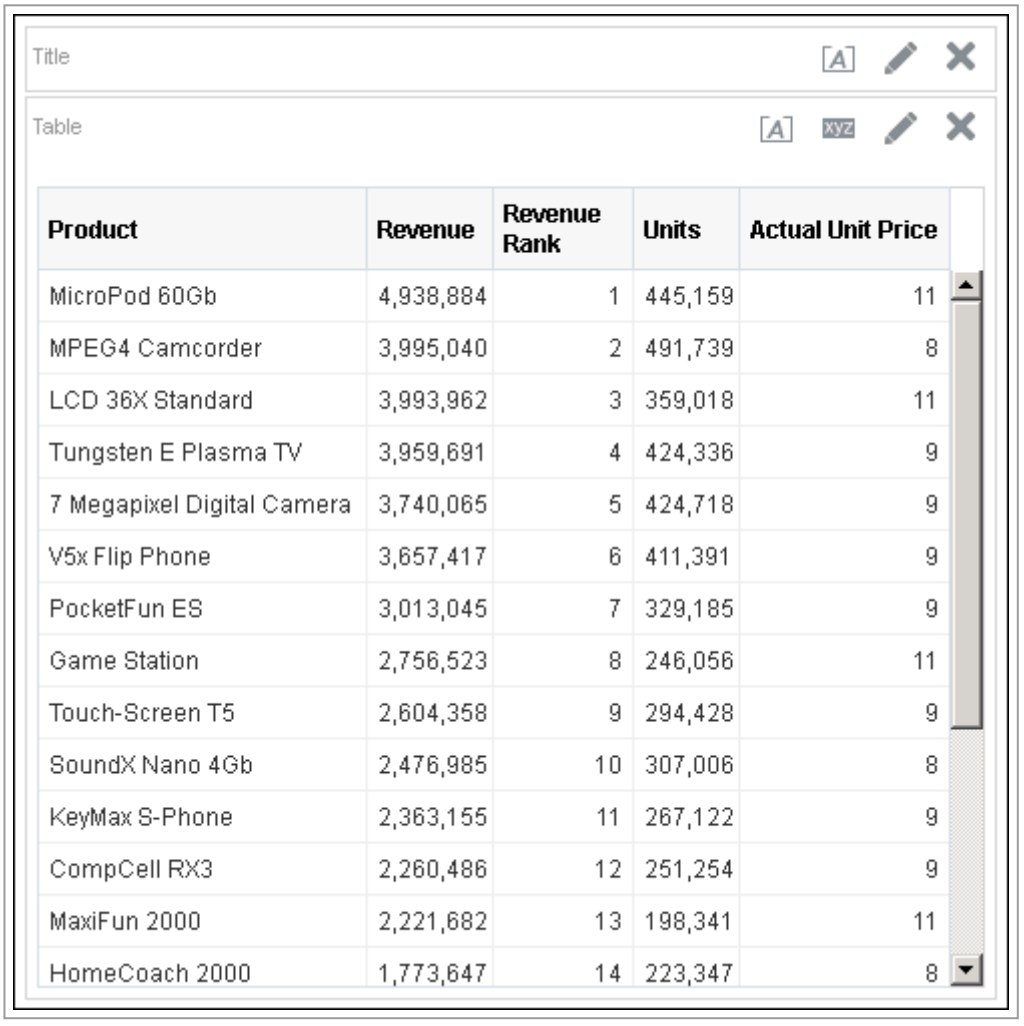
   **Products.Product**
   **Base Facts.Revenue**
   **Base Facts.Revenue Rank**
   **Base Facts.Units**
   **Base Facts.Actual Unit Price**




**7.** Sort **Revenue Rank** in ascending order.




**8.** Click **Results** to view the analysis results.




| Product | Revenue | Revenue Rank | Units | Actual Unit Price |
|---|---|---|---|---|
| MicroPod 60Gb | 4,938,884 | 1 | 445,159 | 11 |
| MPEG4 Camcorder | 3,995,040 | 2 | 491,739 | 8 |
| LCD 36X Standard | 3,993,962 | 3 | 359,018 | 11 |
| Tungsten E Plasma TV | 3,959,691 | 4 | 424,336 | 9 |
| 7 Megapixel Digital Camera | 3,740,065 | 5 | 424,718 | 9 |
| V5x Flip Phone | 3,657,417 | 6 | 411,391 | 9 |
| PocketFun ES | 3,013,045 | 7 | 329,185 | 9 |
| Game Station | 2,756,523 | 8 | 246,056 | 11 |
| Touch-Screen T5 | 2,604,358 | 9 | 294,428 | 9 |
| SoundX Nano 4Gb | 2,476,985 | 10 | 307,006 | 8 |
| KeyMax S-Phone | 2,363,155 | 11 | 267,122 | 9 |
| CompCell RX3 | 2,260,486 | 12 | 251,254 | 9 |
| MaxiFun 2000 | 2,221,682 | 13 | 198,341 | 11 |
| HomeCoach 2000 | 1,773,647 | 14 | 223,347 | 8 |

Please note that the Actual Unit Price calculation is correct, although it does not make sense from a business perspective. For example, the unit price for an LCD HD Television would not be 9 dollars. This is a result of the underlying sample data.

## Creating Logical Dimensions with Level-Based Hierarchies

In this set of steps you add logical dimension hierarchies to the business model. A logical dimension represents a hierarchical organization of logical columns belonging to a single logical dimension table. Logical dimensions can exist in the Business Model and Mapping layer and in the Presentation Layer. Adding logical dimensions to the Presentation layer exposes them to users, which enables users to create hierarchy-based queries. You implement four logical dimensions for ABC: Time, Product, Office, and Customer. Creating logical dimensions with hierarchies allows you to build level-based measures, define aggregation rules that vary by dimension, provide drill down on charts and tables in analyses and dashboards, and define the content of aggregate sources.To create logical dimensions with level-based hierarchies, you perform the following steps:

- Opening the Repository in Offline Mode
- Creating a Logical Dimension for Product
- Creating Logical Levels
- Associating Logical Columns with Logical Levels
- Setting Logical Level Keys
- Creating a Logical Dimension for Time
- Associating Time Logical Columns with Logical Levels
- Creating a Logical Dimension for Customer
- Setting Aggregation Content for Logical Table Sources
- Testing Your Work

### Opening the Repository in Offline Mode

1. Return to the Administration Tool, which should still be open.

2. Select **File > Open > Offline**.

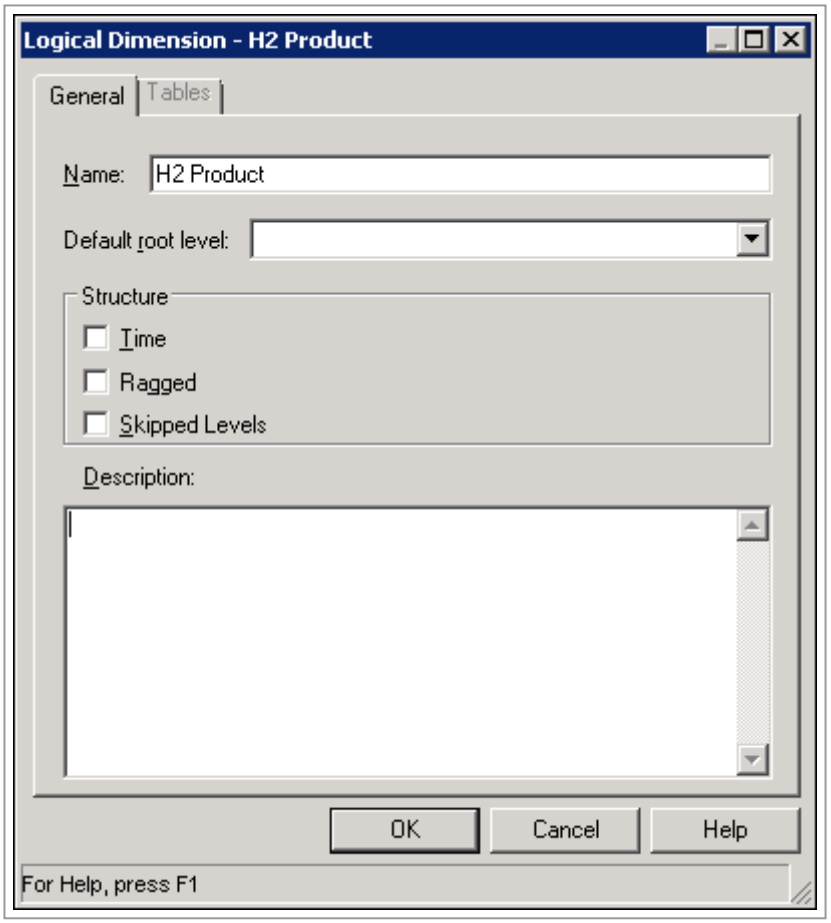
3. Select **BISAMPLE.rpd** and click **Open**.

4. Enter **Admin123** as the repository password and click **OK** to open the repository.

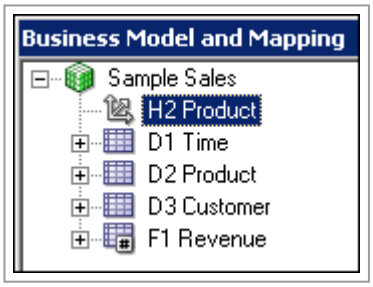
### Creating a Logical Dimension for Product

1. In the **BMM** layer, right-click the **Sample Sales** business model and select **New Object > Logical Dimension > Dimension with Level-Based Hierarchy** to open the Logical Dimension dialog box.

2. Name the logical dimension **H2 Product.**

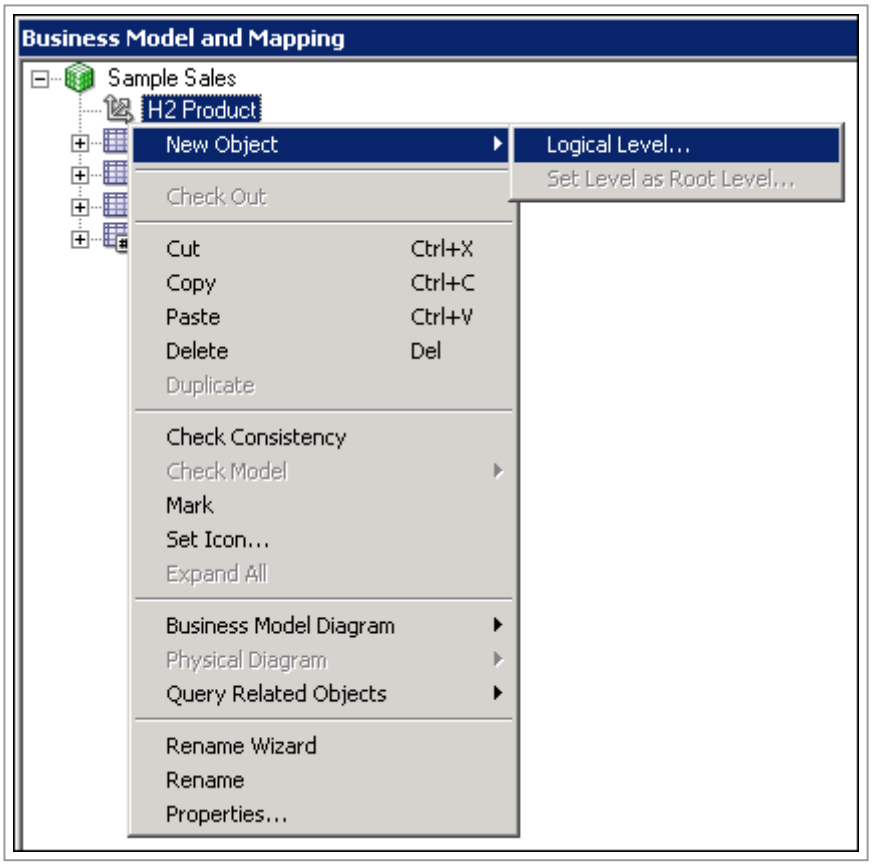



3. Click **OK**. The logical dimension is added to the Sample Sales business model.

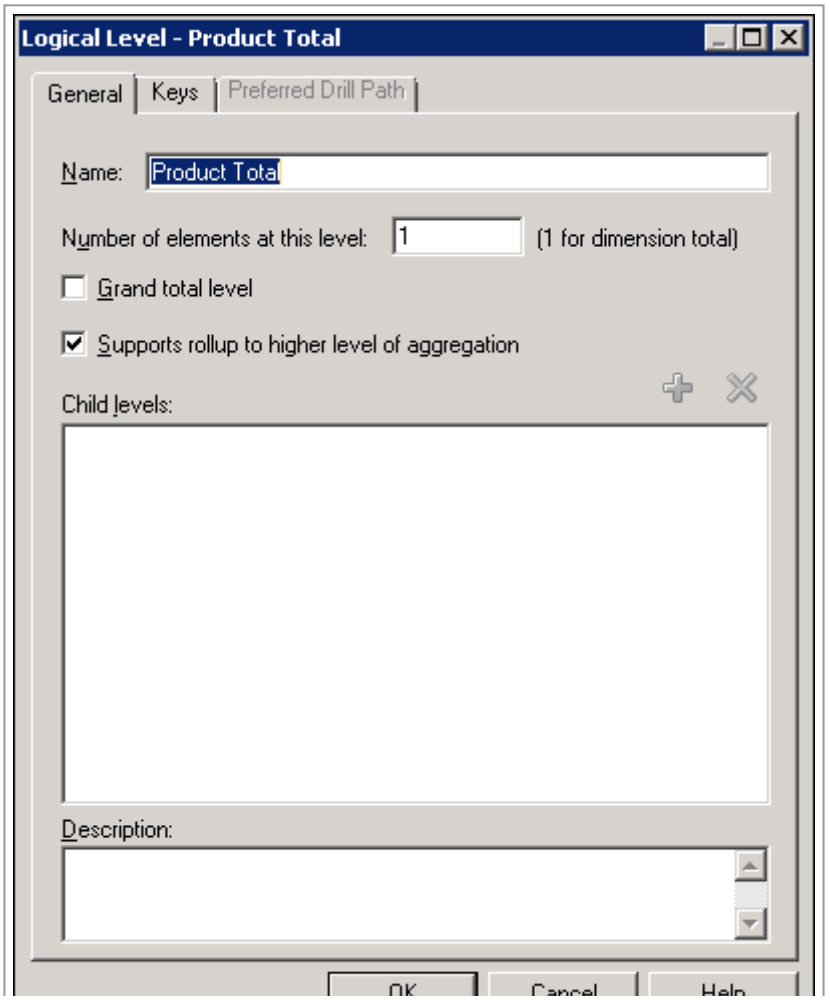


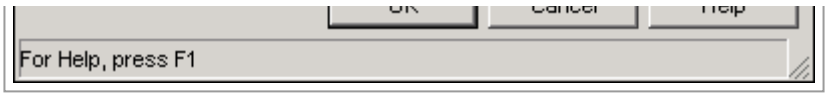

## Creating Logical Levels

1. Right-click **H2 Product** and select **New Object > Logical Level**.

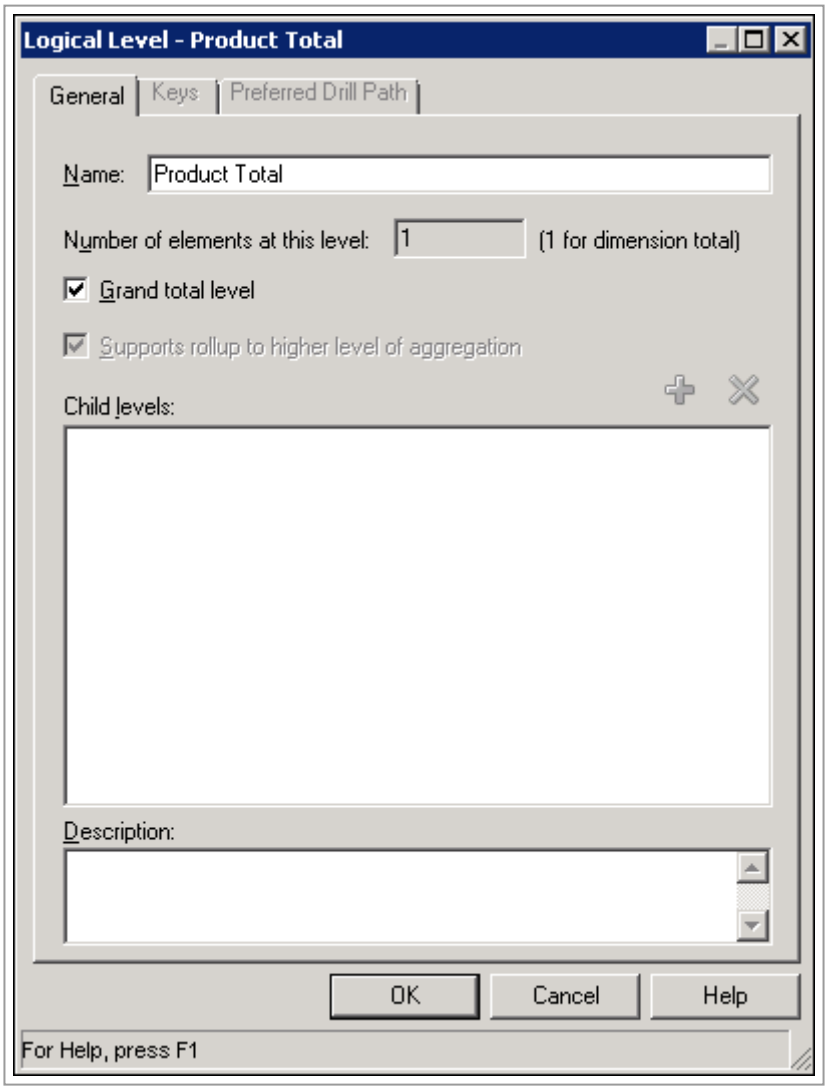



**2.** Name the logical level as **Product Total.**

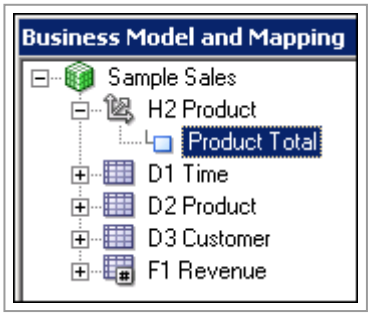




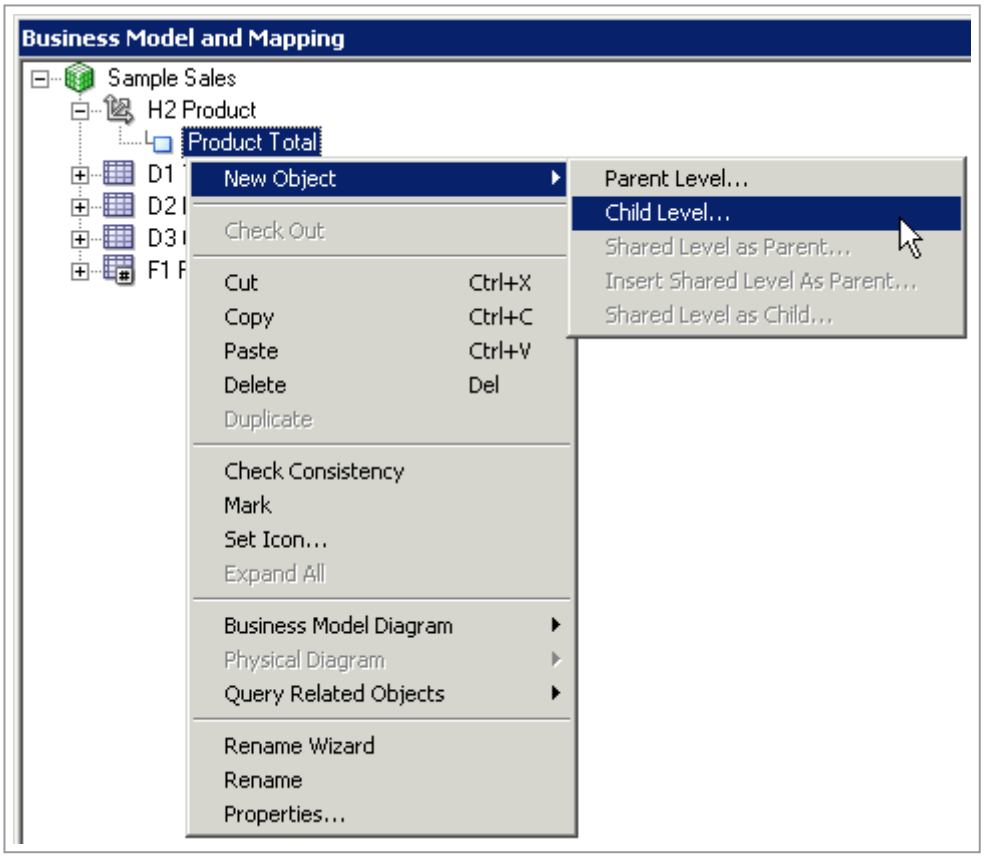

3. Because this level represents the grand total for products, select the **Grand total level** check box. Note that when you do this, the **Supports rollup to higher level of aggregation** field is grayed out and protected.



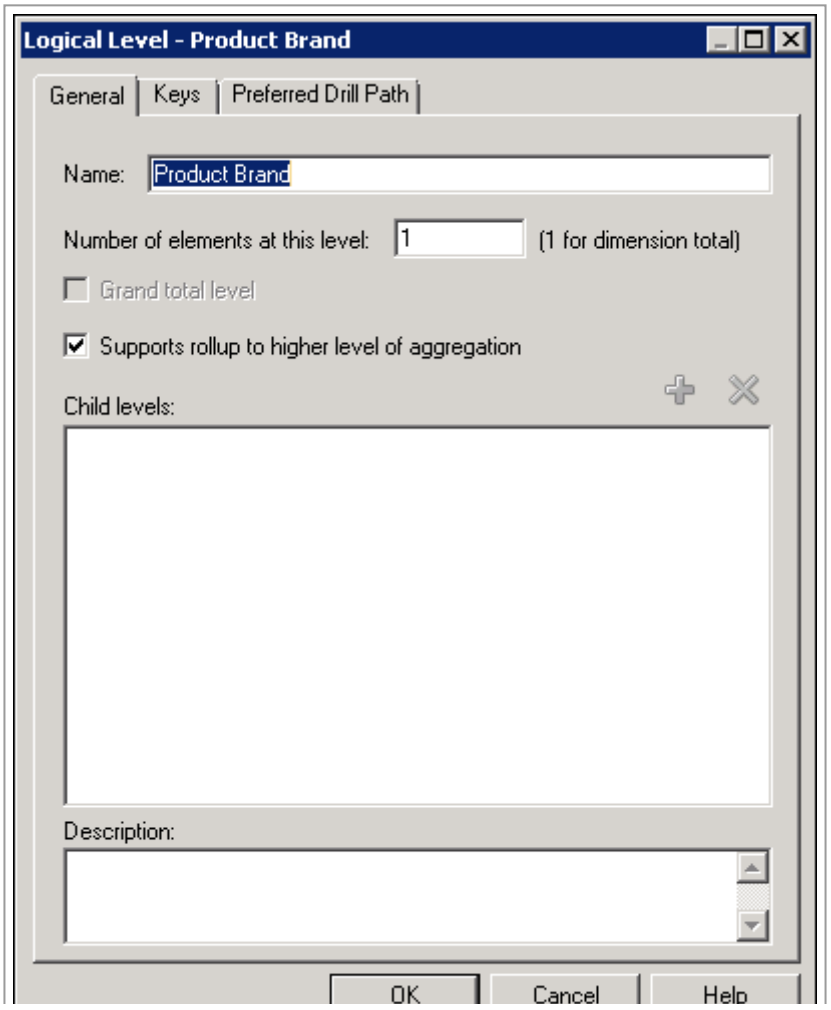

4. Click **OK** to close the Logical Level dialog box. The **Product Total** level is added to the H2 Product logical dimension.



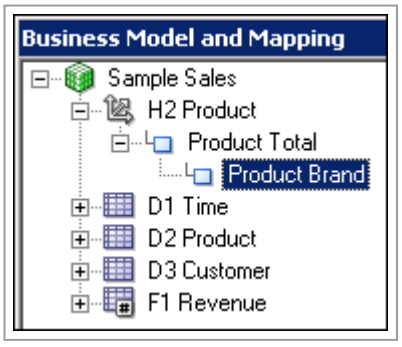

5. Right-click **Product Total** and select **New Object > Child Level** to open the Logical Level dialog box.




**6.** Name the logical level **Product Brand**.



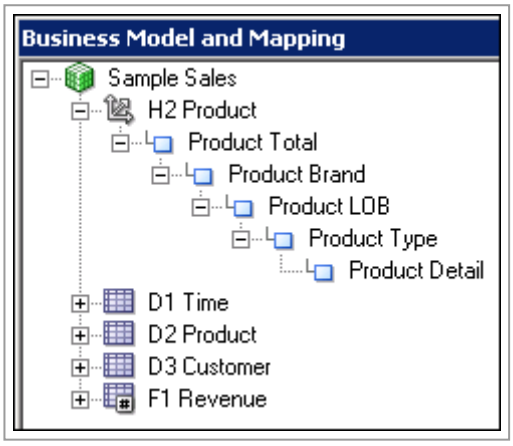

7. Click **OK** to close the Logical Level dialog box. The **Product Brand** level is added to the logical dimension.




8. Repeat the steps to add the following child levels:

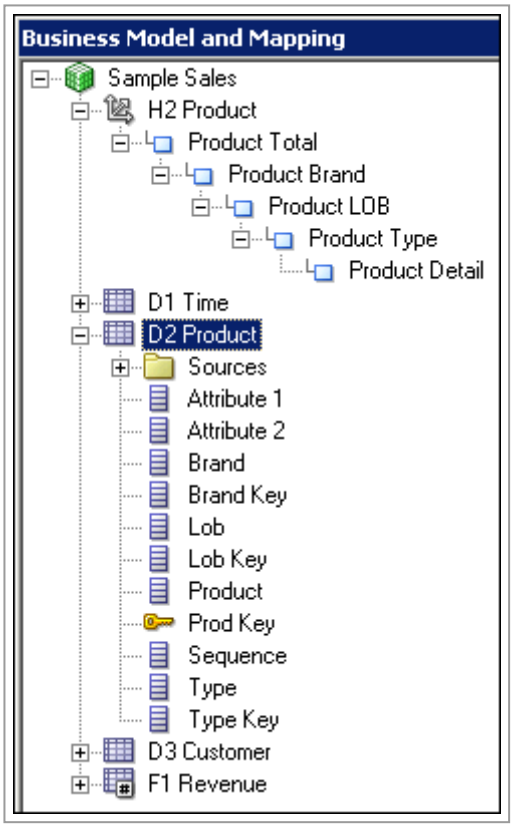
   **Product LOB** as a child of Product Brand
   **Product Type** as a child of Product LOB
   **Product Detail** as a child of Product Type

   Use the screenshot as a guide:

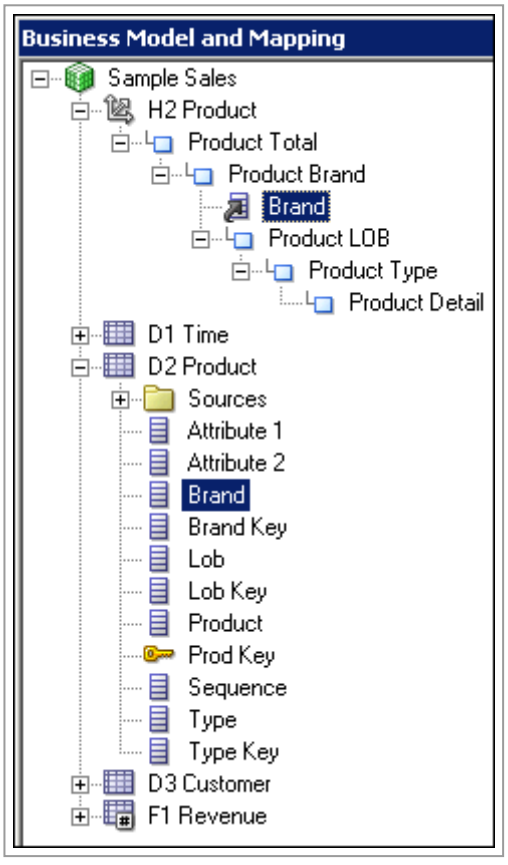



## Associating Logical Columns with Logical Levels

1. Expand the **D2 Product** logical table.




2. Drag the **Brand** column from D2 Product to the **Product Brand** level in H2 Product.



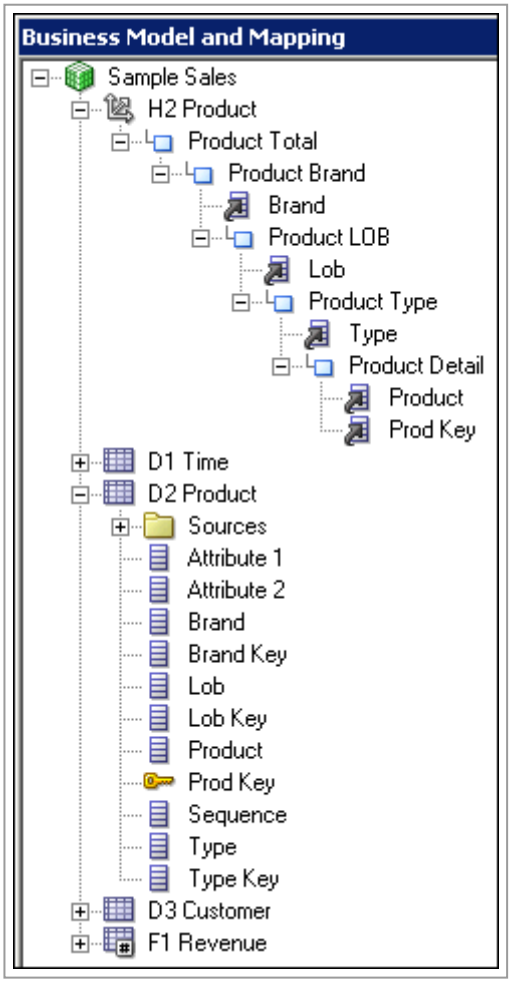

3. Continue dragging logical columns from the **D2 Product** logical table to their corresponding levels in the **H2 Product** logical dimension:

| Logical Column | Logical Level |
|---|---|

| | |
|---|---|
| Lob | Product LOB |

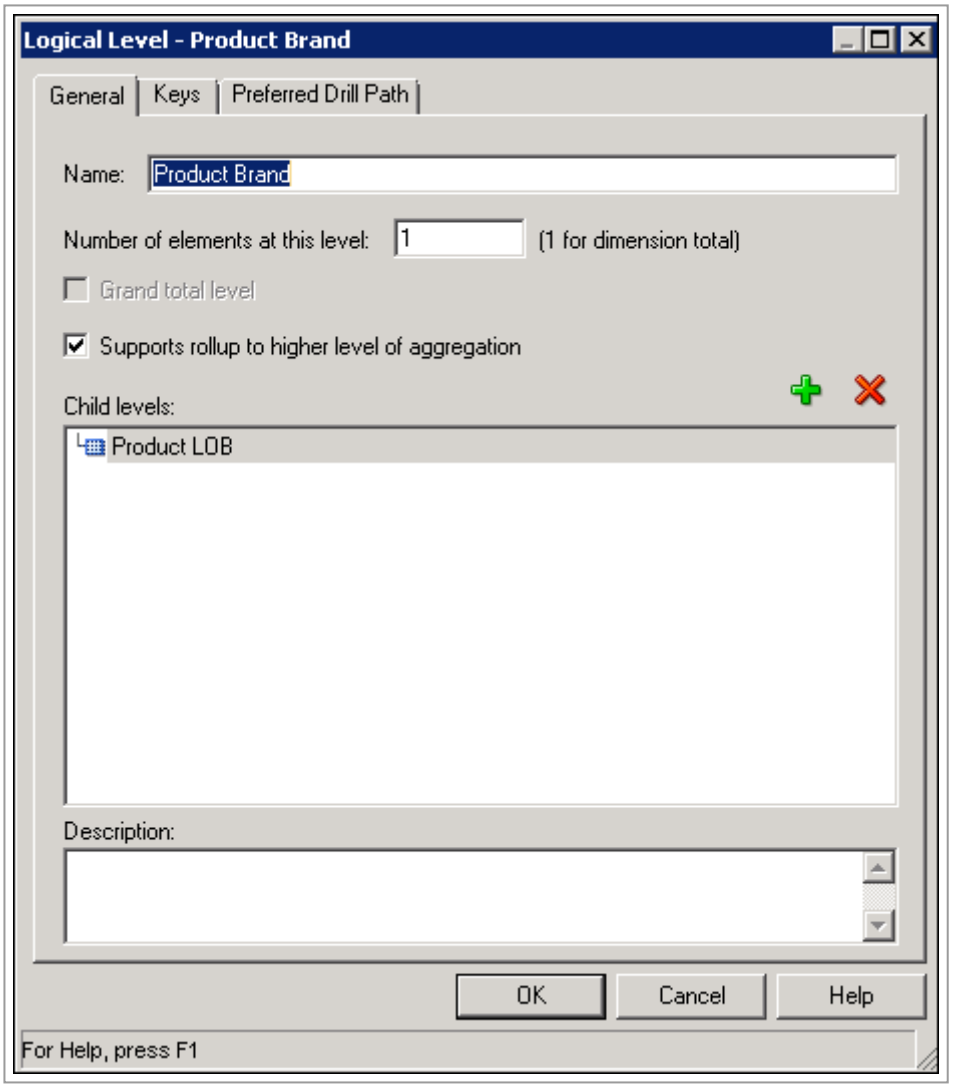
| | |
|---|---|
| Type | Product Type |
| Product | Product Detail |
| Prod Key | Product Detail |

Your results should look similar to the screenshot:

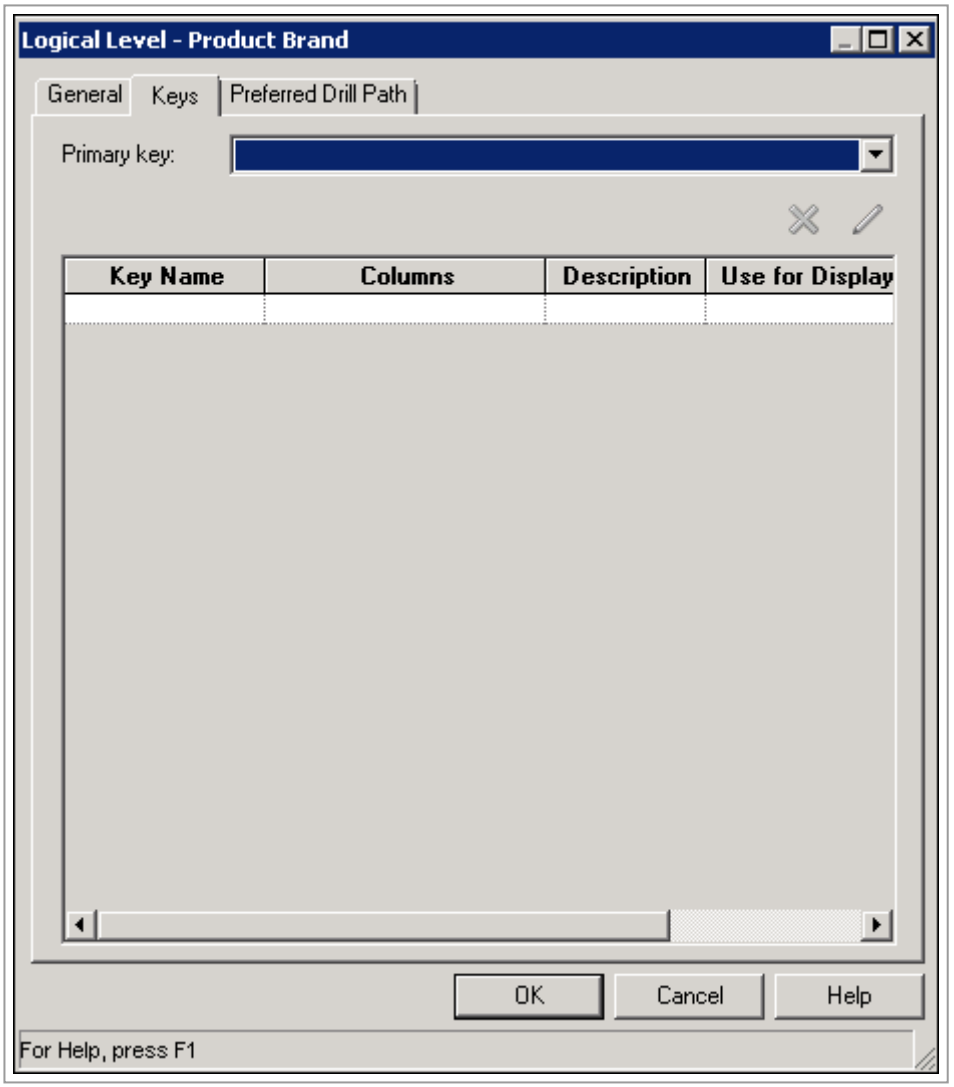



## Setting Logical Level Keys

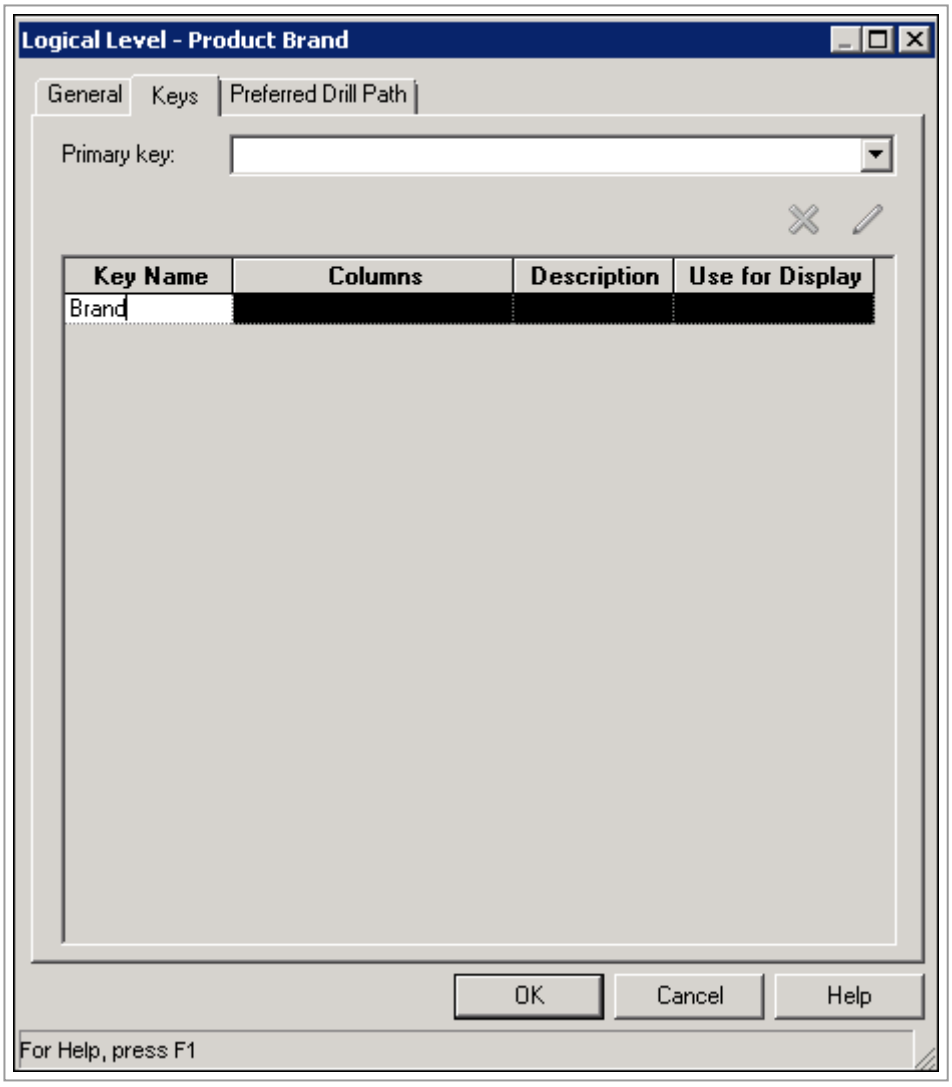
1. Double-click the **Product Brand** logical level to open the **Logical Level** dialog box. On the **General** tab, notice that the **Product LOB** child level is displayed.



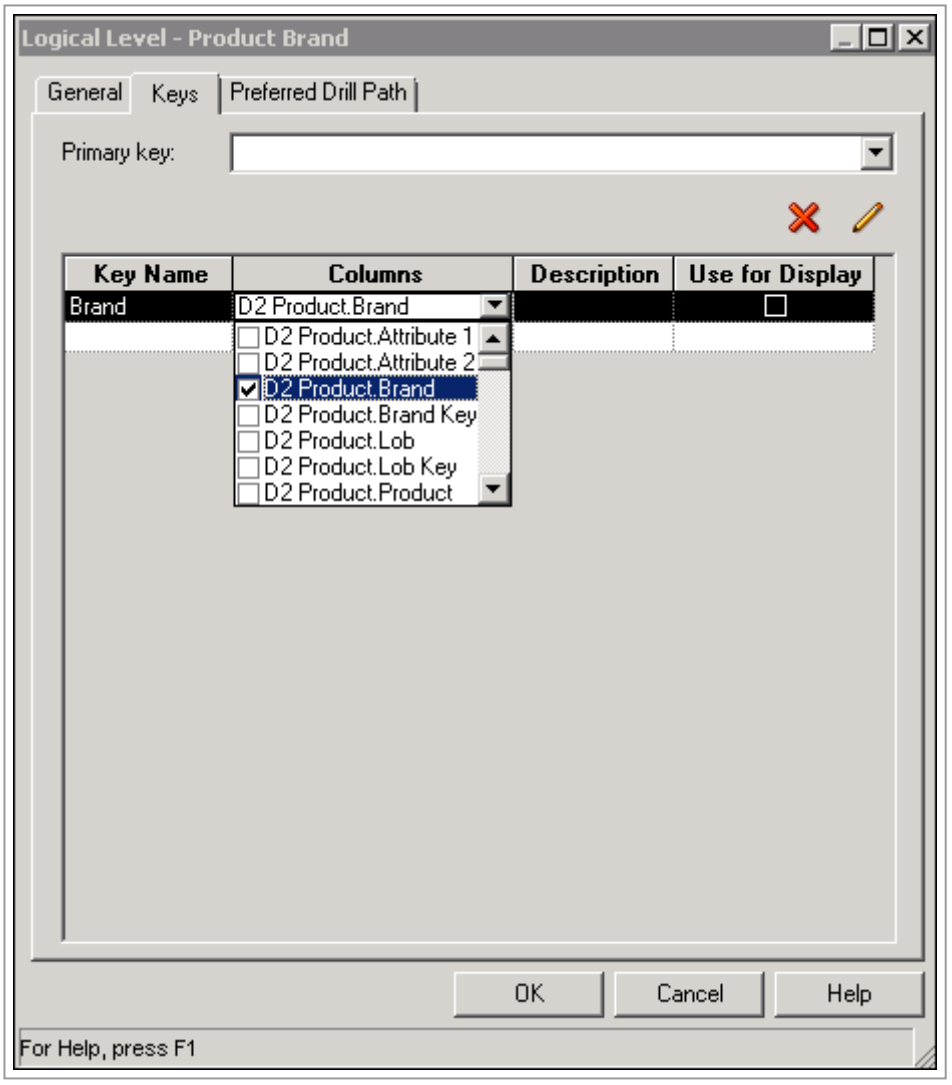

**2.** Click the **Keys** tab.



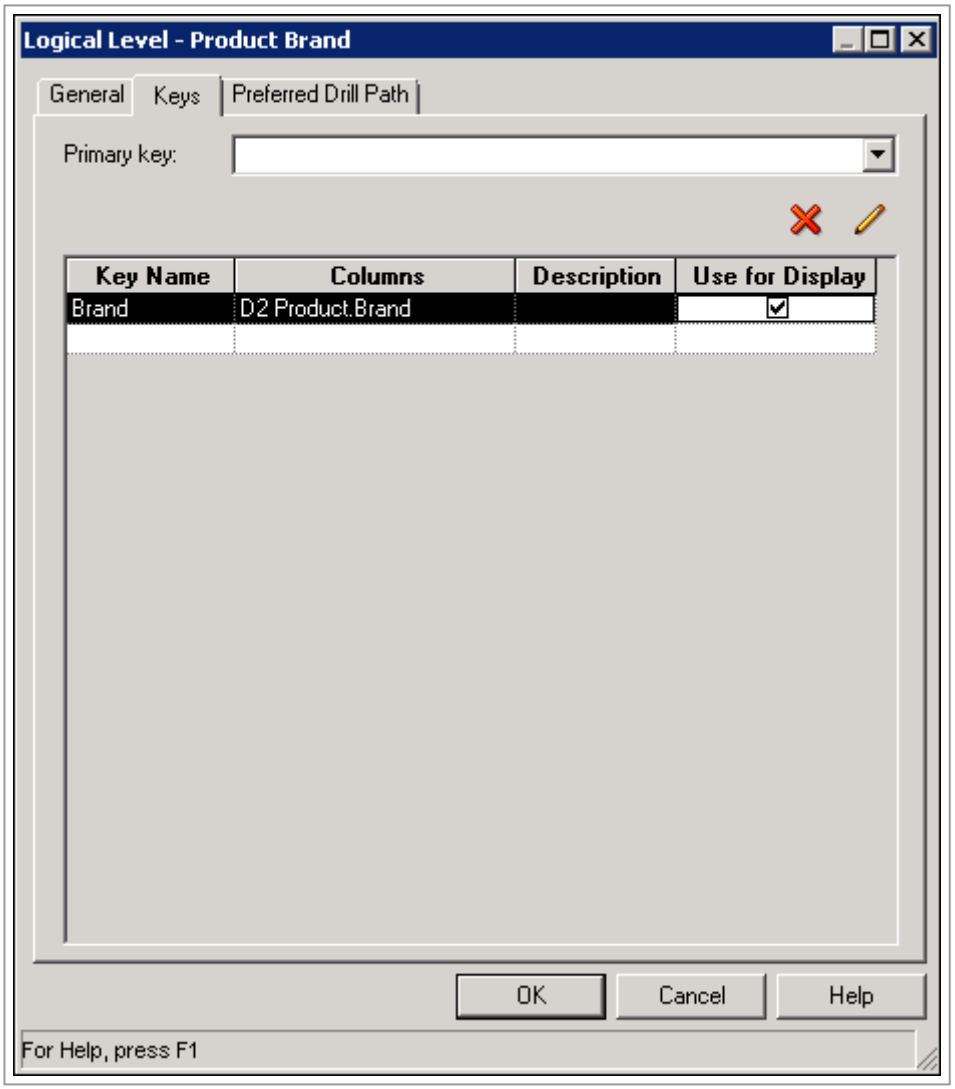

**3.** Enter **Brand** for Key Name.



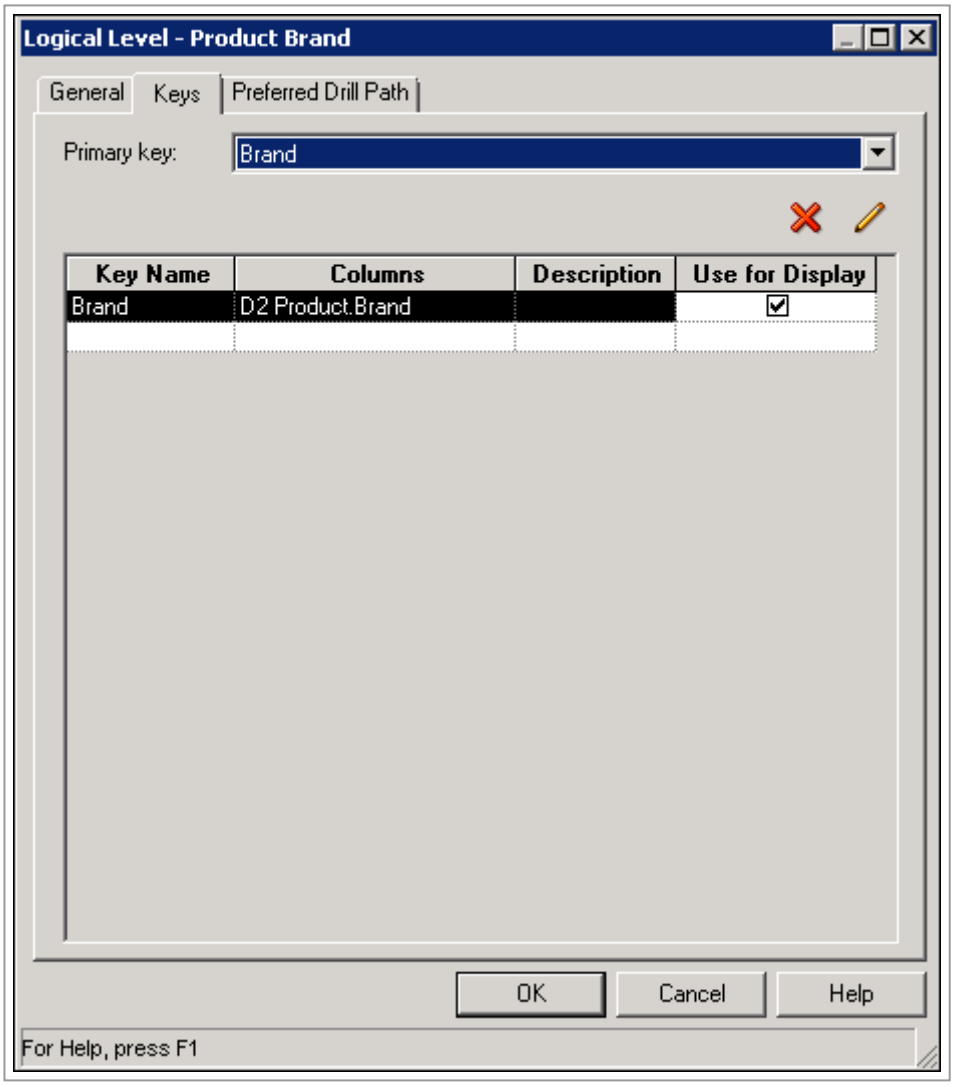

**4.** In the Columns field, use the drop down list to select **D2 Product.Brand**.



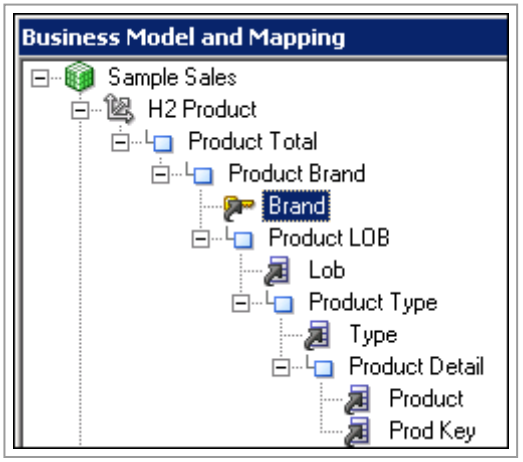

**5.** Check **Use for Display**. When this is selected, users can drill down to this column from a higher level.



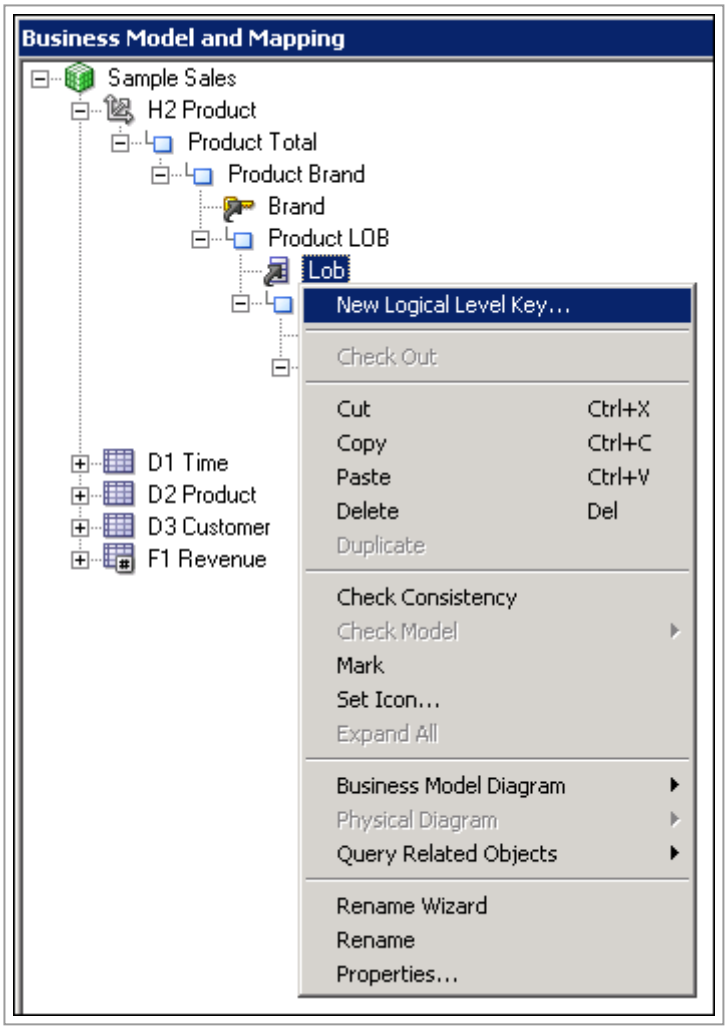

**6.** Set **Brand** as the Primary key.



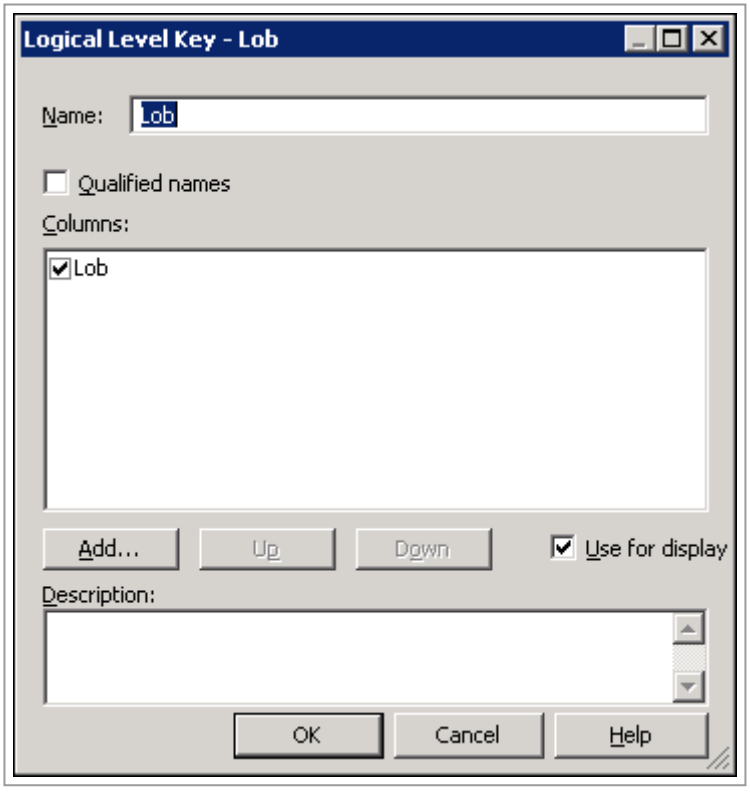

7. Click **OK** to close the Logical Level dialog box. The icon changes for Brand to show that it is the key for the Product Brand level.



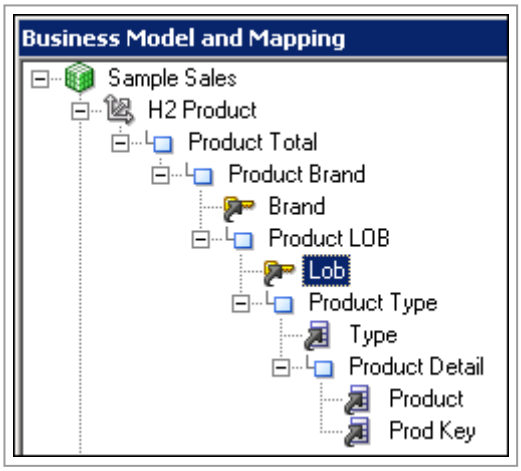

8. Use a different technique to create a logical level key: Right-click **Lob** for the Product LOB level and select **New Logical Level Key** to open the Logical Level Key dialog box.




**9.** In the Logical Level Key dialog box, accept the defaults and click **OK**.



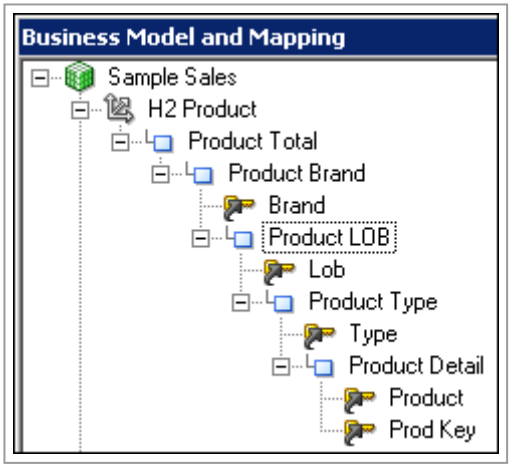
10. The icon changes for **Lob** to show that it is the key for the **Product LOB** level.




11. Use either method to set the remaining keys for the H2 Product logical dimension:

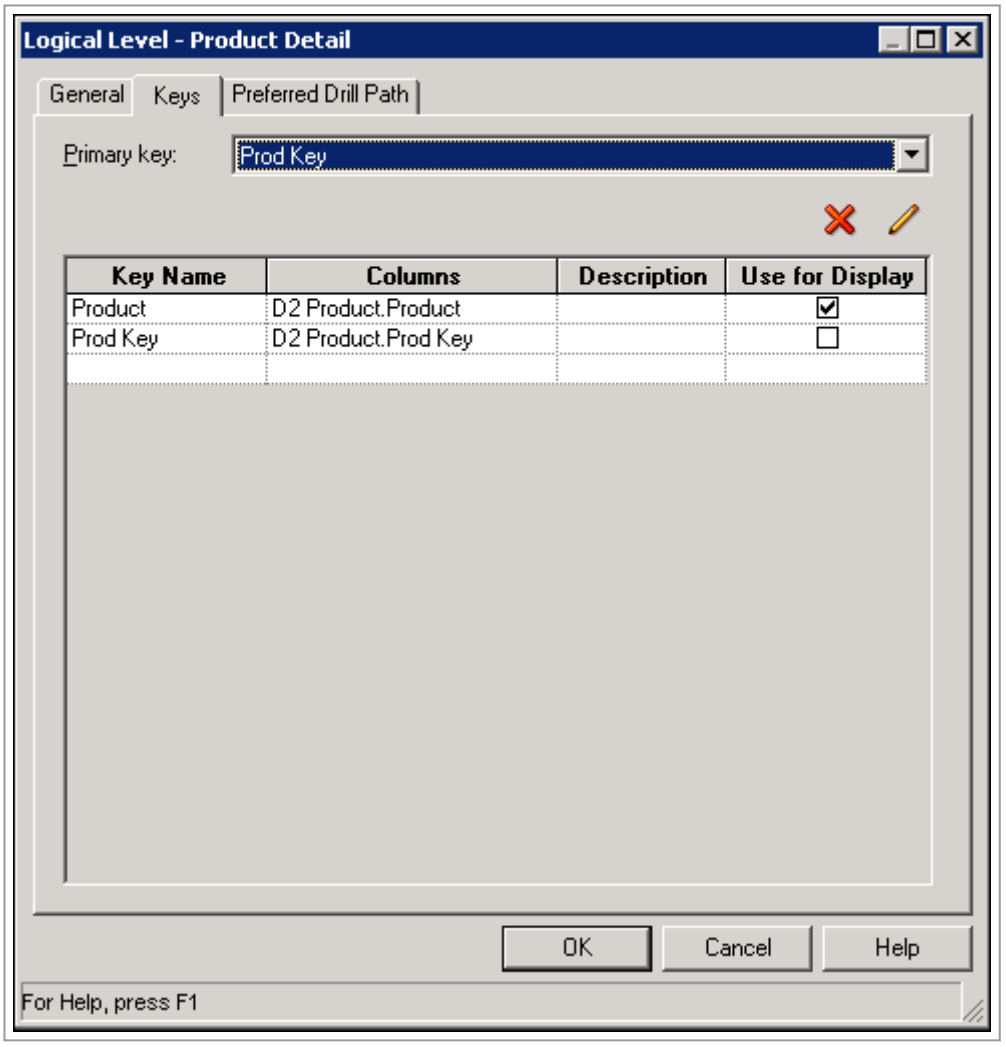
| Logical Level | Logical Level Key | Use for Display |
|---|---|---|
| Product Type | Type | Yes |
| Product Detail | Product | Yes |
| Product Detail | Prod Key | No |

Your results should look similar to the screenshot:



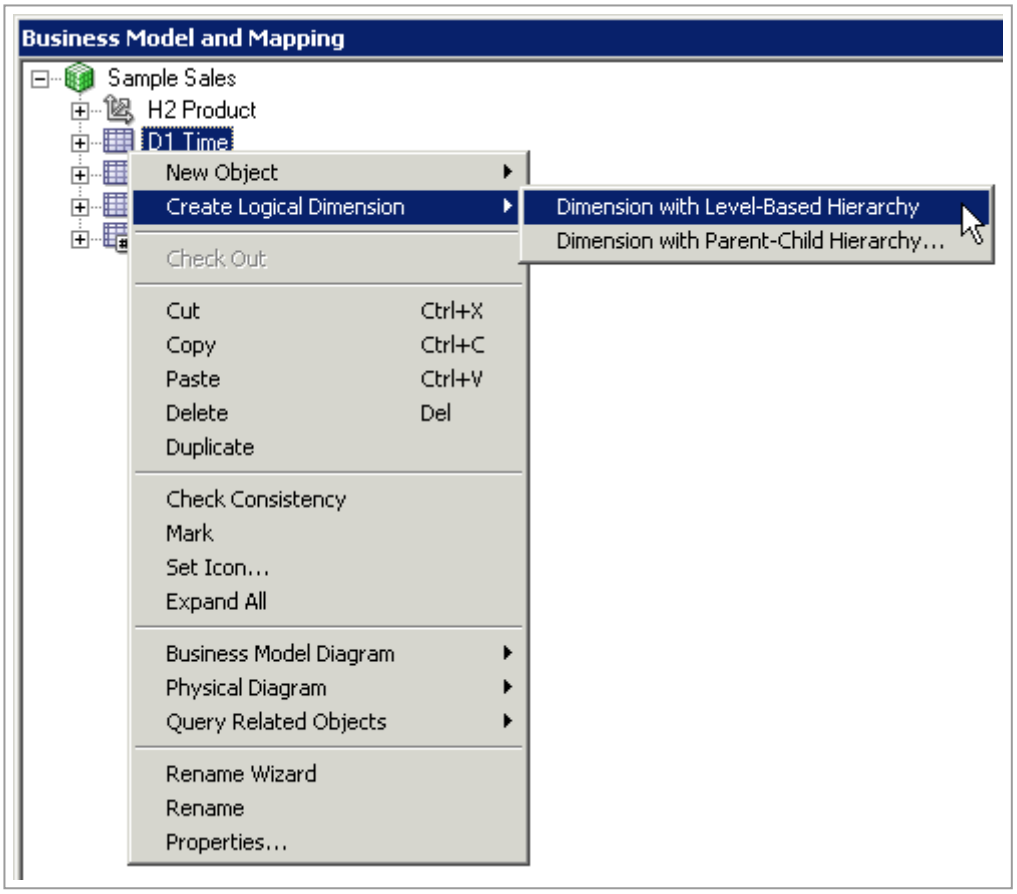

Please note that the Detail level (lowest level of the hierarchy) must have the column that is the logical key of the dimension table associated with it and it must be the key for that level: Prod Key in this example.

12. Set **Prod Key** as the primary key for the Product Detail level. Hint: Double-click the level and select the **Keys** tab.

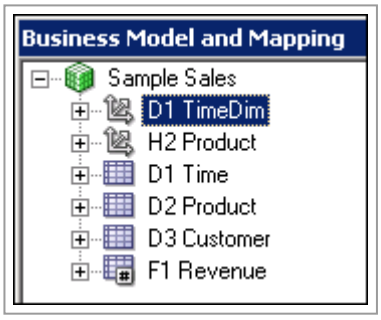



## Creating a Logical Dimension for Time

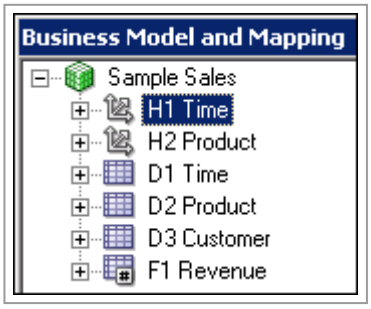
1. Use a different technique to create a logical dimension for Time. Right-click the **D1 Time** logical table and select **Create Logical Dimension > Dimension with Level-Based Hierarchy**.



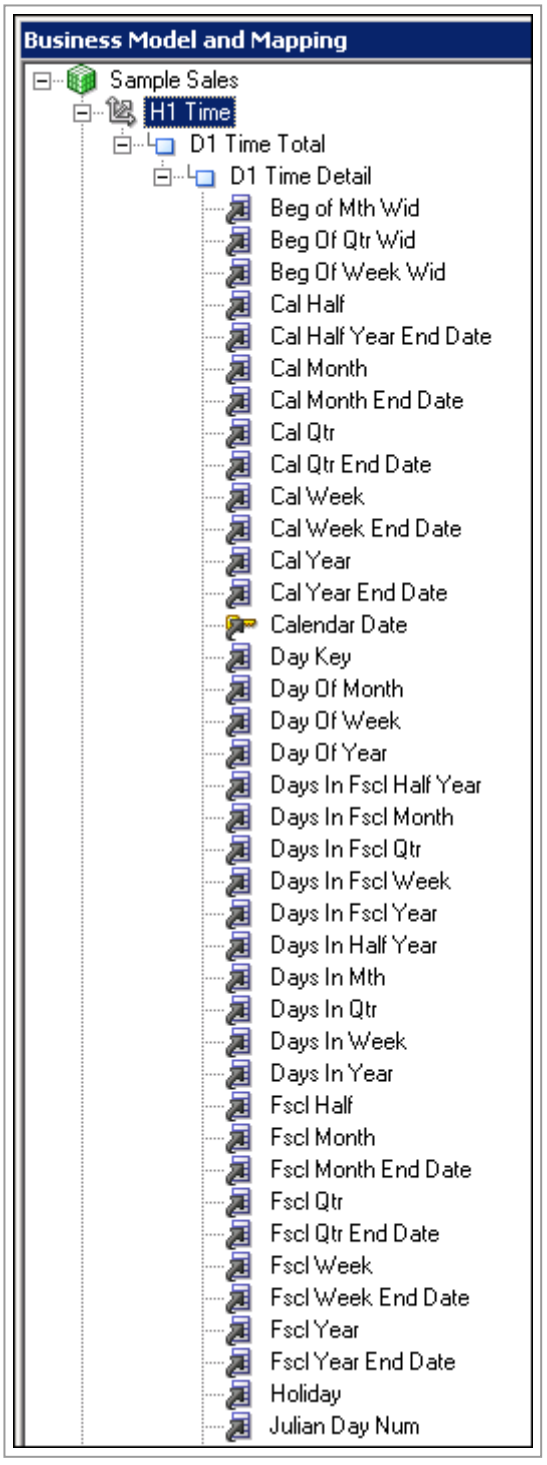

**2.** A new logical dimension, **D1 TimeDim** in this example, is automatically added to the business model.



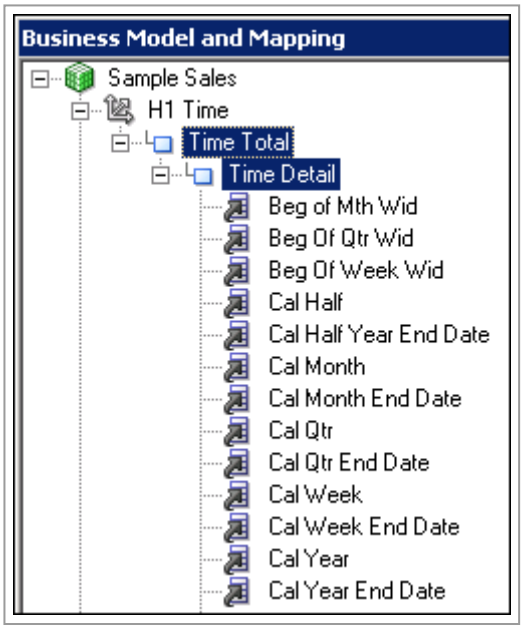

**3.** Rename **D1 TimeDim** to **H1 Time.**



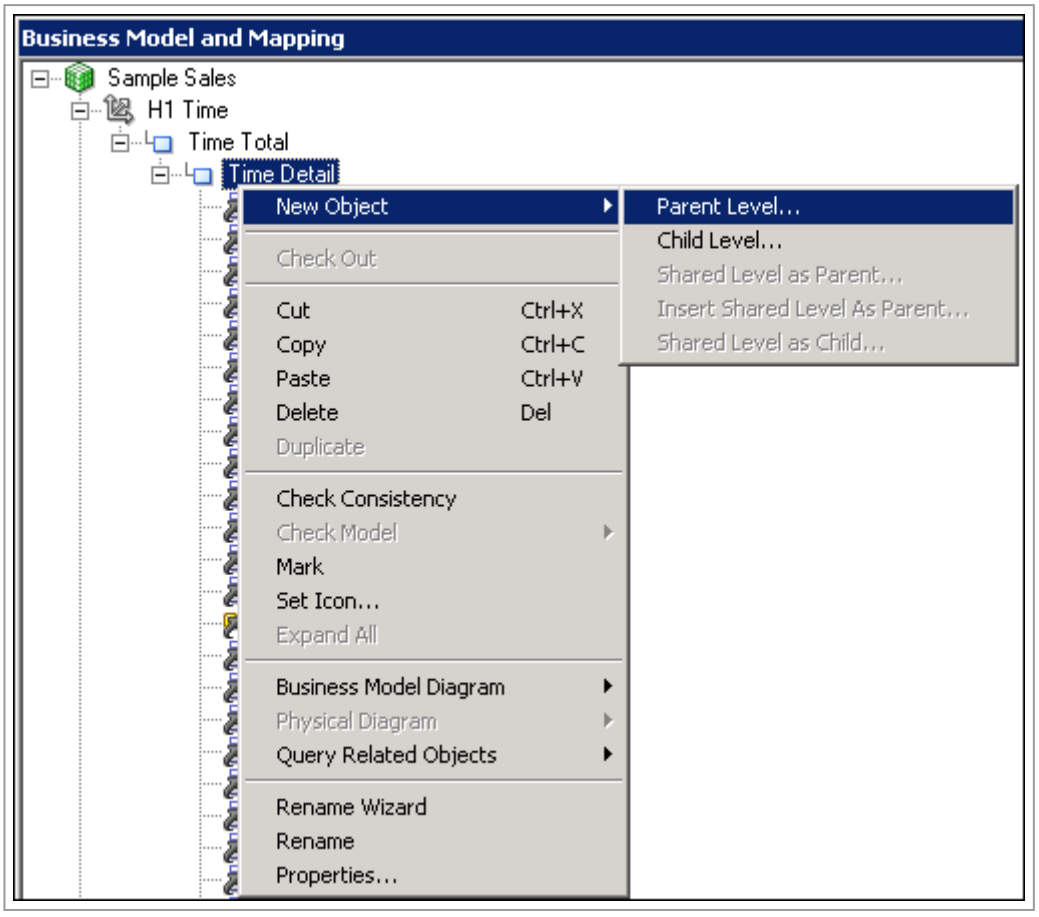

**4.** Expand **H1 Time** . Notice that two levels were created automatically: **D1 Time Total** and **D1 Time Detail**. D1 Time Detail is populated with all of the columns from the D1 Time logical table.



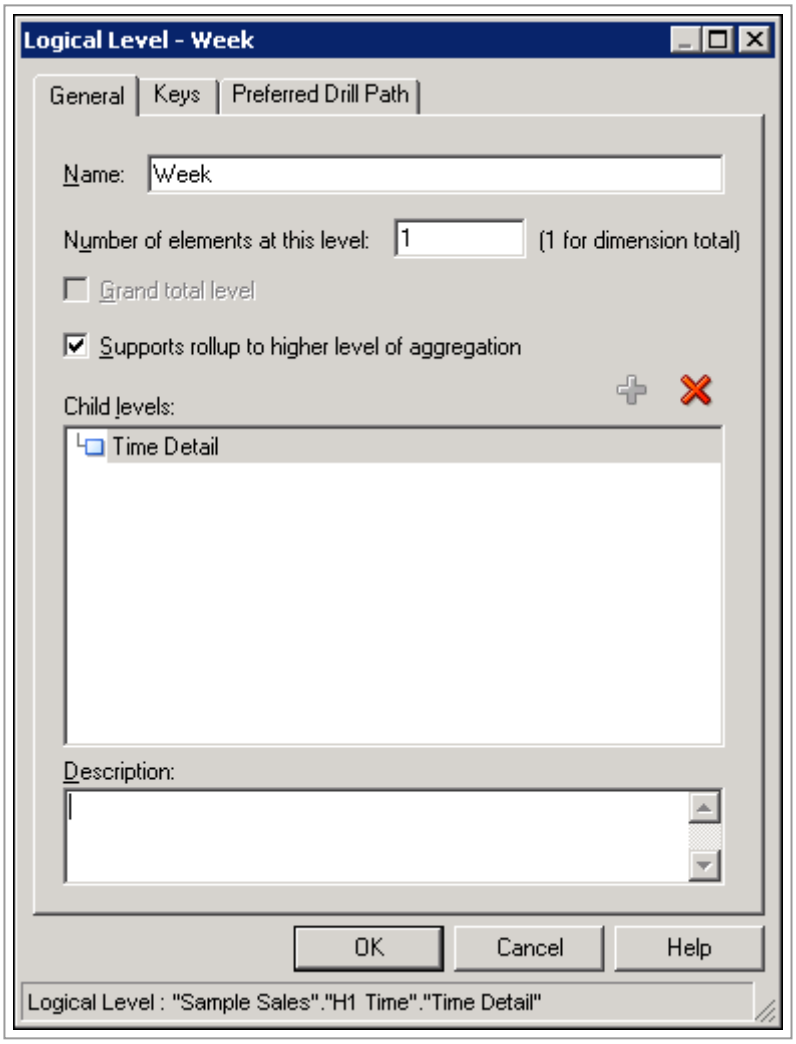

**5.** Rename **D1 Time Total** to **Time Total**, and rename **D1 Time Detail** to **Time Detail**.



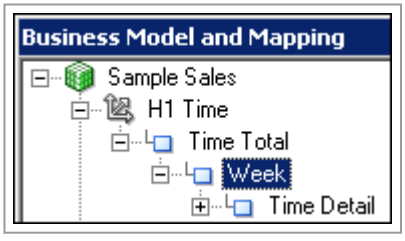

**6.** Right-click **Time Detail** and select **New Object > Parent Level** to open the Logical Level dialog box.




**7.** On the General tab, name the logical level **Week,** and check **Supports rollup to higher level of aggregation.**



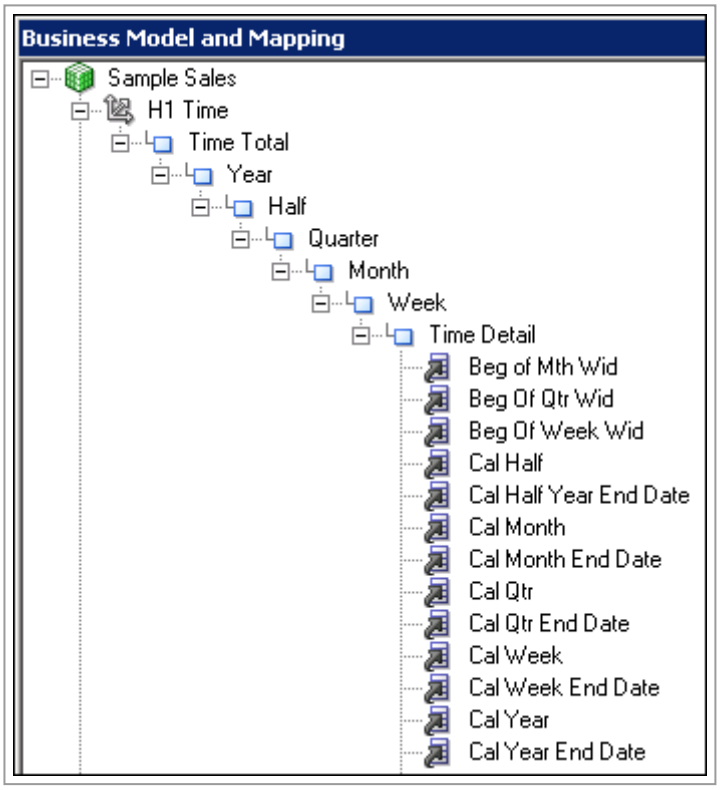

8. Click **OK** to close the Logical Level dialog box. The Week level is added to the H1 Time logical dimension.




9. Repeat the steps to add the remaining logical levels:

   **Month** as a parent of Week
   **Quarter** as a parent of Month
   **Half** as a parent of Quarter
   **Year** as a parent of Half

   Your final results should look similar to the screenshot:

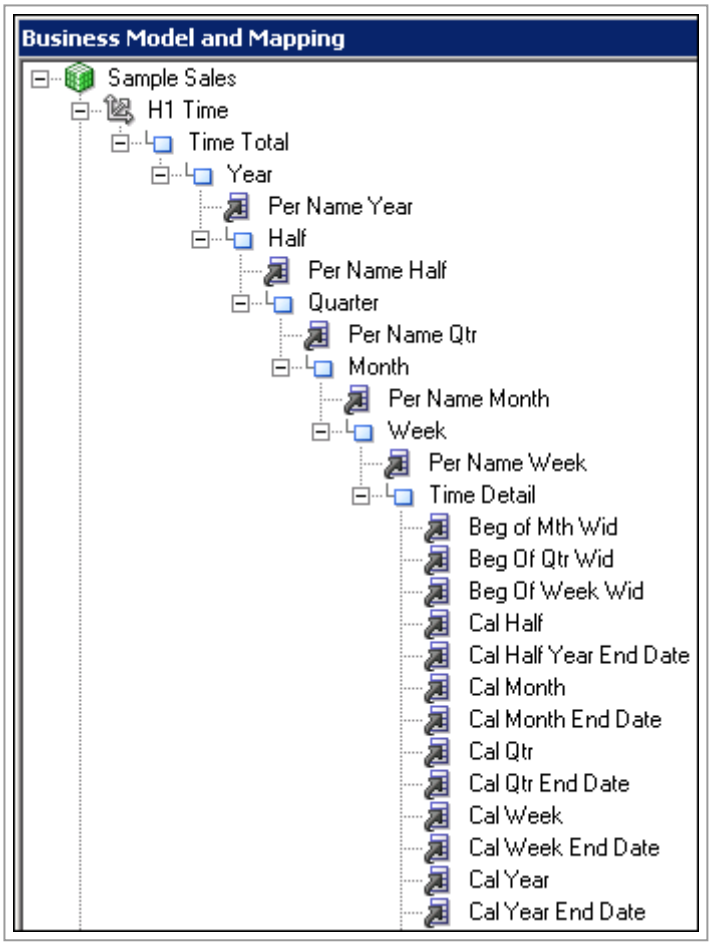



## Associating Time Logical Columns with Logical Levels

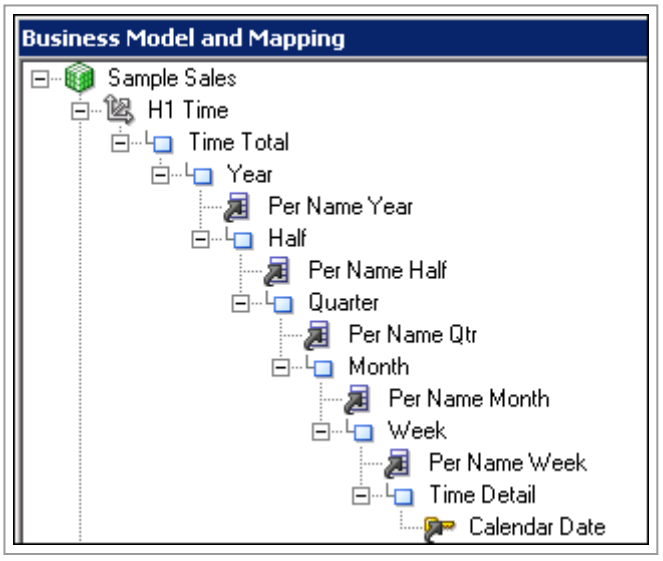
1. Use a different technique to associate logical columns with logical levels. Drag the logical columns from the Time Detail logical level (not from the D1 Time logical table) to their corresponding levels in the H1 Time logical dimension. This is a convenient technique when logical columns are buried deep in the business model.

| Logical Column | Logical Level |
|---|---|
| Per Name Year | Year |
| Per Name Half | Half |
| Per Name Qtr | Quarter |
| Per Name Month | Month |
| Per Name Week | Week |

Your results should look similar to the screenshot:




2. Delete all remaining columns from the Time Detail level except for **Calendar Date** so that only **Calendar Date** is associated with the Time Detail level. Notice that deleting objects from the hierarchy does not delete them from the logical table in the business model.

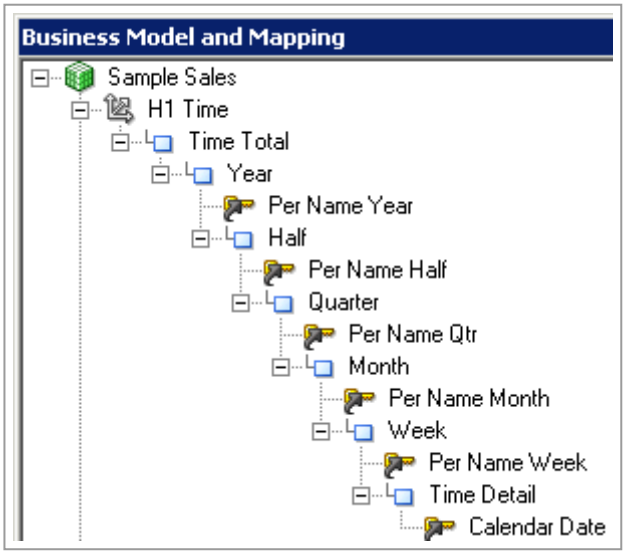



3. Set the logical keys for the H1 Time logical dimension according to the following table:

| Logical Level | Level Key | Use for Display |
|---|---|---|
| Year | Per Name Year | Yes |
| Half | Per Name Half | Yes |

| | | |
|---|---|---|
| Quarter | Per Name Qtr | Yes |

| | | |
|---|---|---|
| Month | Per Name Month | Yes |
| Week | Per Name Week | Yes |
| Time Detail | Calendar Date | Yes |




## Creating a Logical Dimension for Customer

1. Use either technique to create a logical dimension with a level-based hierarchy named **H3 Customer** for the D3 Customer logical table with the following levels, columns, and keys. Hint: Create the levels first, then double-click a logical column to open the Logical Column dialog box and use the Levels tab to associate the logical column with a logical level.

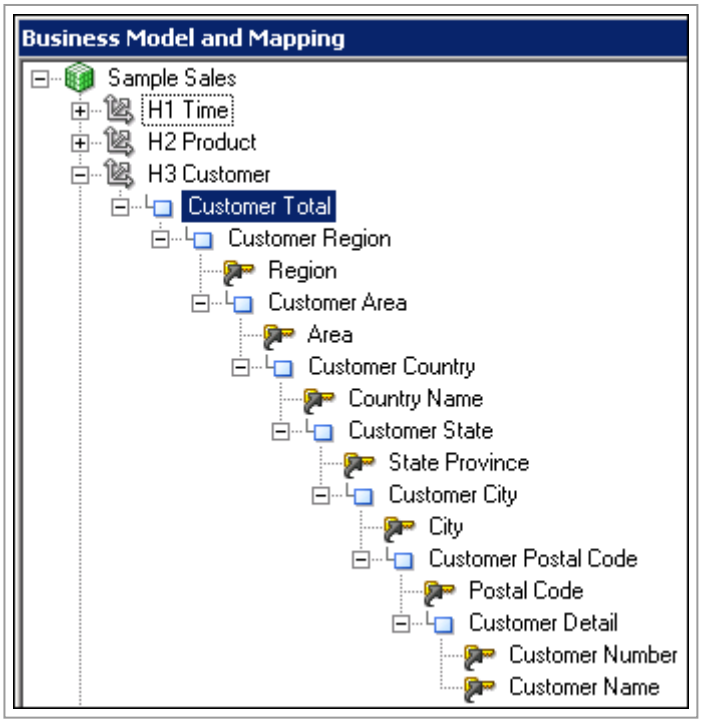
| **Level** | **Column** | **Key** | **Use for Display** |
|---|---|---|---|
| Customer Total | <none> | <none> | <none> |
| Customer Region | Region | Region | Yes |
| Customer Area | Area | Area | Yes |
| Customer Country | Country Name | Country Name | Yes |
| Customer State | State Province | State Province | Yes |
| Customer City | City | City | Yes |
| Customer Postal Code | Postal Code | Postal Code | Yes |

| | | | |
|---|---|---|---|
| Customer Detail | Customer Name<br><br>Customer Number | Customer Name<br><br>Customer Number | Yes<br><br>No |

Set **Customer Total** as the grand total level.

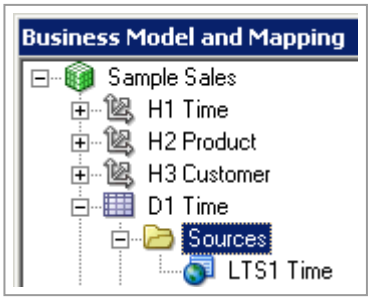
Set **Customer Number** as the primary key for the Customer Detail level.

Your results should look similar to the screenshot:

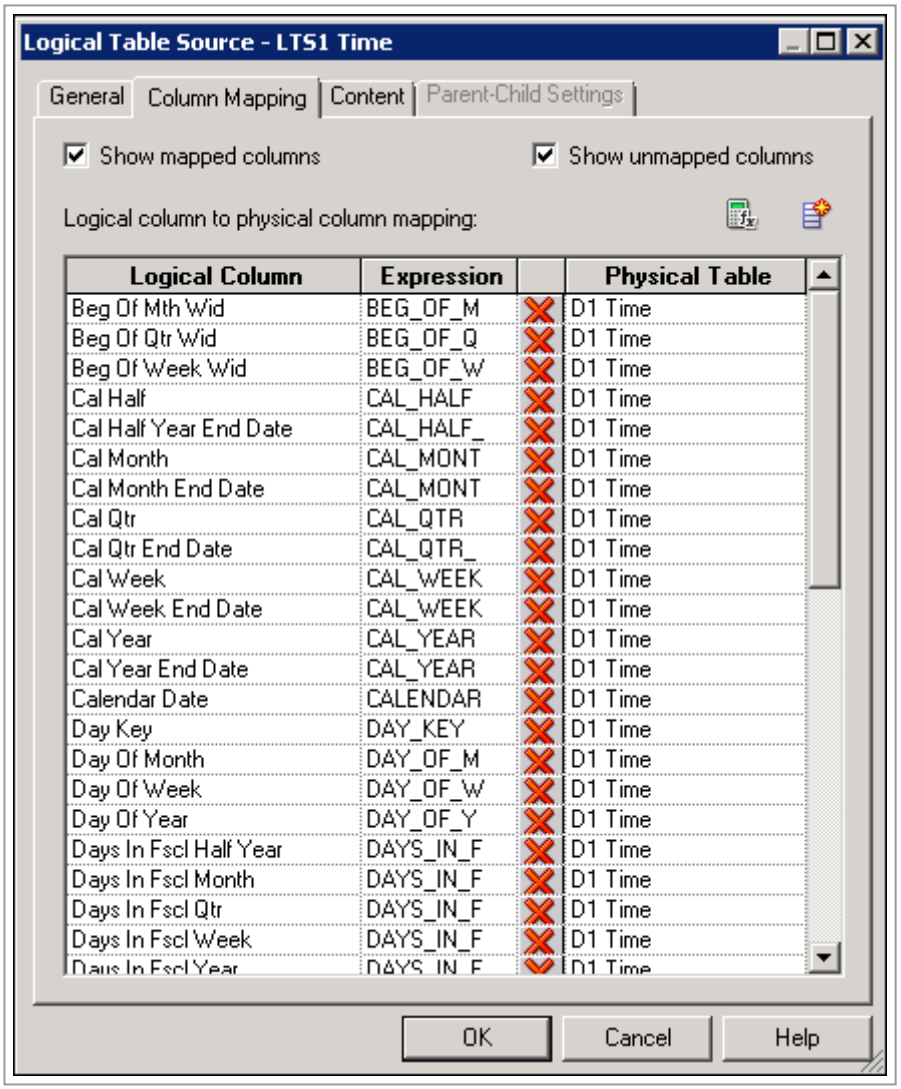



## Setting Aggregation Content for Logical Table Sources

1. Expand **D1 Time > Sources**.

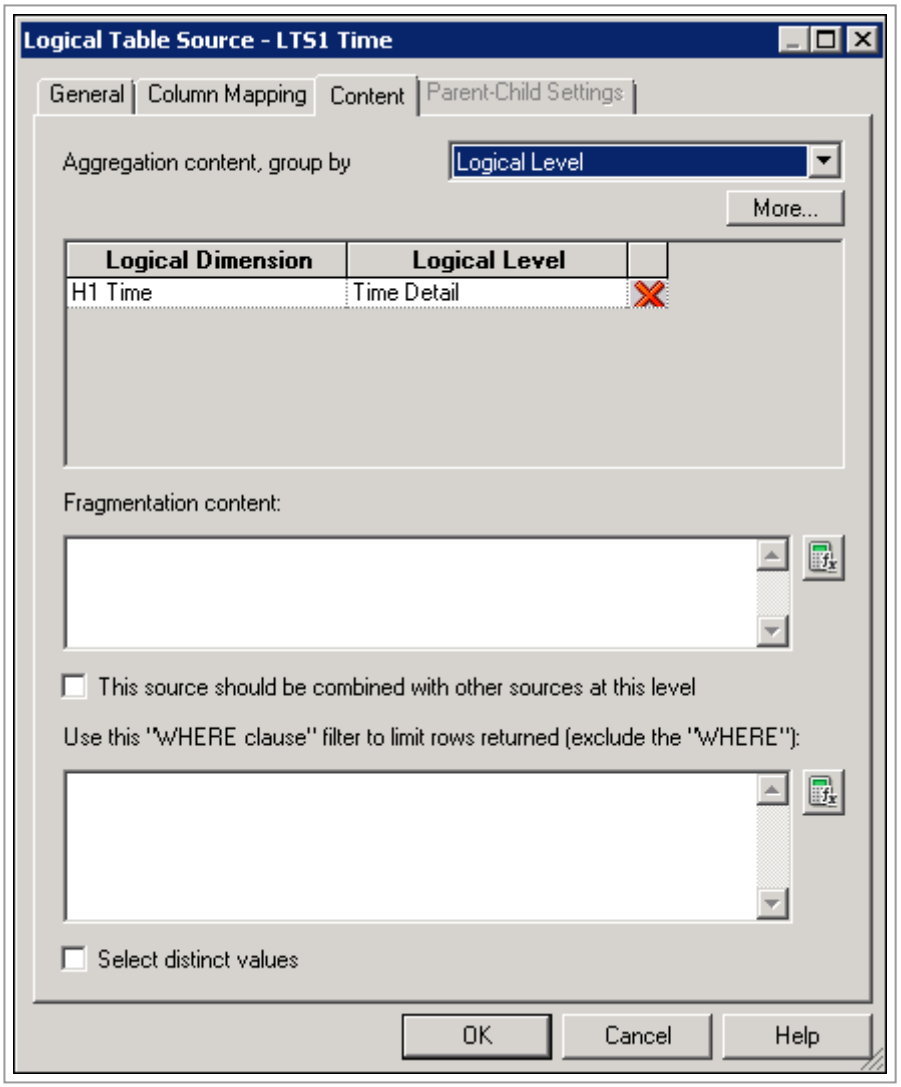



2. Double-click the **LTS1 Time** logical table source to open the Logical Table Source dialog box.




3. Click the **Content** tab.

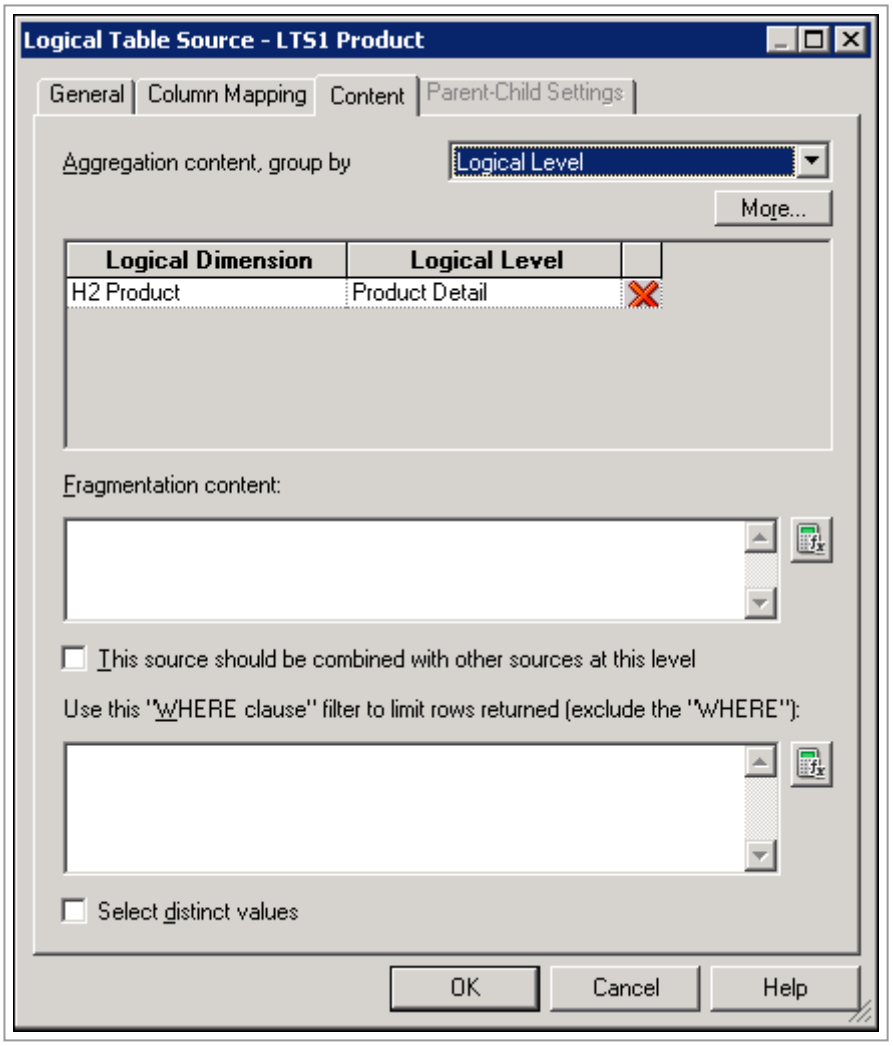
4. Confirm that **Aggregation content, group by** is set to **Logical Level** and the logical level is set to **Time Detail** for the **H1 Time** logical dimension.

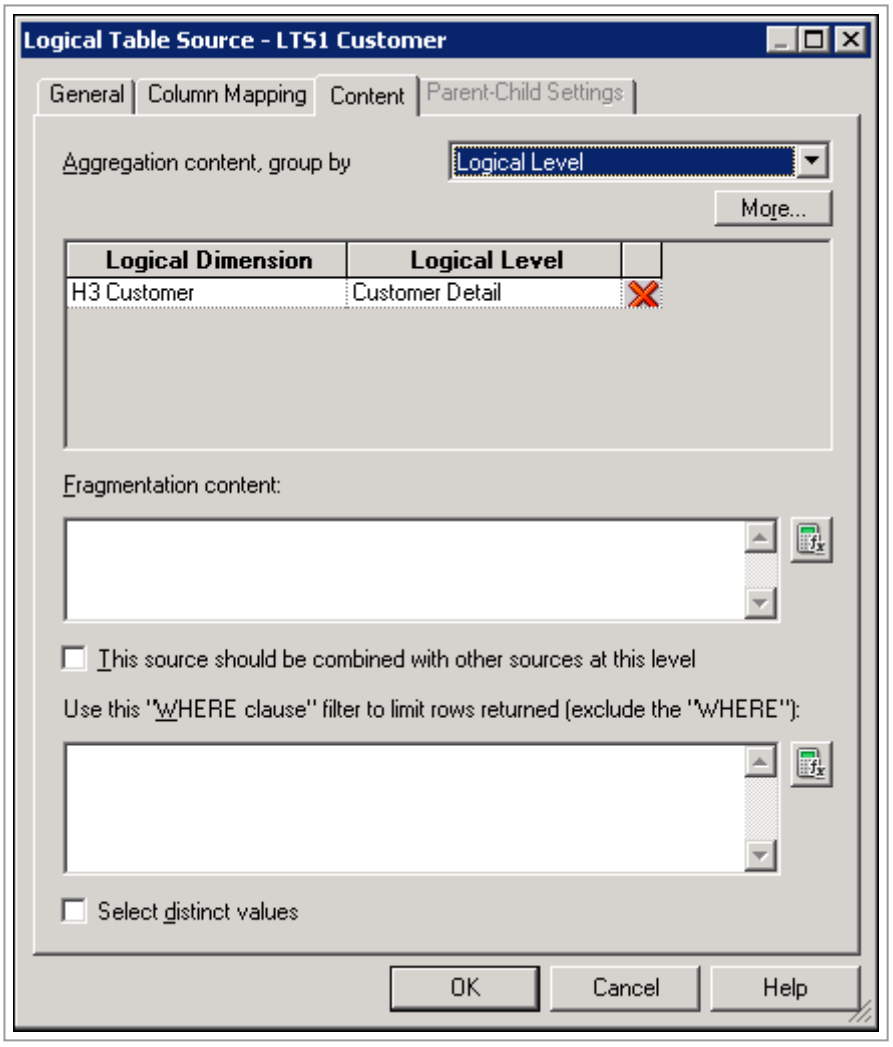


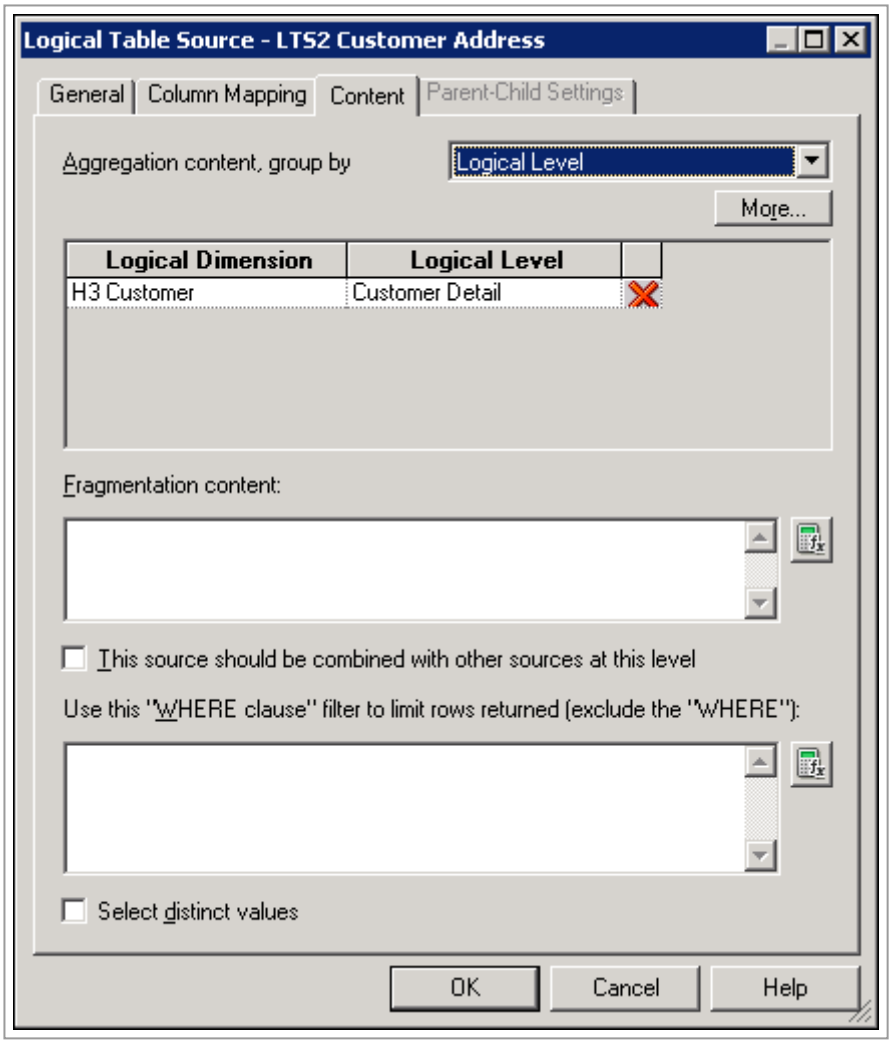

5. Click **OK** to close the Logical Table Source dialog box.

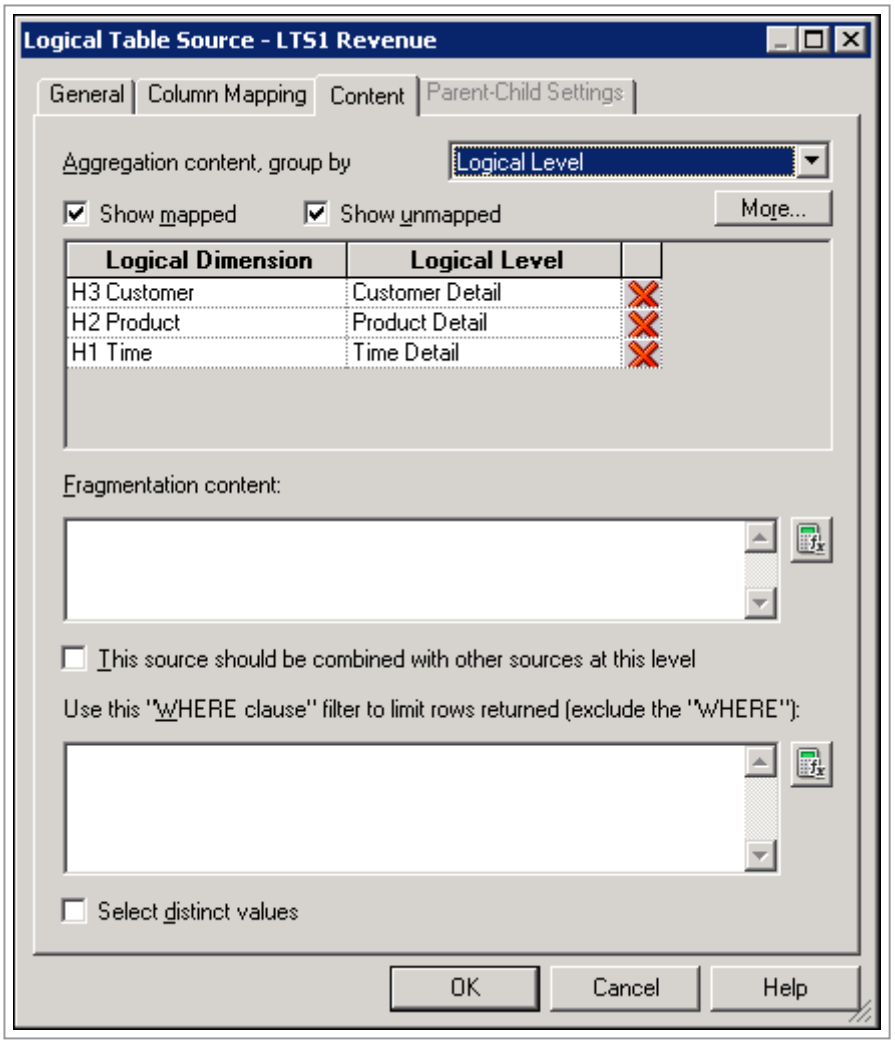
6. Repeat to verify or set content settings for the remaining logical table sources using the table and screenshots as a guide:

| **Logical Table Source** | **Logical Dimension** | **Logical Level** |
|---|---|---|
| LTS1 Product | H2 Product | Product Detail |
| LTS1 Customer | H3 Customer | Customer Detail |
| LTS2 Customer Address | H3 Customer | Customer Detail |
| LTS1 Revenue | H1 Time<br>H2 Product<br>H3 Customer | Time Detail<br>Product Detail<br>Customer Detail |


Logical Table Source - LTS1 Product
General
Column Mapping
Content
Parent-Child Settings
Aggregation content, group by
Logical Level
More...
Logical Dimension
Logical Level
H2 Product
Product Detail
Fragmentation content:
This source should be combined with other sources at this level
Use this "WHERE clause" filter to limit rows returned (exclude the "WHERE"):
Select distinct values
OK
Cancel
Help


Logical Table Source - LTS1 Customer
General
Column Mapping
Content
Parent-Child Settings
Aggregation content, group by
Logical Level
More...
Logical Dimension
Logical Level
H3 Customer
Customer Detail
Fragmentation content:
This source should be combined with other sources at this level
Use this "WHERE clause" filter to limit rows returned (exclude the "WHERE"):
Select distinct values
OK
Cancel
Help


Logical Table Source - LTS2 Customer Address
General
Column Mapping
Content
Parent-Child Settings
Aggregation content, group by
Logical Level
More...
Logical Dimension
Logical Level
H3 Customer
Customer Detail
Fragmentation content:
This source should be combined with other sources at this level
Use this "WHERE clause" filter to limit rows returned (exclude the "WHERE"):
Select distinct values
OK
Cancel
Help




7. Save the repository and check global consistency. Fix any errors or warnings before proceeding. Notice that you did not have to make any changes to the **Presentation** layer.

8. Close the repository. Leave the Administration Tool open.

## Testing Your Work

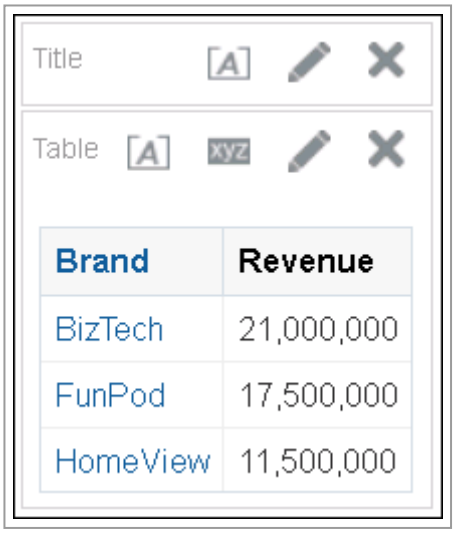
1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

2. Return to Oracle BI, which should still be open, and sign in if necessary.

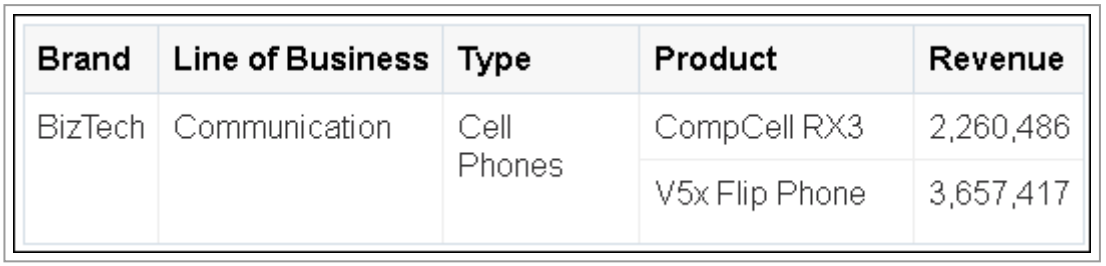
3. Create the following analysis to test the Product hierarchy.

   **Products.Brand**
   **Base Facts.Revenue**

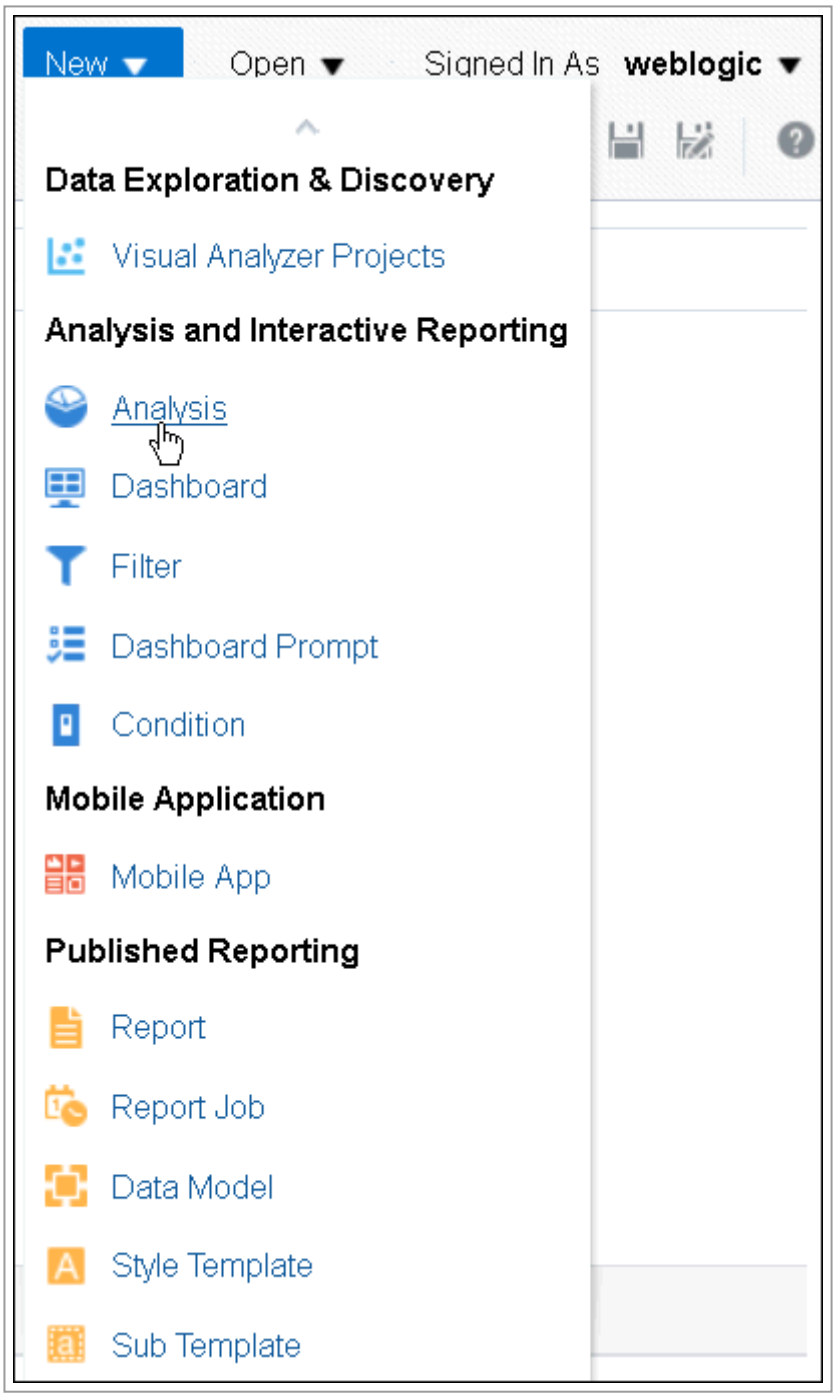
4. Click **Results**.



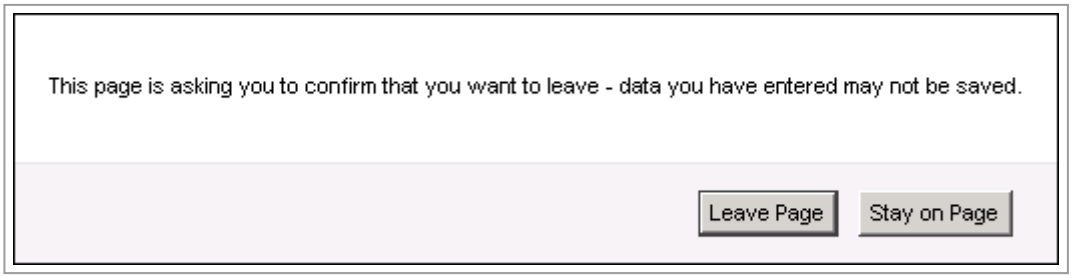

5. Click on the **BizTech** brand and verify that you can drill down through the hierarchy to see revenue data at each level.




6. Select **New > Analysis > Sample Sales**.




7. Click **Leave Page** when prompted with the message: **"This page is asking you to confirm that you want to leave - data you have entered may not be saved."**

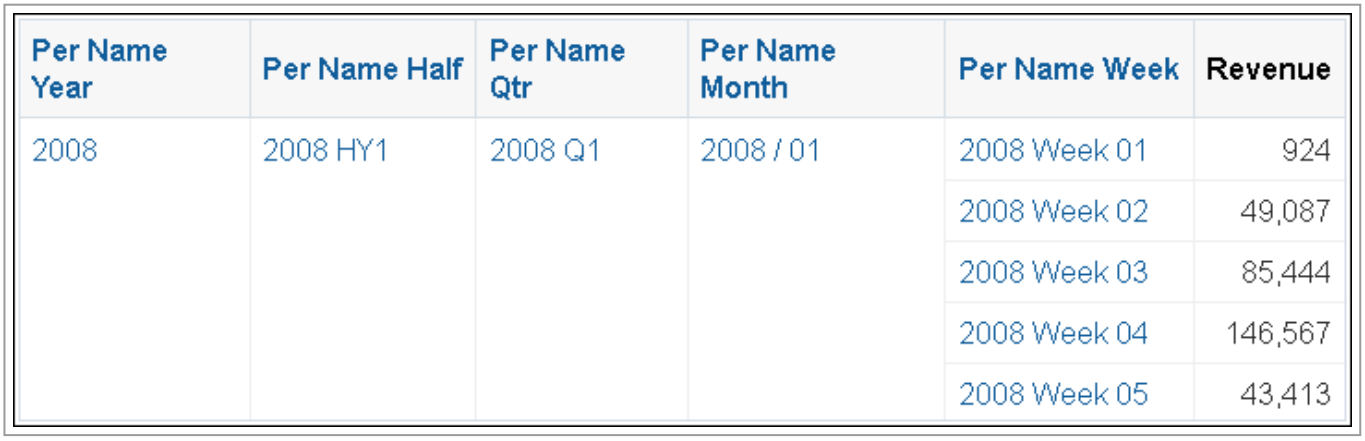



8. Create the following analysis:

**Time.Per Name Year**

**Base Facts.Revenue**


9. Click **Results** and verify that you can drill down through the Time hierarchy.



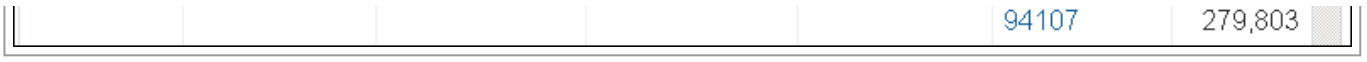
| Per Name Year | Per Name Half | Per Name Qtr | Per Name Month | Per Name Week | Revenue |
|---|---|---|---|---|---|
| 2008 | 2008 HY1 | 2008 Q1 | 2008 / 01 | 2008 Week 01 | 924 |
| | | | | 2008 Week 02 | 49,087 |
| | | | | 2008 Week 03 | 85,444 |
| | | | | 2008 Week 04 | 146,567 |
| | | | | 2008 Week 05 | 43,413 |

10. Repeat the steps and create the following analysis to test the Customers hierarchy:

**Customer Regions.Region**
**Base Facts.Revenue**


11. Click **Results** and verify that you can drill down through the Customers hierarchy.

| Region | Area | Country Name | State Province | City | Postal Code | Revenue |
|---|---|---|---|---|---|---|
| AMERICAS | North America | United States | California | San Francisco | 94005 | 80,426 |
| | | | | | 94014 | 119,606 |
| | | | | | 94015 | 434,123 |
| | | | | | 94044 | 58,049 |
| | | | | | 94066 | 480,316 |
| | | | | | 94080 | 506,799 |
| | | | | | 94102 | 415,789 |
| | | | | | 94103 | 719,924 |
| | | | | | 94104 | 87,536 |
| | | | | | 94105 | 532,912 |




12. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Creating Level-Based Measures

In this set of steps you create level-based measures that calculate total dollars at various levels in the Product hierarchy, and then use a level-based measure to create a share measure.

To create level-based measures and a share measure, you perform the following steps:

- Opening the Repository in Offline Mode
- Creating Level-Based Measures
- Creating a Share Measure
- Testing Your Work

### Opening the Repository in Offline Mode

1. Return to the Administration Tool, which should still be open.

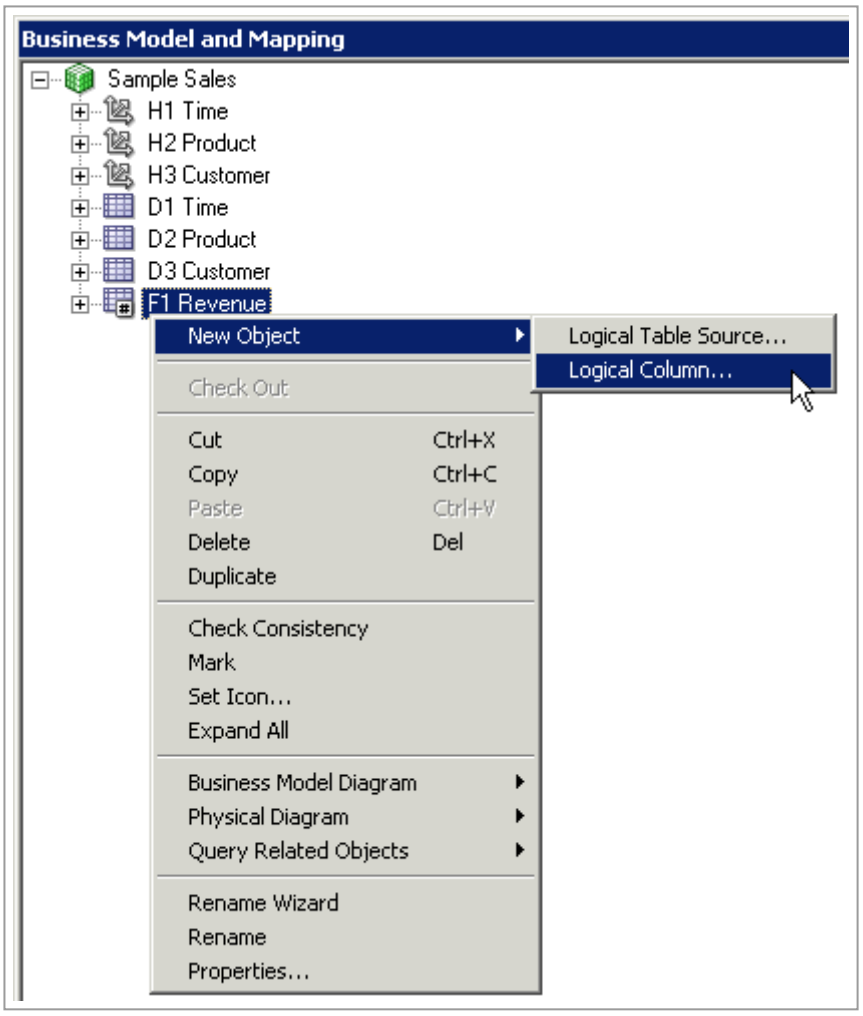
2. Select **File > Open > Offline**.

3. Select **BISAMPLE.rpd** and click **Open**.

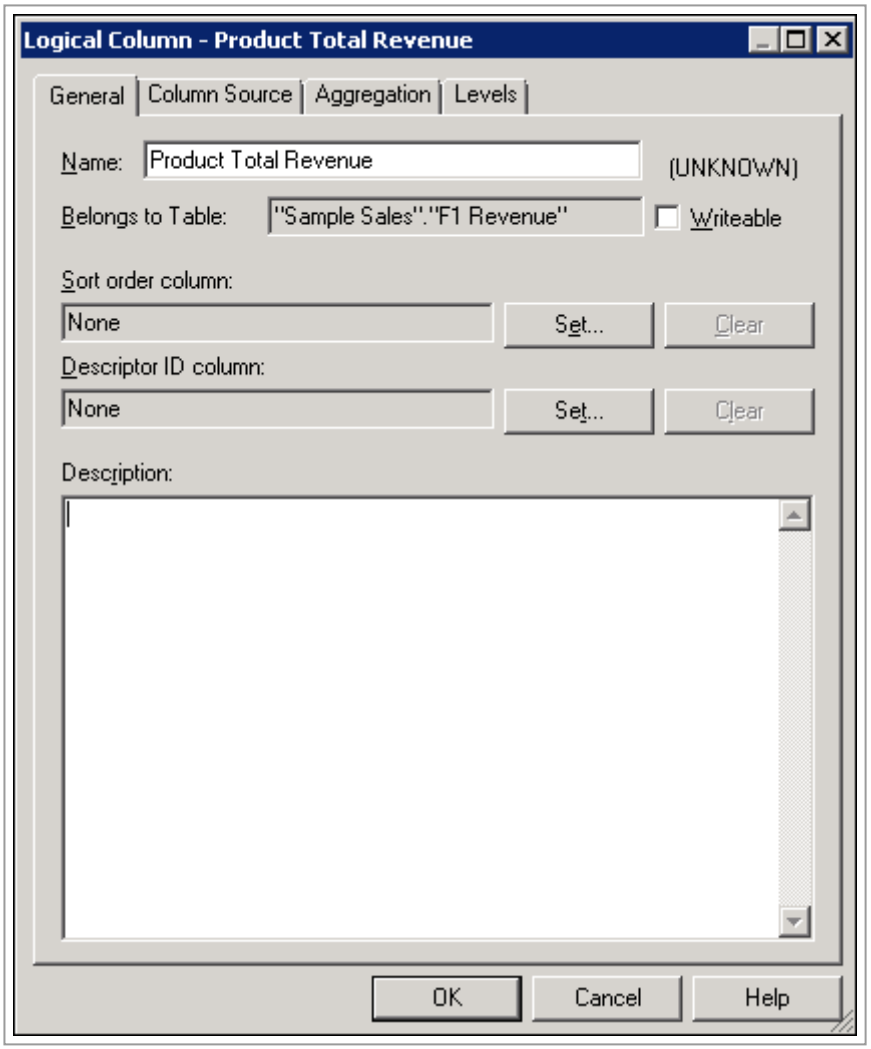
4. Enter **Admin123** as the repository password and click **OK** to open the repository.

### Creating Level-Based Measures

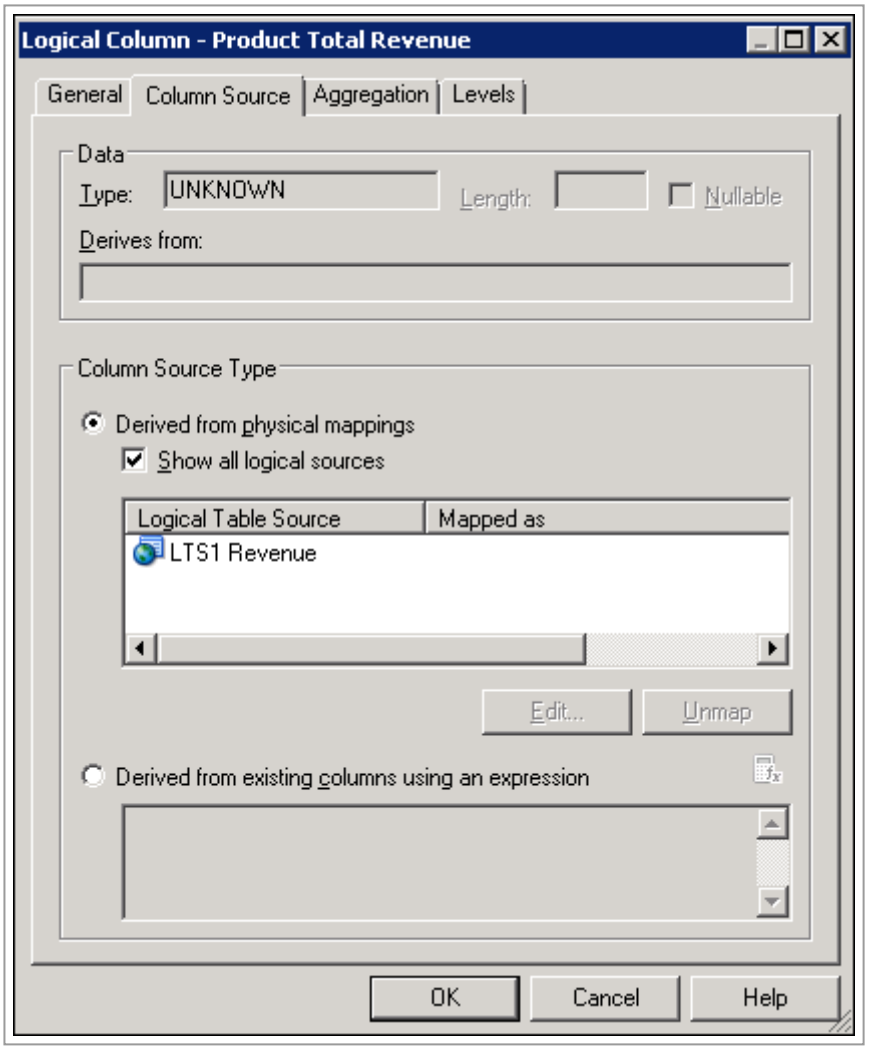
1. In the **BMM** layer, right-click the **F1 Revenue** table and select **New Object > Logical Column** to open the Logical Column dialog box.



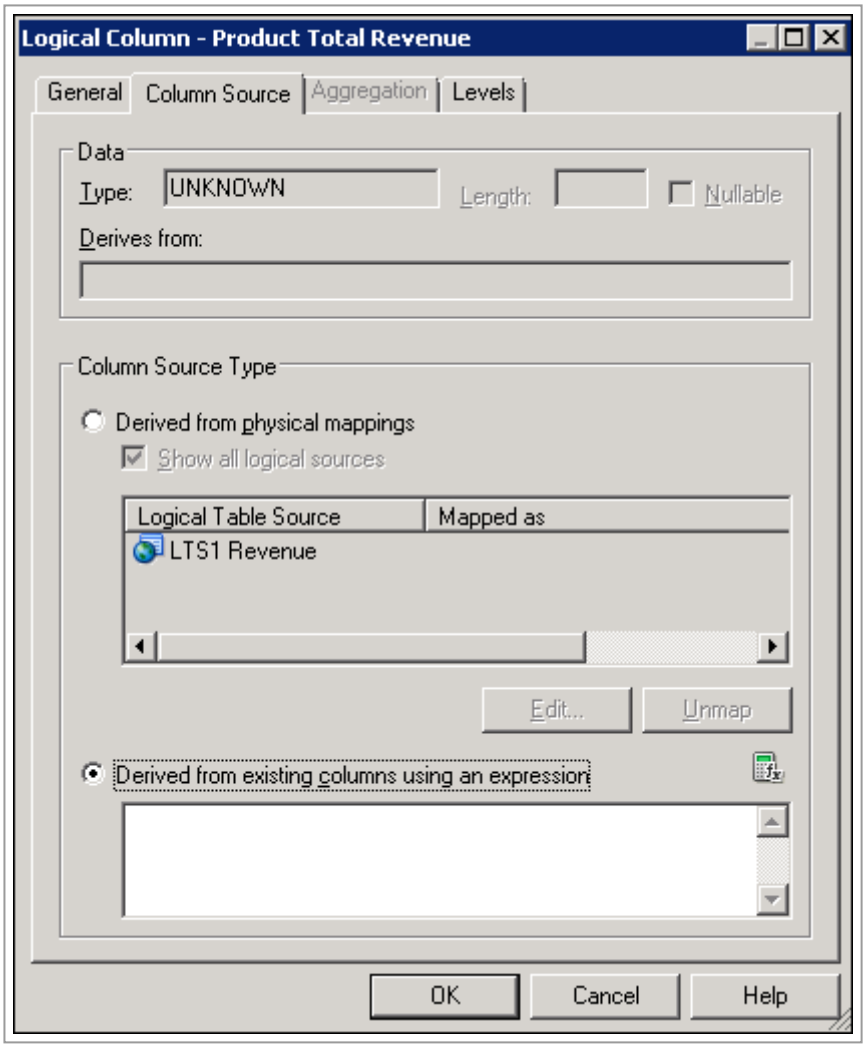

**2.** On the **General** tab, enter **Product Total Revenue** in the Name field.



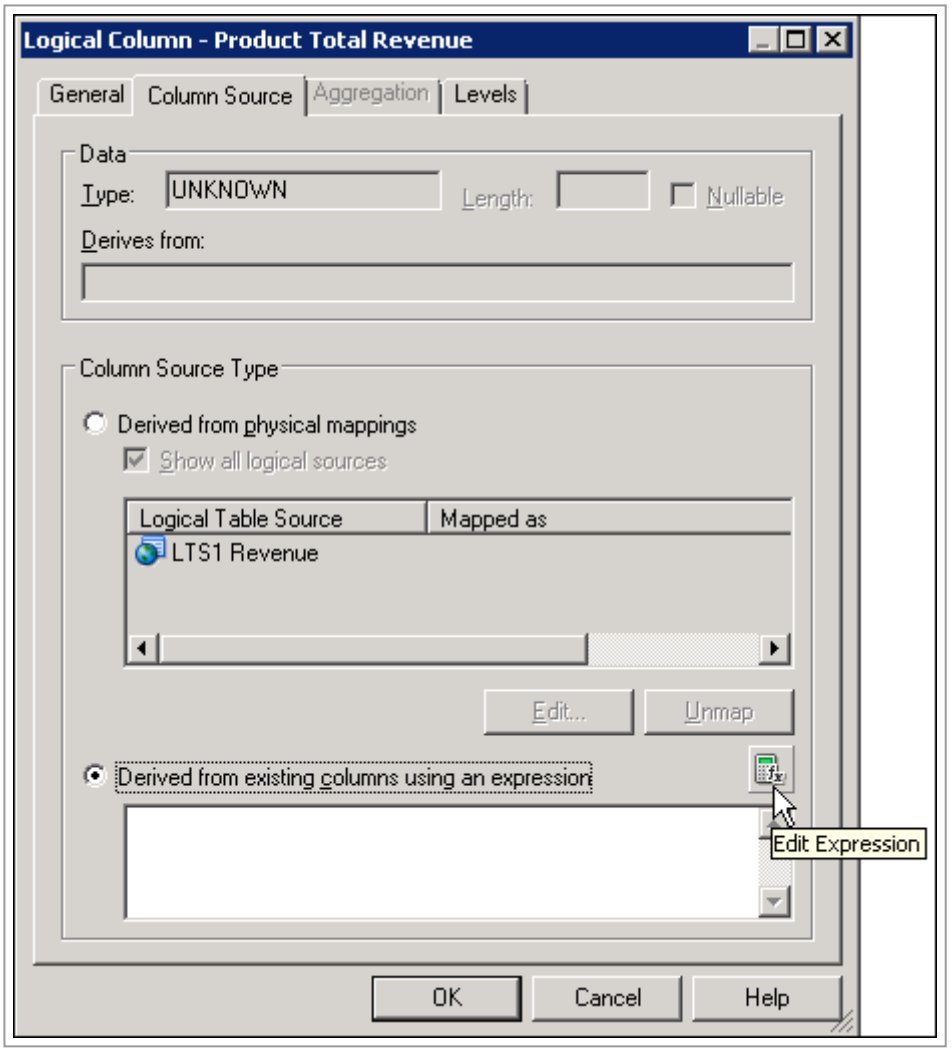

**3.** Click the **Column Source** tab.



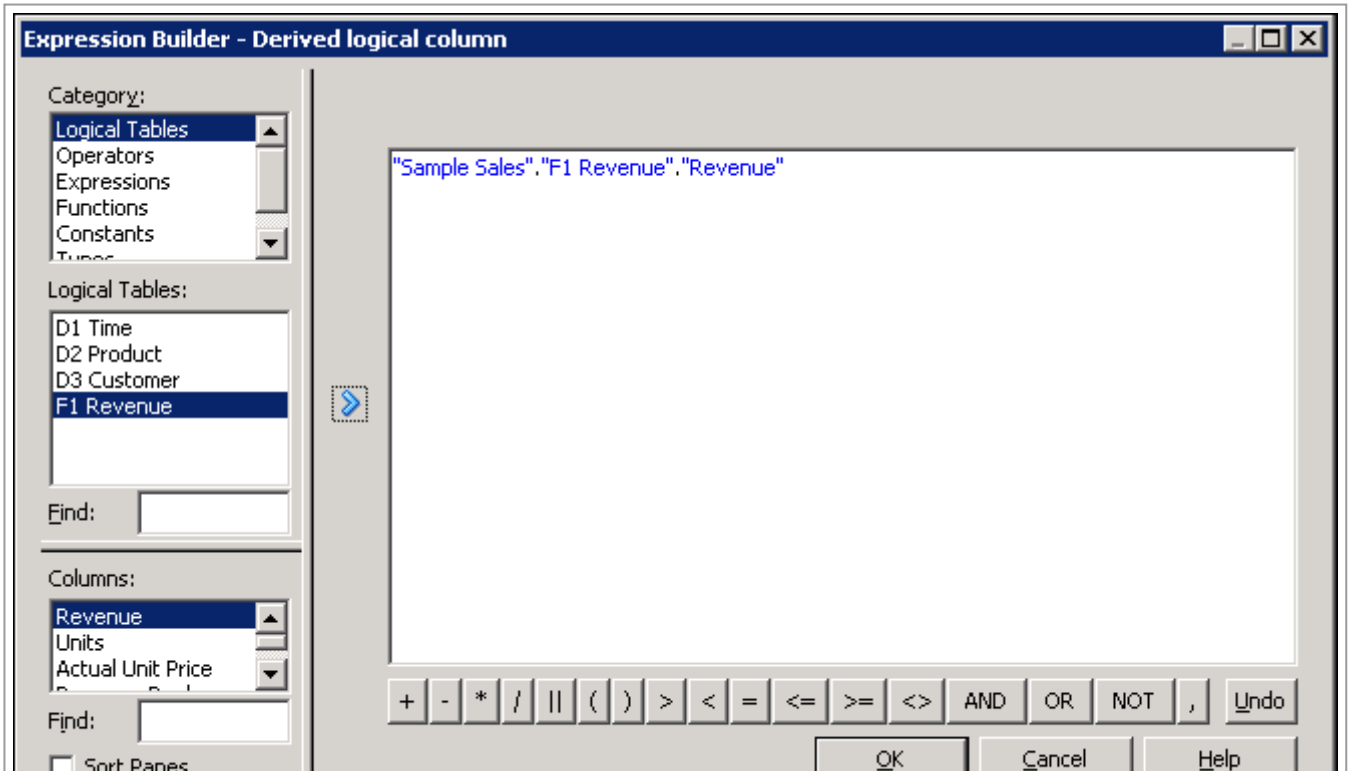

**4.** Select **Derived from existing columns using an expression**.

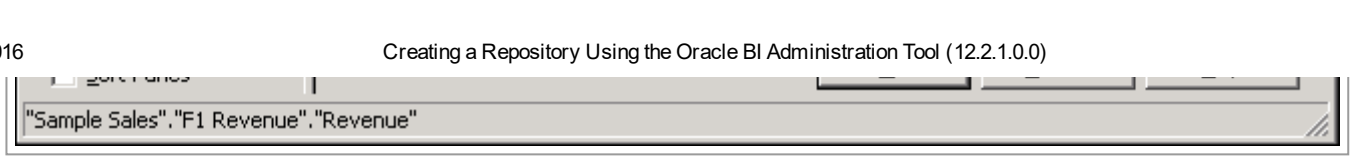



**5.** Open the **Expression Builder**.



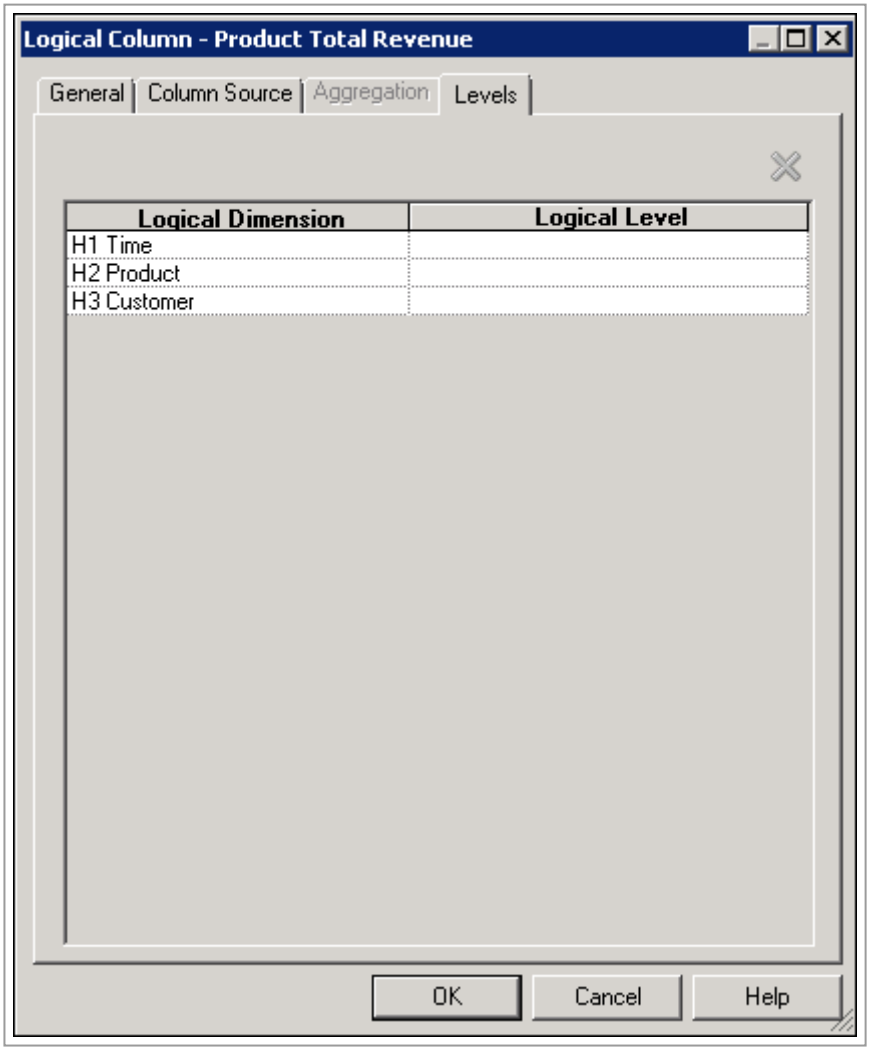

6. In the Expression Builder, add **Logical Tables > F1 Revenue > Revenue** to the expression. Recall that the Revenue column already has a default aggregation rule of Sum.



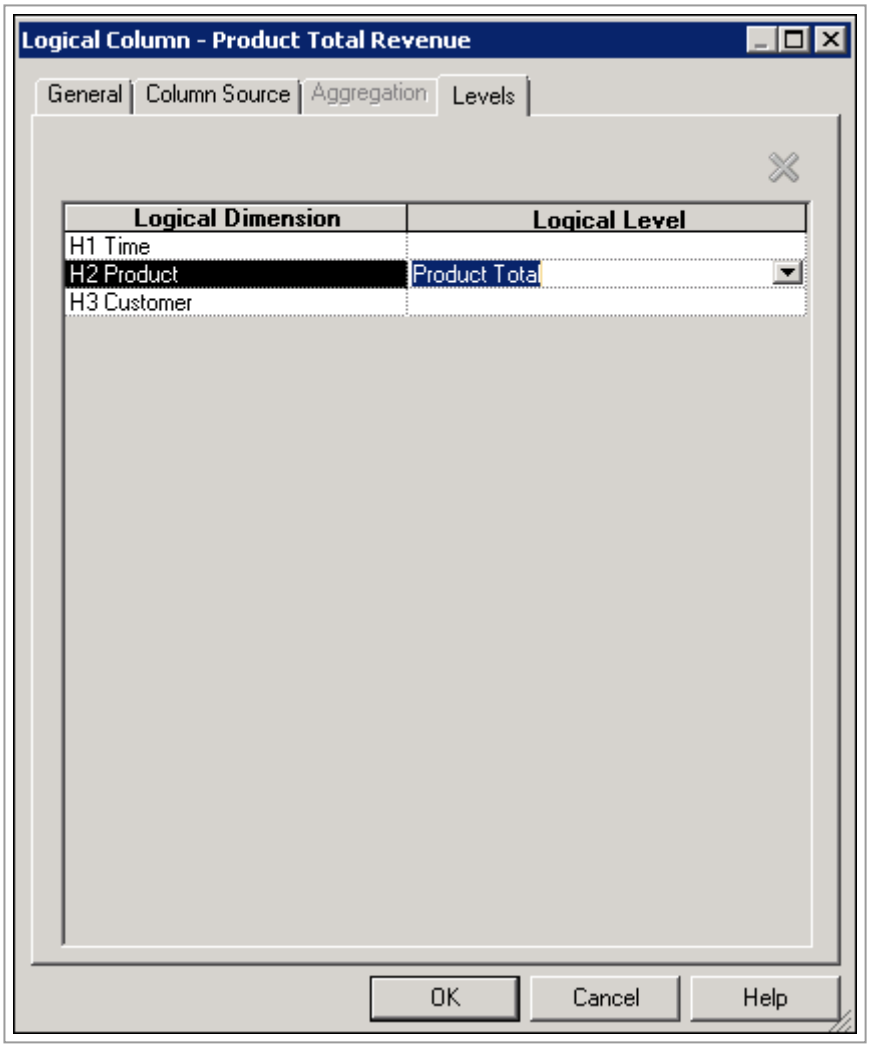



7. Click **OK** to close Expression Builder.

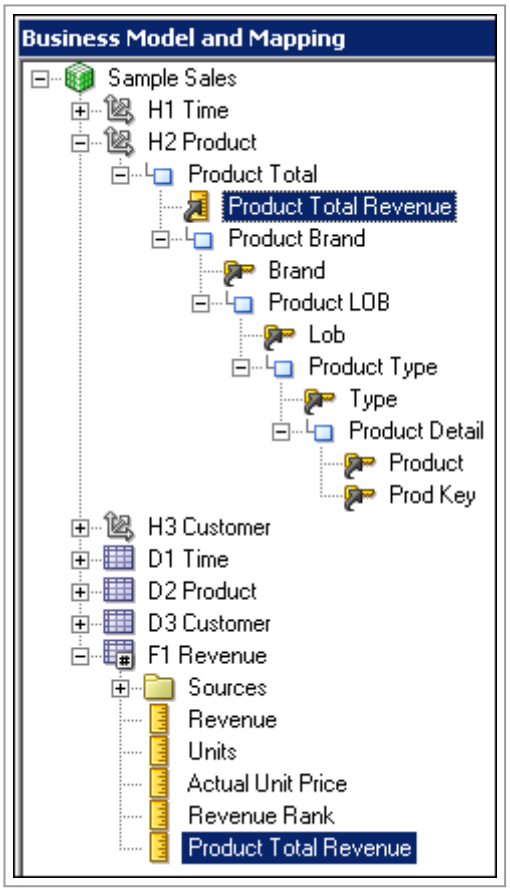
8. Click the **Levels** tab.




9. For the **H2 Product** logical dimension, select **Product Total** from the Logical Level drop-down list to specify that this measure should be calculated at the grand total level in the product hierarchy.

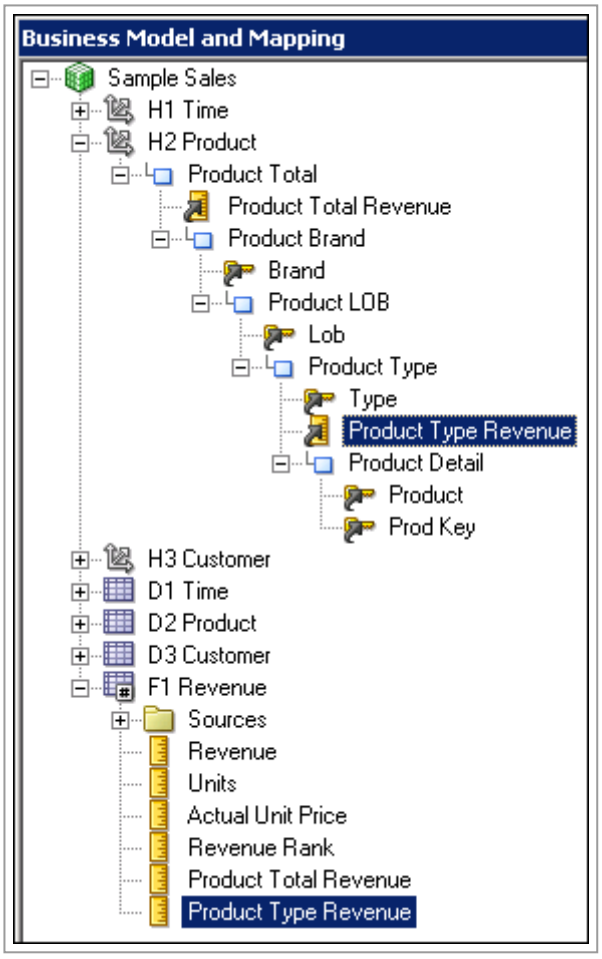



10. Click **OK** to close the Logical Column dialog box. The Product Total Revenue measure appears in the Product Total level of the H2 Product logical dimension and the F1 Revenue logical fact table.

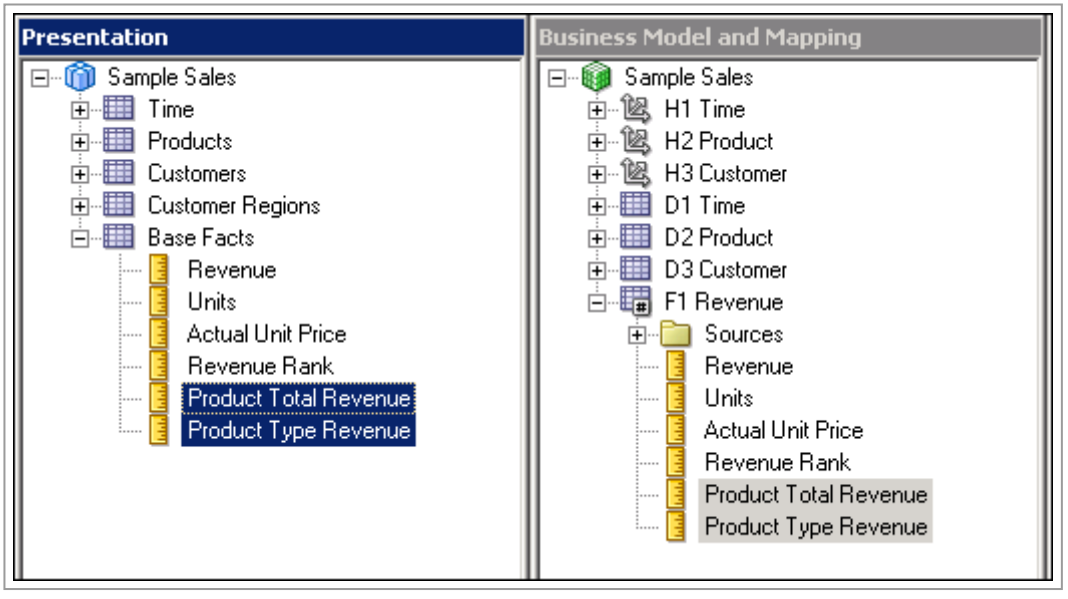



**11.** Repeat the steps to create a second level-based measure:

| **Name** | **Logical Dimension** | **Logical Level** |
|---|---|---|
| Product Type Revenue | H2 Product | Product Type |



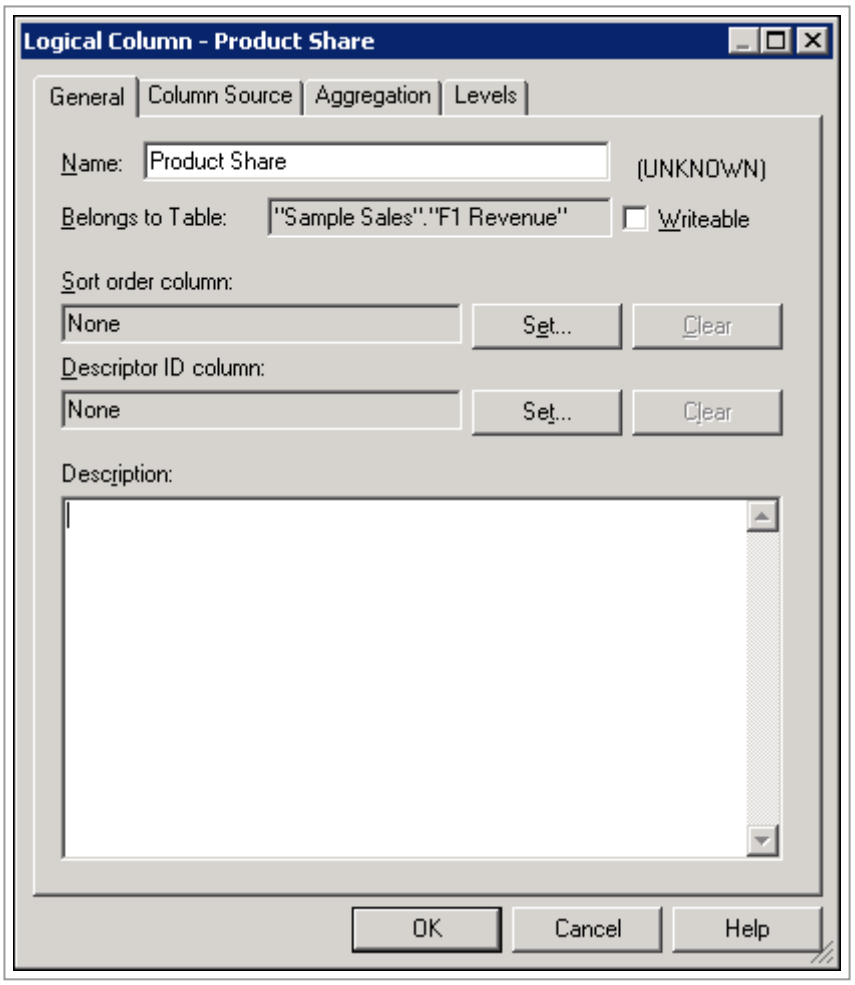

12. Expose the new columns to users by dragging **Product Total Revenue** and **Product Type Revenue** to the **Base Facts** presentation table in the **Sample Sales** subject area in the **Presentation** layer. You can drag the columns from either the H2 Product logical dimension or the F1 Revenue logical table.



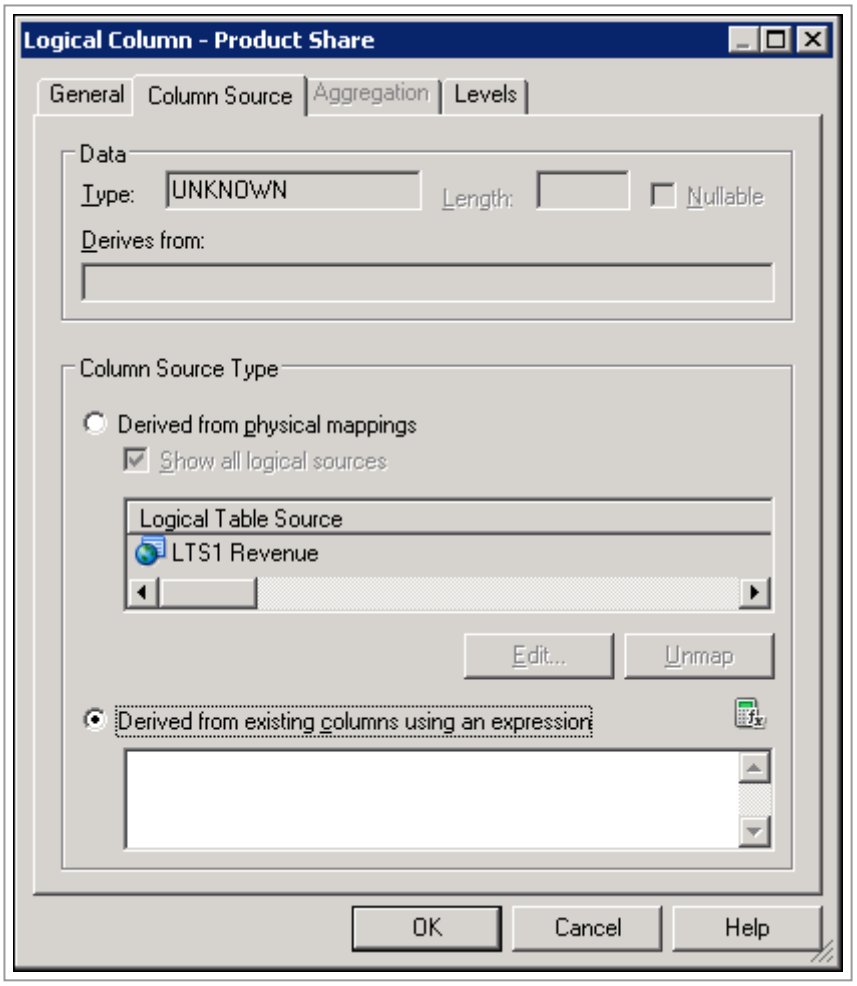

## Creating a Share Measure

1. In the **BMM** layer, right-click the **F1 Revenue** table and select **New Object > Logical Column** to open the Logical Column dialog box.

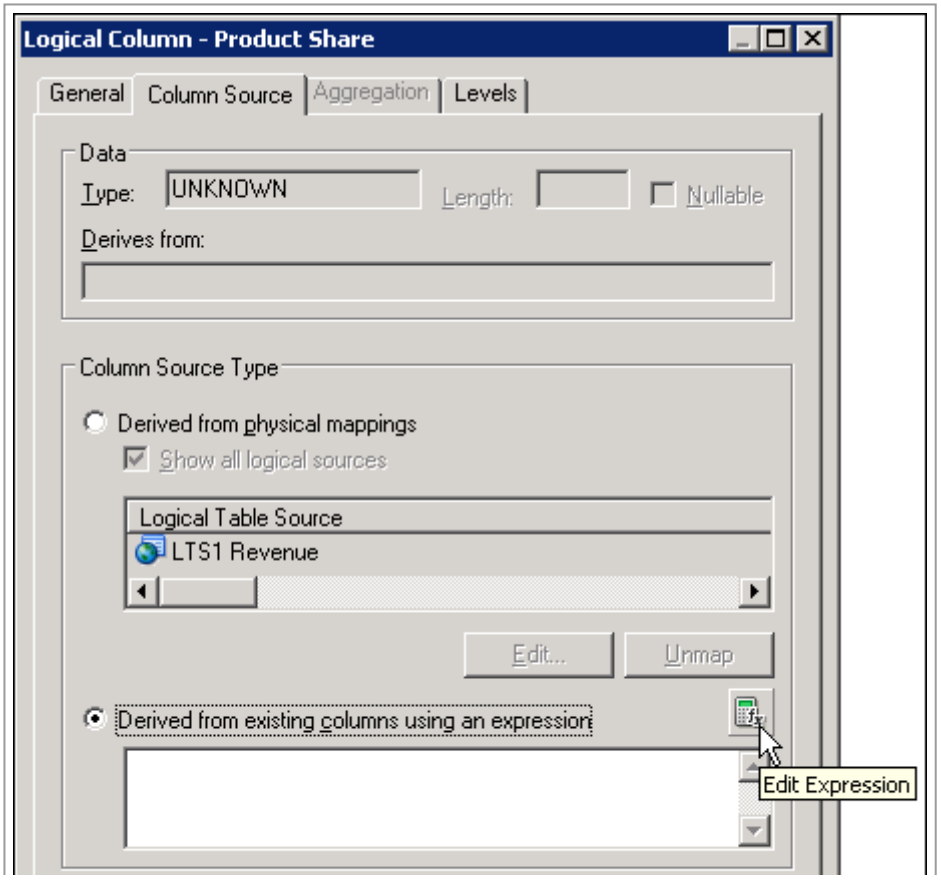
**2.** On the General tab, name the logical column **Product Share**.



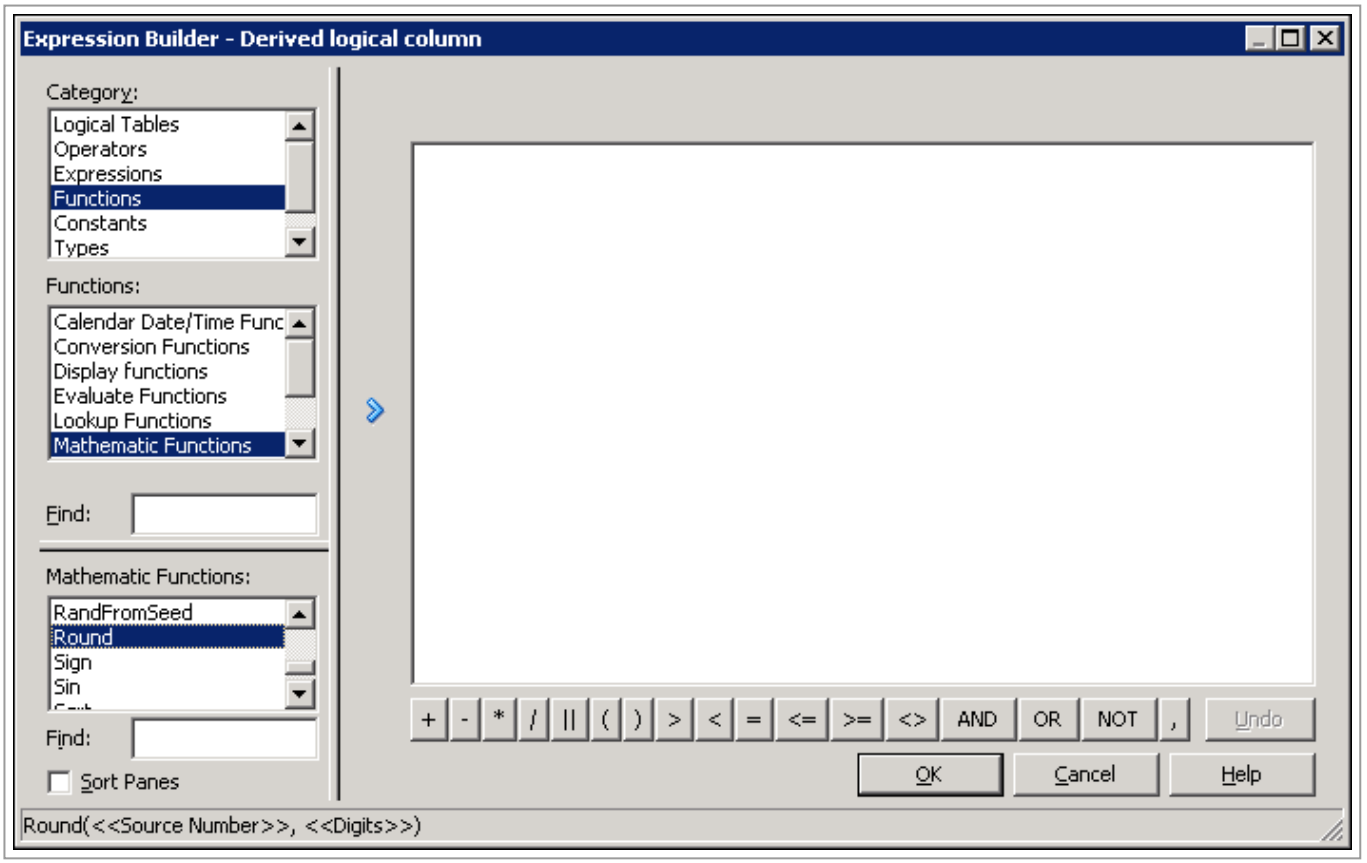

**3.** On the Column Source tab, select **"Derived from existing columns using an expression"**.



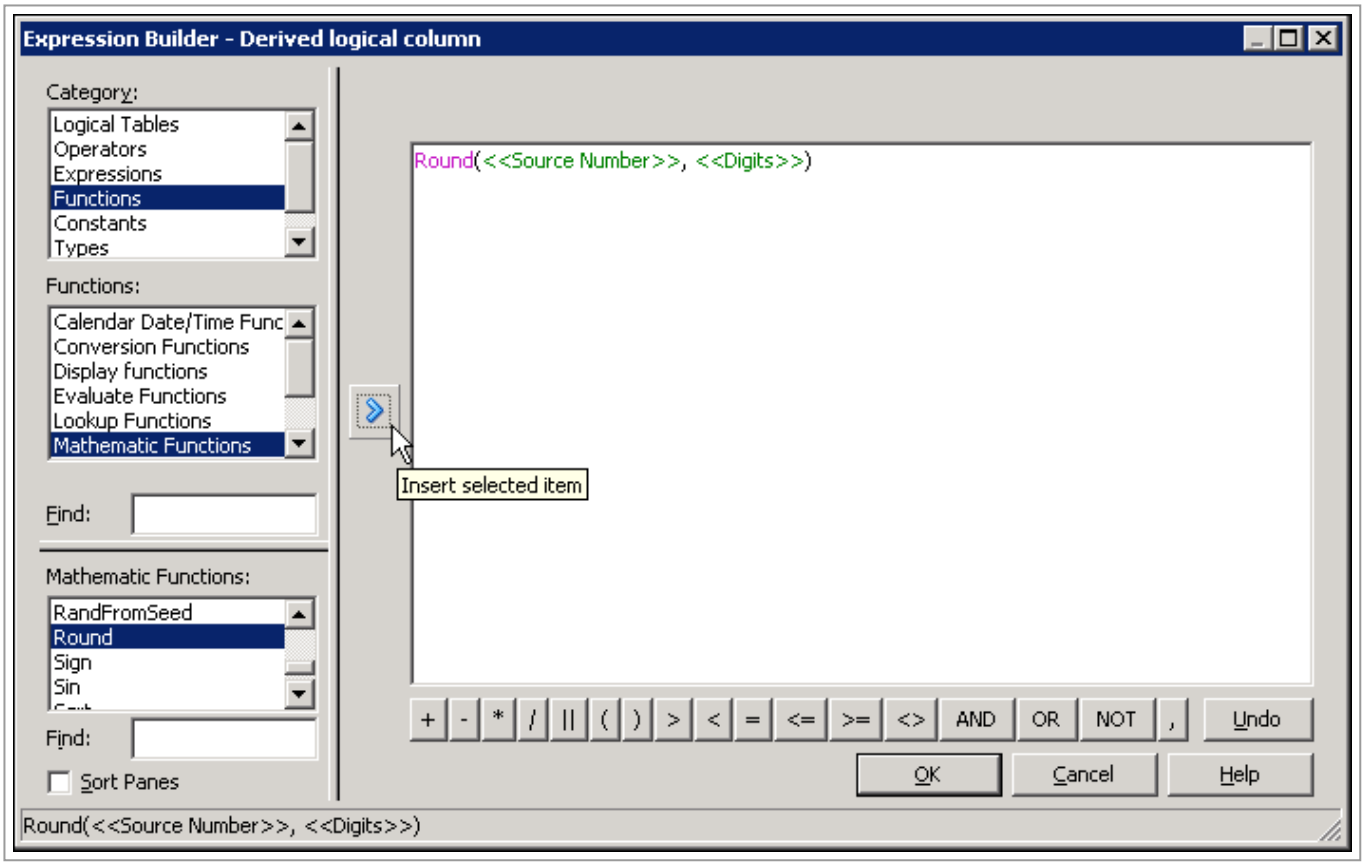

**4.** Open the **Expression Builder**.



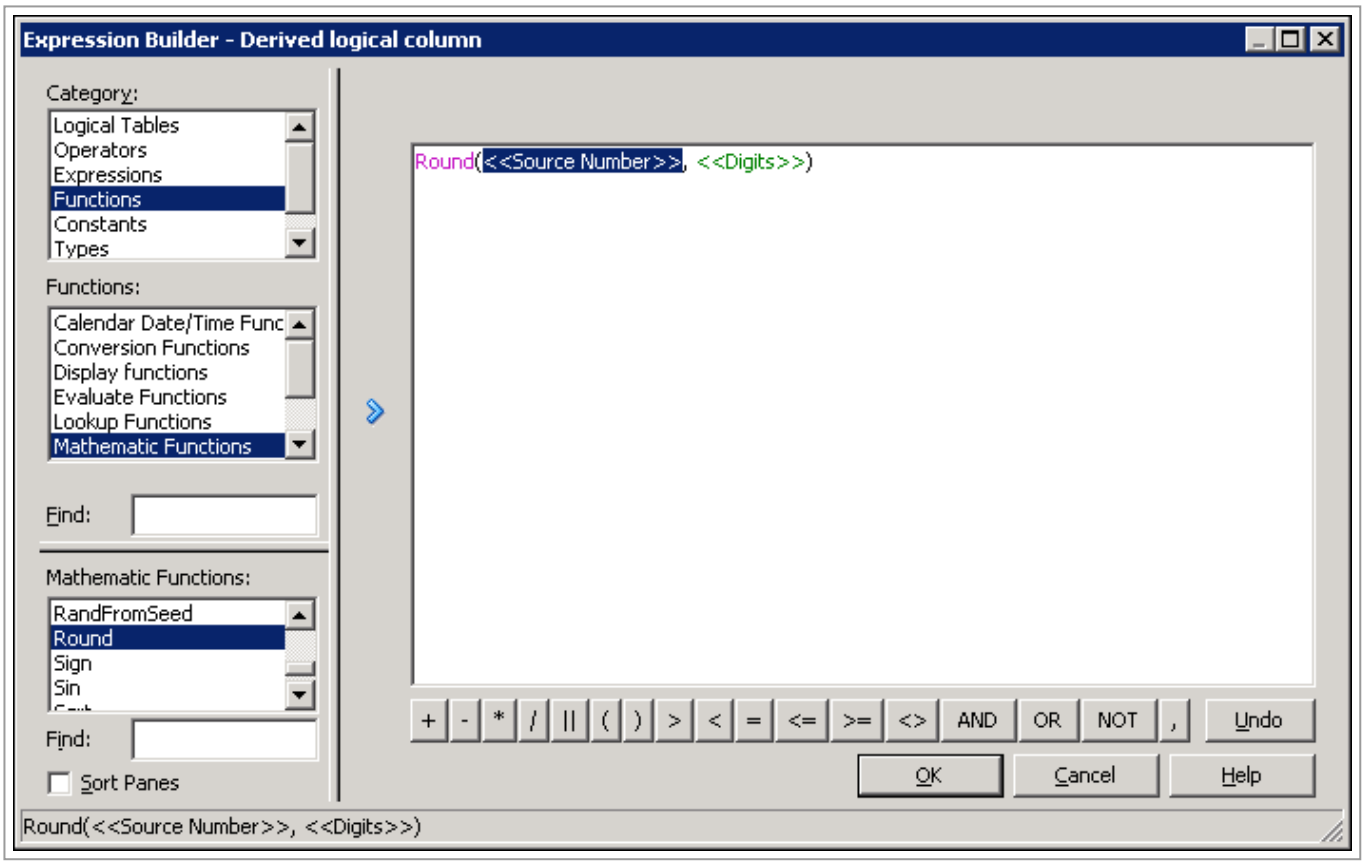

**5.** In the Expression Builder, Select **Functions > Mathematic Functions > Round**.



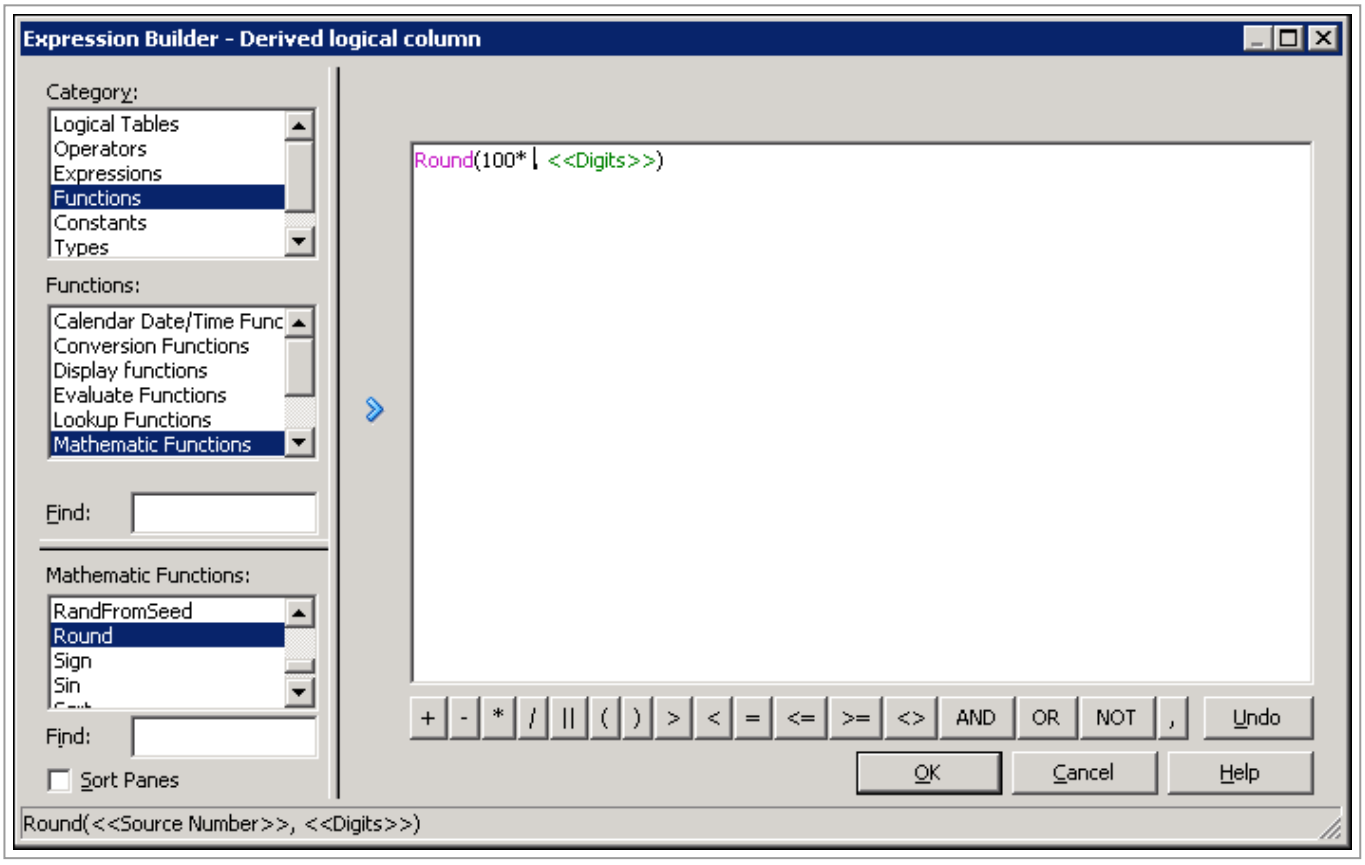

**6.** Click **Insert selected item**. The function appears in the edit box.



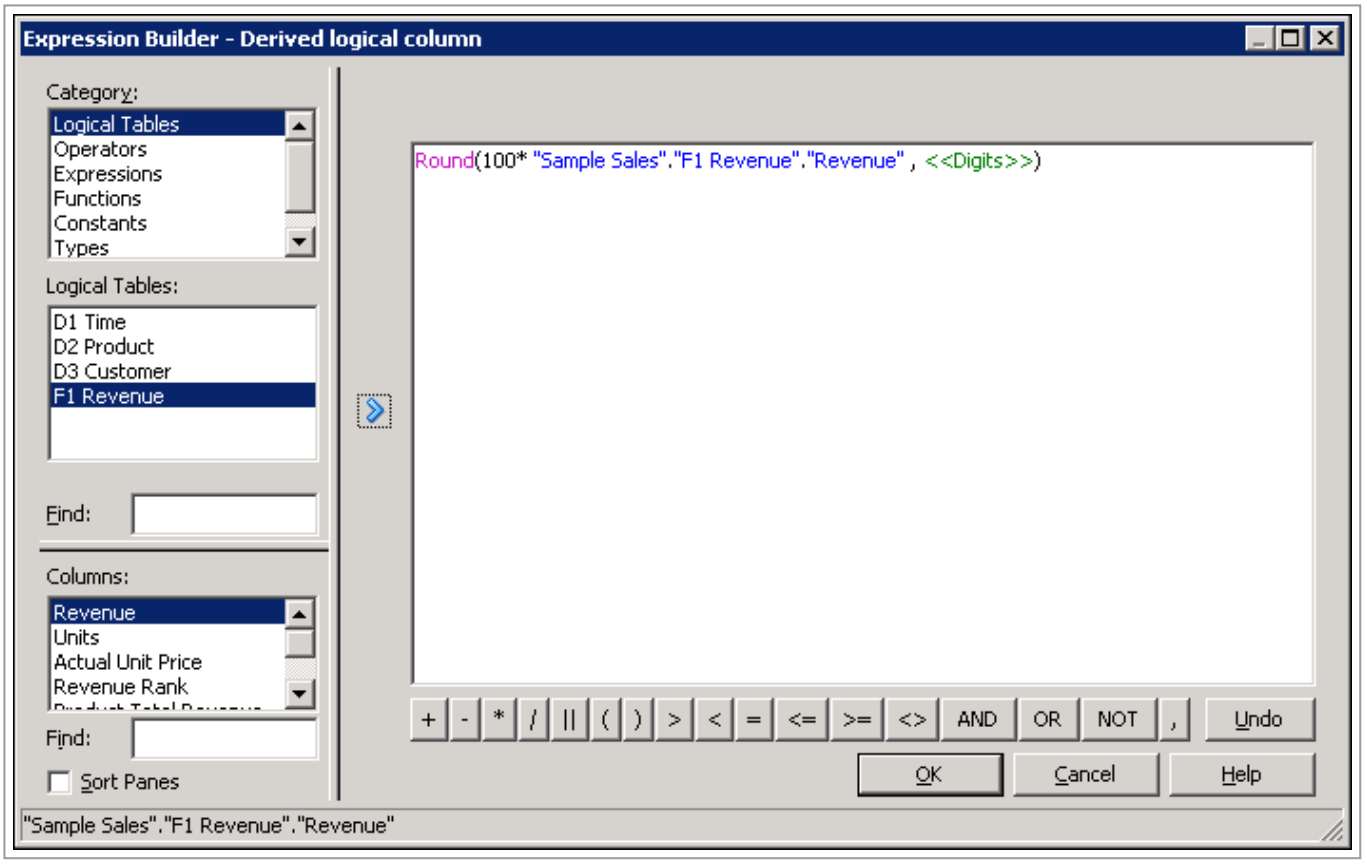
**7.** Click **Source Number** in the formula.



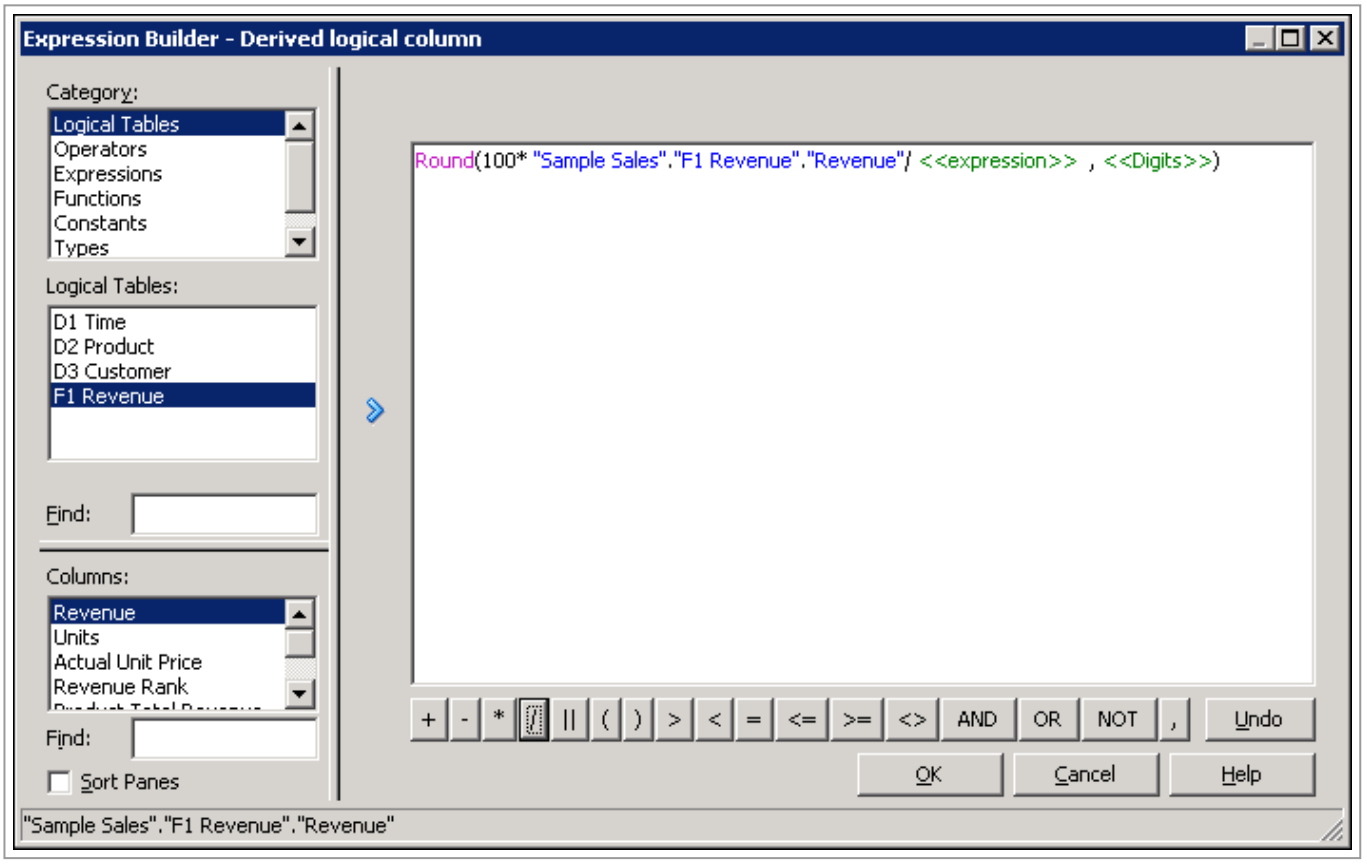

**8.** Enter **100*** followed by a space.



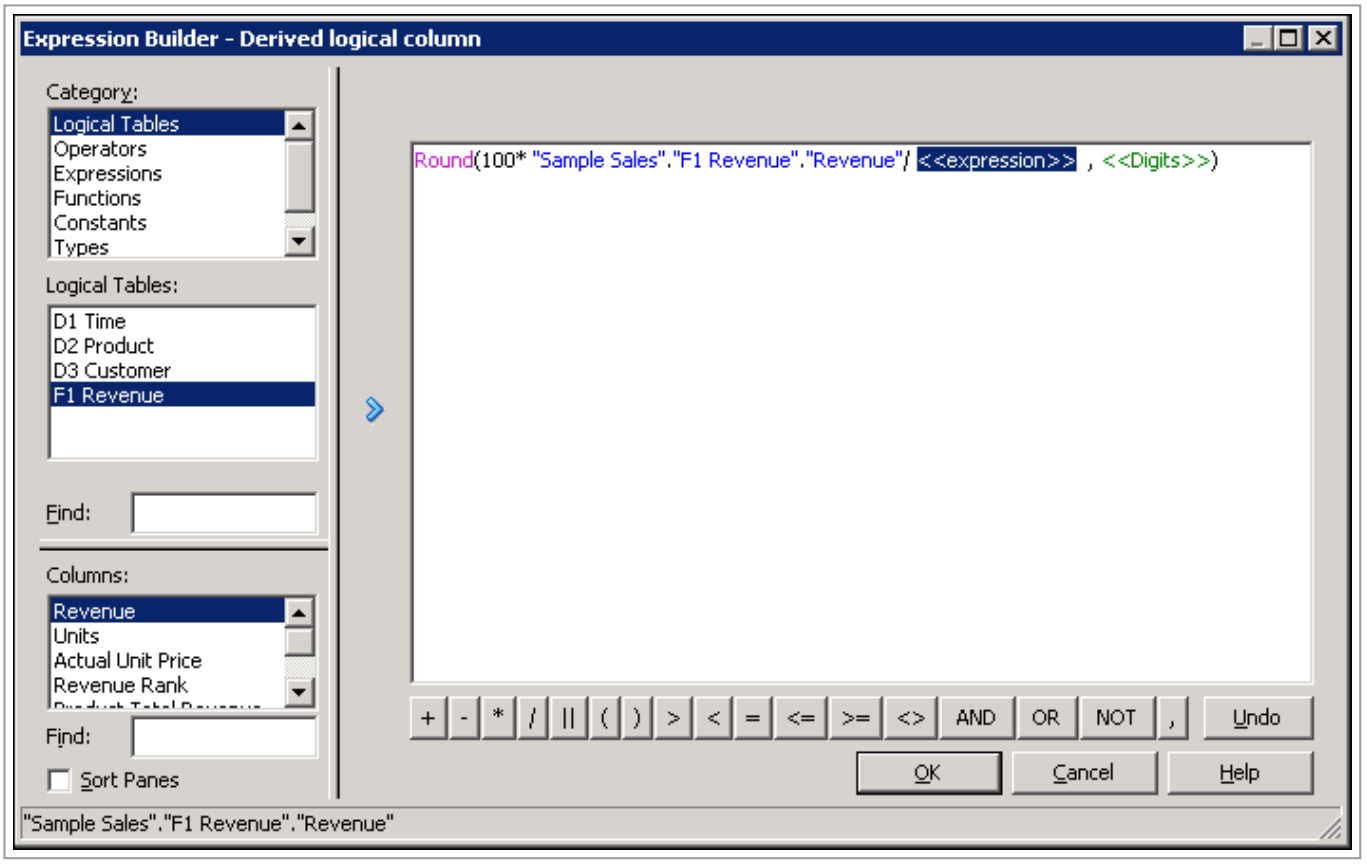

**9.** Insert **Logical Tables > F1 Revenue > Revenue**.



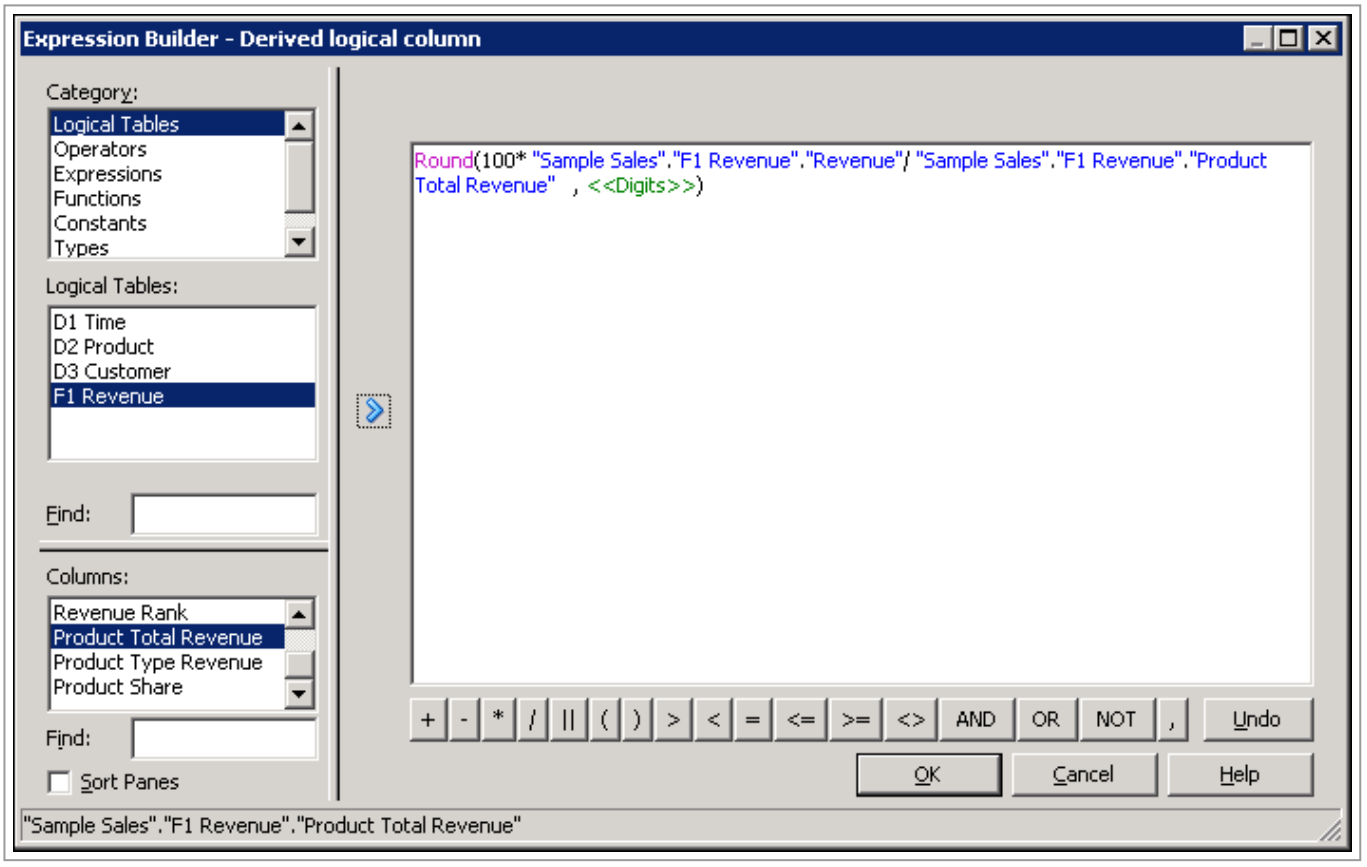

**10.** Using the toolbar, click the **Division button**. Another set of angle brackets appears, <<expr>>.



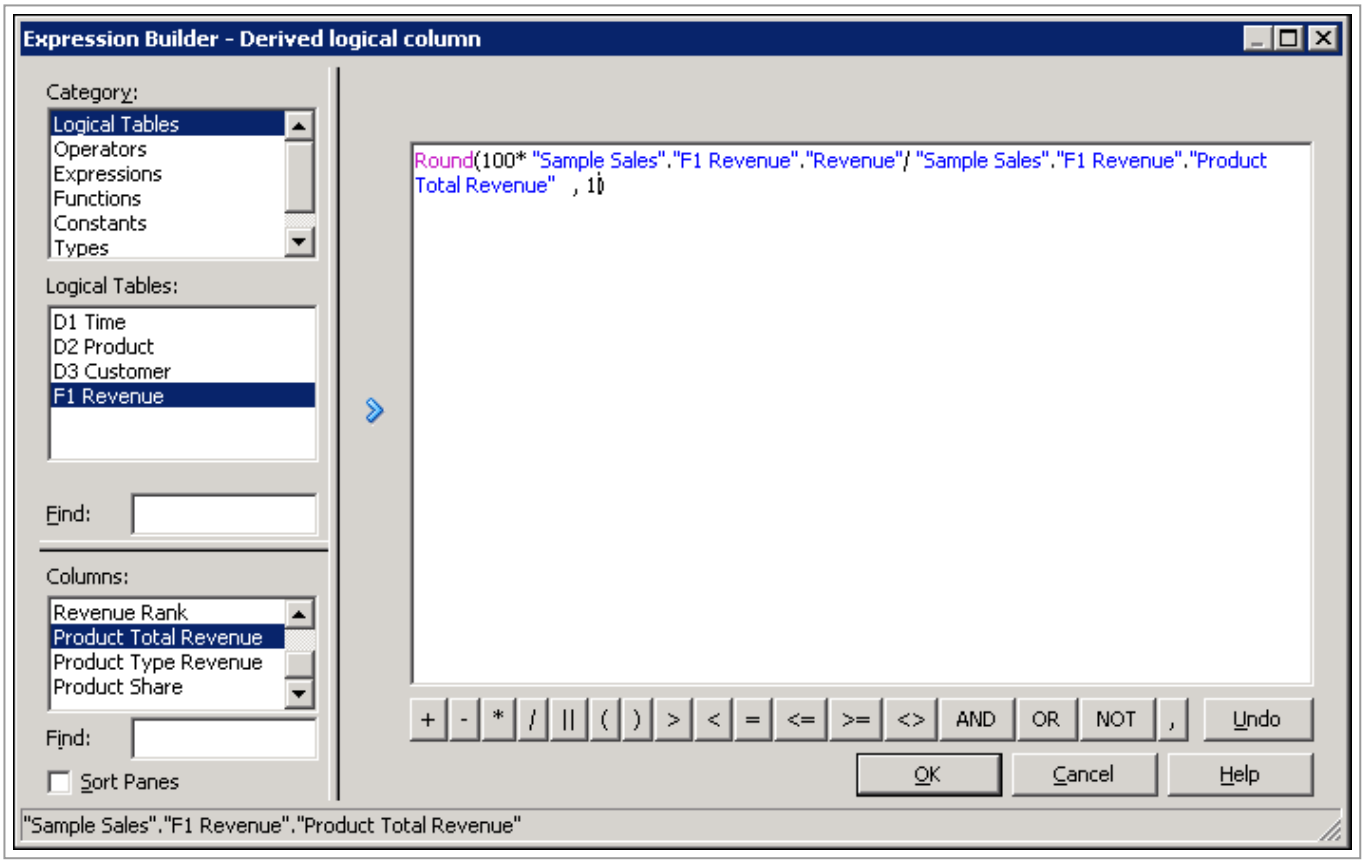

**11.** Click **<<expression>>**.




12. Insert **Logical Tables > F1 Revenue > Product Total Revenue**. Recall that this is the total measure for the hierarchy.




13. Click between the last set of angle brackets, **<<Digits>>**, and enter **1**. This represents the number of digits of precision with which to round the integer.

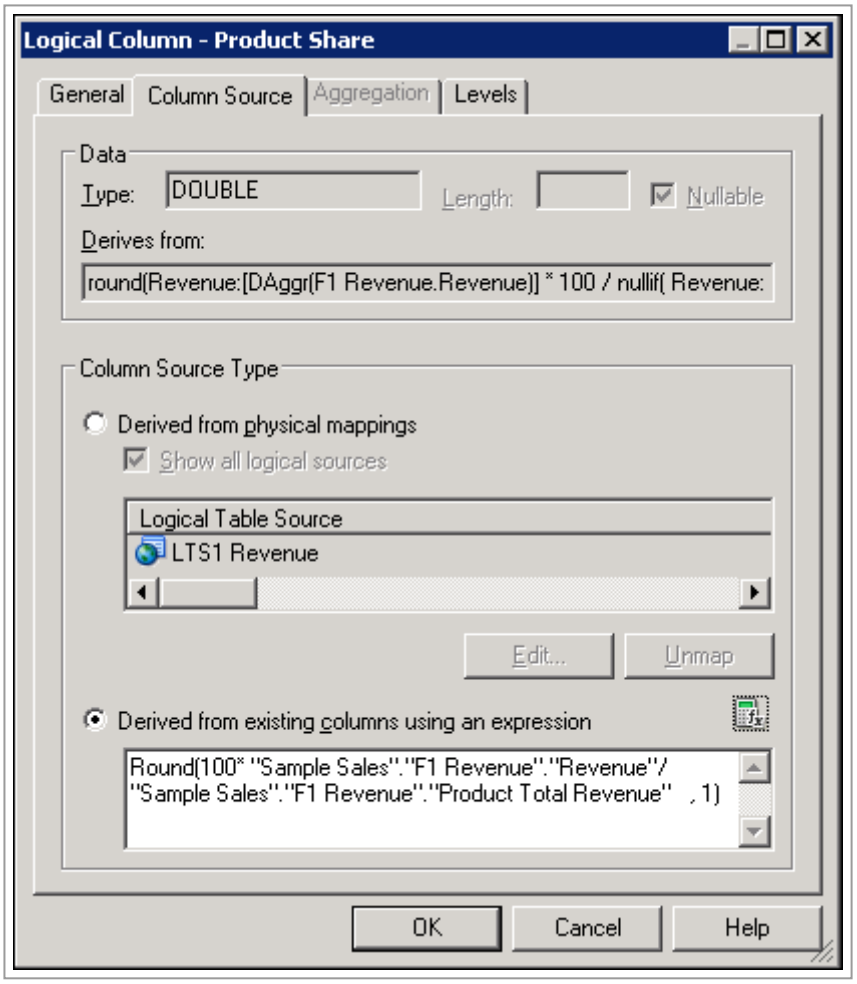



**14.** Check your work:

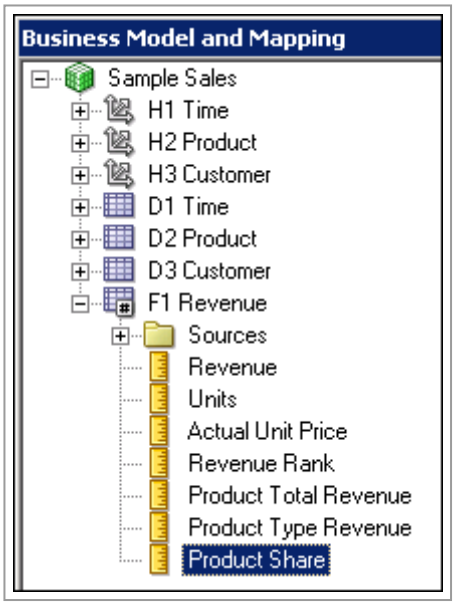
Round(100* "Sample Sales"."F1 Revenue"."Revenue" / "Sample Sales"."F1 Revenue"."Product Total Revenue" , 1)

This share measure will allow you to run an analysis that shows how revenue of a specific product compares to total revenue for all products.

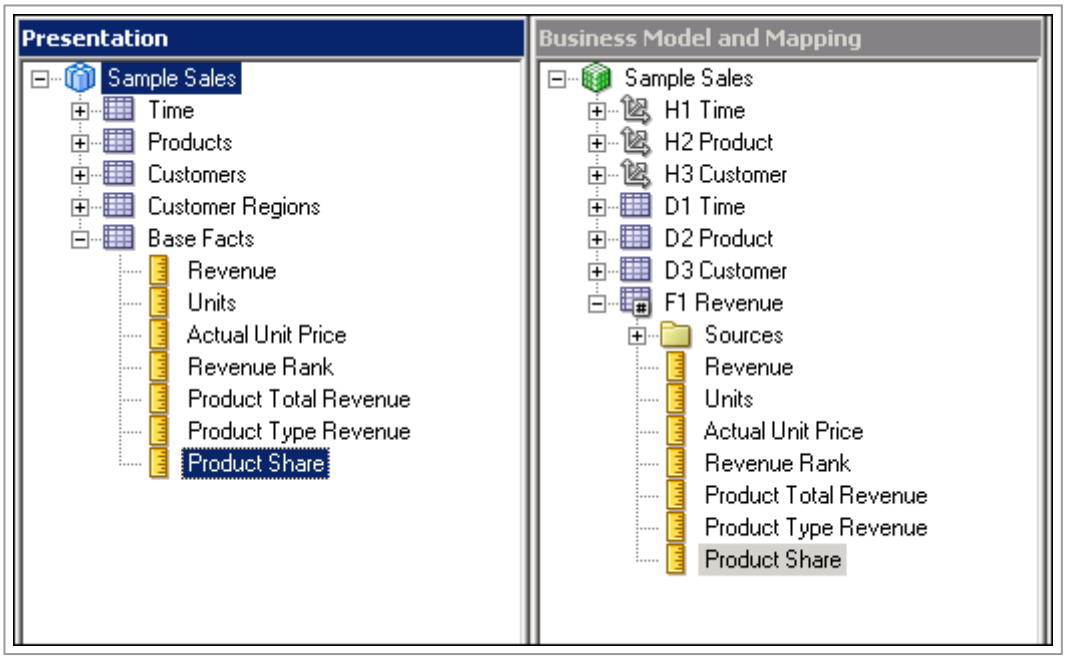
**15.** Click **OK** to close the Expression Builder. The formula is visible in the Logical Column dialog box.



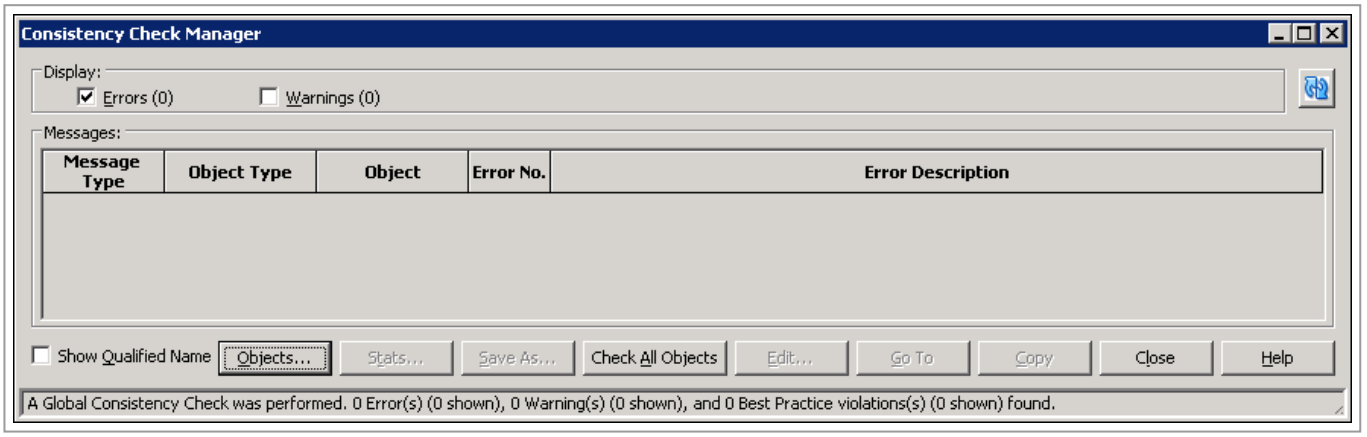

**16.** Click **OK** to close the Logical Column dialog box. The **Product Share** logical column is added to the business model.




**17.** Add the **Product Share** measure to the **Base Facts** presentation table.




18. Save the repository. Check consistency. You should receive the following message.




If there are consistency errors or warnings, correct them before you proceed.

19. Close the repository.

## Testing Your Work

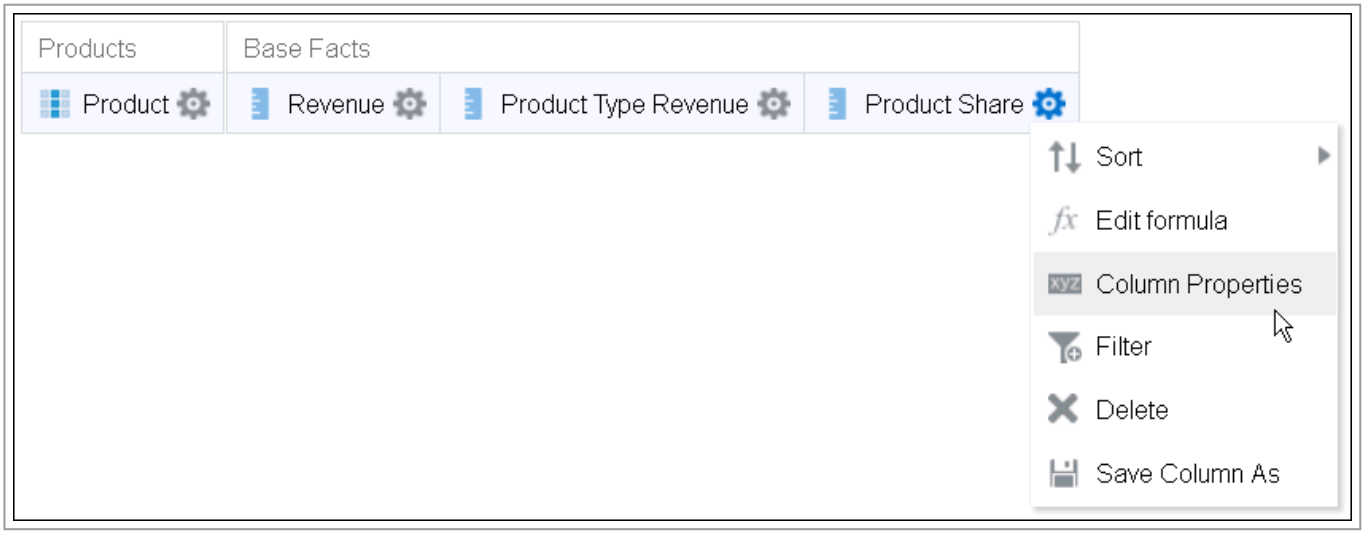
1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

2. Return to **Oracle BI**, which should still be open, and sign in.

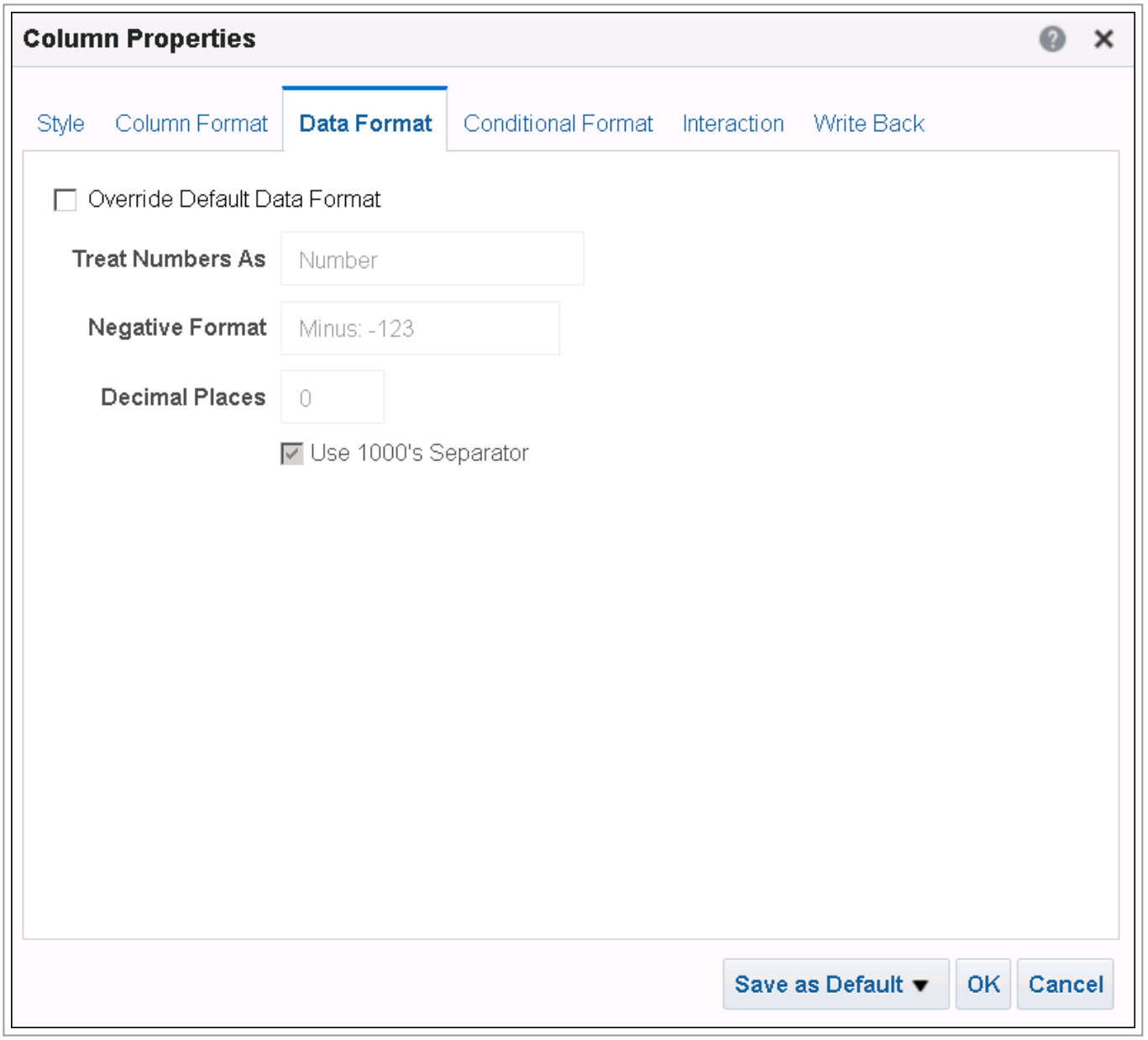
3. Create the following analysis to test the level-based and share measures.

   **Products.Product**
   **Base Facts.Revenue**
   **Base Facts.Product Type Revenue**
   **Base Facts.Product Share**

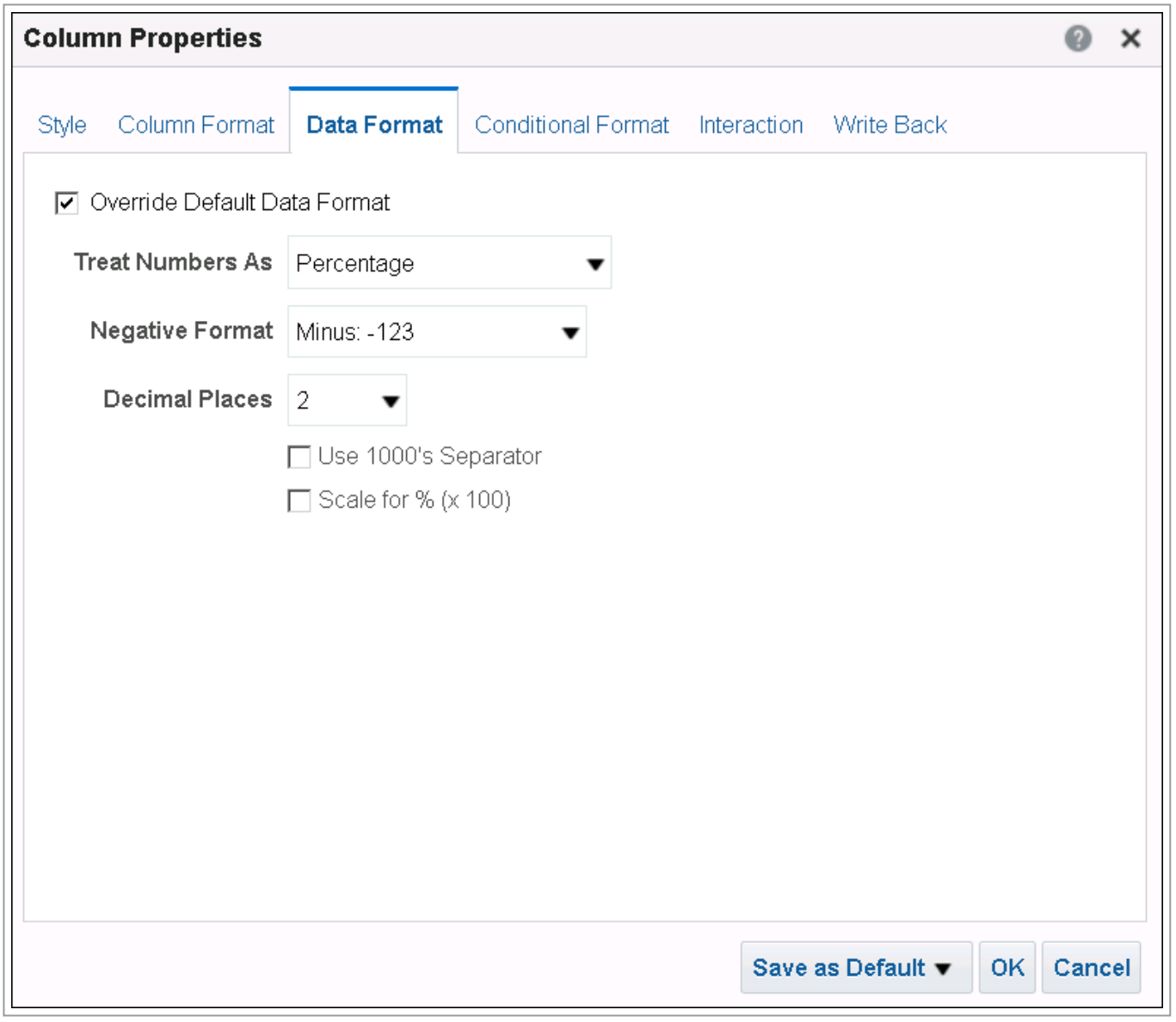

**4.** For the Product Share column, select **Column Properties**.




**5.** On the Data Format tab, select **Override Default Data Format**.

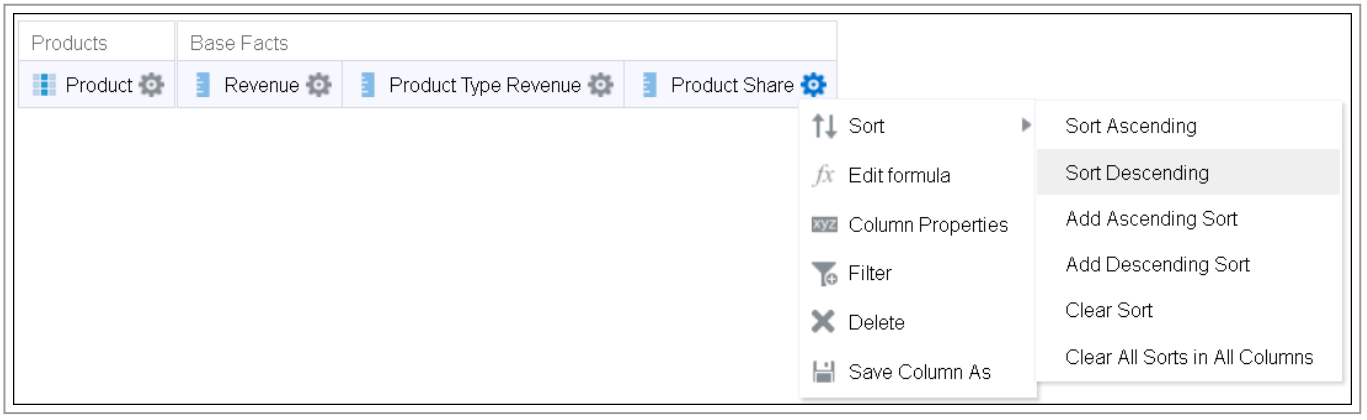



6. Change **Treat Numbers As** to **Percentage** and set **Decimal Places** to **2**. Deselect **Use 1000's separator**.

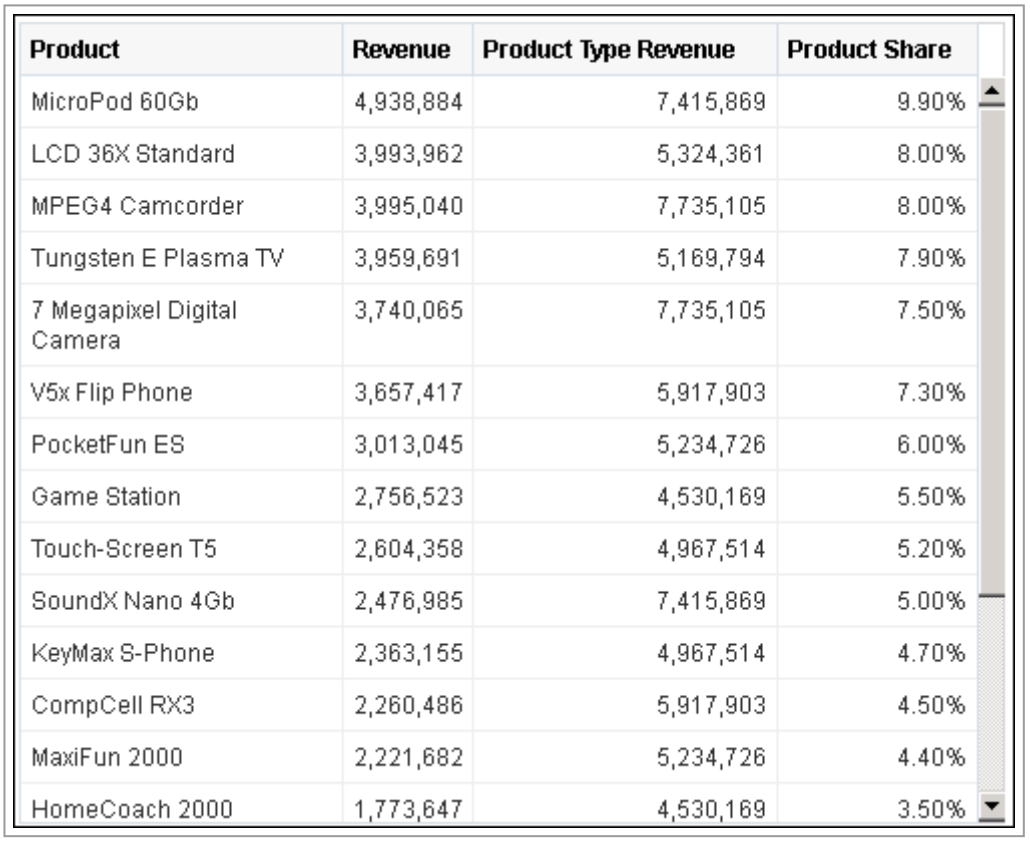



7. Click **OK** to close the Column Properties dialog box.

8. Sort **Product Share** in descending order.




9. Click **Results**. Notice that Product Type Revenue returns dollars grouped by Type even though the query is at a different level than Type; Product in this example. Product Share shows the percent of total revenue for each product sorted in descending order.



| Product | Revenue | Product Type Revenue | Product Share |
|---|---|---|---|
| MicroPod 60Gb | 4,938,884 | 7,415,869 | 9.90% |
| LCD 36X Standard | 3,993,962 | 5,324,361 | 8.00% |
| MPEG4 Camcorder | 3,995,040 | 7,735,105 | 8.00% |
| Tungsten E Plasma TV | 3,959,691 | 5,169,794 | 7.90% |
| 7 Megapixel Digital Camera | 3,740,065 | 7,735,105 | 7.50% |
| V5x Flip Phone | 3,657,417 | 5,917,903 | 7.30% |
| PocketFun ES | 3,013,045 | 5,234,726 | 6.00% |
| Game Station | 2,756,523 | 4,530,169 | 5.50% |
| Touch-Screen T5 | 2,604,358 | 4,967,514 | 5.20% |
| SoundX Nano 4Gb | 2,476,985 | 7,415,869 | 5.00% |
| KeyMax S-Phone | 2,363,155 | 4,967,514 | 4.70% |
| CompCell RX3 | 2,260,486 | 5,917,903 | 4.50% |
| MaxiFun 2000 | 2,221,682 | 5,234,726 | 4.40% |
| HomeCoach 2000 | 1,773,647 | 4,530,169 | 3.50% |

10. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Creating Logical Dimensions with Parent-Child Hierarchies

A parent-child hierarchy is a hierarchy of members that all have the same type. This contrasts with level-based hierarchies, where members of the same type occur only at a single level of the hierarchy. The most common real-life occurrence of a parent-child hierarchy is an organizational reporting hierarchy chart, where the following all apply:
• Each individual in the organization is an employee.
• Each employee, apart from the top-level managers, reports to a single manager.
• The reporting hierarchy has many levels.
In relational tables, the relationships between different members in a parent-child hierarchy are implicitly defined by the identifier key values in the associated base table. However, for each Oracle BI Server parent-child hierarchy defined on a relational table, you must also explicitly define the inter-member relationships in a separate parent-child relationship table.

To create a logical dimension with a parent-child hierarchy, perform the following steps:

- Opening the Repository in Offline Mode
- Importing Metadata and Define Physical Layer Objects
- Creating Logical Table and Logical Columns
- Creating a Logical Join
- Creating a Parent-Child Logical Dimension

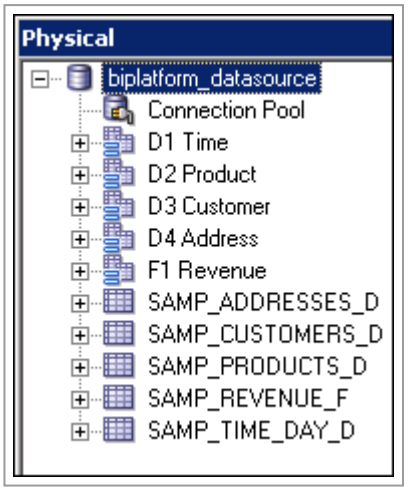
- Defining Parent-Child Settings
- Creating Presentation Layer Objects
- Testing Your Work

## Opening the Repository in Offline Mode

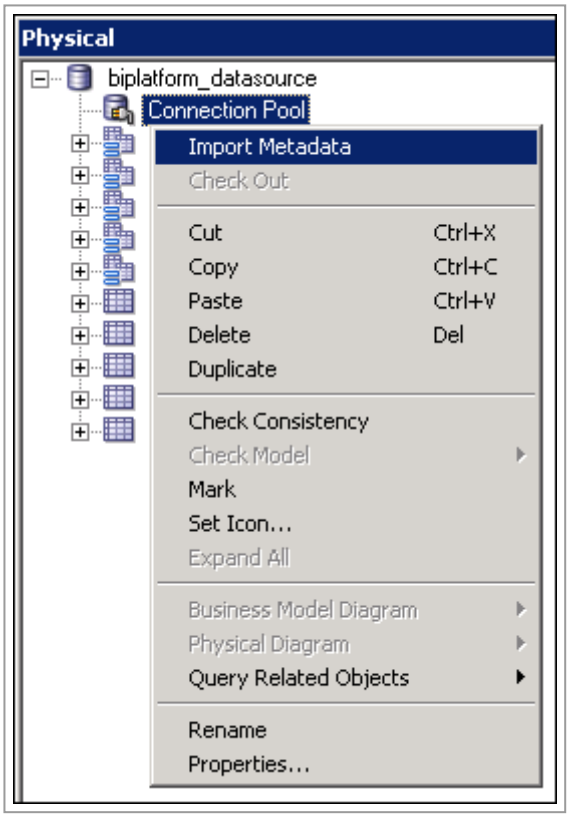
1. Return to the Administration Tool, which should still be open.
2. Select **File > Open > Offline**.
3. Select **BISAMPLE.rpd** and click **Open**.
4. Enter **Admin123** as the repository password and click **OK** to open the repository.

## Importing Metadata and Define Physical Layer Objects

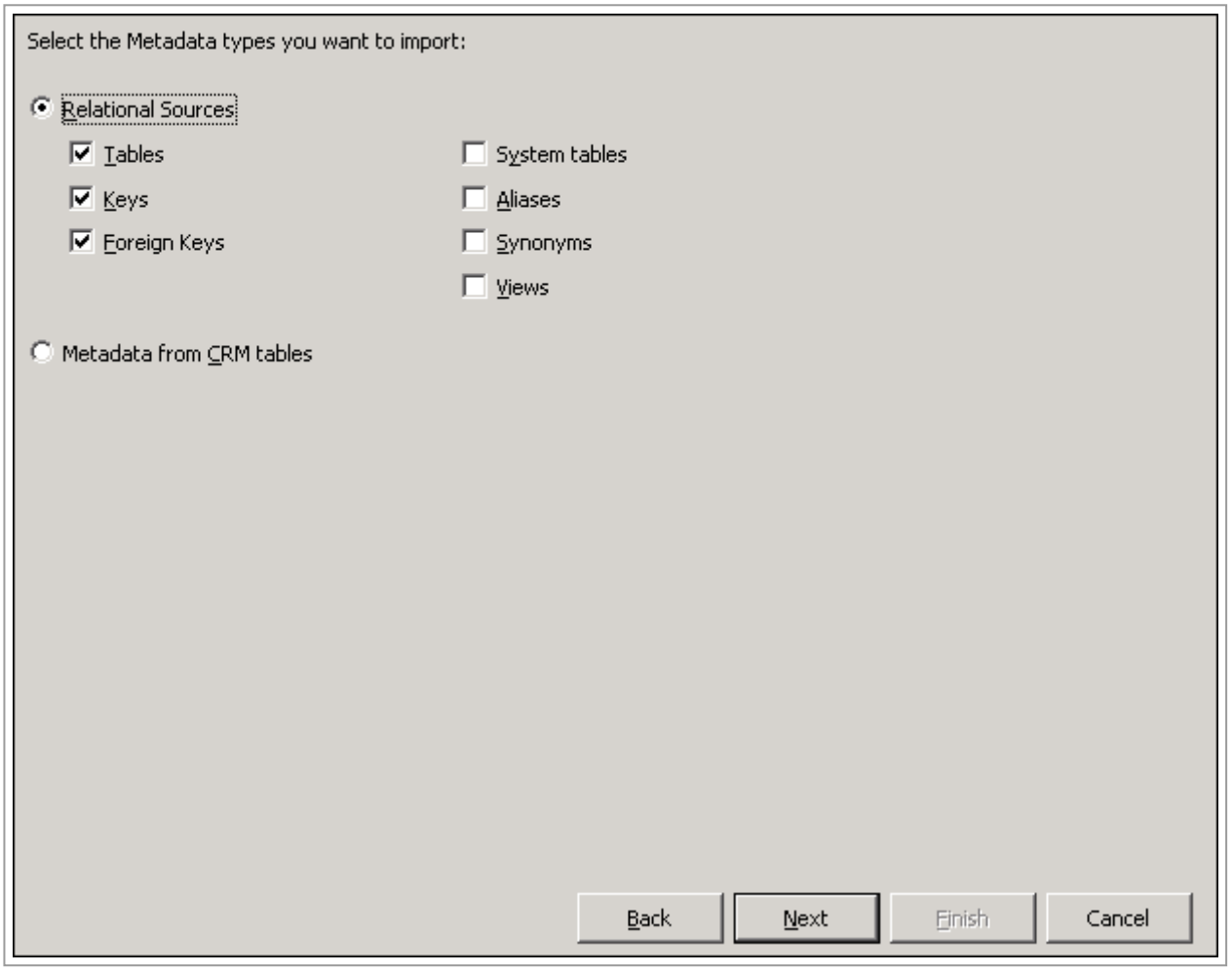
1. In the **Physical** layer, expand **biplatform_datasource**.




2. Right-click **Connection Pool** and select **Import Metadata** to open the Import Wizard.



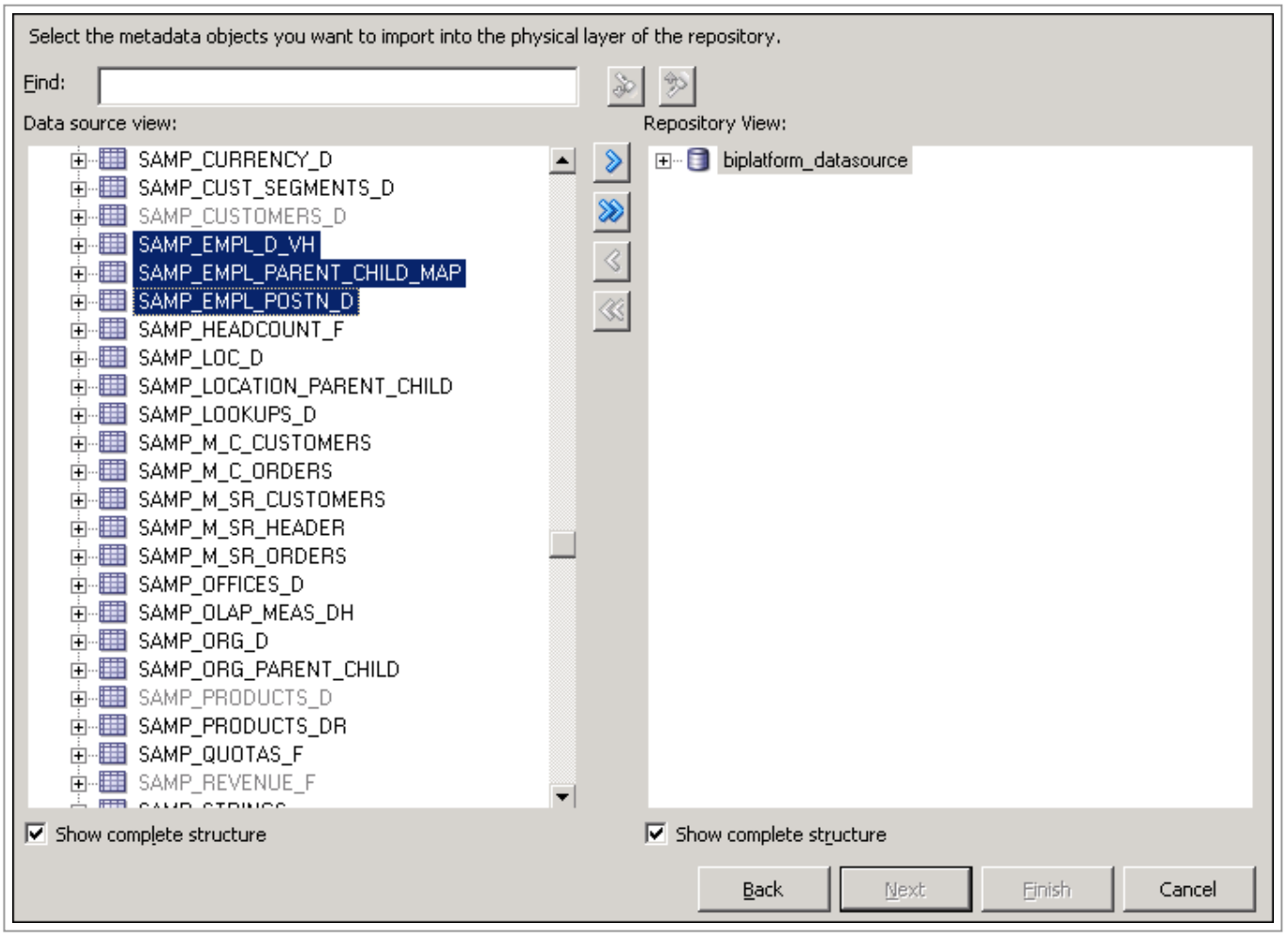

**3.** In the Select Metadata Types screen, accept the defaults and click **Next**.



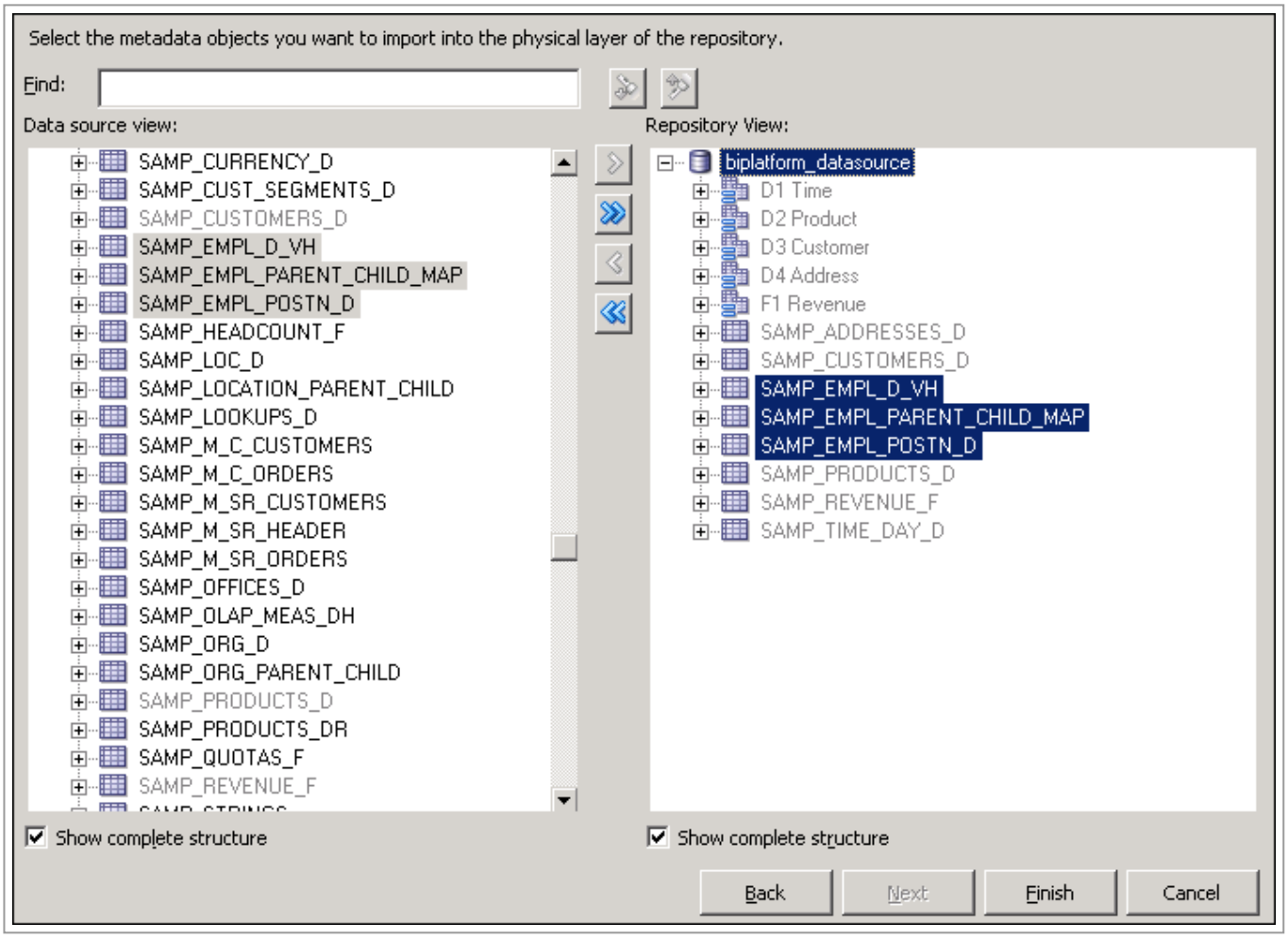

**4.** In the Select Metadata Objects screen, in the data source view, expand**biplatform_datasource** and

4. In the Select Metadata Objects screen, in the data source view, expand **biplatform_datasource** and select the following tables for import:

**SAMP_EMPL_D_VH**
**SAMP_EMPL_PARENT_CHILD_MAP**
**SAMP_EMPL_POSTN_D**

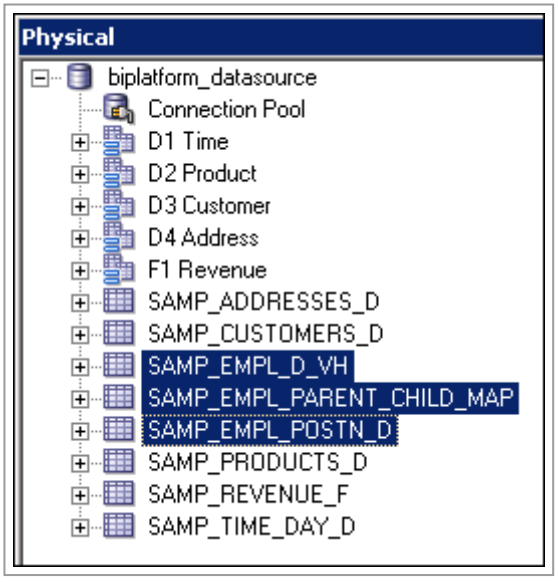



5. Click the **Import Selected** button to move the tables to the Repository View.

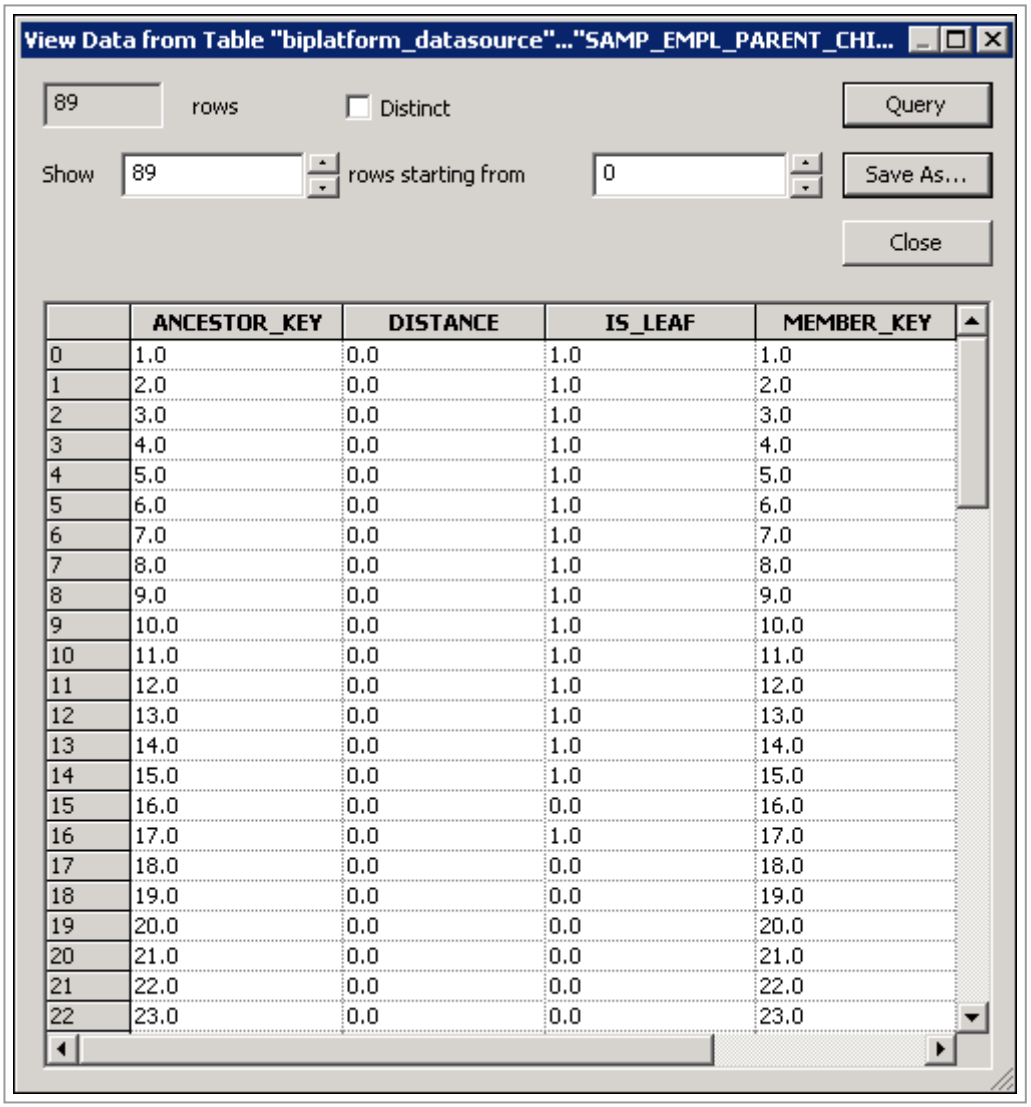



**6.** Click **Finish** to close the Import Wizard.

**7.** Confirm that the three tables are visible in the **Physical** layer of the repository.




**8.** Right-click **SAMP_EMPL_PARENT_CHILD_MAP** and select **View Data**.

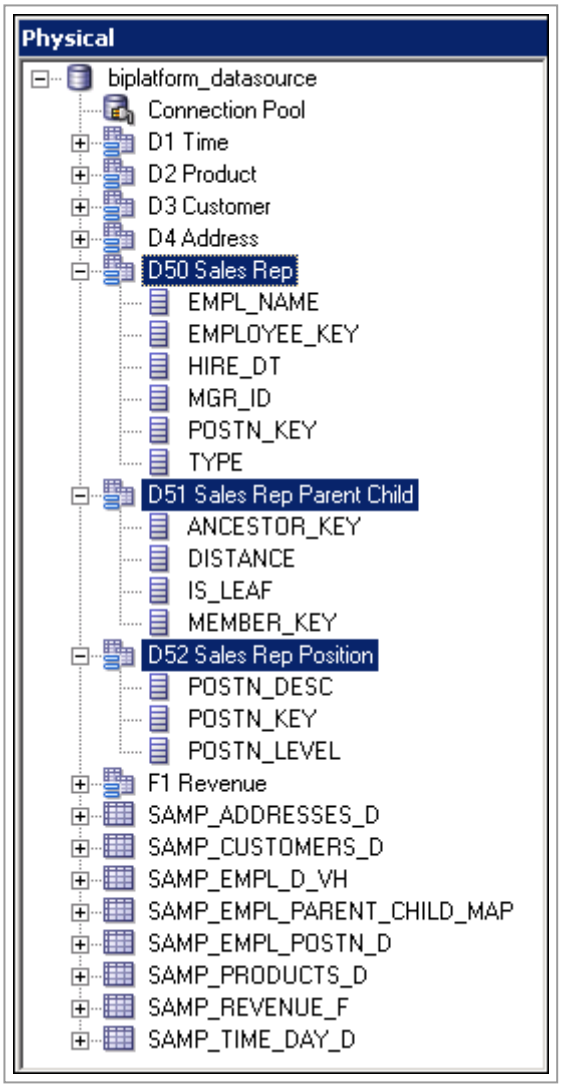



| | ANCESTOR_KEY | DISTANCE | IS_LEAF | MEMBER_KEY |
|---|---|---|---|---|
| 0 | 1.0 | 0.0 | 1.0 | 1.0 |
| 1 | 2.0 | 0.0 | 1.0 | 2.0 |
| 2 | 3.0 | 0.0 | 1.0 | 3.0 |
| 3 | 4.0 | 0.0 | 1.0 | 4.0 |
| 4 | 5.0 | 0.0 | 1.0 | 5.0 |
| 5 | 6.0 | 0.0 | 1.0 | 6.0 |
| 6 | 7.0 | 0.0 | 1.0 | 7.0 |
| 7 | 8.0 | 0.0 | 1.0 | 8.0 |
| 8 | 9.0 | 0.0 | 1.0 | 9.0 |
| 9 | 10.0 | 0.0 | 1.0 | 10.0 |
| 10 | 11.0 | 0.0 | 1.0 | 11.0 |
| 11 | 12.0 | 0.0 | 1.0 | 12.0 |
| 12 | 13.0 | 0.0 | 1.0 | 13.0 |
| 13 | 14.0 | 0.0 | 1.0 | 14.0 |
| 14 | 15.0 | 0.0 | 1.0 | 15.0 |
| 15 | 16.0 | 0.0 | 0.0 | 16.0 |
| 16 | 17.0 | 0.0 | 1.0 | 17.0 |
| 17 | 18.0 | 0.0 | 0.0 | 18.0 |
| 18 | 19.0 | 0.0 | 0.0 | 19.0 |
| 19 | 20.0 | 0.0 | 0.0 | 20.0 |
| 20 | 21.0 | 0.0 | 0.0 | 21.0 |
| 21 | 22.0 | 0.0 | 0.0 | 22.0 |
| 22 | 23.0 | 0.0 | 0.0 | 23.0 |

This is an example of a parent-child relationship table with rows that define the inter-member relationships of an employee hierarchy. It includes a Member Key column, which identifies the member (employee); an Ancestor Key, which identifies the ancestor (manager) of the member; a Distance column, which specifies the number of parent-child hierarchy levels from the member to the ancestor; and a Leaf column, which indicates if the member is a leaf member.

**9.** Create the following aliases for the tables:

| **Table** | **Alias** |
|---|---|
| SAMP_EMPL_D_VH | D50 Sales Rep |
| SAMP_EMPL_PARENT_CHILD_MAP | D51 Sales Rep Parent Child |
| SAMP_EMPL_POSTN_D | D52 Sales Rep Position |

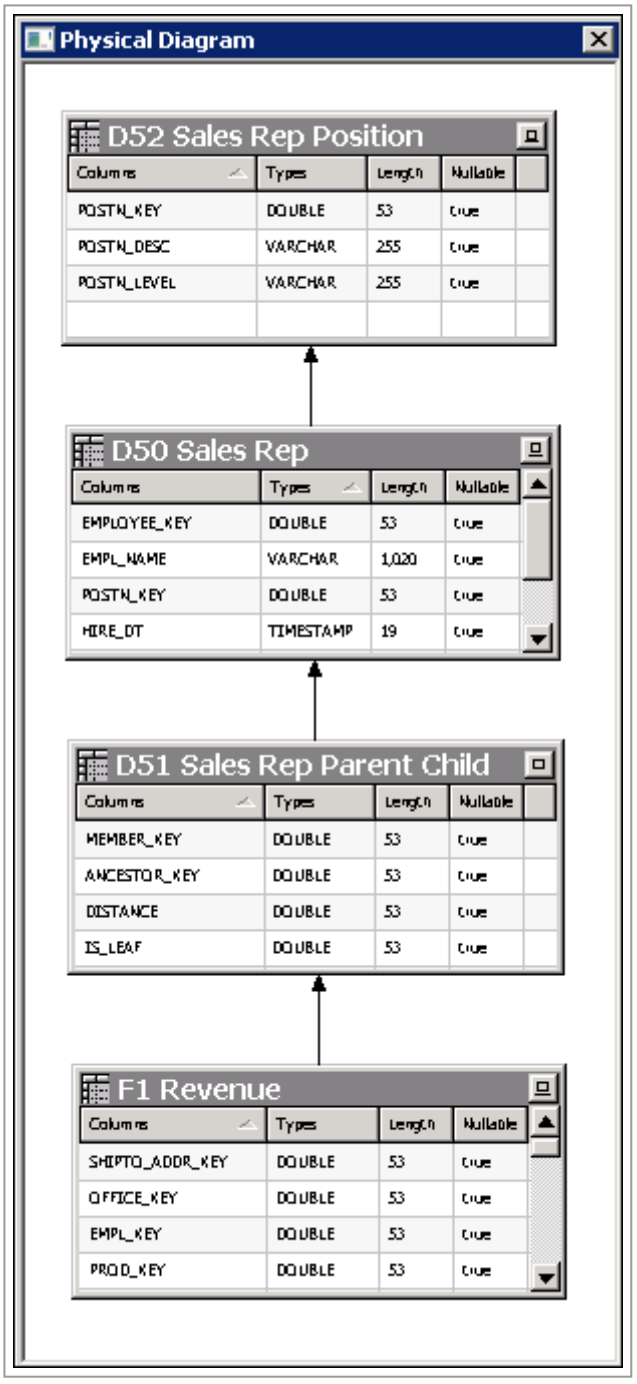



**10.** Use the Physical Diagram to create the following physical joins for the alias tables:

"biplatform_datasource"."".""."D52 Sales Rep Position"."POSTN_KEY" = "biplatform_datasource"."".""."D50 Sales Rep"."POSTN_KEY"

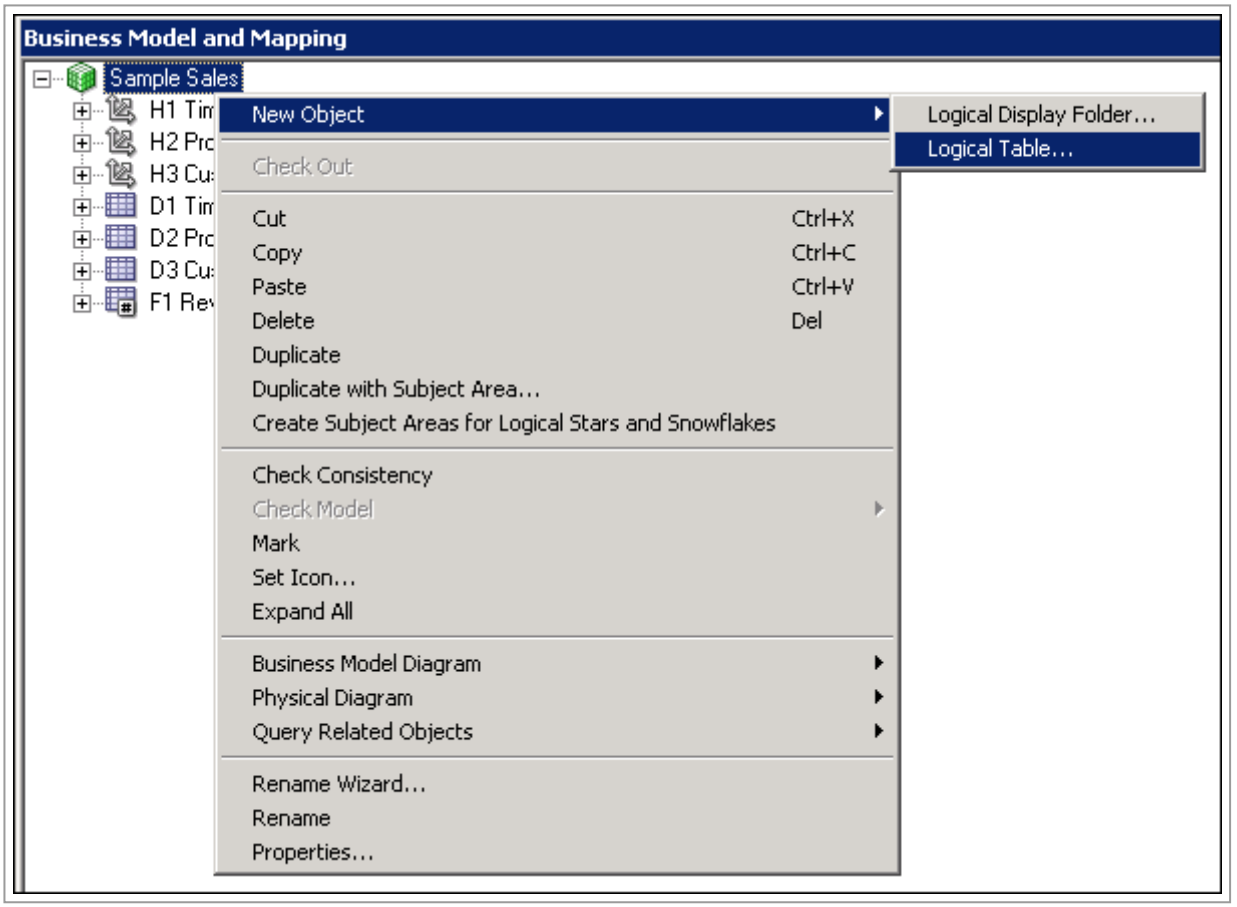
"biplatform_datasource"."".""."D50 Sales Rep"."EMPLOYEE_KEY" = "biplatform_datasource"."".""."D51 Sales Rep Parent Child"."ANCESTOR_KEY"

"biplatform_datasource"."".""."D51 Sales Rep Parent Child"."MEMBER_KEY" = "biplatform_datasource"."".""."F1 Revenue"."EMPL_KEY"

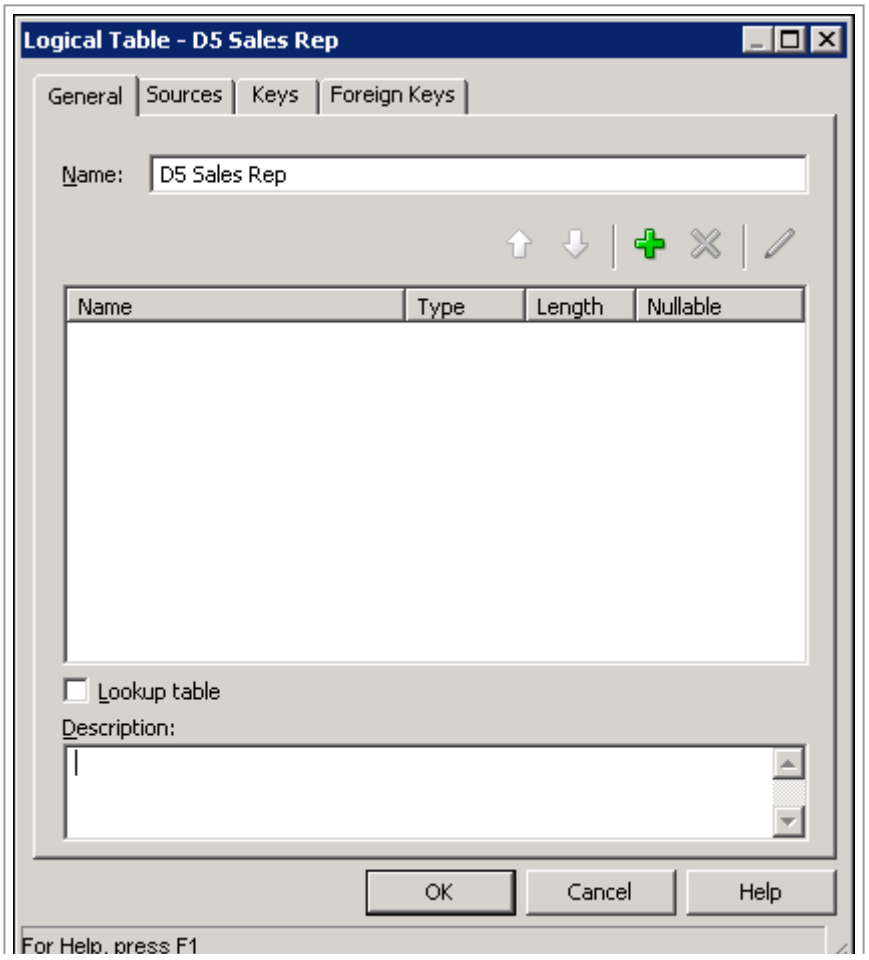



## Creating Logical Table and Logical Columns

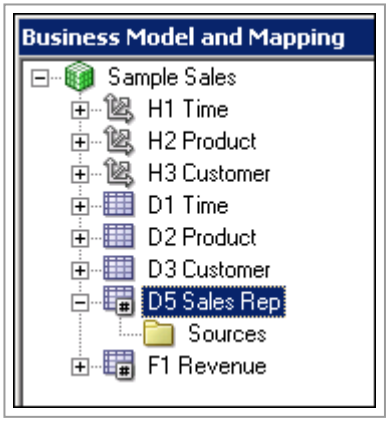
1. In the **BMM** layer, right-click the **Sample Sales** business model and select **New Object > Logical Table** to open the Logical Table dialog box.




**2.** On the **General** tab, name the logical table **D5 Sales Rep**.

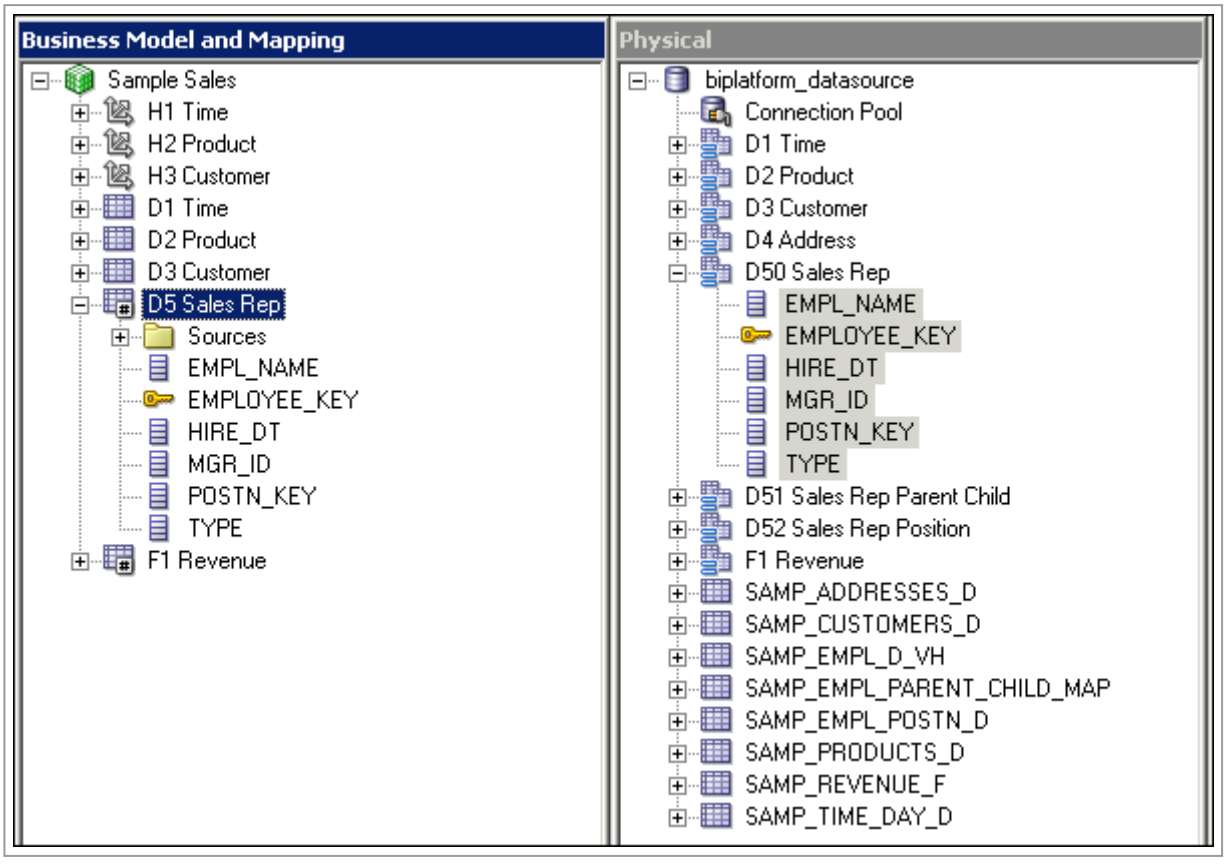


3. Click **OK** to add the logical table to the business model.

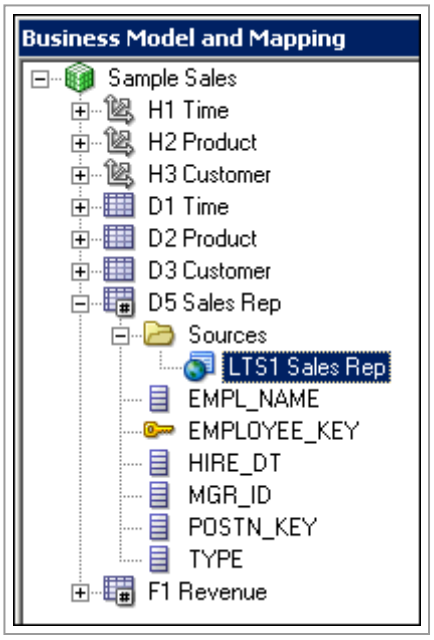



Notice that the D5 Sales Rep icon has a # sign. This is because you have not yet defined the logical join relationship. When you define the logical join later in this tutorial the icon will change accordingly.

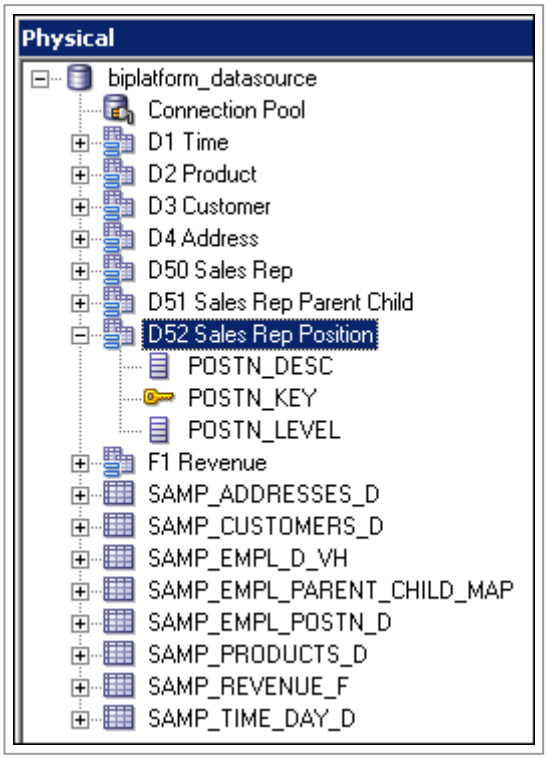
4. Drag all six columns from **D50 Sales Rep** in the Physical layer to **D5 Sales Rep** in the **BMM** layer. This action creates logical columns and adds a D50 Sales Rep logical table source to D5 Sales Rep.



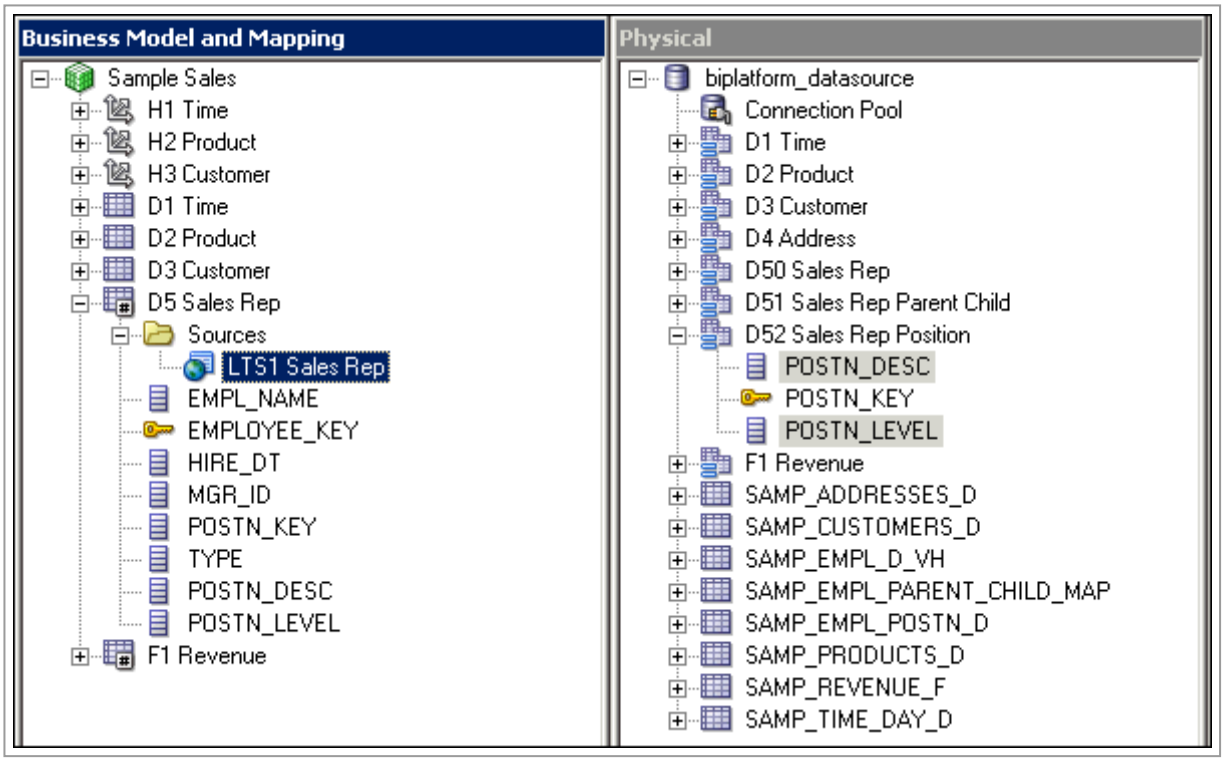

5. Rename the D50 Sales Rep logical table source to **LTS1 Sales Rep**.



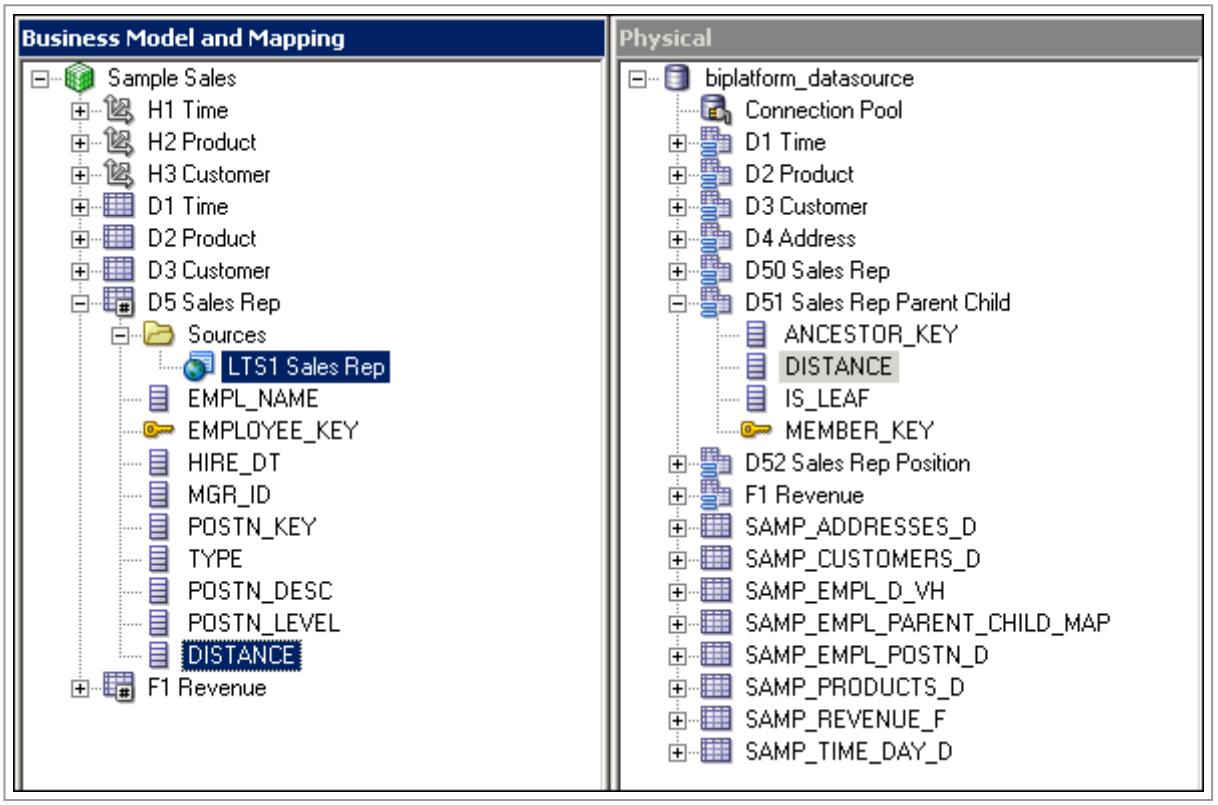

6. In the **Physical** layer, expand **D52 Sales Rep Position**.




7. Drag **POSTN_DESC** and **POSTN_LEVEL** from **D52 Sales Rep Position** to **LTS1 Sales Rep**. Note that you are dragging the columns to the logical table source, not the logical table. Dragging to the logical table would create a second logical table source.




8. Drag **DISTANCE** from **D51 Sales Rep Parent Child** to **LTS1 Sales Rep**. Again, you drag the column to the logical table source, not the logical table.

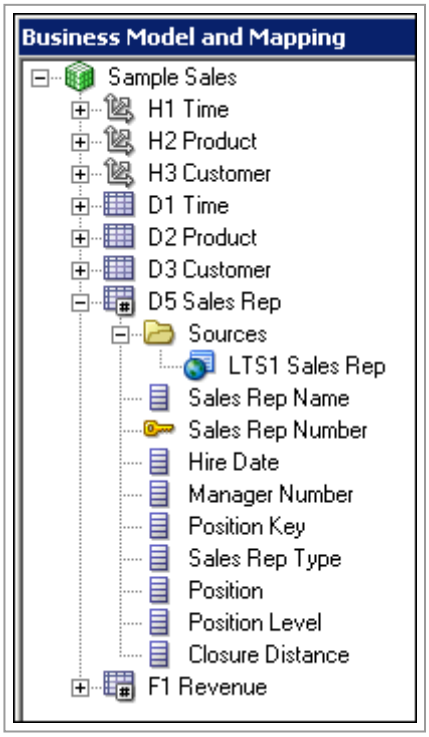



9. Rename the logical columns:

| Old Name | New Name |
|---|---|
| POSTN_KEY | Position Key |
| TYPE | Sales Rep Type |

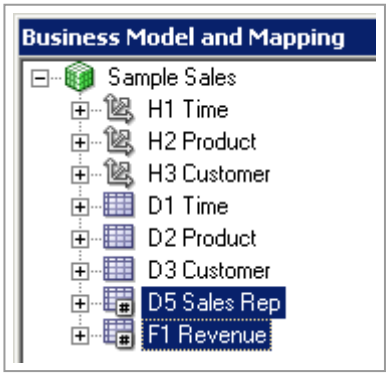
| | |
|---|---|
| EMPL_NAME | Sales Rep Name |

| | |
|---|---|
| EMPLOYEE_KEY | Sales Rep Number |
| HIRE_DT | Hire Date |
| MGR_ID | Manager Number |
| POSTN_DESC | Position |
| POSTN_LEVEL | Position Level |
| DISTANCE | Closure Distance |

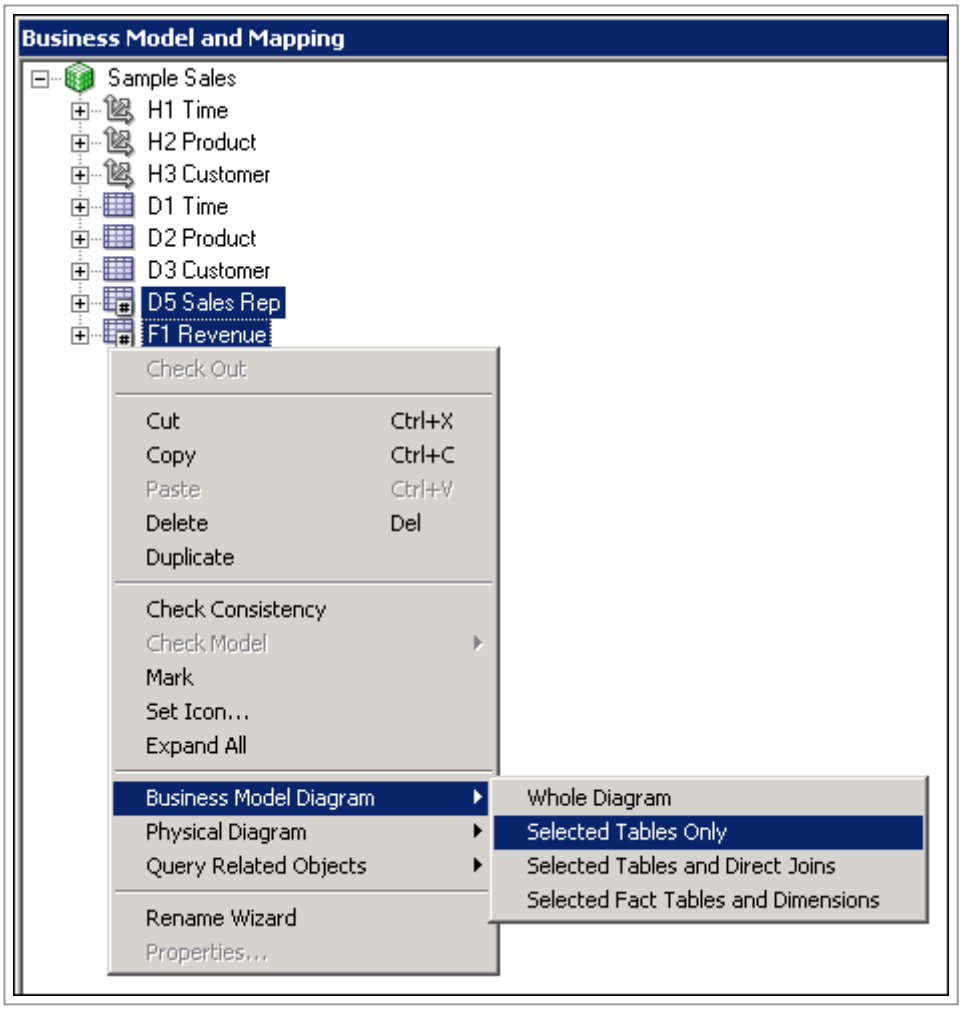



## Creating a Logical Join

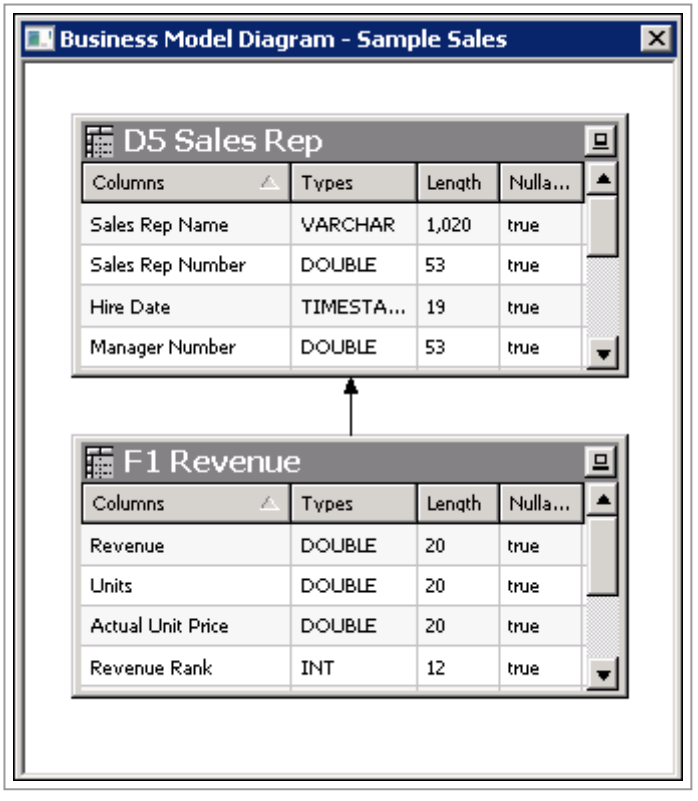
1. In the **BMM** layer, select **D5 Sales Rep** and **F1 Revenue**.



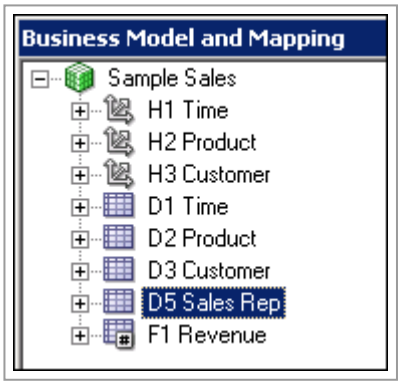

2. Right-click either highlighted table and select **Business Model Diagram > Selected Tables Only** to open the Business Model Diagram.




3. Create a logical join between **D5 Sales Rep** and **F1 Revenue** with F1 Revenue at the many end of the join.



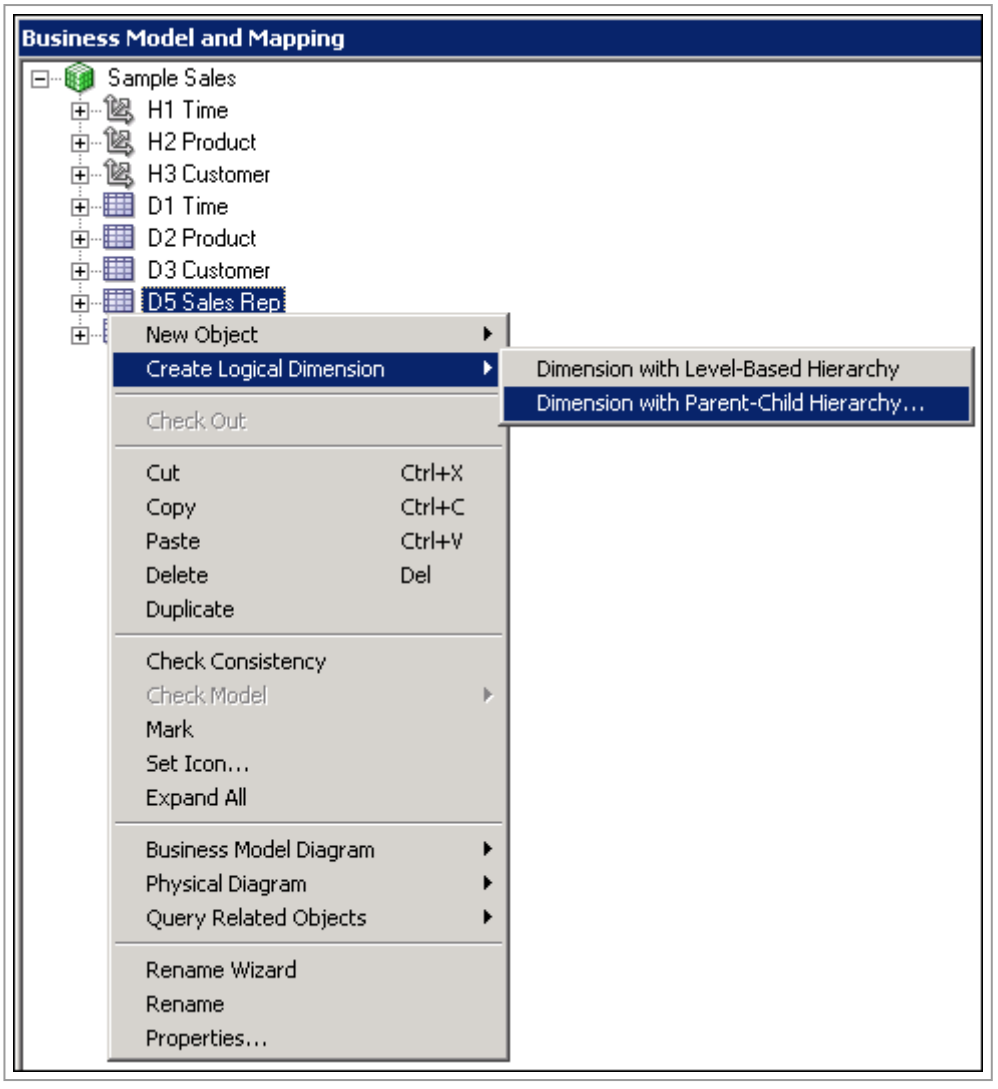
4. Close the Business Model Diagram. Notice that the icon has changed for the D5 Sales Rep table.



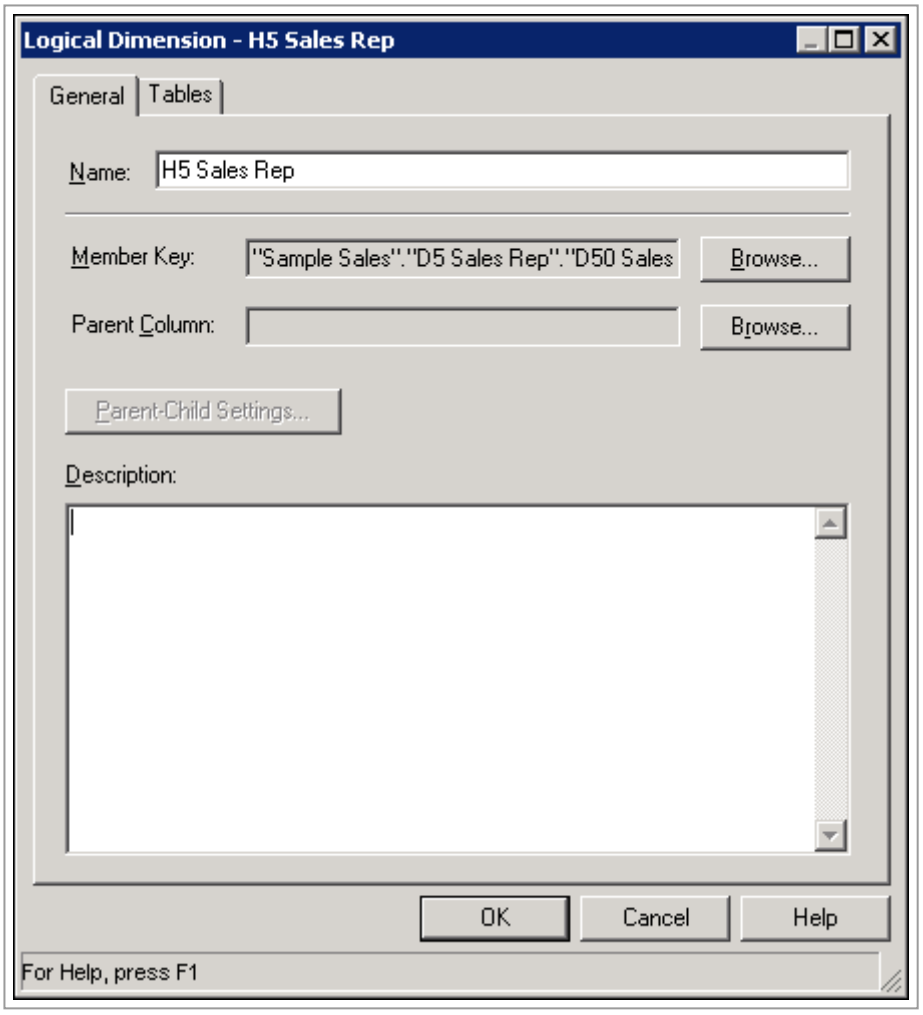

Notice that the D5 Sales Rep icon is not having # sign.

## Creating a Parent-Child Logical Dimension

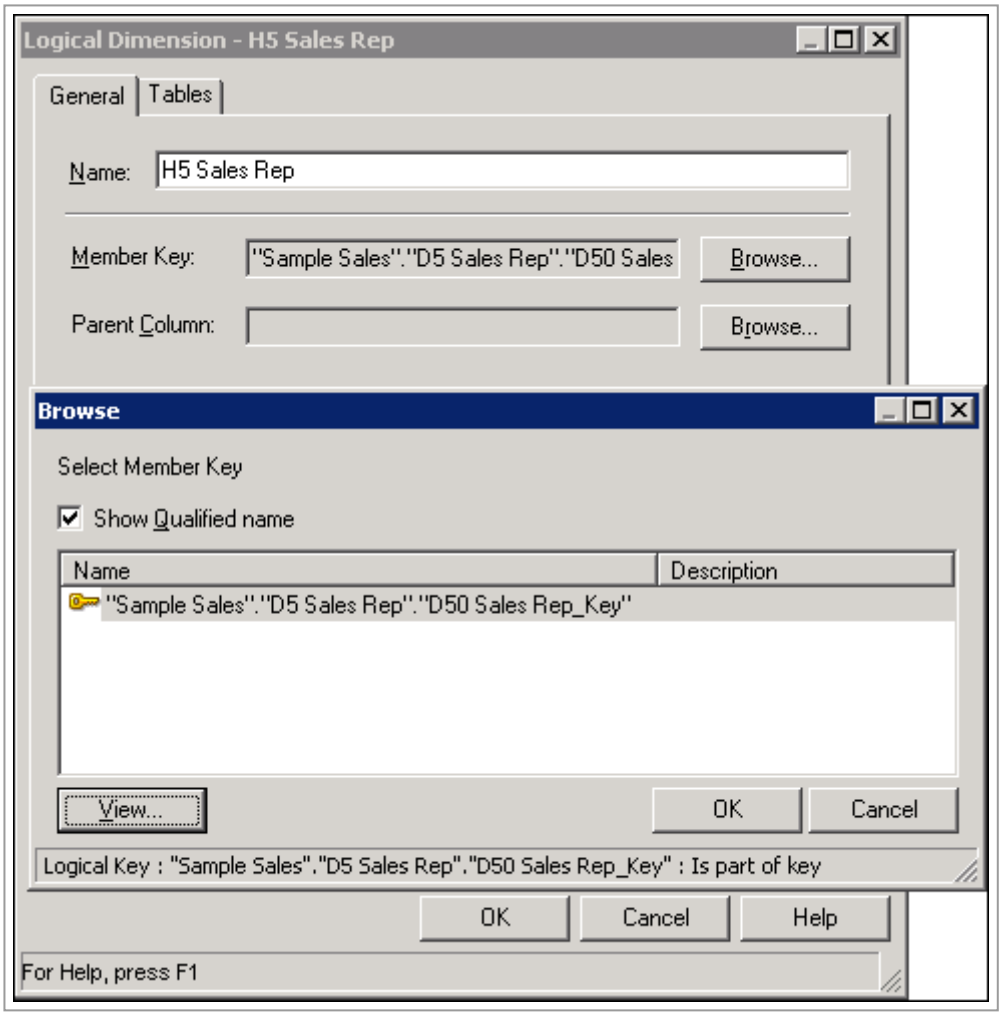
1. Right-click the **D5 Sales Rep** logical table and select **Create Logical Dimension > Dimension with Parent-Child Hierarchy**.



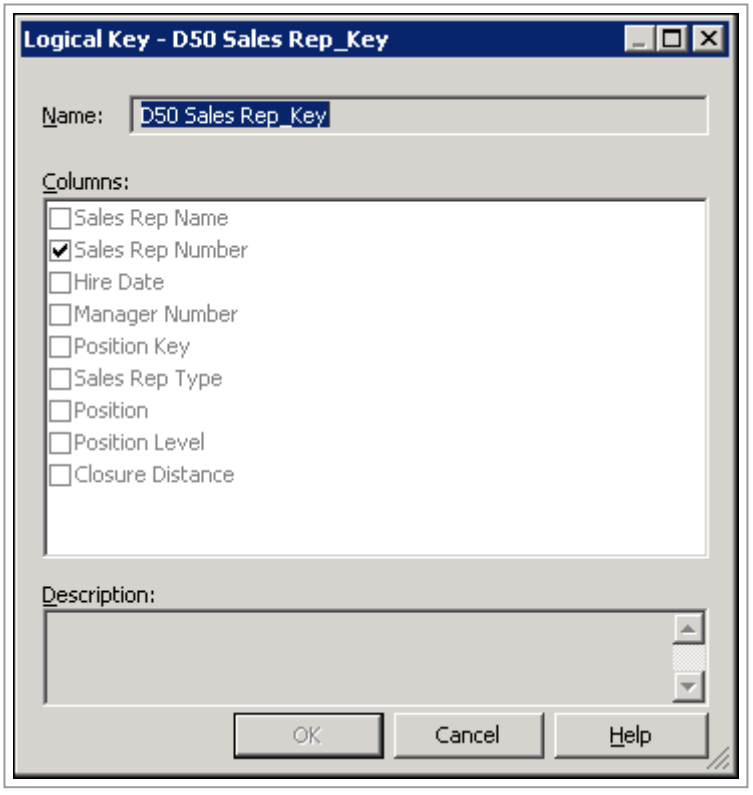

2. In the Logical Dimension dialog box, on the **General** tab, name the logical dimension **H5 Sales Rep**.




3. Click **Browse** next to Member Key. The Browse window shows the physical table and its corresponding key.



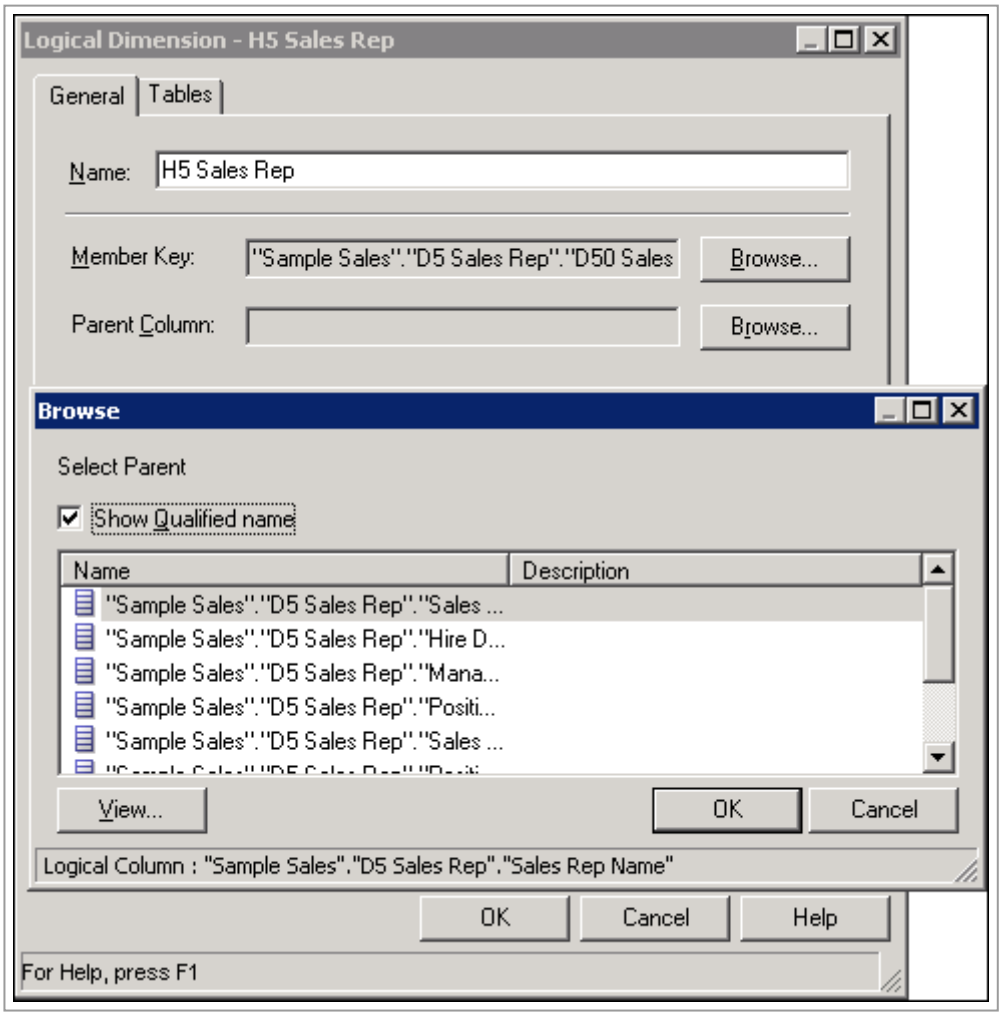

**4.** Click **View** to open the Logical Key dialog box. Confirm that the **Sales Rep Number** column is selected.



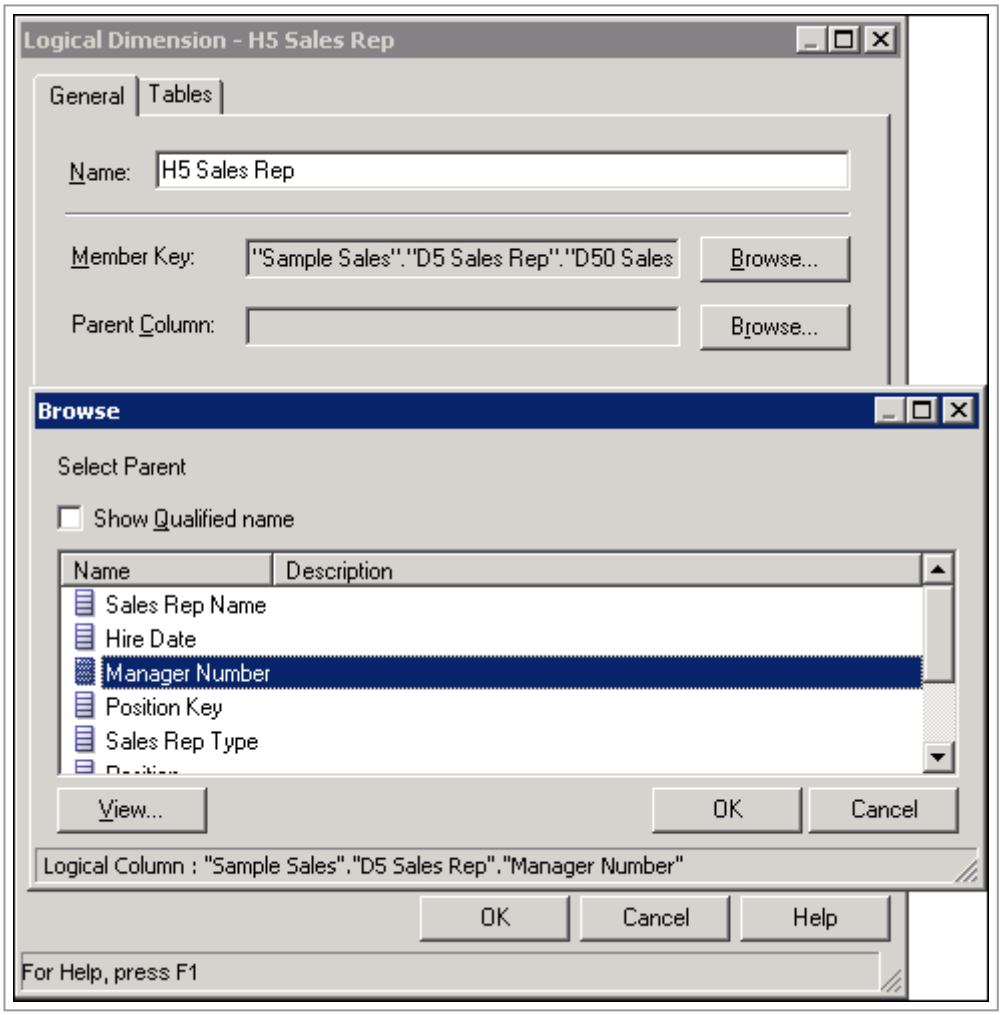

**5.** Click **Cancel** to close the Logical Key dialog box.

6. Click **OK** to close the Browse window.

7. Click **Browse** next to Parent Column. The Browse window shows the columns other than the member key.



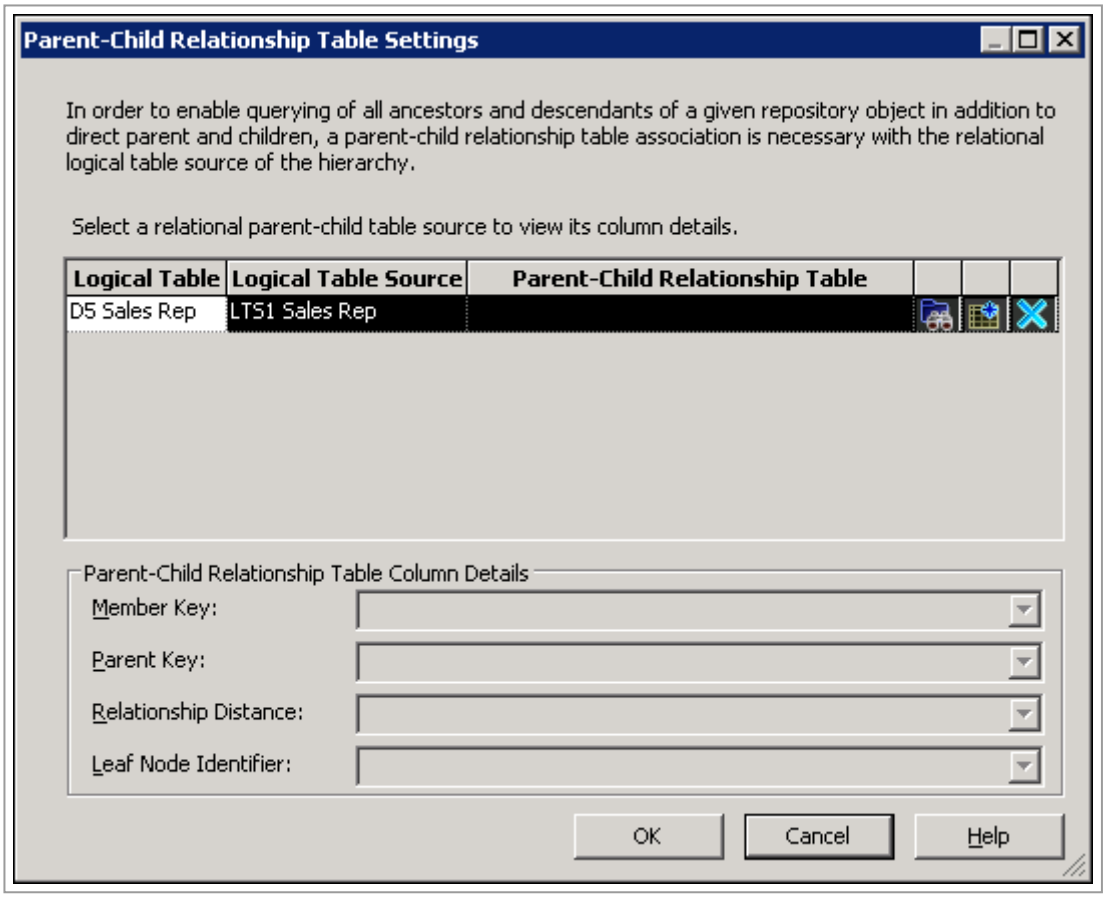

8. Deselect **Show Qualified Names** and select **Manager Number** as the parent column for the parent-child hierarchy.




9. Click **OK** to close the Browse window, but do not close the Logical Dimension dialog box.

## Defining Parent-Child Settings

1. Click **Parent-Child Settings** to display the Parent-Child Relationship Table Settings dialog box. Note that at this point the Parent-Child Relationship Table is not defined.



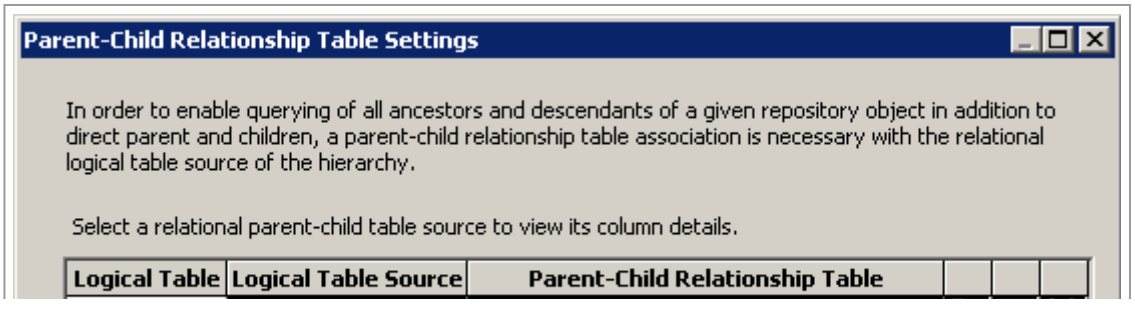

For each parent-child hierarchy defined on a relational table, you must explicitly define the inter-member relationships in a separate parent-child relationship table. In the process of creating the parent-child relationship table, you may choose one of the following options:

1. Select a previously-created parent-child relationship table.

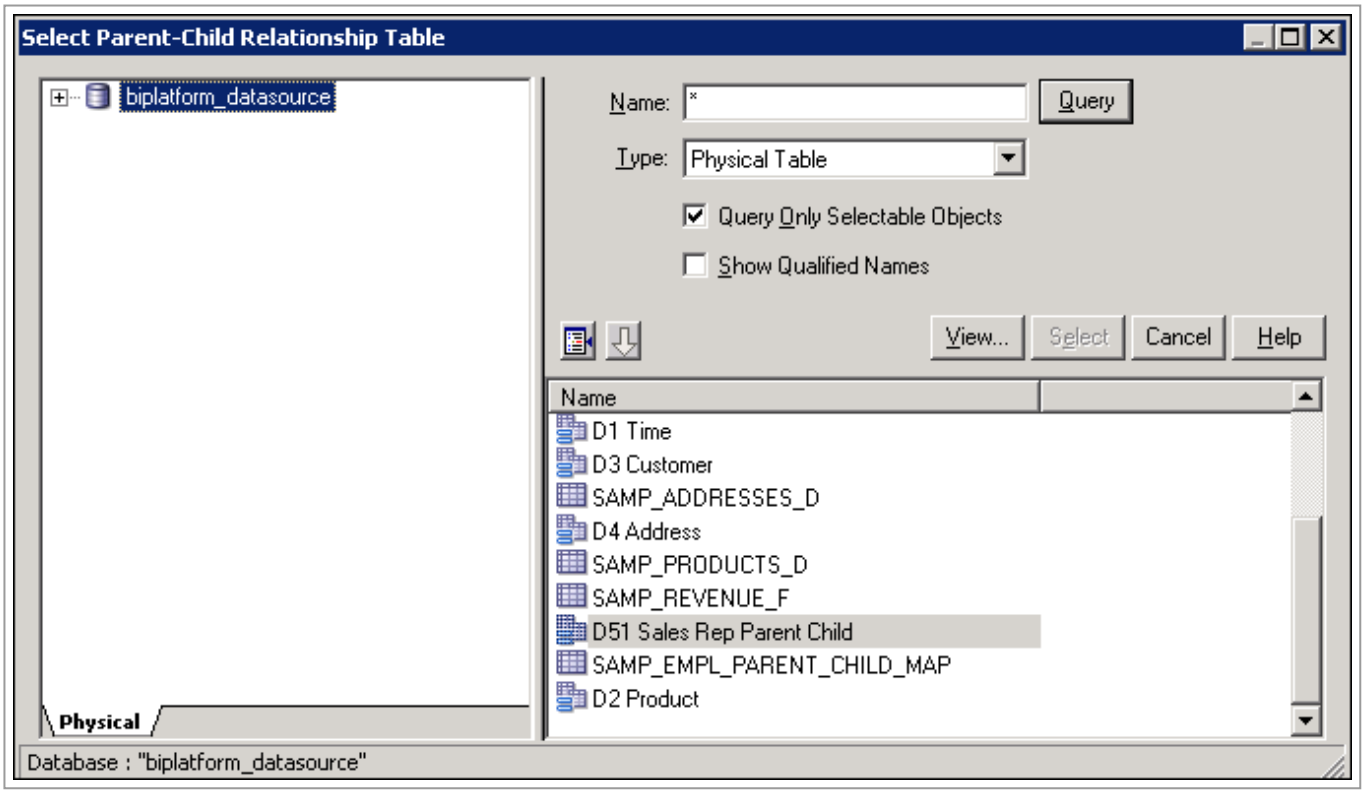
2. Use a wizard that will generate scripts to create and populate the parent-child relationship table. In the next set of steps you select a previously created and populated parent-child relationship table.

For your information only: To start the wizard you would click the Create Parent-Child Relationship Table button. The wizard creates the appropriate repository metadata objects and generates SQL scripts for creating and populating the parent-child relationship table. At the end of the wizard, Oracle BI Server stores the scripts into directories chosen during the wizard session. The scripts can then be run against the database to create and populate the parent-child relationship table. Running the wizard is not necessary in this tutorial because the parent-child relationship table is already created and populated.

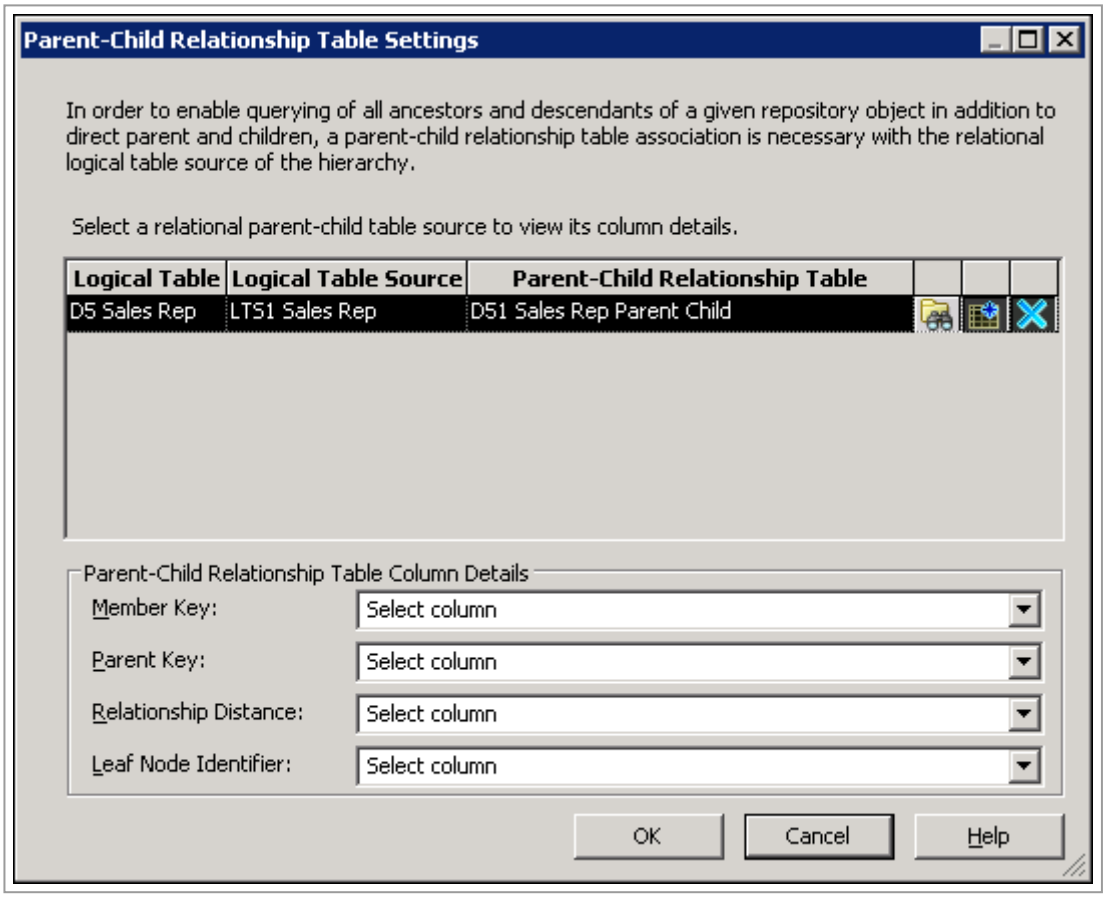
**2.** Click the **Select Parent-Child Relationship Table** button to open the Select Physical Table dialog box.




**3.** In the Select Physical Table dialog box, select the **D51 Sales Rep Parent Child** alias you created.



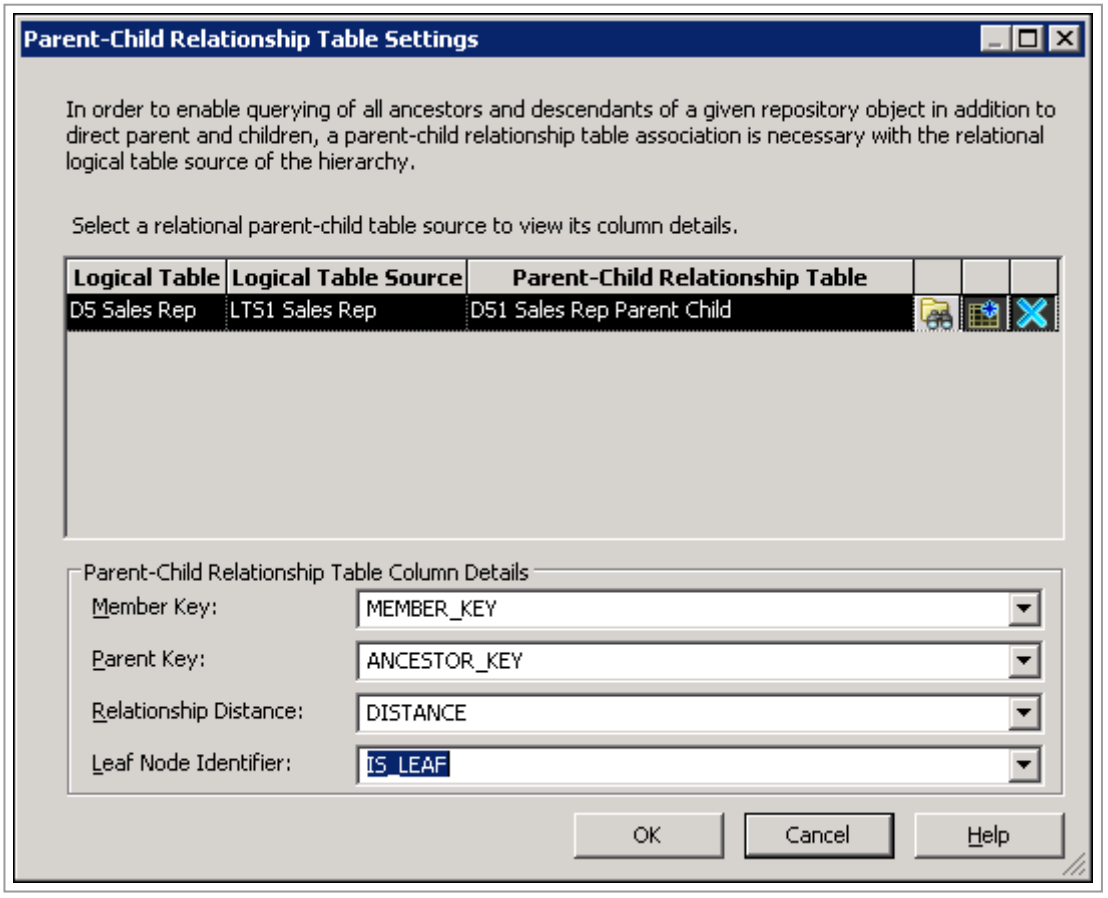

**4.** The D51 Sales Rep Parent Child alias is now displayed in the Parent-Child Relationship Table column.




**5.** In the **Parent-Child Table Relationship Column Details** section, set the appropriate columns:

5. In the **Parent-Child Table Relationship Column Details** section, set the appropriate columns:

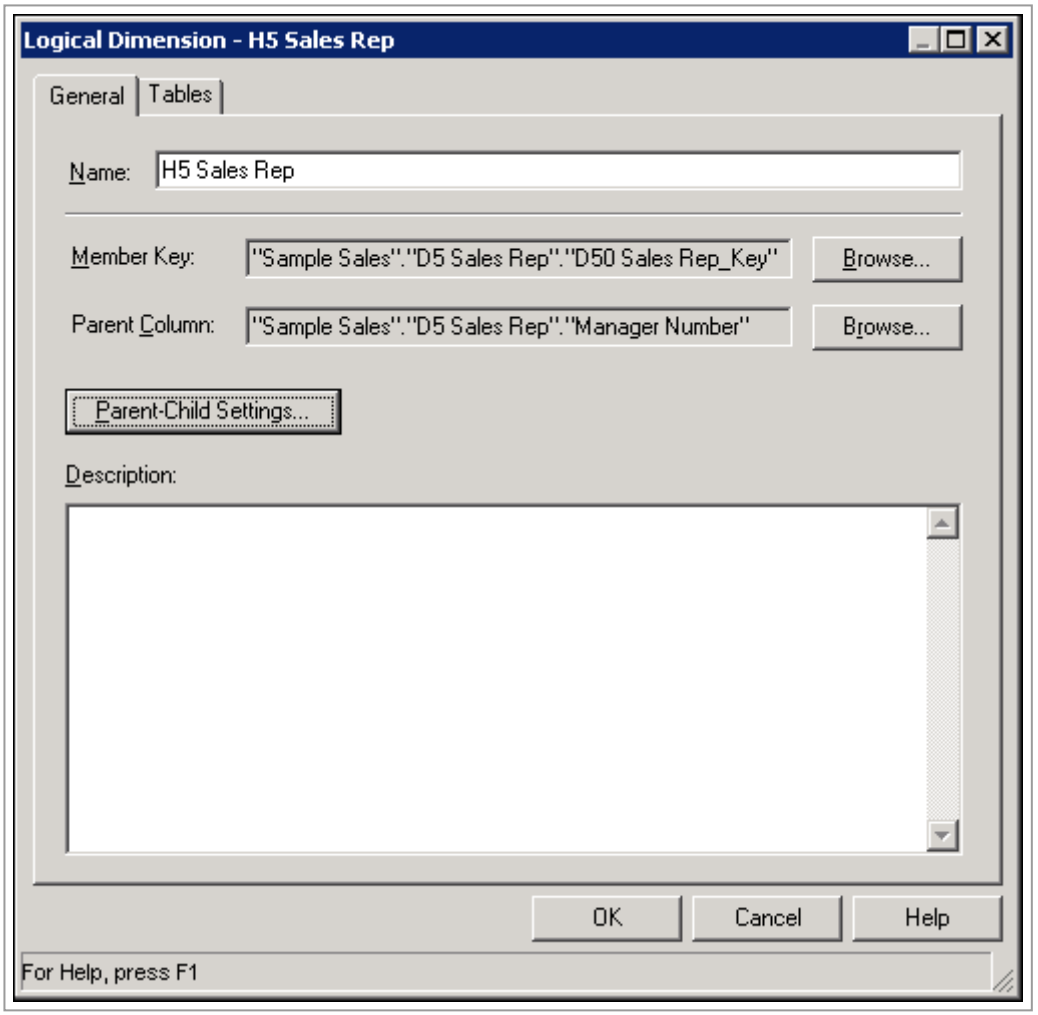
| | |
|---|---|
| **Member Key** | MEMBER_KEY |
| **Parent Key** | ANCESTOR_KEY |
| **Relationship Distance** | DISTANCE |
| **Leaf Node Identifier** | IS_LEAF |



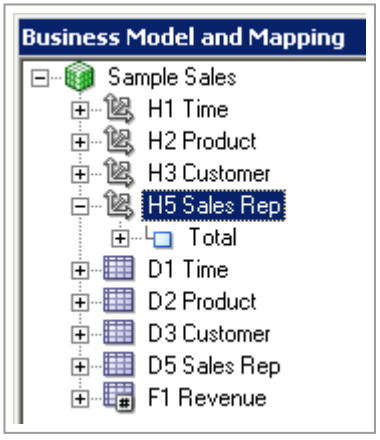

Explanation:

Member Key identifies the member.
Parent Key identifies an ancestor of the member, The ancestor may be the parent of the member, or a higher-level ancestor.
Relationship Distance specifies the number of parent-child hierarchical levels from the member to the ancestor.
Leaf Node Identifier indicates if the member is a leaf member (1=Yes, 0=No).

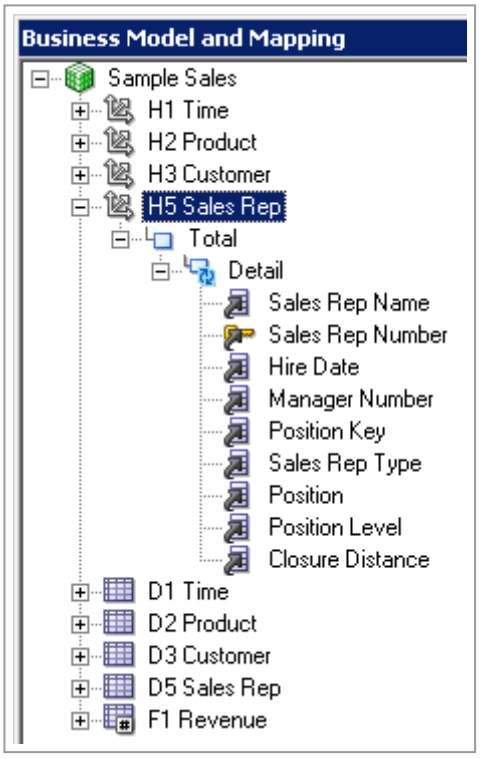
6. Click **OK** to close the Parent-Child Relationship Table Settings dialog box.



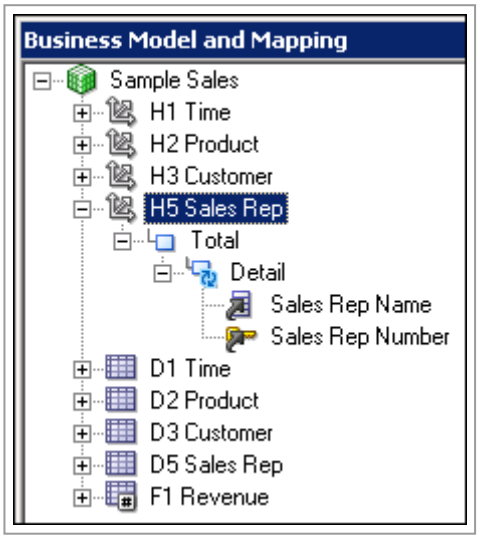

7. Click **OK** to close the Logical Dimension dialog box.




8. Right-click **H5 Sales Rep** and select **Expand All**. Note that a parent-child logical dimension has only two levels.

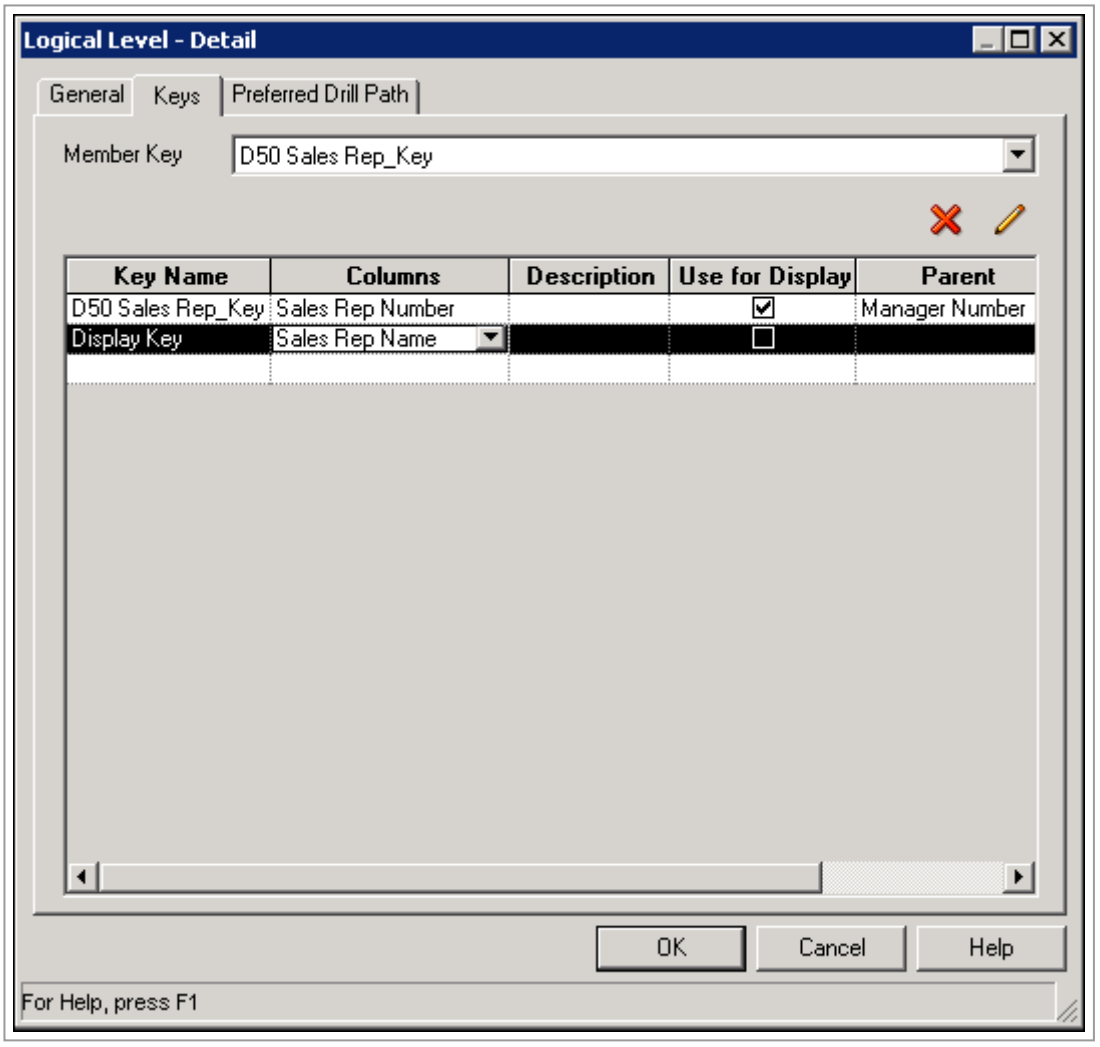



**9.** Delete all columns from the Detail level except for **Sales Rep Name** and **Sales Rep Number**.

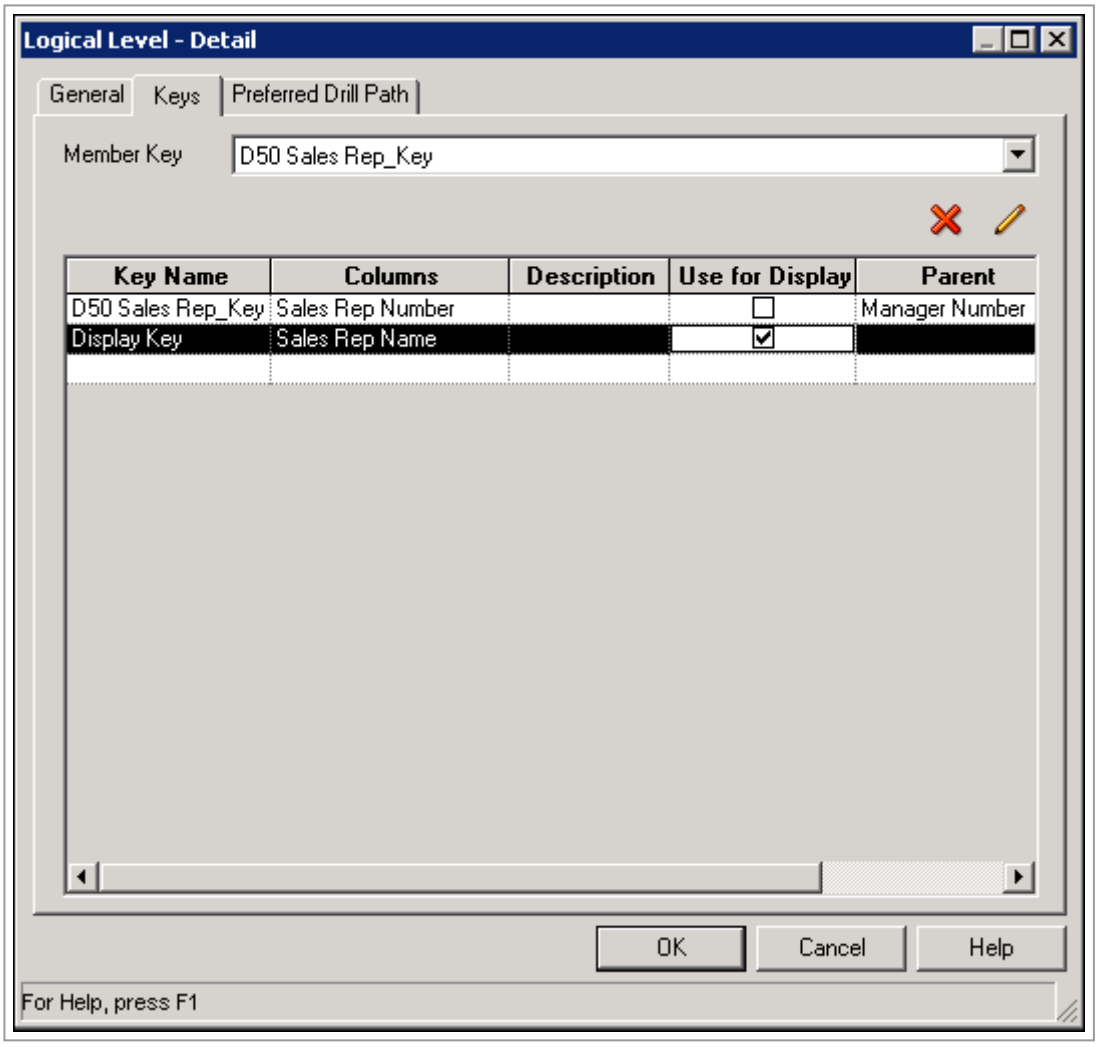



**10.** Double-click the **Detail** level to open the Logical Level dialog box.

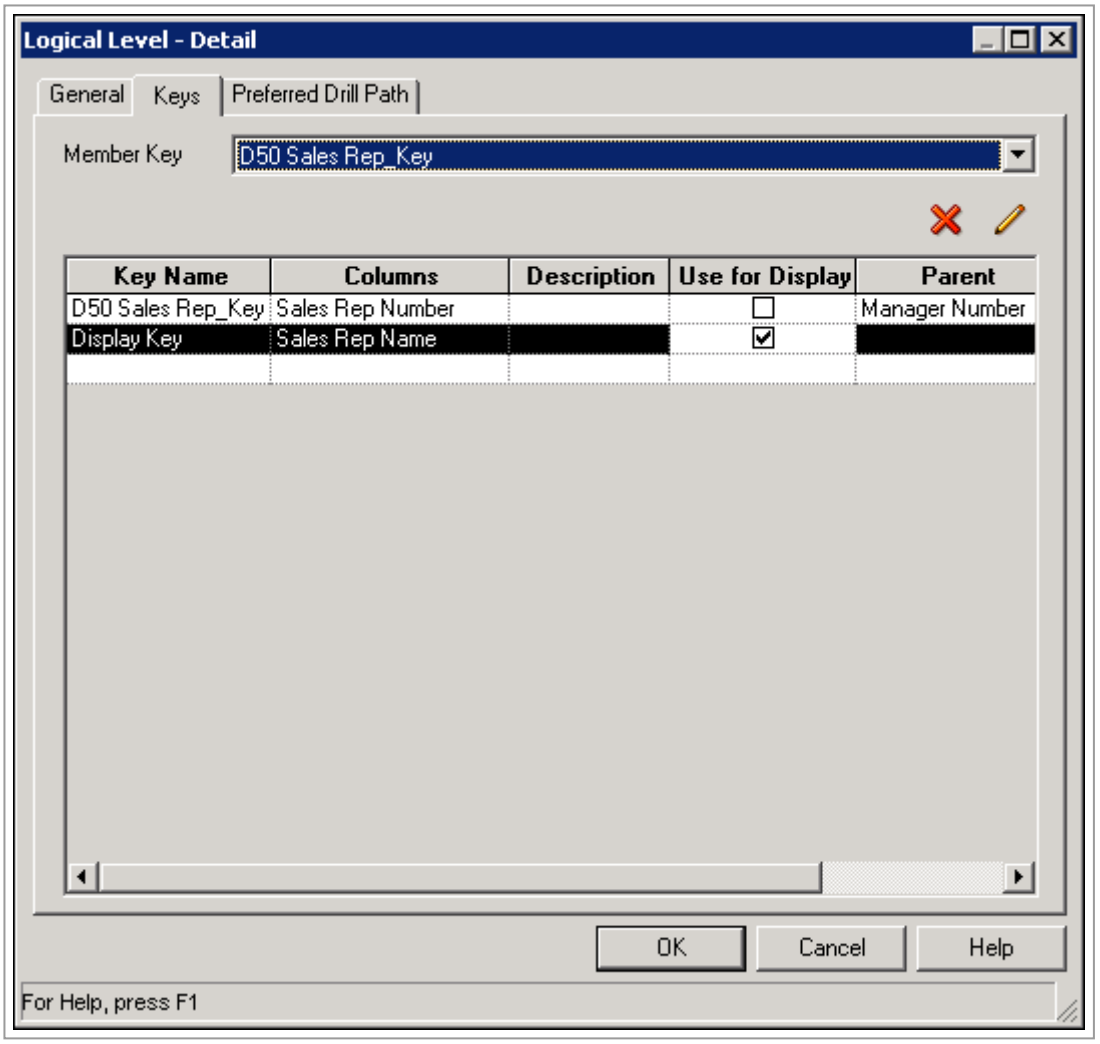
**11.** On the Keys tab, create a new key named **Display Key** that maps to the **Sales Rep Name** column.



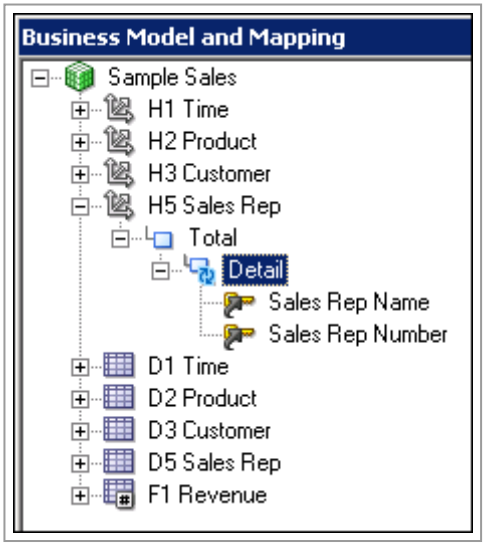

12. Deselect **Use for Display** for the **Sales Rep Number** column and select **Use for Display** for the **Sales Rep Name** column.



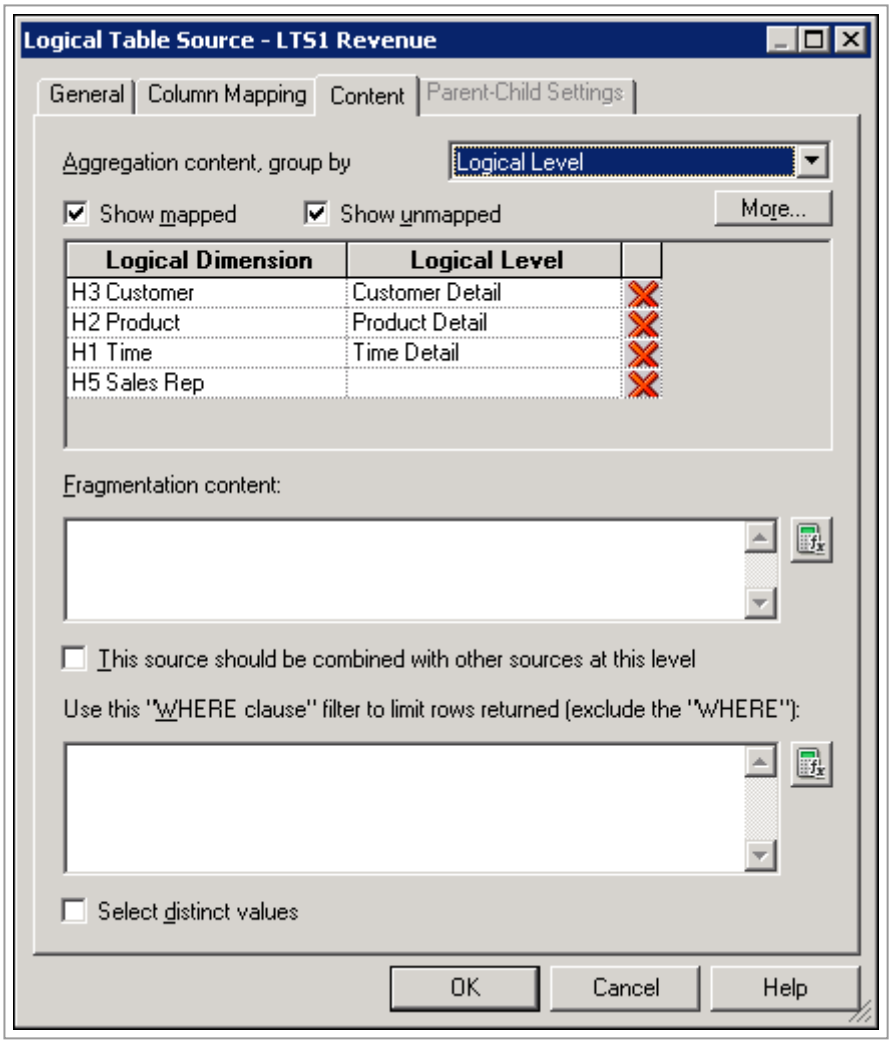

**13.** Make sure that Member Key is still set to **D50 Sales Rep_Key**.



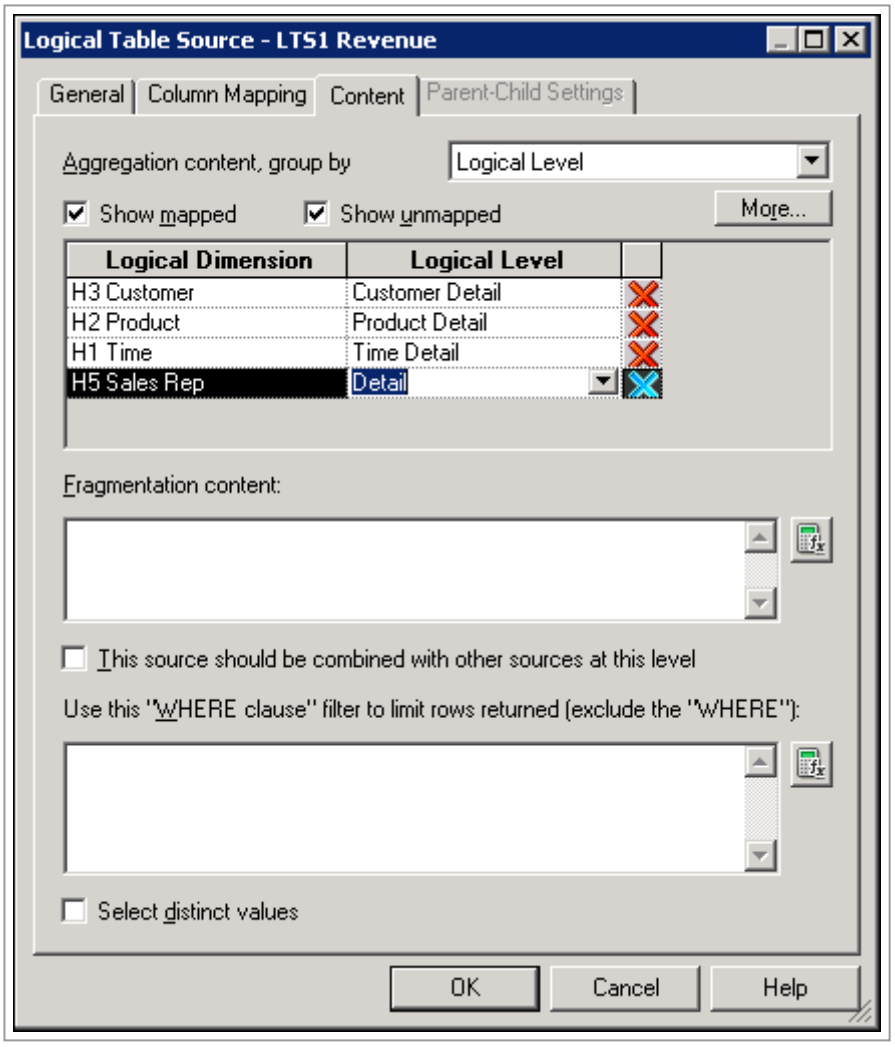

**14.** Click **OK** to close the Logical Level dialog box.



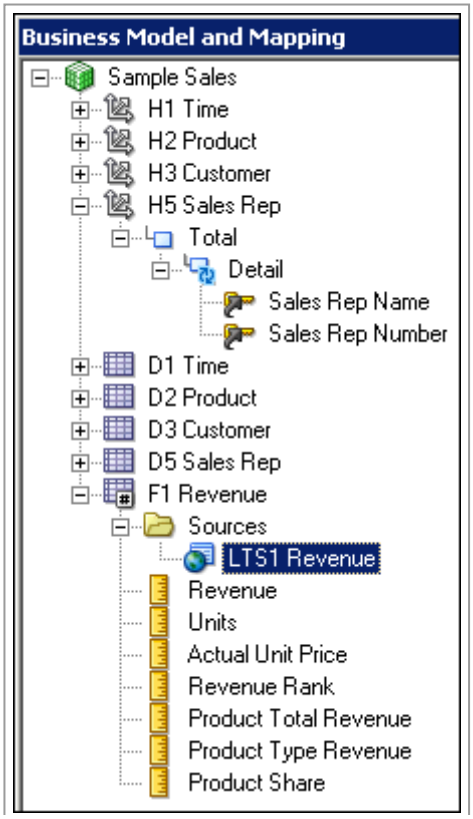

**15.** Expand **F1 Revenue > Sources** and double-click **LTS1 Revenue** to open the Logical Table Source dialog box.




**16.** On the Content tab, set the logical level to **Detail** for the **H5 Sales Rep** logical dimension.

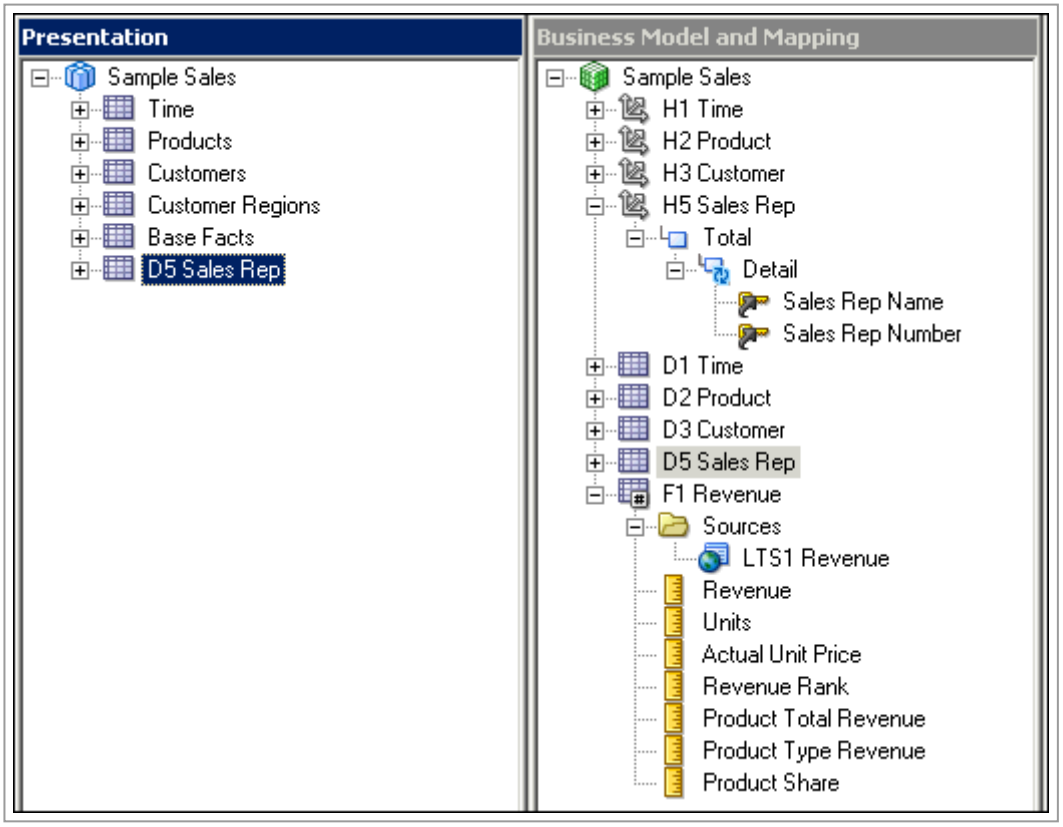



**17.** Click **OK** to close the Logical Table Source dialog box.

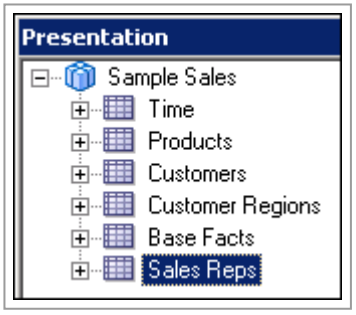


## Creating Presentation Layer Objects

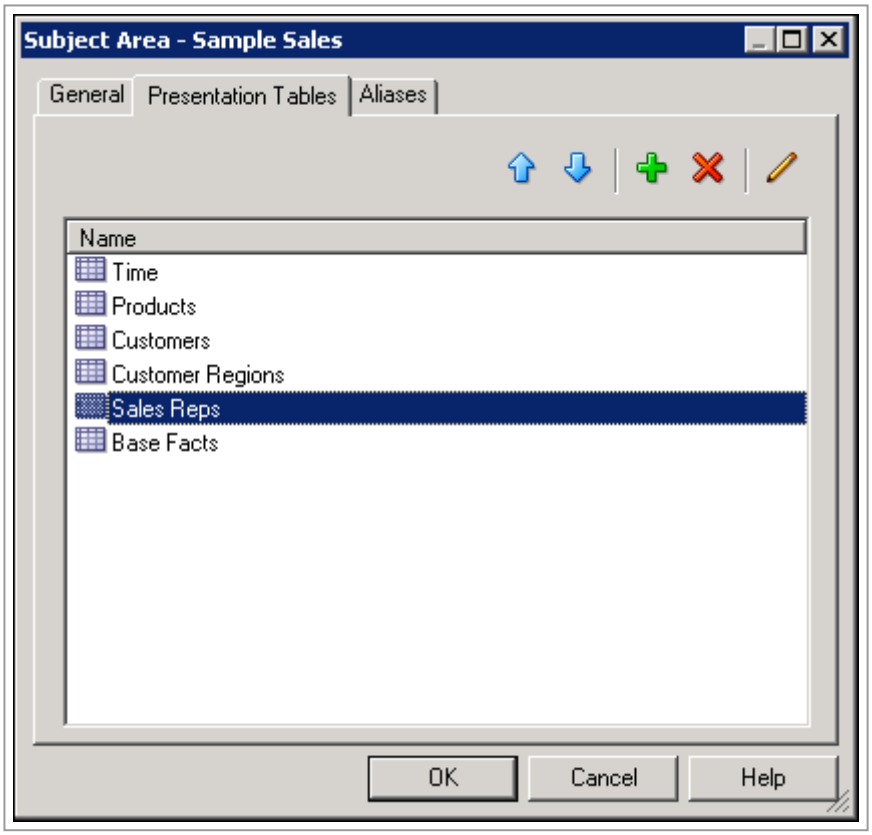
1. Drag the **D5 Sales Rep** logical table from the **BMM** layer to the **Sample Sales** subject area in the **Presentation** layer.



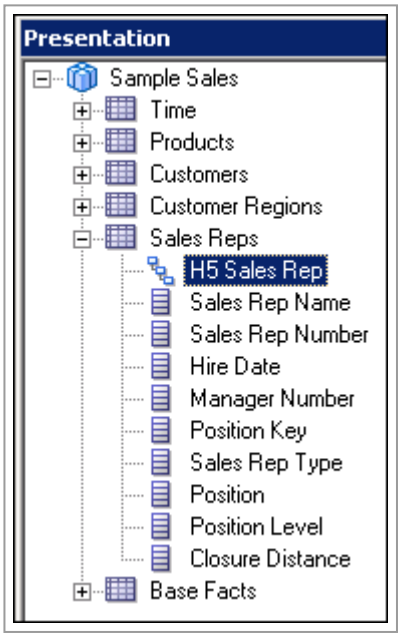

2. Rename the **D5 Sales Rep** presentation table to **Sales Reps**.




3. Move the **Sales Reps** presentation table above the **Base Facts** table.

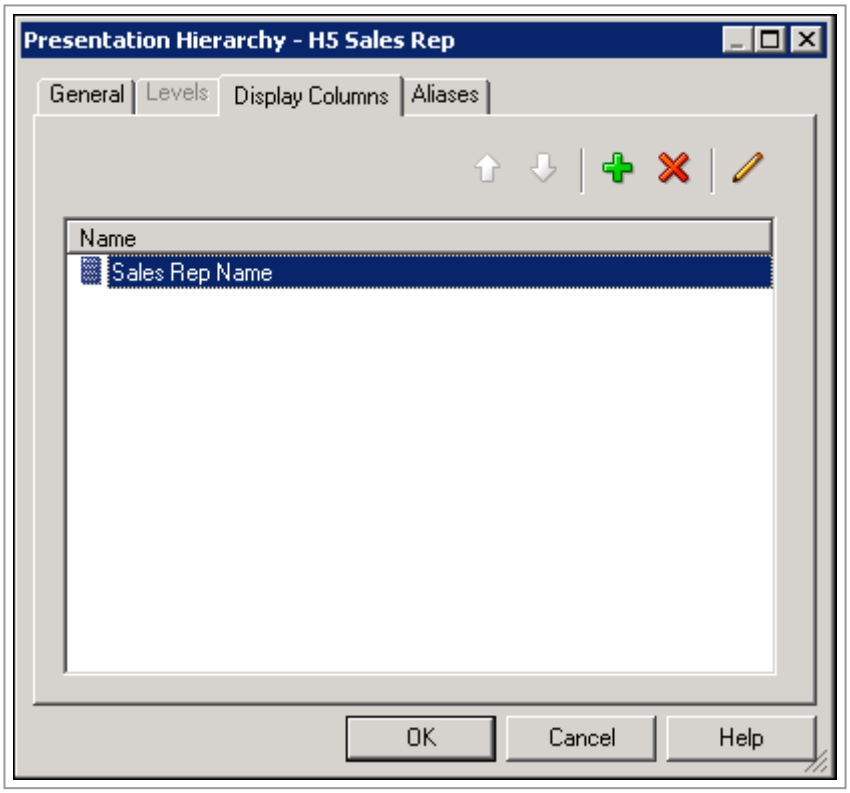



4. Expand the **Sales Reps** presentation table and notice that the **H5 Sales Rep** parent-child logical dimension is automatically included as a presentation hierarchy.




5. Double-click the **H5 Sales Rep** presentation hierarchy to open the Presentation Hierarchy dialog box.

6. On the **Display Columns** tab, confirm that **Sales Rep Name** is set as the display column.




7. Click **OK** to close the Presentation Hierarchy dialog box.

8. Save the repository and check consistency. Fix any errors or warnings before proceeding.

9. Close the repository. Leave the Administration Tool open.

## Testing Your Work

1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

2. Return to **Oracle BI**, which should still be open, and sign in.

3. Create the following analysis to test the parent-child logical dimension.

   **Sales Reps.H5 Sales Reps**
   **Sales Reps.Position**
   **Base Facts.Revenue**


4. Click **Results**.

| | | |
|---|---|---|
| ▶ Michele Lombardo | President | 50,000,000 |

5. Expand the pivot table to view data at different levels of the hierarchy. Notice that the Revenue measure rolls up through each level.

| H5 Sales Rep | Position | Revenue |
|---|---|---|
| ◢ Michele Lombardo | President | 50,000,000 |
| Aurelio Miranda | Director | 2,300,000 |
| ◢ Helen Mayes | Vice President | 9,460,000 |
| Angela Richards | Sr. Manager | 1,107,205 |
| ◢ Chris Jones | Sr. Manager | 7,973,087 |
| Charles Brooks | Senior | 4,092,130 |
| Edilberto Mandani | Principal | 3,880,957 |
| ◢ Monica Velasquez | Vice President | 17,090,000 |
| ▶ Anne Green | Sr. Manager | 7,729,728 |
| ▶ James Dowel | Manager | 4,484,754 |
| ▶ Steve Atkins | Sr. Manager | 4,473,686 |
| ▶ Paul Atkinson | Director | 4,220,000 |
| ▶ Sophie Bergman | Vice President | 16,820,000 |

6. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Creating Logical Dimensions with Ragged and Skipped-Level Hierarchies

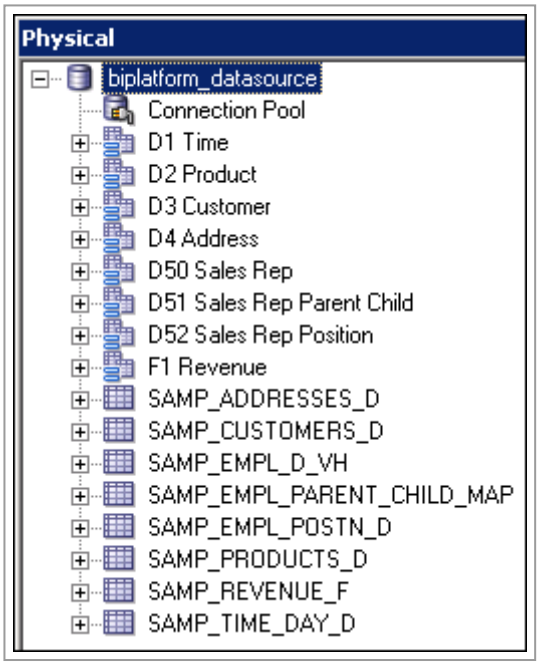
To create logical dimensions with ragged and skipped-level hierarchies, you perform the following steps:

- Importing Metadata and Define Physical Layer Objects
- Creating Logical Table and Logical Columns
- Creating a Ragged/Skipped Levels Logical Dimension
- Creating Presentation Layer Objects
- Testing Your Work

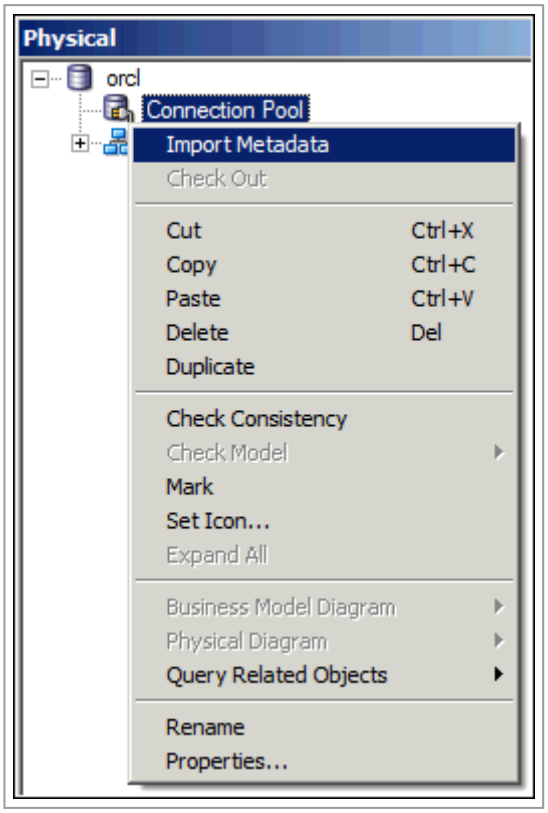
### Importing Metadata and Define Physical Layer Objects

1. Open the **BISAMPLE** repository in offline mode.

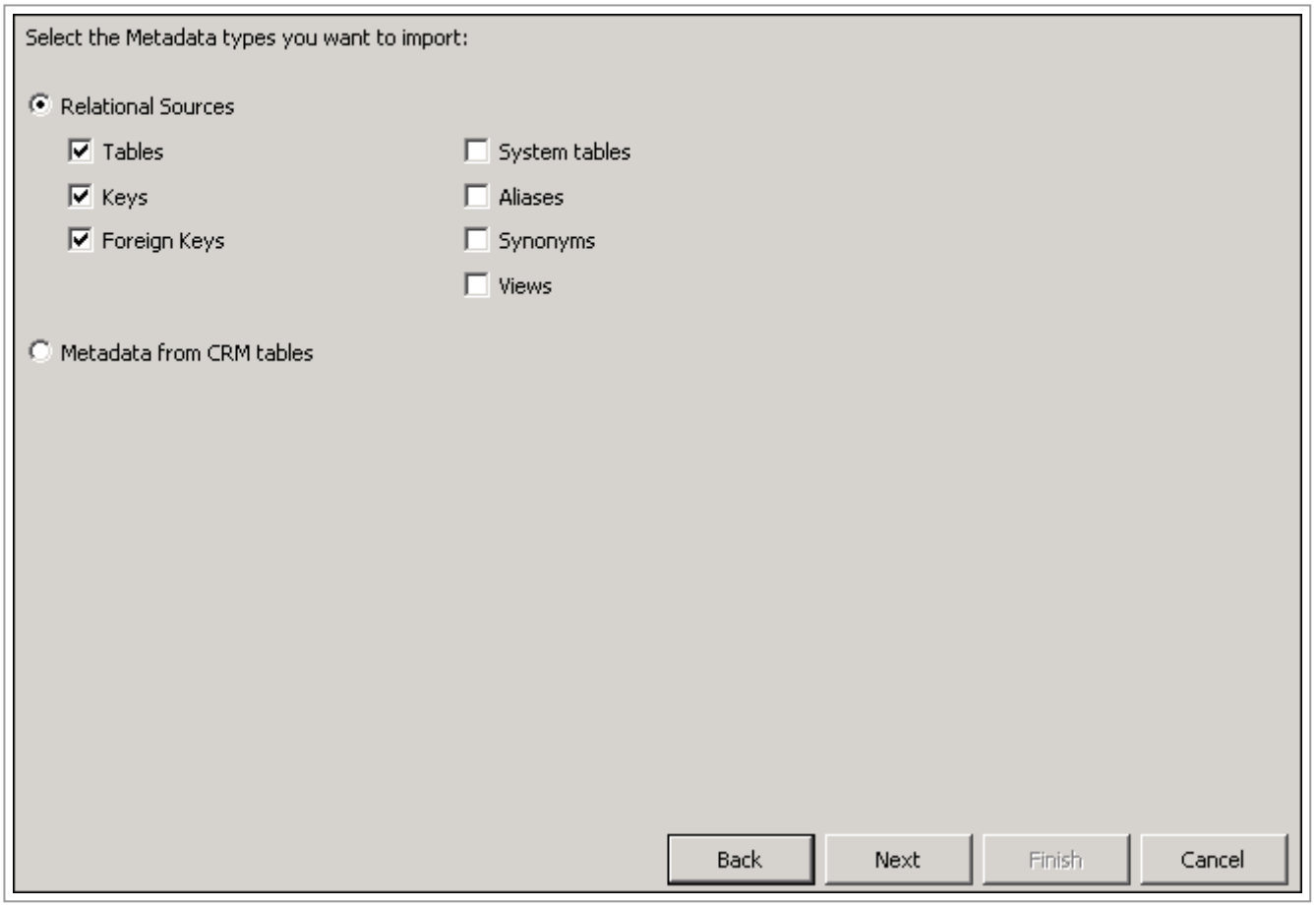
2. In the Physical layer, expand **biplatform_datasource**.




**3.** Right-click **Connection Pool** and select **Import Metadata** to open the Import Wizard.

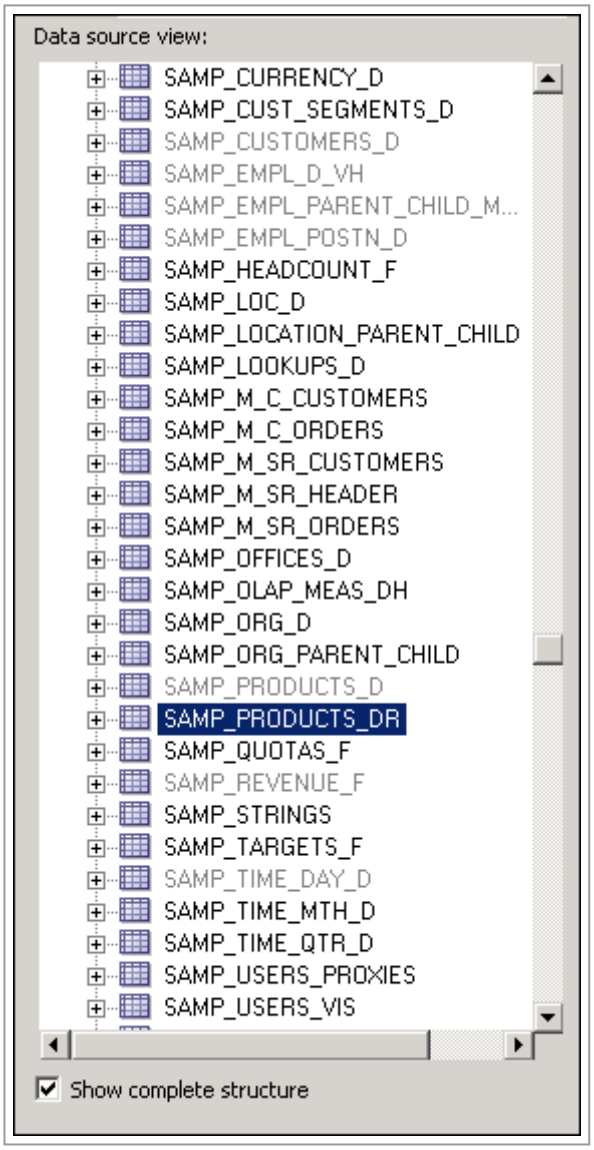



**4.** In the Select Metadata Types screen, accept the defaults and click **Next**.



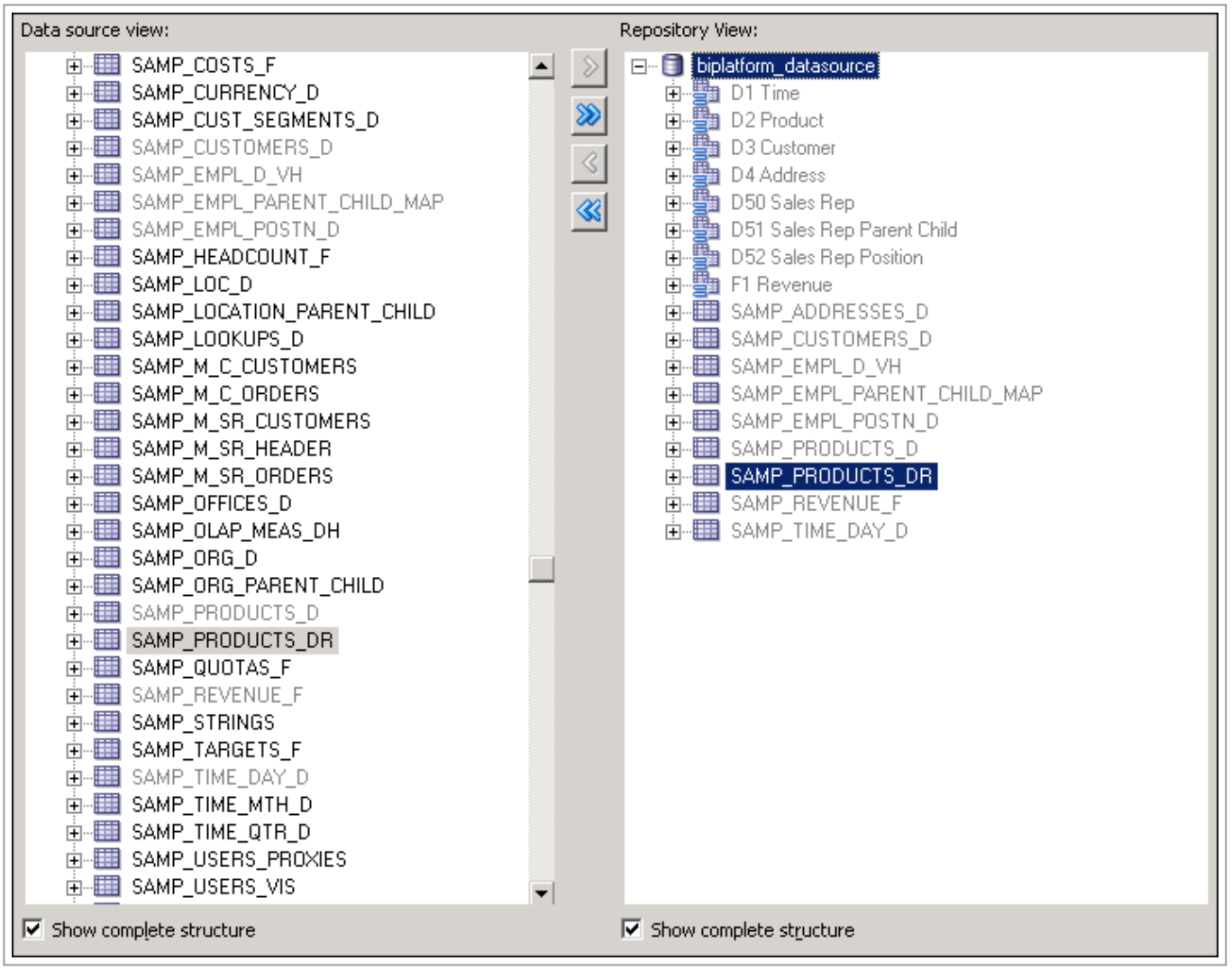

**5.** In the Select Metadata Objects screen, in the data source view, expand **biplatform_datasource**.

**6.** In the data source view, select the **SAMP_PRODUCTS_DR** table for import:



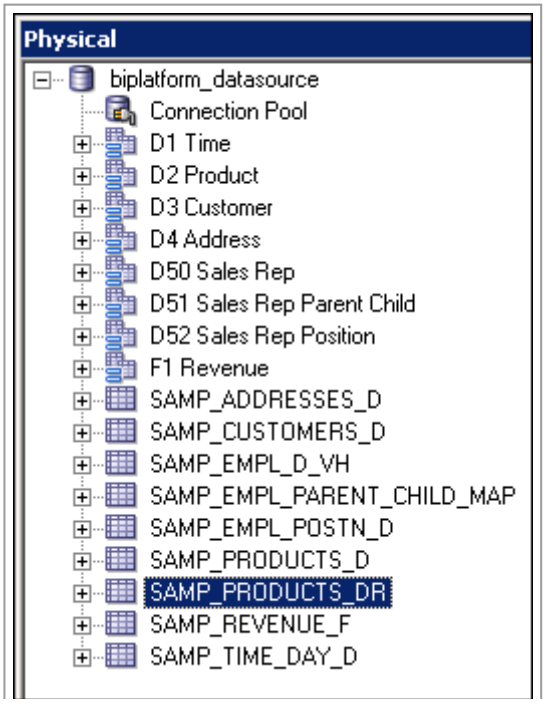

7. Click the **Import Selected button** to move the table to the Repository View.

8. Expand **biplatform_datasource** in the Repository View and confirm that the **SAMP_PRODUCT_DR** table is visible.

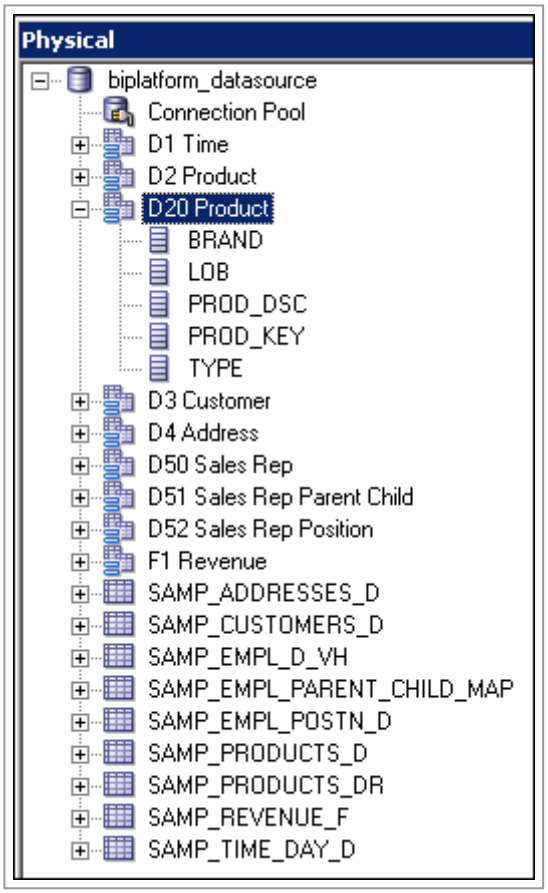



**9.** Click **Finish** to close the Import Wizard.

**10.** Confirm that the **SAMP_PRODUCT_DR** table is visible in the Physical layer of the repository.

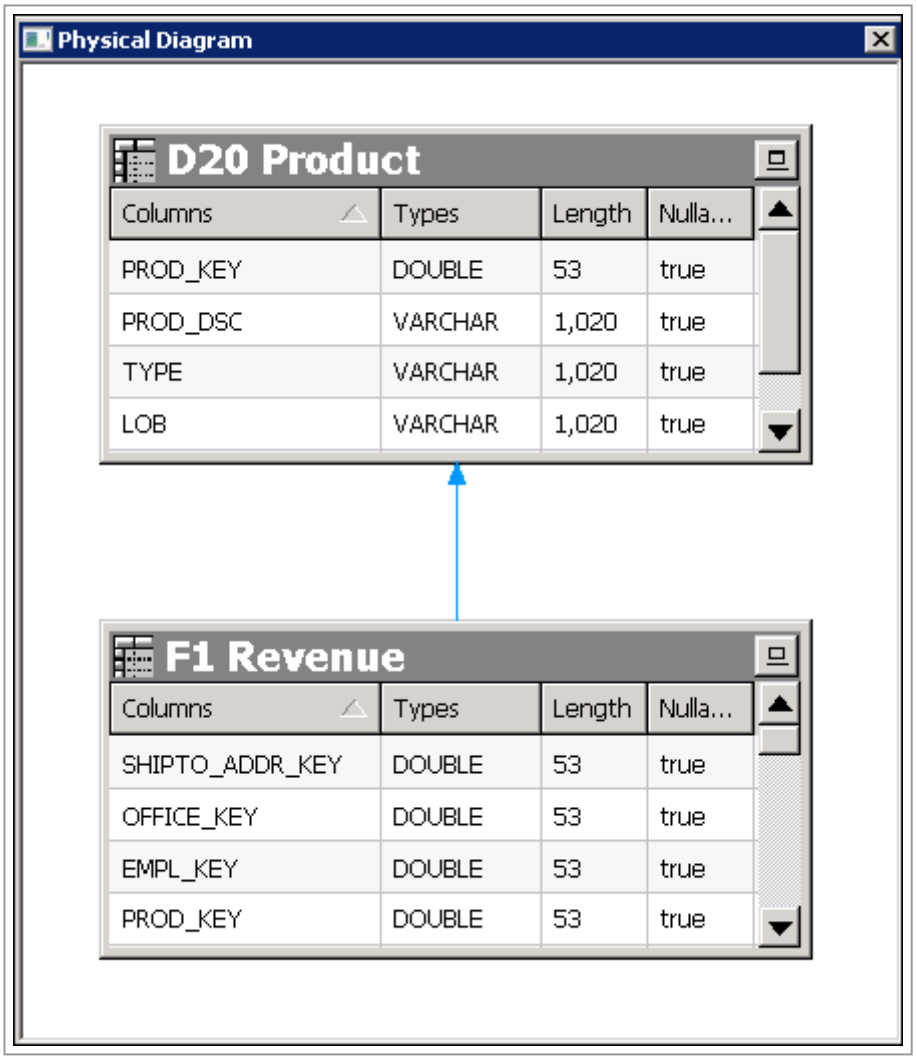


**11.** Create the following alias for the table: **D20 Product**

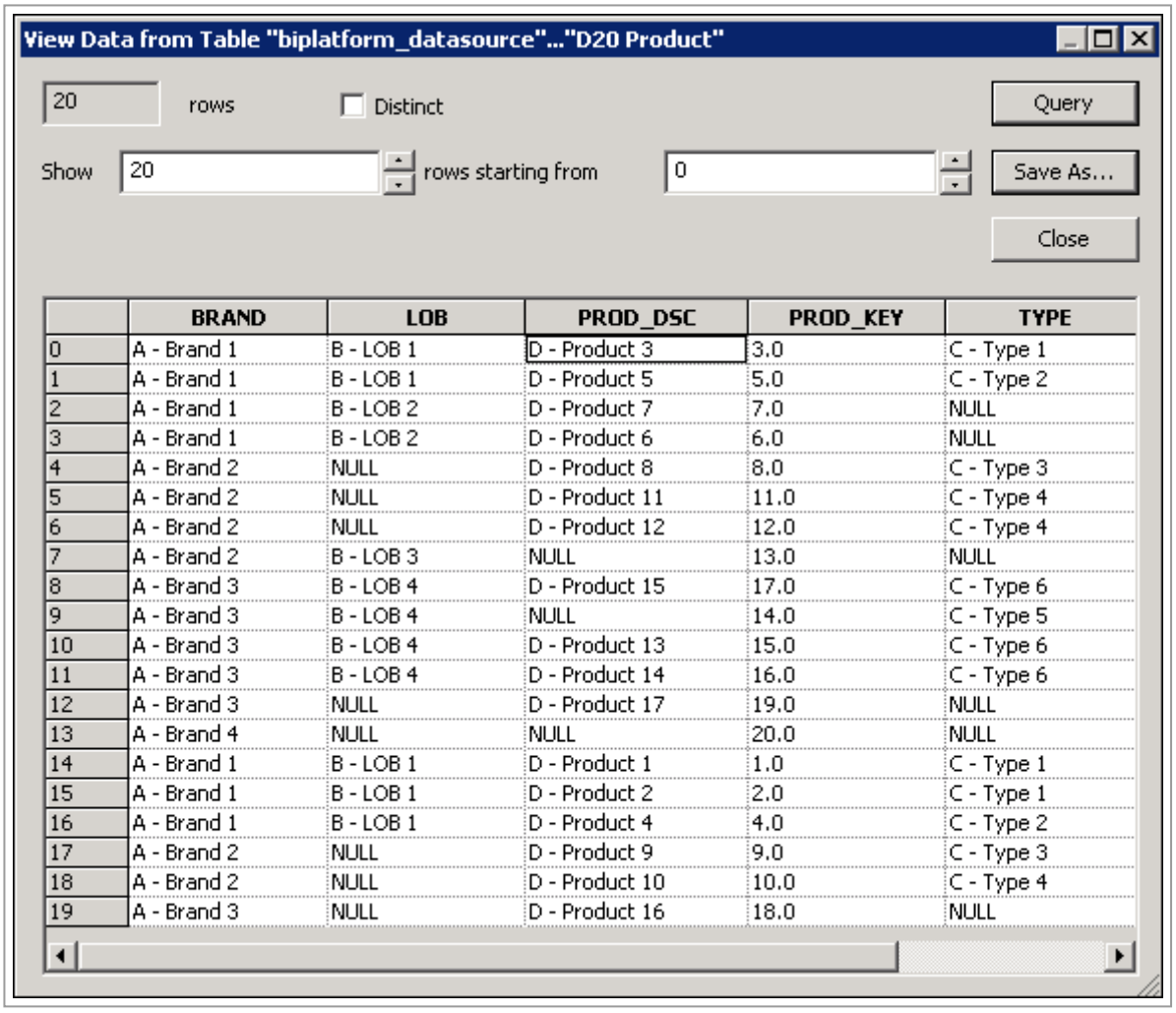



**12.** Use the Physical Diagram to create the following physical join for the alias table:

"biplatform_datasource"."".""."D20 Product"."PROD_KEY" = "biplatform_datasource"."".""."F1 Revenue"."PROD_KEY"




**13.** Right-click **D20 Product** and select **View Data**.



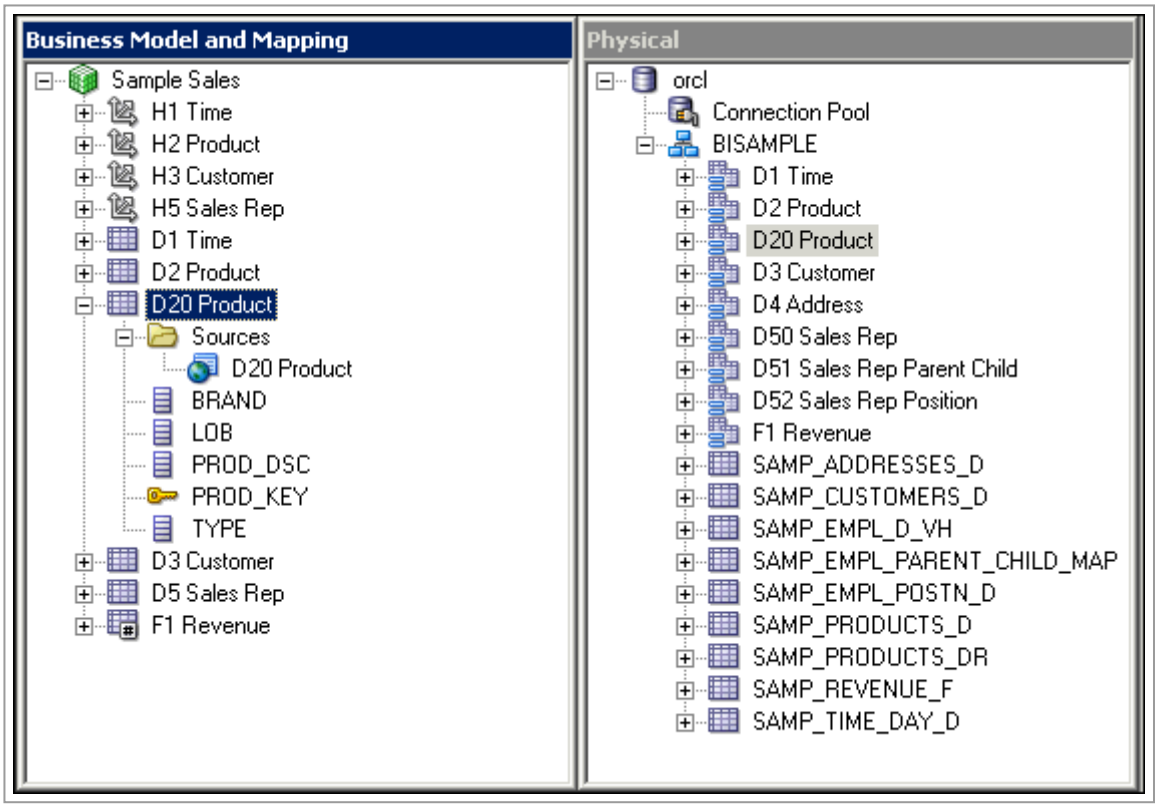

| | BRAND | LOB | PROD_DSC | PROD_KEY | TYPE |
|---|---|---|---|---|---|
| 0 | A - Brand 1 | B - LOB 1 | D - Product 3 | 3.0 | C - Type 1 |
| 1 | A - Brand 1 | B - LOB 1 | D - Product 5 | 5.0 | C - Type 2 |
| 2 | A - Brand 1 | B - LOB 2 | D - Product 7 | 7.0 | NULL |
| 3 | A - Brand 1 | B - LOB 2 | D - Product 6 | 6.0 | NULL |
| 4 | A - Brand 2 | NULL | D - Product 8 | 8.0 | C - Type 3 |
| 5 | A - Brand 2 | NULL | D - Product 11 | 11.0 | C - Type 4 |
| 6 | A - Brand 2 | NULL | D - Product 12 | 12.0 | C - Type 4 |
| 7 | A - Brand 2 | B - LOB 3 | NULL | 13.0 | NULL |
| 8 | A - Brand 3 | B - LOB 4 | D - Product 15 | 17.0 | C - Type 6 |
| 9 | A - Brand 3 | B - LOB 4 | NULL | 14.0 | C - Type 5 |
| 10 | A - Brand 3 | B - LOB 4 | D - Product 13 | 15.0 | C - Type 6 |
| 11 | A - Brand 3 | B - LOB 4 | D - Product 14 | 16.0 | C - Type 6 |
| 12 | A - Brand 3 | NULL | D - Product 17 | 19.0 | NULL |
| 13 | A - Brand 4 | NULL | NULL | 20.0 | NULL |
| 14 | A - Brand 1 | B - LOB 1 | D - Product 1 | 1.0 | C - Type 1 |
| 15 | A - Brand 1 | B - LOB 1 | D - Product 2 | 2.0 | C - Type 1 |
| 16 | A - Brand 1 | B - LOB 1 | D - Product 4 | 4.0 | C - Type 2 |
| 17 | A - Brand 2 | NULL | D - Product 9 | 9.0 | C - Type 3 |
| 18 | A - Brand 2 | NULL | D - Product 10 | 10.0 | C - Type 4 |
| 19 | A - Brand 3 | NULL | D - Product 16 | 18.0 | NULL |

Notice there are skipped levels in the hierarchy. For example, brand A - Brand2 has a NULL value for LOB for the product D - Product 8.

14. Close View Data.

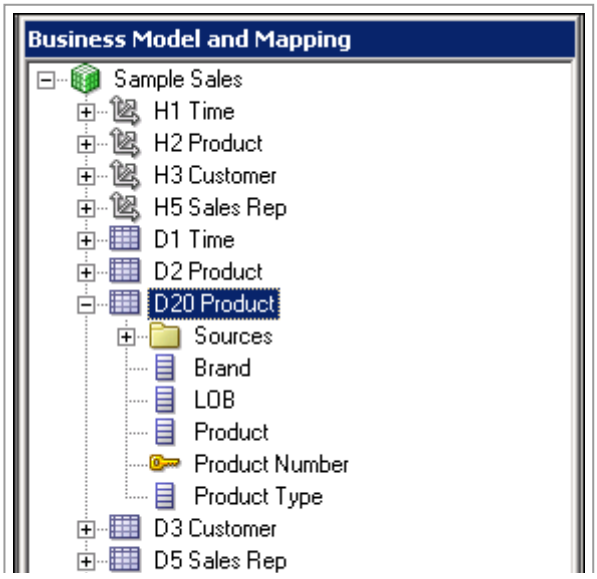
## Creating Logical Table and Logical Columns

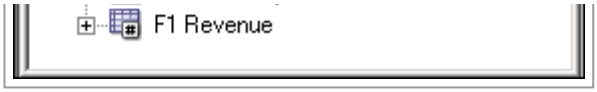
1. Drag **D20 Product** from the Physical layer to the **Sample Sales** business model in the **BMM** layer to create a **D20 Product** logical table. The logical join to F1 Revenue is created automatically based on the join in the **Physical** layer.



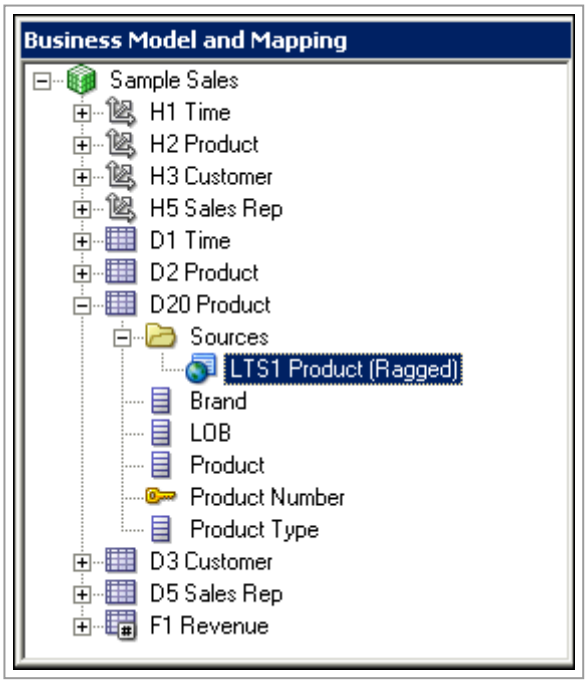

**2.** Rename the **D20 Product** logical columns:

| Old Name | New Name |
|---|---|
| BRAND | Brand |
| LOB | LOB |
| PROD_DSC | Product |
| PROD_KEY | Product Number |
| Type | Product Type |



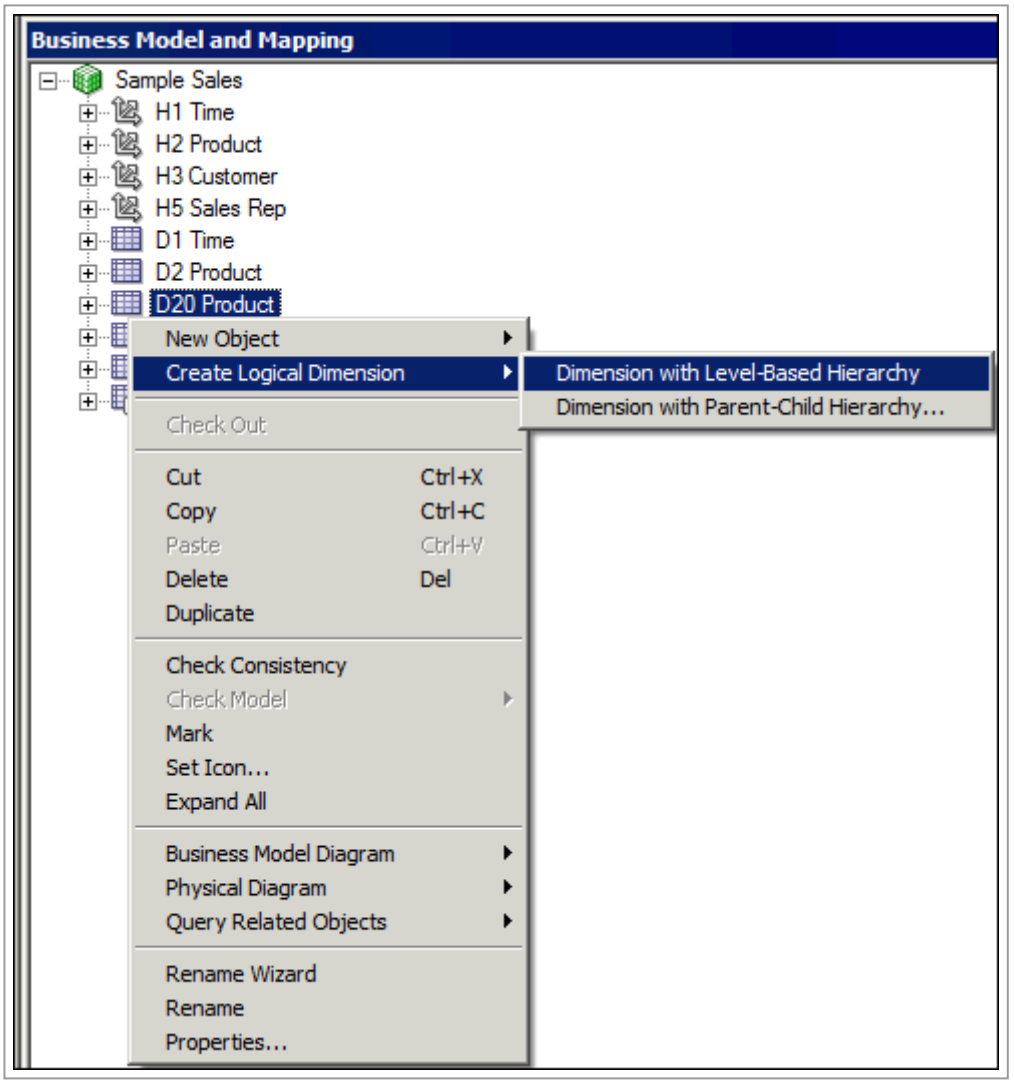


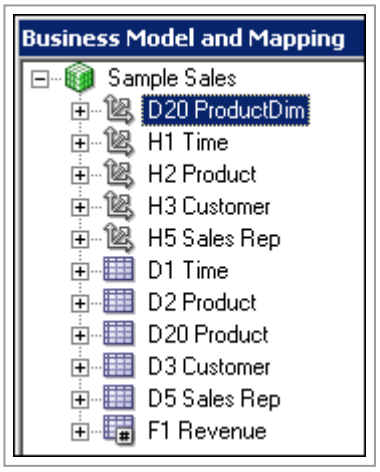

3. Rename the **D20 Product** logical table source to **LTS1 Product (Ragged)**.



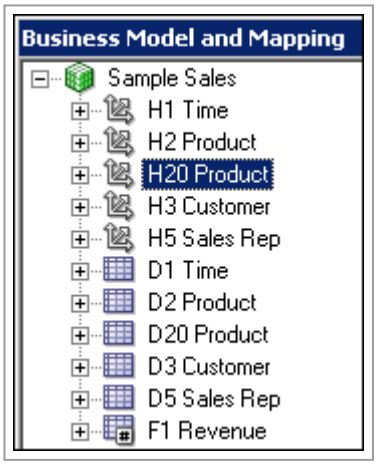

## Creating a Ragged/Skipped Levels Logical Dimension

1. Right-click the **D20 Product** logical table and select **Create Logical Dimension > Dimension with Level-Based Hierarchy** to automatically create a logical dimension named **D20 ProductDim**.

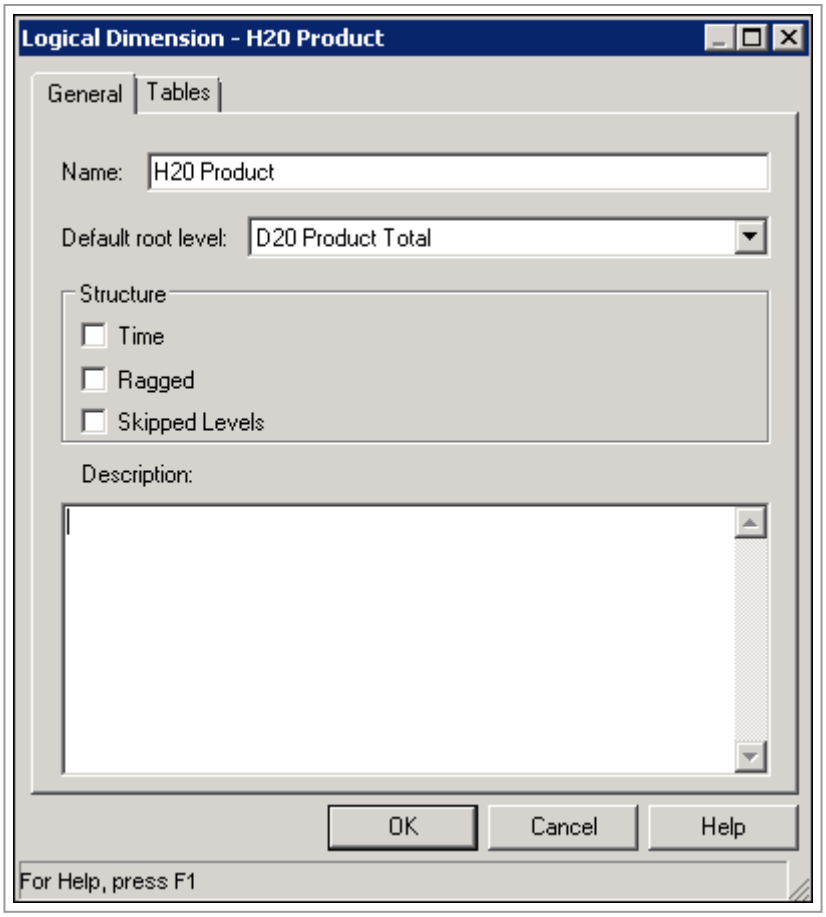


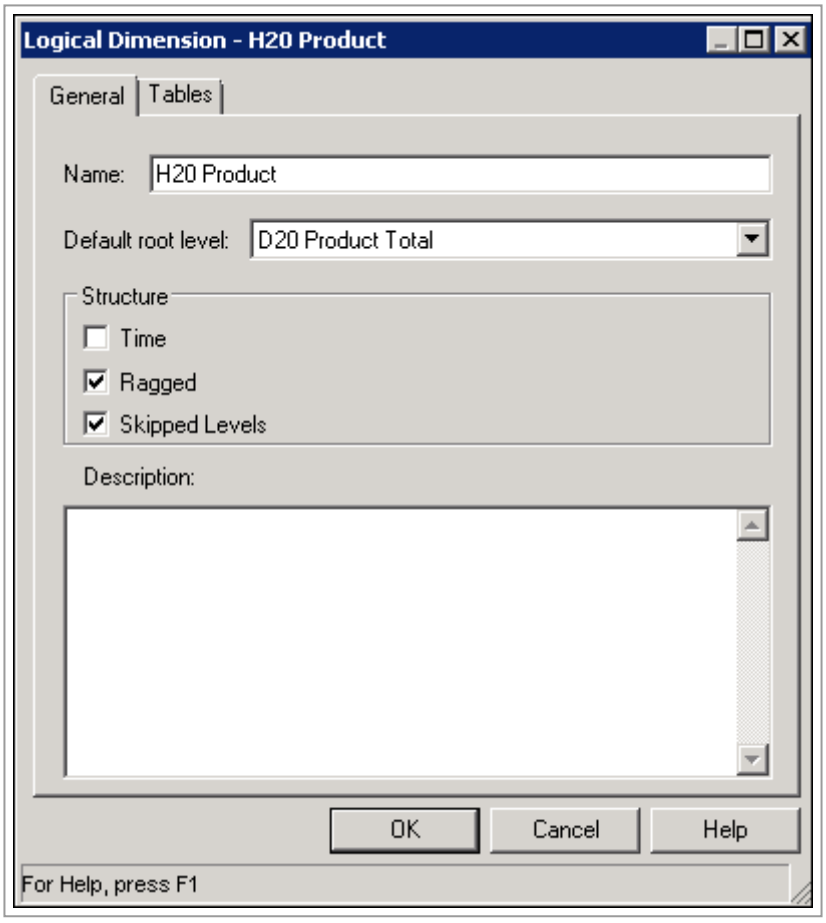

**2.** Rename **D20 ProductDim** to **H20 Product**.




**3.** Double-click the **H20 Product** logical dimension to open the Logical Dimension dialog box.

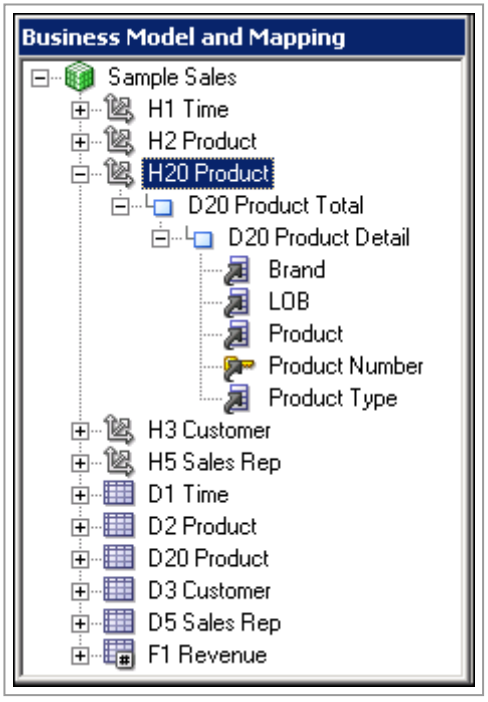



**4.** On the **General** tab, select both **Ragged** and **Skipped Levels**.




**5.** Click **OK** to close the Logical Dimension dialog box.

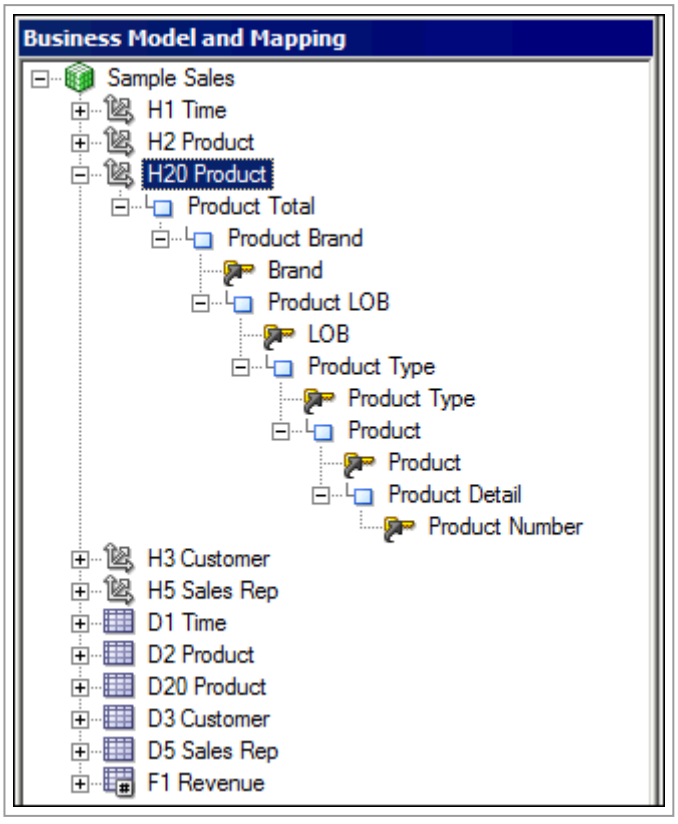
**6.** Expand **H20 Product**.




**7.** Create the following hierarchy:

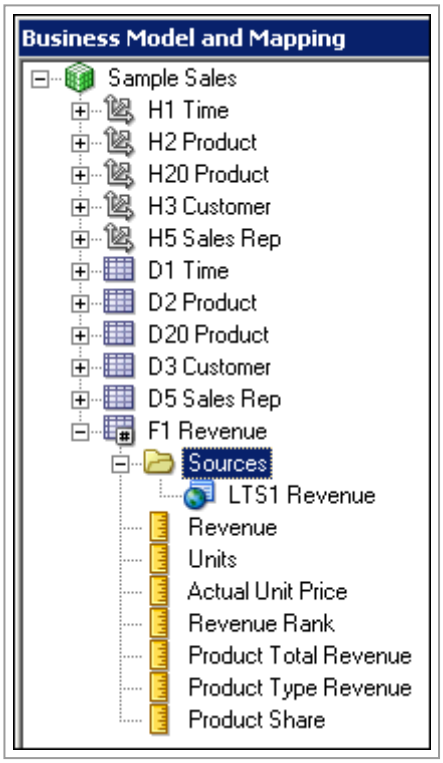
| Level | Column | Key | Use for |
|---|---|---|---|

| | | | **Display** |
|---|---|---|---|
| Product Total | <none> | <none> | <none> |
| Product Brand | Brand | Brand | Yes |
| Product LOB | LOB | LOB | Yes |
| Product Type | Product Type | Product Type | Yes |
| Product | Product | Product | Yes |
| Product Detail | Product Number | Product Number | Yes |

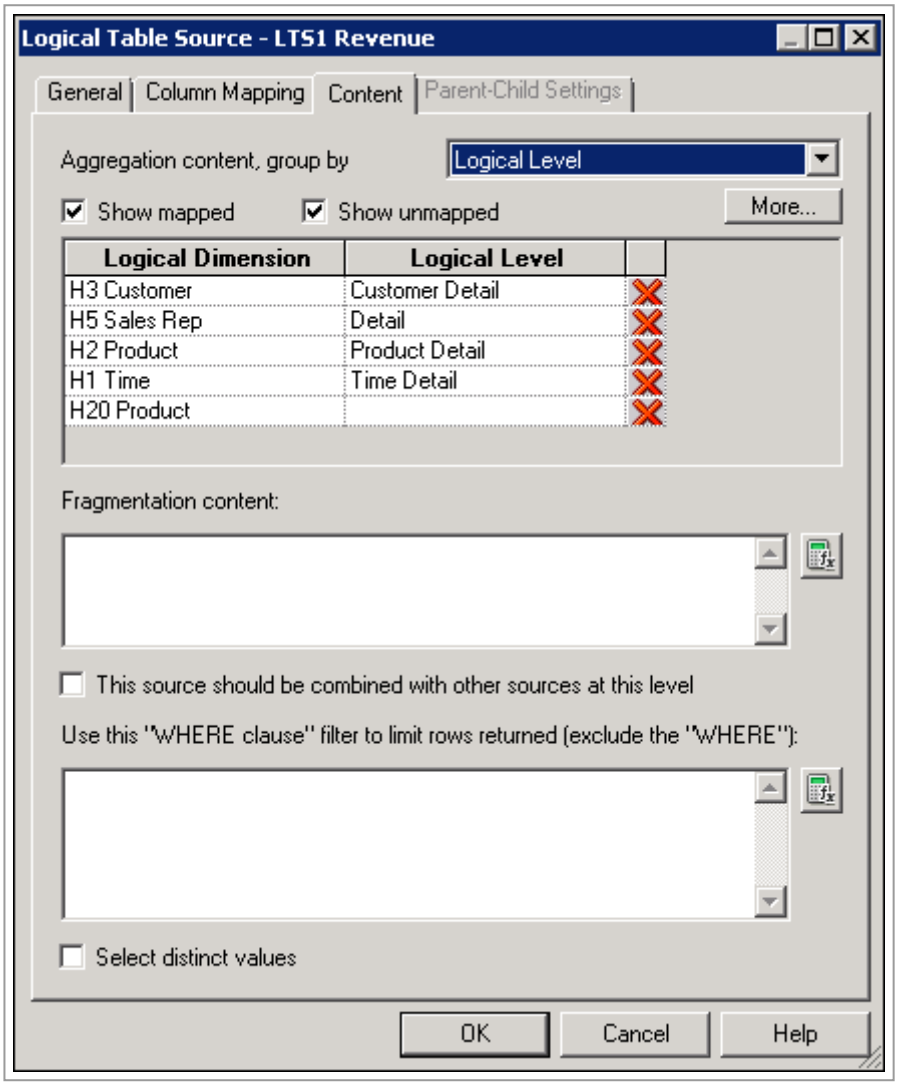



## Setting Aggregation Content

1. Expand **F1 Revenue > Sources**.

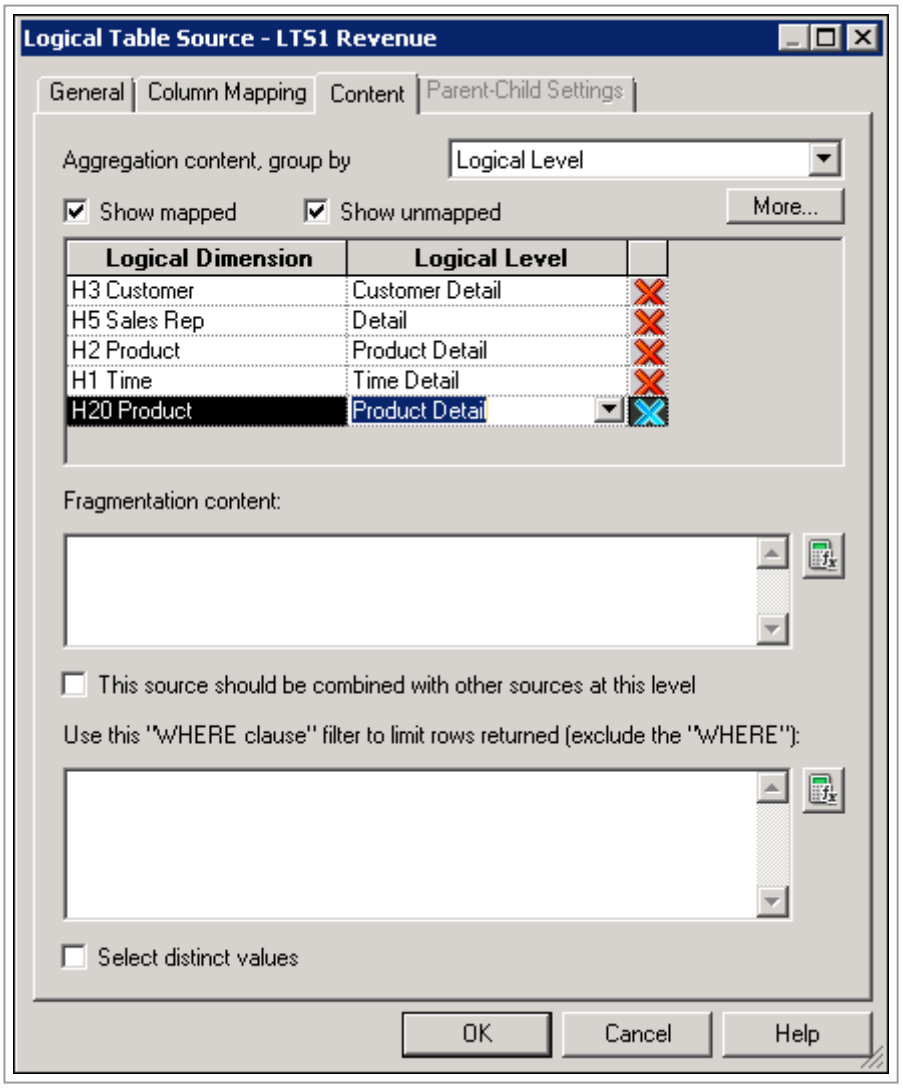



**2.** Double-click the **LTS1 Revenue** logical table source to open the Logical Table Source dialog box.



3. Click the **Content** tab.

4. Confirm that **Aggregation content, group by** is set to **Logical Level** and the logical level is set to **Product Detail** for the **H20 Product** logical dimension.

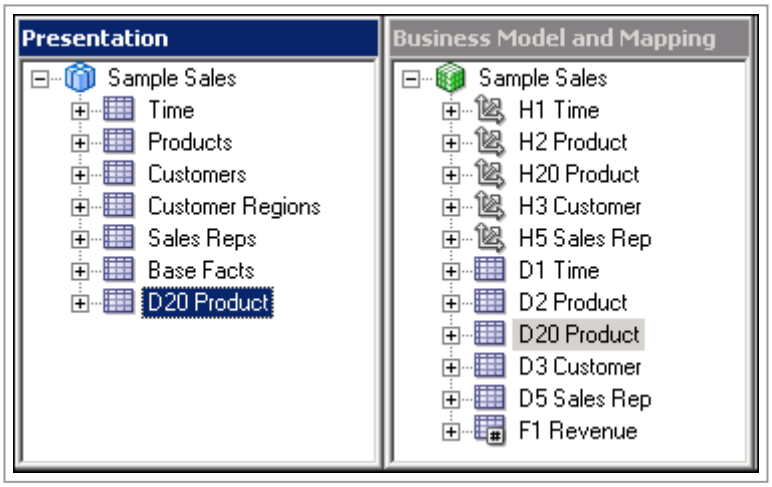



5. Click **OK** to close the Logical Table Source dialog box.

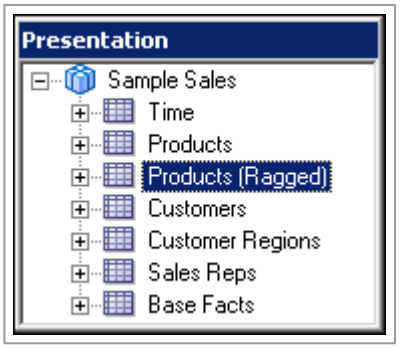
6. Save the repository and check global consistency. Fix any errors or warnings before proceeding.

## Creating Presentation Layer Objects

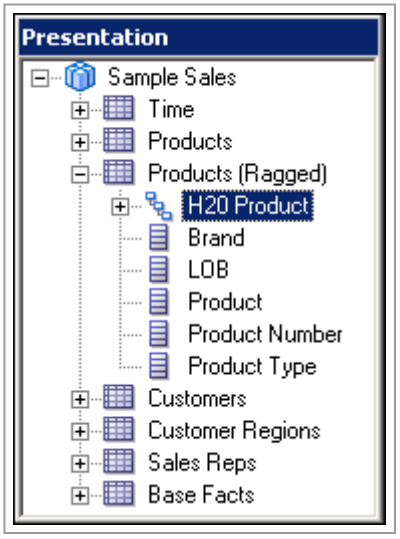
1. Drag the **D20 Product** logical table to the **Sample Sales** subject area in the **Presentation** layer.




2. In the Presentation layer, rename **D20 Product** to **Products (Ragged)** and move **Products (Ragged)** to appear after **Products**.




3. Expand **Products (Ragged)** and notice that the **H20 Product** logical dimension is automatically added to the **Presentation** layer.




4. Save the repository and check consistency. Fix any errors or warnings before proceeding.

5. Close the repository. Leave the Administration Tool open.

## Testing Your Work

1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

2. Return to **Oracle BI**, which should still be open, and sign in.

3. Create the following analysis to test the ragged / skipped level hierarchy:

   Products (Ragged).Brand
   Products (Ragged).LOB
   Products (Ragged).Product Type
   Products (Ragged).Product
   Base Facts.Revenue


4. Click **Results**.

| Brand | LOB | Product Type | Product | Revenue |
|---|---|---|---|---|
| A - Brand 1 | B - LOB 1 | C - Type 1 | D - Product 1 | 1,014,224 |
| | | | D - Product 2 | 2,476,985 |
| | | | D - Product 3 | 4,938,884 |
| | | C - Type 2 | D - Product 4 | 3,995,040 |
| | | | D - Product 5 | 2,756,523 |
| | B - LOB 2 | | D - Product 6 | 3,013,045 |
| | | | D - Product 7 | 3,740,065 |
| A - Brand 2 | B - LOB 3 | | | 518,288 |
| | | C - Type 3 | D - Product 8 | 3,657,417 |
| | | | D - Product 9 | 2,604,358 |
| | | C - Type 4 | D - Product 10 | 2,363,155 |
| | | | D - Product 11 | 1,210,103 |
| | | | D - Product 12 | 1,330,399 |

   The results display correctly even though there are skipped levels (levels with NULL values) and ragged levels (leaves with varying depth).

5. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Using Aggregates

In this set of steps you set up and use aggregate tables to improve query performance. Aggregate tables

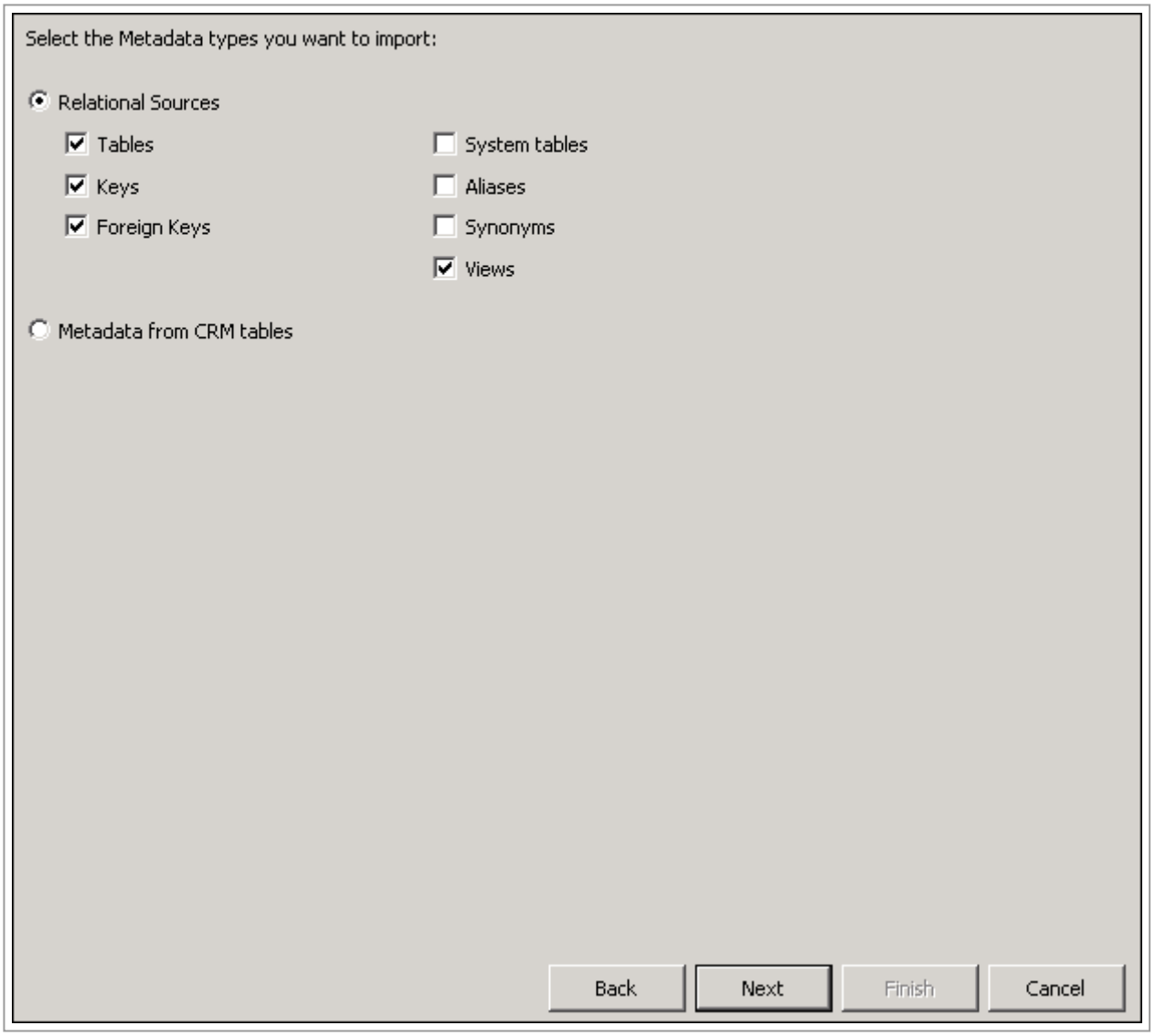
In this set of steps you set up and use aggregate tables to improve query performance. Aggregate tables store pre-computed results, which are measures that have been aggregated (typically summed) over a set of dimensional attributes. Using aggregate tables is a popular technique for speeding up query response times in decision support systems. This eliminates the need for run-time calculations and delivers faster results to users. The calculations are done ahead of time and the results are stored in the tables. Aggregate tables typically have many fewer rows than the non-aggregate tables and, therefore, processing is faster.

To set up and use aggregate tables, perform the following steps:

- Importing Metadata
- Creating New Logical Table Sources
- Setting Aggregate Content
- Testing Your Work

## Importing Metadata

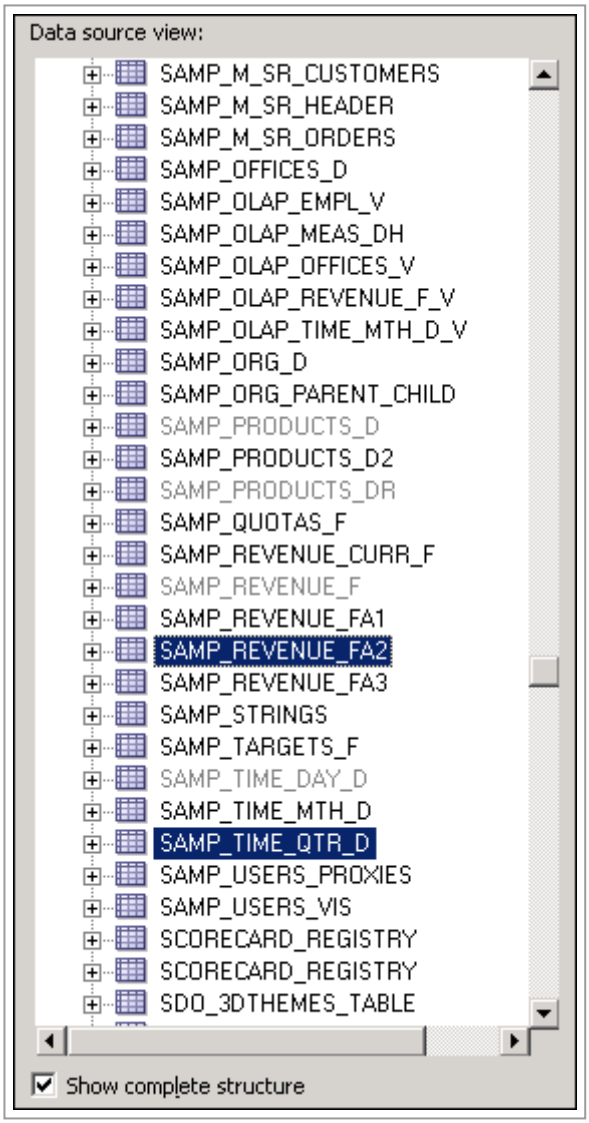
1. Return to the Administration Tool and open the **BISAMPLE** repository in offline mode.
2. In the Physical layer, expand **biplatform_datasource**.
3. Right-click **Connection Pool** and select **Import Metadata** to open the Import Wizard.
4. In the Select Metadata Types screen, select **Views** and click **Next**.




**5.** In the Select Metadata Objects screen, in the data source view, expand **BISAMPLE**.

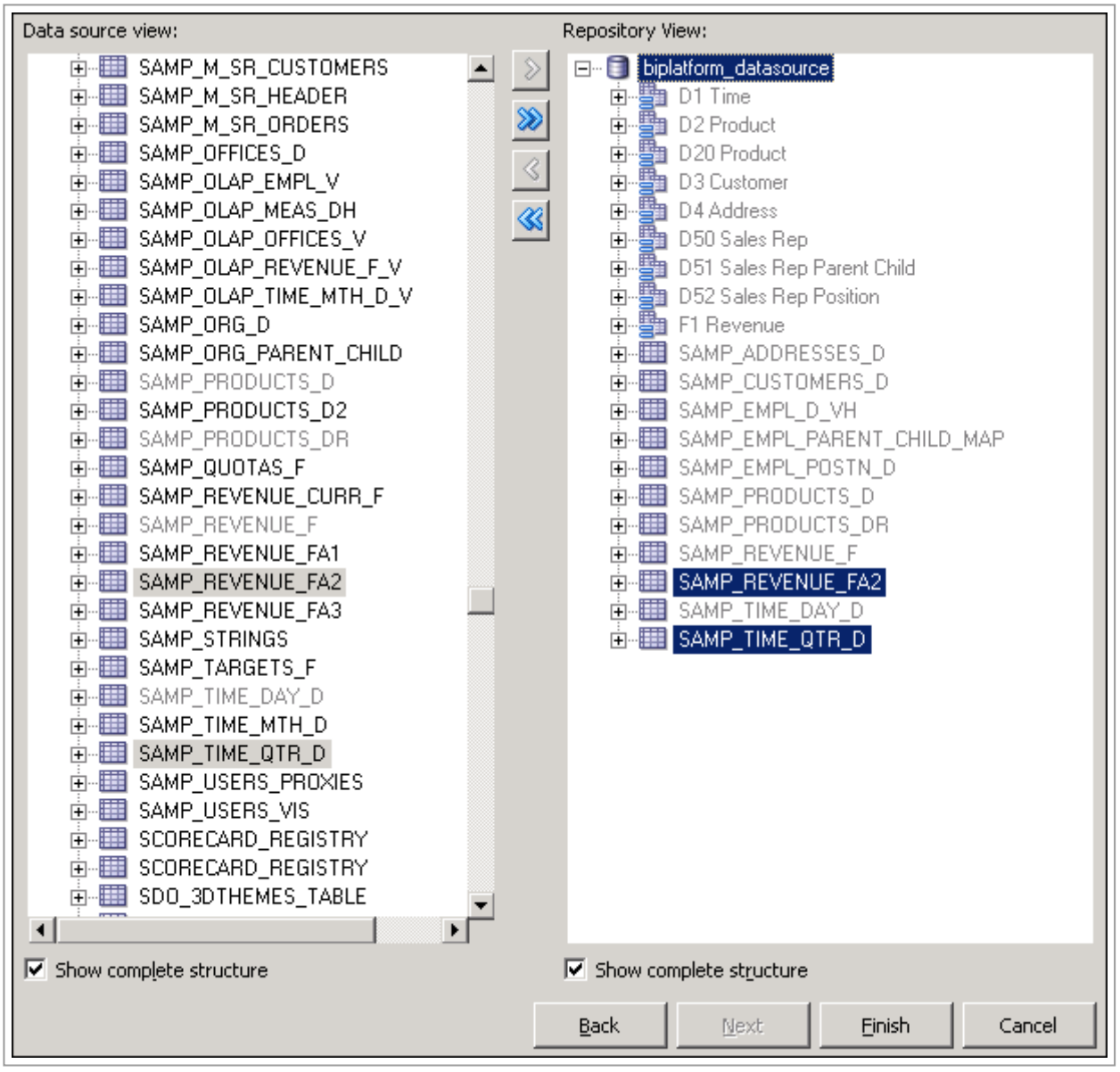
**6.** In the data source view, select the following for import:

**SAMP_REVENUE_FA2**
**SAMP_TIME_QTR_D**



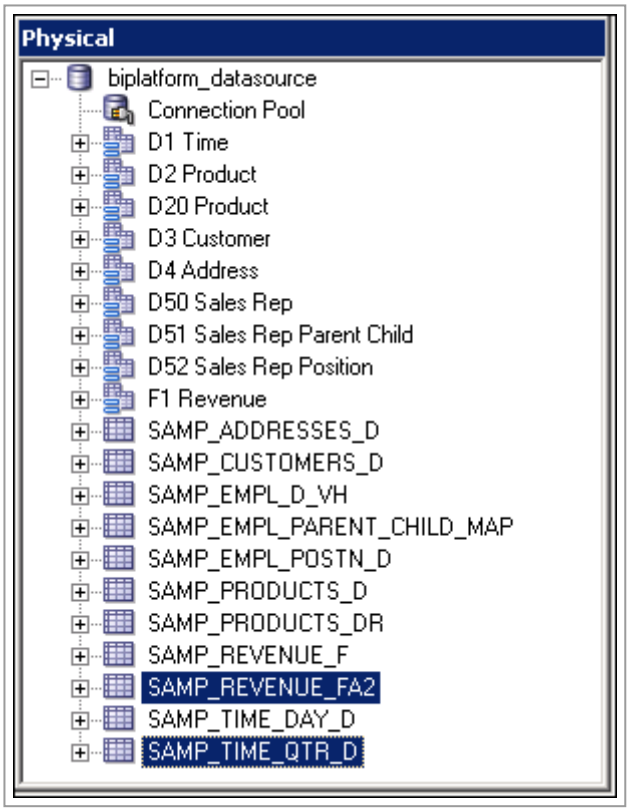

**7.** Click the **Import Selected button** to move the objects to the Repository View.

**8.** Expand **BISAMPLE** in the Repository View and confirm that the objects are visible.



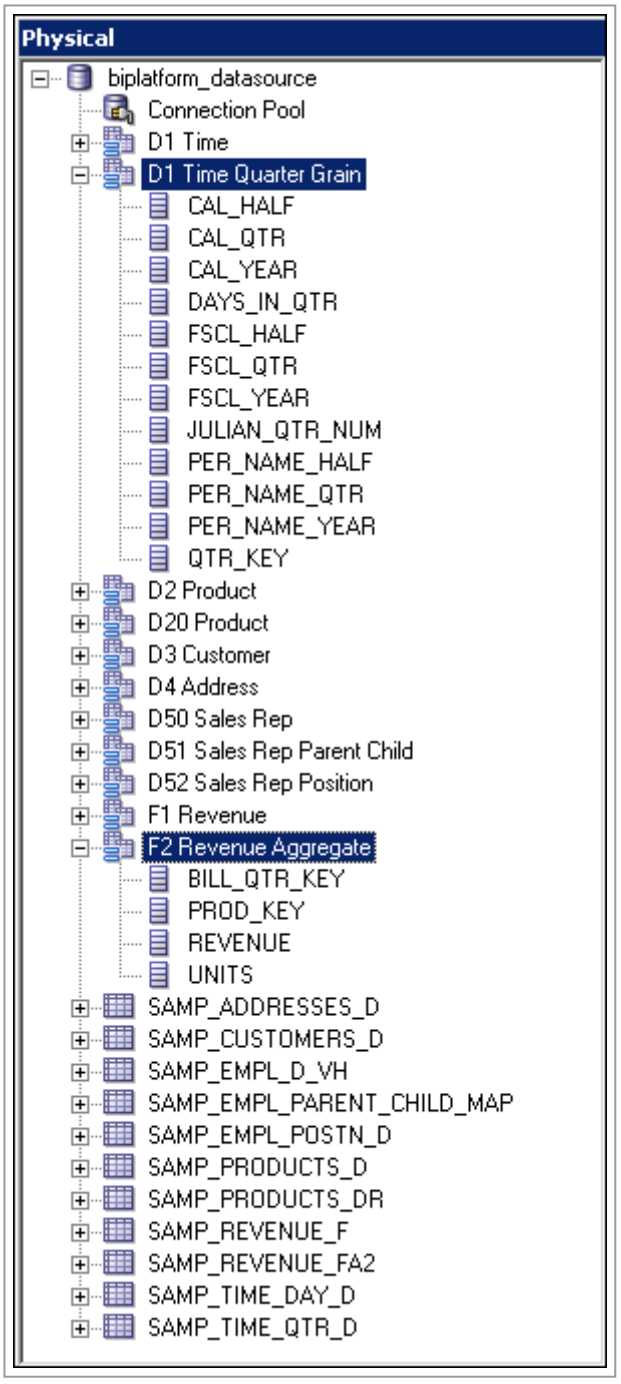

**9.** Click **Finish** to close the Import Wizard.

**10.** Confirm that the objects are visible in the Physical layer of the repository.

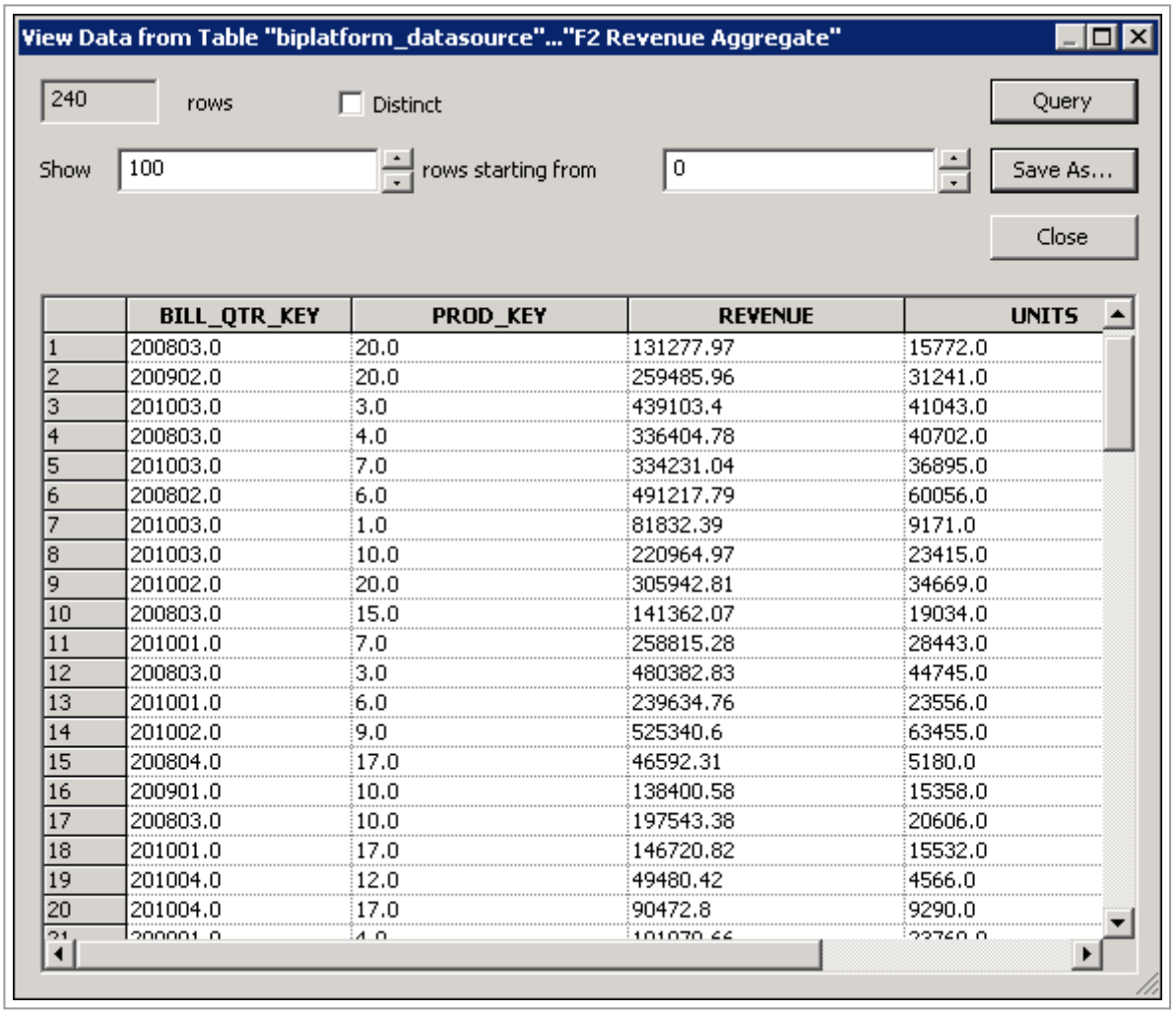



**11.** Create the following aliases:

| Table | Alias |
|---|---|
| SAMP_REVENUE_FA2 | F2 Revenue Aggregate |
| SAMP_TIME_QTR_D | D1 Time Quarter Grain |

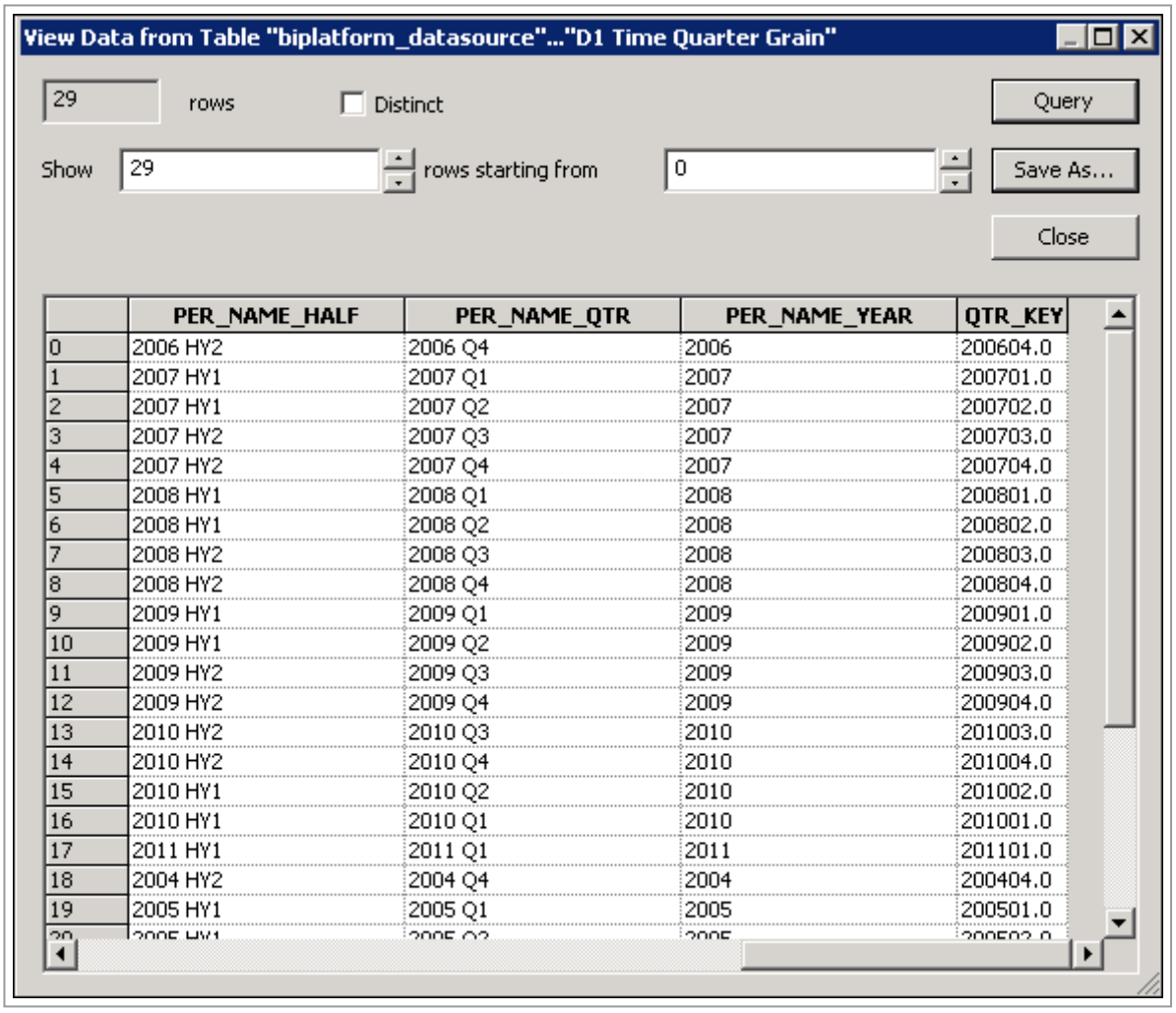



12. Right-click **F2 Revenue Aggregate** and select **View Data**. F2 Revenue Aggregate stores aggregated fact information for revenue and units at the quarter and product grain.




| | BILL_QTR_KEY | PROD_KEY | REVENUE | UNITS |
|---|---|---|---|---|
| 1 | 200803.0 | 20.0 | 131277.97 | 15772.0 |
| 2 | 200902.0 | 20.0 | 259485.96 | 31241.0 |
| 3 | 201003.0 | 3.0 | 439103.4 | 41043.0 |
| 4 | 200803.0 | 4.0 | 336404.78 | 40702.0 |
| 5 | 201003.0 | 7.0 | 334231.04 | 36895.0 |
| 6 | 200802.0 | 6.0 | 491217.79 | 60056.0 |
| 7 | 201003.0 | 1.0 | 81832.39 | 9171.0 |
| 8 | 201003.0 | 10.0 | 220964.97 | 23415.0 |
| 9 | 201002.0 | 20.0 | 305942.81 | 34669.0 |
| 10 | 200803.0 | 15.0 | 141362.07 | 19034.0 |
| 11 | 201001.0 | 7.0 | 258815.28 | 28443.0 |
| 12 | 200803.0 | 3.0 | 480382.83 | 44745.0 |
| 13 | 201001.0 | 6.0 | 239634.76 | 23556.0 |
| 14 | 201002.0 | 9.0 | 525340.6 | 63455.0 |
| 15 | 200804.0 | 17.0 | 46592.31 | 5180.0 |
| 16 | 200901.0 | 10.0 | 138400.58 | 15358.0 |
| 17 | 200803.0 | 10.0 | 197543.38 | 20606.0 |
| 18 | 201001.0 | 17.0 | 146720.82 | 15532.0 |
| 19 | 201004.0 | 12.0 | 49480.42 | 4566.0 |
| 20 | 201004.0 | 17.0 | 90472.8 | 9290.0 |

13. Right-click **D1 Time Quarter Grain** and select **View Data**. D1 Time Quarter Grain stores time data at the quarter grain. It stores one record for each quarter beginning with Q4 2006 and ending with Q4 2011.

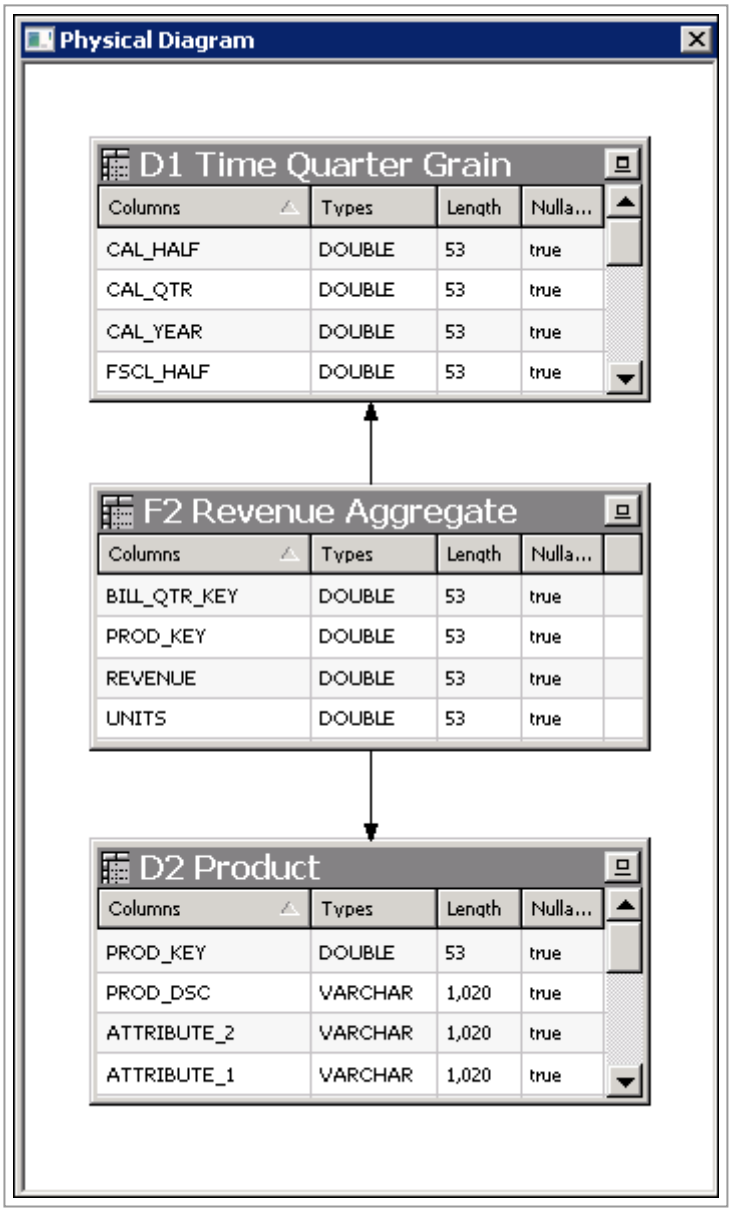



| | PER_NAME_HALF | PER_NAME_QTR | PER_NAME_YEAR | QTR_KEY |
|---|---|---|---|---|
| 0 | 2006 HY2 | 2006 Q4 | 2006 | 200604.0 |
| 1 | 2007 HY1 | 2007 Q1 | 2007 | 200701.0 |
| 2 | 2007 HY1 | 2007 Q2 | 2007 | 200702.0 |
| 3 | 2007 HY2 | 2007 Q3 | 2007 | 200703.0 |
| 4 | 2007 HY2 | 2007 Q4 | 2007 | 200704.0 |
| 5 | 2008 HY1 | 2008 Q1 | 2008 | 200801.0 |
| 6 | 2008 HY1 | 2008 Q2 | 2008 | 200802.0 |
| 7 | 2008 HY2 | 2008 Q3 | 2008 | 200803.0 |
| 8 | 2008 HY2 | 2008 Q4 | 2008 | 200804.0 |
| 9 | 2009 HY1 | 2009 Q1 | 2009 | 200901.0 |
| 10 | 2009 HY1 | 2009 Q2 | 2009 | 200902.0 |
| 11 | 2009 HY2 | 2009 Q3 | 2009 | 200903.0 |
| 12 | 2009 HY2 | 2009 Q4 | 2009 | 200904.0 |
| 13 | 2010 HY2 | 2010 Q3 | 2010 | 201003.0 |
| 14 | 2010 HY2 | 2010 Q4 | 2010 | 201004.0 |
| 15 | 2010 HY1 | 2010 Q2 | 2010 | 201002.0 |
| 16 | 2010 HY1 | 2010 Q1 | 2010 | 201001.0 |
| 17 | 2011 HY1 | 2011 Q1 | 2011 | 201101.0 |
| 18 | 2004 HY2 | 2004 Q4 | 2004 | 200404.0 |
| 19 | 2005 HY1 | 2005 Q1 | 2005 | 200501.0 |

**14.** Use the Physical Diagram to create the following physical joins:

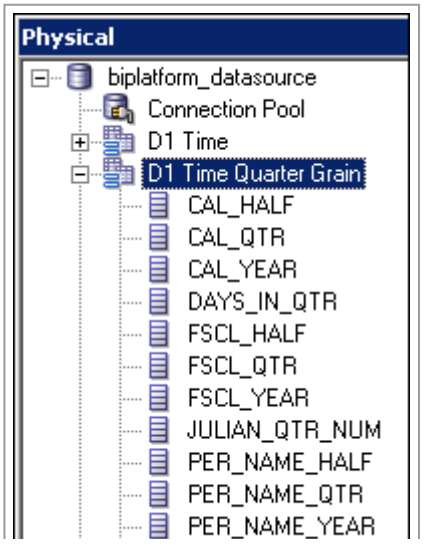
"biplatform_datasource"."".""."D2 Product"."PROD_KEY" = "biplatform_datasource"."".""."F2 Revenue Aggregate"."PROD_KEY"

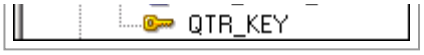
"biplatform_datasource"."".""."D1 Time Quarter Grain"."QTR_KEY" = "biplatform_datasource"."".""."F2 Revenue Aggregate"."BILL_QTR_KEY"



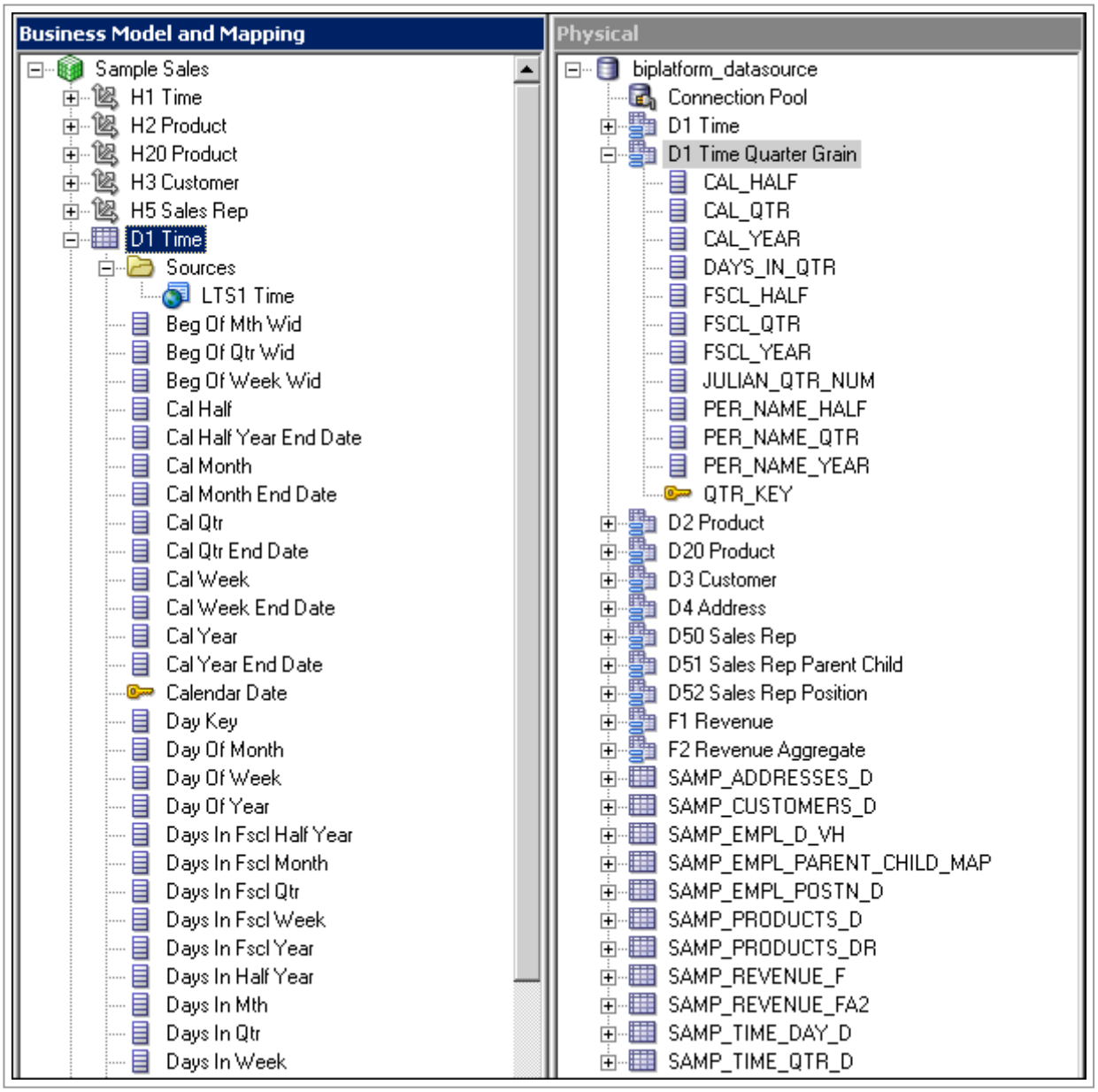

## Creating New Logical Table Sources

1. In the **Physical** layer, expand **D1 Time Quarter Grain**.






2. In the **BMM** layer, expand **D1 Time**.

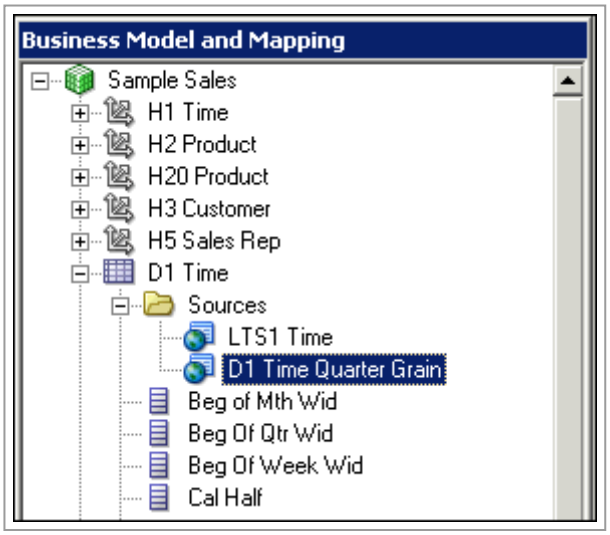



3. Drag the following columns from **D1 Time Quarter Grain** to their corresponding columns in **D1 Time**:
**Note**: Make sure to drag them to their **corresponding** columns.

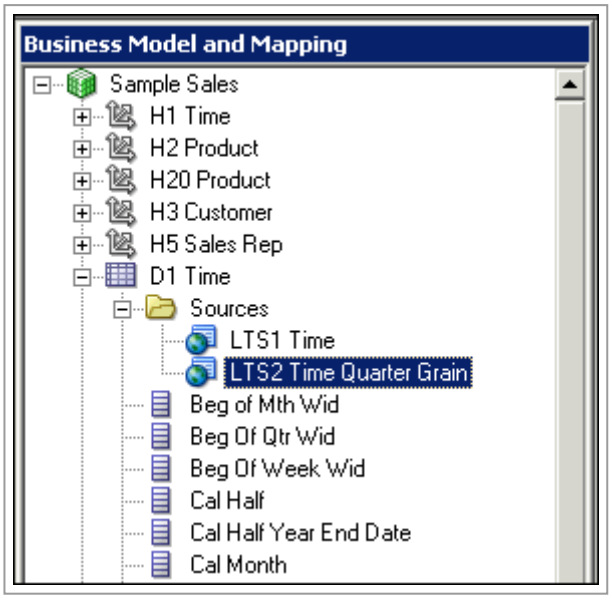
| D1 Time Quarter Grain | D1 Time |
|---|---|
| CAL_HALF | Cal Half |
| CAL_QTR | Cal Qtr |
| CAL_YEAR | Cal Year |
| DAYS_IN_QTR | Days in Qtr |
| JULIAN_QTR_NUM | Julian Qtr Num |
| PER_NAME_HALF | Per Name Half |
| PER_NAME_QTR | Per Name Qtr |

| PER_NAME_YEAR | Per Name Year |
|---|---|

This action creates a new logical table source named **D1 Time Quarter Grain** for D1 Time.

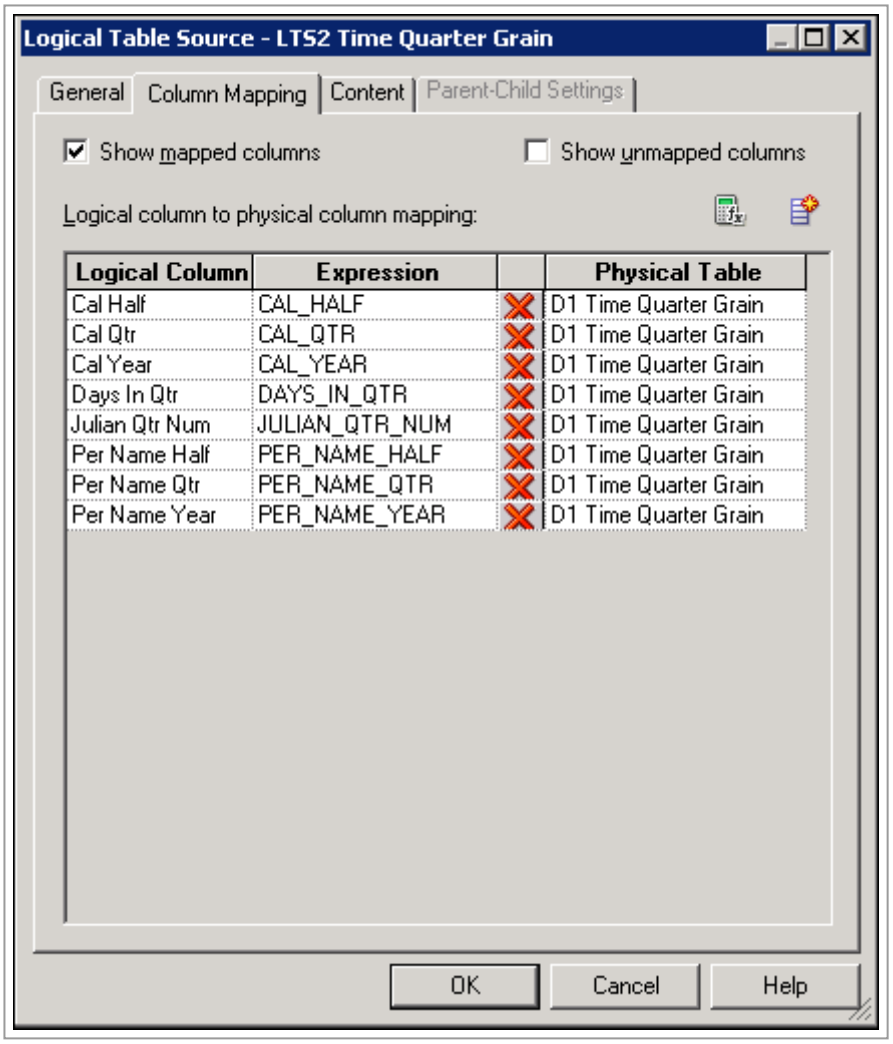



4. Rename the D1 Time Quarter Grain logical table source to **LTS2 Time Quarter Grain**.



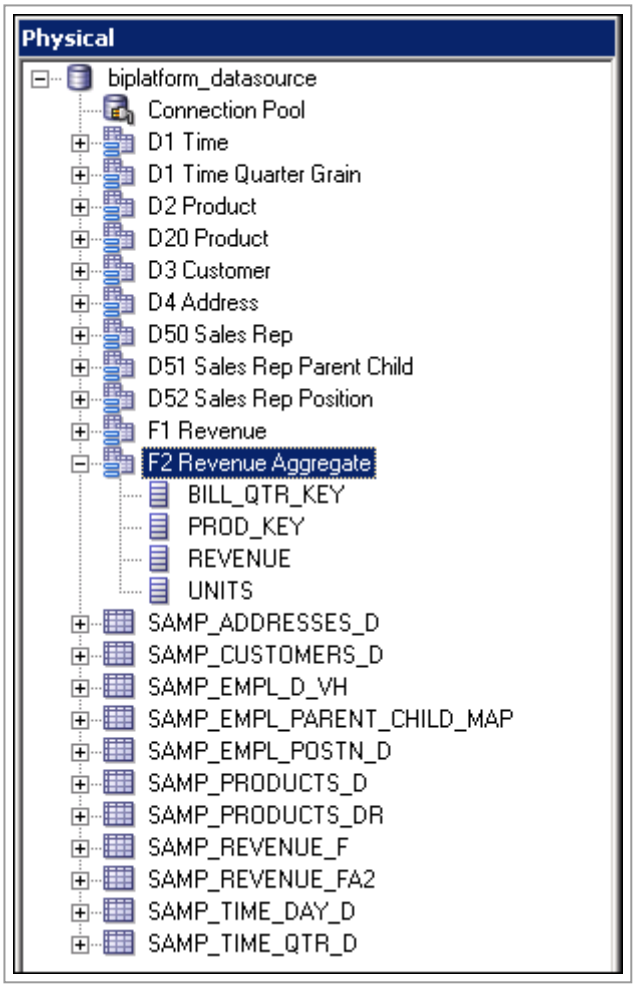

5. Double-click **LTS2 Time Quarter Grain** to open the Logical Table Source dialog box.

6. On the **Column Mapping** tab make sure **Show mapped columns** is selected and note the column mappings. The logical columns now map to columns in two physical tables: D1 Time and D1 Time Quarter Grain.

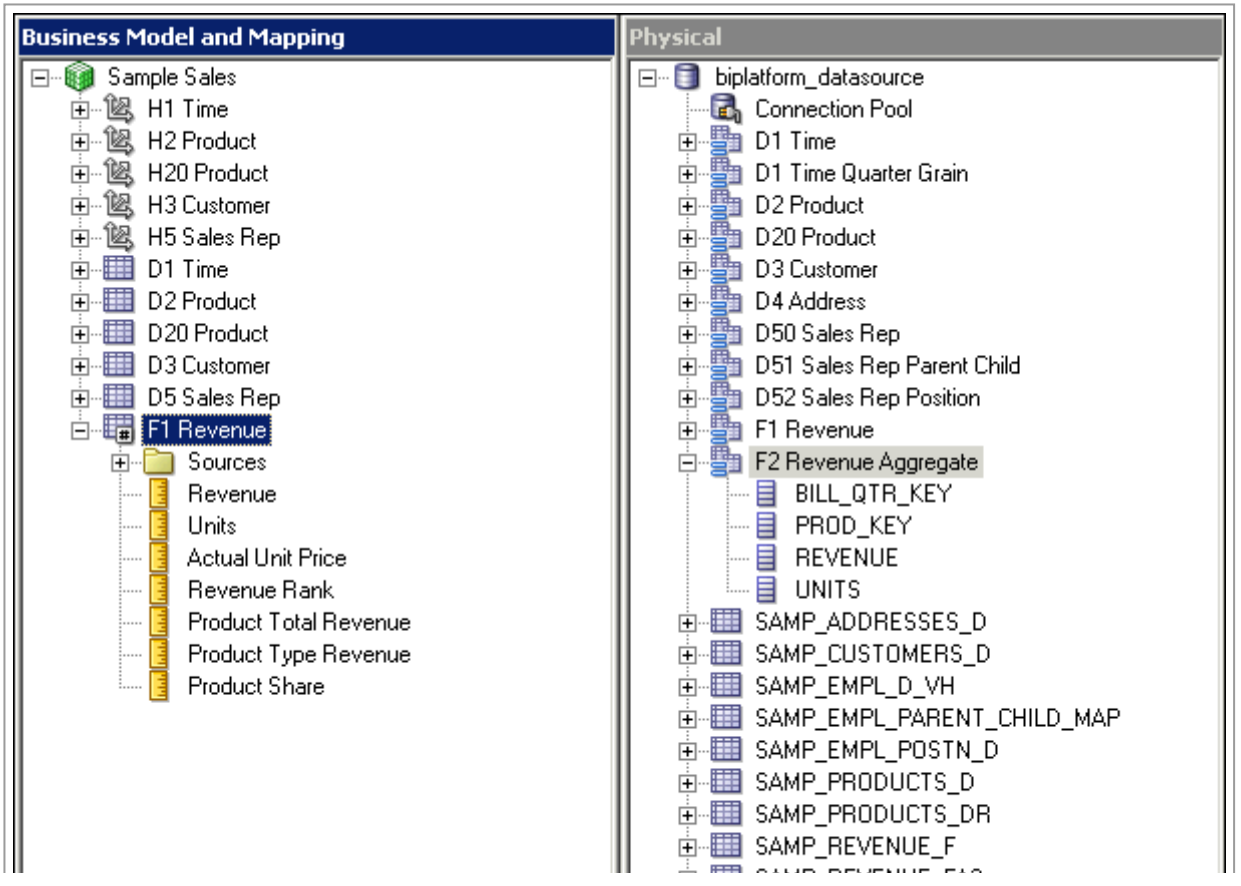


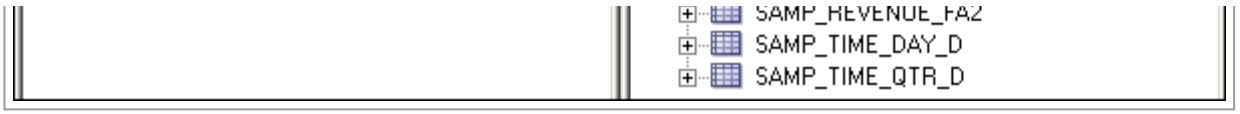

**7.** Click **OK** to close the Logical Table Source dialog box.

**8.** In the **Physical** layer expand **F2 Revenue Aggregate**.




**9.** In the **BMM** layer expand **F1 Revenue**.

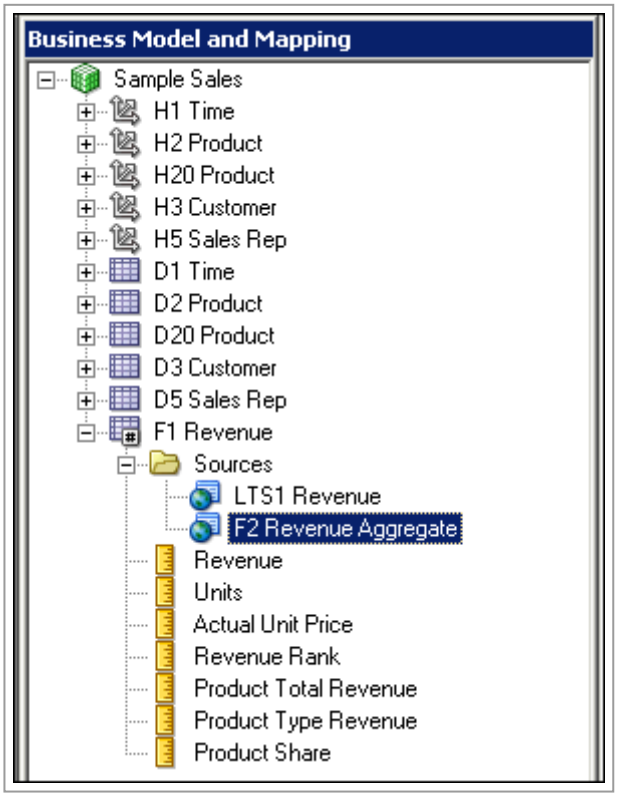




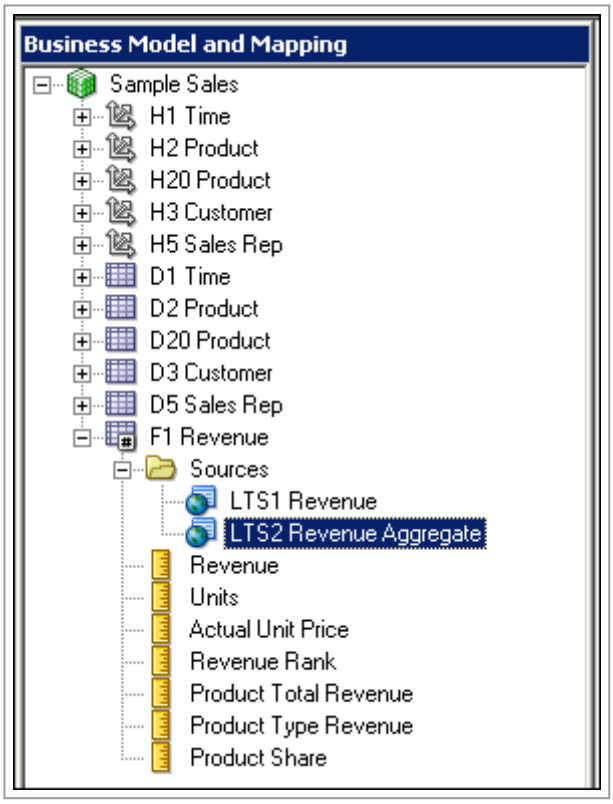

10. Drag the following physical columns from **F2 Revenue Aggregate** to their corresponding logical columns in **F1 Revenue**: **Note:** Do not add them as new columns.

| F2 Revenue Aggregate | F1 Revenue |
|---|---|
| UNITS | Units |
| REVENUE | Revenue |

This action creates a new logical table source named **F2 Revenue Aggregate** for F1 Revenue.

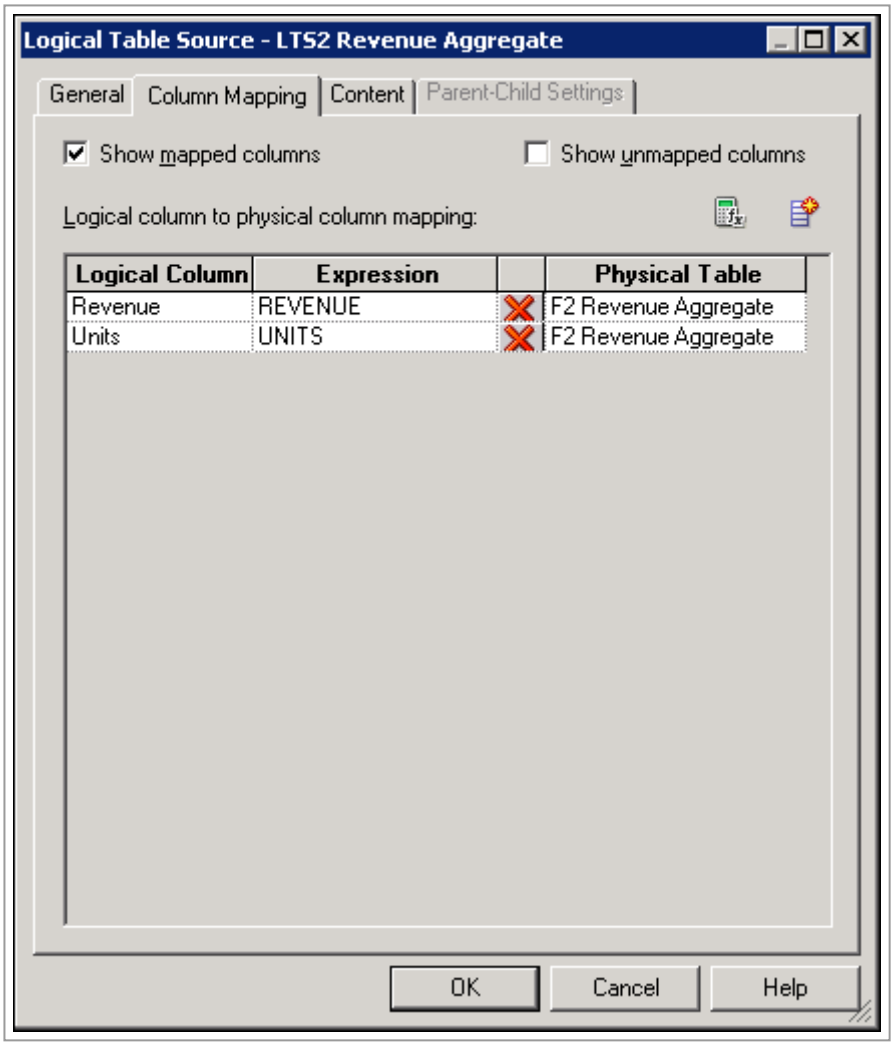



11. Rename the F2 Revenue Aggregate logical table source to **LTS2 Revenue Aggregate**.




12. Double-click **LTS2 Revenue Aggregate** to open the Logical Table Source dialog box.

13. On the Column Mappings tab make sure show mapped columns is selected and note the column mappings. The Revenue and Units logical columns now map to columns in two physical tables: F1 Revenue and F2 Revenue Aggregate.



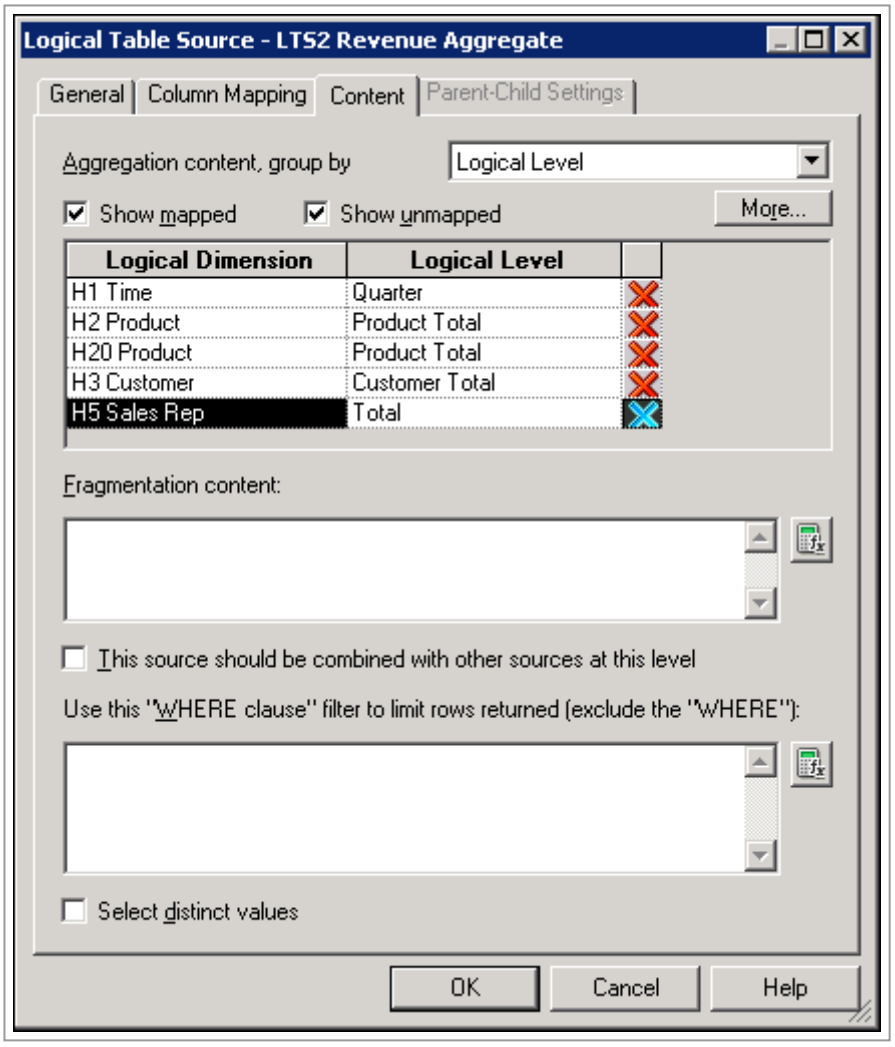

**14.** Leave the **Logical Table Source - LTS2 Revenue Aggregate** dialog box open.

## Setting Aggregate Content

**1.** Click the **Content** tab.

**2.** Set the following logical levels for the logical dimensions:

| Logical Dimension | Logical Level |
|---|---|
| H1 Time | Quarter |
| H2 Product | Product Total |
| H20 Product | Product Total |
| H3 Customer | Customer Total |
| H5 Sales Rep | Total |



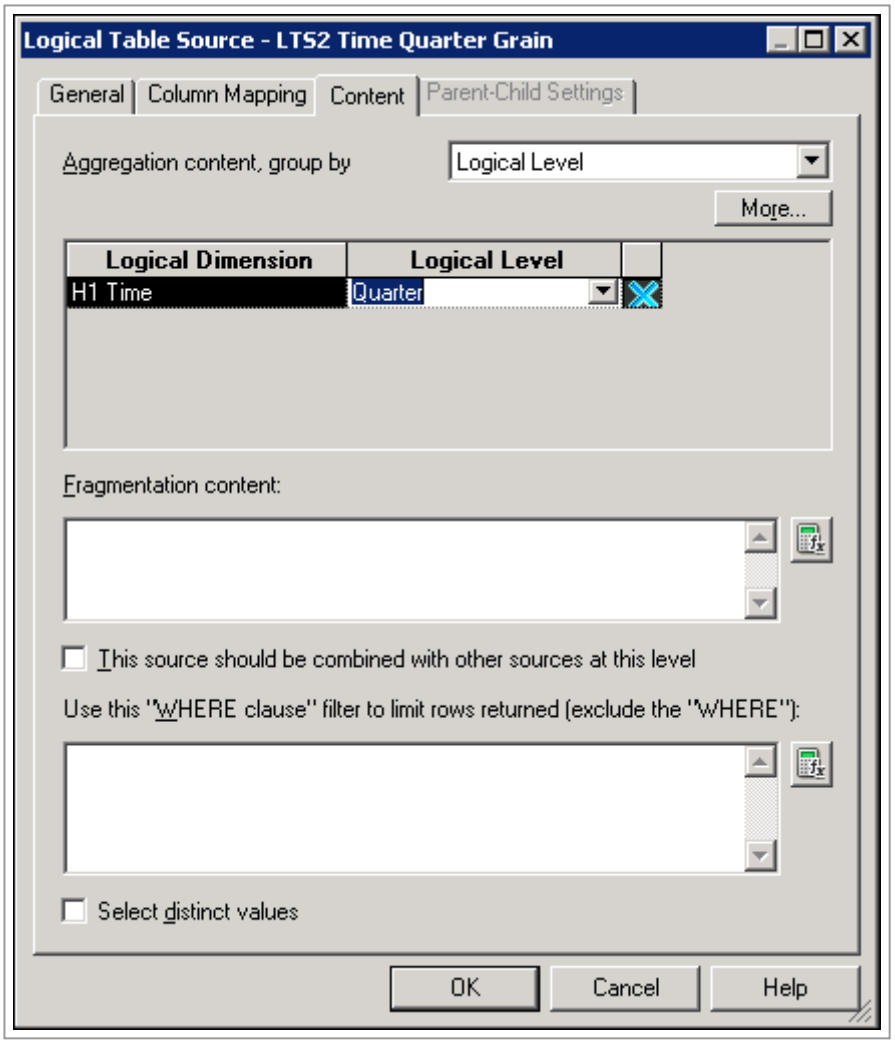

Explanation: You are setting aggregation content for the fact table to the corresponding levels in the dimension hierarchies. In a subsequent step, you set similar levels for the aggregate logical table source for the Time dimension. Note that all levels are set to the total level except for the H1 Time logical dimension, which is set to Quarter. The result is, when a user queries against a particular level, Oracle BI Server will “know” to access the aggregate tables instead of the detail tables.

For example, if a user queries for total sales by product by quarter, the server will access the F2 Revenue Aggregate fact table and the corresponding aggregate dimension table, D1 Time Quarter Grain. If a user queries for a level lower than the level specified here, for example Month instead of Quarter, then the server will access the detail tables (F1 Revenue and D1 Time). If a user queries for higher level (year instead of quarter) the aggregate tables will be used, because whenever a query is run against a logical level or above, the aggregate tables are used.

3. Click **OK** to close the Logical Table Source dialog box.

4. Double-click the **LTS2 Time Quarter Grain** logical table source to open the Logical Table Source dialog box.

5. On the Content tab, set the logical level to **Quarter**.




6. Click **OK** to close the Logical Table Source dialog box.

7. Save the repository and check global consistency. Fix any errors or warnings before proceeding.

8. Close the repository. Leave the Administration Tool open. Note that you did not need to change the **Presentation** layer. You made changes in the business model that impact how queries are processed and which sources are accessed. However, the user interface remains the same, so there is no need to change the Presentation layer. Oracle BI Server will automatically use the appropriate sources based on the user query.

## Testing Your Work

1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

2. Return to **Oracle BI**, which should still be open, and sign in.

3. Create the following analysis to test the aggregate tables.

   **Time.Per Name Qtr**
   **Base Facts.Revenue**

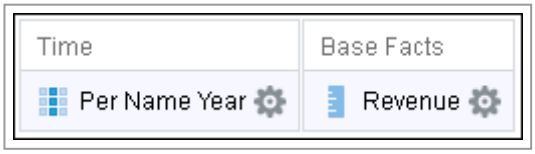

4. Click **Results**.

| Per Name Qtr | Revenue |
|---|---|
| 2008 Q1 | 2,707,686 |
| 2008 Q2 | 8,109,716 |
| 2008 Q3 | 4,338,844 |
| 2008 Q4 | 1,343,754 |
| 2009 Q1 | 2,535,988 |
| 2009 Q2 | 7,646,006 |
| 2009 Q3 | 3,596,955 |
| 2009 Q4 | 1,221,051 |
| 2010 Q1 | 3,238,167 |
| 2010 Q2 | 9,171,760 |
| 2010 Q3 | 4,564,476 |
| 2010 Q4 | 1,525,598 |

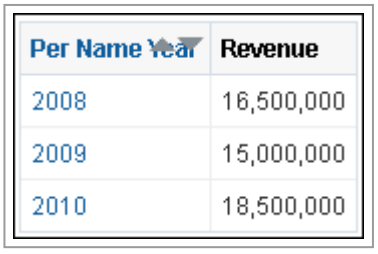
5. Click **New > Analysis > Sample Sales**.

6. Create the following analysis to test the aggregate tables.

**Time.Per Name Year**
**Base Facts.Revenue**




7. Click **Results**.

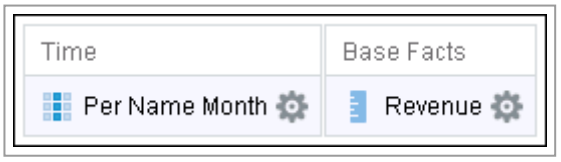


| Per Name Year | Revenue |
|---|---|
| 2008 | 16,500,000 |
| 2009 | 15,000,000 |
| 2010 | 18,500,000 |

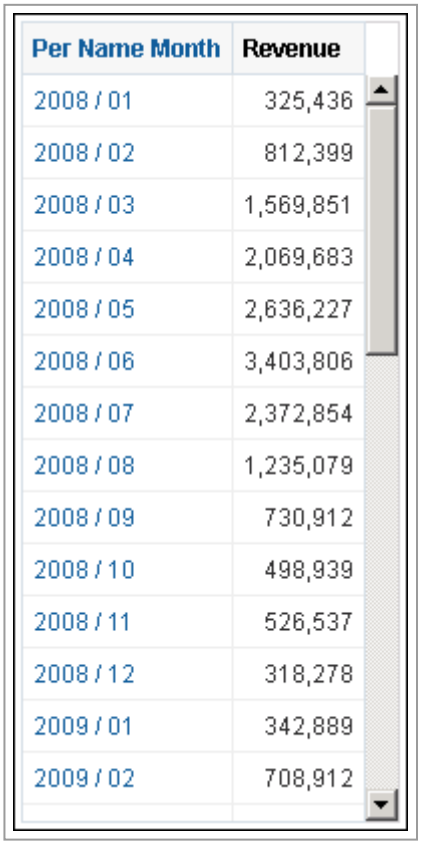
8. Click **New > Analysis > Sample Sales**.

9. Create one more analysis to test the aggregate tables.

**Time.Per Name Month**
**Base Facts.Revenue**




10. Click **Results**.



| Per Name Month | Revenue |
|---|---|
| 2008 / 01 | 325,436 |
| 2008 / 02 | 812,399 |
| 2008 / 03 | 1,569,851 |
| 2008 / 04 | 2,069,683 |
| 2008 / 05 | 2,636,227 |
| 2008 / 06 | 3,403,806 |
| 2008 / 07 | 2,372,854 |
| 2008 / 08 | 1,235,079 |
| 2008 / 09 | 730,912 |
| 2008 / 10 | 498,939 |
| 2008 / 11 | 526,537 |
| 2008 / 12 | 318,278 |
| 2009 / 01 | 342,889 |
| 2009 / 02 | 708,912 |

11. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Using Initialization Blocks and Variables

You can use variables in a repository to streamline administrative tasks and modify metadata content dynamically to adjust to a changing data environment. A variable has a single value at any point in time. Variables can be used instead of literals or constants in the Expression Builder in the Administration Tool or in end-user analyses. At run time, Oracle BI Server substitutes the value of the variable.

In this set of steps you create a new Initialization Block - Current Periods, and three new Dynamic Repository Variables - CurrentYear, CurrentMonth, and CurrentDay. You then use the variables as column filters in an Oracle BI analysis. You use the Variable Manager in the Administration Tool to define variables and initialization blocks.

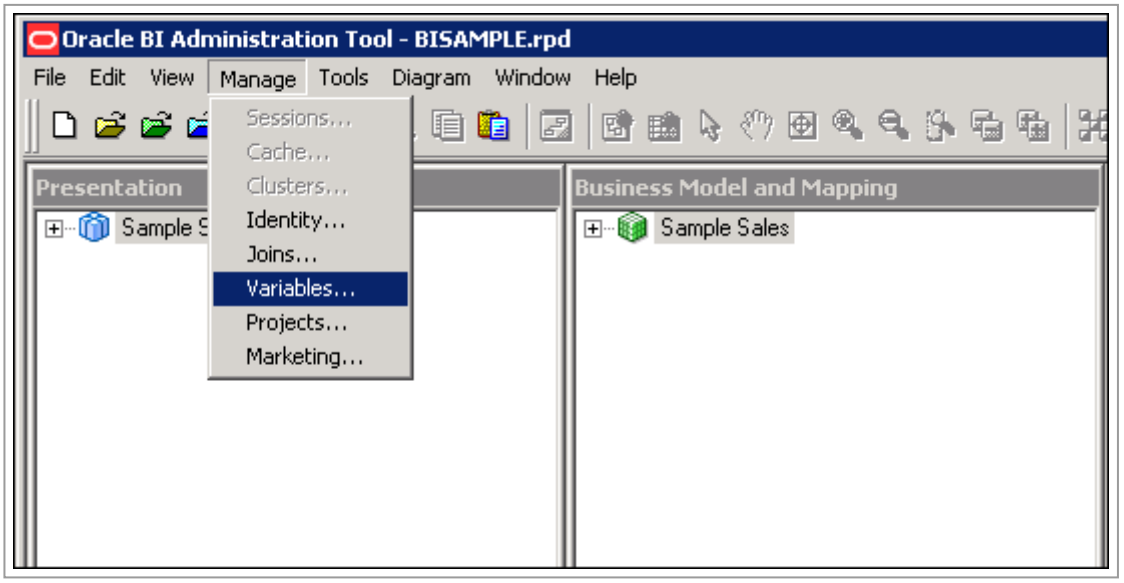
To set up and use initialization blocks and variables, perform the following steps:

- Creating an Initialization Block
- Creating Variables
- Testing Your Work

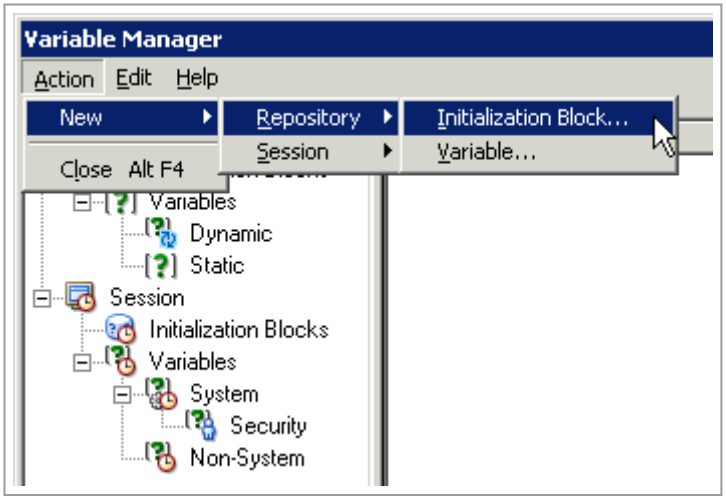
## Creating an Initialization Block

1. Open the **BISAMPLE** repository in offline mode.

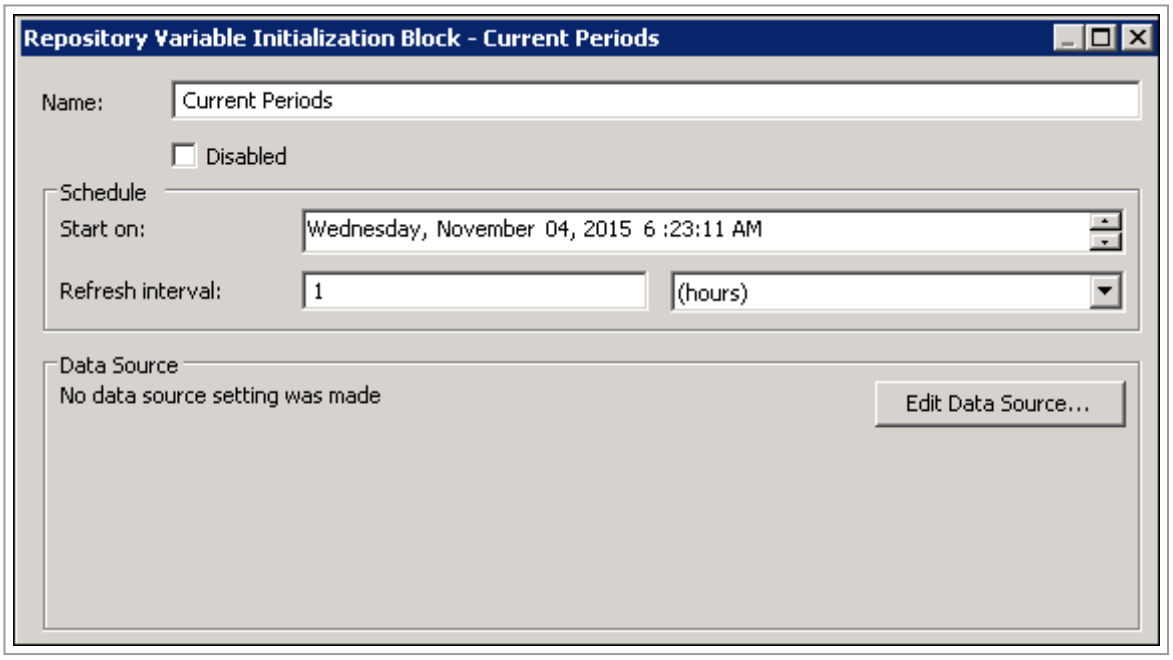
2. Select **Manage > Variables** to open the Variable Manager.



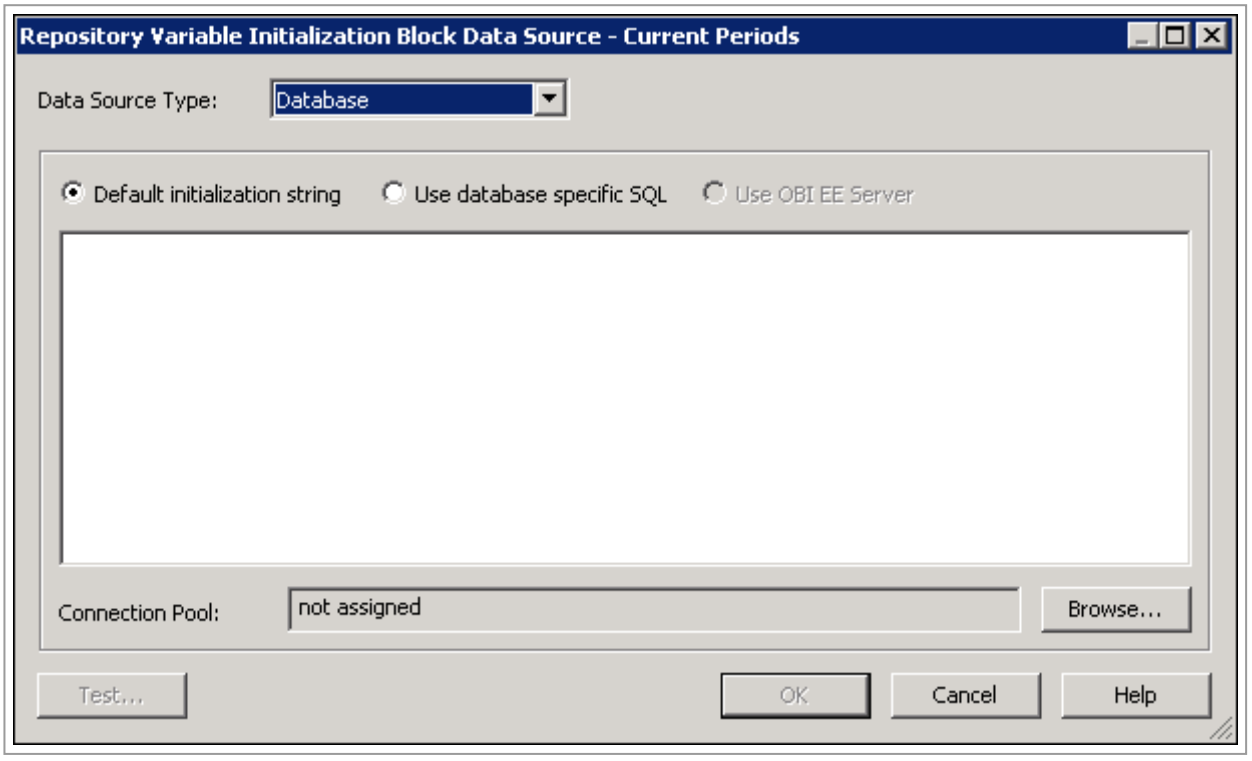

3. Select **Action > New > Repository > Initialization Block**.



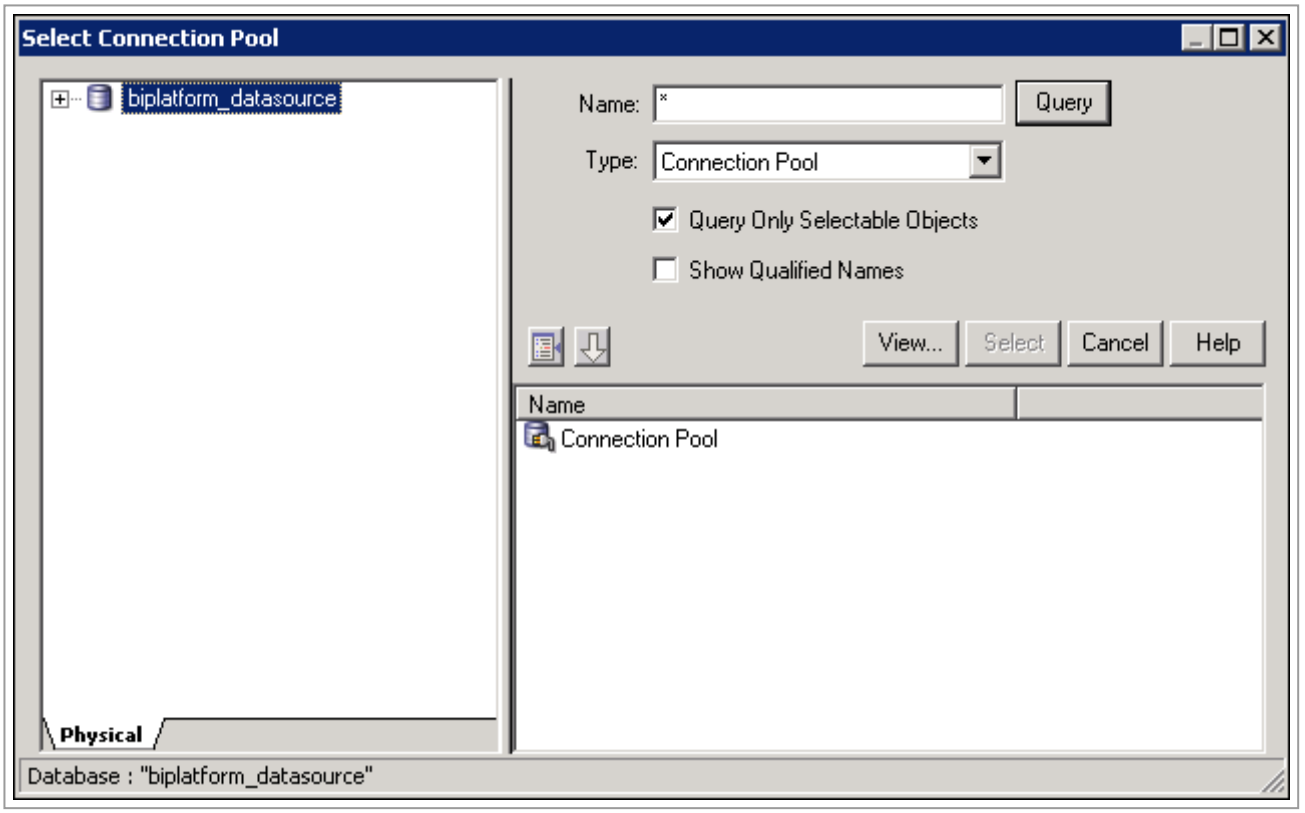

4. Name the initialization block **Current Periods**.



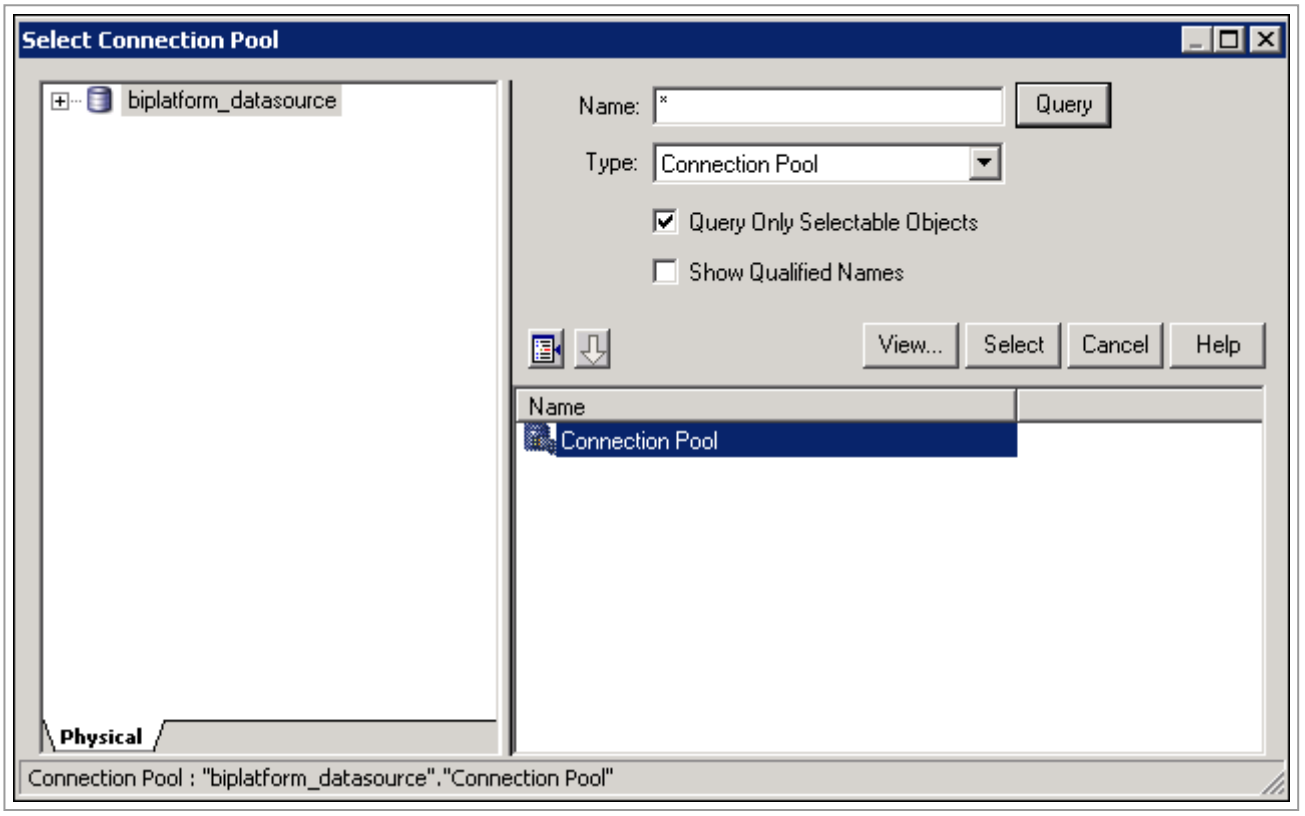

5. Click the **Edit Data Source...** button to open the Repository Variable Initialization Block Data Source dialog box.



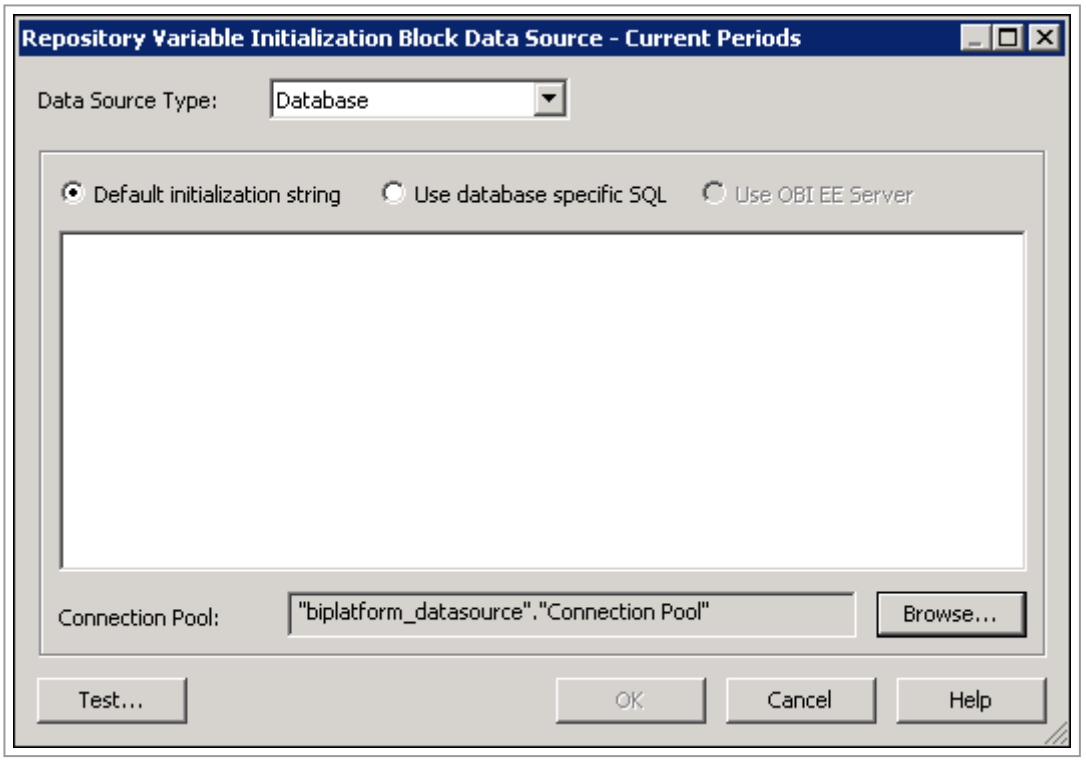

6. Click the **Browse button** to open the Select Connection Pool dialog box.




**7.** Double-click the **Connection Pool** object to select it.

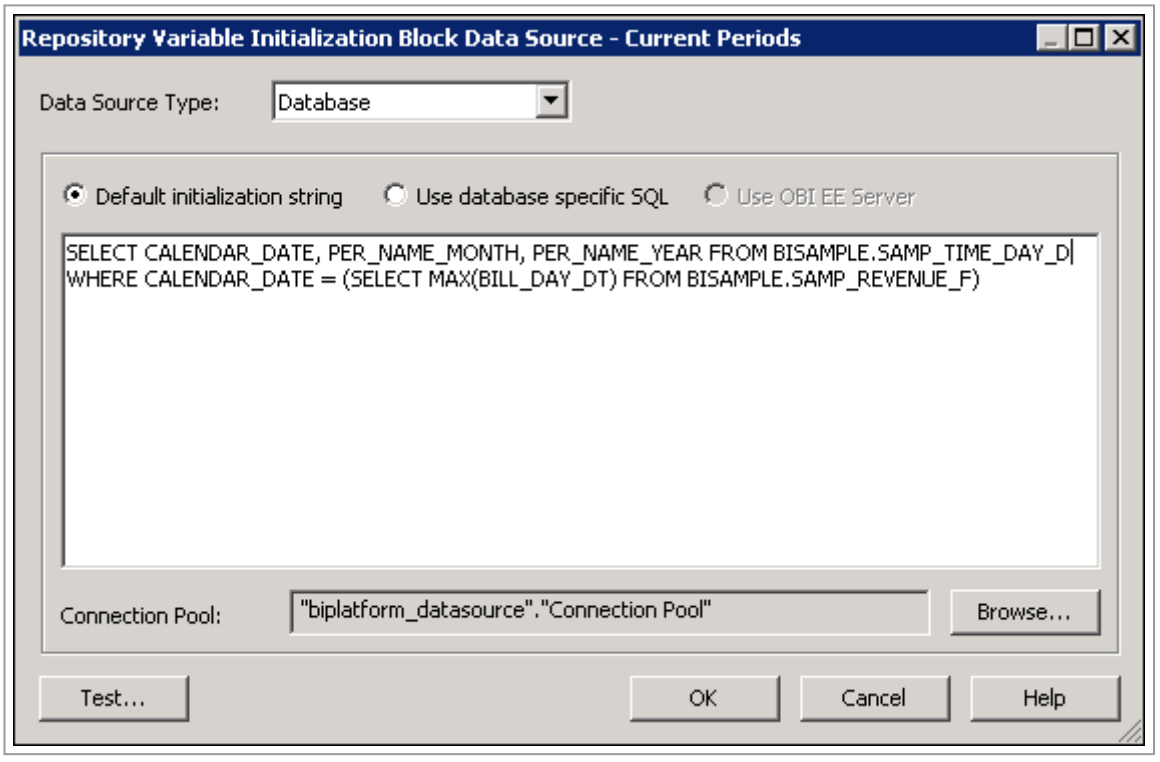



The connection pool is added.

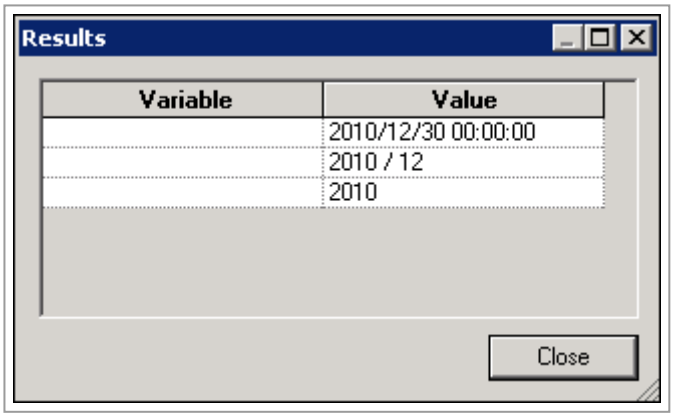



8. Enter the following SQL to determine the value of the current day, month, and year by finding the maximum value of the period key (BILL_DAY_DT) in the fact table:

   SELECT CALENDAR_DATE, PER_NAME_MONTH, PER_NAME_YEAR FROM BISAMPLE.SAMP_TIME_DAY_D WHERE CALENDAR_DATE = (SELECT MAX(BILL_DAY_DT) FROM BISAMPLE.SAMP_REVENUE_F)

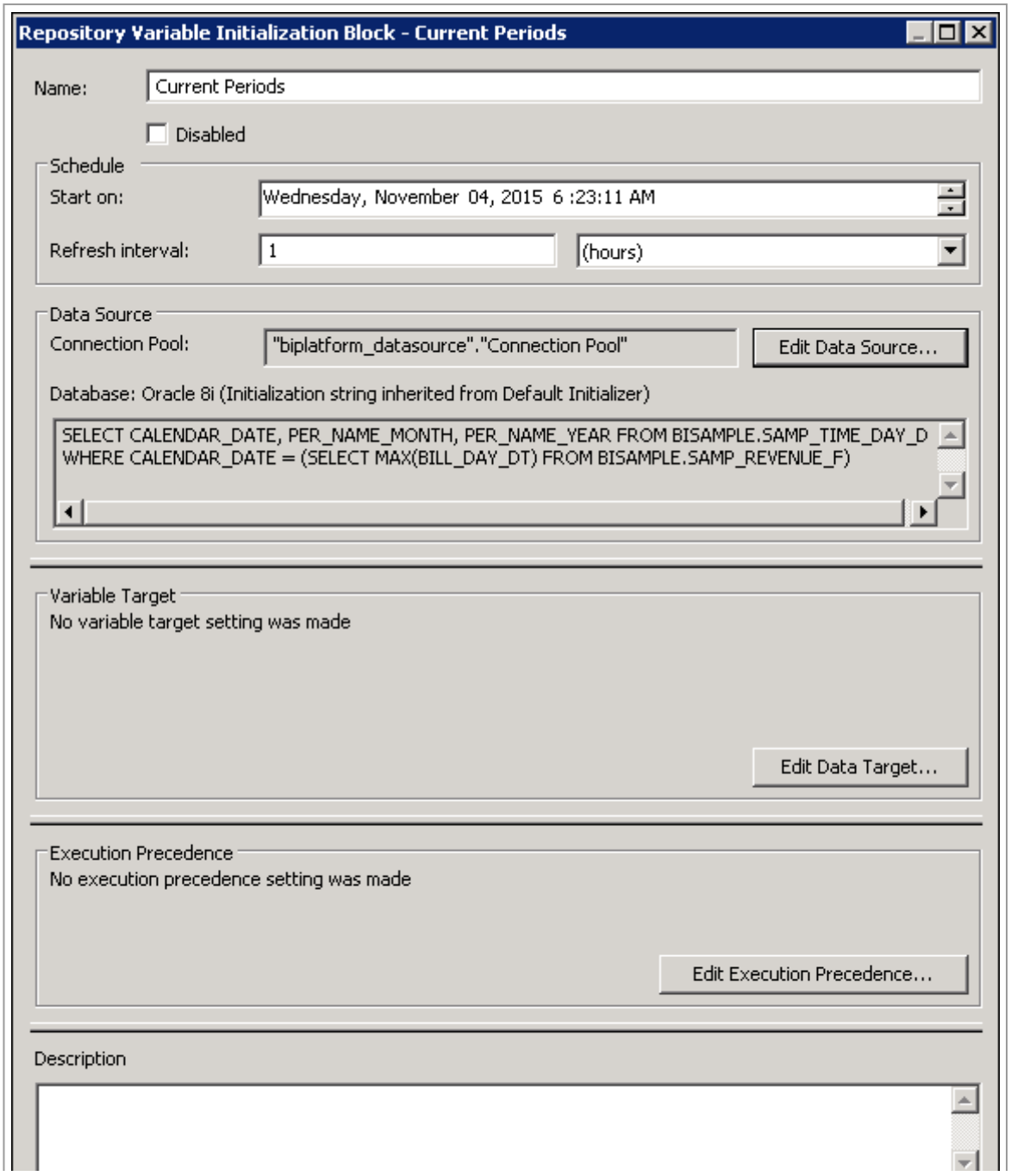



9. Click **Test** and confirm the expected results are returned. In this example, the results are determined by the data in the sample database used for this tutorial, which holds data through December 2010.



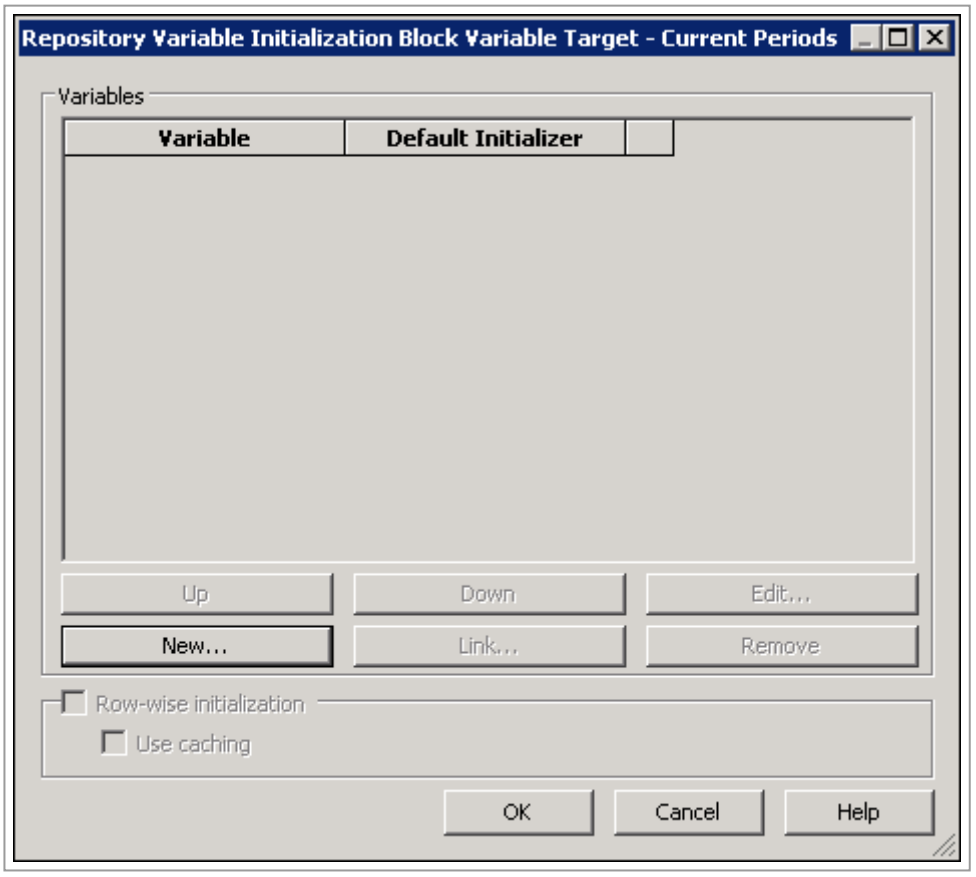

**10.** Close the Results window.

**11.** Click **OK** to close the Repository Variable Initialization Block Data Source dialog box. Check your work:

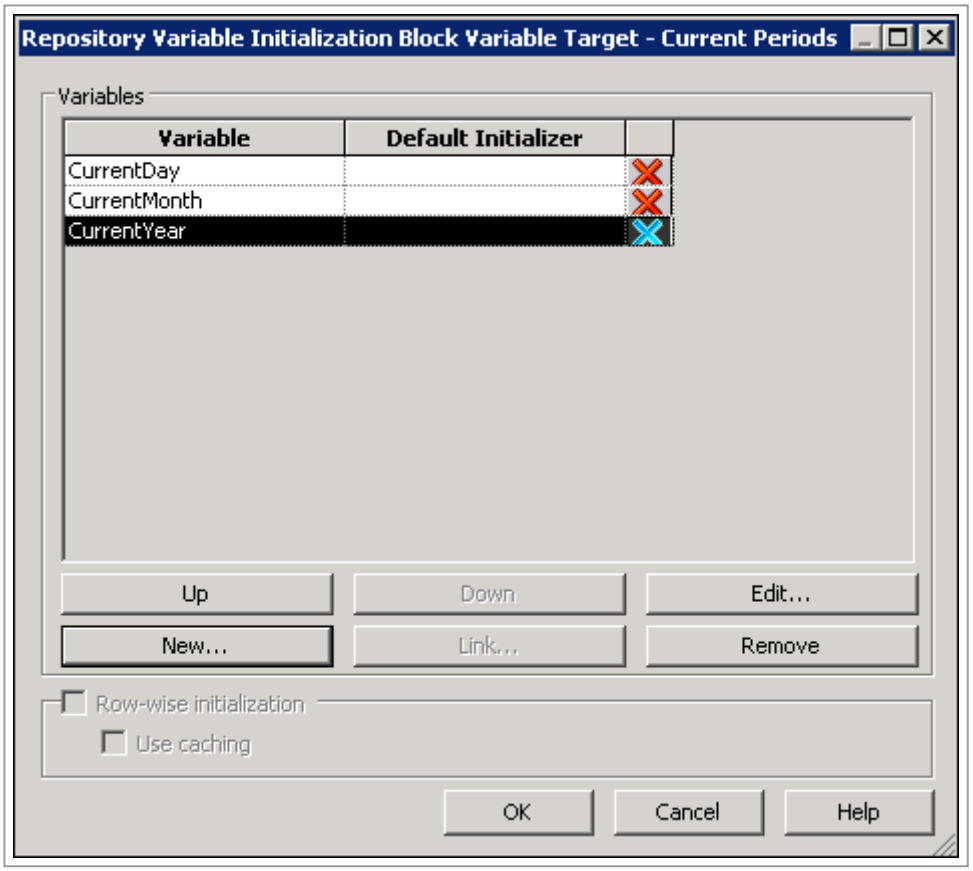



## Creating Variables

1. Click **Edit Data Target** to open the Repository Variable Initialization Block Variable Target dialog box.

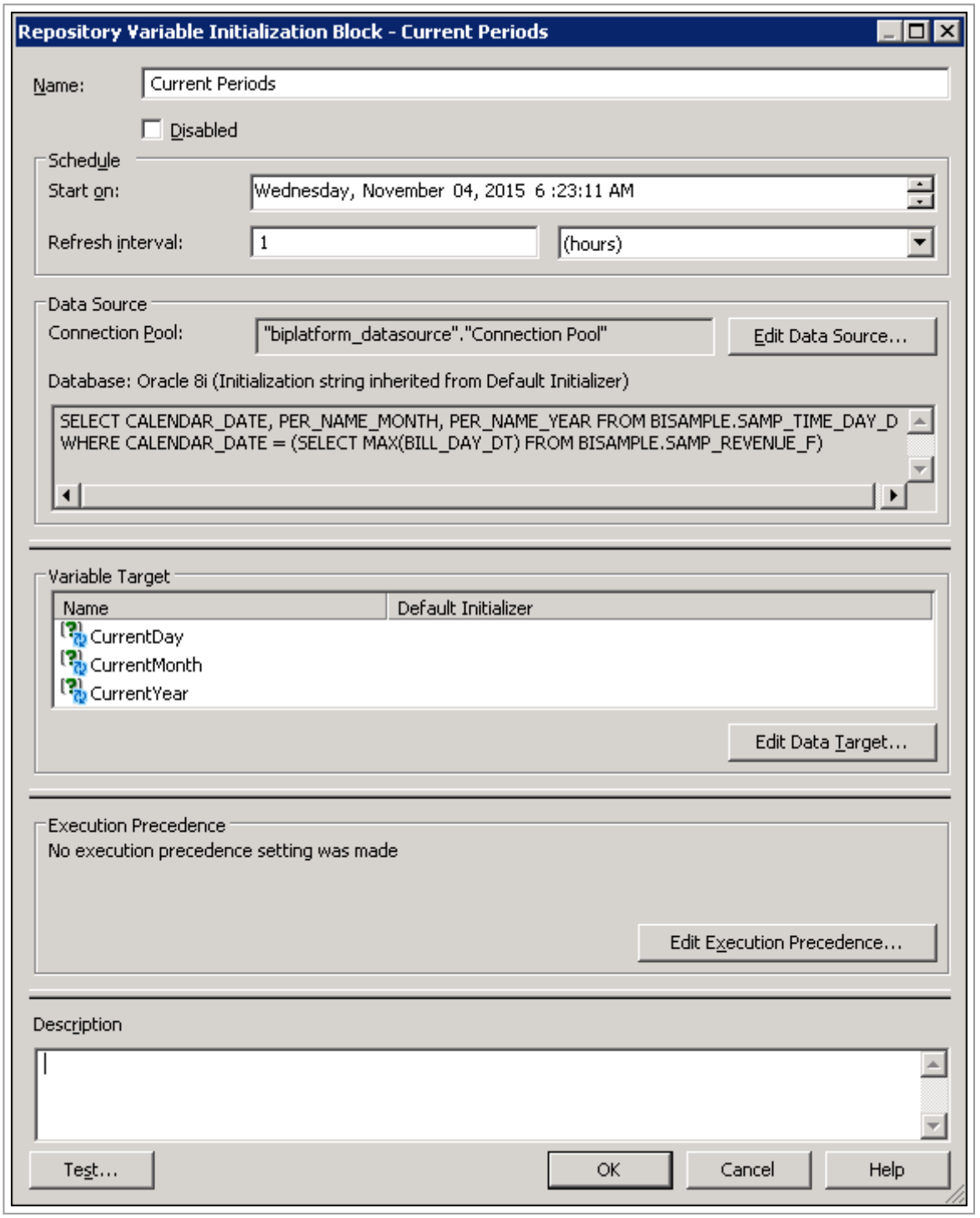



2. Use the **New** button to create three new variables: **CurrentDay**, **CurrentMonth**, **CurrentYear**. The order is important. The value returned from the first column in the initialization block SQL, CALENDAR_DATE, is assigned to the CurrentDay variable. The value of the second column, PER_NAME_MONTH, is assigned to CurrentMonth (the second variable), and the value of the third column, PER_NAME_YEAR, is assigned to CurrentYear (the third variable). If necessary, use the Up and Down buttons to arrange the variables.

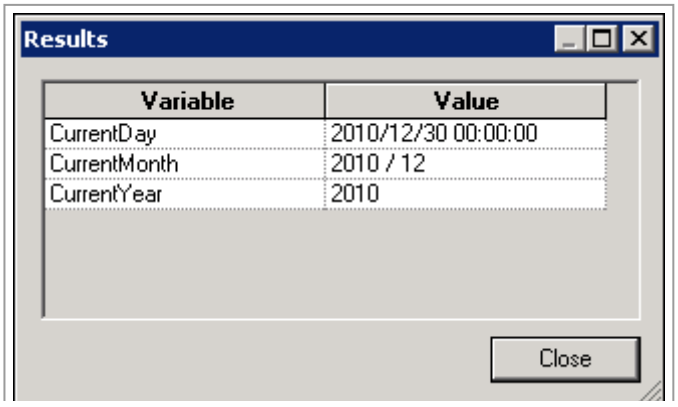



3. Click **OK** to close the Repository Variable Initialization Block Variable Target dialog box.

4. Leave the default **Refresh Interval** set to every hour. This means that the variables will be reinitialized every hour.




**5.** Click the **Test** button and check the results:

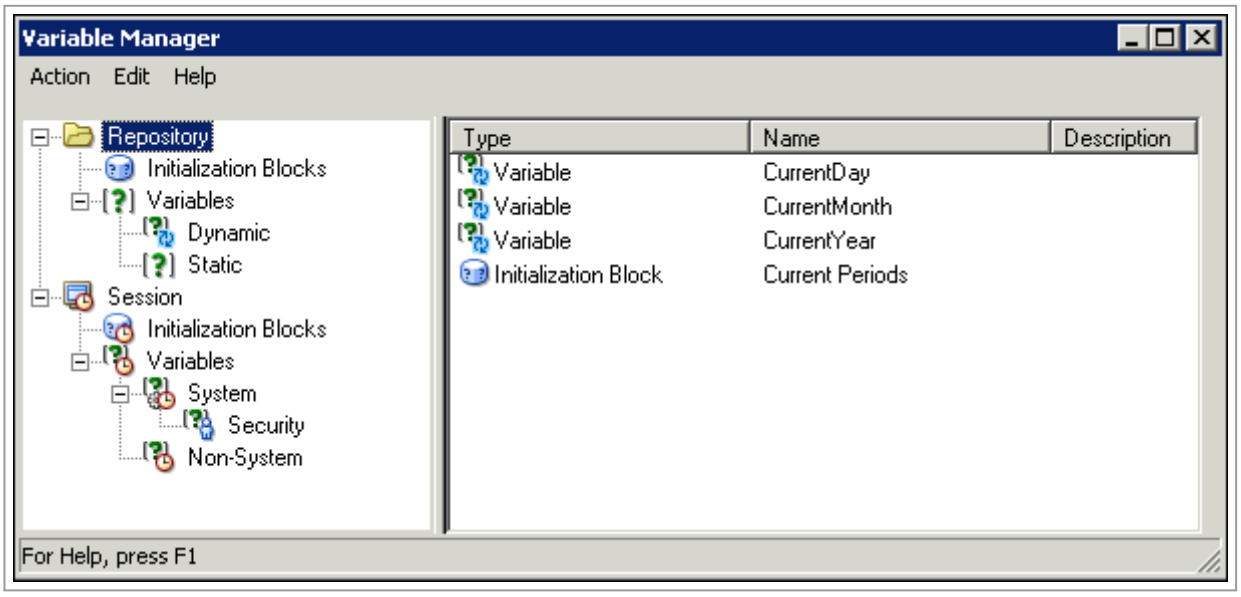


In this example, the results are determined by the data in the sample database used for this tutorial, which holds data through December 2010.

6. Close the Results window.

7. Click **OK** to close the Repository Variable Initialization Block dialog box.

8. Check your work in the Variable Manager




9. Close the Variable Manager.

10. Save the repository and check consistency. Fix any errors or warnings before proceeding.

11. Close the repository. Leave the Administration Tool open.

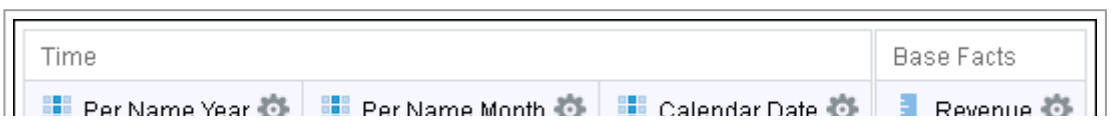
## Testing Your Work

1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

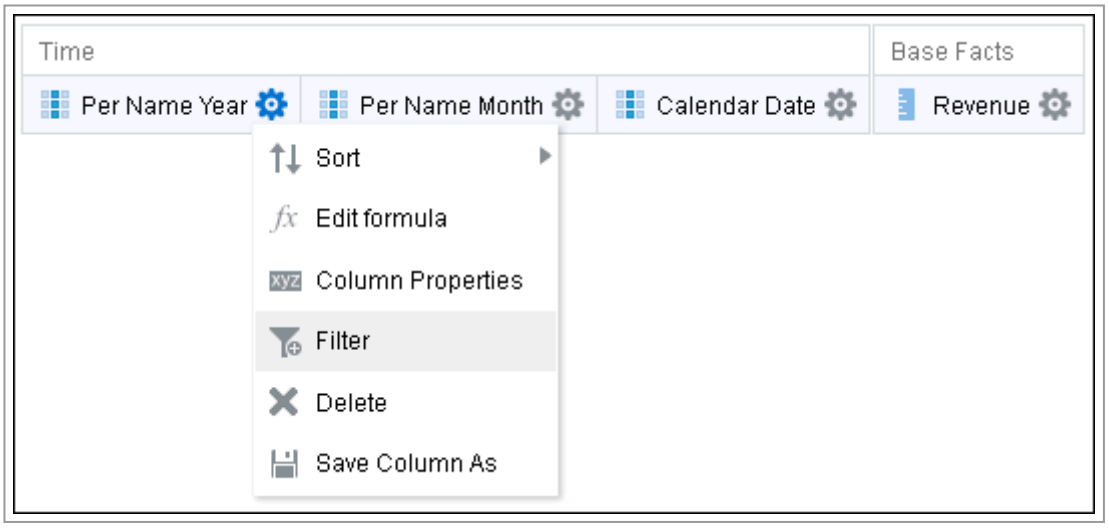
2. Return to **Oracle BI** and sign in.

3. Create the following analysis to test the variables.

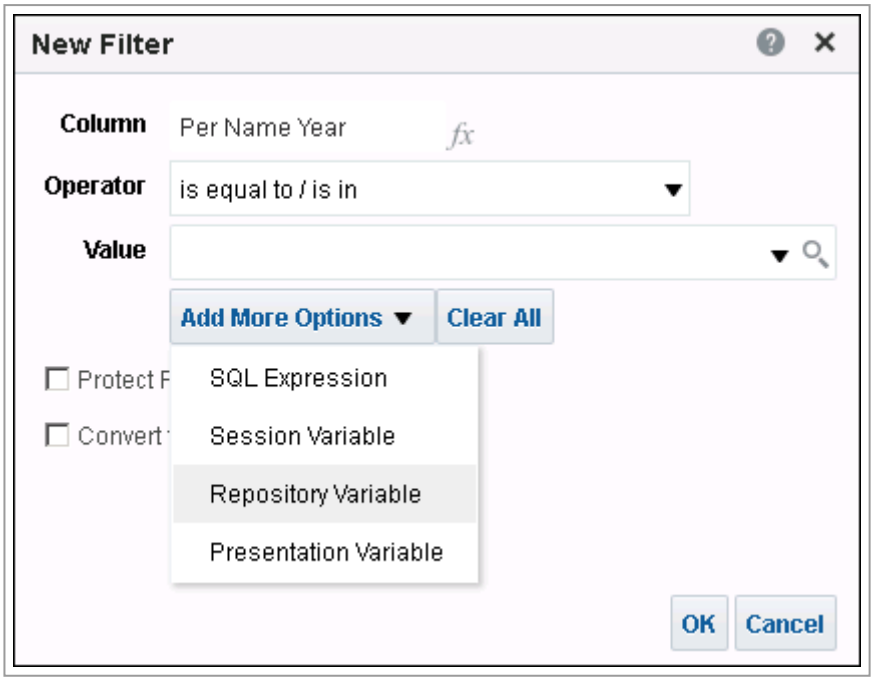
**Time.Per Name Year**
**Time.Per Name Month**
**Time.Calendar Date**
**Base Facts.Revenue**



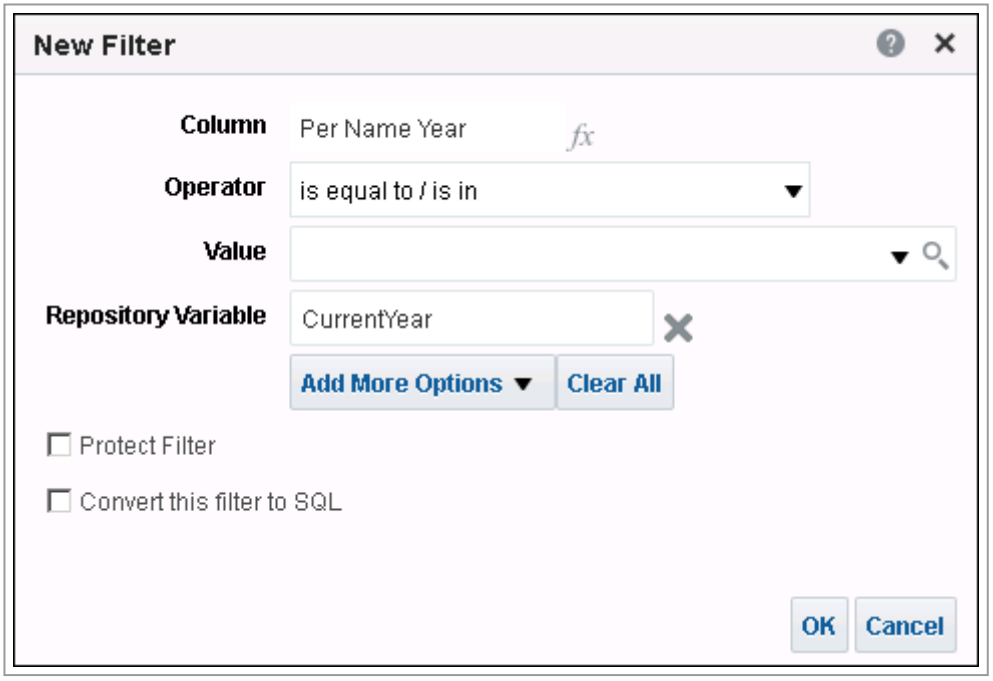
4. Click **Filter** for the Per Name Year column. The New Filter dialog box opens.




5. Select **Add More Options > Repository Variable**.

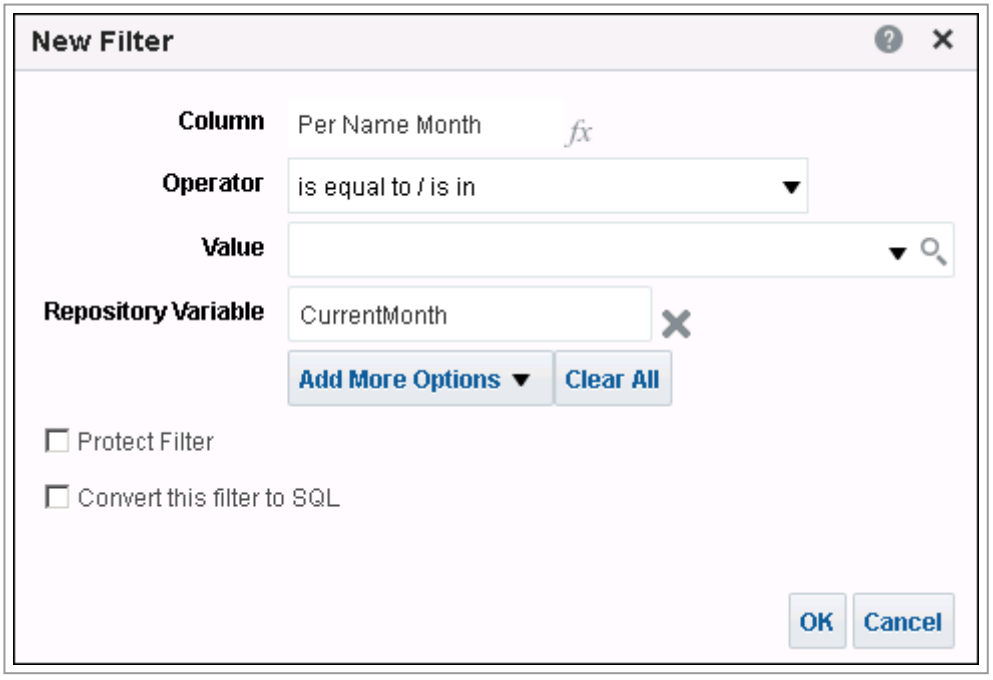


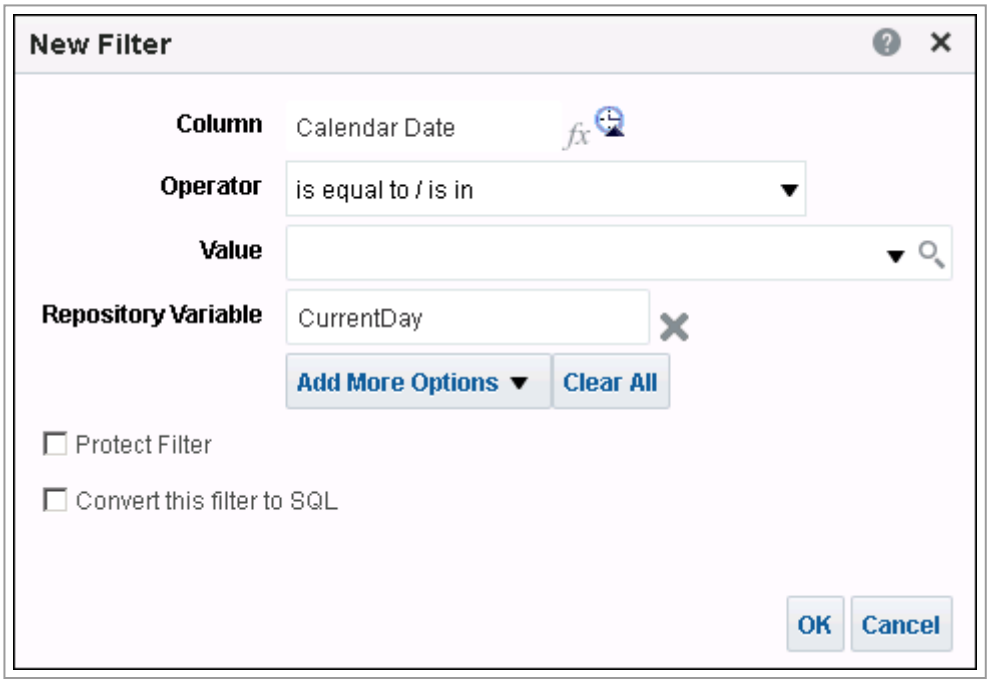

6. In the Repository Variable field, enter **CurrentYear** to create a filter for the Per Name Year column using the CurrentYear repository variable.




7. Click **OK** to close the New Filter dialog box. The filter is added to the Filters pane.

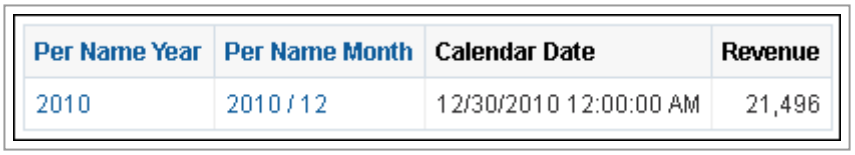

8. Repeat the steps to add the **CurrentMonth** and **CurrentDay** repository variables as filters for **Per Name Month** and **Calendar Date** columns, respectively.







9. Click **Results** and confirm that data only for the current year, month, and day is returned (based on the sample data set).



| Per Name Year | Per Name Month | Calendar Date | Revenue |
|---|---|---|---|
| 2010 | 2010 / 12 | 12/30/2010 12:00:00 AM | 21,496 |

10. Sign out of Oracle BI. Click **Leave Page** when prompted about navigating away from this page. Leave the Oracle BI browser page open.

## Creating Time Series Measures

In this topic you create time series calculation measures using Oracle BI time series functions.

Time series functions include AGO, TODATE, and PERIODROLLING. These functions let you use Expression Builder to call a logical function to perform time series calculations instead of creating aliases for physical tables and modeling logically. The time series functions calculate AGO, TODATE, and PERIODROLLING functions based on the calendar tables in your data warehouse, not on standard SQL date manipulation functions.

To create time series measures, you perform the following steps:

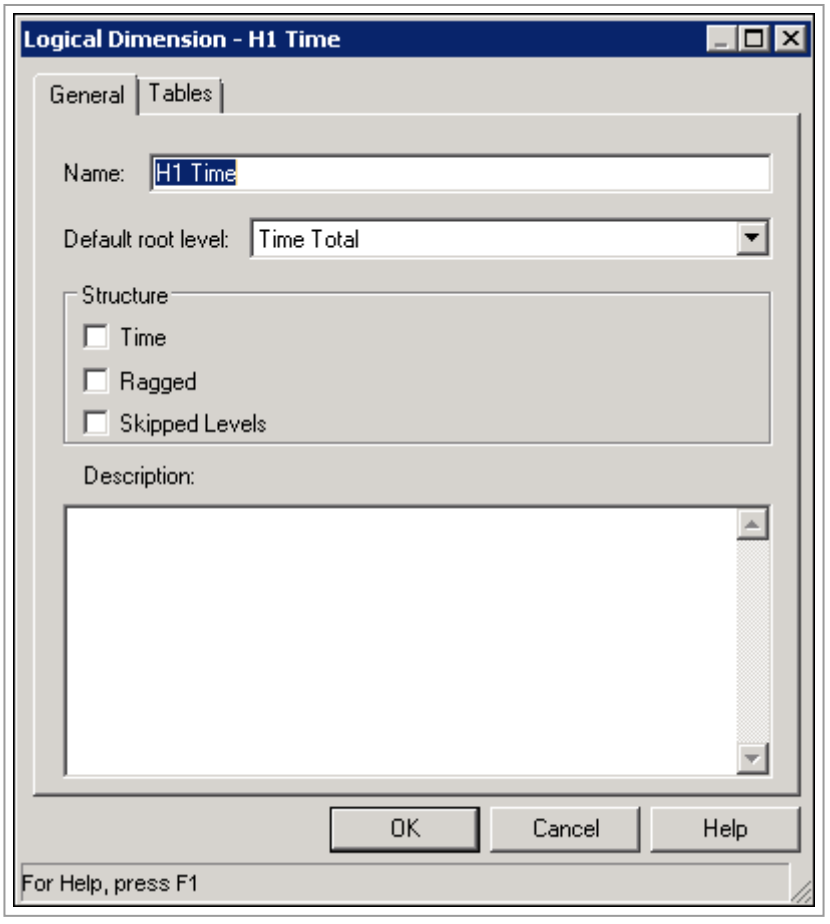
- Identifying a Logical Dimension as a Time Dimension

- Identifying Level Keys as Chronological Keys
- Creating a Measure Using the AGO Function
- Creating a Measure Using the TODATE Function
- Creating a Measure Using the PERIODROLLING Function
- Testing Your Work

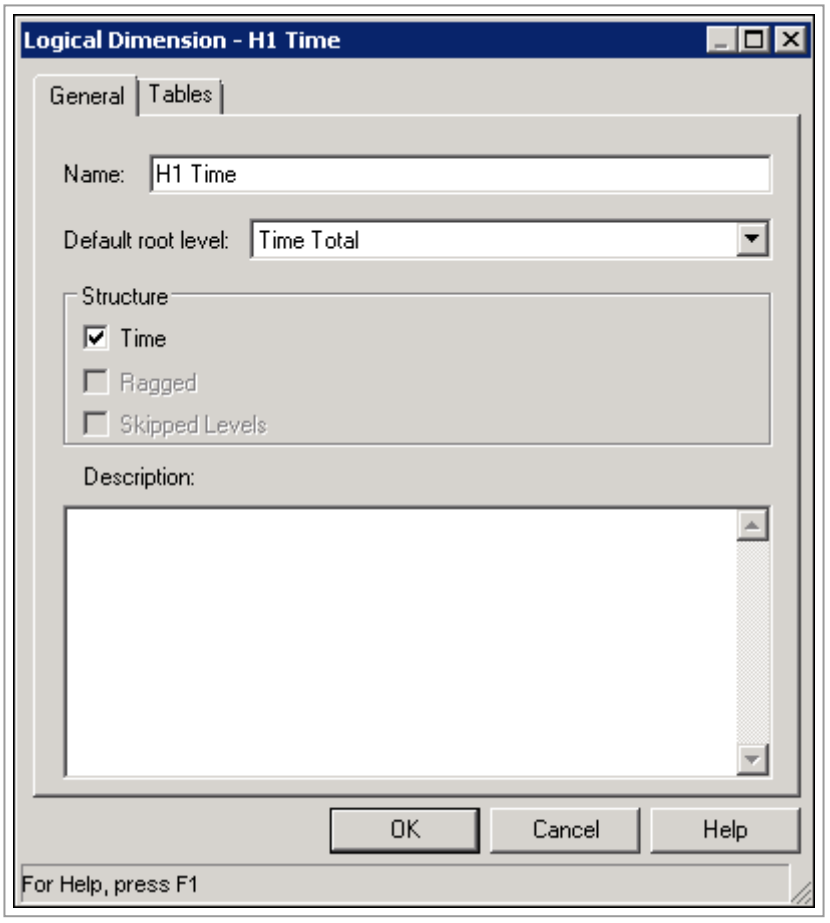
## Identifying a Logical Dimension as a Time Dimension

1. Return to the Administration Tool and open the **BISAMPLE** repository in offline mode.

2. In the **BMM** layer, double-click the **H1 Time** logical dimension to open the Logical Dimension dialog box.




3. In the Structure section, select **Time**.



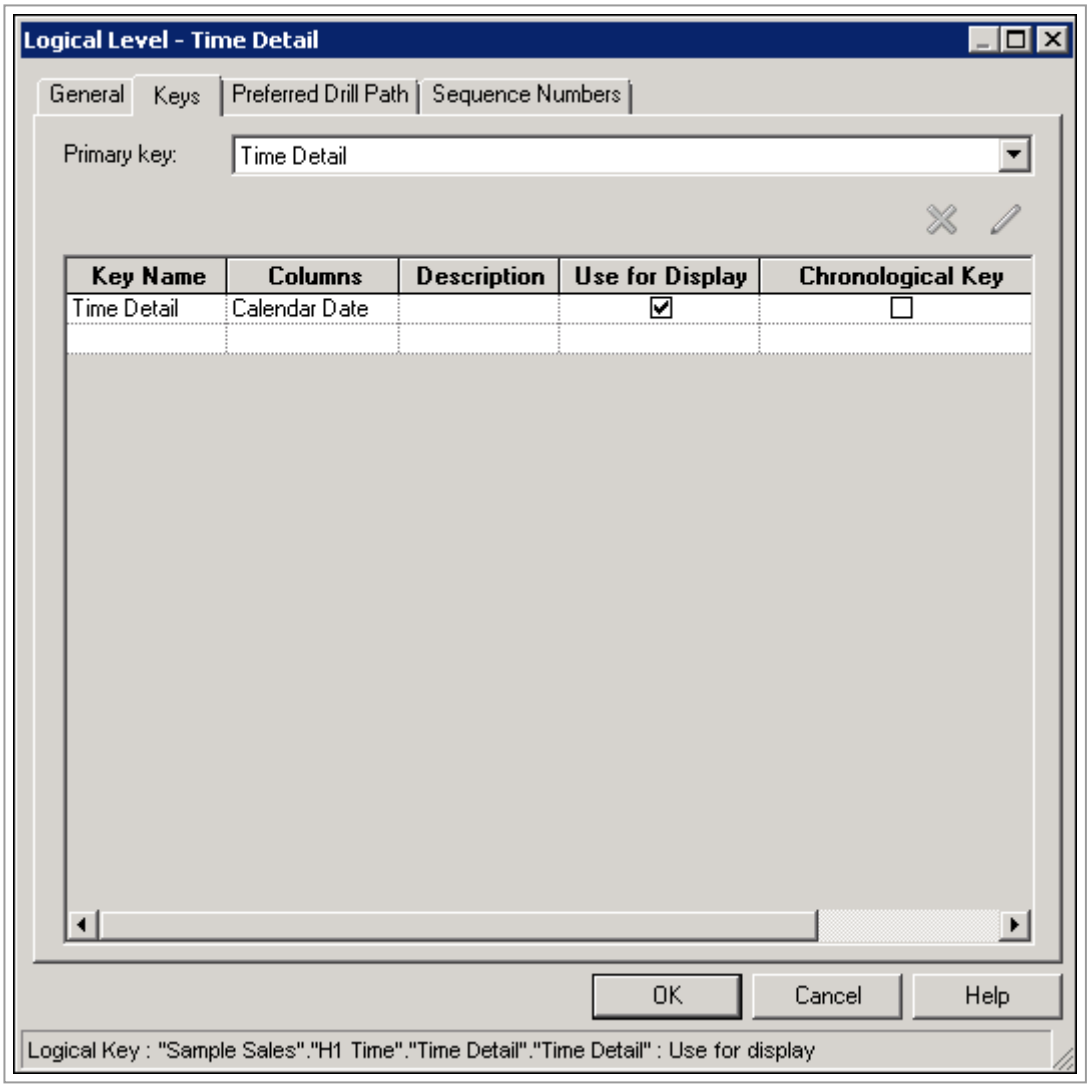

Time series functions operate on time-oriented dimensions. To use these functions on a particular dimension, you must designate the dimension as a Time dimension.

4. Click **OK** to close the Logical Dimension dialog box.

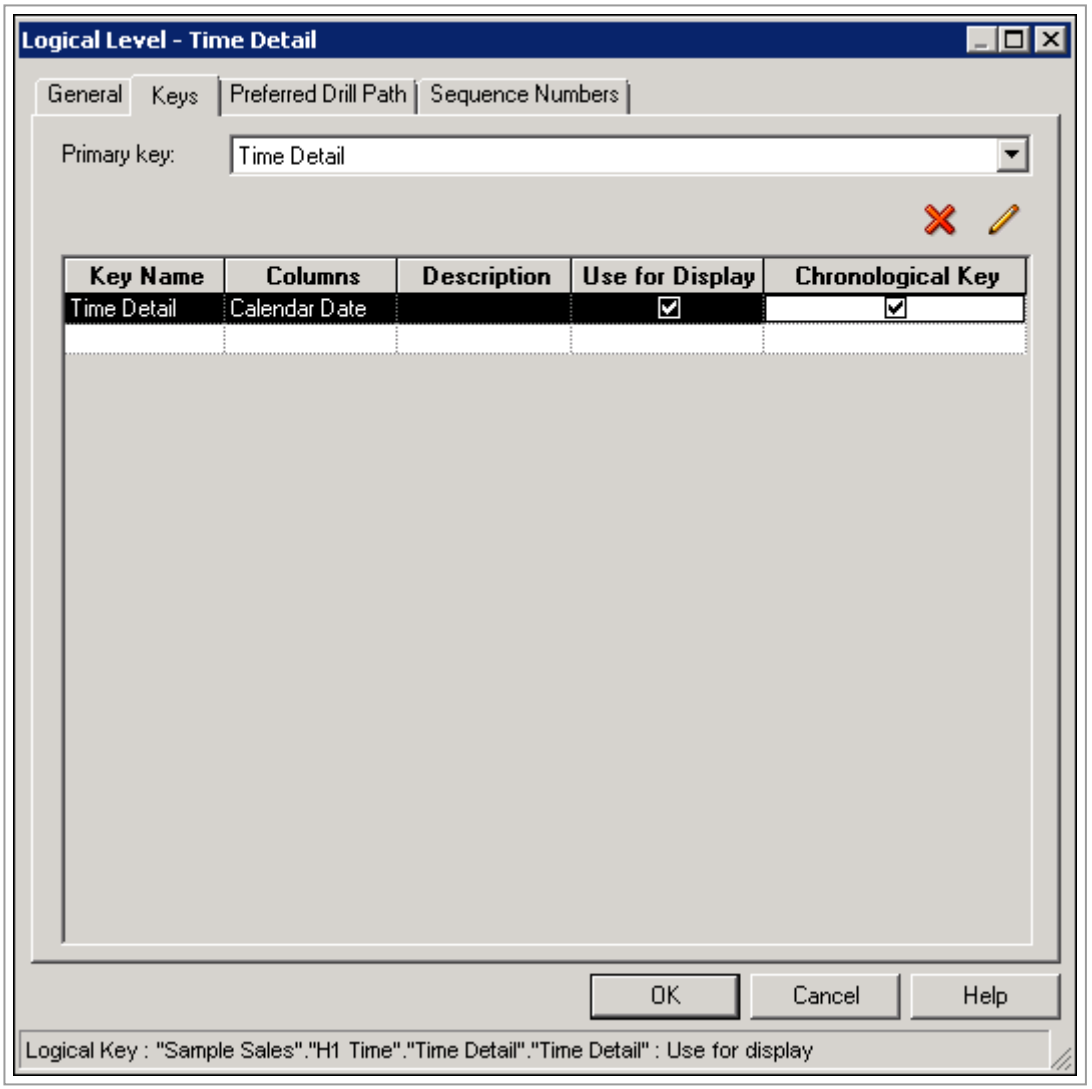
## Identifying Level Keys as Chronological Keys

1. Expand the **H1 Time** logical dimension and double-click the **Time Detail** level to open the Logical Level dialog box.

2. Click the **Keys** tab.




**3.** Select the **Chronological Key** check box for **Calendar Date**.




4. Click **OK** to close the Logical Level dialog box.

5. Repeat and set chronological keys for the following levels:

| Logical Level | Chronological Key |
|---|---|
| Year | Per Name Year |
| Half | Per Name Half |
| Quarter | Per Name Qtr |
| Month | Per Name Month |
| Week | Per Name Week |

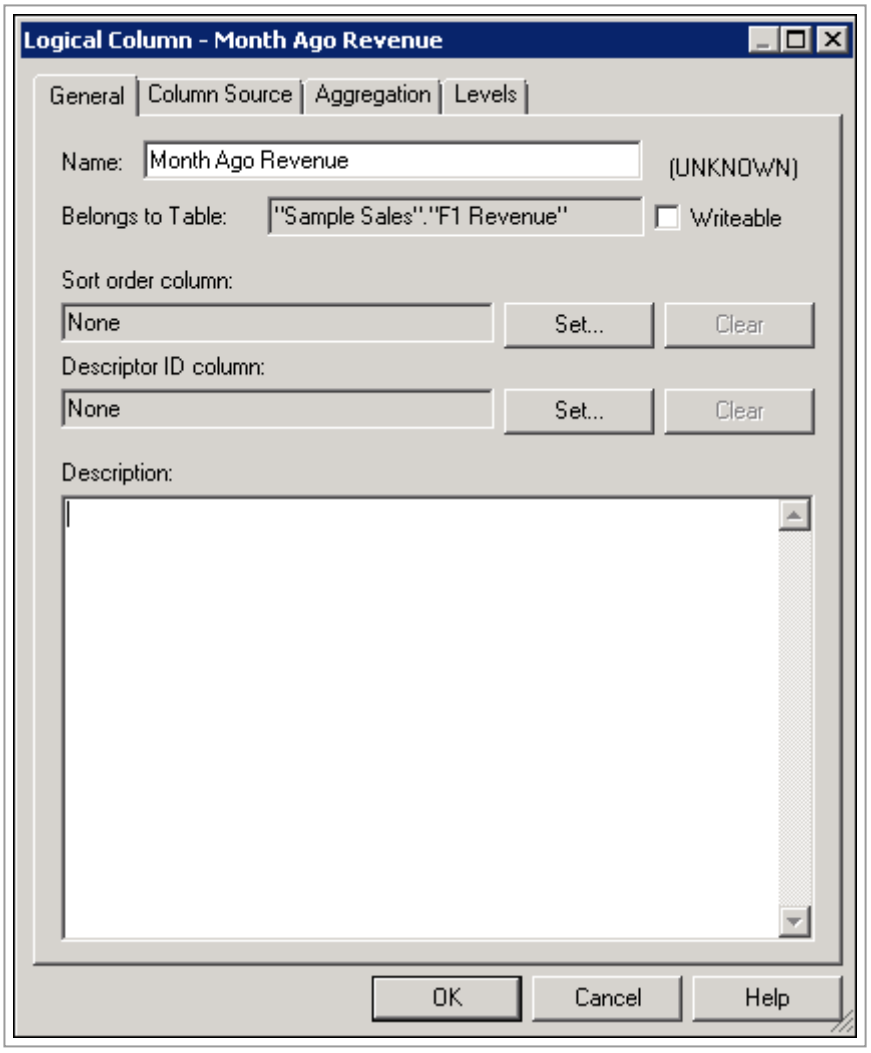
It is best practice to designate a chronological key for every level of a time logical dimension.

## Creating a Measure Using the AGO Function

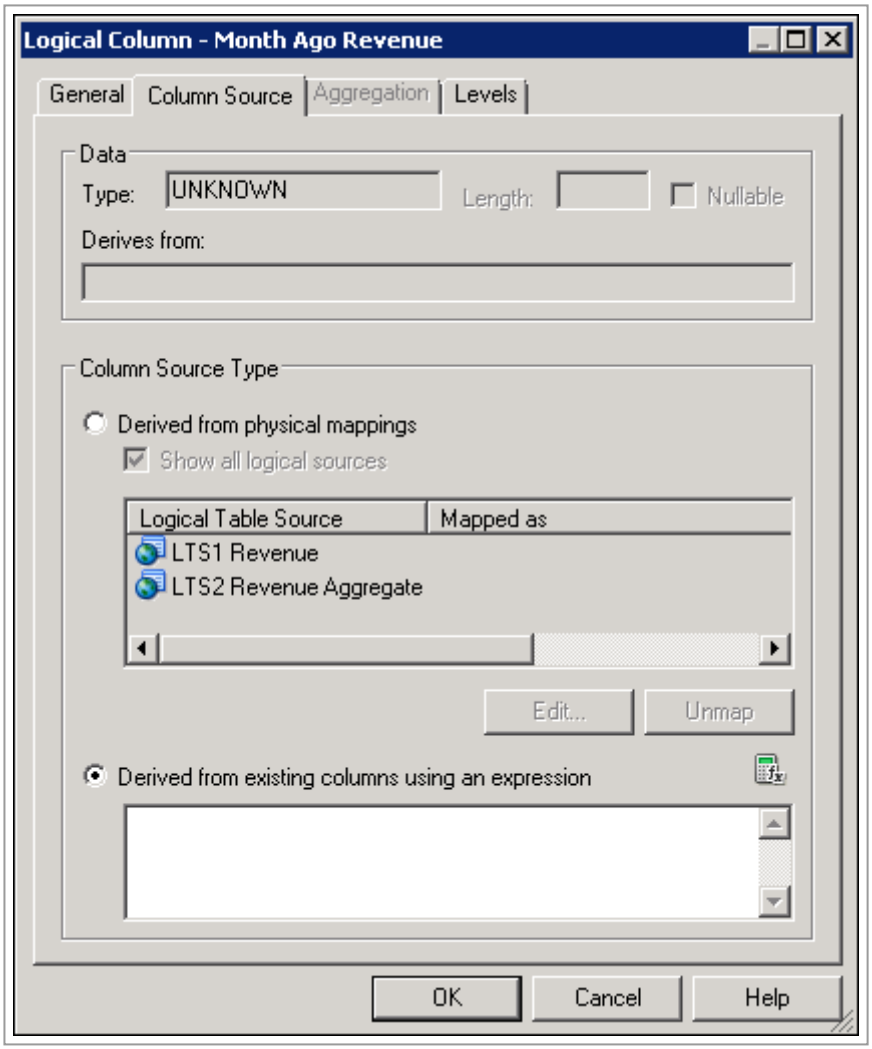
1. Right-click the **F1 Revenue** logical table and select **New Object > Logical Column**.

2. On the **General** tab, name the column **Month Ago Revenue**.

**2.** On the **General** tab, name the column **Month Ago Revenue**.

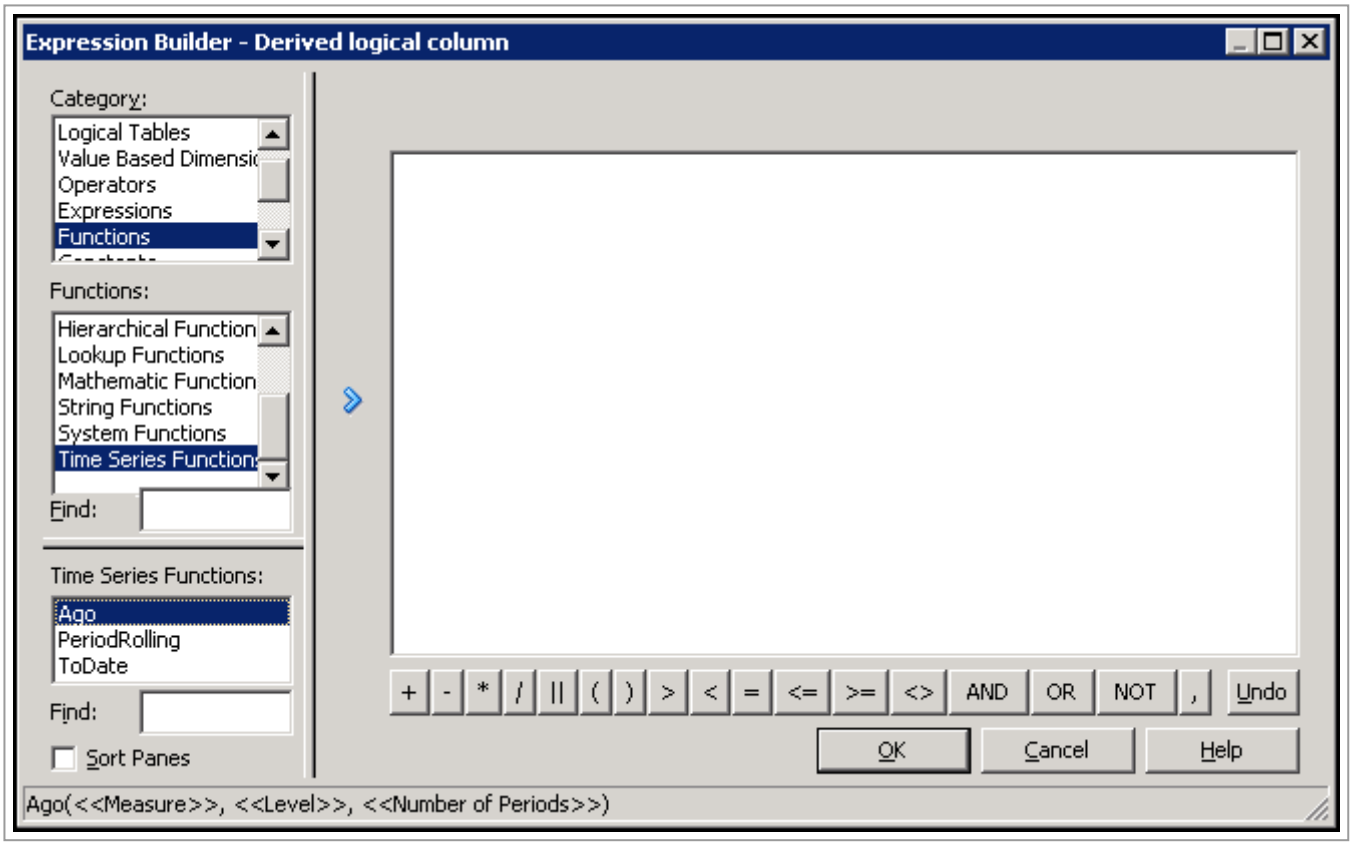



**3.** On the **Column Source** tab, select "Derived from existing columns using an expression".

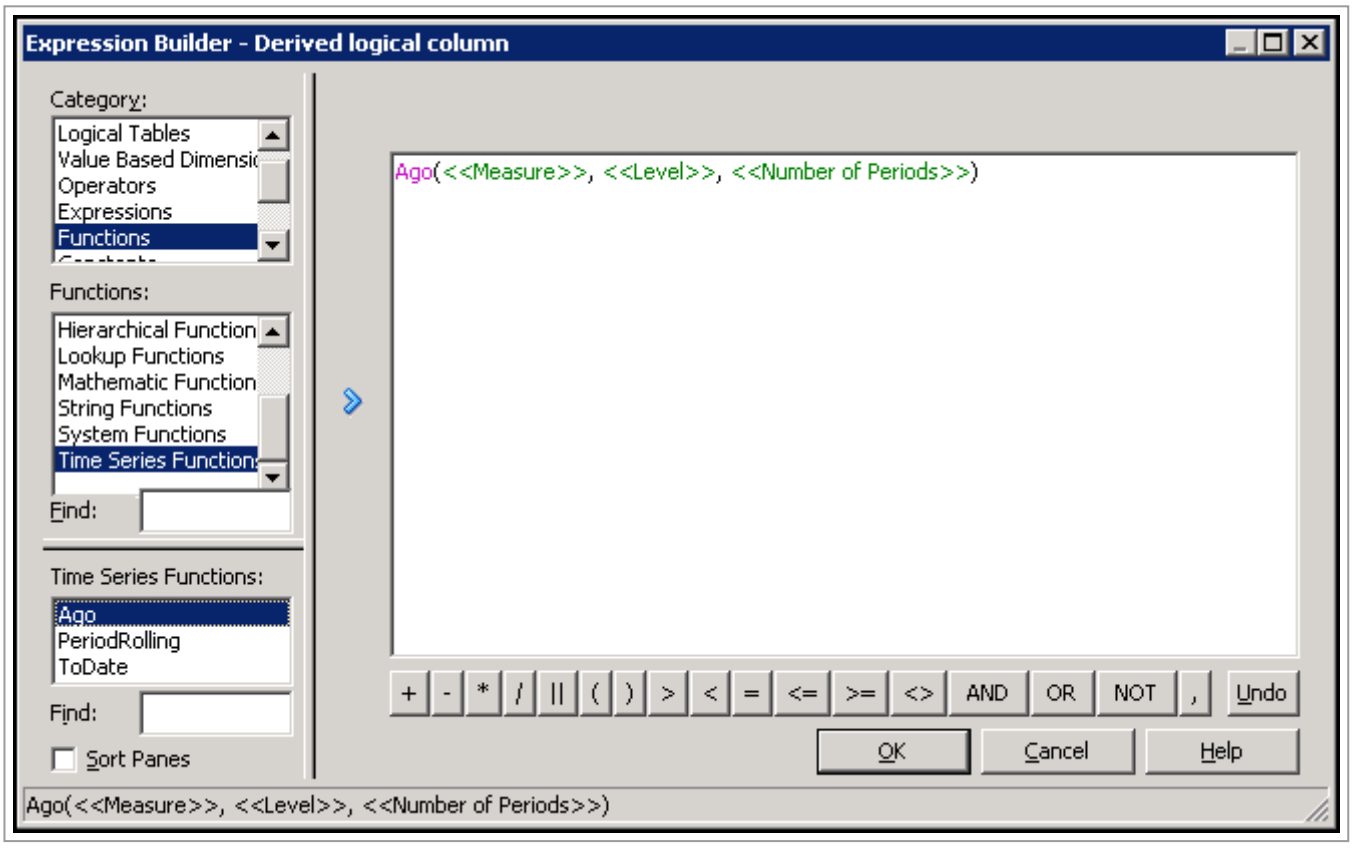



**4.** Open the Expression Builder.

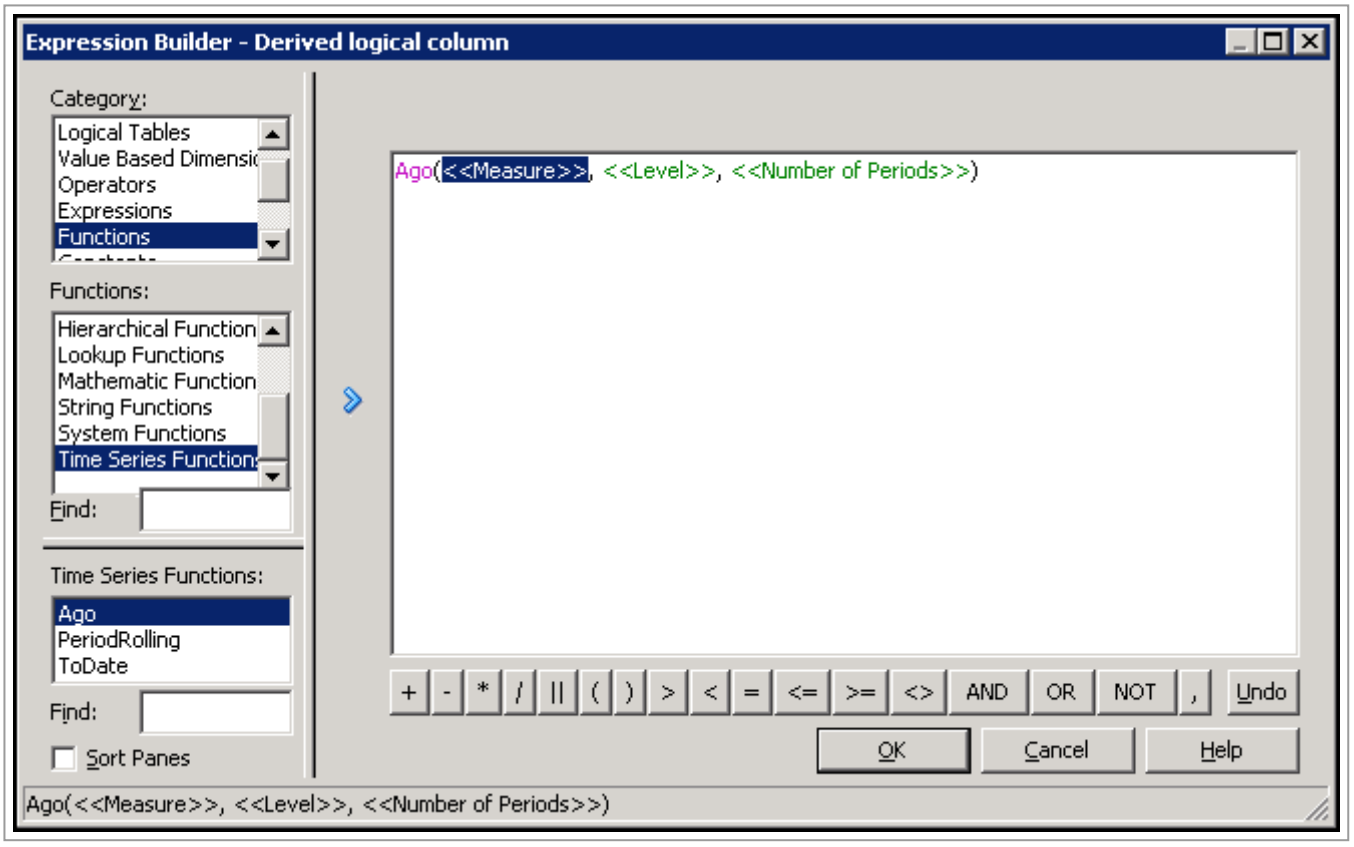
**5.** Select **Functions > Time Series Functions > Ago**.



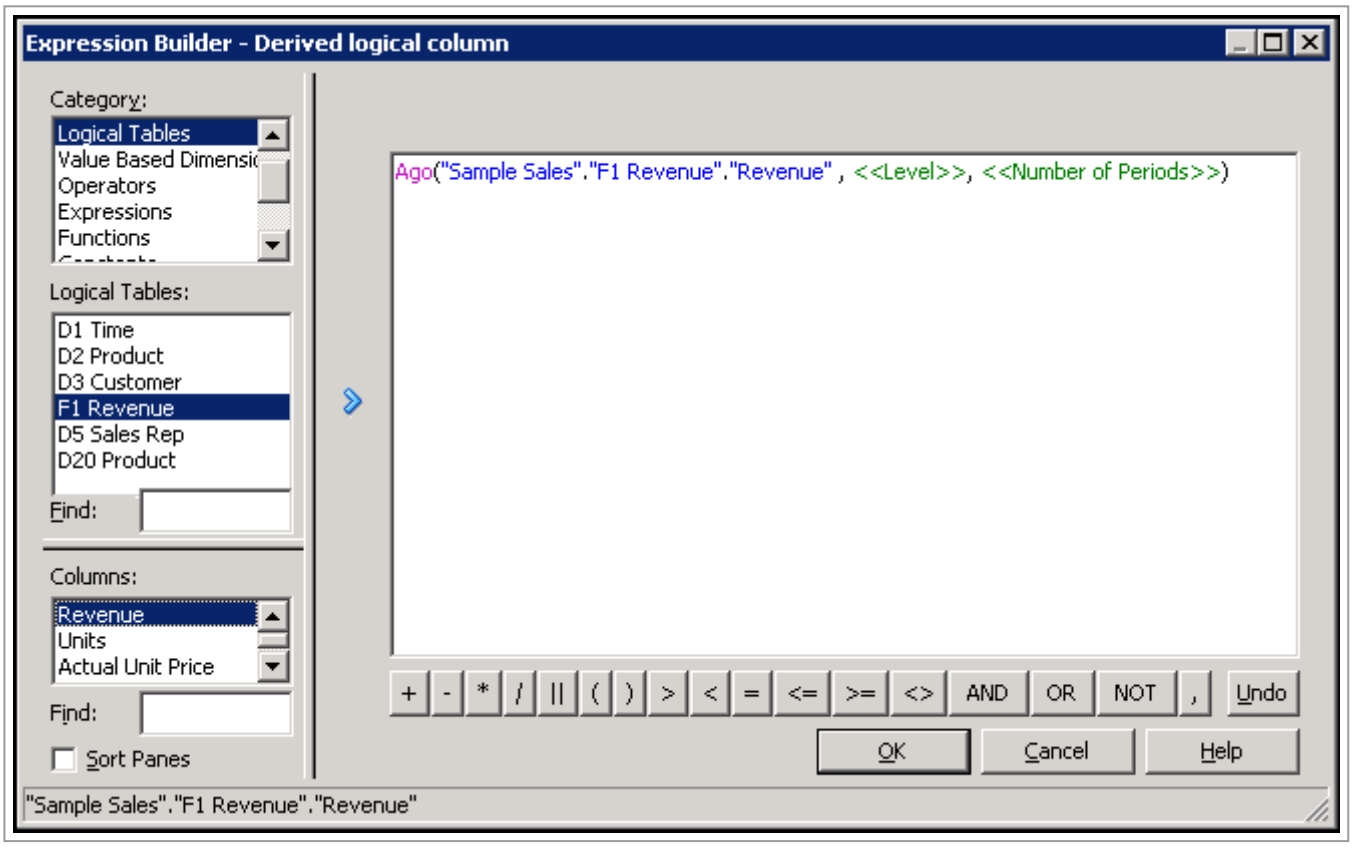

**6.** Double-click **Ago** or click **Insert selected item** to add the Ago function to the Expression Builder.



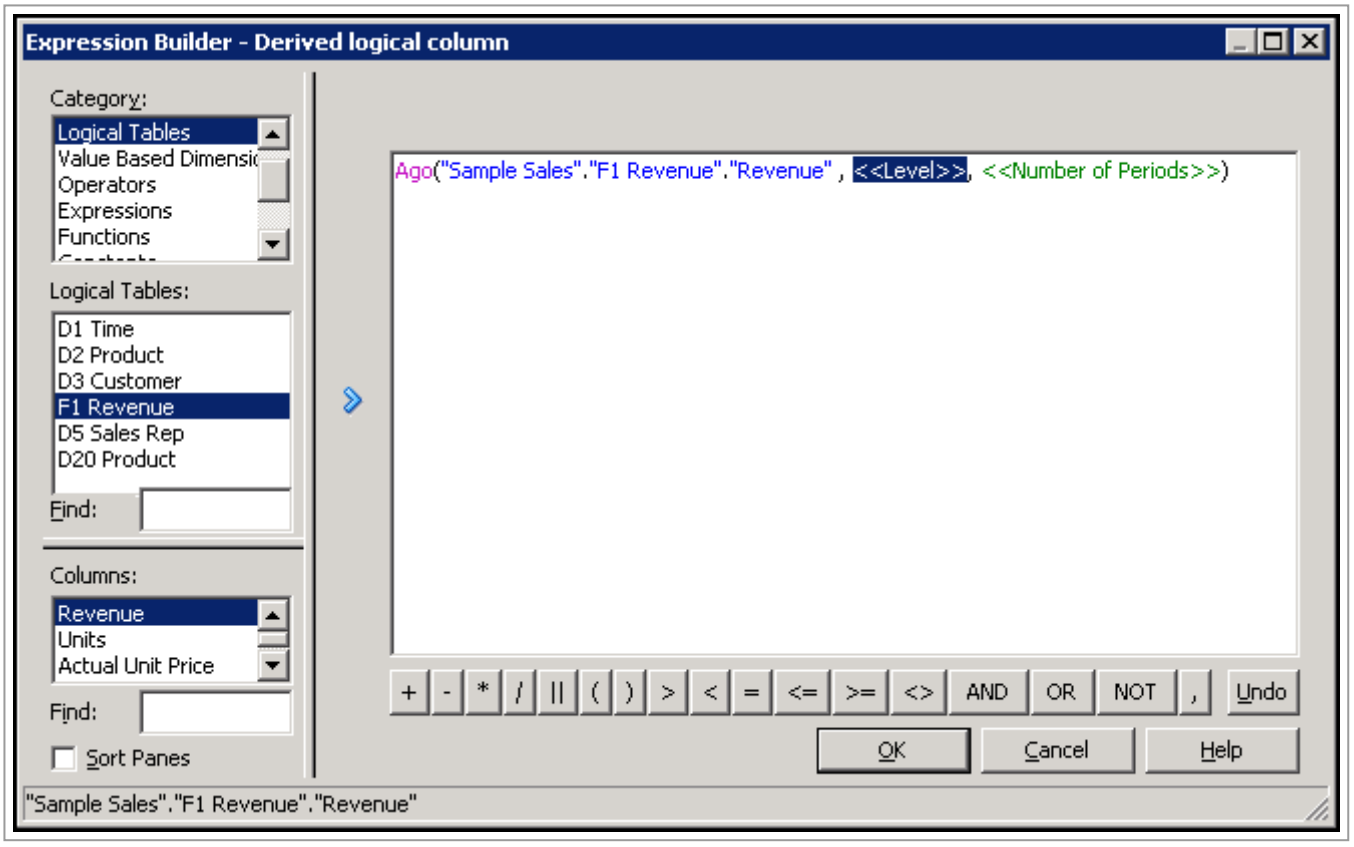

**7.** Click **<<Measure>>**in the expression.



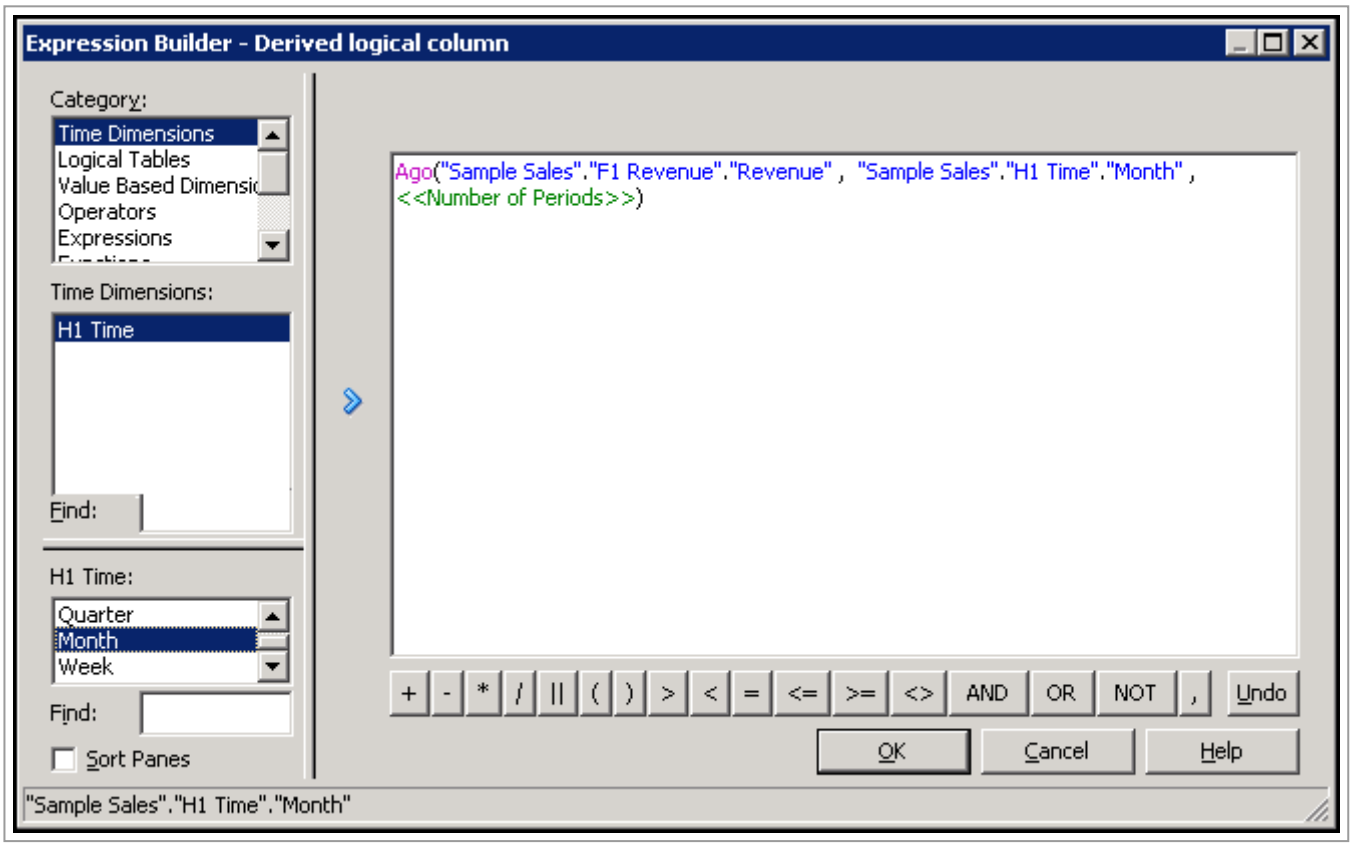

**8.** Select **Logical Tables > F1 Revenue** and then double-click **Revenue** to add it to the expression.



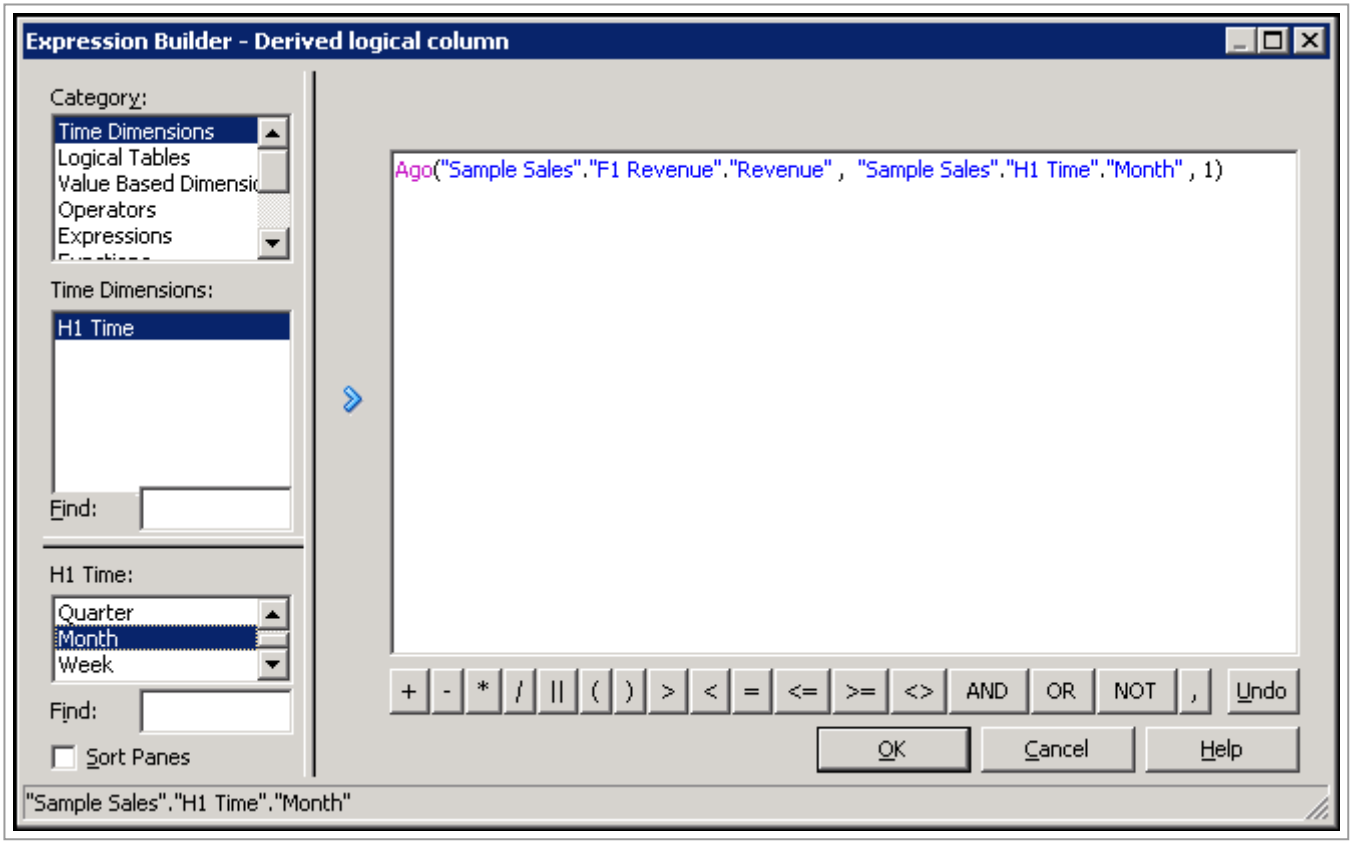

**9.** Click **<<Level>>** in the expression.



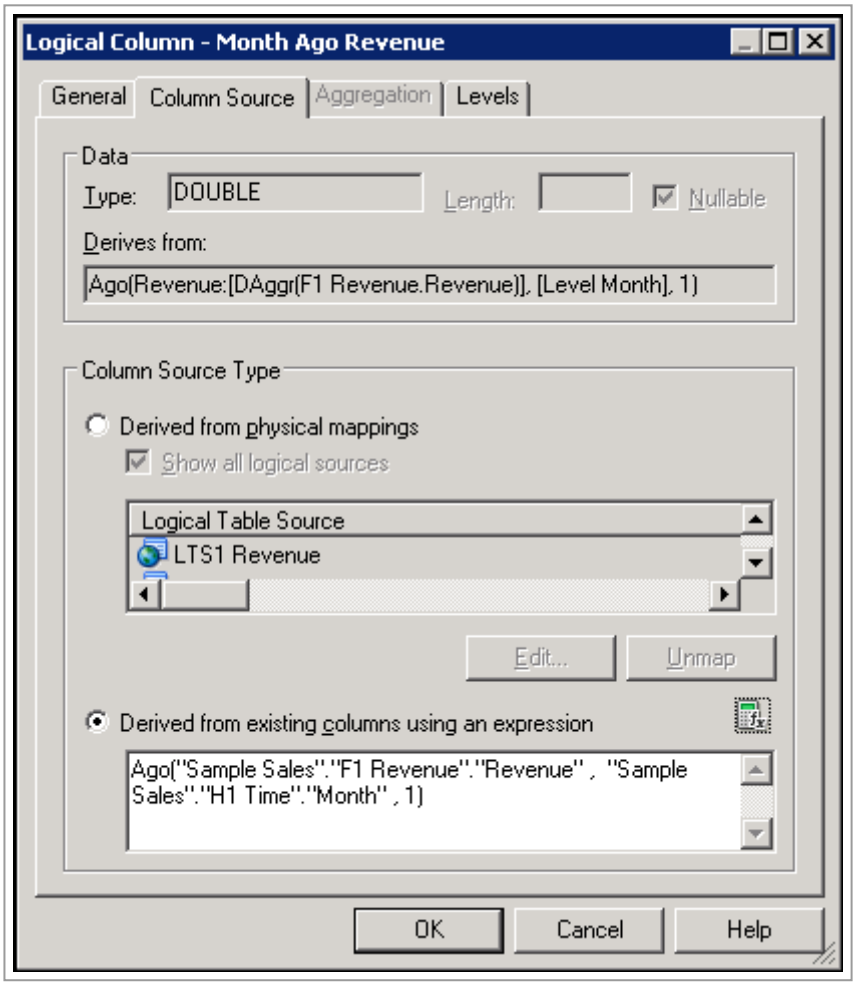

**10.** Select **Time Dimensions > H1 Time** and then double-click **Month** to add it to the expression.



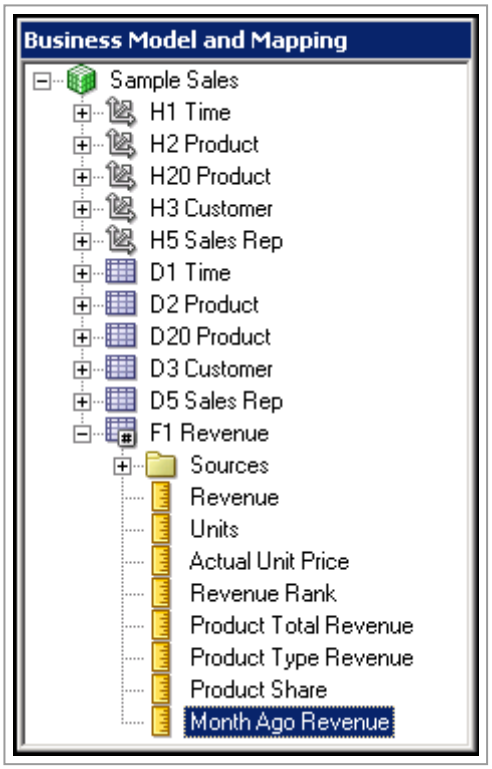

**11.** Click **<<Number of Periods>>** and enter **1**. The Ago function will calculate the Revenue value one month before the current month.



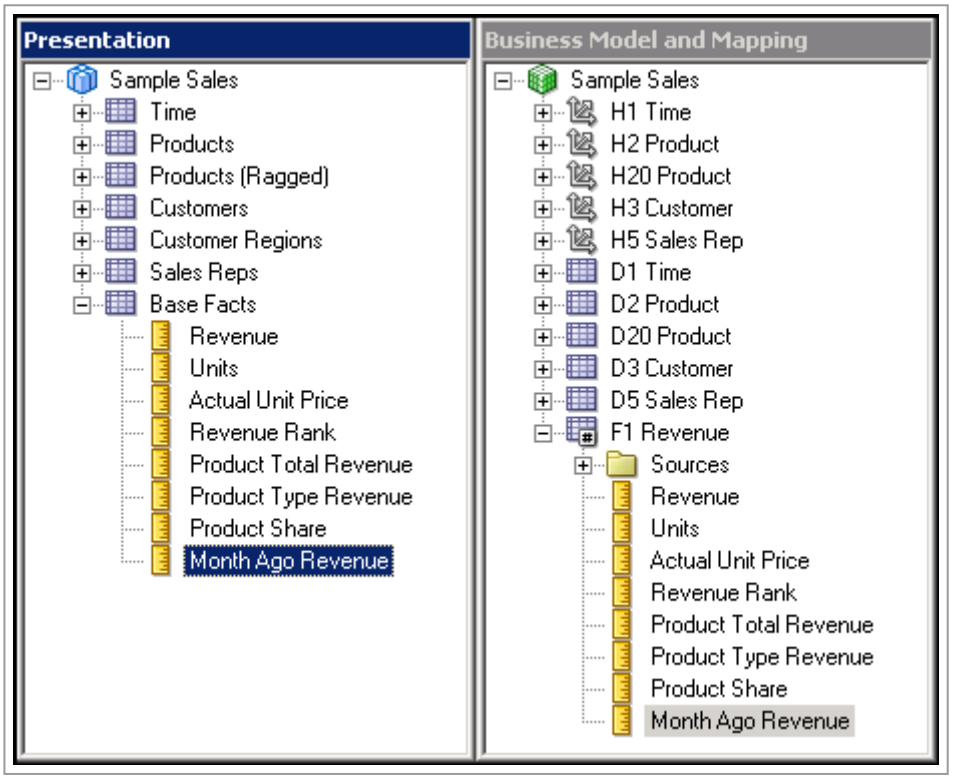

**12.** Click **OK** to close the Expression Builder. Check your work in the Logical Column dialog box:



13. Click **OK** to close the Logical Column dialog box. The Month Ago Revenue time series measure is added to the F1 Revenue logical table.



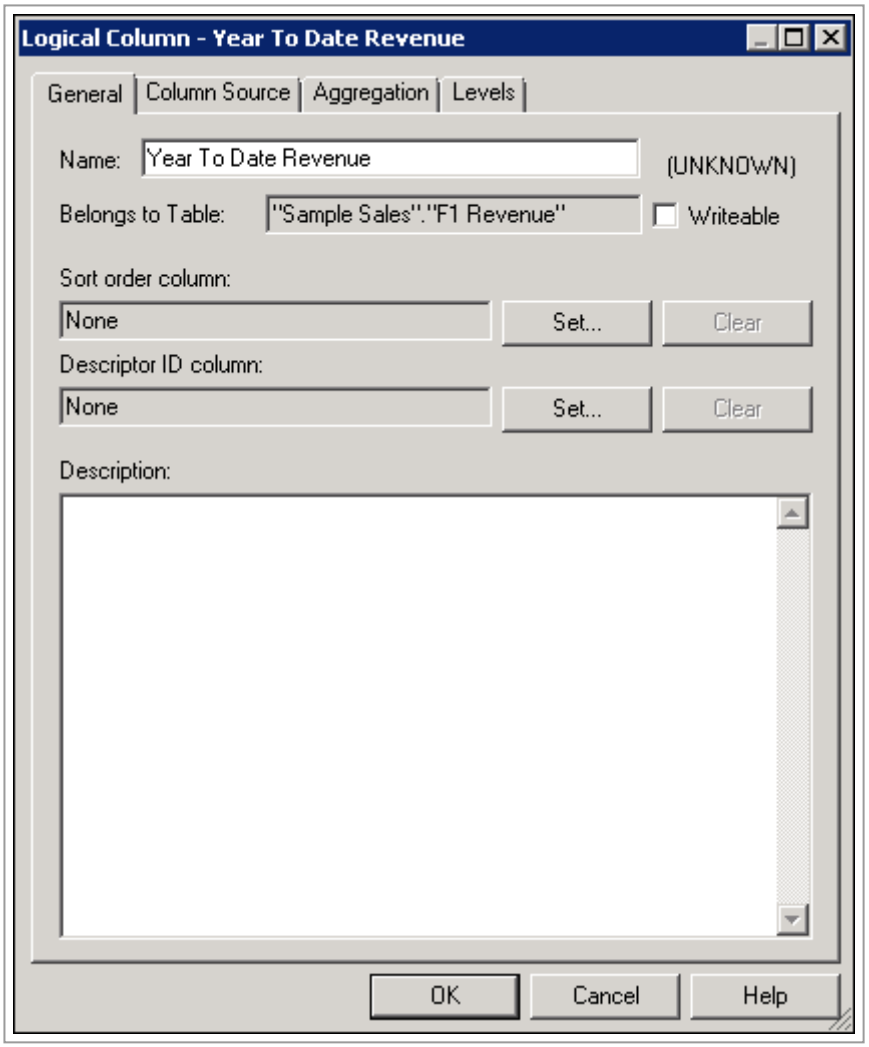

14. Drag the **Month Ago Revenue** logical column to the **Base Facts** presentation folder.



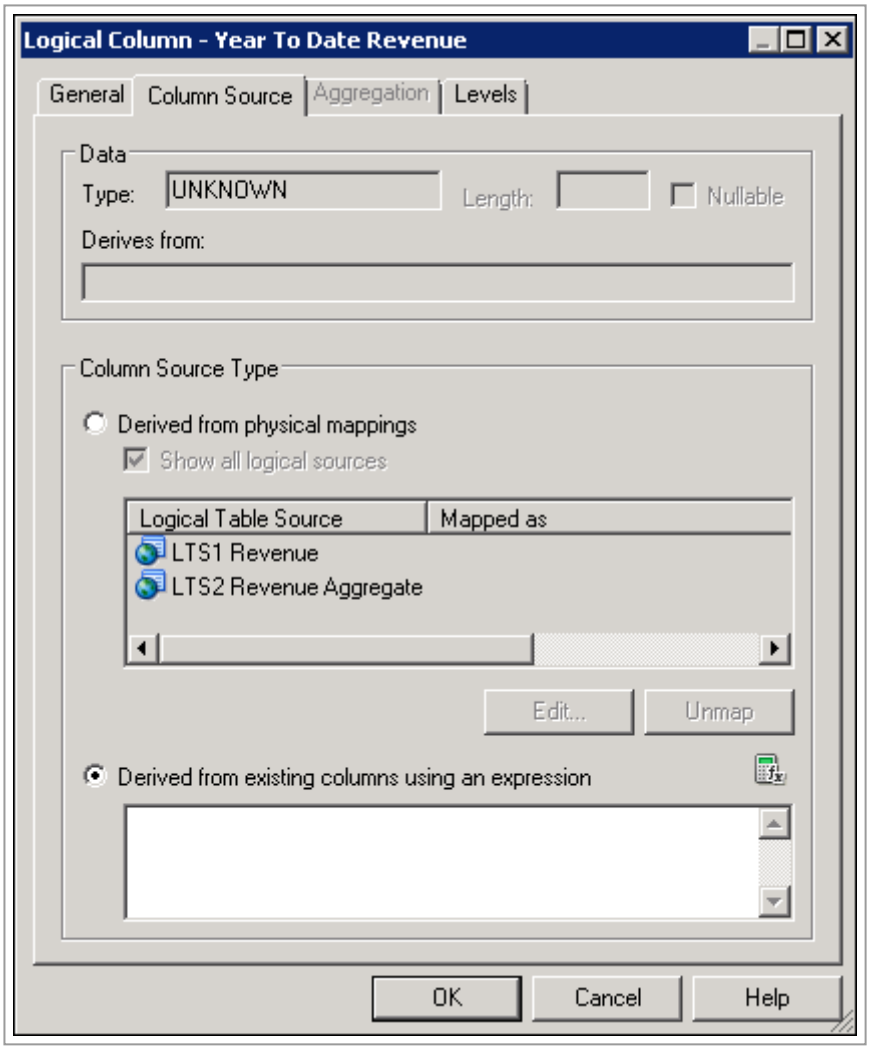

## Creating a Measure Using the TODATE Function

1. Right-click the **F1 Revenue** logical table and select **New Object > Logical Column**.

2. On the **General** tab, name the new logical column **Year To Date Revenue**.

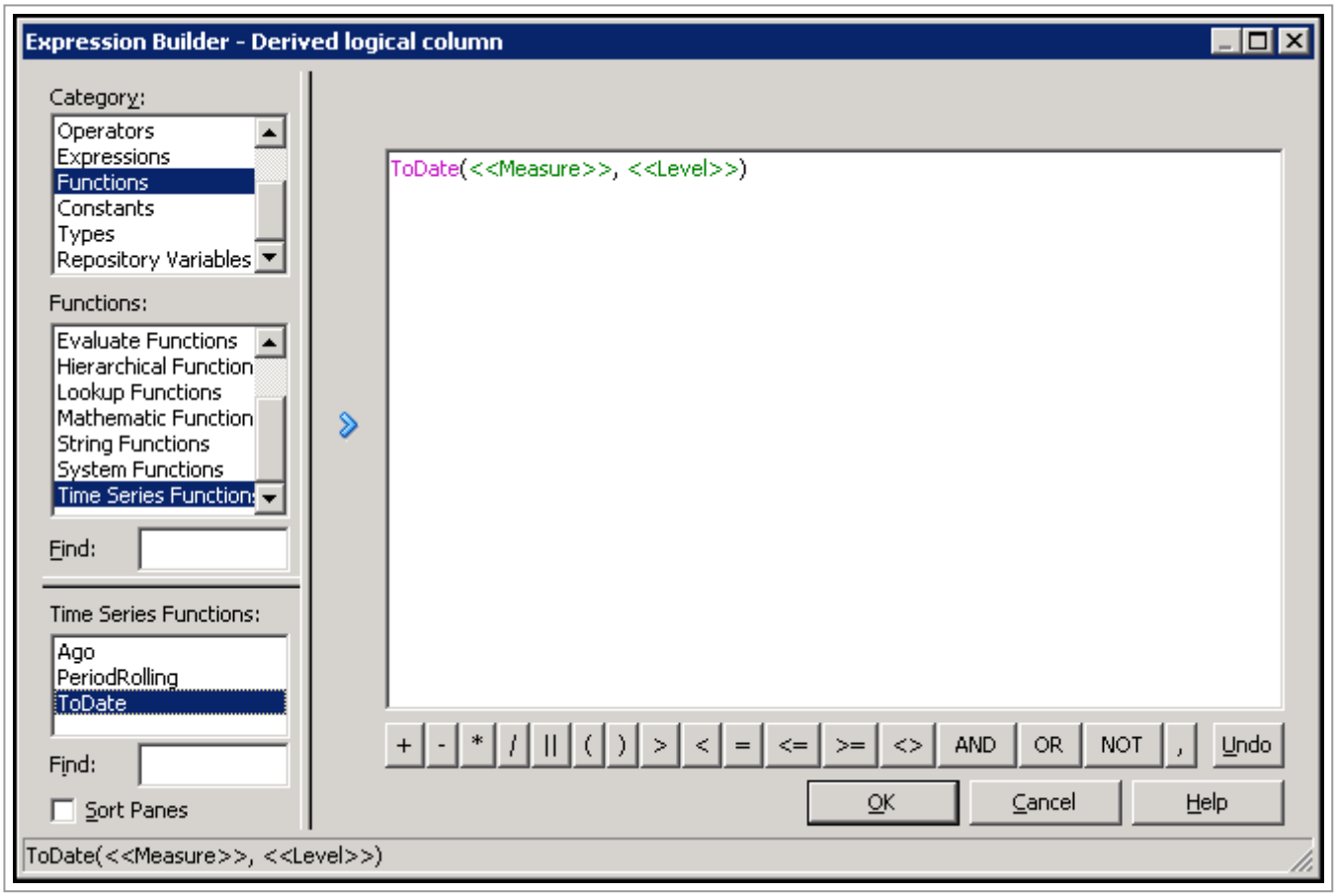



**3.** On the **Column Source** tab, select "Derived from existing columns using an expression".

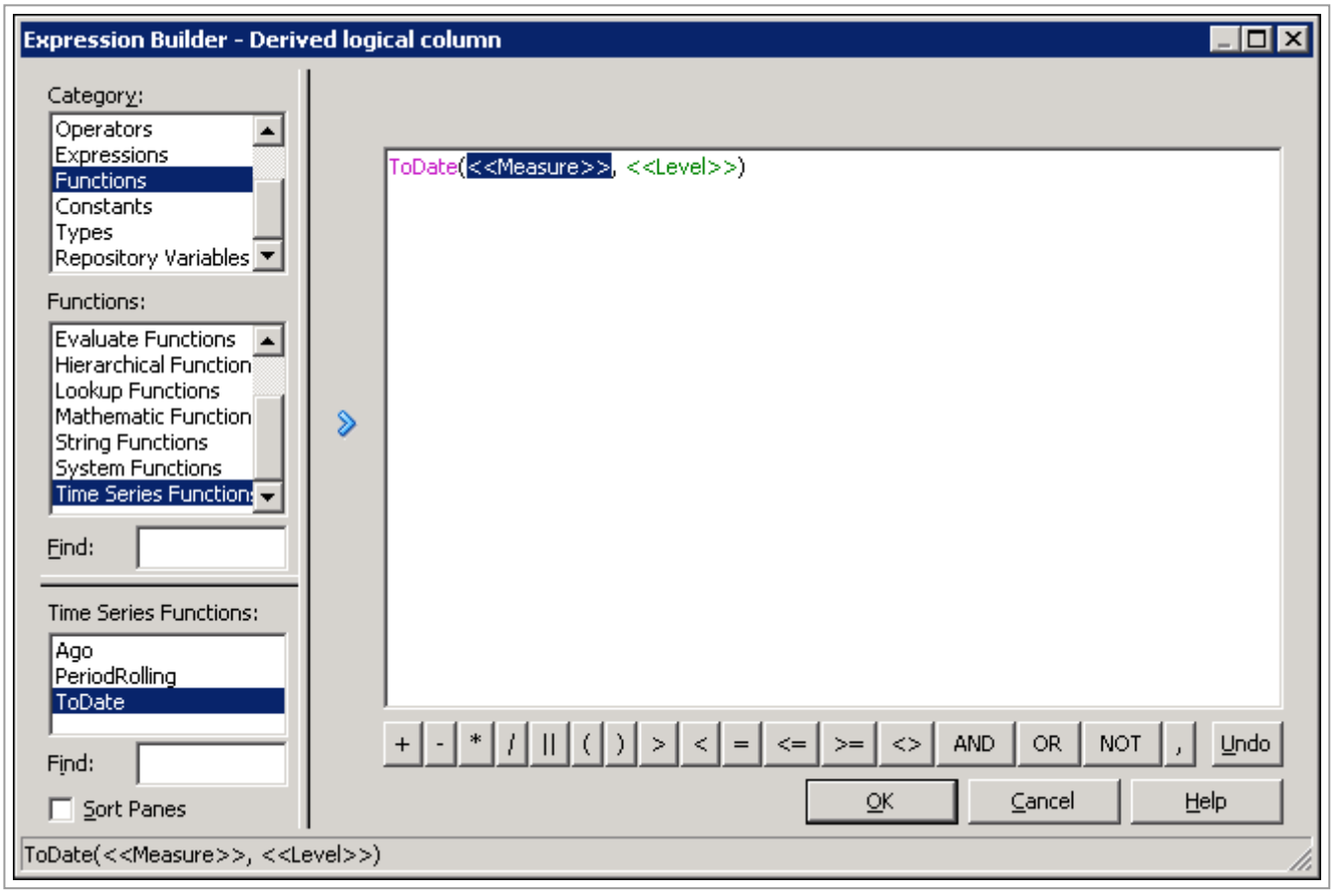



**4.** Open the Expression Builder.

**5.** Select **Functions > Time Series Functions** and double-click **ToDate** to insert the expression.

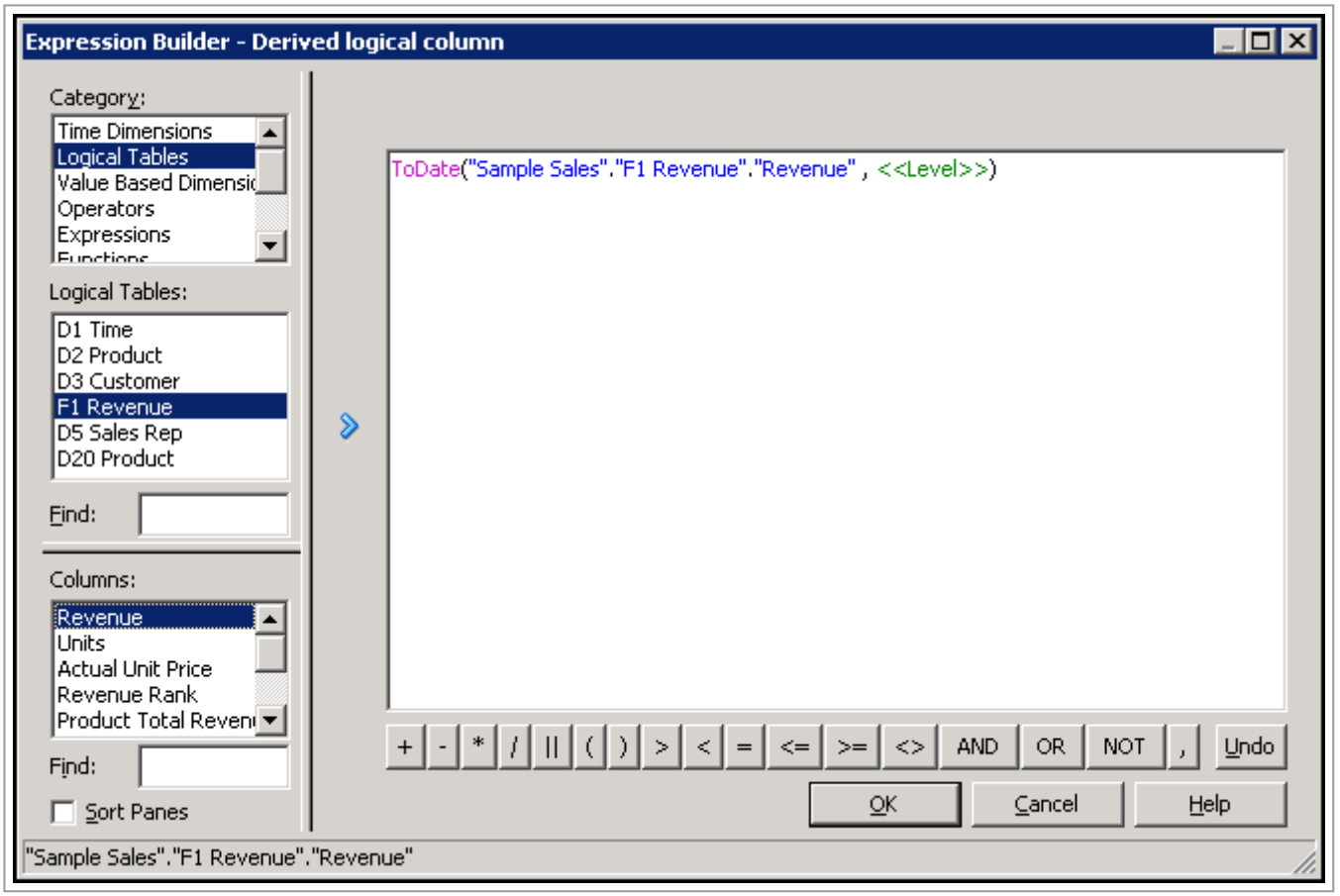



**6.** Click **<<Measure>>** in the expression.

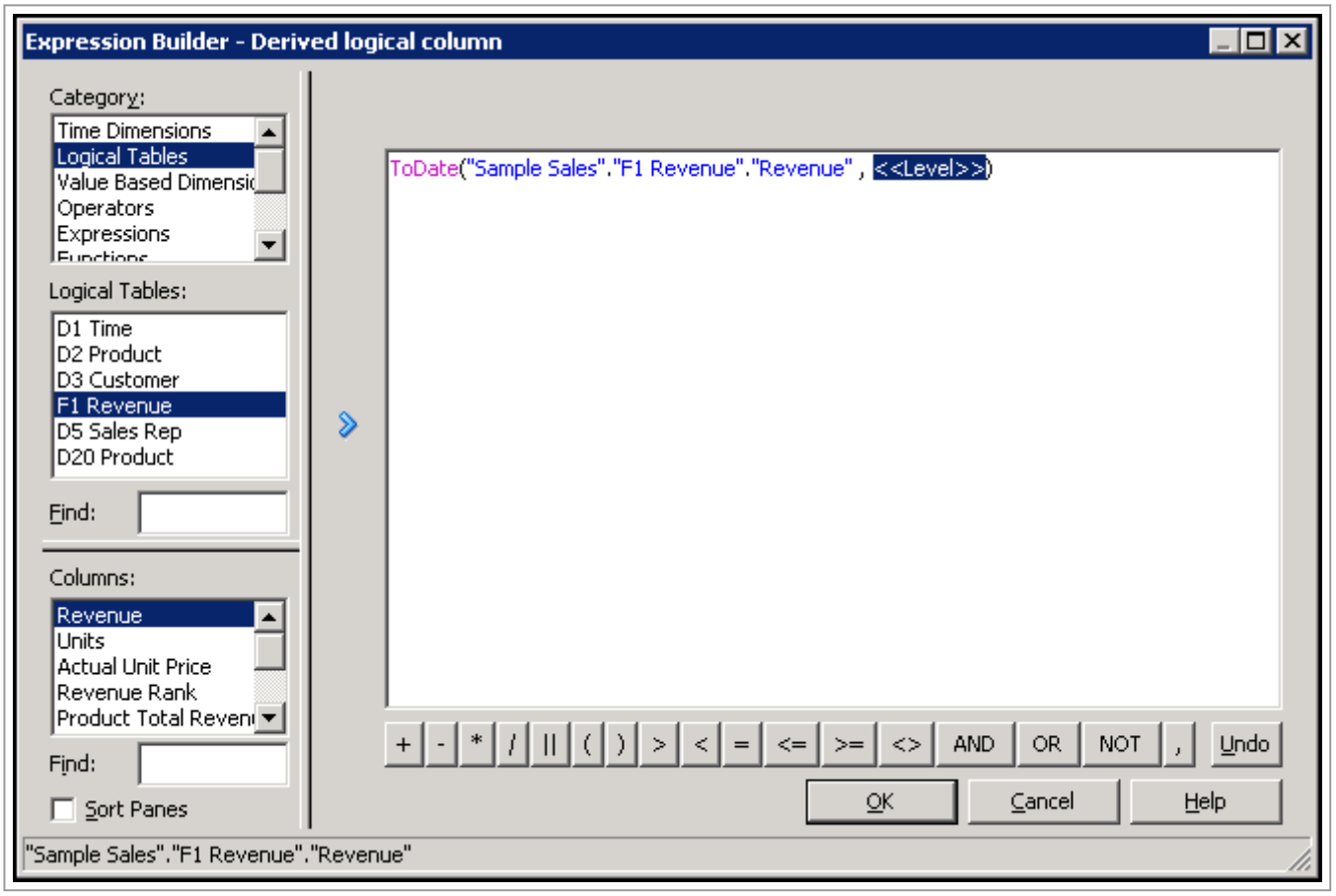



**7.** Select **Logical Tables > F1 Revenue** and then double-click **Revenue** to add it to the expression

7. Select **Logical Tables > F1 Revenue** and then double-click **Revenue** to add it to the expression.

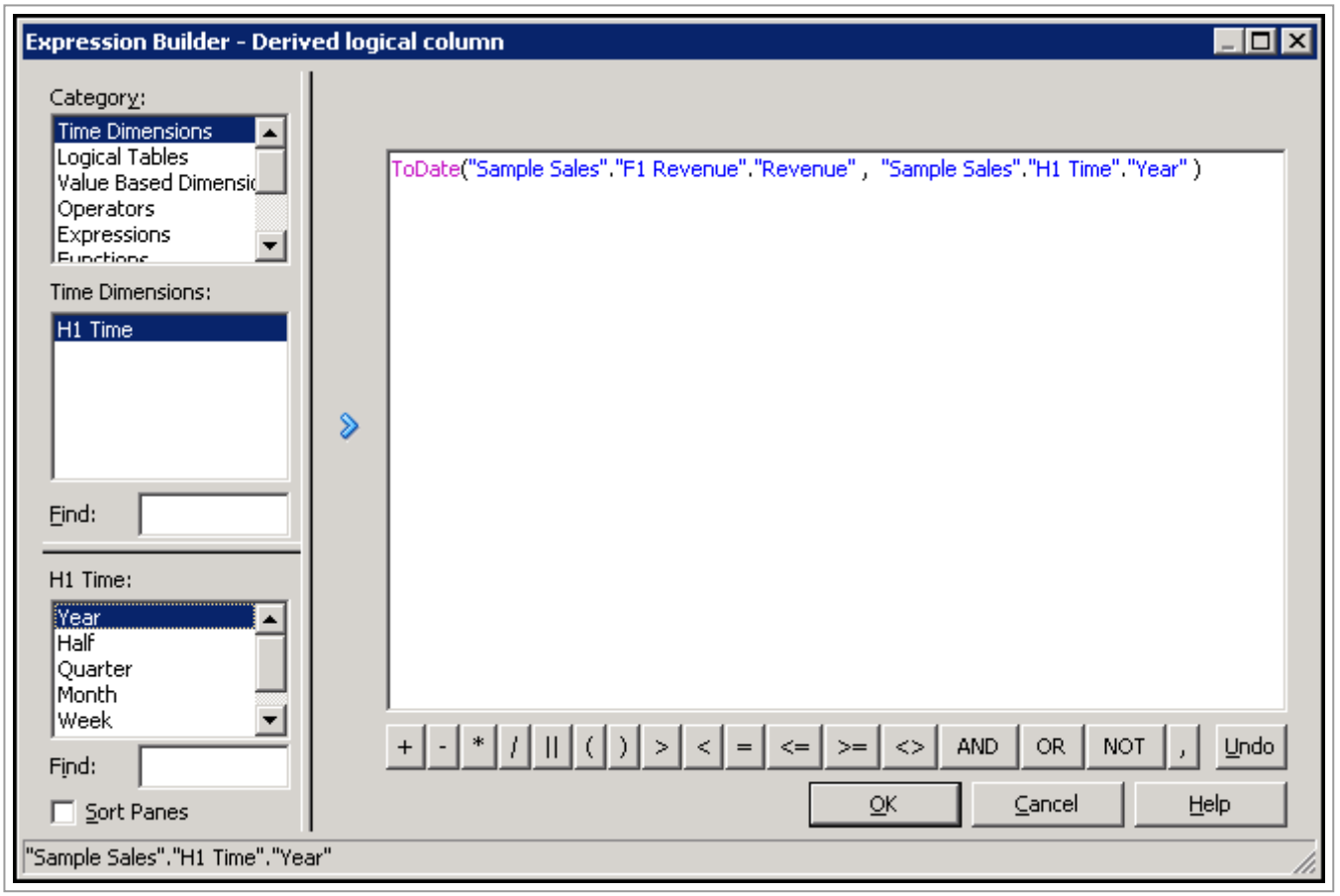



8. Click **<<Level>>** in the expression.



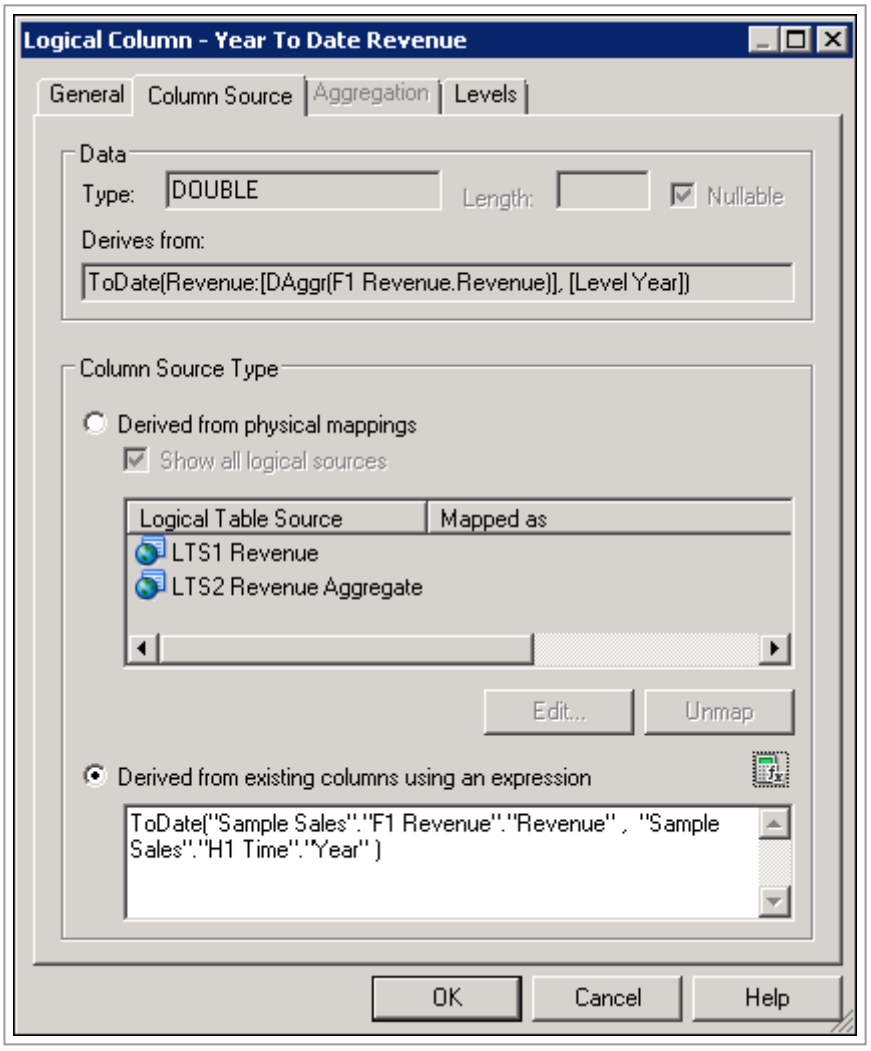

9. Select **Time Dimensions > H1 Time** and then double-click **Year** to add it to the expression.

9. Select **Time Dimensions > H1 Time** and then double-click **Year** to add it to the expression.



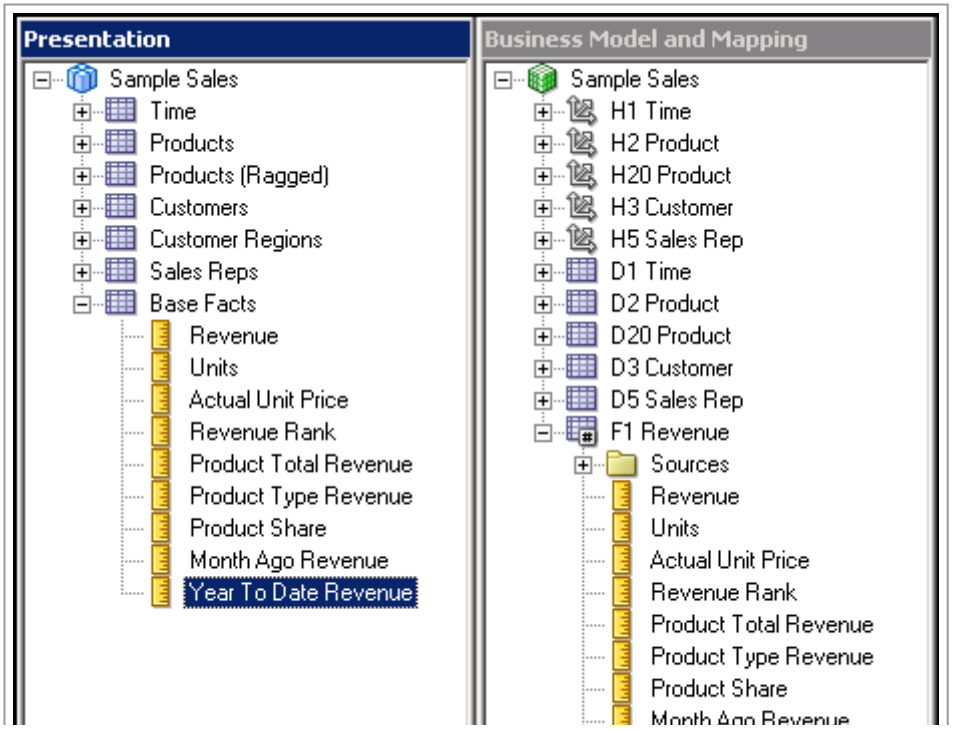

10. Click **OK** to close the Expression Builder.

11. Check your work in the Logical Column dialog box.




**12.** Click **OK** to close the Logical Column dialog box.

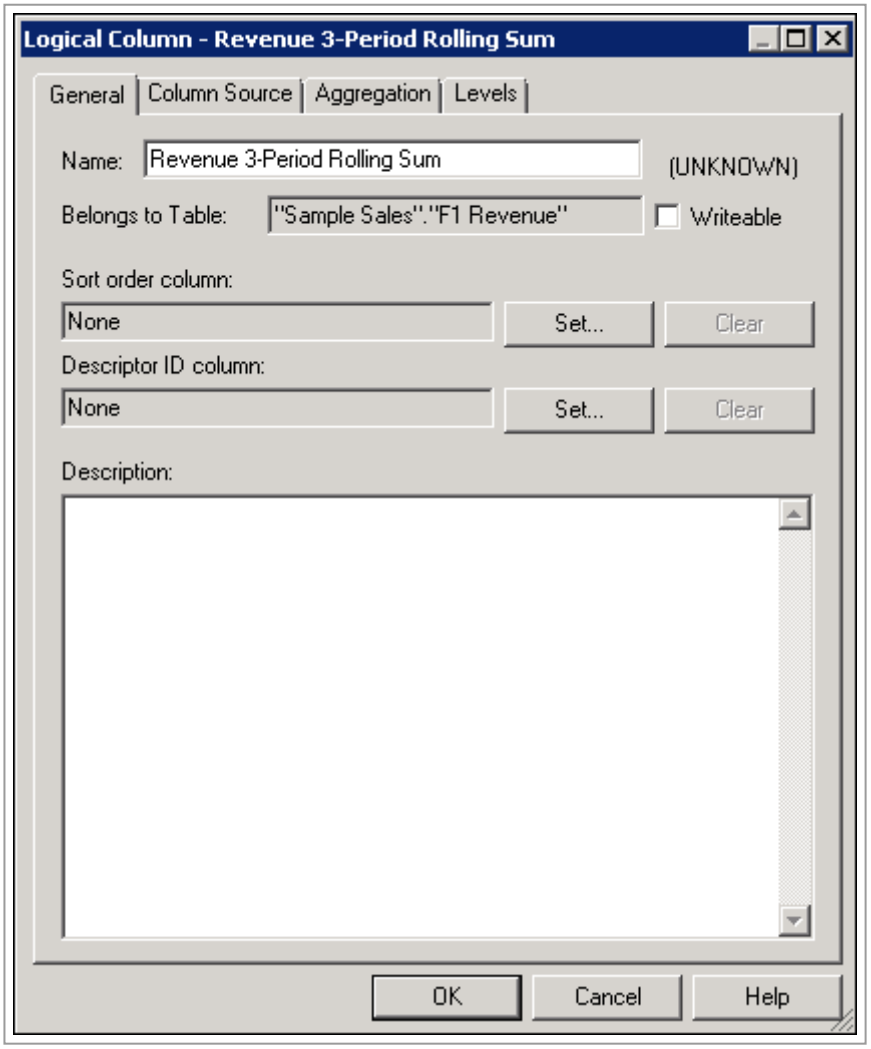
**13.** Drag the **Year To Date Revenue** logical column to the **Base Facts** presentation folder.



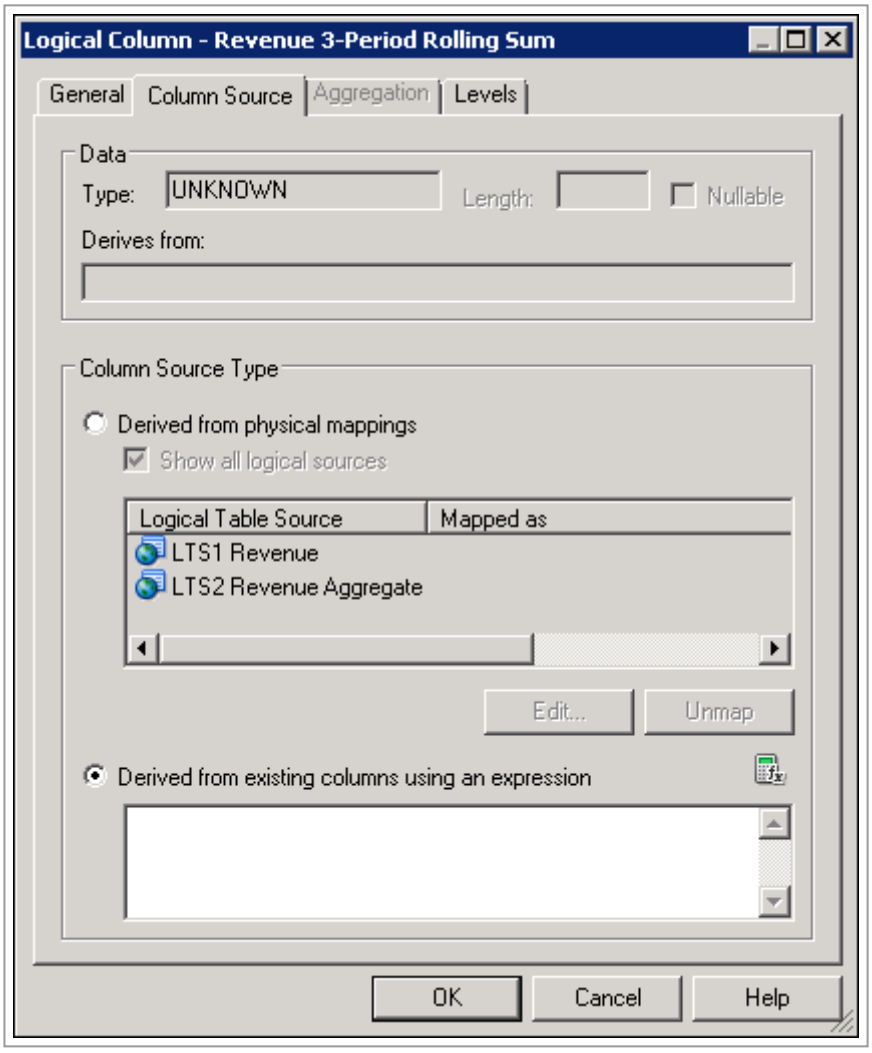

## Creating a Measure Using the PERIODROLLING Function

1. Right-click the **F1 Revenue** logical table and select **New Object > Logical Column**.

2. On the **General** tab, name the new logical column **Revenue 3-Period Rolling Sum**.

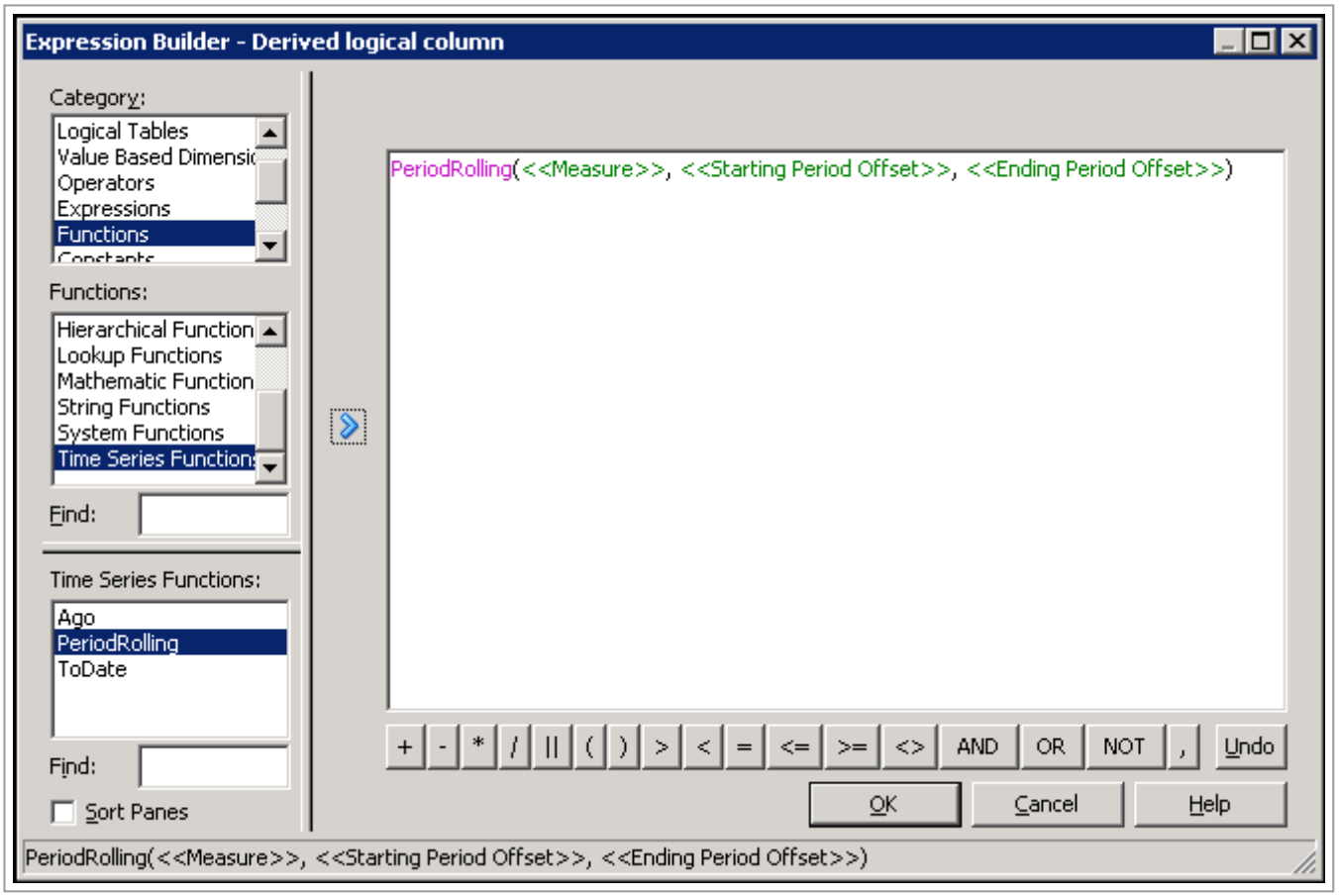



3. On the **Column Source** tab, select "Derived from existing columns using an expression".

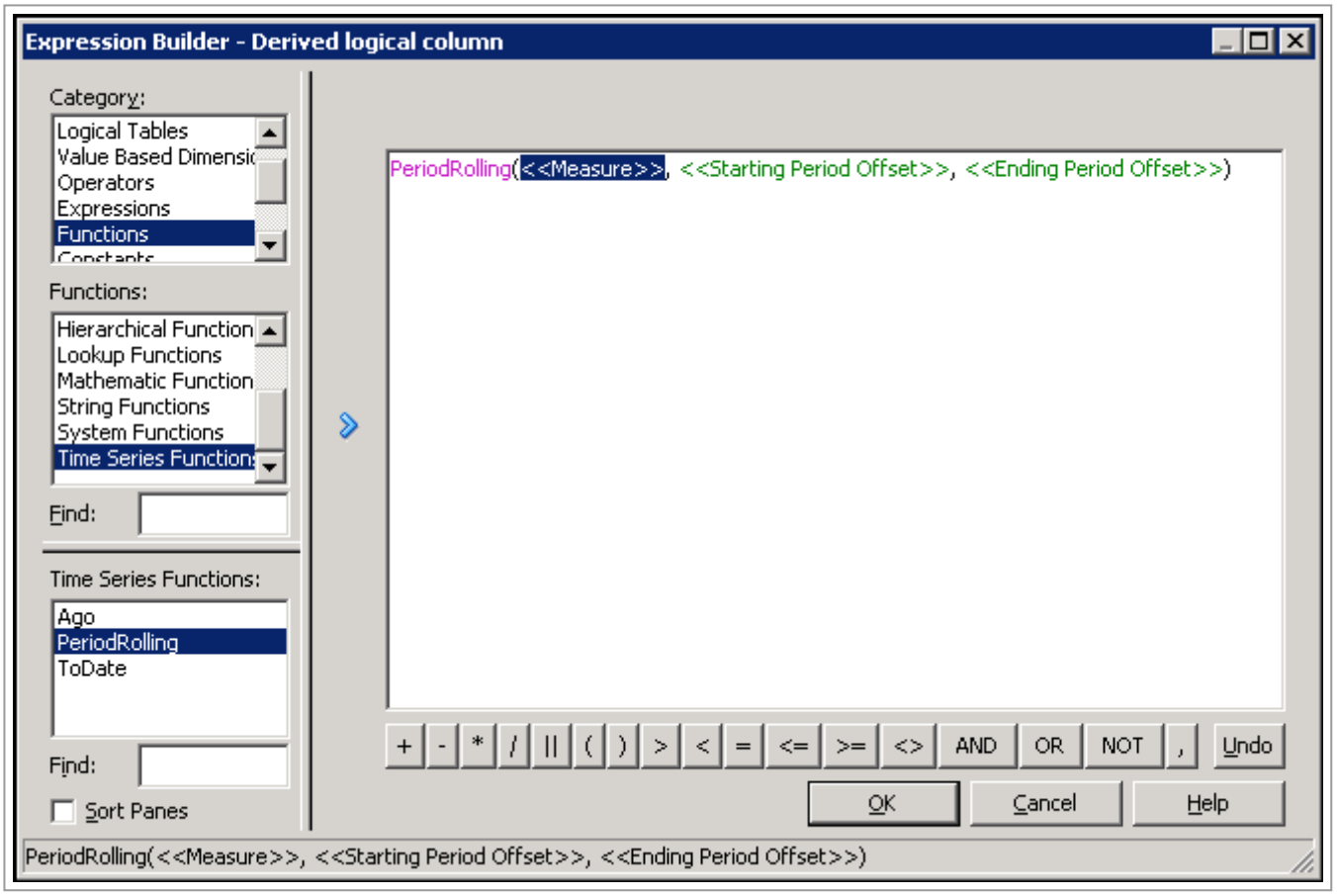



**4.** Open the Expression Builder.

**5.** Select **Functions > Time Series Functions** and double-click **PeriodRolling** to insert the expression.

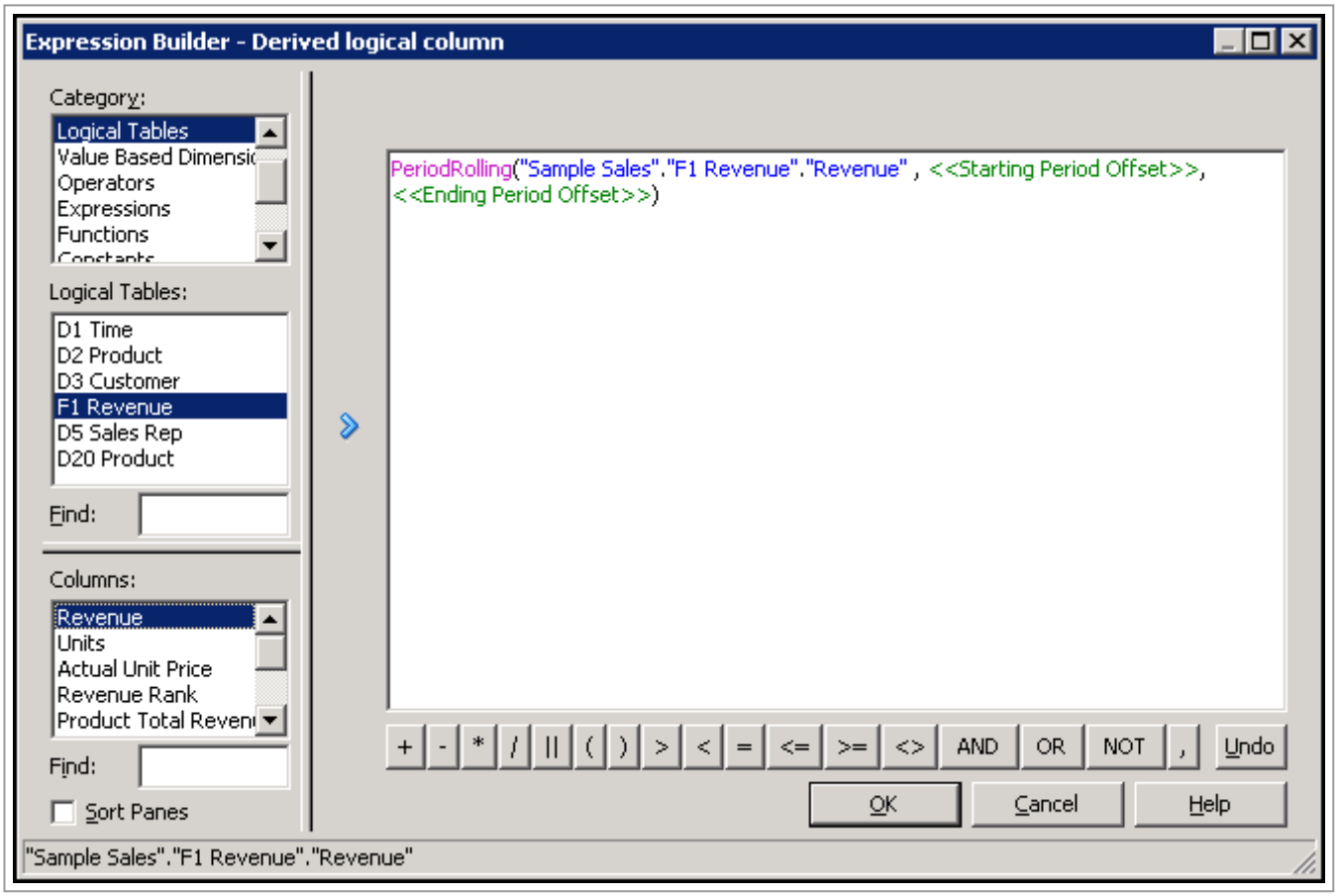



**6.** Click **<<Measure>>** in the expression.

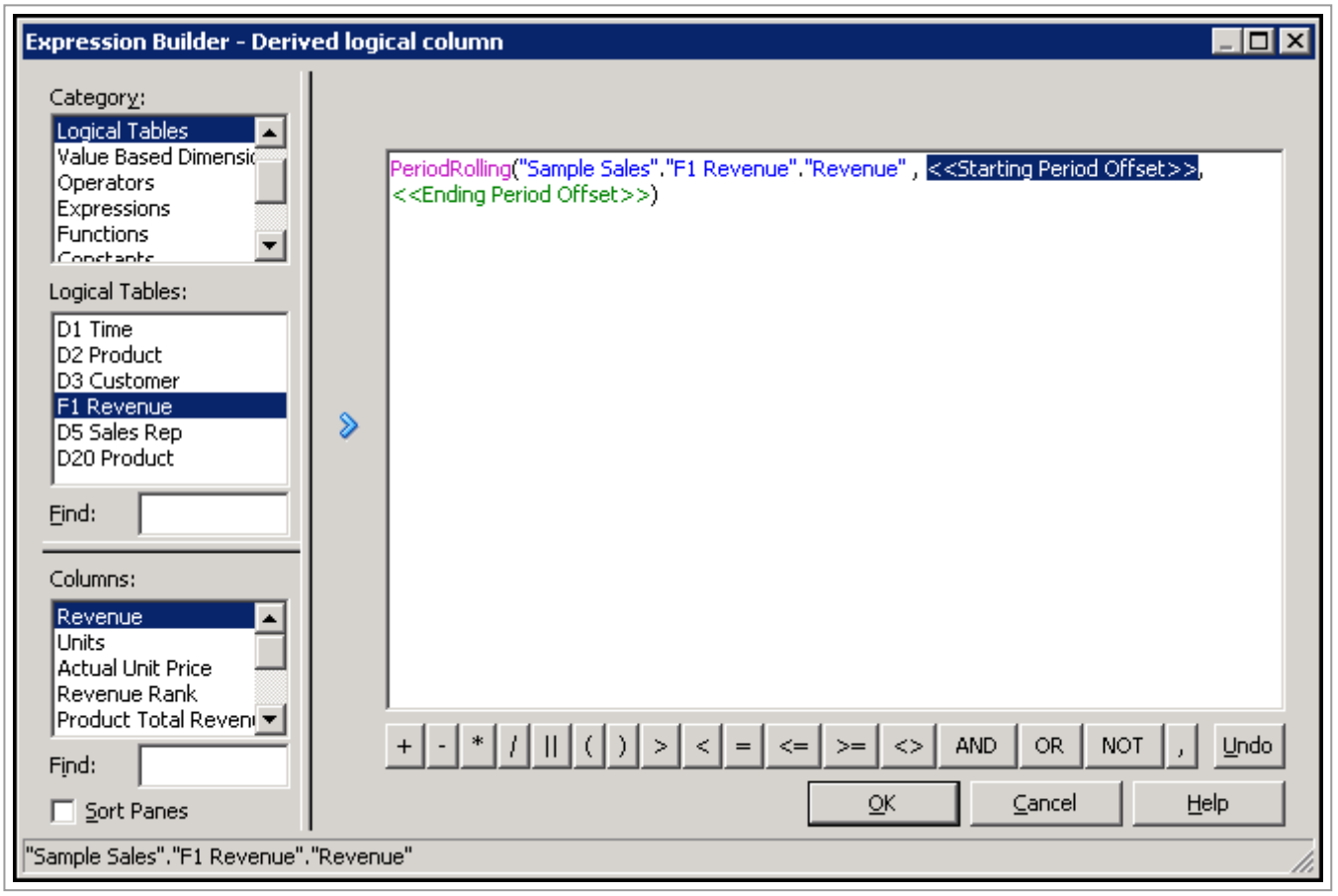



**7.** Select **Logical Tables > F1 Revenue** and then double-click **Revenue** to add it to the expression

7. Select **Logical Tables** > **F1 Revenue** and then double-click **Revenue** to add it to the expression.

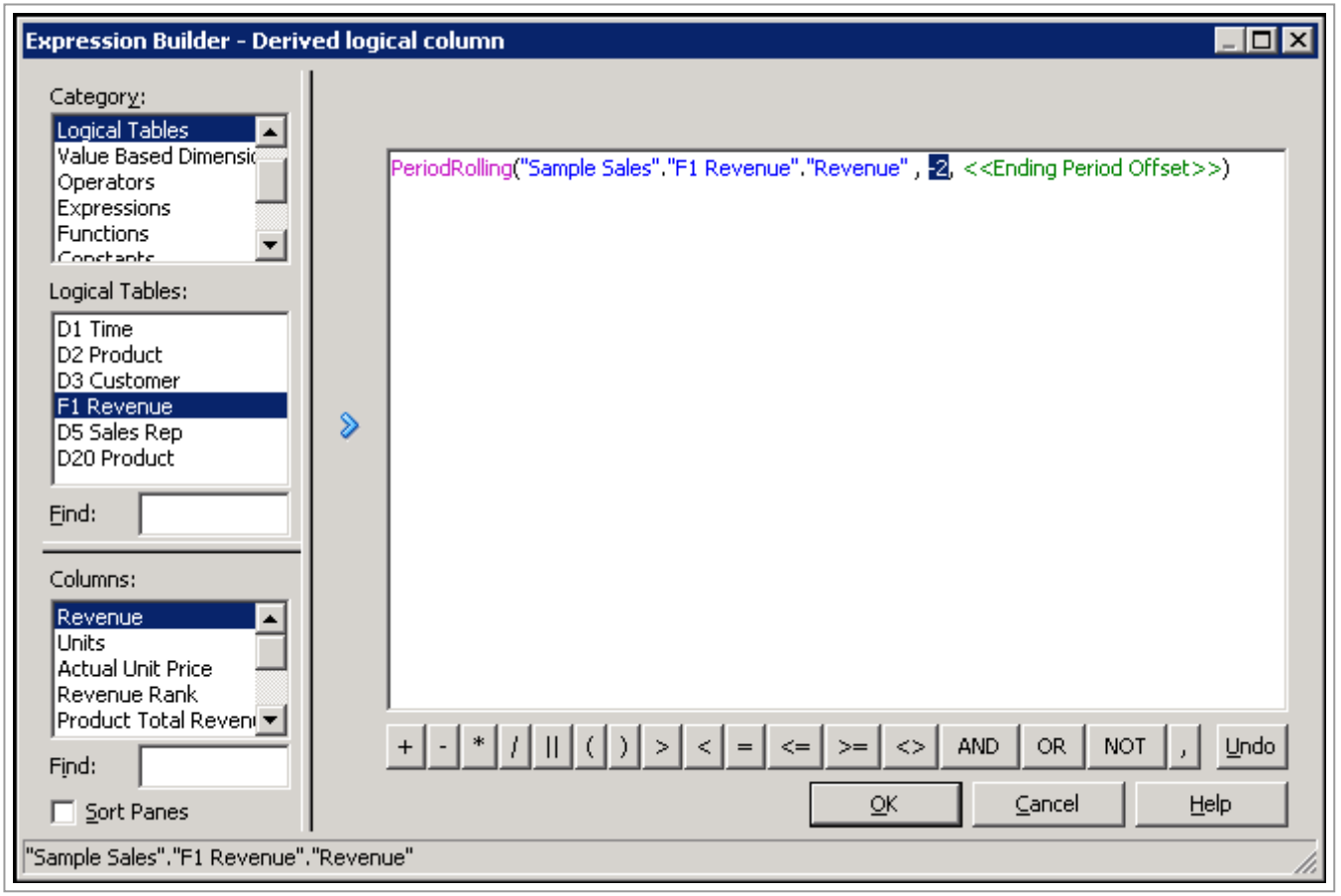



8. Click **<<Starting Period Offset>>** in the expression.

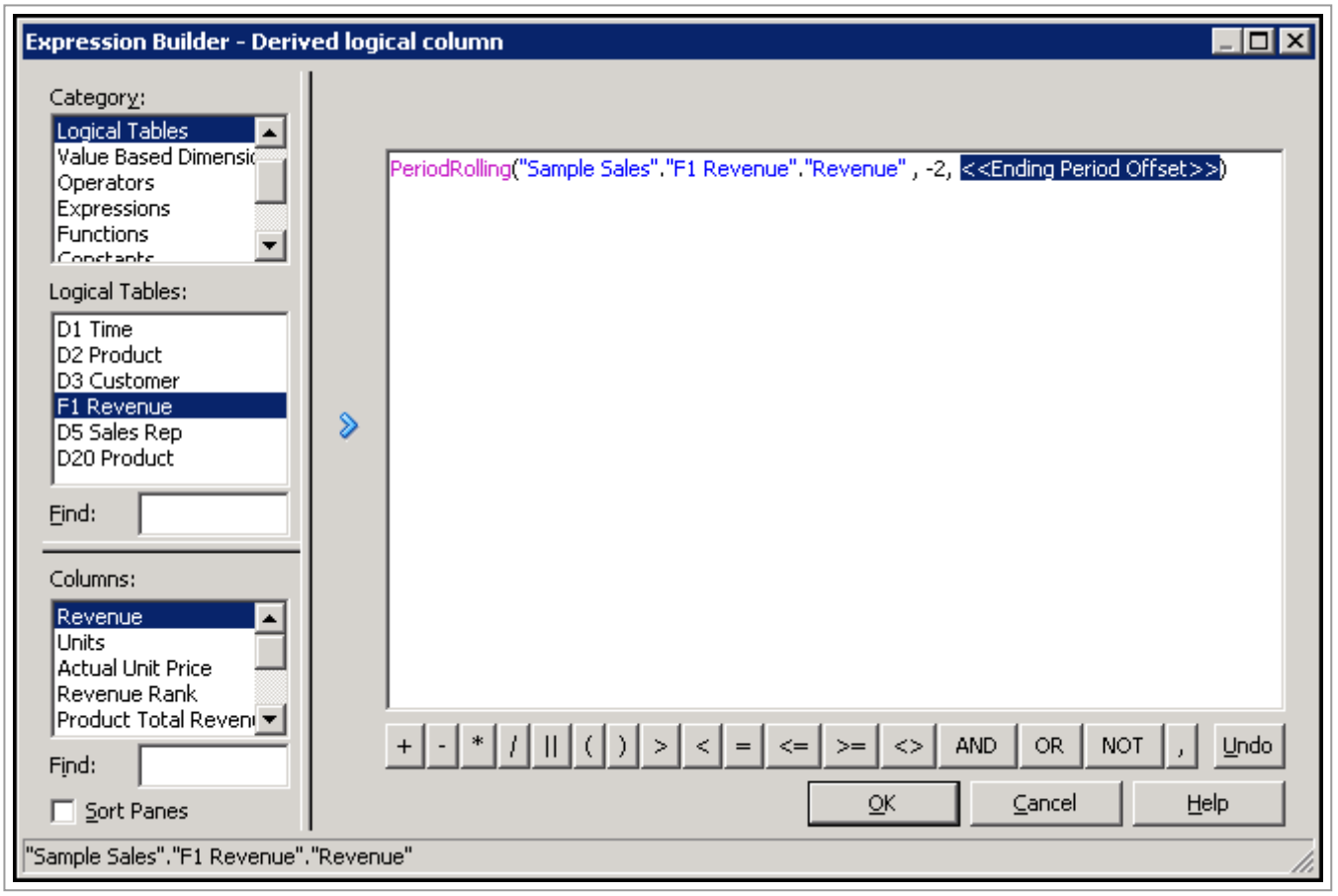



9. Enter **-2**. This identifies the first period in the rolling aggregation.

9. Enter **-2**. This identifies the first period in the rolling aggregation.

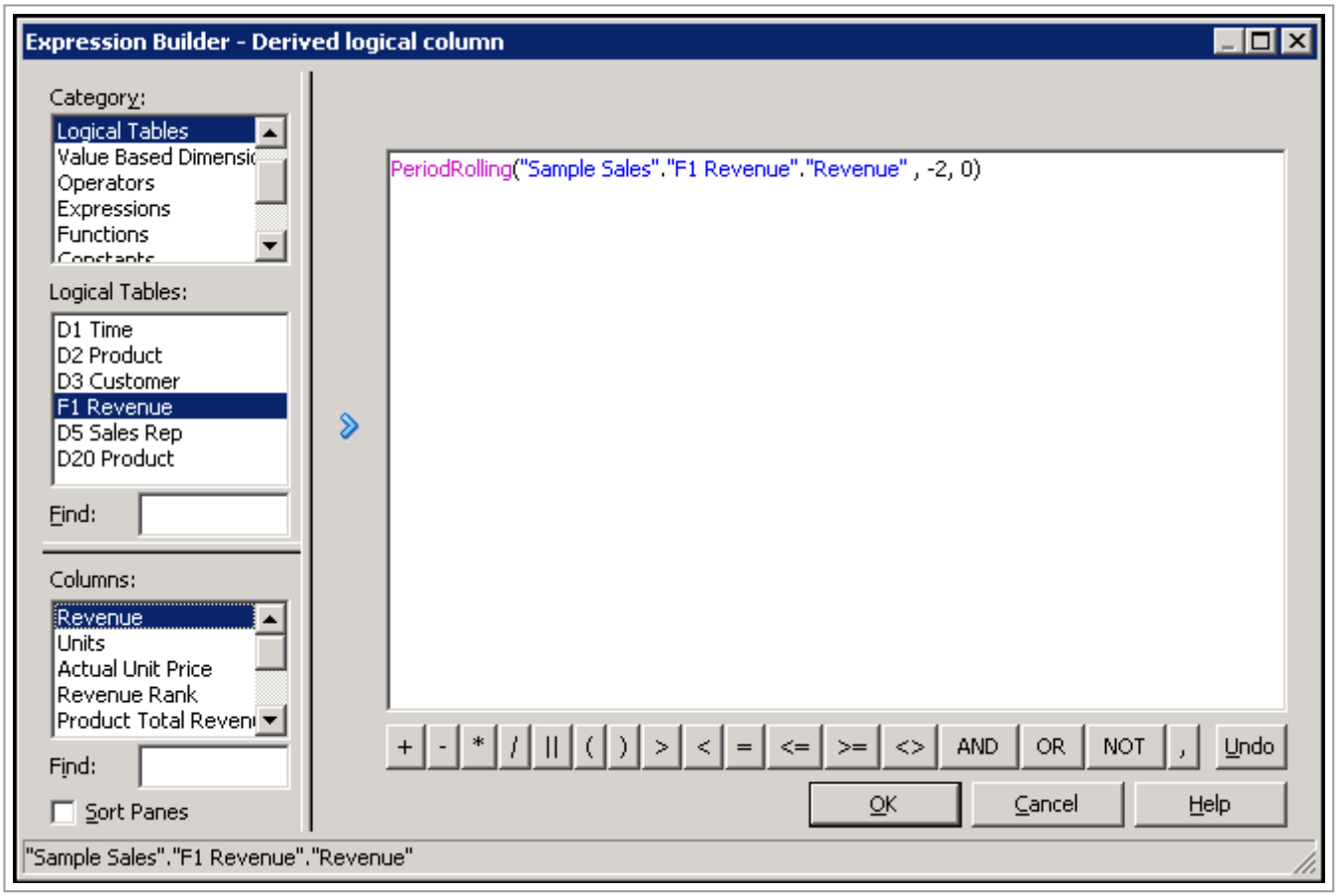



10. Click **<<Ending Period Offset>>**.




11. Enter **0**. This identifies the last period in the rolling aggregation.

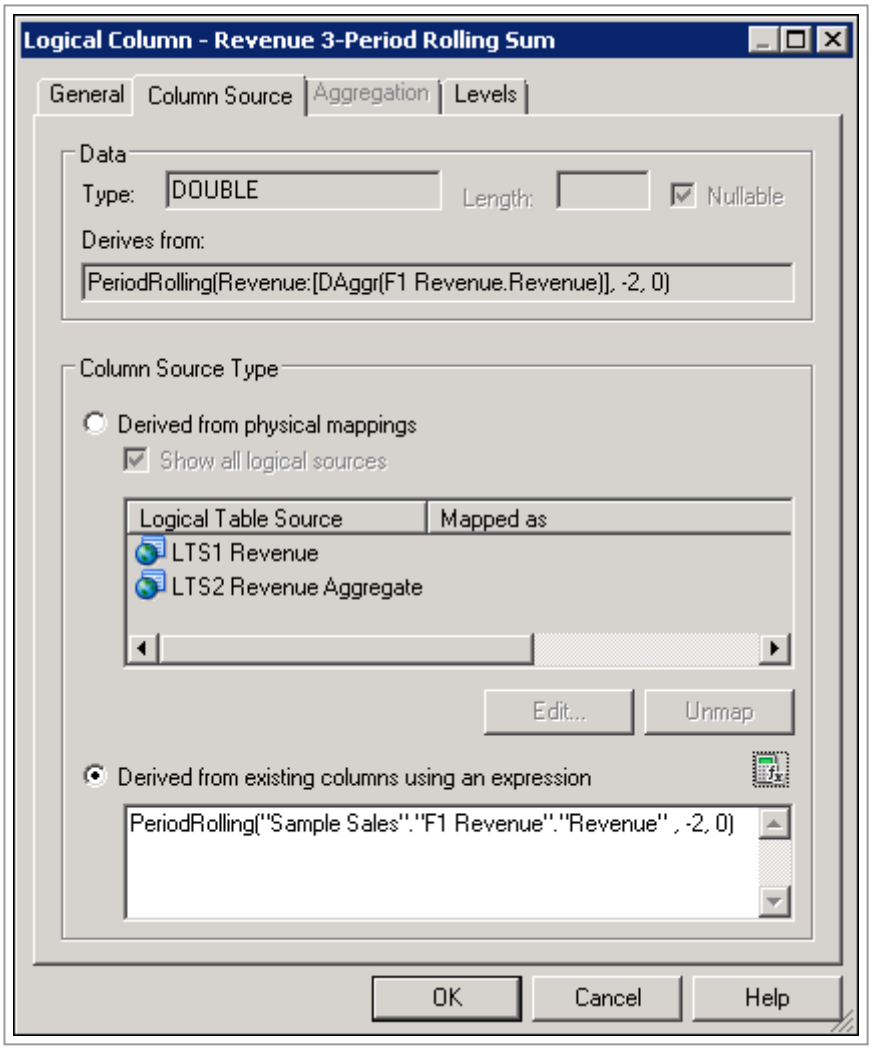
**11.** Enter **0**. This identifies the last period in the rolling aggregation.




These integers are the relative number of periods from a displayed period. In this example, if the query grain is month, the 3 month rolling sum starts two months in the past (-2) and includes the current month (0).

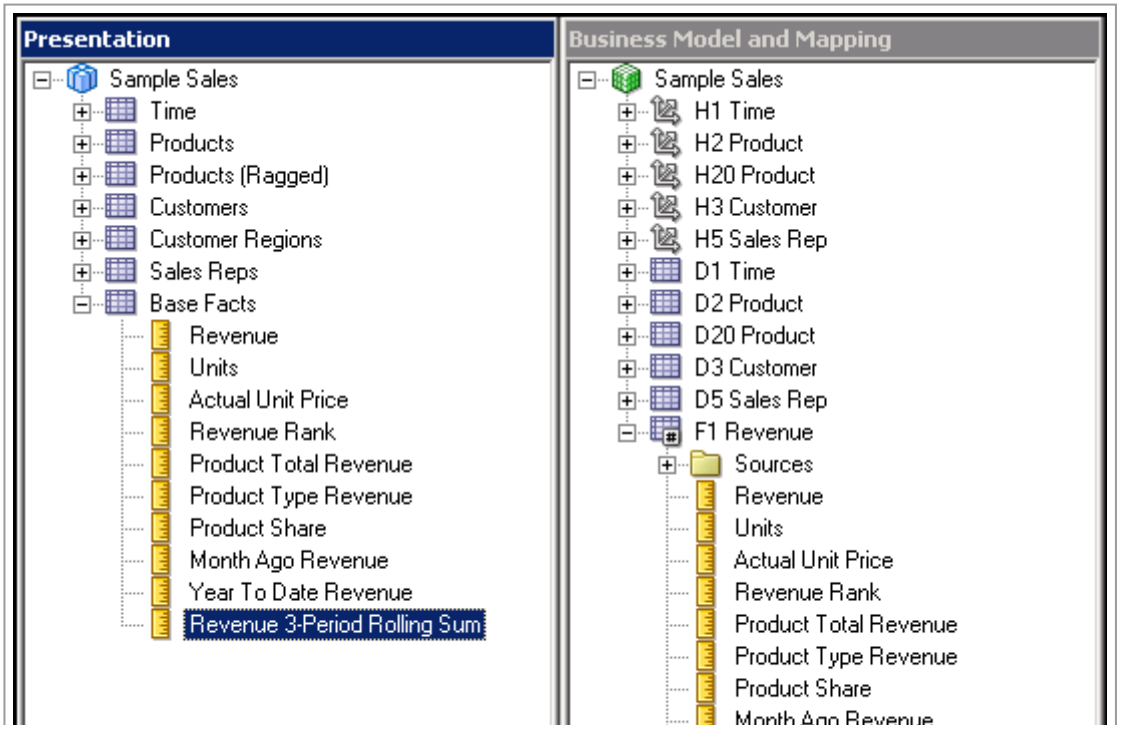
**12.** Click **OK** to close the Expression Builder.

**13.** Check your work in the Logical Column dialog box.




**14.** Click **OK** to close the Logical Column dialog box.

**15.** Drag the **Revenue 3-Period Rolling Sum** logical column to the **Base Facts** presentation folder.




16. Save the repository and check consistency. Fix any errors or warnings before you proceed.

17. Close the repository. Leave the Administration Tool open.

## Testing Your Work

1. Return to **data-model-cmd.cmd** utility and load the **BISAMPLE** repository.

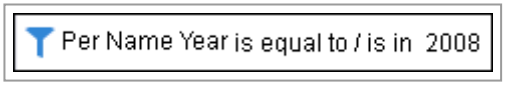
2. Return to **Oracle BI** and sign in.

3. Create the following analysis to test AGO and TODATE functions:

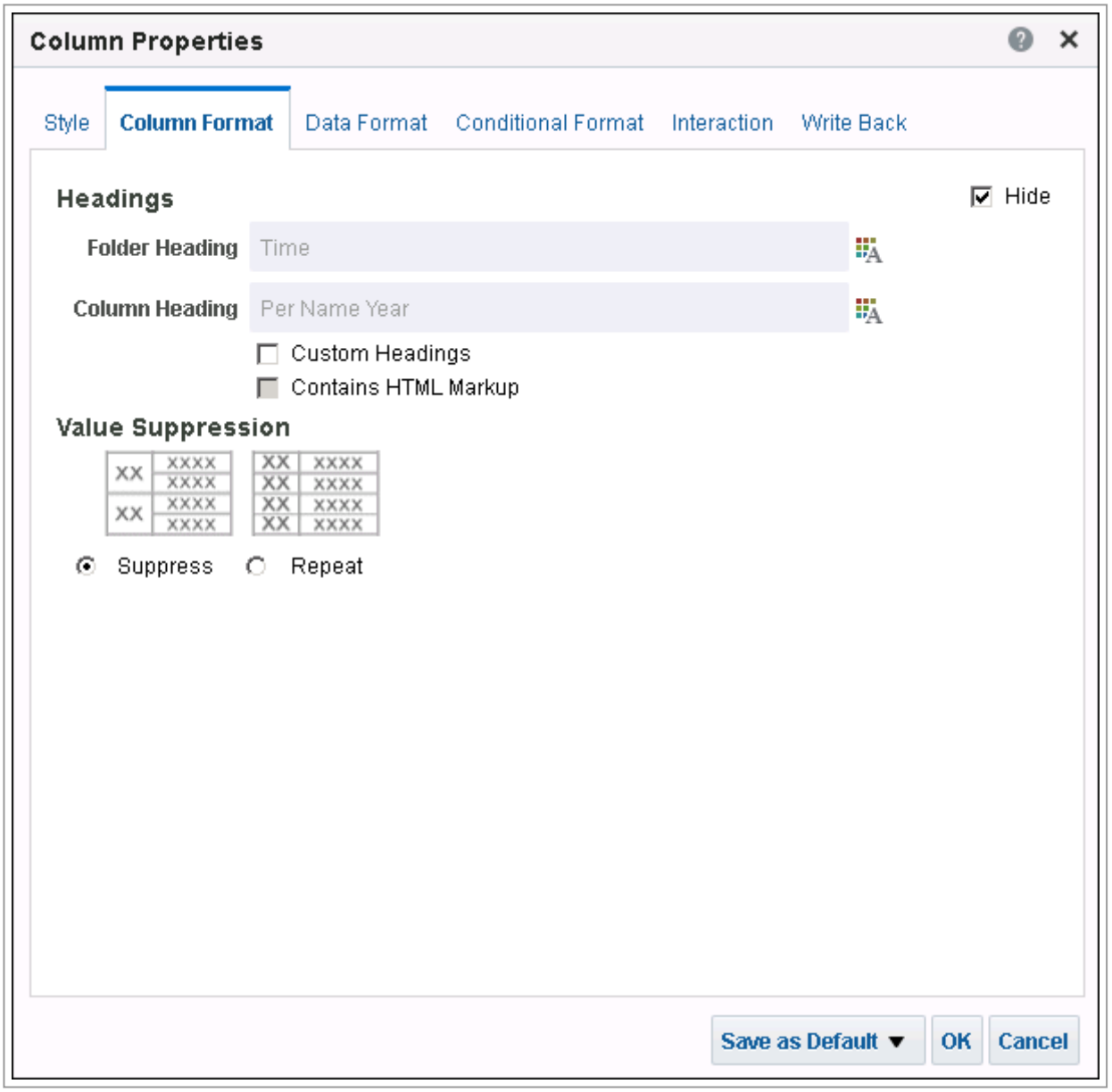
   **Time.Per Name Month**
   **Time.Per Name Year**
   **Base Facts.Revenue**
   **Base Facts.Month Ago Revenue**
   **Base Facts.Year to Date Revenue**


4. Set the following filter for the analysis:

   **Per Name Year is equal to / is in 2008**.




5. For the **Per Name Year** column, select **Column Properties > Column Format > Hide**. This will prevent Per Name Year from displaying in the analysis results.




**6.** Sort **Per Name Month** in ascending order.


**7.** Click **Results**.

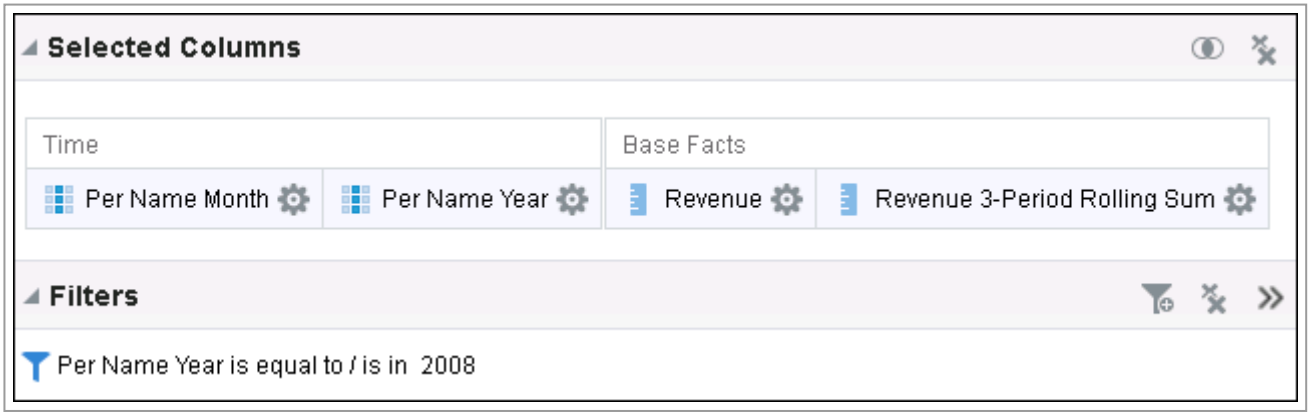
| Per Name Month | Revenue | Month Ago Revenue | Year To Date Revenue |
|---|---|---|---|
| 2008 / 01 | 325,436 | | 325,436 |
| 2008 / 02 | 812,399 | 325,436 | 1,137,835 |
| 2008 / 03 | 1,569,851 | 812,399 | 2,707,686 |
| 2008 / 04 | 2,069,683 | 1,569,851 | 4,777,369 |
| 2008 / 05 | 2,636,227 | 2,069,683 | 7,413,596 |
| 2008 / 06 | 3,403,806 | 2,636,227 | 10,817,402 |
| 2008 / 07 | 2,372,854 | 3,403,806 | 13,190,256 |
| 2008 / 08 | 1,235,079 | 2,372,854 | 14,425,335 |
| 2008 / 09 | 730,912 | 1,235,079 | 15,156,246 |
| 2008 / 10 | 498,939 | 730,912 | 15,655,185 |
| 2008 / 11 | 526,537 | 498,939 | 16,181,722 |
| 2008 / 12 | 318,278 | 526,537 | 16,500,000 |

**Month Ago Revenue** displays revenue from the previous month. **Year To Date Revenue** calculates a running sum of revenue for the year on a monthly basis.

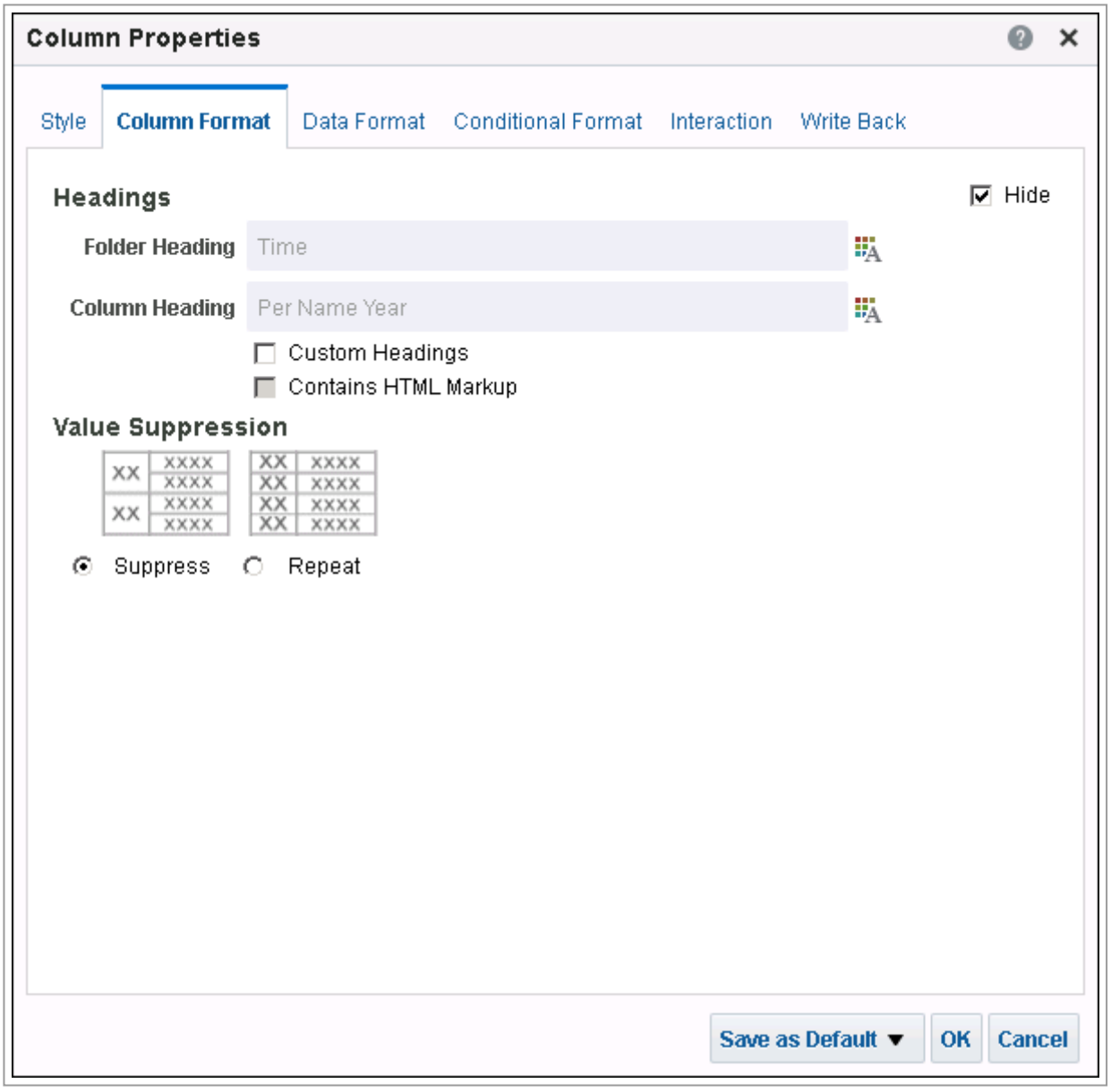
8. Create the following new analysis and filter to test the PERIODROLLING function at the month grain:

**Time.Per Name Month**
**Time.Per Name Year**
**Base Facts.Revenue**
**Base Facts.Revenue 3-Period Rolling Sum**

**Per Name Year is equal to / is in 2008**




9. For the **Per Name Year** column, select **Column Properties > Column Format > Hide**. This will prevent Per Name Year from displaying in the analysis results.




**10.** Sort **Per Name Month** in ascending order.

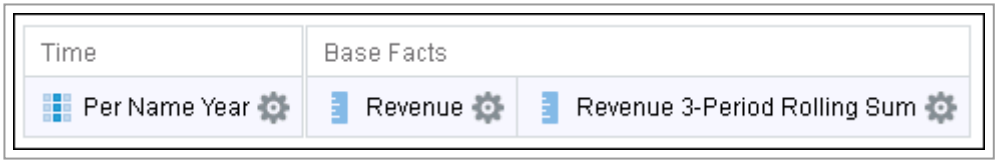

**11.** Click Results.

| Per Name Month | Revenue | Revenue 3-Period Rolling Sum |
|---|---|---|
| 2008 / 01 | 325,436 | 325,436 |
| 2008 / 02 | 812,399 | 1,137,835 |
| 2008 / 03 | 1,569,851 | 2,707,686 |
| 2008 / 04 | 2,069,683 | 4,451,933 |
| 2008 / 05 | 2,636,227 | 6,275,761 |
| 2008 / 06 | 3,403,806 | 8,109,716 |
| 2008 / 07 | 2,372,854 | 8,412,887 |
| 2008 / 08 | 1,235,079 | 7,011,739 |
| 2008 / 09 | 730,912 | 4,338,844 |
| 2008 / 10 | 498,939 | 2,464,930 |
| 2008 / 11 | 526,537 | 1,756,388 |
| 2008 / 12 | 318,278 | 1,343,754 |

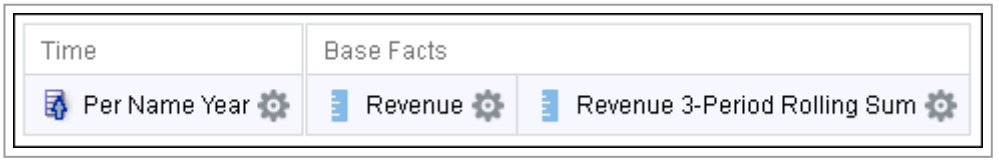
**Revenue 3-Period Rolling Sum** is calculated based on the month grain.

**12.** Create the following new analysis and filter to test the PERIODROLLING function at the year grain:

**Time.Per Name Year**
**Base Facts.Revenue**
**Base Facts.Revenue 3-Period Rolling Sum**




**13.** Sort **Per Name Year** in ascending order.




**14.** Click **Results**.

| Per Name Year | Revenue | Revenue 3-Period Rolling Sum |
|---|---|---|
| 2008 | 16,500,000 | 16,500,000 |
| 2009 | 15,000,000 | 31,500,000 |
| 2010 | 18,500,000 | 50,000,000 |
| 2011 | | 33,500,000 |

Revenue 3-Period Rolling Sum is calculated based on the year grain. A measure with the PERIODROLLING function calculates results based on the query grain.

## Want to Learn More?

- Oracle Learning Library (http://www.oracle.com/goto/oll)
- Oracle Business Intelligence Enterprise Edition Documentation