Content

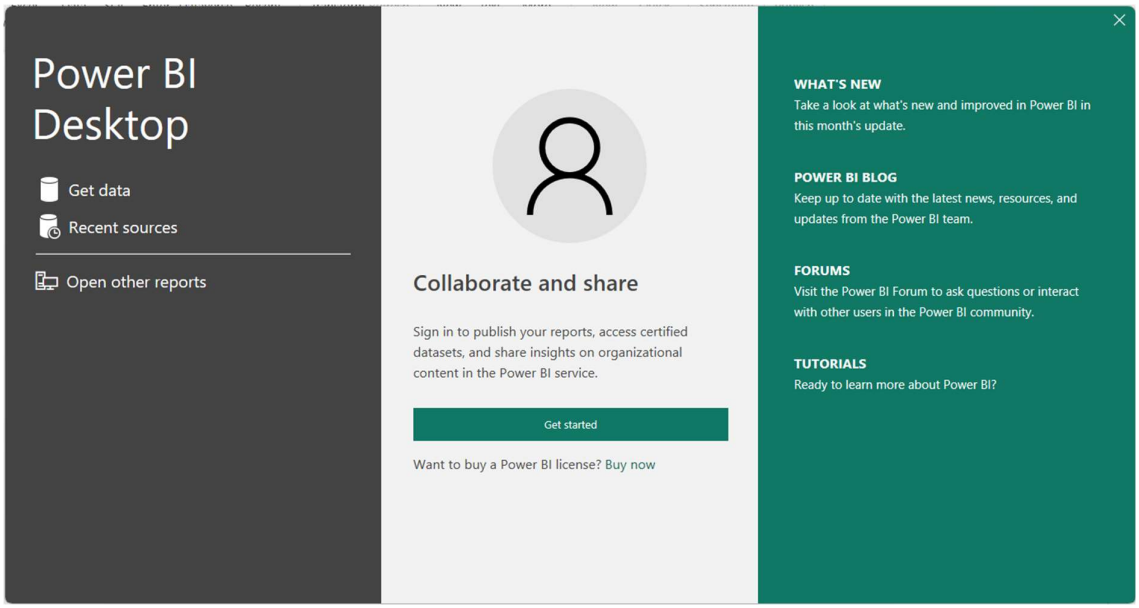
Introduction

DAX 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. Stated more simply, DAX helps you. create new information from data already in your model.

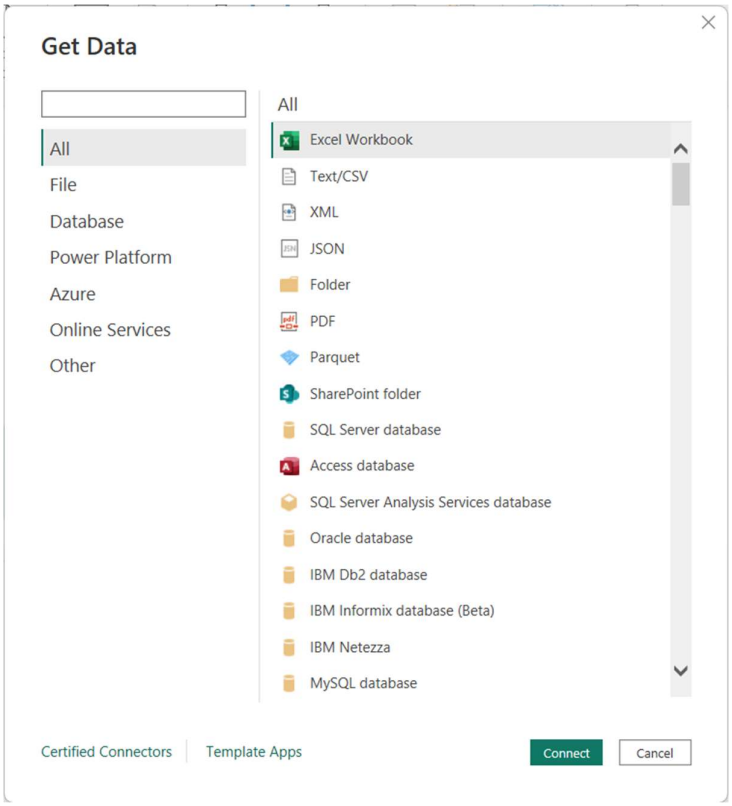
It's easy to create a new Power BI Desktop file and import some data into it. You can even create reports that show valuable insights without using any DAX formulas at all. But what if you need to analyze growth percentage across product categories and for different date ranges? Or you need to calculate year-over-year growth compared to market trends? DAX formulas provide this capability and many other important capabilities as well. Learning how to create effective DAX formulas will help you get the most out of your data. When you get the information you need, you can begin to solve real business problems that affect your bottom line.

Connecting Data

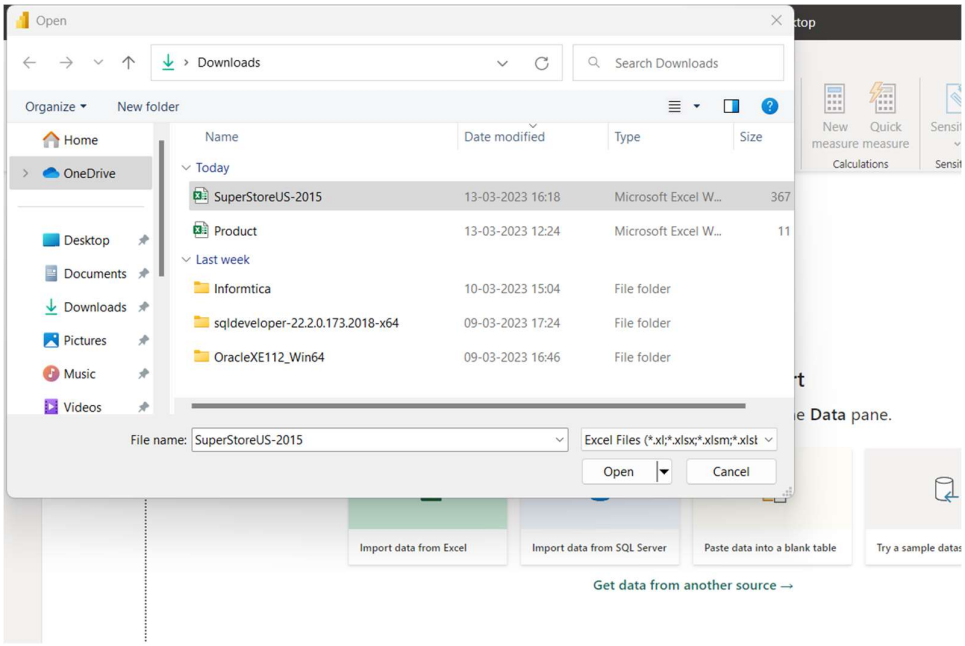
1. When you open the Power BI Desktop app you will see a window as below click Get Data option.



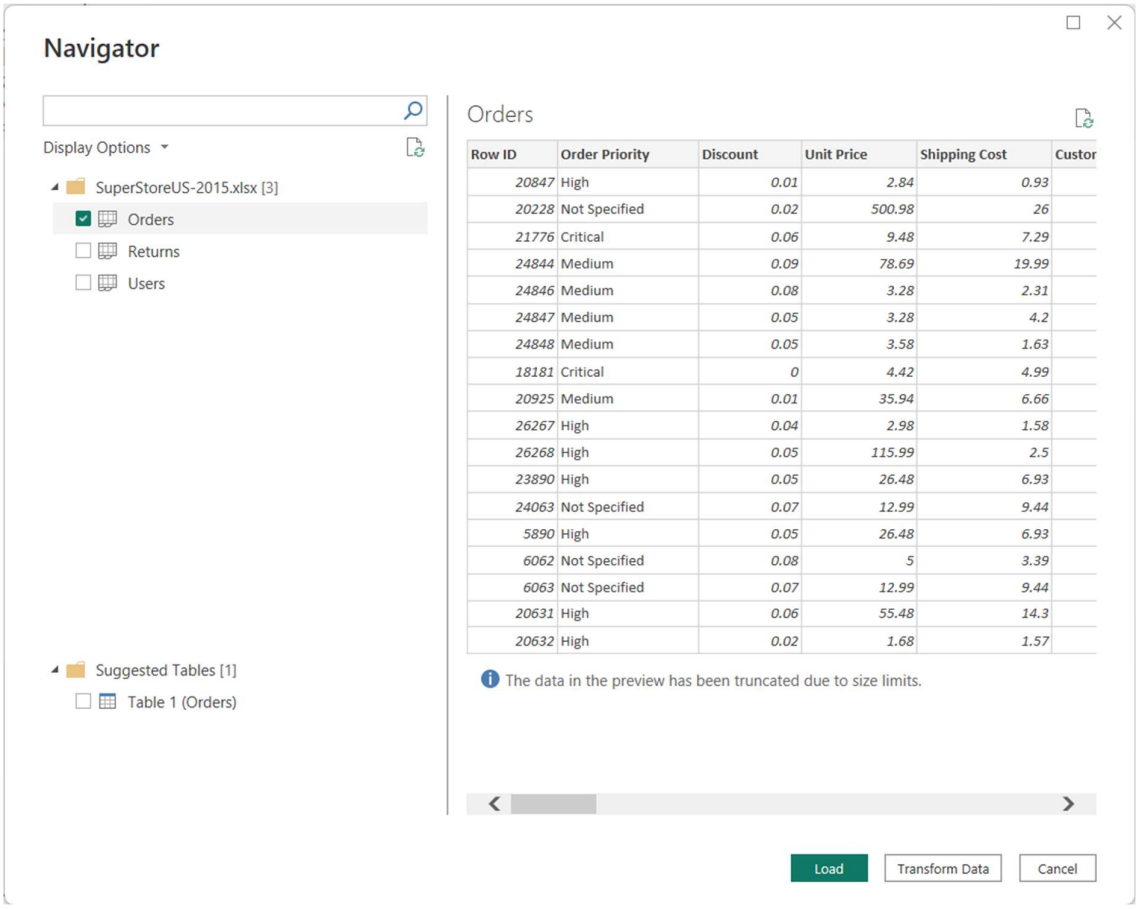
2. As this is an Excel file, select the Excel Workbook option from the drop-down list. Because our file is in Excel file format. And click connect.



3. Select the file named SuperStoreUS-2015 file and click open.



4. After selecting the file and select tables in the file, data will be displayed in the below format.



Filter DAX Functions

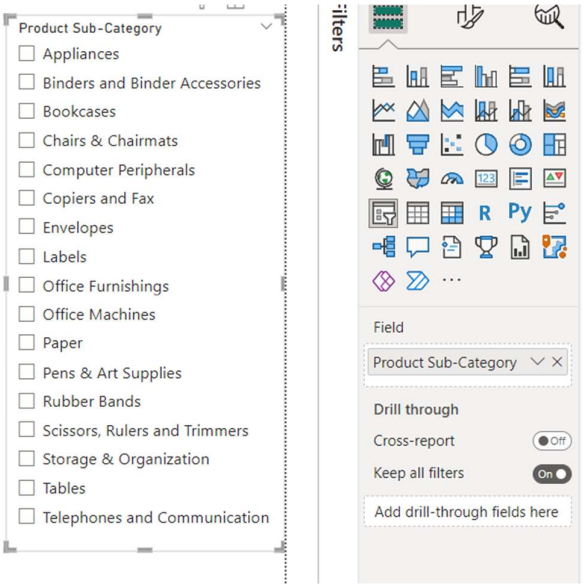
ALL function

As it's name suggests, Returns all the rows in a table, or all the values in a column. ALL function removes the applied filters from the filter context. Its comes under Filter function DAX category.

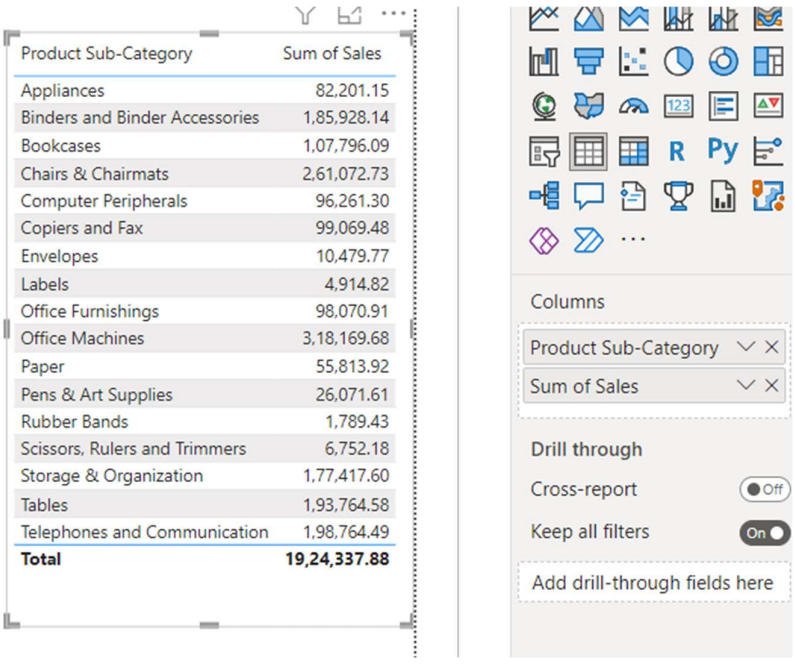
Syntax:

ALL ({<table> or <column>, [<column>], [<column>] …})

1. Create a slicer with following field as shown below.



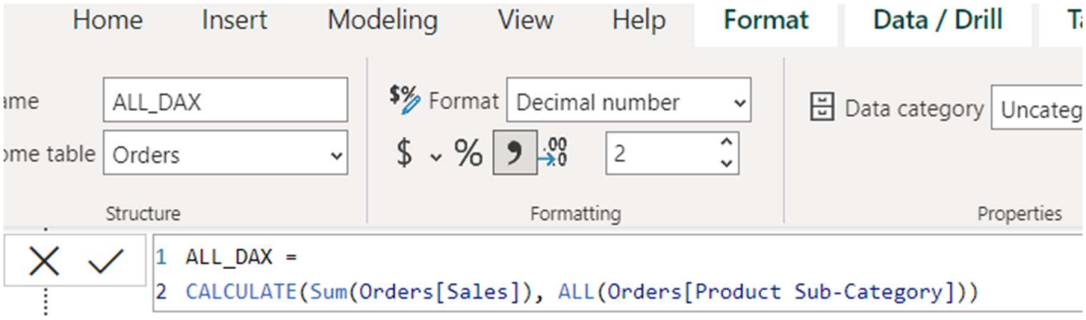
2. Create a table with following fields as shown below.



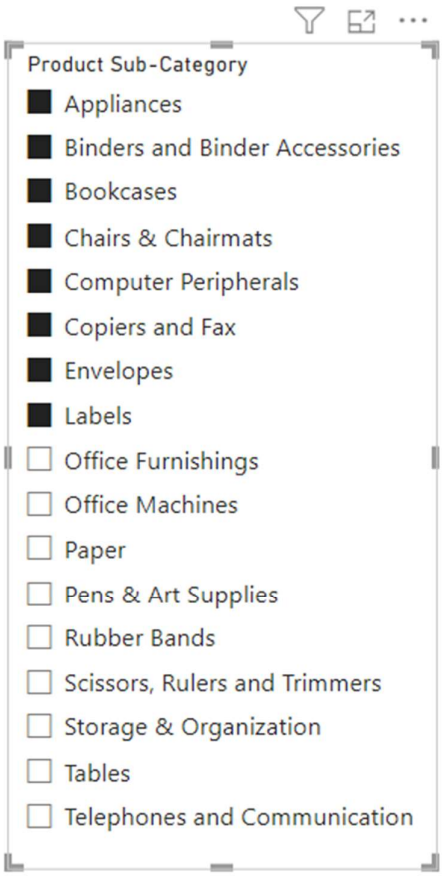
3. So now, we will create Measure to using DAX ALL function and in that measure we will do sum of 'Sales' column.

DAX: ALL_DAX =

CALCULATE(Sum(Orders[Sales]), ALL(Orders[Product Sub-Category]))



4. We passed 'Product sub category' column under ALL function, so in that case whatever filters you apply, it will always returns sum of all 'Product Sub Category and that is 19,24,337.88.



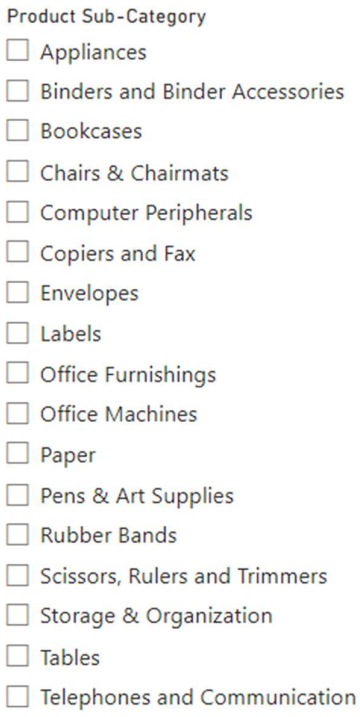
Product Sub-Category	Sum of Sales	ALL_DAX
Appliances	82,201.15	19,24,337.88
Binders and Binder Accessories	1,85,928.14	19,24,337.88
Bookcases	1,07,796.09	19,24,337.88
Chairs & Chairmats	2,61,072.73	19,24,337.88
Computer Peripherals	96,261.30	19,24,337.88
Copiers and Fax	99,069.48	19,24,337.88
Envelopes	10,479.77	19,24,337.88
Labels	4,914.82	19,24,337.88
Total	**8,47,723.48**	**19,24,337.88**

5. As per above filters output screen, we did filter few Product sub categories but it ignores the filters and returned sum of total sales.

ALLSELECTED function

Returns all the rows in a table, or all the values in a column. Removes context filters from columns and rows in the current query. keeping filters that come from outside.

6. Create a slicer and table as shown below.

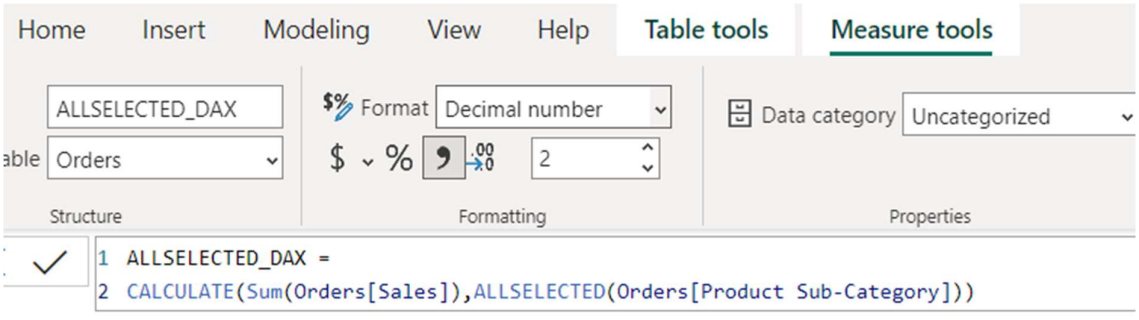


Product Sub-Category	Sum of Sales
Appliances	82,201.15
Binders and Binder Accessories	1,85,928.14
Bookcases	1,07,796.09
Chairs & Chairmats	2,61,072.73
Computer Peripherals	96,261.30
Copiers and Fax	99,069.48
Envelopes	10,479.77
Labels	4,914.82
Office Furnishings	98,070.91
Office Machines	3,18,169.68
Paper	55,813.92
Pens & Art Supplies	26,071.61
Rubber Bands	1,789.43
Scissors, Rulers and Trimmers	6,752.18
Storage & Organization	1,77,417.60
Tables	1,93,764.58
Telephones and Communication	1,98,764.49
Total	**19,24,337.88**

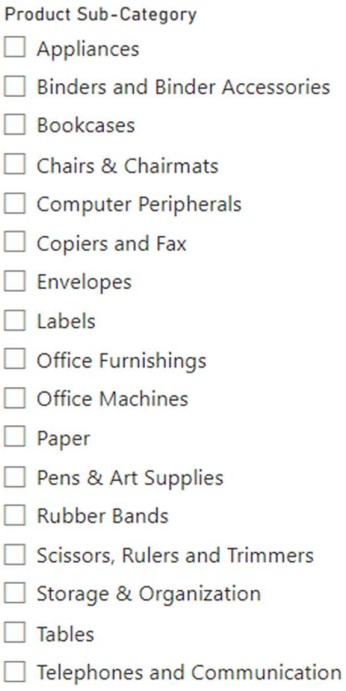
7. Now, create a Measure to using DAX ALLSELECTED function and in that measure do sum of 'Sales' column.

DAX: ALLSELECTED_DAX =

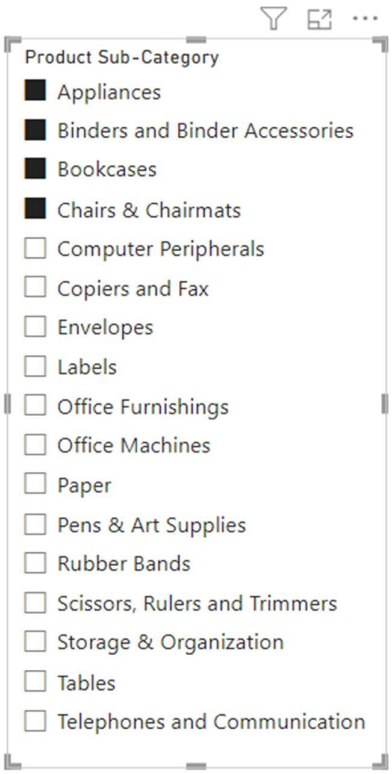
CALCULATE(Sum (Orders[Sales]),ALLSELECTED(Orders[Product Sub-Category]))



8. According to ALLSELECTED definition, by default it returns the sum of total sales of all rows and after filters returns the sum of applied filters values, let's have a look below two screen shots.



Product Sub-Category	Sum of Sales	ALLSELECTED_DAX
Appliances	82,201.15	19,24,337.88
Binders and Binder Accessories	1,85,928.14	19,24,337.88
Bookcases	1,07,796.09	19,24,337.88
Chairs & Chairmats	2,61,072.73	19,24,337.88
Computer Peripherals	96,261.30	19,24,337.88
Copiers and Fax	99,069.48	19,24,337.88
Envelopes	10,479.77	19,24,337.88
Labels	4,914.82	19,24,337.88
Office Furnishings	98,070.91	19,24,337.88
Office Machines	3,18,169.68	19,24,337.88
Paper	55,813.92	19,24,337.88
Pens & Art Supplies	26,071.61	19,24,337.88
Rubber Bands	1,789.43	19,24,337.88
Scissors, Rulers and Trimmers	6,752.18	19,24,337.88
Storage & Organization	1,77,417.60	19,24,337.88
Tables	1,93,764.58	19,24,337.88
Telephones and Communication	1,98,764.49	19,24,337.88
Total	**19,24,337.88**	**19,24,337.88**



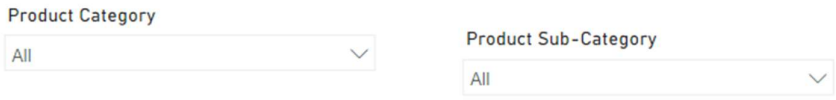
Product Sub-Category	Sum of Sales	ALLSELECTED_DAX
Appliances	82,201.15	6,36,998.11
Binders and Binder Accessories	1,85,928.14	6,36,998.11
Bookcases	1,07,796.09	6,36,998.11
Chairs & Chairmats	2,61,072.73	6,36,998.11
Total	**6,36,998.11**	**6,36,998.11**

ALLEXCEPT function

ALLEXCEPT is a DAX function, and it removes all context filters in the table except filters that have been applied to the specified columns. Its comes under Filter functions Dax category.

Syntax: ALLEXCEPT (<table>, <column>, [<column>] …)

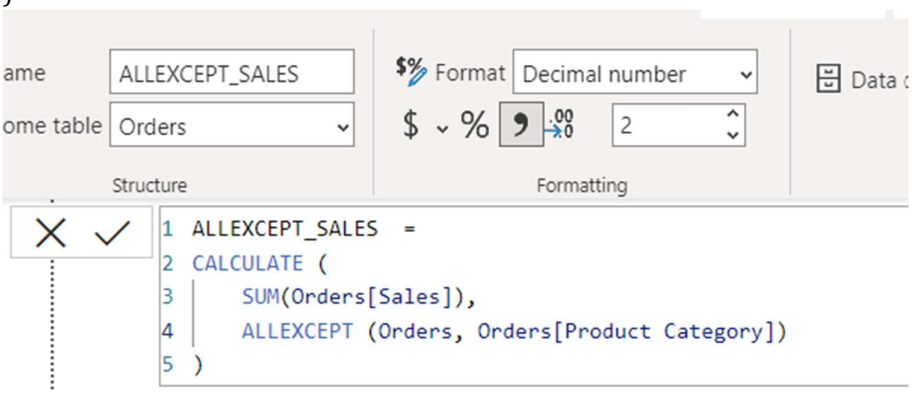
9. Drag Product Category in first slicer & Product subcategory in second slicer.
10. Drag three fields in table, Product Category, Product Subcategory & Sales from Orders Dataset



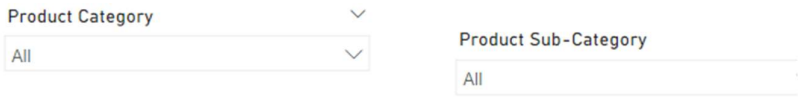
Product Category	Product Sub-Category	Sum of Sales
Furniture	Bookcases	1,07,796.09
Furniture	Chairs & Chairmats	2,61,072.73
Furniture	Office Furnishings	98,070.91
Furniture	Tables	1,93,764.58
Office Supplies	Appliances	82,201.15
Office Supplies	Binders and Binder Accessories	1,85,928.14
Office Supplies	Envelopes	10,479.77
Office Supplies	Labels	4,914.82
Office Supplies	Paper	55,813.92
Office Supplies	Pens & Art Supplies	26,071.61
Office Supplies	Rubber Bands	1,789.43
Office Supplies	Scissors, Rulers and Trimmers	6,752.18
Office Supplies	Storage & Organization	1,77,417.60
Technology	Computer Peripherals	96,261.30
Technology	Copiers and Fax	99,069.48
Technology	Office Machines	3,18,169.68
Technology	Telephones and Communication	1,98,764.49
Total		**19,24,337.88**

11. Create Measure and write DAX formula for ALLEXCEPT function.

DAX: ALLEXCEPT_SALES =
CALCULATE (
  SUM (Orders [Sales]),
  ALLEXCEPT (Orders, Orders [Product Category])
)

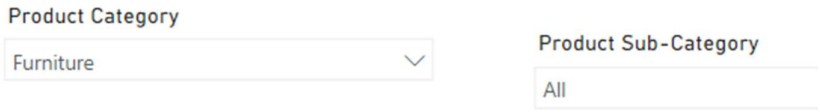


12. Now Drag ALLEXCEPT_SALES measures into table.



Product Category	Product Sub-Category	Sum of Sales	ALLEXCEPT_SALES
Furniture	Bookcases	1,07,796.09	6,60,704.31
Furniture	Chairs & Chairmats	2,61,072.73	6,60,704.31
Furniture	Office Furnishings	98,070.91	6,60,704.31
Furniture	Tables	1,93,764.58	6,60,704.31
Office Supplies	Appliances	82,201.15	5,51,368.62
Office Supplies	Binders and Binder Accessories	1,85,928.14	5,51,368.62
Office Supplies	Envelopes	10,479.77	5,51,368.62
Office Supplies	Labels	4,914.82	5,51,368.62
Office Supplies	Paper	55,813.92	5,51,368.62
Office Supplies	Pens & Art Supplies	26,071.61	5,51,368.62
Office Supplies	Rubber Bands	1,789.43	5,51,368.62
Office Supplies	Scissors, Rulers and Trimmers	6,752.18	5,51,368.62
Office Supplies	Storage & Organization	1,77,417.60	5,51,368.62
Technology	Computer Peripherals	96,261.30	7,12,264.95
Technology	Copiers and Fax	99,069.48	7,12,264.95
Technology	Office Machines	3,18,169.68	7,12,264.95
Technology	Telephones and Communication	1,98,764.49	7,12,264.95
Total		**19,24,337.88**	**19,24,337.88**

13. Now put filter on Product Category & see the measure ALLEXCEPT_SALES result, it is returning Total Sales sum of Furniture.



Product Category	Product Sub-Category	Sum of Sales	ALLEXCEPT_SALES
Furniture	Bookcases	1,07,796.09	6,60,704.31
Furniture	Chairs & Chairmats	2,61,072.73	6,60,704.31
Furniture	Office Furnishings	98,070.91	6,60,704.31
Furniture	Tables	1,93,764.58	6,60,704.31
Total		**6,60,704.31**	**6,60,704.31**

14. Now put filters on both slicers and see the result.

Product Category	Product Sub-Category	Sum of Sales	ALLEXCEPT_SALES
Furniture	Bookcases	1,07,796.09	6,60,704.31
Total		**1,07,796.09**	**6,60,704.31**

15. According to definition, it is avoiding filter for Product sub category slicer and only returning specified filter values like Product Category.

CALCULATE Function

CALCULATE DAX function evaluates an expression in a modified filter context. Its comes under Filter DAX function category.

Syntax:

CALCULATE(<expression>, <filter1>, <filter2>…)

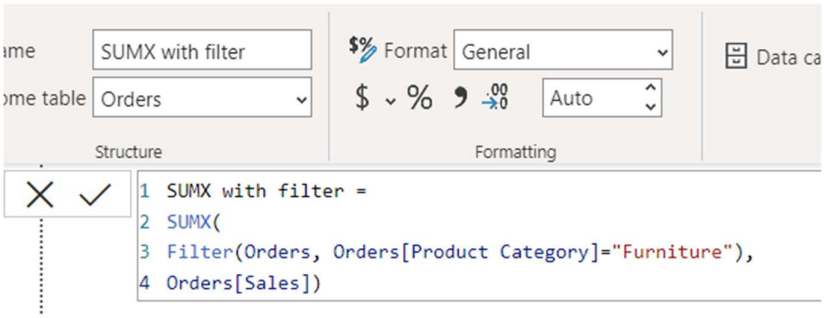
16. Suppose you want to calculate sum of sales for particular Product category, so for this you can use SUMX function because SUMX function support two argument, in first argument you can pass filter condition and in second argument you can pass sales column name.

DAX: SUMX with filter =

SUMX(

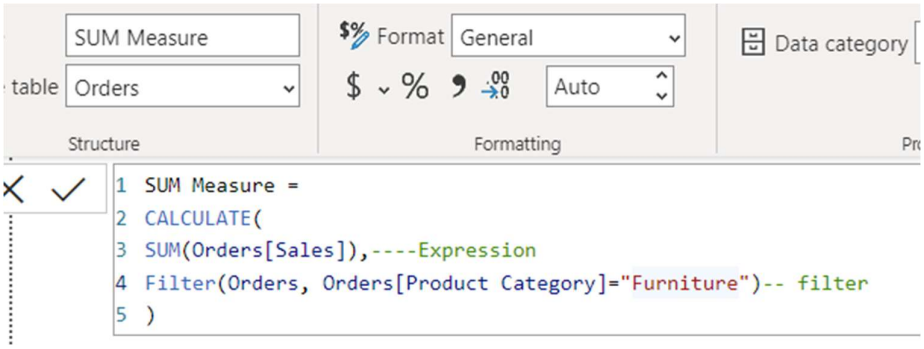
Filter(Orders, Orders[Product Category]="Furniture"),

Orders[Sales])

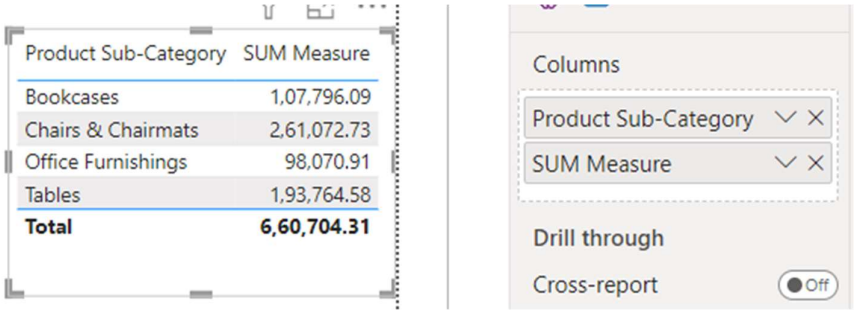


17. But if same thing you want to perform with SUM function, then how you will do that?
18. Here challenge is that SUM function only supports single argument that is column name.
19. So, you can use CALCULATE function with SUM function, as you know it support two argument, expression & filter.
20. In first argument (expression) you can use aggregation function like sum and in second argument you can use Filter condition.

DAX: SUM Measure =
CALCULATE (
SUM(Orders[Sales]),----Expression
Filter(Orders, Orders[Product Category]="Furniture")-- filter
)



21. To check the measure, drag and drop the following fields as shown below.



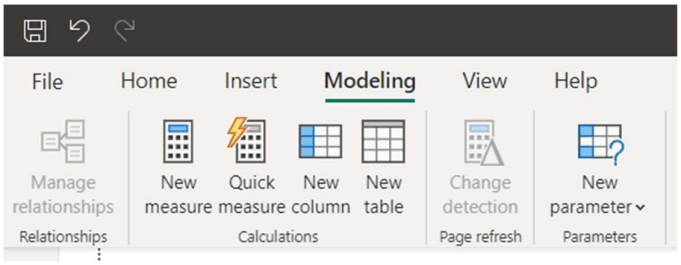
CALCULATETABLE Function

CALCULATETABLE DAX function comes under Power BI Filter DAX category and it evaluates a table expression in a context modified by the given filters. It returns a table of values.

CALCULATETABLE allows you to create virtual tables that you can filter using multiple conditions and use that table to make further calculations.

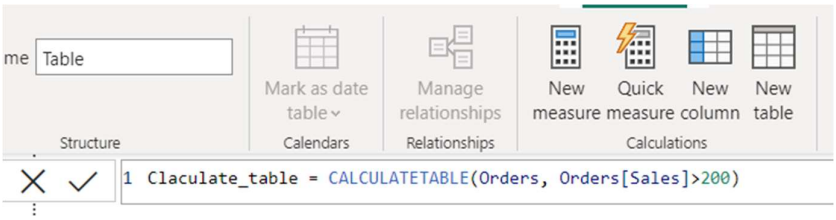
Syntax: CALCULATETABLE(<expression>, <filter1> , <filter2> , …)

22. Go to Modelling tab and click on New Table option as shown below.

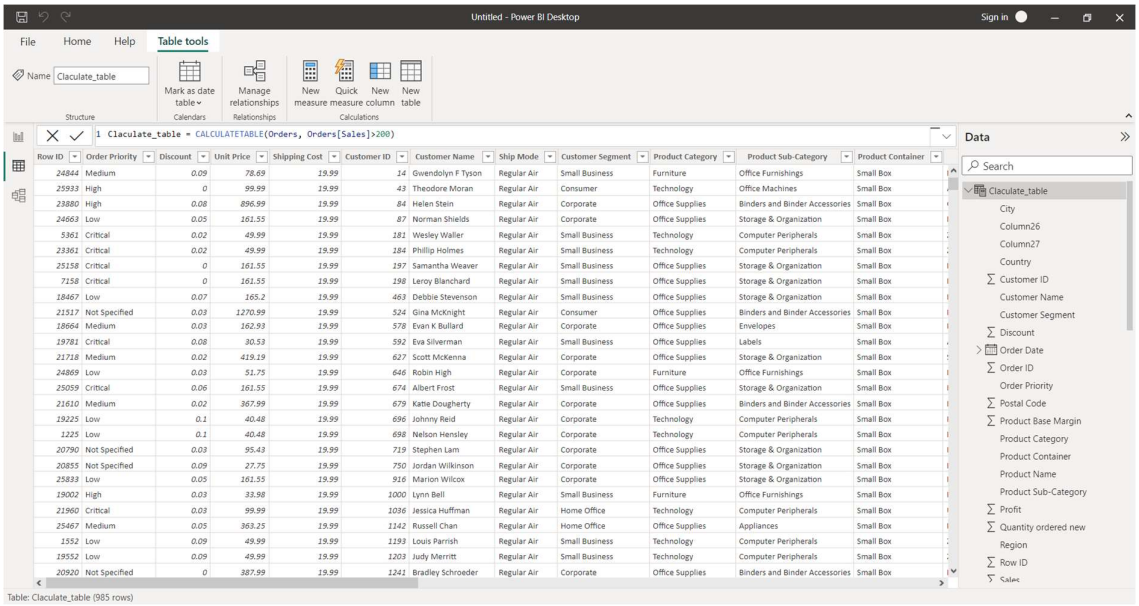


23. After that Write below DAX function.

DAX: Claculate_table = CALCULATETABLE(Orders, Orders[Sales]>200)



24. As you can see in below screenshot, it returns new table with given condition data where sales is > 200.



CALCULATETABLE with Measure

You can also use CALCULATETABLE with measure based on your requirement. It will create virtual tables that you can filter using multiple conditions and use that table to make further calculations.

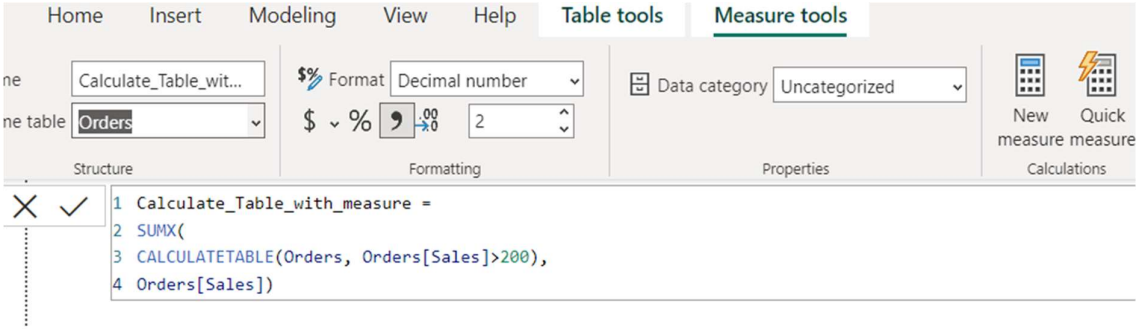
25. Create one measure and write below DAX code.

DAX: Calculate_Table_with_measure =

SUMX(

CALCULATETABLE(Orders, Orders[Sales]>200),

Orders[Sales])



26. Now create a table with following fields in the Calculate_table.
27. Create a Card with following field in the Order table as shown below.

Product Category	Sum of Sales
Furniture	6,50,463.91
Office Supplies	5,03,875.31
Technology	7,01,062.88
Total	**18,55,402.10**

28. Here we are getting same total values as shown above.

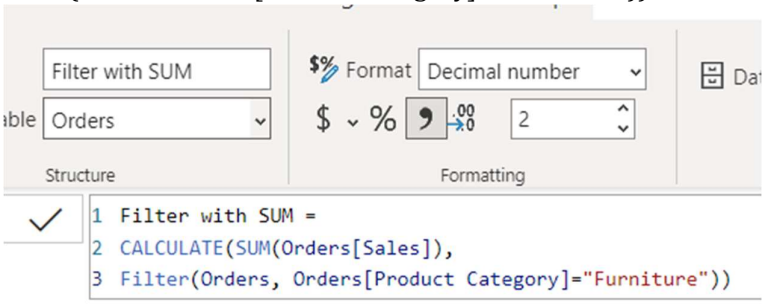
Filter Function

Returns a table that represents a subset of another table or expression. Its comes under Filter DAX functions category.

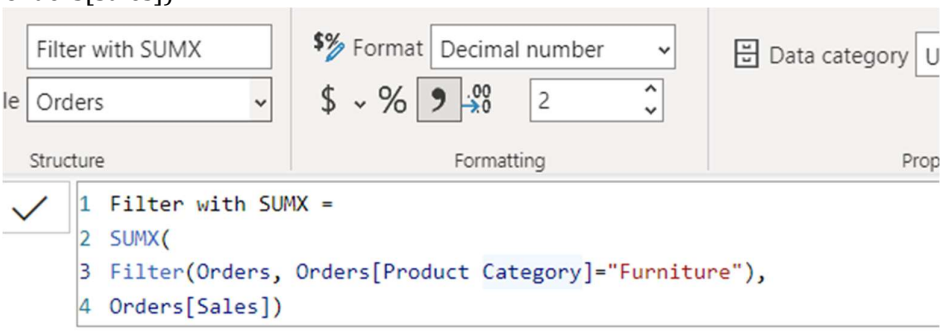
Syntax: FILTER(<table>, <filter>)

29. Calculate sum of sales only for Furniture Category:
30. Here, filter function filters the rows only for "Furniture" category and returns the table, then sum function will summation the sales values.

DAX: Filter with SUM =
CALCULATE(SUM(Orders[Sales]),
Filter(Orders, Orders[Product Category]="Furniture"))



DAX: Filter with SUMX =
SUMX(
Filter(Orders, Orders[Product Category]="Furniture"),
Orders[Sales])



LOOKUPVALUE Function

Returns the value for the row that meets all criteria specified by search conditions. The function can apply one or more search conditions. It comes under Filter function DAX category.

Syntax: LOOKUPVALUE(

<result_columnName>,

<search_columnName>,

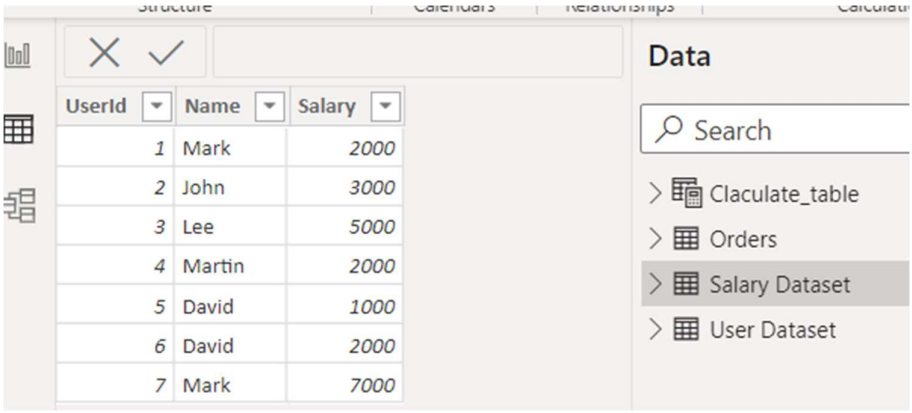
<search_value>,

<search2_columnName>, <search2_value>…

, <alternateResult>

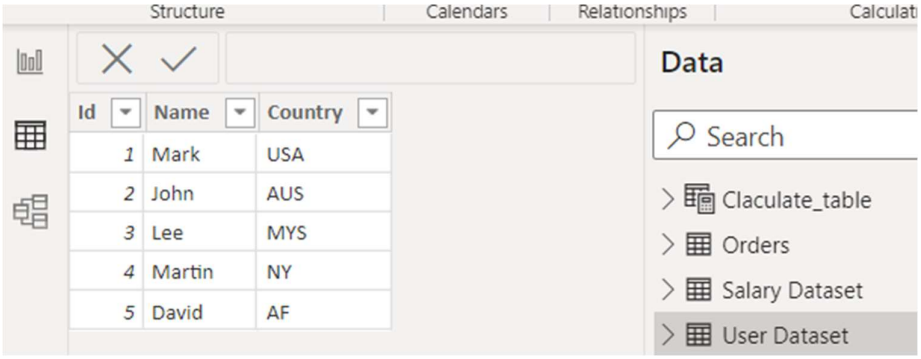
)

31. Here we have created two tables User Dataset & Salary Dataset and there is no relationship between both tables.
32. This Salary Dataset table.



UserId	Name	Salary
1	Mark	2000
2	John	3000
3	Lee	5000
4	Martin	2000
5	David	1000
6	David	2000
7	Mark	7000

33. This is our User Dataset table.

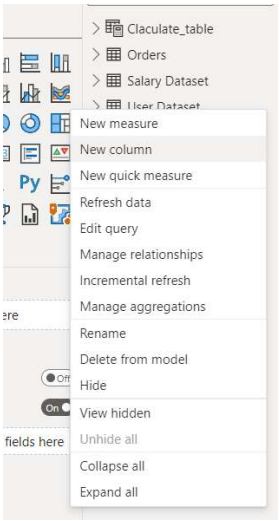


Id	Name	Country
1	Mark	USA
2	John	AUS
3	Lee	MYS
4	Martin	NY
5	David	AF

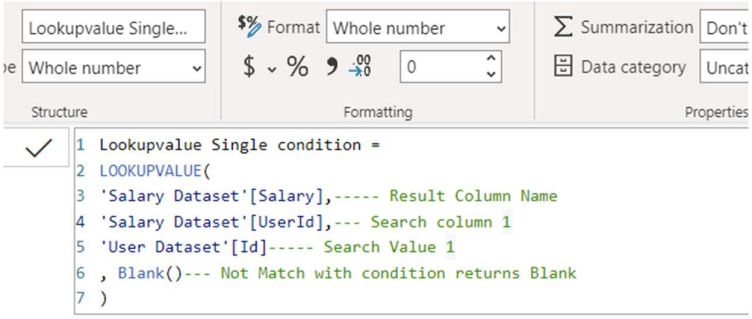
34. But in both tables, we have some user name's & Id's are common. So help of LOOKUPVALUE DAX, we will fetch salary values from "Salary Dataset" Table and will add into "User Dataset" table.

LOOKUPVALUE DAX with Single Condition:

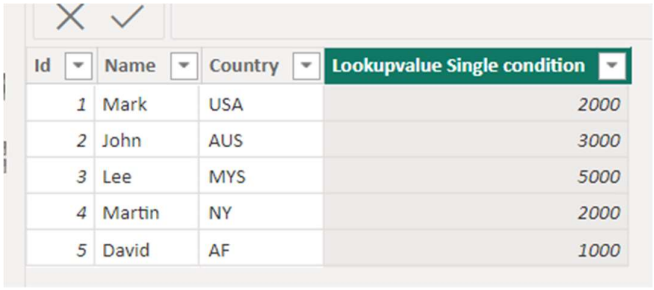
35. Right click to user dataset and add New Column.



DAX: Lookupvalue Single condition =
LOOKUPVALUE(
'Salary Dataset'[Salary],----- Result Column Name
'Salary Dataset'[UserId],--- Search column 1
'User Dataset'[Id]----- Search Value 1
, Blank()--- Not Match with condition returns Blank
)



36. Now see the result in the User Dataset table. A new column has been added as shown below.

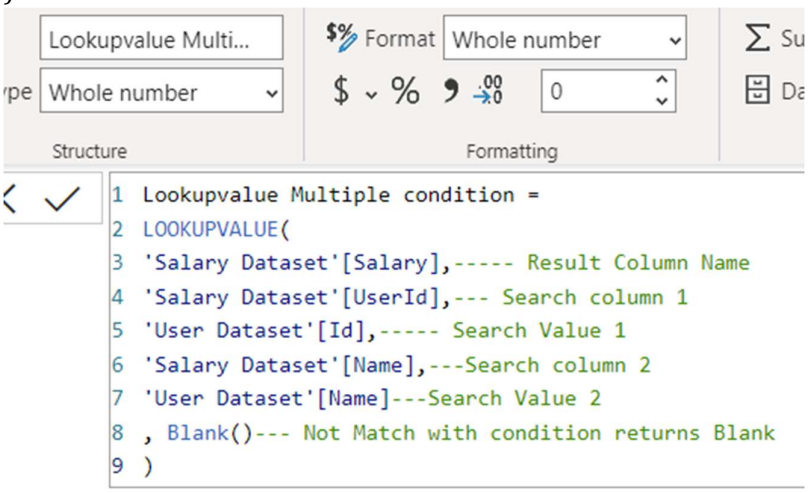


Id	Name	Country	Lookupvalue Single condition
1	Mark	USA	2000
2	John	AUS	3000
3	Lee	MYS	5000
4	Martin	NY	2000
5	David	AF	1000

LOOKUPVALUE DAX with Multiple condition:

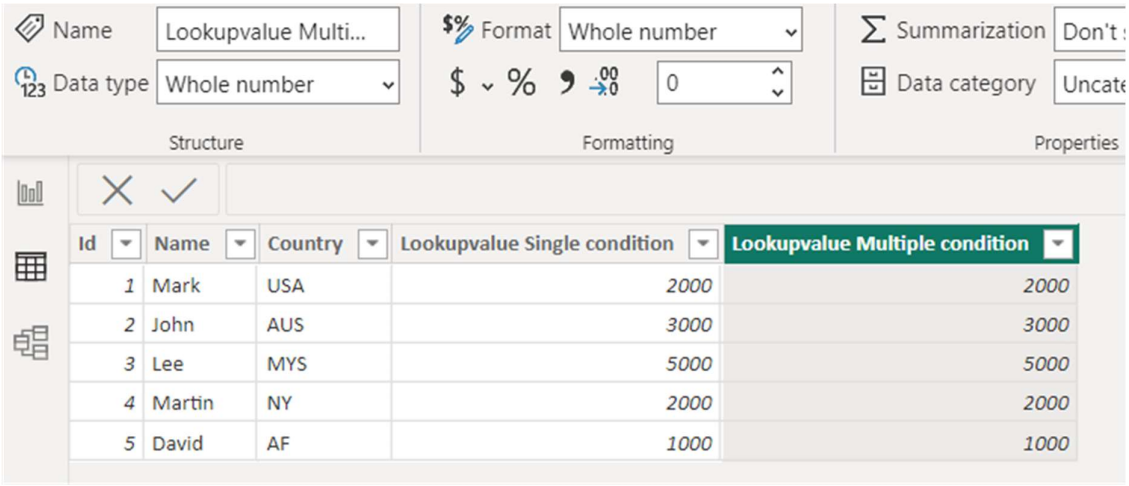
37. Create one more calculated column for Lookupvalue DAX with multiple conditions.

DAX: Lookupvalue Multiple condition =
LOOKUPVALUE(
'Salary Dataset'[Salary],----- Result Column Name
'Salary Dataset'[UserId],--- Search column 1
'User Dataset'[Id],----- Search Value 1
'Salary Dataset'[Name],---Search column 2
'User Dataset'[Name]---Search Value 2
, Blank()--- Not Match with condition returns Blank
)




```
1 Lookupvalue Multiple condition =
2 LOOKUPVALUE(
3 'Salary Dataset'[Salary],----- Result Column Name
4 'Salary Dataset'[UserId],--- Search column 1
5 'User Dataset'[Id],----- Search Value 1
6 'Salary Dataset'[Name],---Search column 2
7 'User Dataset'[Name]---Search Value 2
8 , Blank()--- Not Match with condition returns Blank
9 )
```

38. Now see the result in the User Dataset table. A new column has been added as shown below.



Id	Name	Country	Lookupvalue Single condition	Lookupvalue Multiple condition
1	Mark	USA	2000	2000
2	John	AUS	3000	3000
3	Lee	MYS	5000	5000
4	Martin	NY	2000	2000
5	David	AF	1000	1000

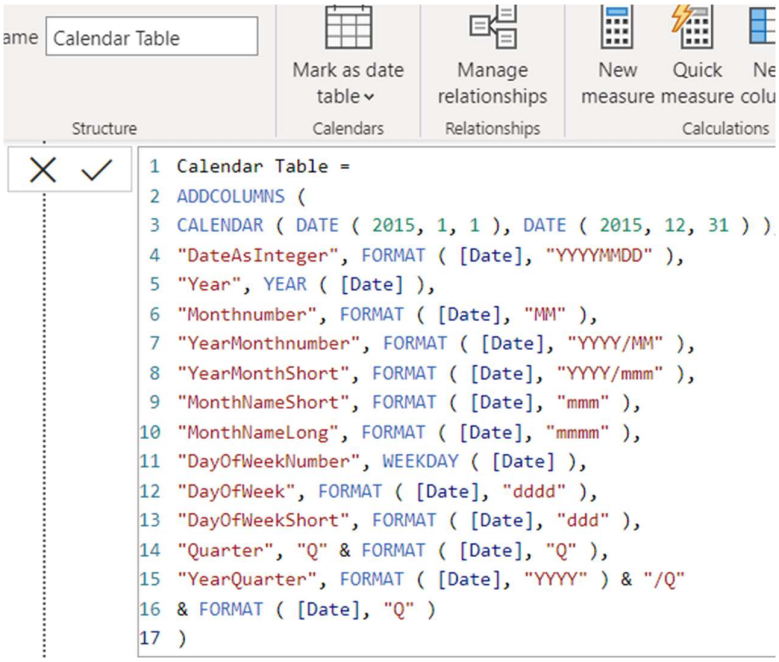
OFFSET DAX Function

OFFSET allows you to perform comparison calculations more easily by retrieving a row that is in a relative position from your current position. In short "Comparing values against a previous value"

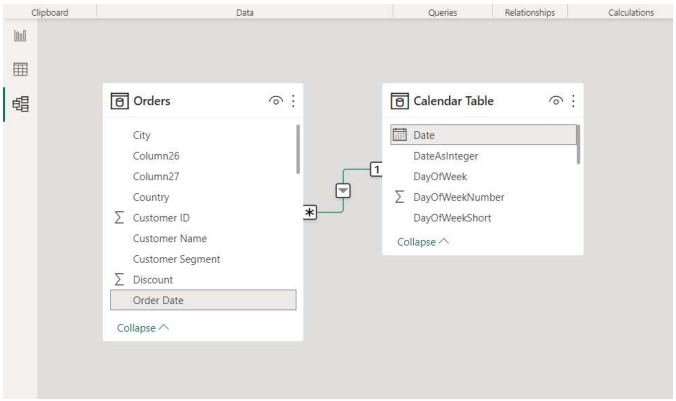
It comes under Filter functions Dax category. Also, you can use two helper functions ORDERBY and PARTITIONBY with OFFSET Function.

Syntax: OFFSET (<delta>, <relation>, <orderBy>, <blanks>, <partitionBy>)

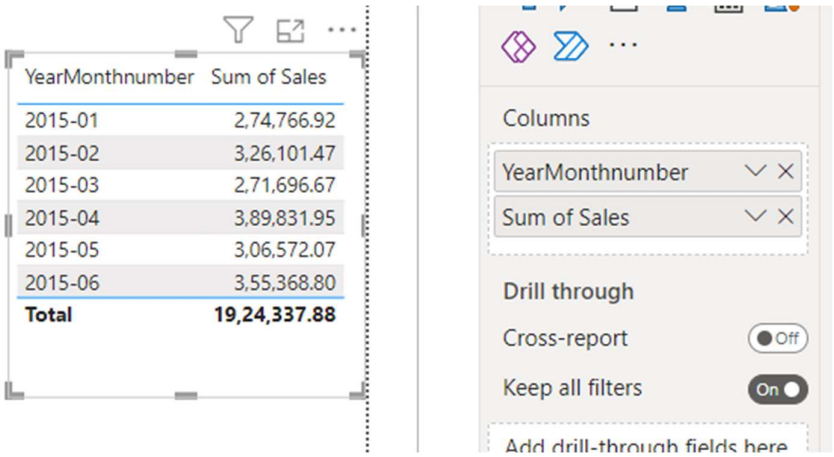
39. Create a Calendar Table using DAX function
    DAX:
    Calendar Table =
    ADDCOLUMNS (
    CALENDAR (DATE (2015, 1, 1), DATE (2015, 12, 31)),
    "DateAsInteger", FORMAT ([Date], "YYYYMMDD"),
    "Year", YEAR ([Date]),
    "Monthnumber", FORMAT ([Date], "MM"),
    "YearMonthnumber", FORMAT ([Date], "YYYY/MM"),
    "YearMonthShort", FORMAT ([Date], "YYYY/mmm"),
    "MonthNameShort", FORMAT ([Date], "mmm"),
    "MonthNameLong", FORMAT ([Date], "mmmm"),
    "DayOfWeekNumber", WEEKDAY ([Date]),
    "DayOfWeek", FORMAT ([Date], "dddd"),
    "DayOfWeekShort", FORMAT ([Date], "ddd"),
    "Quarter", "Q" & FORMAT ([Date], "Q"),
    "YearQuarter", FORMAT ([Date], "YYYY") & "/Q"
    & FORMAT ([Date], "Q")
)



40. Now create a relationship between Calendar table and Order Dataset.



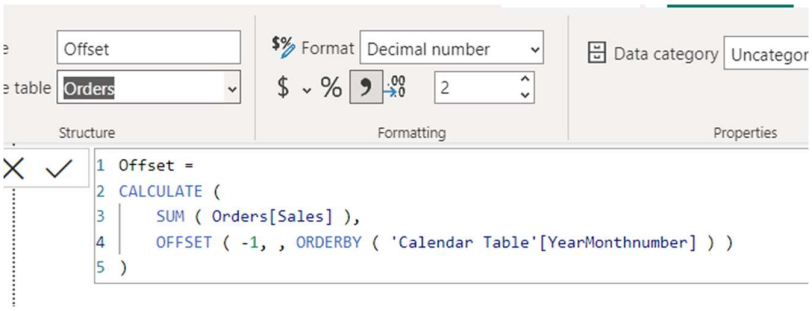
41. After that add one table visual with two fields-
42. Drag YearMonthnumber from Calendar table & Sales field from Order dataset.



43. Create a measure with below DAX formula.

Dax: Offset =

```
CALCULATE (
   SUM ( Orders[Sales] ),
   OFFSET ( -1, , ORDERBY ( 'Calendar Table'[YearMonthnumber] ) )
)
```



44. Now add that measure to table visual and see the result. Here you can compare the current value with previous value.

YearMonthnumber	Sum of Sales	Offset
2015-01	2,74,766.92	
2015-02	3,26,101.47	2,74,766.92
2015-03	2,71,696.67	3,26,101.47
2015-04	3,89,831.95	2,71,696.67
2015-05	3,06,572.07	3,89,831.95
2015-06	3,55,368.80	3,06,572.07
2015-07		3,55,368.80
Total	**19,24,337.88**	**19,24,337.88**

Aggregation Functions

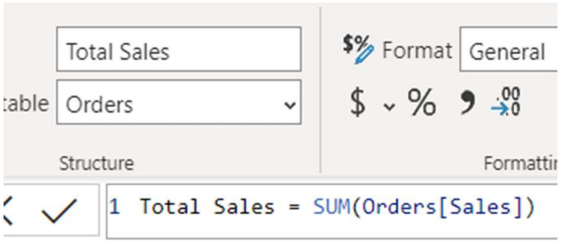
SUM function

The SUM function is a aggregation function and it calculates the sum of all numbers in a column.

Syntax: SUM(<Column>)

1. Create a measure for SUM function.

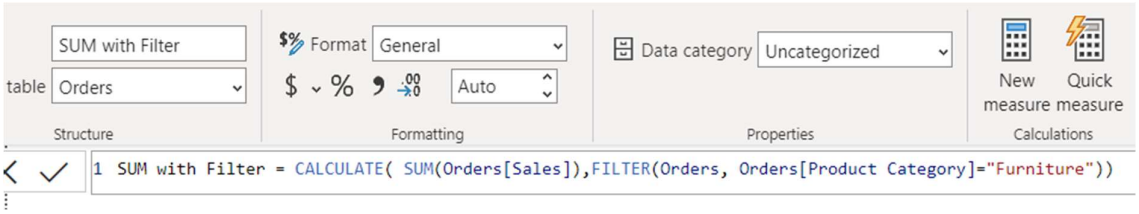
DAX: Total Sales = SUM(Orders[Sales])



2. Now drag “Total Sales” measure to card visual to see the output of sales measure.

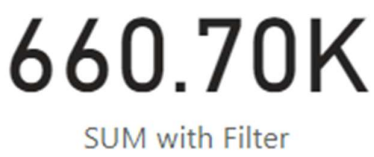
3. SUM function with Filter.
4. Create a measure to get the sales of “Furniture” category.

DAX: SUM with Filter = CALCULATE(SUM(Orders[Sales]),FILTER(Orders, Orders[Product Category]="Furniture"))



5. Output of above measure.

Product Category	Sum of Sales
Furniture	6,60,704.31
Office Supplies	5,51,368.62
Technology	7,12,264.95
Total	**19,24,337.88**



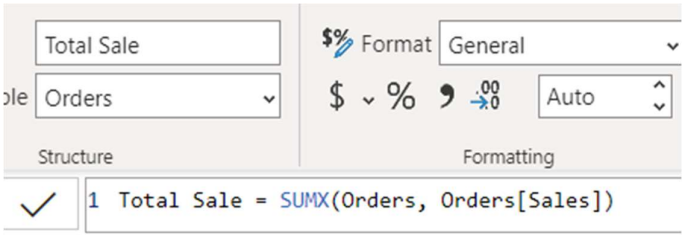
SUMX DAX function

SUMX is an iterator function. Returns the sum of an expression evaluated for each row in a table. With this function you can operate on multiple columns in table row wise.

Syntax: SUMX(<table>, <expression>)

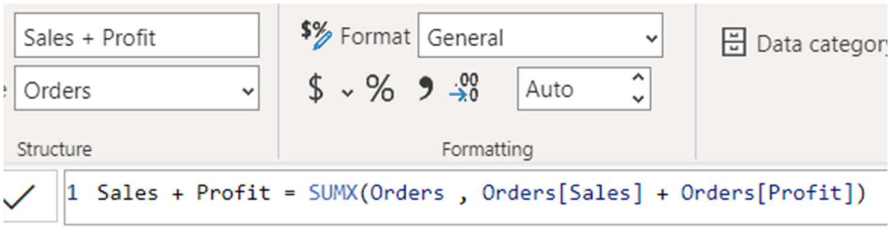
6. Get Total sales to using SUMX function:

DAX: Total Sale = SUMX(Orders, Orders[Sales])



7. Suppose you want to see row wise sum of “Sales” & “Profit” columns together.
8. Here, SUMX helps you because it is iterator function and perform the operation row wise.

DAX: Sales + Profit = SUMX(Orders , Orders[Sales] + Orders[Profit])

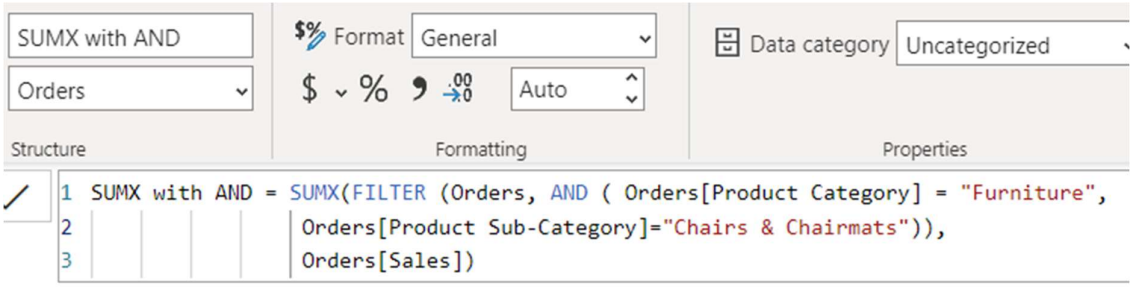


9. Create a table with the following fields as shown below to check the result.

Product Category	Product Sub-Category	Sum of Sales	Sum of Profit	Sales + Profit
Furniture	Bookcases	1,07,796.09	-930.44	1,06,865.65
Furniture	Chairs & Chairmats	2,61,072.73	48,695.84	3,09,768.57
Furniture	Office Furnishings	98,070.91	18,724.12	1,16,795.03
Furniture	Tables	1,93,764.58	-7,240.07	1,86,524.51
Office Supplies	Appliances	82,201.15	12,594.82	94,795.97
Office Supplies	Binders and Binder Accessories	1,85,928.14	59,296.39	2,45,224.53
Office Supplies	Envelopes	10,479.77	-1,194.41	9,285.36
Office Supplies	Labels	4,914.82	7,028.16	11,942.98
Office Supplies	Paper	55,813.92	7,769.32	63,583.24
Office Supplies	Pens & Art Supplies	26,071.61	-257.63	25,813.98
Office Supplies	Rubber Bands	1,789.43	-1,544.83	244.60
Office Supplies	Scissors, Rulers and Trimmers	6,752.18	-1,291.10	5,461.08
Office Supplies	Storage & Organization	1,77,417.60	7,124.29	1,84,541.89
Technology	Computer Peripherals	96,261.30	1,698.04	97,959.34
Technology	Copiers and Fax	99,069.48	23,990.21	1,23,059.69
Technology	Office Machines	3,18,169.68	8,824.39	3,26,994.07
Technology	Telephones and Communication	1,98,764.49	40,790.51	2,39,555.00
Total		**19,24,337.88**	**2,24,077.61**	**21,48,415.49**

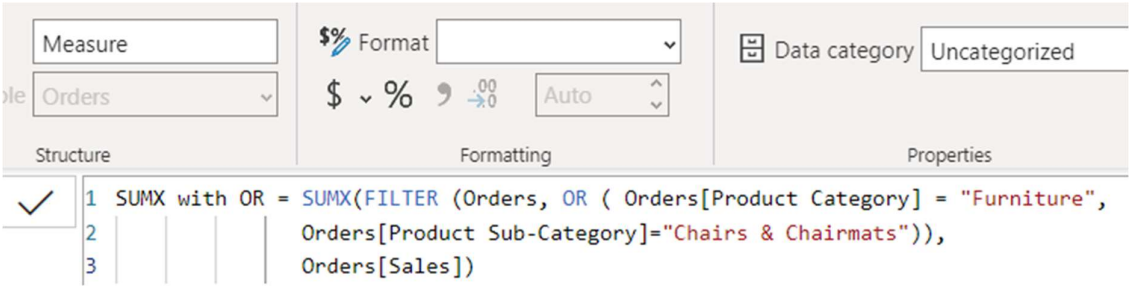
10. SUMX with AND function

DAX: SUMX with AND = SUMX(FILTER (Orders, AND (Orders[Product Category] = "Furniture", Orders[Product Sub-Category]="Chairs & Chairmats")), Orders[Sales])



11. SUMX with OR function

DAX: SUMX with OR = SUMX(FILTER (Orders, OR (Orders[Product Category] = "Furniture", Orders[Product Sub-Category]="Chairs & Chairmats")), Orders[Sales])



12. Create a table with the following fields as shown below to check the results.

Product Category	Product Sub-Category	Sum of Sales	SUMX with AND	SUMX with OR
Furniture	Bookcases	1,07,796.09		1,07,796.09
Furniture	Chairs & Chairmats	2,61,072.73	2,61,072.73	2,61,072.73
Furniture	Office Furnishings	98,070.91		98,070.91
Furniture	Tables	1,93,764.58		1,93,764.58
Office Supplies	Appliances	82,201.15		
Office Supplies	Binders and Binder Accessories	1,85,928.14		
Office Supplies	Envelopes	10,479.77		
Office Supplies	Labels	4,914.82		
Office Supplies	Paper	55,813.92		
Office Supplies	Pens & Art Supplies	26,071.61		
Office Supplies	Rubber Bands	1,789.43		
Office Supplies	Scissors, Rulers and Trimmers	6,752.18		
Office Supplies	Storage & Organization	1,77,417.60		
Technology	Computer Peripherals	96,261.30		
Technology	Copiers and Fax	99,069.48		
Technology	Office Machines	3,18,169.68		
Technology	Telephones and Communication	1,98,764.49		
Total		**19,24,337.88**	**2,61,072.73**	**6,60,704.31**

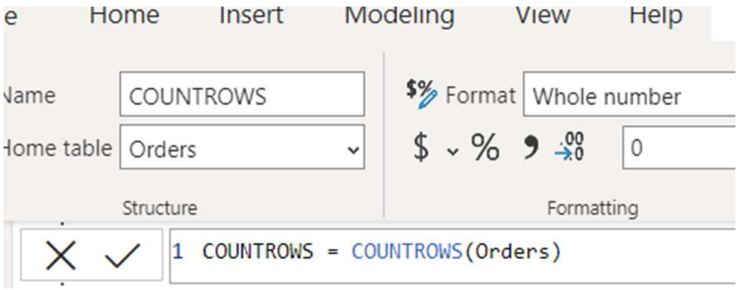
COUNTROWS Function

The COUNTROWS function is used to counts the number of rows in the specified table, or in a table defined by an expression. This function comes under Statistical Functions DAX category.

Syntax: COUNTROWS(<table>)

5. Create a measure for counts total no of rows in Orders Table/ Dataset.

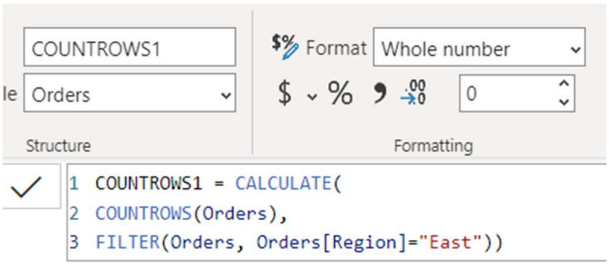
DAX: COUNTROWS = COUNTROWS(Orders)



6. Now take one card visual to see the output of measure.

7. You can also use COUTROWS DAX function with FILTER DAX, suppose you want to count particular Region rows.

DAX: COUNTROWS1 = CALCULATE(
COUNTROWS(Orders),
FILTER(Orders, Orders[Region]="East"))



8. It will return number of rows under East region.

DISTINCTCOUNT Function

Power BI DISTINCTCOUNT DAX function is used to counts the number of distinct values in a column. This functions comes under statistical functions Dax categories.

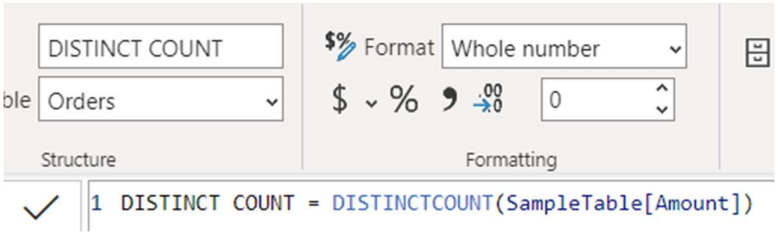
Syntax: DISTINCTCOUNT(<column>)

9. Create a sample table using enter data option in power bi as shown below.

	ID	Amount	Blank	Boolean	SalesDate	Name	+
1	1			TRUE	29/5/2020	A	
2	2	1000		FALSE		B	
3	3	2000		TRUE	29/5/2020		
4	4	1000		FALSE		C	
+							

10. Now we will count Distinct number of values under “Amount” Column. So for this create one new measure.

DAX: DISTINCT COUNT = DISTINCTCOUNT(SampleTable[Amount])



11. Let’s check result for the measure.

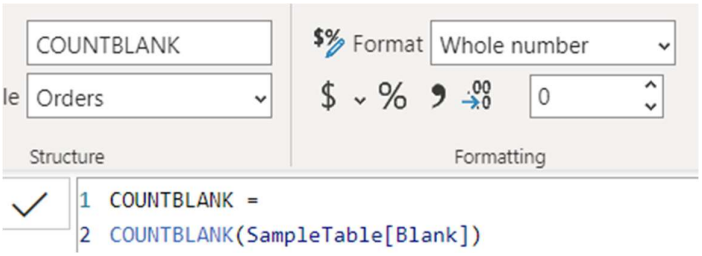
COUNTBLANK Function

COUNTBLANK DAX function is used to counts the number of blank cells in a column. This function comes under Statistical Functions DAX category.

Syntax: COUNTBLANK(<column>)

12. For this example, use sample table that we used before.
13. Now we will count number of Blank rows under “Blank” Column. So for this create one new measure.

DAX: COUNTBLANK = COUNTBLANK(SampleTable[Blank])



14. Above measure will return no of blank rows under “Blank” column is 4.

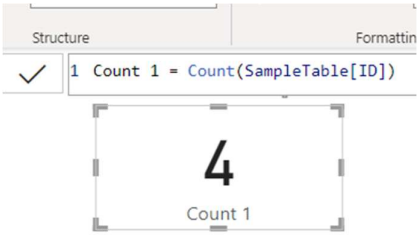
COUNT DAX Function

The COUNT function counts the number of cells in a column that contain non-blank values.

Syntax: COUNT(<column>)

15. For this example we are using previous sample data.
16. Let's create measure for COUNT function with ID column.

DAX: Count 1 = Count(SampleTable[ID])



COUNTA DAX Function

The COUNTA function counts the number of cells in a column that are not empty.

Syntax: COUNTA(<column>)

17. Let's create measure for COUNTA function with Id column.

DAX: COUNTA 1 = COUNTA(SampleTable[ID])

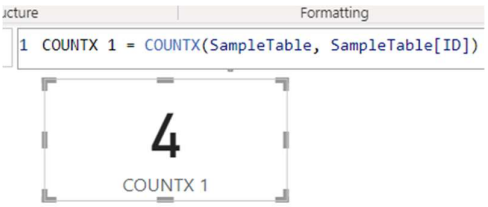
COUNTX DAX Function

The COUNTAX function counts nonblank results when evaluating the result of an expression over a table.

Syntax: COUNTAX(<table>,<expression>)

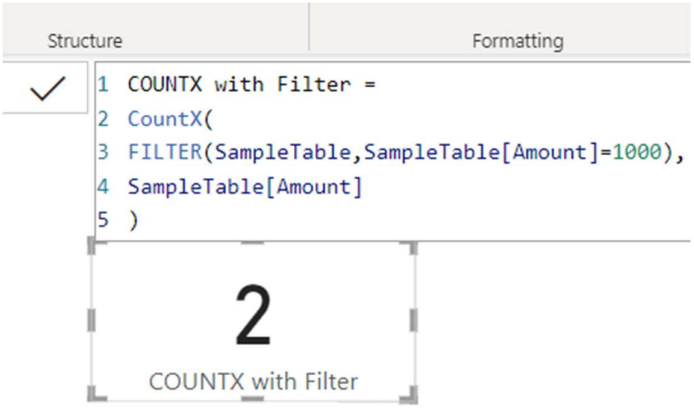
18. Let's create measure for COUNTX function with ID column.

DAX: COUNTX 1 = COUNTX(SampleTable, SampleTable[ID])



19. You can also used Filter DAX function with COUNTX.
20. Suppose you want to count no amount values where amount equal to 1000.

DAX: COUNTX with Filter =
CountX(
FILTER(SampleTable,SampleTable[Amount]=1000),
SampleTable[Amount]
)

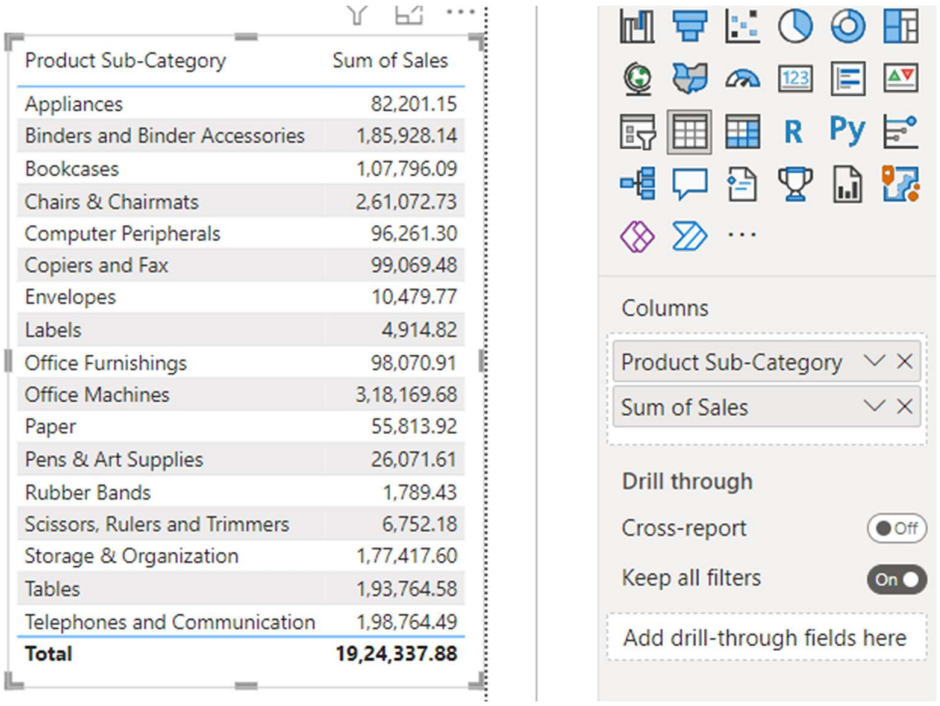


MIN DAX Function

Returns the smallest value in a column.

Syntax: MIN(<Column>)

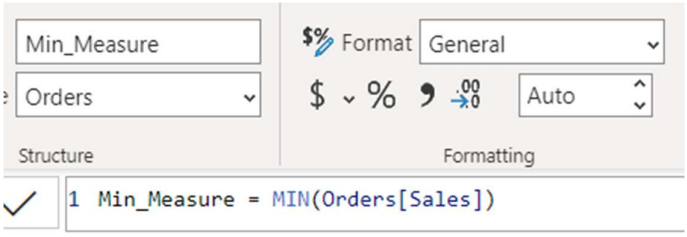
21. Create a table with the following fields as shown below.



Product Sub-Category	Sum of Sales
Appliances	82,201.15
Binders and Binder Accessories	1,85,928.14
Bookcases	1,07,796.09
Chairs & Chairmats	2,61,072.73
Computer Peripherals	96,261.30
Copiers and Fax	99,069.48
Envelopes	10,479.77
Labels	4,914.82
Office Furnishings	98,070.91
Office Machines	3,18,169.68
Paper	55,813.92
Pens & Art Supplies	26,071.61
Rubber Bands	1,789.43
Scissors, Rulers and Trimmers	6,752.18
Storage & Organization	1,77,417.60
Tables	1,93,764.58
Telephones and Communication	1,98,764.49
Total	**19,24,337.88**

22. Now Create Measure to find minimum sale value from sale column.
23. Right click on Dataset and click to New measure, then write below DAX

DAX: Min_Measure = MIN(Orders[Sales])



24. Now take Card visual from Visualization pane to Power Bi Page & drag measure over it.

Product Sub-Category	Min_Measure
Appliances	8.80
Binders and Binder Accessories	3.42
Bookcases	110.75
Chairs & Chairmats	19.97
Computer Peripherals	8.49
Copiers and Fax	706.56
Envelopes	8.34
Labels	3.07
Office Furnishings	2.77
Office Machines	29.85
Paper	5.76
Pens & Art Supplies	2.25
Rubber Bands	5.28
Scissors, Rulers and Trimmers	3.13
Storage & Organization	18.73
Tables	111.86
Telephones and Communication	17.83
Total	**2.25**

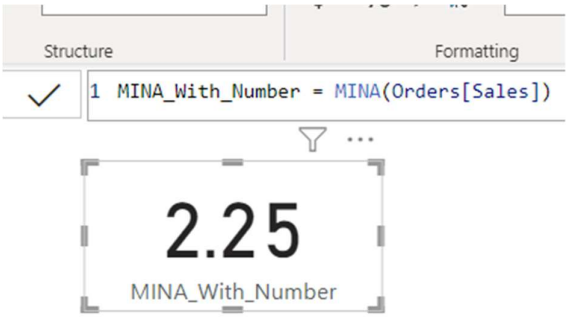
MINA DAX Function

Returns the smallest value in a column.

Syntax: MINA(<Column>)

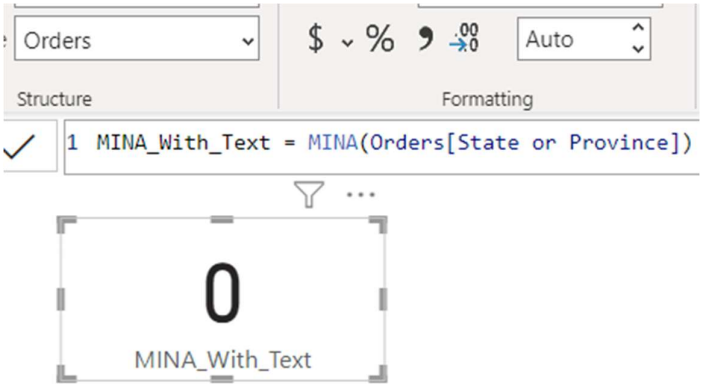
25. MINA function support Date, Number & Boolean data type.

DAX: MINA_With_Number = MINA(Orders[Sales])



26. It does not support Text data type, It will return 0.

DAX: MINA_With_Text = MINA(Orders[State or Province])



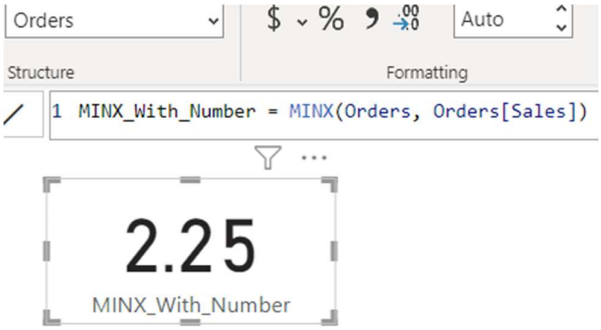
MINX DAX Function

Evaluates an expression for each row of a table and returns the smallest value.

Syntax: MINX(tablename, expression)

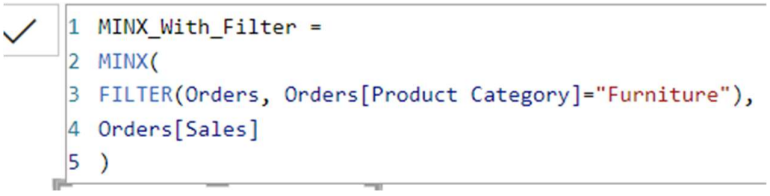
27. MINX function support Date, Number & Text data type.

DAX: MINX_With_Number = MINX(Orders, Orders[Sales])



28. You can use Filter condition with MINX Dax function.
29. Suppose you want minimum sales value only for “Furniture” category, so you can use Filter DAX with MINX.

DAX: MINX_With_Filter =
MINX(
FILTER(Orders, Orders[Product Category]="Furniture"),
Orders[Sales]
)



MAX DAX Function

Returns the largest value in a column.

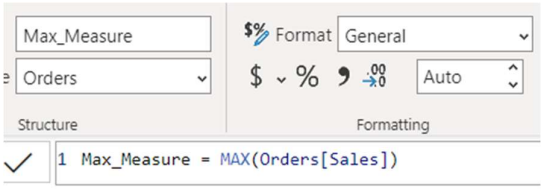
Syntax: MAX(<Column>)

30. Create a table with the following fields as shown below.

Product Sub-Category	Sum of Sales
Appliances	82,201.15
Binders and Binder Accessories	1,85,928.14
Bookcases	1,07,796.09
Chairs & Chairmats	2,61,072.73
Computer Peripherals	96,261.30
Copiers and Fax	99,069.48
Envelopes	10,479.77
Labels	4,914.82
Office Furnishings	98,070.91
Office Machines	3,18,169.68
Paper	55,813.92
Pens & Art Supplies	26,071.61
Rubber Bands	1,789.43
Scissors, Rulers and Trimmers	6,752.18
Storage & Organization	1,77,417.60
Tables	1,93,764.58
Telephones and Communication	1,98,764.49
Total	**19,24,337.88**

31. Now Create Measure to find maximum sale value from sale column.
32. Right click on Dataset and click to New measure, then write below DAX

DAX: Max_Measure = MAX(Orders[Sales])

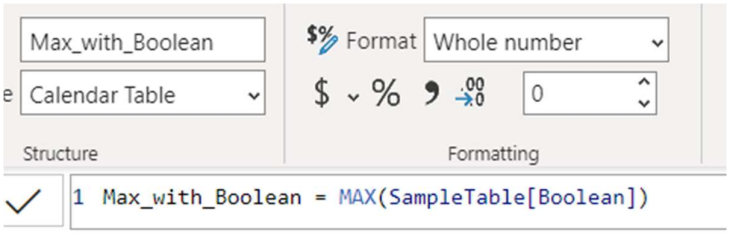


33. Now take Card visual from Visualization pane to Power Bi Page & drag measure over it.
34. For the sales give don't summarize function and set it as descending order as shown below

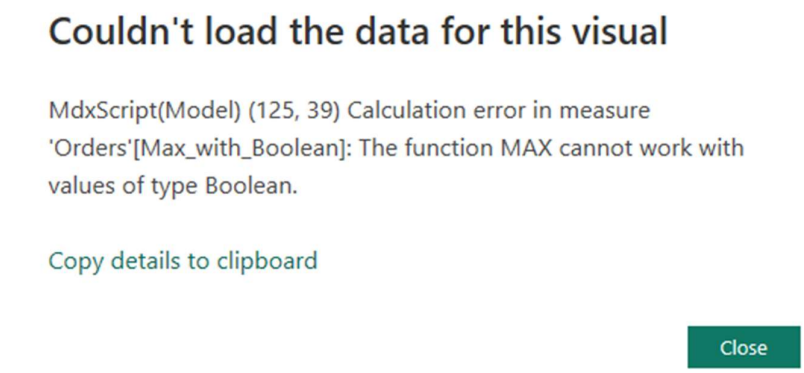
Product Sub-Category	Sales
Binders and Binder Accessories	45,737.33
Office Machines	43,046.20
Office Machines	31,670.60
Chairs & Chairmats	29,718.53
Storage & Organization	27,587.55
Office Machines	20,552.55
Office Furnishings	13,546.94
Office Machines	13,121.07
Copiers and Fax	12,750.99
Storage & Organization	12,599.55
Storage & Organization	12,190.98
Binders and Binder Accessories	11,434.33
Chairs & Chairmats	11,272.77
Copiers and Fax	11,015.82
Binders and Binder Accessories	10,728.00
Chairs & Chairmats	10,554.63
Bookcases	10,364.36
Office Machines	10,331.09

35. It does not support logical value.

DAX: Max_with_Boolean = MAX(SampleTable[Boolean])



36. When you try to load into visual it will throw an error as shown below.



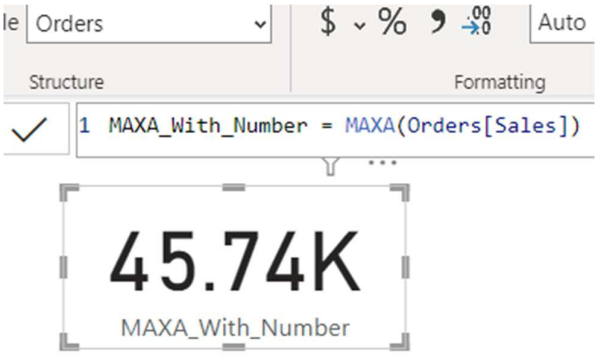
MAXA DAX Function

Returns the largest value in a column.

Syntax: MAXA(<Column>)

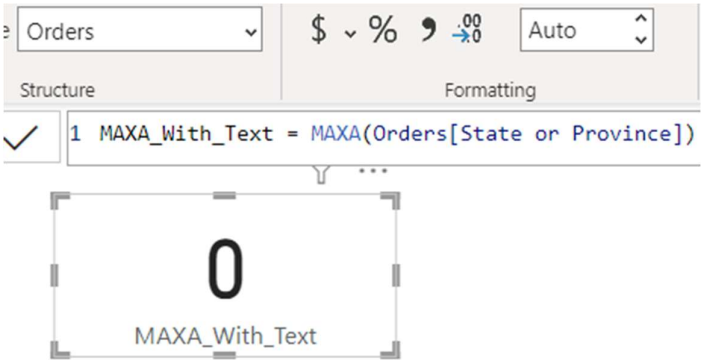
37. MAXA function support Date, Number & Boolean data type.

DAX: MAXA_With_Number = MAXA(Orders[Sales])



38. It does not support Text data type, It will return 0.

DAX: MAXA_With_Text = MAXA(Orders[State or Province])



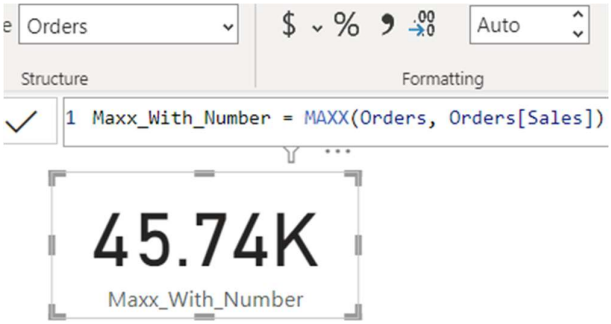
MAXX DAX Function

Evaluates an expression for each row of a table and returns the largest value.

Syntax: MAXX(tablename, expression)

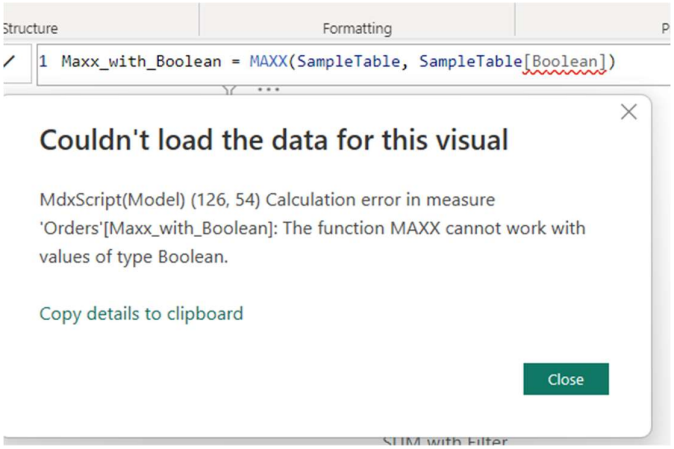
39. MAXX function support Date, Number & Text data type.

DAX: Maxx_With_Number = MAXX(Orders, Orders[Sales])



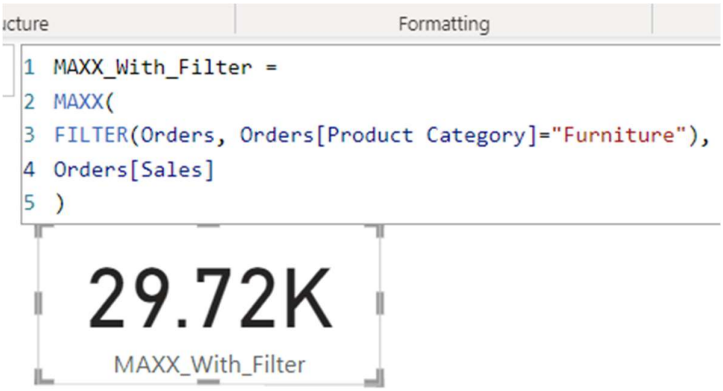
40. It does not support logical value.

DAX: Maxx_with_Boolean = MAXX(SampleTable, SampleTable[Boolean])



41. You can use Filter condition with MAXX Dax function
42. Suppose you want to get MAX sales only from “Furniture” category, so you can use Filter DAX with MAXX.

DAX: MAXX_With_Filter =
MAXX(
FILTER(Orders, Orders[Product Category]="Furniture"),
Orders[Sales]
)



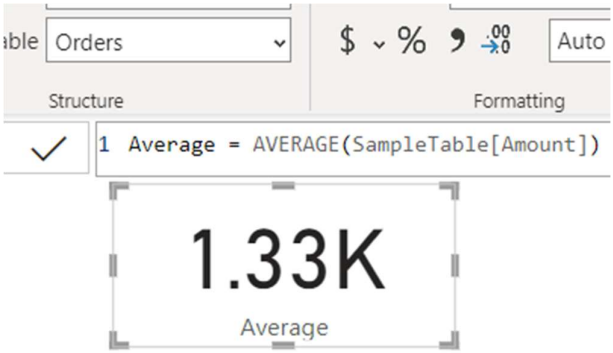
AVERAGE DAX Function

Returns the average (arithmetic mean) of all the numbers in a column.

Syntax: AVERAGE(<Column>)

43. For this example use the Sample Tabel that we used before examples.
44. Let’s create measure for Average function with Amount column.

DAX: Average = AVERAGE(SampleTable[Amount])



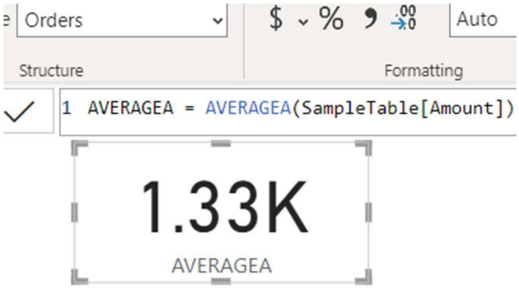
AVERAGEA DAX Function

Returns the average (arithmetic mean) of the values in a column. Handles text and non-numeric values.

Syntax: AVERAGEA(<Column>)

45. Let's create measure for AVERAGEA function with Amount column.

DAX: AVERAGEA = AVERAGEA(SampleTable[Amount])



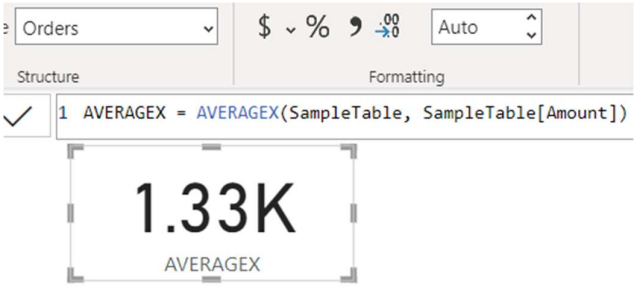
AVERAGEX DAX Function

Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.

Syntax: AVERAGEX(<table>,<expression>)

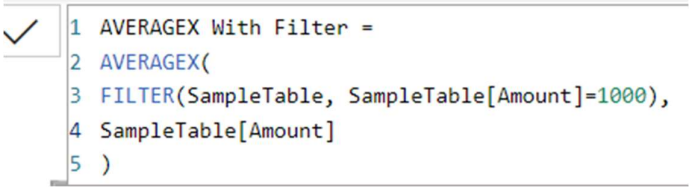
46. Let's create measure for AVERAGEX function with Amount column.

DAX: AVERAGEX = AVERAGEX(SampleTable, SampleTable[Amount])



47. You can use Filter DAX function with AVERAGEX.
48. Suppose you want see Average of Amount column values where amount values are equal to 1000.

DAX: AVERAGEX With Filter =
AVERAGEX(
FILTER(SampleTable, SampleTable[Amount]=1000),
SampleTable[Amount]
)



49. Create a card to check the output as shown below.

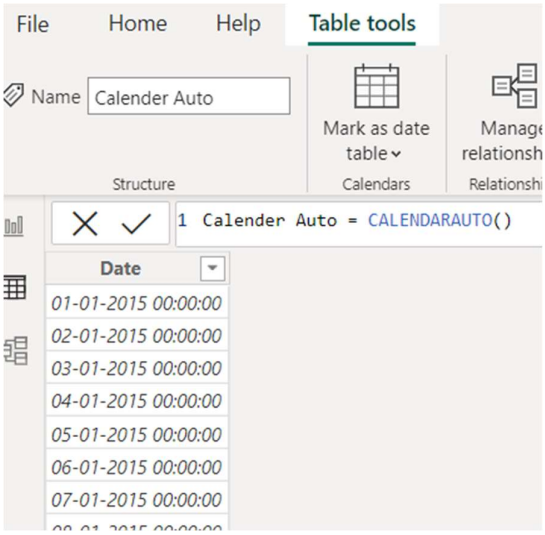
Date & Time DAX Functions

CALENDARAUTO function

Returns a table with a single column named "Date" that contains a contiguous set of dates.

1. Go to modelling and click on Table to create a table. And then type a Dax as shown below.

DAX: Calender Auto = CALENDARAUTO()



CALENDAR function

Returns a table with a single column named "Date" that contains a contiguous set of dates.

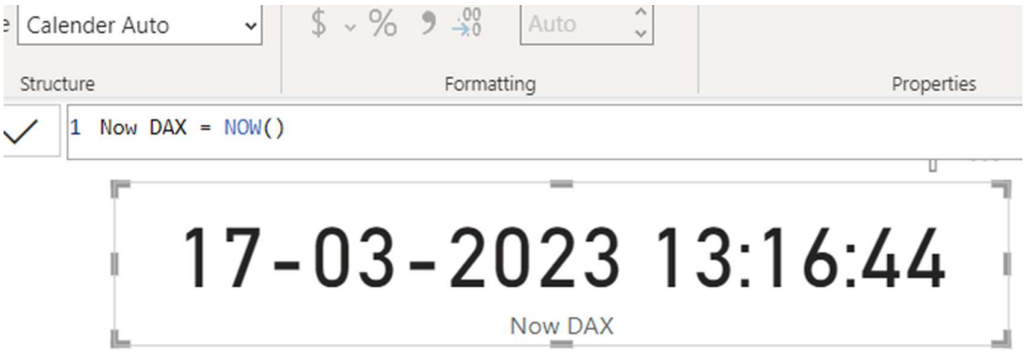
DAX: Calender = CALENDAR(DATE(2015,01,01), DATE(2015,12,31))

NOW function

Returns the current date and time in datetime format.

2. Create a measure for NOW function as shown below.

DAX: Now DAX = NOW()

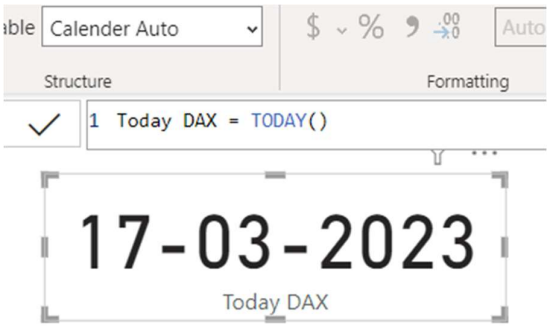


TODAY function

Returns the current date.

3. Create a measure for TODAY function as shown below.

DAX: Today DAX = TODAY()

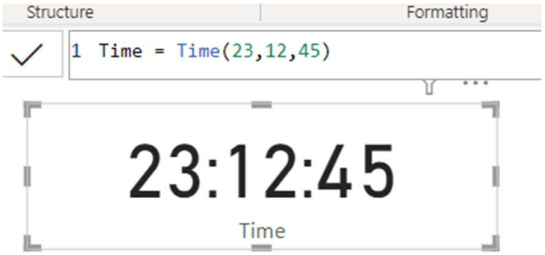


TIME function

Converts hours, minutes, and seconds given as numbers to a time in datetime format.

4. Create a measure for Time function as shown below.

DAX: Time = Time(23,12,45)

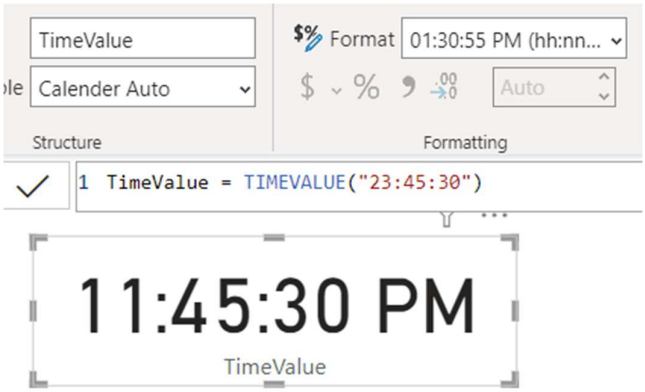


TIMEVALUE function

Converts a time in text format to a time in datetime format.

5. Create a measure for TimeValue function as shown below.

DAX: TimeValue = TIMEVALUE("23:45:30")

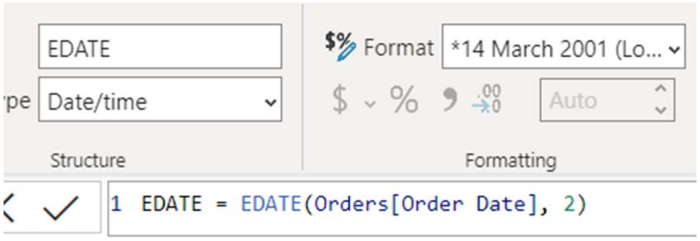


EDATE function

Returns the date that is the indicated number of months before or after the start date.

6. Create a calculated column in the Orders table for EDATE function as shown below.

DAX: EDATE = EDATE(Orders[Order Date], 2)



7. Create a table with the following fields as shown below.

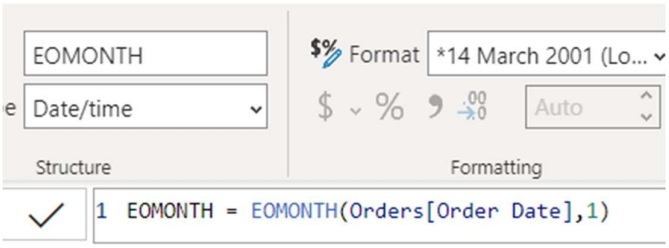
Product Sub-Category	Order Date	EDATE
Appliances	02 January 2015	02 March 2015
Appliances	04 January 2015	04 March 2015
Appliances	06 January 2015	06 March 2015
Appliances	08 January 2015	08 March 2015
Appliances	11 January 2015	11 March 2015
Appliances	13 January 2015	13 March 2015
Appliances	14 January 2015	14 March 2015
Appliances	17 January 2015	17 March 2015
Appliances	19 January 2015	19 March 2015
Appliances	20 January 2015	20 March 2015
Appliances	24 January 2015	24 March 2015
Appliances	25 January 2015	25 March 2015
Appliances	26 January 2015	26 March 2015
Appliances	27 January 2015	27 March 2015
Appliances	28 January 2015	28 March 2015
Appliances	04 February 2015	04 April 2015
Appliances	05 February 2015	05 April 2015
Appliances	06 February 2015	06 April 2015

EOMONTH function

Returns the date in datetime format of the last day of the month, before or after a specified number of months.

8. Create a calculated column in the Orders table for EOMONTH function as shown below.

DAX: EOMONTH = EOMONTH(Orders[Order Date],1)



9. Create a table with the following fields as shown below.

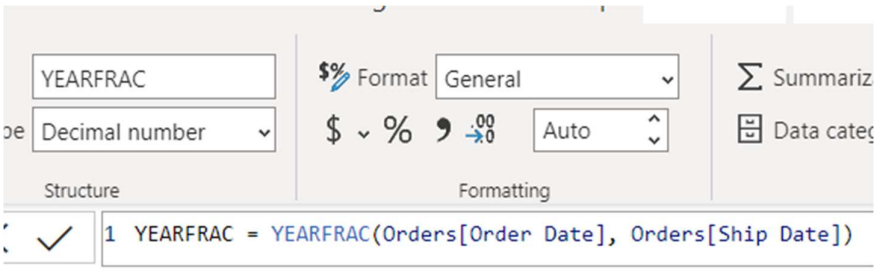
Product Sub-Category	Order Date	EOMONTH
Appliances	02 January 2015	28 February 2015
Appliances	04 January 2015	28 February 2015
Appliances	06 January 2015	28 February 2015
Appliances	08 January 2015	28 February 2015
Appliances	11 January 2015	28 February 2015
Appliances	13 January 2015	28 February 2015
Appliances	14 January 2015	28 February 2015
Appliances	17 January 2015	28 February 2015
Appliances	19 January 2015	28 February 2015
Appliances	20 January 2015	28 February 2015
Appliances	24 January 2015	28 February 2015
Appliances	25 January 2015	28 February 2015
Appliances	26 January 2015	28 February 2015
Appliances	27 January 2015	28 February 2015
Appliances	28 January 2015	28 February 2015
Appliances	04 February 2015	31 March 2015
Appliances	05 February 2015	31 March 2015
Appliances	06 February 2015	31 March 2015

YEARFRAC function

Calculates the fraction of the year represented by the number of whole days between two dates.

10. Create a calculated column in the Orders table for YEARFRAC function as shown below.

DAX: YEARFRAC = YEARFRAC(Orders[Order Date], Orders[Ship Date])



11. Create a table with the following fields to check the output as shown below.

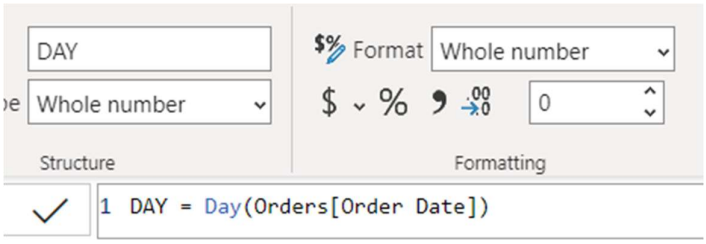
Product Sub-Category	Order Date	Ship Date	YEARFRAC
Appliances	02 January 2015	04 January 2015	0.01
Appliances	04 January 2015	06 January 2015	0.01
Appliances	06 January 2015	08 January 2015	0.01
Appliances	08 January 2015	09 January 2015	0.00
Appliances	11 January 2015	12 January 2015	0.00
Appliances	13 January 2015	17 January 2015	0.01
Appliances	14 January 2015	14 January 2015	0.00
Appliances	14 January 2015	16 January 2015	0.01
Appliances	17 January 2015	19 January 2015	0.01
Appliances	17 January 2015	20 January 2015	0.01
Appliances	19 January 2015	21 January 2015	0.01
Appliances	20 January 2015	21 January 2015	0.00
Appliances	24 January 2015	24 January 2015	0.00
Appliances	24 January 2015	26 January 2015	0.01
Appliances	24 January 2015	29 January 2015	0.01
Appliances	25 January 2015	26 January 2015	0.00
Appliances	26 January 2015	29 January 2015	0.01
Appliances	27 January 2015	28 January 2015	0.00

DAY function

Returns the day of the month, a number from 1 to 31.

12. Create a calculated column in the Orders table for DAY function as shown below.

DAX: DAY = Day(Orders[Order Date])

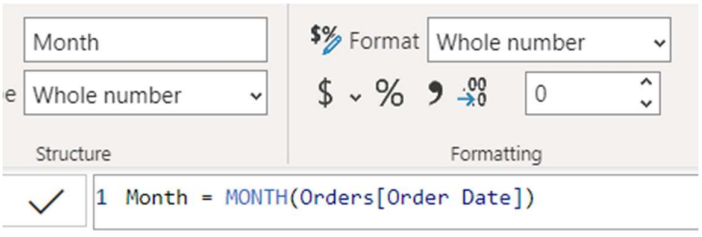


Month function

Returns the month as a number from 1 (January) to 12 (December).

13. Create a calculated column in the Orders table for Month function as shown below.

DAX: Month = MONTH(Orders[Order Date])

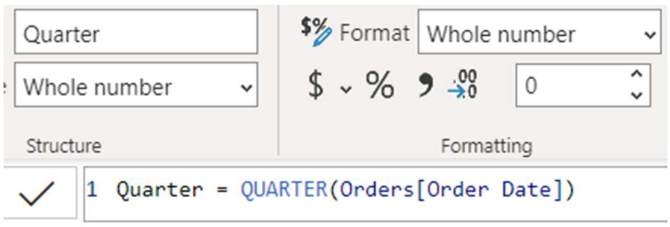


QUARTER function

Returns the quarter as a number from 1 to 4.

14. Create a calculated column in the Orders table for Quarter function as shown below.

DAX: Quarter = QUARTER(Orders[Order Date])

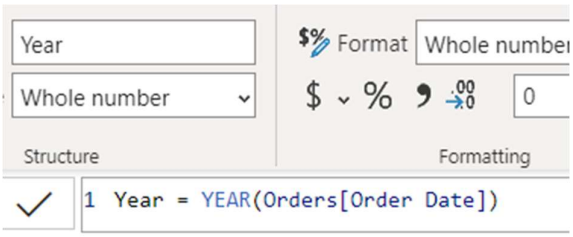


Year function

Returns the year of a date as a four-digit integer in the range 1900-9999.

15. Create a calculated column in the Orders table for YEAR function as shown below.

DAX: Year = YEAR(Orders[Order Date])

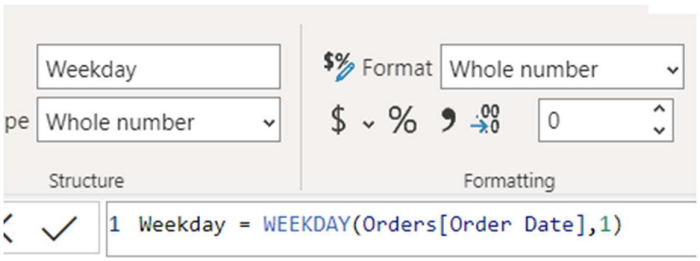


WEEKDAY function

Returns a number from 1 to 7 identifying the day of the week of a date.

16. Create a calculated column in the Orders table for WEEKDAY function as shown below.

DAX: Weekday = WEEKDAY(Orders[Order Date],1)

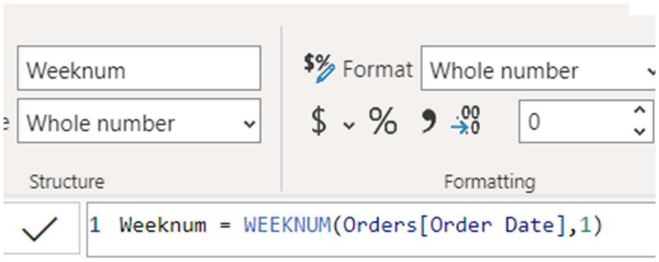


WEEKNUM function

Returns the week number for the given date and year according to the return_type value.

17. Create a calculated column in the Orders table for WEEKNUM function as shown below.

DAX: Weeknum = WEEKNUM(Orders[Order Date],1)



18. Create a table with the following fields to check the output as shown below.

Product Sub-Category	Order Date	DAY	Month	Quarter	Year	Weekday	Weeknum
Appliances	02 January 2015	2	1	Qtr 1	2015	6	1
Appliances	04 January 2015	4	1	Qtr 1	2015	1	2
Appliances	06 January 2015	6	1	Qtr 1	2015	3	2
Appliances	08 January 2015	8	1	Qtr 1	2015	5	2
Appliances	11 January 2015	11	1	Qtr 1	2015	1	3
Appliances	13 January 2015	13	1	Qtr 1	2015	3	3
Appliances	14 January 2015	14	1	Qtr 1	2015	4	3
Appliances	17 January 2015	17	1	Qtr 1	2015	7	3
Appliances	19 January 2015	19	1	Qtr 1	2015	2	4
Appliances	20 January 2015	20	1	Qtr 1	2015	3	4
Appliances	24 January 2015	24	1	Qtr 1	2015	7	4
Appliances	25 January 2015	25	1	Qtr 1	2015	1	5
Appliances	26 January 2015	26	1	Qtr 1	2015	2	5
Appliances	27 January 2015	27	1	Qtr 1	2015	3	5
Appliances	28 January 2015	28	1	Qtr 1	2015	4	5
Appliances	04 February 2015	4	2	Qtr 2	2015	4	6
Appliances	05 February 2015	5	2	Qtr 2	2015	5	6
Appliances	06 February 2015	6	2	Qtr 2	2015	6	6