**DAX in Power BI**

DAX stands for **Data Analysis Expressions** i.e. such expressions or formulas that are used for data analysis and calculations. These expressions are a collection and combination of *functions, operators, and constants* that are evaluated as one formula to yield results (value or values).

**What is power bi Dax?**

Data Analysis Expressions (**DAX**) is a library of functions and operators that can be combined to build formulas and expressions in **Power BI** Desktop, Azure Analysis Services, SQL Server Analysis Services, and **Power Pivot** in Excel.

How do I write a DAX query in power bi?

In **Power BI**, **DAX** calculated fields are called measures. So, we refer to them as measures in this guide.  
**You can write a DAX measure in Power BI as follows:**

1. Right click on the table in the fields list (#1). g. Sales table.
2. Select New measure.
3. Write the measure in the formula bar (#2).
4. Click the tick

Can you do calculations in power bi?

Create calculated measures. A measure is a **calculation** that exists in your **Power BI**data model. To create a measure, in Report view select New Measure from the Modeling tab. ... With DAX **you can** define a measure of time once, and then slice it by as many different fields as **you** want from your data model.

What is calculate function in power bi?

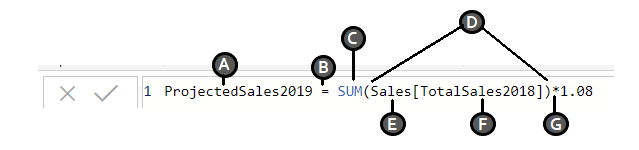
**Calculate** is one of the most versatile **functions in Power BI**. When you begin using anything from simple filters, time intelligence **functions** or even advanced formulas, often the **CALCULATE** formulas are leveraged to produce the desired outcome. Let 's use **CALCULATE** to filer a column in a table.

Is Dax a language?

DAX is not a programming language. DAX is primarily a formula language and is also a query language. You can use DAX to define custom calculations for **Calculated**Columns and for **Calculated** Fields (measures) in Analysis Services **Tabular** Mode

### DAX Formula – Syntax

The initial and most crucial step in learning any language is to break it down into definitive elements and understand its elements. And, that is why we study the syntax of a language. Given below is an example of the DAX formula. We will understand this formula and its syntax elements with the help of this example.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/07/DAX-syntax.png)

We have explained each syntax element labeled in the image in the points below:

A: It is the**name** of the new measure (Projected Sales 2019).

B: It is the equals sign (=) which is an operator indicating the **start of the DAX formula** and equating the two sides.

C: It is the DAX function used to**add the values** of a given field (Total Sales 2018) from a table (Sales). The function used here is **SUM.**

D: The parenthesis () is used to **enclose and define arguments** in an expression. Every function must have at least one argument.

E: It is the**name of the table** from which a field or column is taken in the formula (Sales).

F: It is the **name of the field** from which the formula will use the values. For instance, the function SUM will apply itself on the values of the column or field [Total Sales 2018] of the table Sales.

G: It is**another operator** used for multiplication. Although, the syntax elements from A to F constitutes the basic syntax of DAX.

Thus, in simpler words, this DAX formula commands the system to calculate the product of sum of the values in Total Sales 2018 and 1.08 (8% increase) and store the value in a new field or column known as Projected Sales 2019

#### DAX Calculation Types

So, apparently, the DAX formulas can also be called as calculations as they calculate an input value and return a resultant value. You can create two types of expressions or calculations using DAX in Power BI; calculated columns and calculated measures.

* **Calculated Columns:** The calculated columns create a new column in your existing table. The only difference between a regular column and a calculated column is that it is necessary to have at least one function in the calculated column. These are used when you want to create a column with filtered or sorted information.

To create a calculated column:

1. Go to the **Modeling** tab in Power BI Desktop.
2. Then select**New Column** option. A Formula bar will open showing “Column =”. You can replace the “Column” word with the column name you want.
3. After this, enter the expression for the calculated column on the right of the equals to sign.

* **Calculated Measures:** A calculated measure creates a field having aggregated values such a sum, ratios, percentages, averages, etc.

To create a calculated measure:

1. Go to the **Modeling** tab in Power BI Desktop.
2. Then select **New Measure** option. A Formula bar will open showing “Measure =”. You can replace the “Measure” word with the measure name you want.
3. After this, enter the expression for the calculated measure on the right of the equals to sign.
4. Once you create the measure, you can modify your measure name with a calculator icon next to it, under the table name you created the measure in.

### DAX Functions

A DAX function is a predefined formula which performs calculations on values provided to it in arguments. The arguments in a function need to be in a particular order and can be a column reference, numbers, text, constants, another formula or function, or a logical value such as TRUE or FALSE. Every function performs a particular operation on the values enclosed in an argument. You can use more than one argument in a DAX formula.

##### Key Points about DAX Functions

Here are some unique facts about DAX functions that you must know in order to understand them better:

* Any DAX function always refers to a complete column/field or a table. It will never refer to individual values. If you want to use the functions on separate values within a column, you need to apply filters in a DAX formula.
* DAX functions provide the flexibility to create a formula that is applied on a row-by-row basis. The calculations or formulas get applied as per the context of the values in each row.
* In some cases, DAX functions return a full table which can be used in other DAX formulas that need a complete set of values. However, you cannot display this table’s contents.
* DAX functions have a category known as time intelligence functions. Such functions are used to calculate time/date ranges and periods.

#### Types of DAX functions

##### 1. Date and Time Functions

The date time functions carry out calculations on the date and time values. The data type of these values is always datetime data type.

* CALENDAR
* CALENDARAUTO
* DATE
* DATEDIFF
* DATEVALUE
* DAY
* EOMONTH
* HOUR
* MINUTE
* MONTH
* NOW
* SECOND
* TIME
* TIMEVALUE
* TODAY
* WEEKDAY
* WEEKNUM
* YEAR
* YEARFRAC

##### **2. Time Intelligence Functions**

The time-intelligence functions are used to evaluate values over a fixed period such as days, weeks, months, quarter, years, etc. You can specify a time period using these functions and compare two scenarios in your report.

* CLOSINGBALANCEMONTH
* CLOSINGBALANCEQUARTER
* CLOSINGBALANCEYEAR
* DATEADD
* DATESBETWEEN
* DATESINPERIOD
* DATESMTD
* DATESQTD
* DATESYTD
* ENDOFMONTH
* ENDOFQUARTER
* ENDOFYEAR
* FIRSTDATE
* FIRSTNONBLANK
* LASTDATE
* NEXTQUARTER
* LASTNONBLANK
* NEXTDAY
* NEXTMONTH

##### **3. Information Functions**

The information functions are used to provide certain information on the data values contained in rows and columns. It evaluates the given condition in a function for the value given and return TRUE or FALSE. For instance, the function ISERROR will return TRUE if the value evaluated contains an error.

* CONTAINS
* CUSTOMDATA
* IN Operator / CONTAINSROW function
* ISBLANK
* ISERROR
* ISEVEN
* ISINSCOPE
* ISLOGICAL
* ISNONTEXT
* ISNUMBER
* ISODD
* ISONORAFTER
* ISTEXT
* LOOKUPVALUE
* USERNAME

##### 4. Logical Functions

The logical functions are used to evaluate an expression or argument logically and return TRUE or FALSE if the condition is met or not.

* AND
* FALSE
* IF
* IFERROR
* IN
* NOT
* OR
* SWITCH
* TRUE

##### 5. Mathematical and Trigonometric Functions

The mathematical and trig functions are used to perform all sorts of mathematical functions on the referred values. Given below, is a list of all the available math and trig DAX functions in Power BI.

* ABS
* ACOS
* ACOSH
* ASIN
* ASINH
* ATAN
* ATANH
* CEILING
* COMBIN
* COMBINA
* COS
* COSH
* CURRENCY
* DEGREES
* DIVIDE
* EVEN
* EXP
* FACT
* FLOOR

##### 6. Statistical Functions

These functions carry out statistical and aggregation functions on data values in a DAX expression in Power BI. The list of available statistical functions is given below.

* ADDCOLUMNS
* APPROXIMATEDISTINCTCOUNT
* AVERAGE
* AVERAGEA
* AVERAGEX
* BETA.DIST
* BETA.INV
* CHISQ.INV
* CHISQ.INV.RT
* CONFIDENCE.NORM
* CONFIDENCE.T
* COUNT
* COUNTA
* COUNTAX
* COUNTBLANK
* COUNTROWS
* COUNTX
* CROSSJOIN
* DISTINCTCOUNT
* EXPON.DIST
* GENERATE
* GENERATEALL
* GEOMEAN
* GEOMEANX
* MAX
* MAXA
* MAXX
* MEDIAN
* MEDIANX
* MIN
* MINA
* MINX

##### 7. Text Functions

The text functions in Power BI are very similar to the string functions of Excel. These functions evaluate string values.

* BLANK
* CODE
* COMBINEVALUES
* CONCATENATE
* CONCATENATEX
* EXACT
* FIND
* FIXED
* FORMAT
* LEFT
* LEN
* LOWER
* MID
* REPLACE
* REPT
* RIGHT
* SEARCH
* SUBSTITUTE
* TRIM
* UNICHAR
* UPPER
* VALUE

##### 8. Parent-Child functions

The parent and child functions are used for data values that are a part of a parent-child hierarchy.

* PATH
* PATHCONTAINS
* PATHITEM
* PATHITEMREVERSE
* PATHLENGTH

##### 9. Other functions

There a bunch of functions that do not fit in any particular category. These are also very useful functions.

* DATATABLE
* ERROR
* EXCEPT
* GENERATESERIES
* GROUPBY
* INTERSECT
* ISEMPTY
* ISSELECTEDMEASURE
* NATURALINNERJOIN
* NATURALLEFTOUTERJOIN
* SELECTEDSMEASURE
* SELECTEDMEASUREFORMATSTRING
* SELECTEDSMEASURENAME
* SUMMARIZECOLUMNS
* Table Constructor
* TREATAS
* UNION
* VAR

##### 10. Table functions

The table functions in DAX formulas for Power BI are used to apply operations and conditions on entire tables. The output of table functions is used as inputs in other expressions or arguments in a DAX formula. The results of these functions retain the relationships between columns of that table.

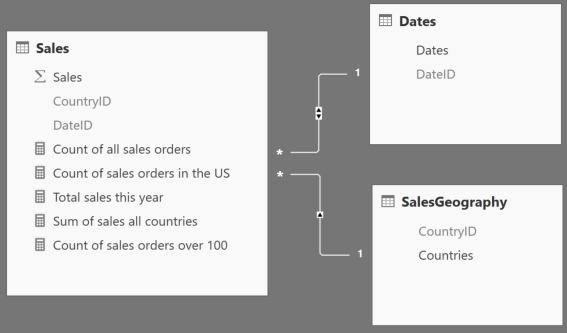
* FILTER
* ALL
* VALUES
* DISTINCT
* RELATEDTABLE

#### DAX Context

The concept of DAX context is essential in gaining a complete understanding of DAX language and how it works in Power BI. There are two types of DAX contexts; Row context and Filter context.

* **Row context:** Row context means taking into account a specific row that has been filtered in a DAX expression. The row context focuses on the operation of the formula on the current row. This type of context is most often applied to the measure rows.
* **Filter context:** The filter context takes focusing on values one step ahead. In row context, we were filtering out and applying operations on specific rows. But in filter context, the expression applies a filter to focus on specific values within a row. Thus, the filter context is applied in addition to the row context to narrow down the scope of calculation to specific values. Filter context is applied when we use functions like CALCULATE, FILTER, RELATED, ALL, etc.

|  |  |  |
| --- | --- | --- |
| 2.png | 3.png | 4.png |

**Relationships View**  


## 1. FILTER

The FILTER function is used to return a subset of a table or expression, as shown below.

**FILTER(<table>,<filter>)**

Let’s say that you want to get a count of items sold at the premium level, which you define as anything over $100. We will use the COUNTROWS function, which counts the number of rows in the specified table, along with the FILTER function to accomplish this:

**Count of sales orders over 100 = COUNTROWS(FILTER('Sales', 'Sales'[Sales] > 100))**

The first parameter, 'Sales', identifies a table or an expression that results in a table. The second parameter, 'Sales'[Sales] > 100, represents a Boolean, or true/false expression that is evaluated for each row of the table. In this expression, we are passing the Sales table to the FILTER function and asking it to return any sales that are over $100. The FILTER function is never used as a standalone function, but is used in conjunction with other functions. In the example above, we use the FILTER function to return a subset and then count the results.

## 2. ALL

The ALL function is used to return all of the rows in a table, or values in a column, ignoring any filters that may have been applied.

**ALL(<table> or <column>)**

In the Report View above, we have a report with multiple cards and a page-level filter that excludes sales in Germany. We would like to keep this filter, but add a card visual that shows the total number of items sold, ignoring any filters placed on the rest of the report. The following expression that incorporates the ALL function can help you to achieve this:

**Count of all sales orders = COUNTROWS(ALL('Sales'))**

In this example, we pass the 'Sales' table to the ALL function, asking it to clear any filters that may have been placed on it. Like the FILTER function, the ALL function is not used standalone but in conjunction with other functions. In this case, we use the ALL function in conjunction with the COUNTROWS function to get a count of all sales records. The ALL function accepts either a table or a column and clears any filters that may have been placed on them.

## 3. RELATED

The RELATED function returns a related value from another table (example shown below).

**RELATED(<column>)**

So far, we’ve worked with functions that can help you to return a subset or clear any filters on a table or column. We would now like to filter our sales for only the United States, but don’t have all of the data we need in one table to accomplish this. Fortunately, we have the RELATED function, which we can use to retrieve values from one table to another through an established relationship. Given that there is a many-to-one relationship between the Sales table and the SalesGeography table, respectively, we can use the following expression that incorporates the RELATED function to return a count of sales orders for only the United States:

**Count of sales orders in the US = COUNTROWS(FILTER(ALL('Sales'), RELATED('SalesGeography'[Countries]) = "United States"))**

## 4. TOTALYTD ****/****TOTALQTD ****/****TOTALMTD

Time intelligence functions in DAX enable you to manipulate data using time periods, including days, months, quarters, and years, and then build and compare calculations over those periods.

**TOTALYTD(<expression>,<dates>[,<filter>][,<year\_end\_date>])**

Continuing from the examples above, let’s say that you would like to see the total sales to date for this year. The following expression that incorporates the TOTALYTD function can enable you to easily do this:

**Total sales this year = TOTALYTD(SUM('Sales'[Sales]), 'Dates'[Dates])**

The first parameter, 'Sales'[Sales], identifies the column that you would like to aggregate. This could also be an expression that returns a scalar, or singular value. The second parameter, 'Date'[Dates], is a column that contains dates. Time intelligence functions are immensely useful functions that eliminate the need for complex code in calculating aggregations over commonly used periods of time.

## 5. CALCULATE

The CALCULATE function evaluates an expression in a context that is modified by specific filters.

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

Let’s say you are now interested in tabulating all sales for all areas. While you could create some piecemeal expressions to accomplish this, you can easily and cleanly accomplish the same thing utilizing the CALCULATE function. The following example, which uses the CALCULATE function, can accomplish this:

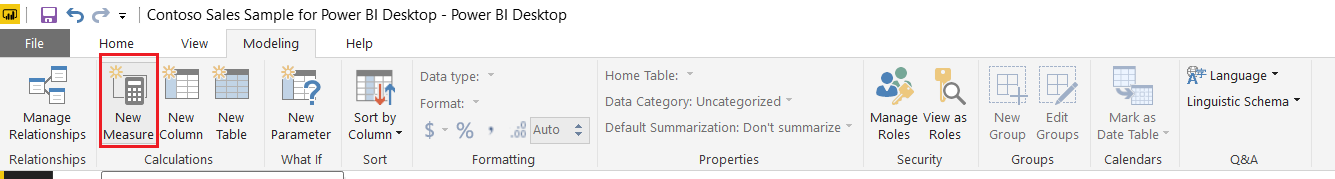
**Sum of sales all countries = CALCULATE(SUM('Sales'[Sales]),ALL('SalesGeography'))**

The first parameter, SUM('Sales'[Sales]), identifies the column that you would like to aggregate. The second parameter, ALL('SalesGeography'), represents a Boolean that removes any filters that may have been placed on the SalesGeography table. Notice that this ignores the page-level filter that excludes sales in Germany. The CALCULATE function is one of the most powerful and useful functions in DAX. It is helpful to think of the CALCULATE function as a supercharged “IF” statement. A couple of rules apply to the CALCULATE function: The filter parameters cannot reference measures, and expressions cannot use any functions that scan or return a table. The CALCULATE function is typically used with aggregation functions, and although the filter parameters are optional, at least one is typically used.

### Creating a Measure Formula using DAX

**Step 1:** To create a measure using the DAX formula, open your Power BI Desktop app. In the Power BI Desktop, you can create a new measure in two ways. One, right-click on the table’s name in which you wish to add the measure field and select **New measure**.

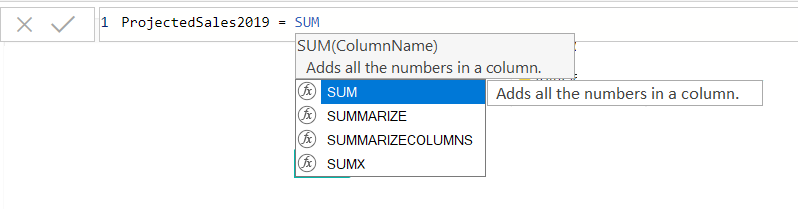
Or, you can go to the **Modeling** tab and click on New measure option.



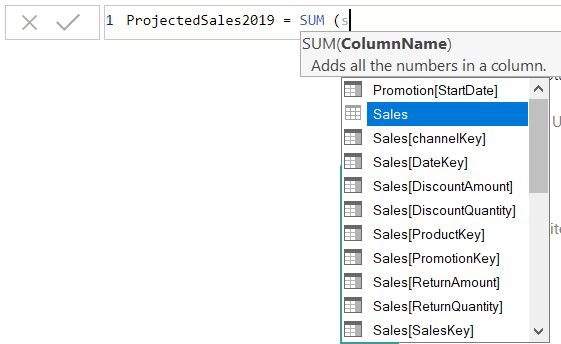
**Step 2:** In the formula bar, start with entering the name of your new measure. In the supposed scenario, we are using the name ProjectedSales2019.

[Power BI DAX Basics - enter name](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/07/enter-name.png)

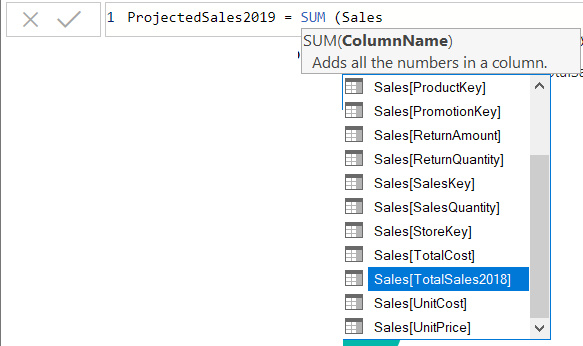
**Step 3:**Next, enter the name of the function that you want to use. Start typing the name of the function and suggestions will appear in the list. A small section showing a description of the function you selected will also appear. Notice in the image below where we have selected the SUM function.



**Step 4:**Then, put a parenthesis “()” and write the name of the table or column you are referring to.



**Step 5:**You can also specify a column within that table. The function will apply to the values within the selected column. Close the parenthesis.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/07/select-column.png)

**Step 6:**Further, you can add more operators, other functions, constants to increase the complexity of the formula.

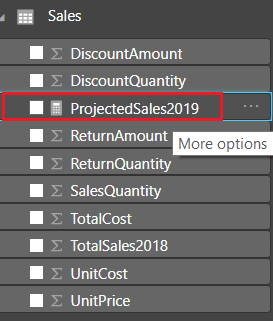
[enter operator - Power BI DAX Basics](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/07/enter-operator.png)

The DAX formula that we created will create a new measure named “ProjectedSales2019”.

ProjectedSales2019 = SUM(Sales[TotalSales2018])\* 1.08

Here, we have multiplied the sum of previous year’s (2018) sales by 1.08 because an increase in sales by 8% is expected in the year 2019. This will give us the projected sales for the year 2019.

**Step 7:**Save this and your newly created measure will appear in the **Fields** column under the table Sales. Every measure field created like this has a calculator icon beside it.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/07/new-measure.png)

Rank

How to calculate top 10 highest in the table and also ranking?

‎You Need to create 3 measure in order to get what you wanted

Salary Measure

Salary\_m = SUM(Ranking[Salary])

Salary Rank

SalaryRank = RANKX(ALL(Ranking),[Salary\_m],,DESC,Skip)

TotalSalaryTop10Rank

SalaryTop10 = CALCULATE([Salary\_m],FILTER(Ranking,[SalaryRank]<=10))

• Top 5 Salespersons per their contribution to Sales

• Top 10 Products per Region

• Top 3 High Value Customers for a specific Financial Period.

• Top Channels per Revenue in the year 2017 and slice it Quarter wise.

Power BI helps you immediately identify your top values and dynamically generate reports to power you with actionable insights and help you plan your strategy ahead.

Assumption – Let us assume we have 2 tables – Manager and Sales

Step 1 – Create a table named “Ranking Selection” for specifying your Top Ranks

For example, you could create a Ranking Selection Table with values – 3,5 and 10. These values can then be used as a slicer for users to select whether they want to see the Top 3, Top 5 or Top 10 Managers or categories in their reports. Make sure that this table does not have a relationship with any other table.

Step 2 – Create a new Measure for capturing the selections of the above Ranks [based on the Slicer selection]

In this case a measure named Rank Select has been defined.

Rank Select = IF (HASONEVALUE( ‘Ranking Selection'[Rank] ), VALUES(‘Ranking Selection'[Rank]),15 )

Step 3 – Create another set of Measures

Manager Ranking = RANKX (ALL ( Managers[Manager] ), Sales[Total Sales],, DESC )

This Measure is required for returning a table with the list of managers with an assigned Rank based on the Descending order of the Total Sales.

Sales by Manager = IF ([Manager Ranking]<=[Rank Select],[Total Sales],BLANK())

This Measure will help display Dynamic reports/visuals based on the Rank selection made. For instance – by selecting Rank 5 from the list, this measure will generate the Total Sales for the Top 5 Managers per Total Sales.

Step 4 – Create a Slicer based on Rank Column from Rank Selection Table

You can decide whether you would like to view the Ranks as a List or a Drop down. Once you have this slicer on the Report Canvas you can view the Top values based on this selection.

Step 5 – Go ahead and start creating your visuals.

You could start with a simple Bar chart and jazz it up with infographics custom visuals. You could build this visual on Manager Name and Sales by Manager Measure.

Similarly, you could identify the Top Categories of products by sales and link it with the Ranking Measure. If you want to view the Top 5 Managers per sales as well as the Top 5 Categories per sales, you can make a Ranking selection and get equipped with solid intelligence on your best performers.

Remarks

If the data has been filtered, the CALCULATE function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter argument, any existing filters on that column are removed, and the filter used in the filter argument is applied instead.

Example

To calculate the ratio of current reseller sales to all reseller sales, you add to the PivotTable a measure that calculates the sum of sales for the current cell (the numerator), and then divides that sum by the total sales for all resellers (the denominator). To ensure that the denominator remains the same regardless of how the PivotTable might be filtering or grouping the data, the part of the formula that represents the denominator must use the ALL function to clear any filters and create the correct total.

The following table shows the results when the new measure, named All Reseller Sales Ratio, is created by using the formula in the code section.

To see how this works, add the field, CalendarYear, to the Row Labels area of the PivotTable, and add the field, ProductCategoryName, to the Column Labels area. Then add the new measure to the Values area of the PivotTable. To display the numbers as percentages, apply percentage number formatting to the area of the PivotTable that contains the new measure, All Reseller Sales Ratio.

All Reseller Sales Column Labels

Row Labels Accessories Bikes Clothing Components Grand Total

2005 0.02% 9.10% 0.04% 0.75% 9.91%

2006 0.11% 24.71% 0.60% 4.48% 29.90%

2007 0.36% 31.71% 1.07% 6.79% 39.93%

2008 0.20% 16.95% 0.48% 2.63% 20.26%

Grand Total 0.70% 82.47% 2.18% 14.65% 100.00%

DAXCopy

=( SUM('ResellerSales\_USD'[SalesAmount\_USD]))

/CALCULATE( SUM('ResellerSales\_USD'[SalesAmount\_USD])

,ALL('ResellerSales\_USD'))

The CALCULATE expression in the denominator enables the sum expression to include all rows in the calculation. This overrides the implicit filters for CalendarYear and ProductCategoryName that exist for the numerator part of the expression.

Limitations of CALCULATE in DAX

Powerful and all that CALCUATE is, there is a limitation to the filters that it can apply. The filters will only work when

• There is a value on the right hand side of the equation

• There is a column name on the left hand side of the equation

Let’s look at this by way of example

=CALCULATE (

[Total Sales],

[Total Sales] >= 700

)

If we were to enter this expression, our logic would be correct. . We are looking to calculate the total of the sales where the total sales is greater or equal to 700. But the problem with this expression is that we have a measure [Total Sales] on the left hand side of the equation. We are expecting a column name

Solution – Just add FILTER inside CALCULATE

If we now re-write the expression to:

=CALCULATE (

[Total Sales],

Filter(Sales, [total sales] >= 700)

)

First thing CALCULATE will do is evaluate the FILTER expression.

Filter is an iterator and iterates the Sales table. But the filter might not be able to see all of the Sales table. This is because of any original filter context applied by the pivot table or visualisation. So the reference to the table for the filter argument is the portion of the table that is visible under the original filter context.

Let’s say we had a pivot table showing only 2017, First Sales would be filtered to 2017 as per the original filter context. Then FILTER will go over 2017 and check to see row by row if each sale is greater or equal to 700. A table will be returned with the rows that match this criteria.

Running total

Cumulative Sales:=IF(MIN(DimDate[Datekey])<=CALCULATE(MAX(FactSales[DateKey]),ALL(FactSales)),CALCULATE([Total Sales],FILTER(All(DimDate[Datekey]),DimDate[Datekey]<=MAX((DimDate[Datekey])))),BLANK())

Moving Average

Firstly you need a column of date with full date format. Then you can use calculated measure to get the expected result. Please refer to following steps.

1. Create a calculated column for the date.

2. FullDate =

3. DATE ( 2016, 'Session'[Month of the Year], 1 )

4. Create a measure for 3 months moving average. You can change the number of months if you want.

5. Moving\_Average\_3\_Months =

6. CALCULATE (

7. AVERAGEX ( 'Session', 'Session'[Sessions] ),

8. DATESINPERIOD (

9. 'Session'[FullDate],

10. LASTDATE ( 'Session'[FullDate] ),

11. -3,

12. MONTH

13. )

)

Running % measure:

Running % =

CALCULATE (

SUM ( [Percentage] ),

FILTER ( ALL ( YourTable), YourTable[Bucket] <= MAX ( YourTable[Bucket] ) )

)

And try this for Cumulative count measure:

Cumulative Count =

CALCULATE (

SUM ( [Count] ),

FILTER ( ALL ( YourTable ), YourTable[Bucket] <= MAX ( YourTable[Bucket] ) )

)

You can create a cumulative count using CALCULATE function, which lets us calculate the Running % measure:

Cumulative Count =

CALCULATE (

[Event Count],

FILTER (

ALL ( EventTable ),

[Duration\_Bucket] <= MAX ( EventTable[Duration\_Bucket] )

)

)

Now calculate the Running % measure using:

Running % =

DIVIDE (

[Cumulative Count],

CALCULATE ( [Event Count], ALL ( EventTable ) ),

BLANK ()

)