DAT225x

Developing an Analysis Services Tabular Model

Lab 03 | Enhancing the Tabular Model

Estimated time to complete this lab is 60 minutes

Overview

In this lab, you will enhance each of the model tables to support an intuitive and friendly experience, and also encapsulate business logic with measures and a KPI.

Note: The four labs in this course are accumulative. You cannot complete this lab if you did not successfully complete **Lab 02**.

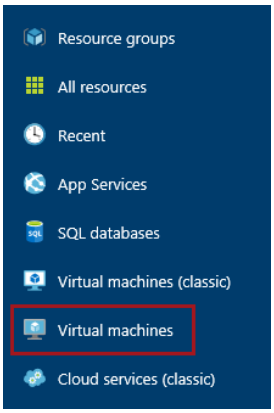
It is possible to commence from the solution available in the **F:\Labs\Lab02\Solution** folder, providing that you execute **F:\Labs\Lab02\Assets\Script-01.sql** first.

Getting Started

In this exercise, you will start the VM provisioned in **Lab 01**. You will then connect to the VM to complete the exercises in this lab.

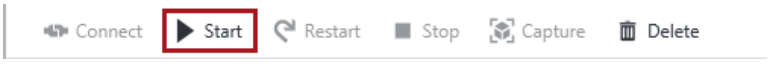
1. Sign in to the **Azure Portal** by using your subscription.

2. In the left pane, select **Virtual Machines**—do not select **Virtual Machines (Classic)**.



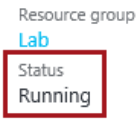
3. In the **Virtual Machines** blade, select the VM you provisioned in **Lab 01**.

4. In the VM blade, click **Start**.



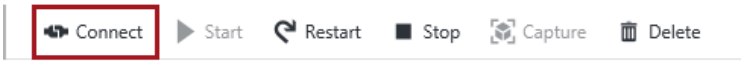
5. Wait for the VM status to update to **Running**.

   It usually takes 1-2 minutes for the VM to start.



6. To connect to the VM, click **Connect**.

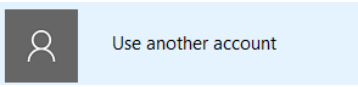
   Take care not to use the RDP file downloaded in the previous lab. It is likely that a different IP address has be assigned.



7. When prompted to open the Remote Desktop File, click **Open**.

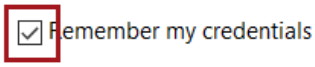
8. If prompted to connect to the unknown publisher, click **Connect**.

9. If prompted, in the **Windows Security** dialog window, click **Use Another Account**.



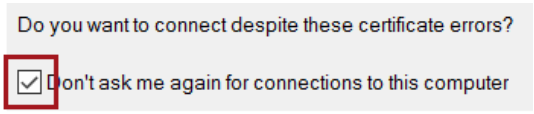
10. Enter the credentials you created for your VM.

11. Check the **Remember My Credentials** checkbox.



12. Click **OK**.

13. In the **Remote Desktop Connection** dialog window, check the **Don't Ask Me Again for Connections to This Computer** checkbox.



14. Click **Yes**.

15. Open SSDT.

16. On the **File** menu, select **Recent Projects and Solutions** to re-open your project and the model designer.

Exercise 1: Enhancing the Model Interface

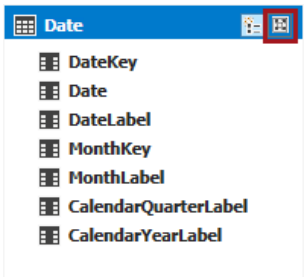
In this exercise, you will enhance each table in the model by renaming columns, hiding columns not intended for reporting, sorting column values, and creating hierarchies.

Enhancing the Date Table

In this task, you will enhance the **Date** table.

1. In the model designer, switch to Diagram View.

2. Hover the cursor over the **Date** table, and then at the top-right corner of the table, click **Maximize**.



   Maximizing the table is a very convenient way to view its definition and to configure it.

3. To rename the **DateLabel** column, right-click the column, and then select **Rename**.

   Tip: You can also double-click the column to rename it.

4. Rename the column as **Day**, and then press **Enter**.

5. Rename also the following columns.

Column	New Column Name
MonthLabel	Month
CalendarQuarterLabel	Calendar Quarter
CalendarYearLabel	Calendar Year

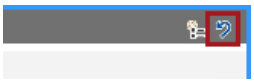
6. To hide the **DateKey** column, right-click the column, and then select **Hide from Client Tools**.

   Hidden columns are not made available by reporting tools.

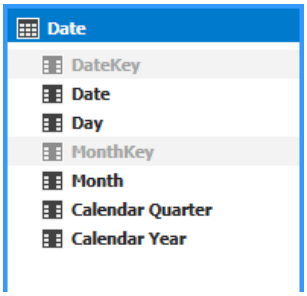
7. Also hide the **MonthKey** column.

   *This column will be used to sort the **Month** column values.*

8. To restore the table to normal size, at the top-right corner of the table, click **Restore**.

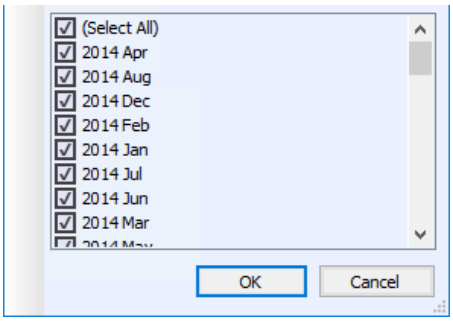


9. Verify that the table looks like the following.



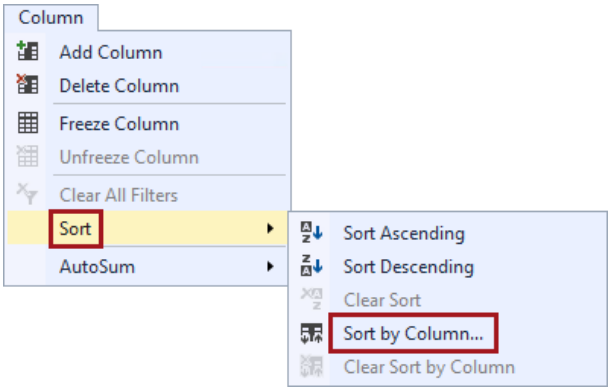
10. To configure the **Date** table in Data View, right-click the table header, and then select **Go to**.

11. To view the **Month** column values, in the column header, click the down-arrow, and then review the distinct column values, available for filtering, found in the column.

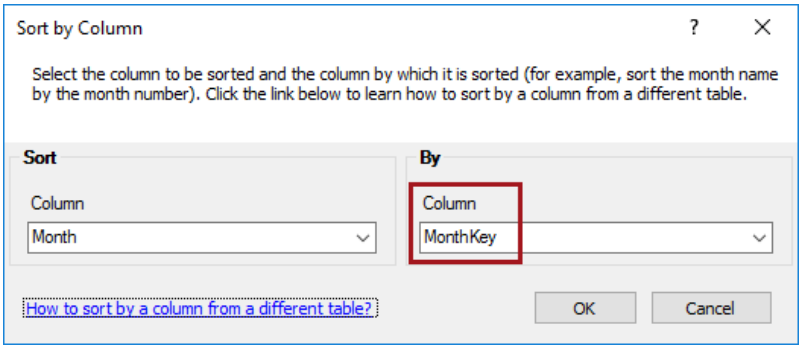


12. Notice that the months are sorted alphabetically, and then click **Cancel**.

13. To configure the months to sort chronologically, select the **Month** column header, and then on the **Column** menu, select **Sort | Sort by Column**.



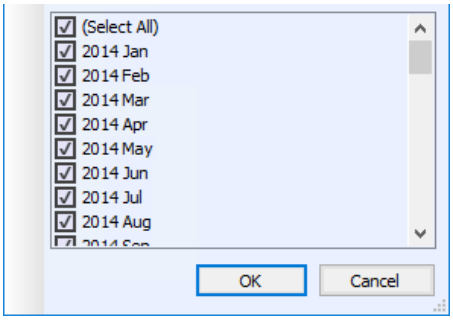
14. In the **Sort by Column** window, in the second dropdown list, select the **MonthKey** column.



*The **MonthKey** column values stored a number which is the year multiplied by 100 with the month number of year added (e.g., March 2017 is 201703).*

15. Click **OK**.

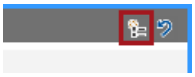
16. In the **Month** column, review the distinct values found in the column, and notice that they are now sorted chronologically.



17. Configure the **Day** column to sort by the **DateKey** column.

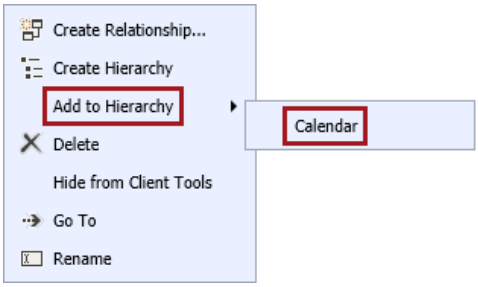
18. Switch to Diagram View, and maximize the **Date** table.

19. To create a hierarchy, at the top right corner, click **Create Hierarchy**.



20. When the hierarchy is added to the table, replace the default name with **Calendar**, and then press **Enter**.

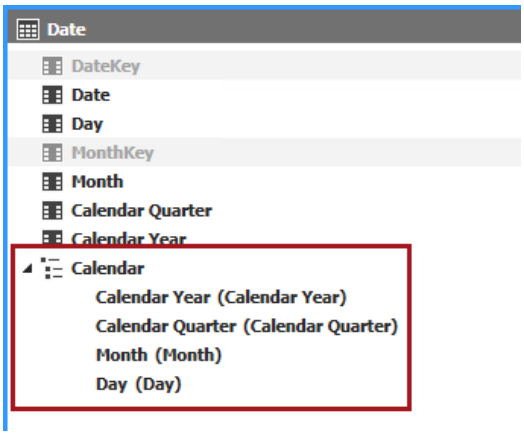
21. To add the **Year** column as the first level of the hierarchy, right-click the **Calendar Year** column, and then select **Add to Hierarchy | Calendar**.



Tip: It is also possible to drag and drop columns into the hierarchy.

22. Add also the **Calendar Quarter**, **Month** and **Day** columns to the hierarchy.

23. Verify that the **Calendar** hierarchy looks like the following.



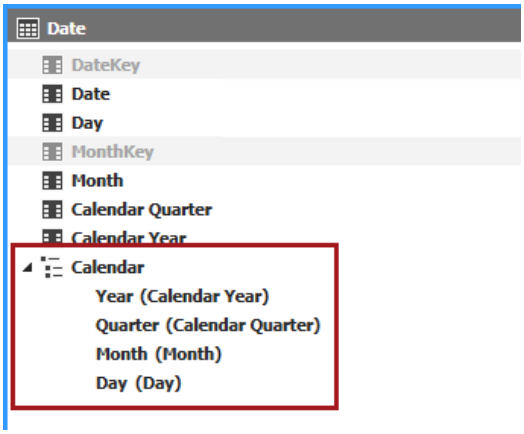
24. To rename the **Calendar Year** level, right-click the level, and then select **Rename**.

    Tip: You can also double-click the level to rename it.

25. Replace the text with **Year**, and then press **Enter**.

26. Rename also the **Calendar Quarter** level as **Quarter**.

27. Verify that the **Calendar** hierarchy looks like the following.

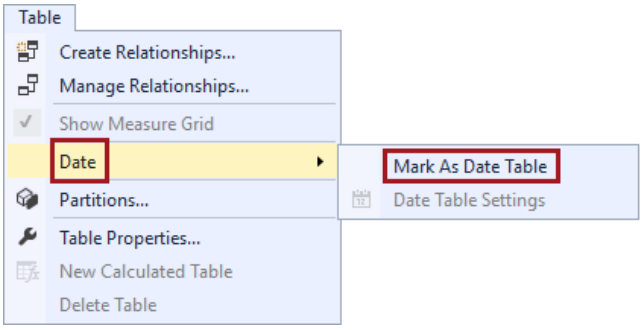


28. Restore the **Date** table to normal size.

29. If necessary, resize the table to reveal all columns and the hierarchy.

30. To mark the **Date** table as a date table, ensure the table is selected.

31. On the **Table** menu, select **Date | Mark as Date Table**.



32. In the **Mark as Date Table** window, in the **Date** dropdown list, notice that the **Date** column is selected.

    Marking a date table will help client applications understand how time is defined in the data model. The Excel PivotTable Fields pane is one such example that interrogates the data model for a date table, and it will surface appropriate time-based filter options based on a marked date table. Also, DAX Time Intelligence formulas require a marked date table to function correctly.

33. Click **OK**.

Creating the Ship Date Table

In this task, you will create the Ship Date table, and configure relationships.

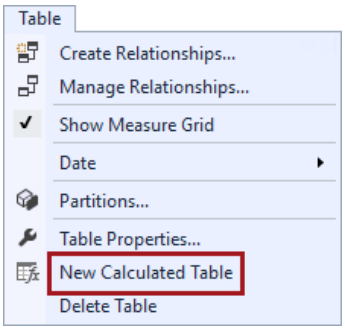
1. To configure the **Date** table in Data View, right-click the table header, and then select **Go to**.

Lab Check
Lab 03 ► Enhancing the Tabular Model

What is the latest **Date** stored in the **Date** table? _____

Commented [PM1]: December 31, 2017

2. On the **Table** menu, select **New Calculated Table**.



3. In the formula bar, after the equals sign (=), enter the following DAX reference to the **Date** table.

DAX

```
='Date'
```

4. Press **Enter**.

   *The calculated table is a copy of the **Date** table. When the **Date** table is refreshed with updated data, the calculated table will also refresh. It does not, however, copy the enhancements configured in the previous task.*

5. To rename the calculated table, right-click the table tab, and then select **Rename**.

6. Rename the table as **Ship Date**, and then press **Enter**.

7. Switch to Diagram View.

8. Delete the inactive relationship between the **Sales** and **Date** tables.

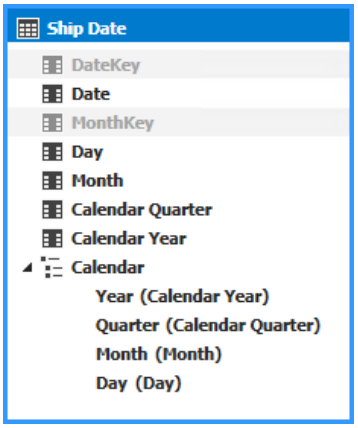
9. Create a relationship between the **Sales** and **Ship Date** tables between the **DateKey** columns.

   Having independent date tables delivers several advantages: There is no need to define explicit calculation logic to navigate the inactive relationship, and users can apply filters to both date tables simultaneously.

10. Configure the enhancements made to the **Date** table to the **Ship Date** table.

    - Hide the **DateKey** and **MonthKey** columns
    - Sort the **Month** column by the **MonthKey** column
    - Sort the **Day** column by the **DateKey** column
    - Create a **Calendar** hierarchy, consisting of **Year**, **Quarter**, **Month** and **Day** levels
    - Mark the **Ship Date** table as a date table

11. Verify that the **Ship Date** table looks like the following.



Enhancing the Region Table

In this task, you will enhance the **Region** table.

1. Rename the following columns.

Column	New Column Name
SalesTerritoryRegion	Region
SalesTerritoryCountry	Country
SalesTerritoryGroup	Group

2. Hide the **SalesTerritoryKey** column.

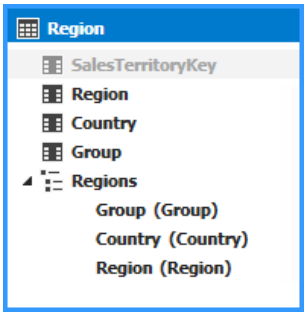
3. To create a hierarchy, to multi-select columns, first select the **Region** column, and then while pressing the **Shift** key, select the **Group** column.

   When using the multi-select method to create a hierarchy, the levels will be ordered based on column cardinality (the column with fewer unique members will be the higher level in the hierarchy). This is to be interpreted to be a "suggested" order of levels only since this may not necessarily be the correct order.

4. Right-click the selection, and then select **Create Hierarchy**.

5. Set the hierarchy name as **Regions**.

6. Verify that the **Region** table looks like the following.



Enhancing the Reseller Table

In this task, you will enhance the **Reseller** table.

1. In the **Reseller** table, rename the following columns.

Column	New Column Name
BusinessType	Business Type
ResellerName	Reseller
EnglishCountryRegionName	Country
StateProvinceName	State

2. Hide the **ResellerKey** column.

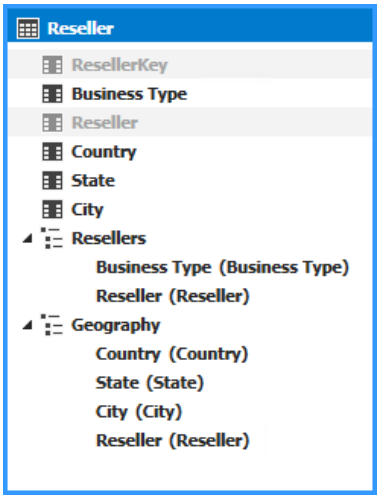
3. Create a hierarchy named **Resellers**, with **Business Type** and **Reseller** levels.

4. Create a second hierarchy named **Geography**, with **Country**, **State**, **City** and **Reseller** levels.

5. Hide the **Reseller** column.

   *Now that hierarchies have been defined, it can make good sense to hide the **Reseller** column which can contain many unique values. By hiding this column, there is reduced potential for it to be added unfiltered to a report. Now, users will need to navigate through the levels of a hierarchy and will arrive at a subset of resellers.*

6. Verify that the **Reseller** table looks like the following.



Enhancing the Salesperson Table

In this task, you will enhance the **Salesperson** table.

1. Go to the Data View for the **Salesperson** table.

Lab Check
Lab 03 ► Enhancing the Tabular Model

How many salespeople are stored in the **Salesperson** table? _____

Commented [PM2]: 18

2. To create a column, in the **Add Column** column, select any cell.

3. In the formula bar, enter the following formula.

   To inject the column references into the expression, when you are ready to enter the column name, simply click anywhere inside the column.

DAX

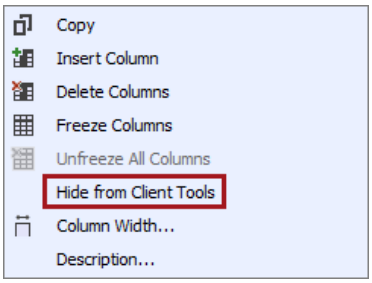
```
=[FirstName] & " " & [LastName]
```

4. Press **Enter**.

5. Rename the new column as **Salesperson**.

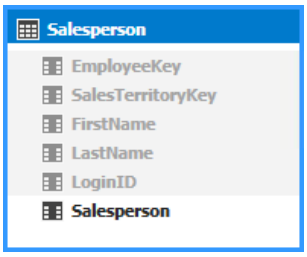
6. To hide a range of columns, first select the **EmployeeKey** column, and then while pressing the **Shift** key, select the **LoginID** column.

7. Right-click the selected columns, and then select **Hide from Client Tools**.



8. Switch to Diagram View.

9. Verify that the **Salesperson** table looks like the following.



Enhancing the Product Table

In this task, you will enhance the **Produce** table with calculated columns to introduce the related **Subcategory** and **Category** columns. You will then create the **Products** hierarchy.

1. Go to the Data View for the **Product** table.

2. Rename the **EnglishProductName** column as **Product**.

3. Hide the **ProductKey** and **ProductSubcategoryKey** columns.

4. Create a calculated column named **Subcategory** with the following formula.

   *Tip: For your convenience, you can copy all formulas in this lab from the **F:\Labs\Lab03\Assets\Snippets.txt** file.*

   DAX

   ```
   =RELATED(Subcategory[EnglishProductSubcategoryName])
   ```

   *This expression navigates the relationship to the **Subcategory** table to retrieve the EnglishProductSubcategoryName column value.*

5. Create a calculated column named **Category** with the following formula.

   DAX

   ```
   =RELATED(Category[EnglishProductCategoryName])
   ```

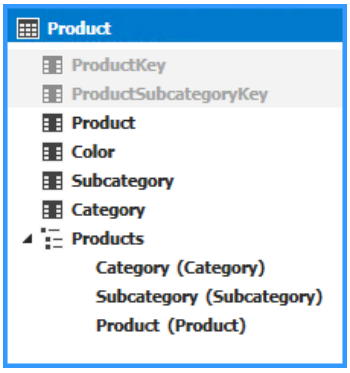
   *This expression navigates two relationships, first to the **Subcategory** table, then the **Category** table, to lookup the **EnglishProductCategoryName** column value.*

6. Switch to Diagram View.

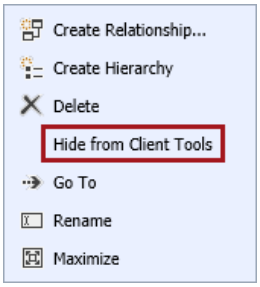
7. Create a hierarchy named **Products**, with **Category**, **Subcategory** and **Product** levels.

   A hierarchy can only add levels based on columns in the same table. This is the reason why you added calculated columns to introduce the related subcategory and category values.

8. Verify that the **Product** table looks like the following.

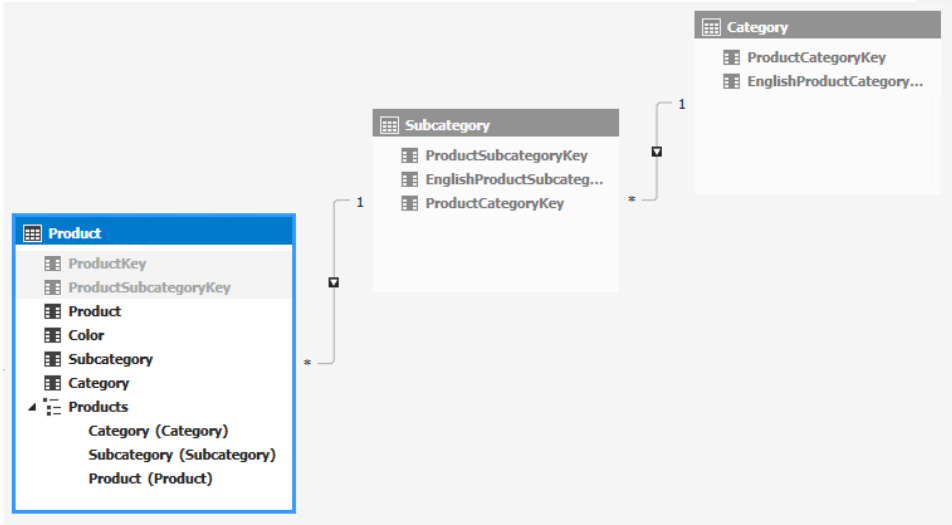


9. To hide the **Subcategory** table, right-click the table header, and then select **Hide from Client Tools**.



10. Hide also the **Category** table.

11. Verify that the three product tables look like the following.



12. To save the project, on the **File** menu, select **Save All**.

All dimension-type tables (i.e. tables which define business entities), have now been enhanced. In the next exercise, you will enhance fact-type tables which typically consist only of measure (i.e. aggregation logic).

Exercise 2: Adding Aggregation Logic

In this exercise, you will add measures to the **Sales** and **Target** tables.

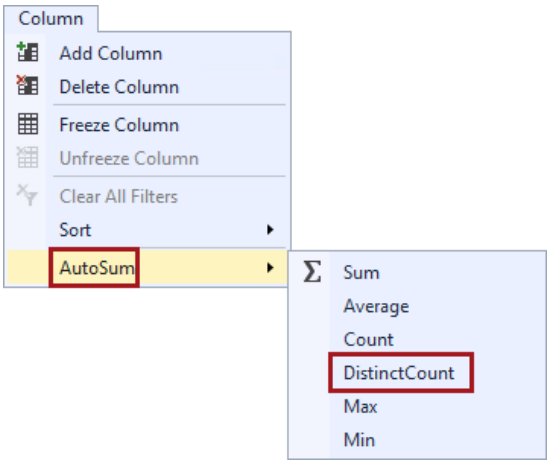
Creating the Sales Table Measures

In this task, you will create measures in the **Sales** table.

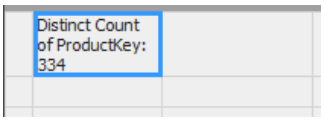
1. Go to the Data View for the **Sales** table.

2. Select the **ProductKey** column header.

3. On the **Column** menu, and then select **AutoSum | DistinctCount**.



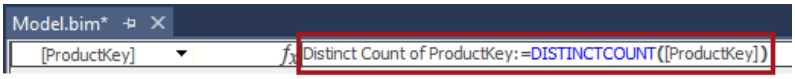
4. In the Measure Grid (located at the bottom of the table grid), notice the addition of the **Distinct Count of ProductKey** measure.



*Tip: On the **Table** menu, you can configure to show or hide the Measure Grid for each table.*

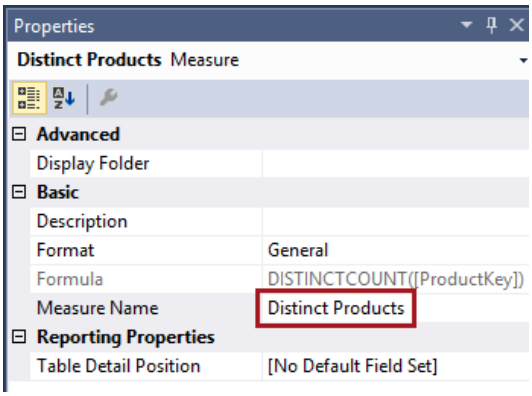
When adding a measure in this way it will be placed in the grid below the column it is based on. Note that the location of the measure within the Measure Grid does not matter. The column used to define the measure, or the sequence of measures within a column, does not impact on how it is evaluated, and you can move a measure to any location of the grid without impacting the formula.

5. In the formula bar, notice the expression that defines the measure, and notice also that the measure name followed by a colon (:) precedes the expression.

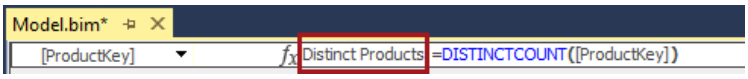


6. In the Measure Grid, select the **Distinct Count of ProductKey** measure.

7. In the **Properties** window (located at the bottom-right), modify the **Measure Name** property to **Distinct Products**.



8. In the formula bar, notice the updated name that precedes the expression.



You can choose to modify the measure name in either location.

Note that measure names must be unique within the model. Also, it is not possible to have a measure with the same name as a column.

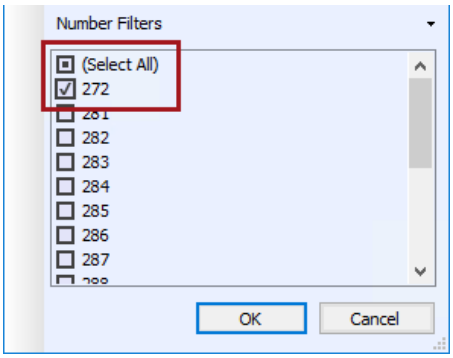
9. In the **Properties** window, modify the following properties of the **Distinct Products** measure.

Property	Value
Format	Whole Number
Show Thousand Separator	True

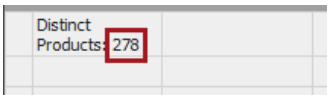
10. In the Measure Grid, notice that the measure displays the value **334**.

This represents the distinct number of products shown for all sales.

11. Filter the **EmployeeKey** column to filter on the value **272**.

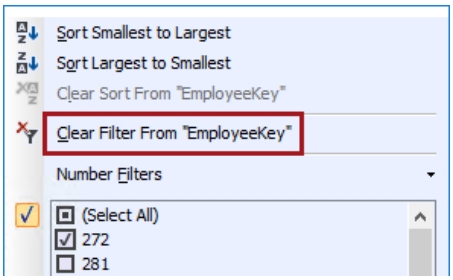


12. In the Measure Grid, notice that the value of the measure has changed to **278**.



The table filters can help test the measure expressions, however filters applied to other tables in the model designer will not be considered. The true test of a measure is in a tool like an Excel PivotTable where filter context can be set by using fields from different tables.

13. Clear the table filter.



14. To add measures based on the **OrderQuantity**, **TotalProductCost** and **SalesAmount** columns, multi-select the columns, and then on the **Column** menu, select **AutoSum | Sum**.

15. In the Measure Grid, select the **Sum of OrderQuantity** measure.

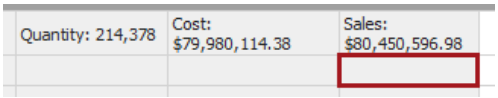
16. In the **Properties** window, modify the following properties.

Property	Value
Measure Name	Quantity
Format	Whole Number
Show Thousand Separator	True

17. Rename the following two measures.

Measure	New Measure Name
Sum of TotalProductCost	Cost
Sum of SalesAmount	Sales

18. To add a measure based on an expression, in the Measure Grid, select the cell beneath the **Sales** measure.



19. In the formula bar, enter the following expression.

For convenience, the measure expressions defined in this lab can be copied from the snippets tile.

DAX

```
Profit:=[Sales] - [Cost]
```

20. In the **Properties** window, set the **Format** property to **Currency**.

21. Add the following measure beneath the last.

DAX

```
Profit %:=DIVIDE([Profit], [Sales])
```

The DIVIDE function divides two expressions, if the second argument results in a non-zero number. If the second argument results in zero or blank (missing), then the function will return blank.

22. Format the **Profit %** measure as **Percentage**.

23. Add the following measure beneath the last.

DAX

```
Sales YTD:=TOTALYTD([Sales], 'Date'[Date])
```

*The TOTALYTD function is used to calculate year-to-date sales values based on the dates in the related **Date** table. The result of (blank) is displayed because a value cannot be displayed for this measure until it is filtered by the **Date** table.*

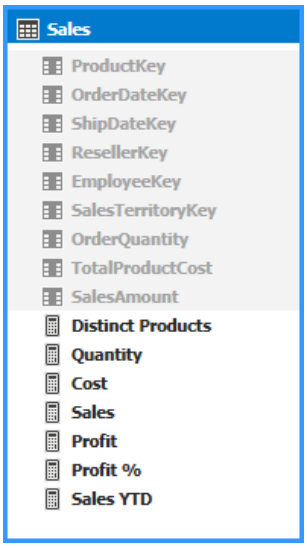
24. Format the **Sales YTD** measure as **Currency**.

25. Hide all **Sales** table columns.

When enhancing the design of a fact-type table, it is common to hide the dimension keys and measure columns, and then define explicit measures as you have just done.

26. Switch to Diagram View.

27. Verify that the **Sales** table looks like the following.



Creating the Target Table Measures

In this task, you will create measures in the **Target** table.

1. Go to the Data View for the **Target** table.

2. In the Measure Grid, in any cell, add the following measure.

DAX

```
Target:=
IF(
      ISFILTERED('Date'[Month]) || ISFILTERED('Date'[Day]),
      CALCULATE(
            SUM([SalesAmountQuota]),
            ALL('Date'[Month]),
            ALL('Date'[Day])
      )
            * DIVIDE(
                  COUNTROWS(VALUES('Date'[Day])),
                  CALCULATE(COUNTA('Date'[Day]), ALL('Date'[Month]), ALL('Date'[Day]))
            ),
      SUM([SalesAmountQuota])
)
```

*This complex formula addresses the fact that target values are stored for salespeople at quarter level. As the **Target** table is related to the **Date** table, it is possible that users could analyze targets at month or day level. It will respond by calculating a proportion of the quarter target based on the number of days in context divided by the number of days in the quarter.*

The ISFILTERED function determines whether the measure is being evaluated at month or day level, and if so, it uses the CALCULATE function to determine the quarter target value, which it then multiplies by the number of days in context (which is achieved with the VALUES function) and then divides by the number of days in the quarter that the month or day belongs.

3. Format the measure as **Currency**.

4. Add the following measure beneath the last.

DAX

```
Variance:=[Sales] - [Target]
```

5. Format the **Variance** measure as **Currency**.

6. Add the following measure beneath the last.

DAX

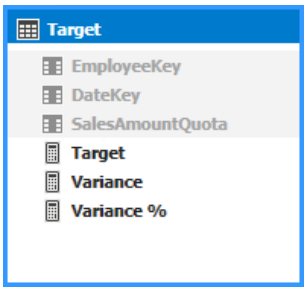
```
Variance %:=DIVIDE([Variance], [Target])
```

7. Format the **Variance** measure as **Percentage**.

8. Hide all **Target** table columns.

9. Switch to Diagram View.

10. Verify that the **Target** table looks like the following.



Creating the Sales Performance KPI

In this task, you will create a KPI in the **Target** table.

1. In the **Target** table, create the following measure beneath the last.

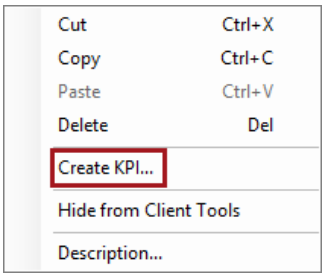
DAX

```
Sales Performance:=[Sales]
```

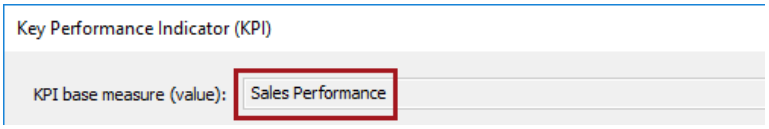
*This measure is a direct reference to the **Sales** measure. It will become the base measure used to create a KPI.*

2. Format the **Sales Performance** measure as **Currency**.

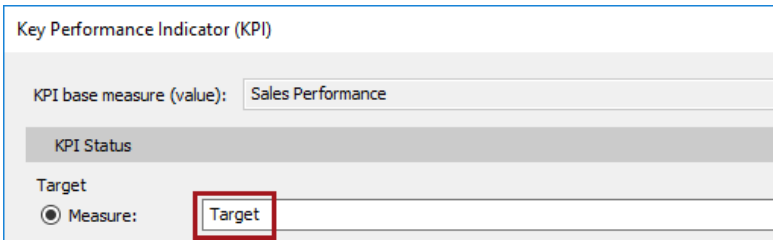
3. Right-click the **Sales Performance** measure, and then select **Create KPI**.



4. In the **Key Performance Indicator (KPI)** window, notice that the KPI base measure (value) is based on the **Sales Performance** measure.



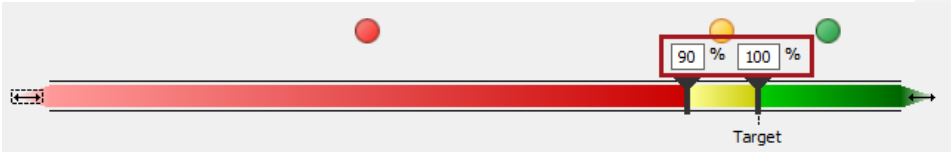
5. In the **KPI Status** section, in the **Measure** dropdown list, select **Target**.



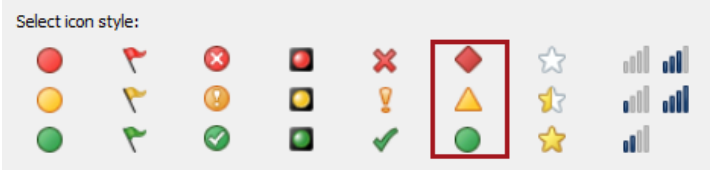
*This configuration will produce a status based on the ratio of **Sales** over **Target**.*

A value of 100% or more means that the sales are on, or exceeding, target. A value less than 0% means that sales are not meeting target.

6. Set the threshold box values to **90** and **100**.



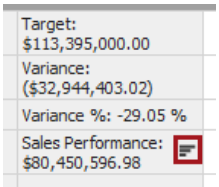
7. In the icon style gallery, select the sixth from the left.



Tip: The combination of shape and color is helpful for users with visual impairment.

8. Click **OK**.

9. In the Measure Grid, notice the icon added to the **Sales Performance** measure to denote that it has been configured as a KPI.

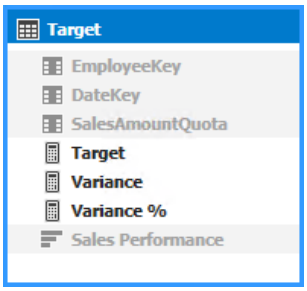


10. Hide the **Sales Performance** measure.

    *Hiding the measure will ensure that the table does not surface the measure (it is, after all, a reference to the **Sales** measure). The model will still surface the KPI.*

11. Switch to Diagram View.

12. Verify that the **Target** table looks like the following.



13. To save the project, on the **File** menu, select **Save All**.

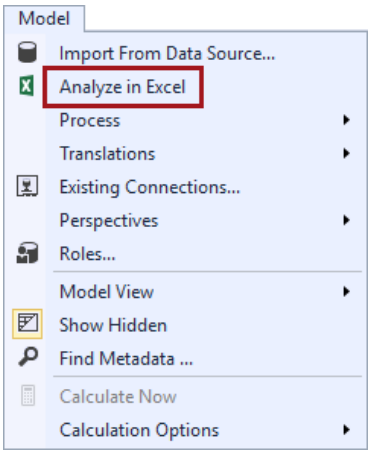
Exercise 3: Exploring the Model Interface

In this exercise, you will explore the model interface with Excel.

Exploring the Model Interface

In this task, you will explore the model interface with Excel.

1. To analyze in Excel, on the **Model** menu, select **Analyze in Excel**.



2. In the **Analyze in Excel** window, click **OK**.

3. In Excel, if prompted to activate Office, click **Cancel**.

4. In the **PivotTable Fields** pane (located at the right), review the fields available in each of the tables.

5. Notice the following:

   - The fields groups consist of "measure groups", KPIs, and tables
   - The "measure groups" are fact-type tables (**Sales** and **Target**) and they contain only measures (because all columns were hidden)
   - The **Sales Performance** KPI is available in the KPIs group, and it consists of a **Value**, **Goal** and **Status** metric
   - The remaining tables represent dimension-type tables, and if the table has one or more hierarchies, then these are listed, with visible columns available in the **More Fields** folder

6. Close Excel, and do not save any changes.

You have now completed the lab. In the next lab, you will manage the tabular model by creating partitions, and roles to enforce row-level security.

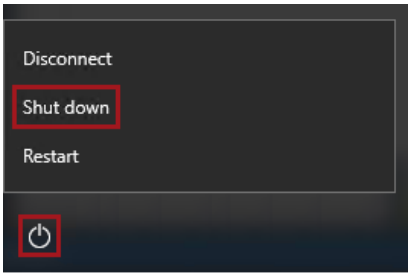
If you are not immediately continuing with the next lab, you should complete the ***Finishing Up*** *exercise to shut down and stop the VM.*

Finishing Up

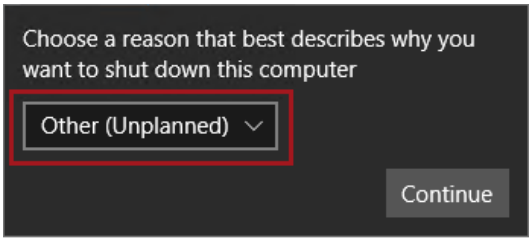
In this exercise, you will shut down and stop the VM.

1. Close all open applications.

2. Press the **Windows** key, and then in the **Start** page, located at the bottom-left, click the **Power** button, and then select **Shut Down**.

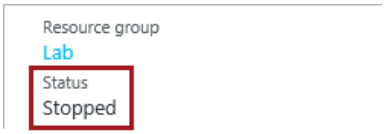


3. When prompted to choose a reason, to accept the default.



4. Click **Continue**.

5. In the **Azure Portal** Web browser page, wait until the status of the VM updates to **Stopped**.

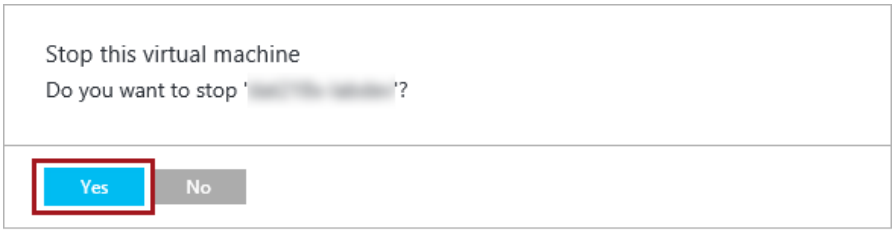


In this state, however, the VM is still billable.

6. Optionally, to deallocate the VM, click **Stop**.

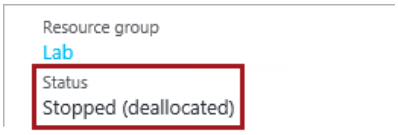
   Deallocation will take some minutes to complete, and also extends the time required to restart the VM. Consider deallocating the VM if you want to reduce costs, or if you choose to complete the next lab after an extended period.

7. When prompted to stop the VM, click **Yes**.



   The deallocation can take several minutes to complete.

8. Verify that the VM status updates to **Stopped (Deallocated)**.



   In this state, the VM is now not billable—except for a relatively smaller storage cost.

   Note that a deallocated VM will likely acquire a different IP address the next time it is started.

9. Sign out of the **Azure Portal**.