TEAM ACADEMY

Introduction to Creating Measures Using DAX in Power BI

Introduction to DAX

TEAM ACADEMY

- Data Analysis Expressions (DAX) is a programming language that is used throughout Microsoft Power BI for creating calculated columns, measures, and custom tables.
- It is a collection of functions, operators, and constants that can be used in a formula, or expression, to calculate and return one or more values.
- You can use DAX to solve a number of calculation and data analysis problems, which can help you create new information from data that is already in your model.

Use Calculated Columns

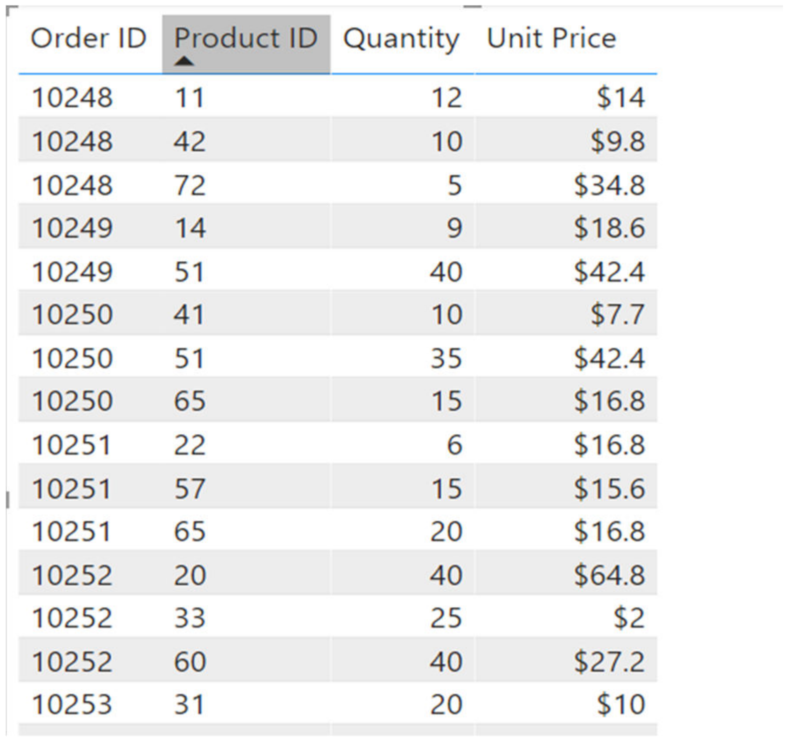
TEAM ACADEMY

- DAX allows you to augment the data that you bring in from different data sources by creating a calculated column that didn't originally exist in the data source.
- This feature should be used sparingly, which will be explained later in this module.
- **Example:** Assume that you are importing data from a database that contains sales transactions.
- Each individual sales transaction has the following columns: **Order ID**, **Product ID**, **Quantity**, and **Unit Price**. Notice that a column doesn't exist for the total sales amount for each order.

Continue …

TEAM ACADEMY

- The following figure shows how the initial shape of the data appears in a Power BI table visual.

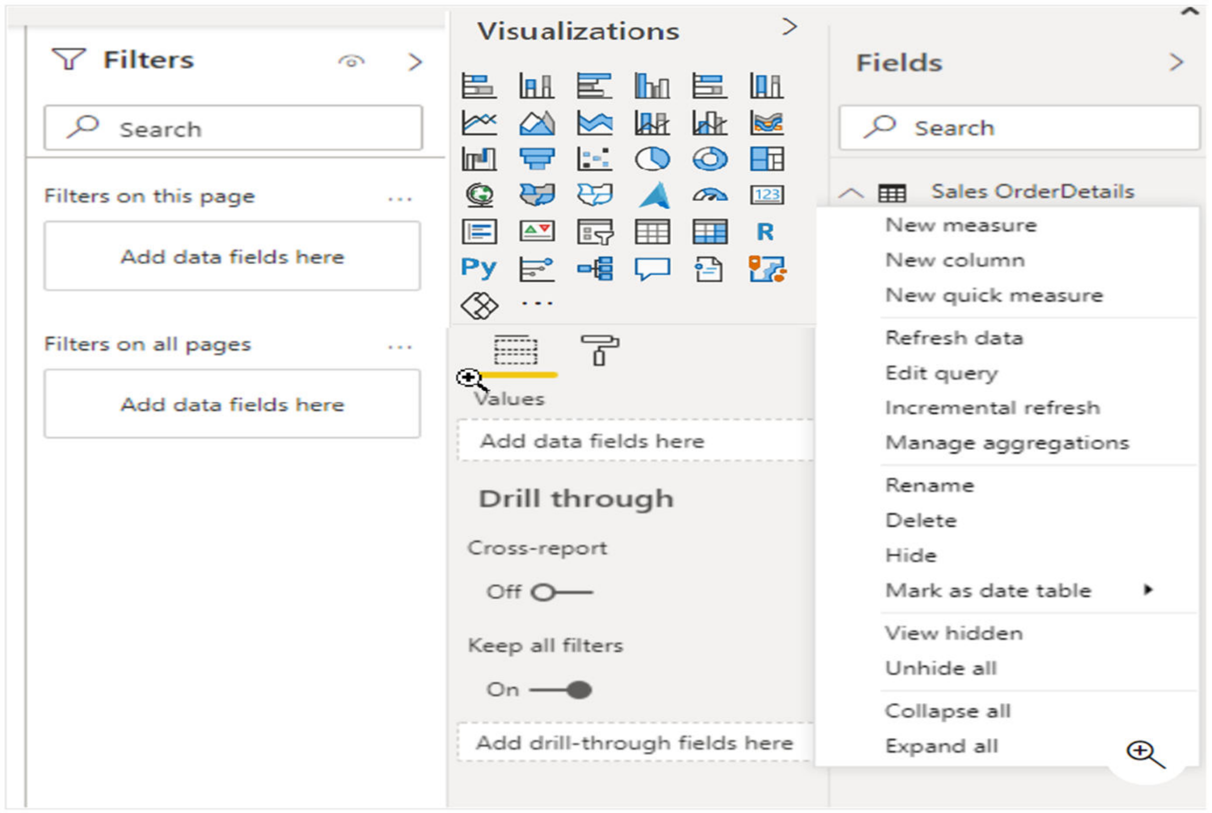


Order ID	Product ID	Quantity	Unit Price
10248	11	12	$14
10248	42	10	$9.8
10248	72	5	$34.8
10249	14	9	$18.6
10249	51	40	$42.4
10250	41	10	$7.7
10250	51	35	$42.4
10250	65	15	$16.8
10251	22	6	$16.8
10251	57	15	$15.6
10251	65	20	$16.8
10252	20	40	$64.8
10252	33	25	$2
10252	60	40	$27.2
10253	31	20	$10

Continue …

TEAM ACADEMY

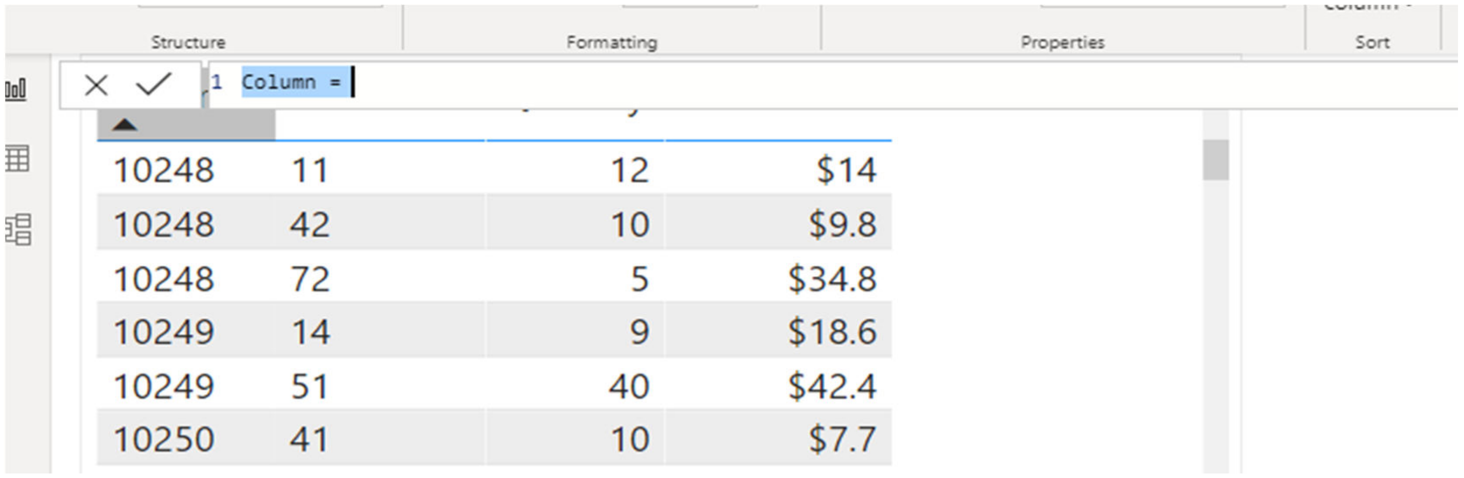
- You can start using DAX by creating a calculated column that multiplies the unit price with the quantity.
- The calculated column will create a value for each row called Total Price.
- Create the new column by selecting the ellipsis (**…**) button on the table in the **Fields** list and then selecting **New column**.



Continue ...

TEAM ACADEMY

- A new DAX formula appears in the formula bar underneath the ribbon at the top.



```
1 Column =
```

10248	11	12	$14
10248	42	10	$9.8
10248	72	5	$34.8
10249	14	9	$18.6
10249	51	40	$42.4
10250	41	10	$7.7

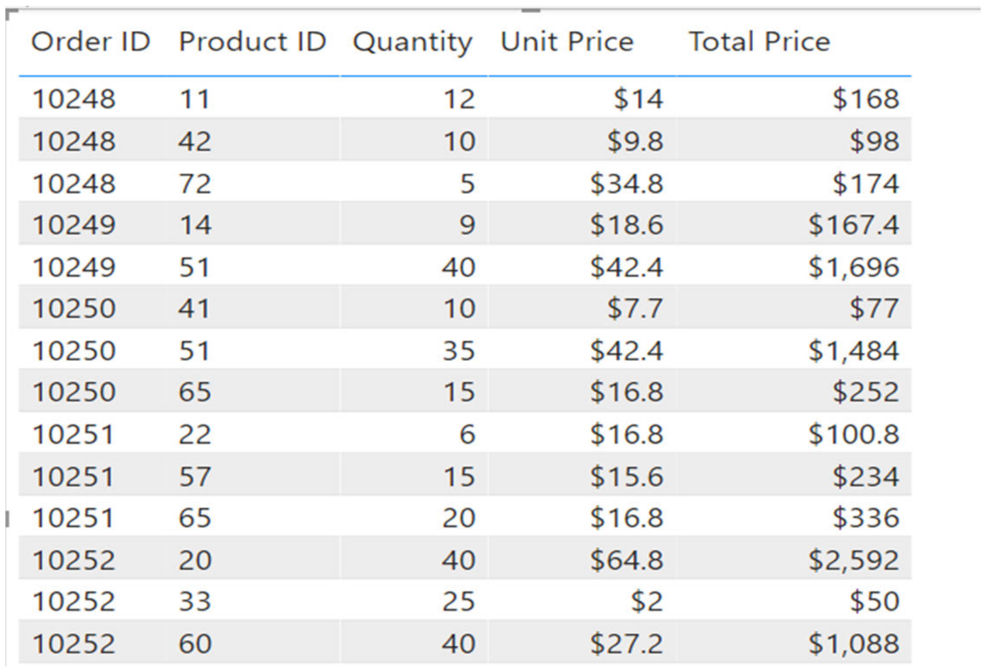
- You can replace the "Column =" default text with the following example text:

```
Total Price = 'Sales OrderDetails'[Quantity] * 'Sales OrderDetails'[Unit Price]
```

Continue …

TEAM ACADEMY

- The value on the left side of the equal sign is the column name.
- The text on the right side of the equal sign is the DAX expression.
- This simple DAX expression takes the quantity value and multiplies it with the unit price value for each individual row.
- It will produce one value for each record in the table. If you drag the new column from the **Fields** list to the visual, you will see the new values.



Order ID	Product ID	Quantity	Unit Price	Total Price
10248	11	12	$14	$168
10248	42	10	$9.8	$98
10248	72	5	$34.8	$174
10249	14	9	$18.6	$167.4
10249	51	40	$42.4	$1,696
10250	41	10	$7.7	$77
10250	51	35	$42.4	$1,484
10250	65	15	$16.8	$252
10251	22	6	$16.8	$100.8
10251	57	15	$15.6	$234
10251	65	20	$16.8	$336
10252	20	40	$64.8	$2,592
10252	33	25	$2	$50
10252	60	40	$27.2	$1,088

Continue …

TEAM ACADEMY

- The previous screenshot shows that DAX is calculating correctly and displaying the results that you wanted.
- Calculated columns are materialized in the .pbix Power BI file extension, meaning that each time you add a calculated column, you are increasing the size of the overall file.
- Having too many calculated columns will slow performance and will cause you to reach the maximum Power BI file size sooner.

Create a custom column

- Three ways to create a custom column in Power BI are:

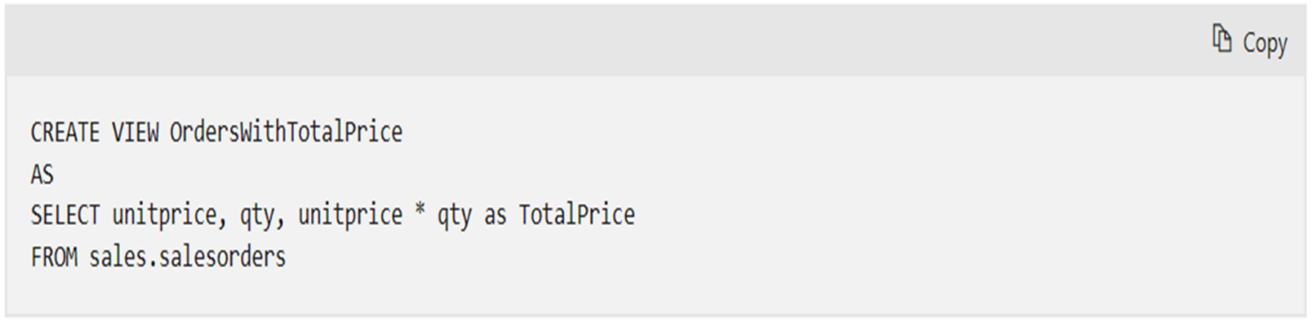
1. Create the column in the source query when you get the data, for instance, by adding the calculation to a view in a relational database.
2. Create a custom column in Power Query.
3. Create a calculated column by using DAX in Power BI desktop.

- You can create a calculated column when you pull the data from the data source.
- Each data source would have a different technique for completing this action. If you were pulling data from a relational data source by using a view that was written in the SQL language, it would look like the following example:

Continue …

TEAM ACADEMY

- If you were pulling data from a relational data source by using a view that was written in the SQL language, it would look like the following example:



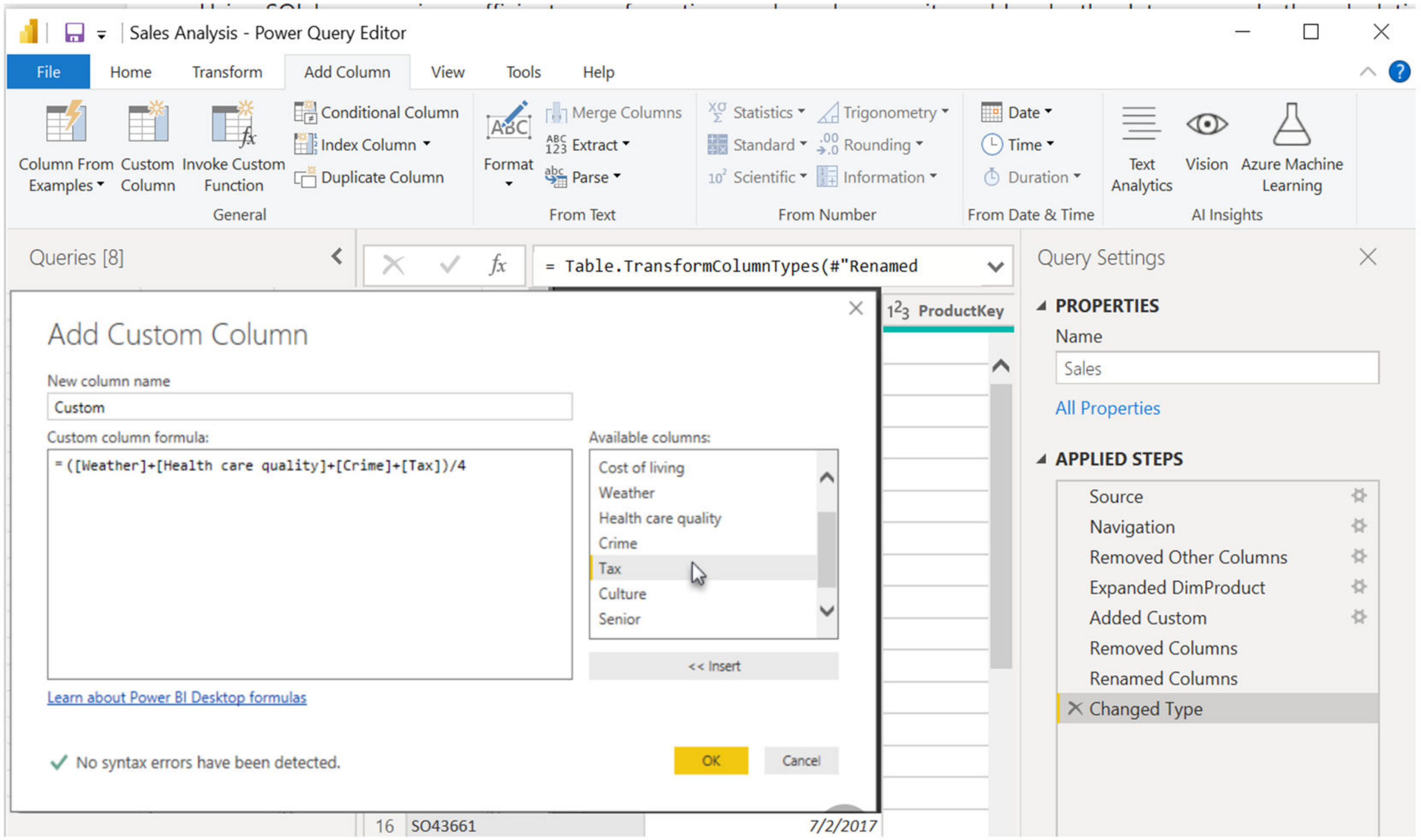
```
CREATE VIEW OrdersWithTotalPrice
AS
SELECT unitprice, qty, unitprice * qty as TotalPrice
FROM sales.salesorders
```

- Using SQL language is an efficient way of creating a column because it would make the data source do the calculations for you.
- In Power BI, the calculated column would appear like any other column.

Continue …

TEAM ACADEMY

- You can also use Power Query to create a custom column.



= Table.TransformColumnTypes(#"Renamed

Add Custom Column

New column name

Custom

Custom column formula:

```
= ([Weather]+[Health care quality]+[Crime]+[Tax])/4
```

Available columns:

Cost of living
Weather
Health care quality
Crime
Tax
Culture
Senior

Learn about Power BI Desktop formulas

✓ No syntax errors have been detected.

16 SO43661 7/2/2017

PROPERTIES

Name

Sales

All Properties

APPLIED STEPS

- Source
- Navigation
- Removed Other Columns
- Expanded DimProduct
- Added Custom
- Removed Columns
- Renamed Columns
- Changed Type

Continue ...

TEAM ACADEMY

- The custom column dialog uses the M language to create the new column.
- M language is out of scope for the purposes of this module.
- When you create a calculated column by using DAX, you do not need to refresh the dataset to see the new column.
- In the other methods, you would need a refresh to see changes.
- This process can be lengthy if you are working with a lot of data.
- The DAX calculated column does not compress as well as the other methods.
- The other column types do get compressed, which makes the .pbix file smaller and the performance is usually faster.

Use Measures

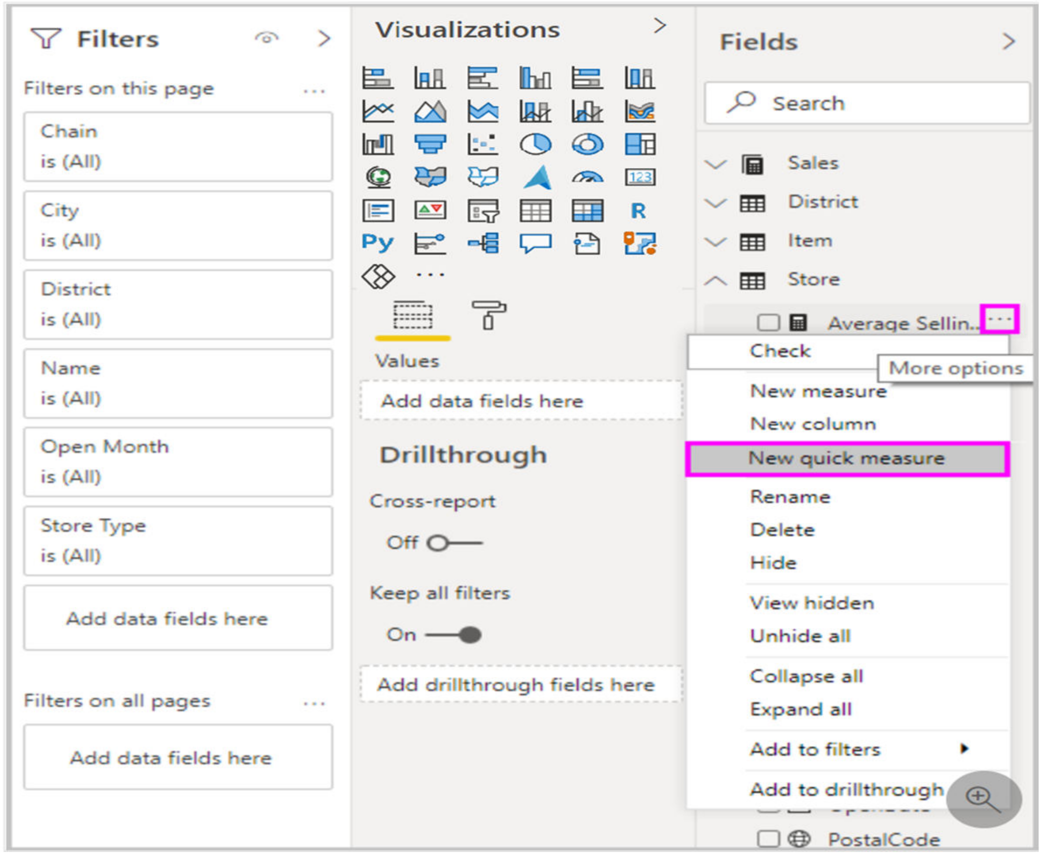
TEAM ACADEMY

- Calculated columns are useful, when you are required to operate row by row.
- Consider a situation where you want an aggregation that operates over the entire dataset and you want the total sales of all rows.
- Furthermore, you want to slice and dice that data by other criteria like total sales by year, by employee, or by product.
- To accomplish those tasks, you would use a measure. You can build a measure without writing DAX code; Power BI will write it for you when you create a quick measure.
- You can see the DAX that's implemented by the quick measure while jumpstarting or expanding your own DAX knowledge.

Create a Quick Measure

TEAM ACADEMY

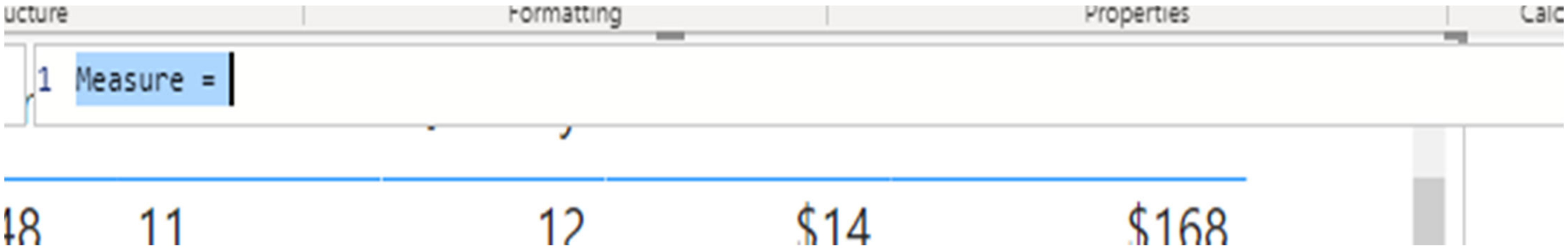
- To create a quick measure in Power BI Desktop, right-click or select the ellipsis (**...**) button next to any item in the **Fields** pane and then select **New quick measure** from the menu that appears.
- The **Quick measures** screen will appear. In the **Quick measures** window, you can select the calculation that you want and the fields to run the calculation against.



Create a measure

TEAM ACADEMY

- Measures are used in some of the most common data analyses.
- To continue with the previous scenario, you want to create a measure that totals your new column for the entire dataset.
- Similar to how you created a calculated column, you can go to the **Fields** list, click the three-dot ellipsis on the selected field and select **New measure**.
- Text will now appear in the formula bar underneath the ribbon.



```
1 Measure =
```

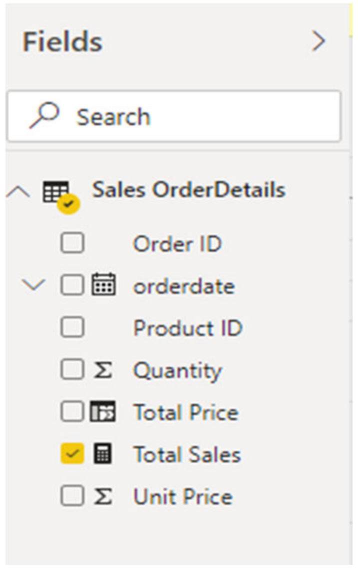
Continue …

TEAM ACADEMY

- You can replace the "Measure =" text with the following text:

```
Total Sales = sum('Sales OrderDetails'[Total Price])
```

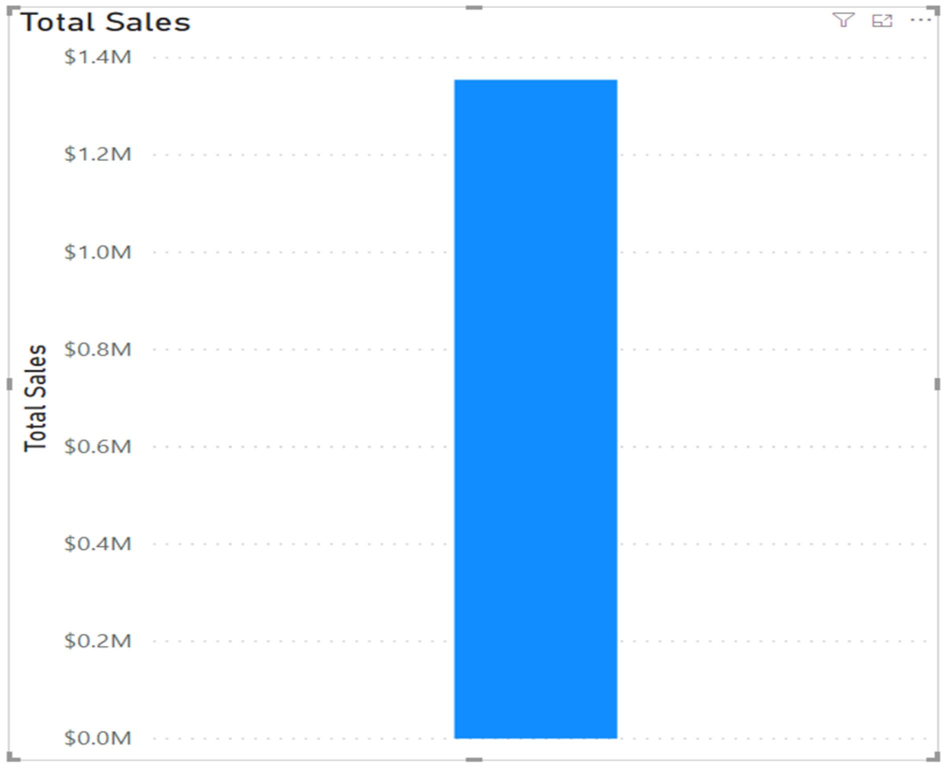
- The new measure will now appear in the **Fields** list.



Continue …

TEAM ACADEMY

- When you drag Total Sales over to the report design surface, you will see the total sales for the entire organization in a column chart.



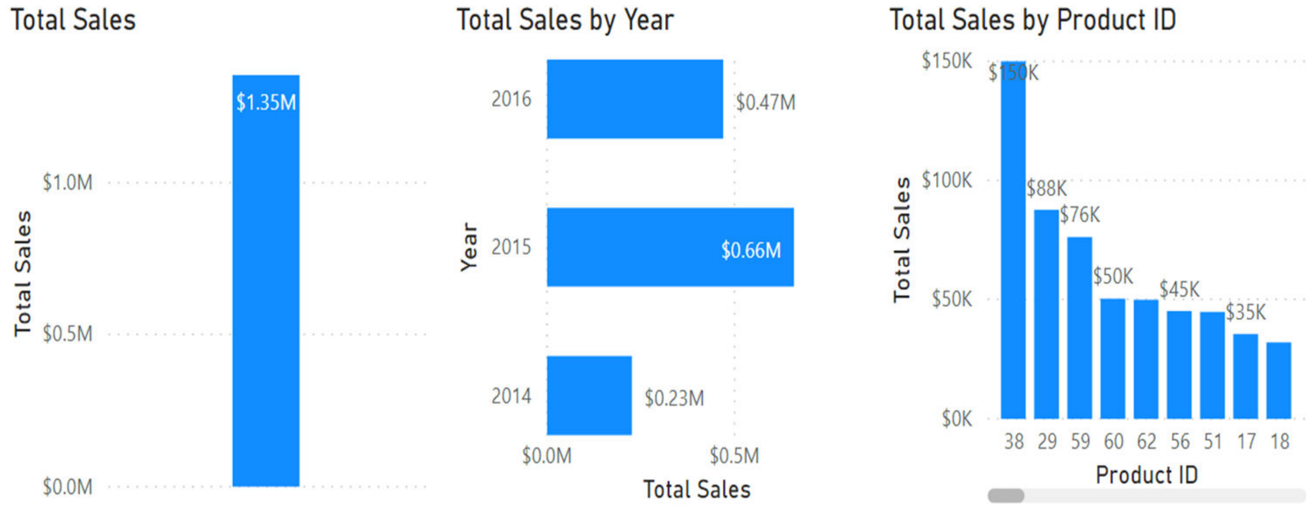
Difference between a calculated column and a measure

- The fundamental difference between a calculated column and a measure is that a calculated column creates a value for each row in a table.
- **Example:** If the table has 1,000 rows, it will have 1,000 values in the calculated column.
- Calculated column values are stored in the Power BI .pbix file. Each calculated column will increase the space that is used in that file and potentially increase the refresh time.
- Measures are calculated on demand. Power BI calculates the correct value when the user requests it.
- Measures do not add to the overall disk space of the Power BI .pbix file.
- Measures are calculated based on the filters that are used by the report user. These filters combine to create the filter context.

Understand Context

TEAM ACADEMY

- How context affects DAX measures is a difficult concept to comprehend.
- The ensuing visuals will demonstrate how context affects DAX measures so you can see how they interact together.
- The following three visuals use the exact same DAX measure: Total Sales.



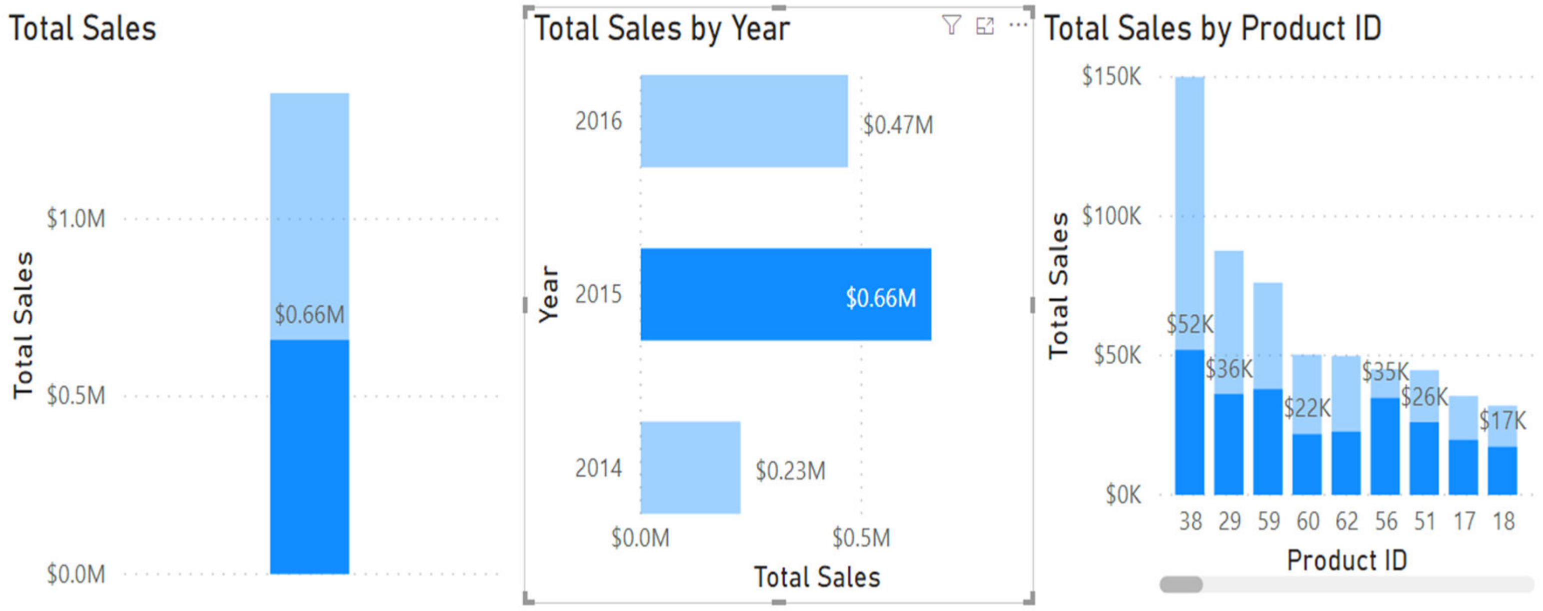
Continue ...

- Though each visual uses the same DAX measure and, therefore, the same DAX formula, the visuals produce different results.
- For instance, the first visual shows the Total Sales measure for the entire dataset.
- In this dataset, Total Sales is USD1.35 million. In the second visual, Total Sales is broken down by year.
- For instance, in 2014, Total Sales is USD0.23 million. In the third visual, Total Sales is broken down by Product ID.
- With Power BI, even though the measure was only defined once, it can be used in these visuals in different ways.
- Each of the totals is accurate and performs quickly.
- It is the context of how the DAX measure is used that calculates these totals accurately.

Continue …

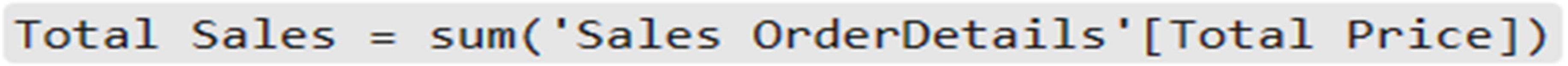
TEAM ACADEMY

- Interactions between visuals will also change how the DAX measure is calculated.
- For instance, if you select the second visual and then select **2015**, the results appear as shown in the following screenshot.



Continue …

- Selecting **2015** in the second visual changed the filter context for the DAX measure.
- It modified the first visual to equal the sales for 2015: USD0.66 million.
- It also broke down the Total Sales By Product ID, but only shows the results for 2015.
- Those calculations quickly changed in memory and displayed the results in a highly interactive manner to the user.
- The definition of the DAX measure has not changed; it's still the original, as shown in the following example:



```
Total Sales = sum('Sales OrderDetails'[Total Price])
```

Use the calculate function

- The CALCULATE function is your method of creating a DAX measure that will override certain portions of the context that are being used to express the correct result.
- For instance, if you want to create a measure that always calculates the total sales for 2015, regardless of which year is selected in any other visual in Power BI, you would create a measure that looks like the following sample:

```
Total Sales for 2015 = CALCULATE(SUM('Sales OrderDetails'[Total Price]), YEAR('Sales OrderDetails'[orderdate]) =
2015)
```

Continue …

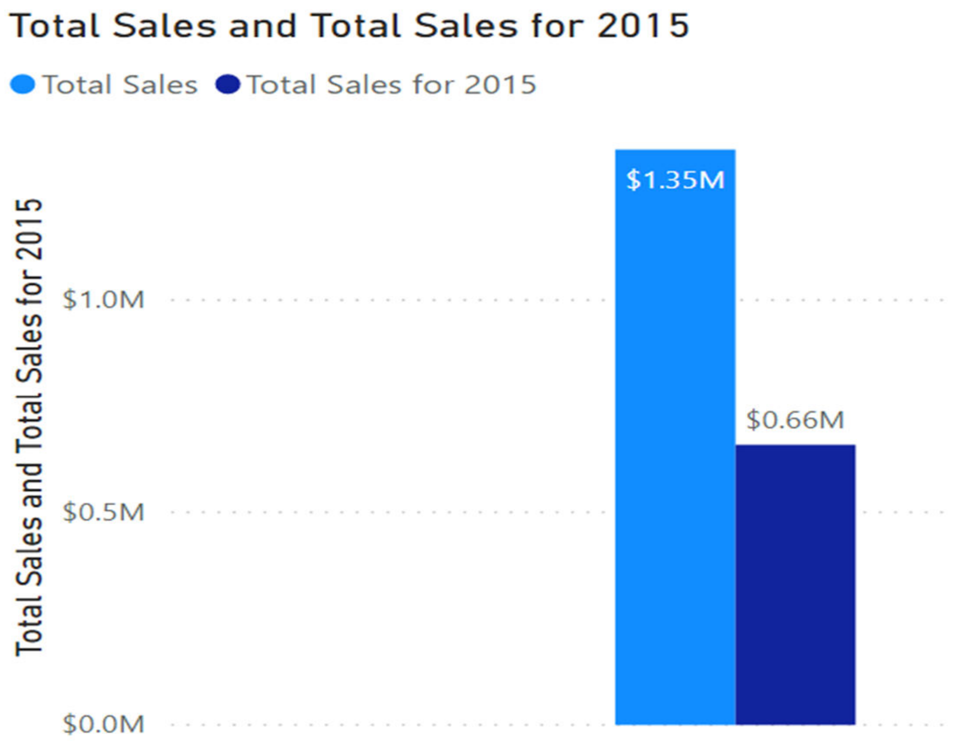
TEAM ACADEMY

- When you use the CALCULATE function to override the context, it is helpful to name the measure in a way that describes exactly how you are overriding it.
- **Example:** CALCULATE is aggregating the Total Price column, just as you did in the previous measure.
- Instead of operating over the entire dataset while using whatever the filter context tells it to do, you are overriding the filter context for the year 2015.
- No matter what year is selected as a filter for other reports, you will always get the total for 2015 using this measure; all other filters still apply.

Continue …

TEAM ACADEMY

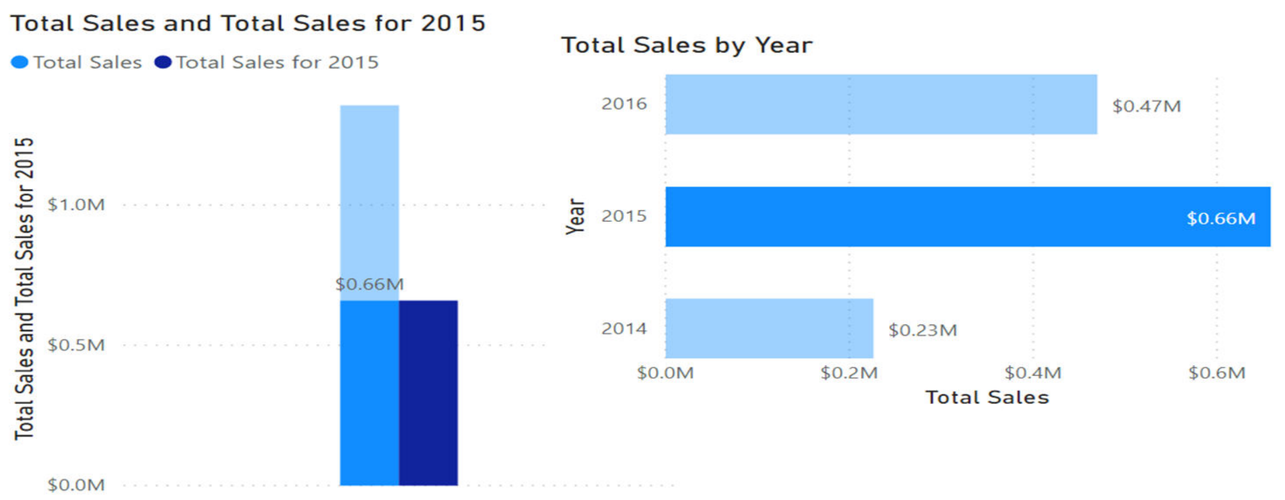
- When both measures are added to the previous visual they will resemble the following screenshot.



Continue …

TEAM ACADEMY

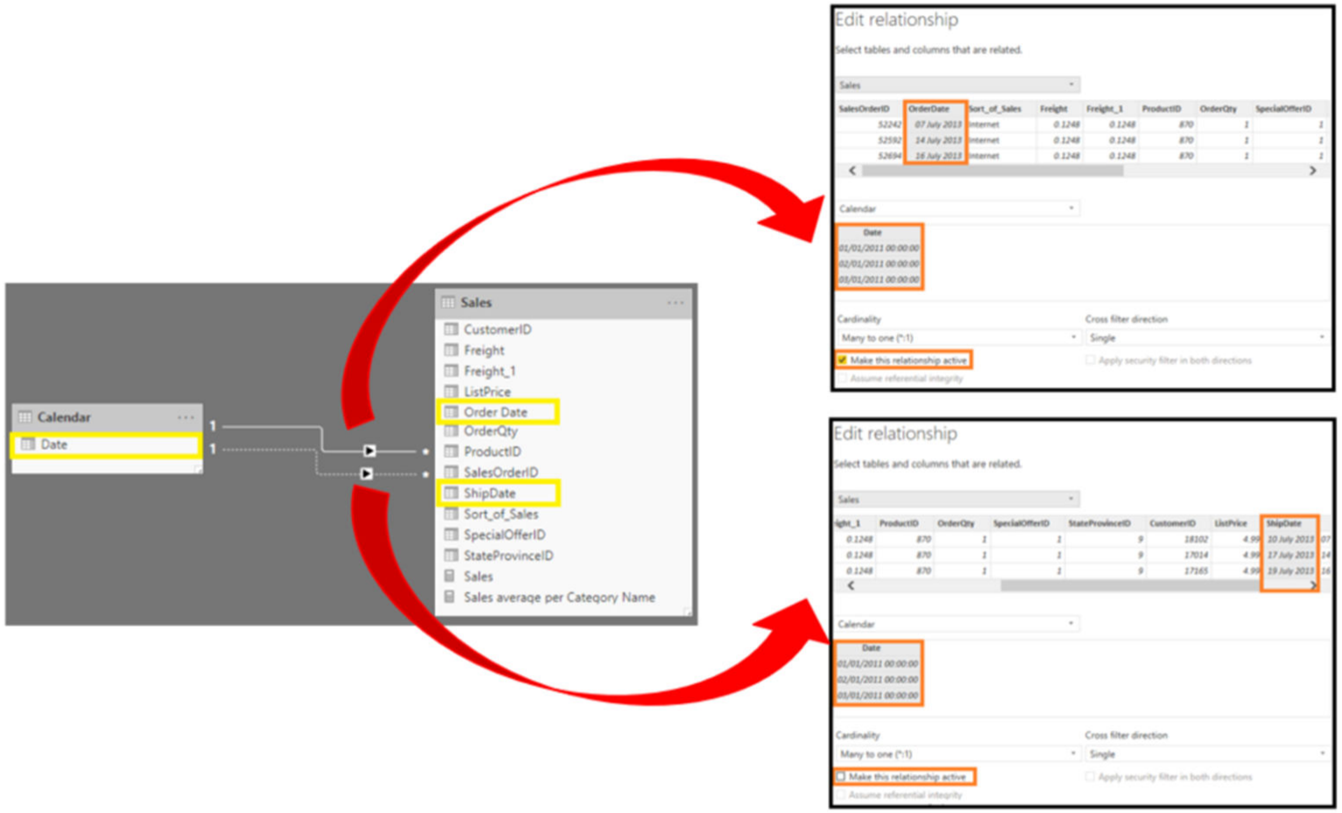
- Total Sales is still USD1.35 million, while the 2015 Total Sales is USD0.66 million.
- When you add the other visual onto the report, as you did previously, and then select 2015, the results will look like the following image.
- If you select 2016, Total Sales for 2015 will remain at USD0.66 million.



Use relationship effectively

TEAM ACADEMY

- Another DAX function that allows you to override the default behavior is USERELATIONSHIP.
- Consider the following data model example.



Edit relationship

Select tables and columns that are related.

Sales

SalesOrderID	OrderDate	Sort_of_Sales	Freight	Freight_1	ProductID	OrderQty	SpecialOfferID
52242	07 July 2013	Internet	0.1248	0.1248	870	1	1
52592	14 July 2013	Internet	0.1248	0.1248	870	1	1
52694	16 July 2013	Internet	0.1248	0.1248	870	1	1

Calendar

Date
01/01/2011 00:00:00
02/01/2011 00:00:00
03/01/2011 00:00:00

Cardinality: Many to one (*:1)

Cross filter direction: Single

[✓ Make this relationship active]

Apply security filter in both directions

Assume referential integrity

Edit relationship

Select tables and columns that are related.

Sales

ight_1	ProductID	OrderQty	SpecialOfferID	StateProvinceID	CustomerID	ListPrice	ShipDate	
0.1248	870	1	1	9	18102	4.99	10 July 2013	07
0.1248	870	1	1	9	17014	4.99	17 July 2013	14
0.1248	870	1	1	9	17165	4.99	19 July 2013	16

Calendar

Date
01/01/2011 00:00:00
02/01/2011 00:00:00
03/01/2011 00:00:00

Cardinality: Many to one (*:1)

Cross filter direction: Single

Make this relationship active

Apply security filter in both directions

Assume referential integrity

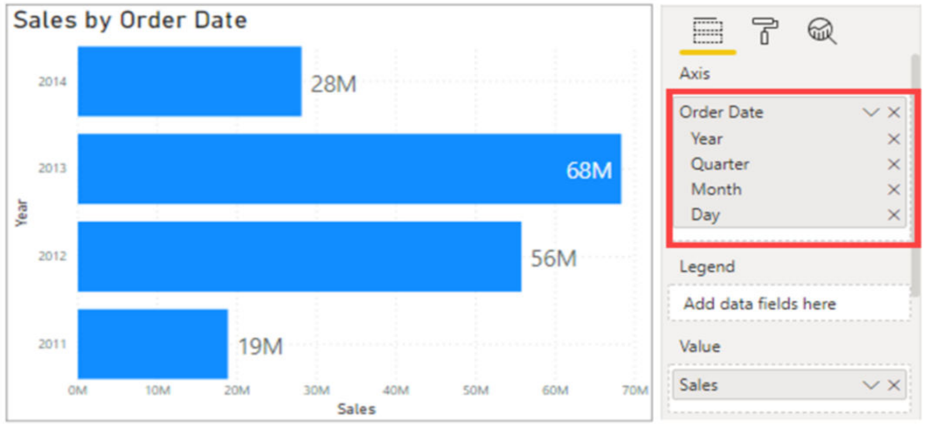
Continue …

TEAM ACADEMY

- The preceding screenshot shows an established relationship between the **Date and OrderDate** columns, as shown by the highlighted line connecting the two.
- The solid line between the two tables indicates that it is the active relationship, meaning that by default, any slicing on the date table where measures in the Sales data are being displayed will be along the **OrderDate** column.
- A dashed relationship exists between the **Date** and **ShipDate** columns, indicating that it is an inactive relationship.
- This relationship will never be used unless explicitly declared in a measure.
- The goal is to build the following report, where you have two visuals: **Sales by Ship Date** and **Sales by Order Date**.

Continue ...

TEAM ACADEMY



Axis

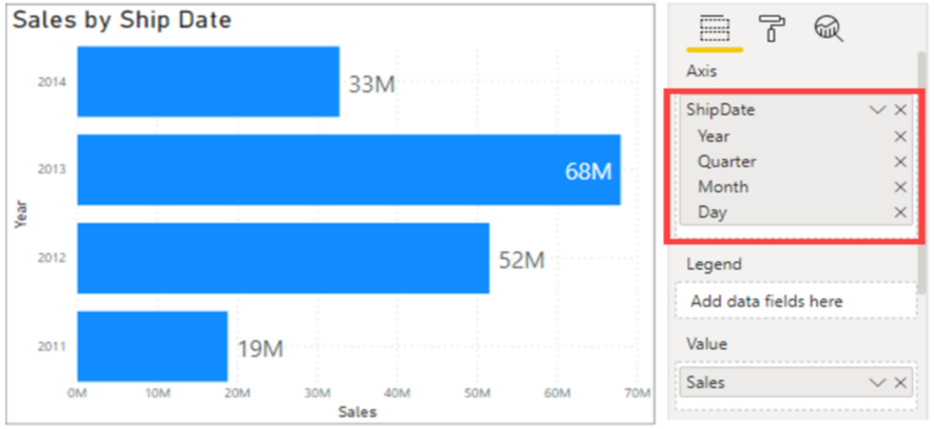
Order Date
Year
Quarter
Month
Day

Legend

Add data fields here

Value

Sales



Axis

ShipDate
Year
Quarter
Month
Day

Legend

Add data fields here

Value

Sales

Continue …

TEAM ACADEMY

- To create this measure for Sales by Ship Date, you can use the DAX function USERELATIONSHIP().
- This function is used to specify a relationship to be used in a specific calculation and is done without overriding any existing relationships.
- It is a beneficial feature in that it allows developers to make additional calculations on inactive relationships by overriding the default active relationship between two tables in a DAX expression

```
Sales by Ship Date = CALCULATE(Sum(Sales[TotalPrice]), USERELATIONSHIP(Sales[ShipDate],'Calendar'[Date]))
```

- Now, you will be able to create the second visual.

Create semi – additive measures

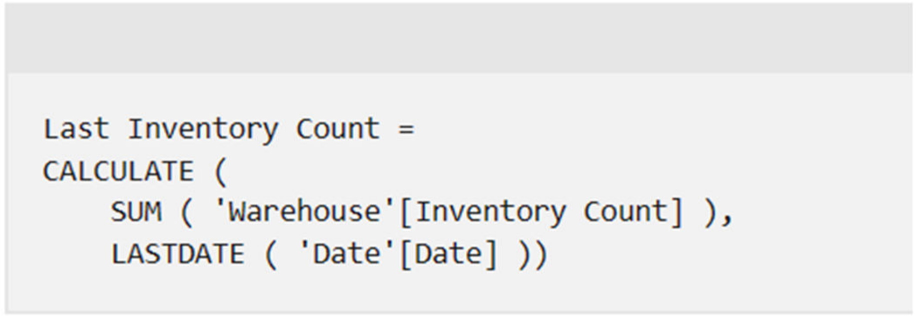
TEAM ACADEMY

- In situations where you don't want the standard evaluation behavior in Power BI, you can use the CALCULATE and/or USERELATIONSHIP functions.
- More circumstances exist where you don't want the standard behavior.
- One of those situations is when you have a semi-additive problem to resolve.
- Standard measures are simple concepts, where they might use the SUM, AVERAGE, MIN, and MAX functions.
- Thus far, you've been using SUM for the **Total Sales** measure.
- **Example**: If on Monday, you have 100 mountain bikes, and on Tuesday you have 125 mountain bikes, you wouldn't want to add those together to indicate that you had 225 mountain bikes between those two days.

Continue ...

TEAM ACADEMY

- In this circumstance, if you want to know your stock levels for March, you would need to tell Power BI not to add the measure but instead take the last value for the month of March and assign it to any visual.



```
Last Inventory Count =
CALCULATE (
    SUM ( 'Warehouse'[Inventory Count] ),
    LASTDATE ( 'Date'[Date] ))
```

- This approach will stop the SUM from crossing all dates. Instead, you will only use the SUM function on the last date of the time period, thus effectively creating a semi-additive measure.

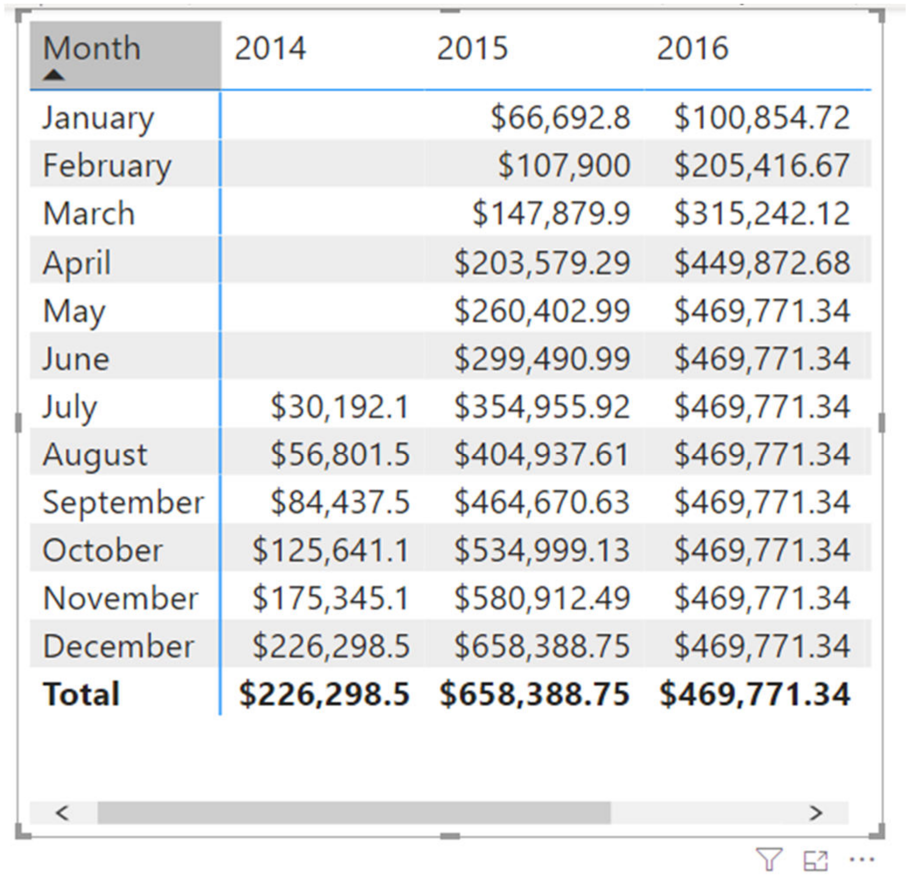
Work with time intelligence

TEAM ACADEMY

- All data analysts will have to deal with time. Dates are important, so we highly recommend that you create or import a dates table.
- This approach will help make date and time calculations much simpler in DAX.
- While some time calculations are simple to do in DAX, others are more difficult.
- For instance, the following screenshot shows what happens if you want to display a running total.

Continue …

TEAM ACADEMY

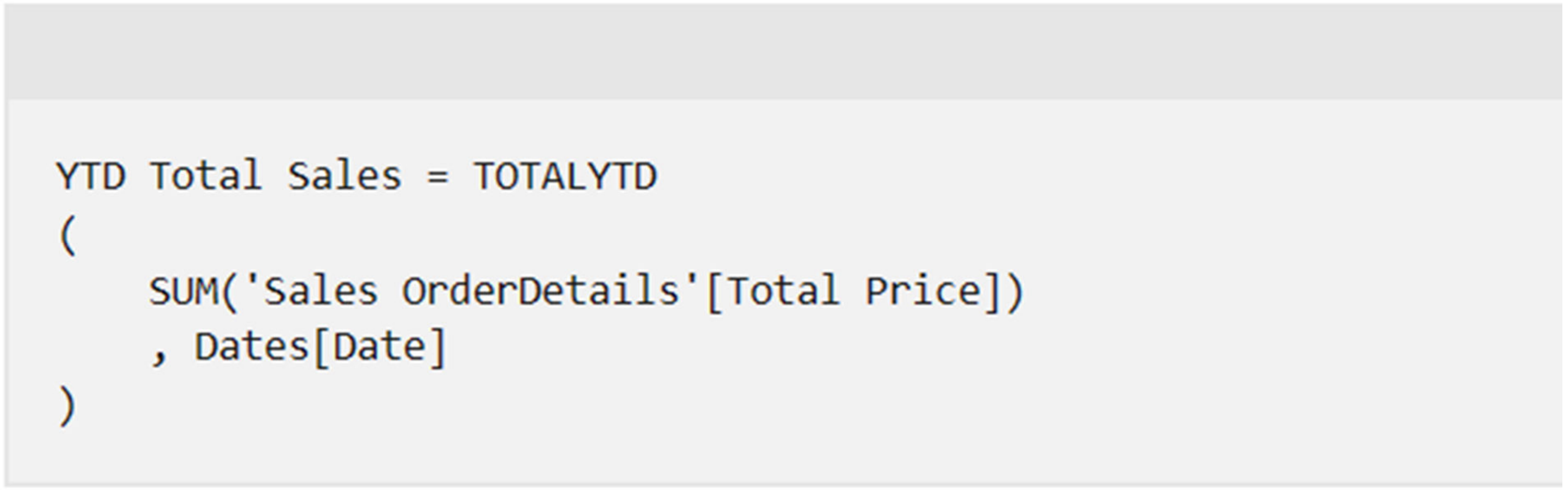


Month	2014	2015	2016
January		$66,692.8	$100,854.72
February		$107,900	$205,416.67
March		$147,879.9	$315,242.12
April		$203,579.29	$449,872.68
May		$260,402.99	$469,771.34
June		$299,490.99	$469,771.34
July	$30,192.1	$354,955.92	$469,771.34
August	$56,801.5	$404,937.61	$469,771.34
September	$84,437.5	$464,670.63	$469,771.34
October	$125,641.1	$534,999.13	$469,771.34
November	$175,345.1	$580,912.49	$469,771.34
December	$226,298.5	$658,388.75	$469,771.34
Total	**$226,298.5**	**$658,388.75**	**$469,771.34**

Continue …

TEAM ACADEMY

- Notice that the totals increment for each month but then reset when the year changes.
- In other programming languages, this result can be fairly complicated, often involving several variables and looping through code.
- DAX makes this process fairly simple, as shown in the following example:



```
YTD Total Sales = TOTALYTD
(
    SUM('Sales OrderDetails'[Total Price])
    , Dates[Date]
)
```

Continue ...

TEAM ACADEMY

- The **YTD Total Sales** measure uses a built-in DAX function called TOTALYTD.
- This function takes an argument for the type of calculation. You can use the SUM function to get the Total Price, as you've done throughout this module.
- The second argument that you want to operate over is the **Dates** field.
- You can use your Dates table and add this measure to your visual, and you'll get the running total result that you're looking for.
- You can use all functions with YTD, MTD, and QTD in a similar fashion.

Continue …

- Another example of working with time would be comparing your current sales with the sales of a previous time period.
- For instance, if you want to see the total sales of the month next to the total sales of the prior month, you would enter the DAX measure definition, as shown in the following example:

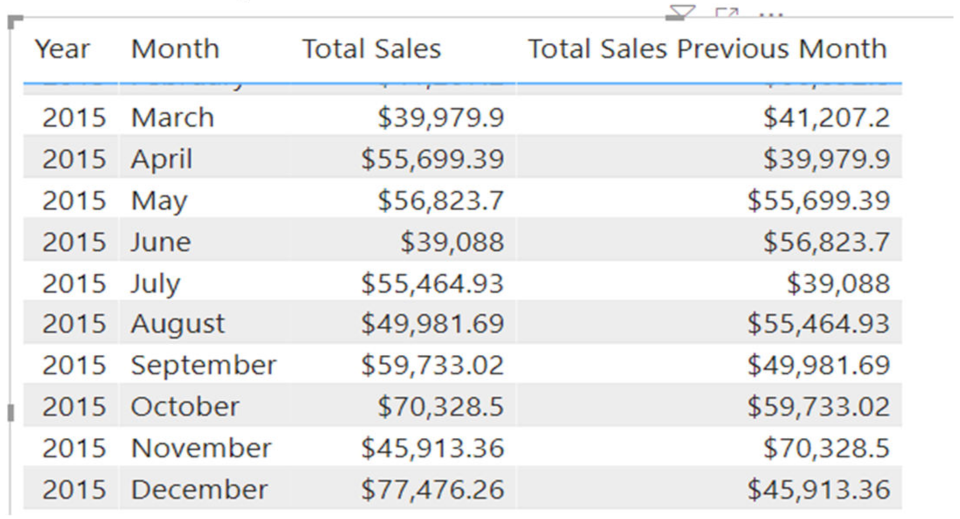
```
Total Sales Previous Month = CALCULATE
(
    sum('Sales OrderDetails'[Total Price])
    , PREVIOUSMONTH(Dates[Date])
)
```

- This measure uses the CALCULATE function, indicating that you're overriding the context to evaluate this expression the way that you want to.

Continue …

TEAM ACADEMY

- Using PREVIOUSMONTH for the override, which tells Power BI that, no matter what month is the default, the system should override it to be the previous month.
- The following screenshot shows the results in a table visual.



Year	Month	Total Sales	Total Sales Previous Month
2015	March	$39,979.9	$41,207.2
2015	April	$55,699.39	$39,979.9
2015	May	$56,823.7	$55,699.39
2015	June	$39,088	$56,823.7
2015	July	$55,464.93	$39,088
2015	August	$49,981.69	$55,464.93
2015	September	$59,733.02	$49,981.69
2015	October	$70,328.5	$59,733.02
2015	November	$45,913.36	$70,328.5
2015	December	$77,476.26	$45,913.36

- When you examine the months side-by-side, notice that the total sales for July compare to the total sales for June.

Visit

teamacademy.qa

Contact us

info@teamacademy.net

+974 6623 0468

Follow us on

Team Academy Global

Team Academy Global

Team Academy Global