Introducing DAX — Data Analysis Expressions

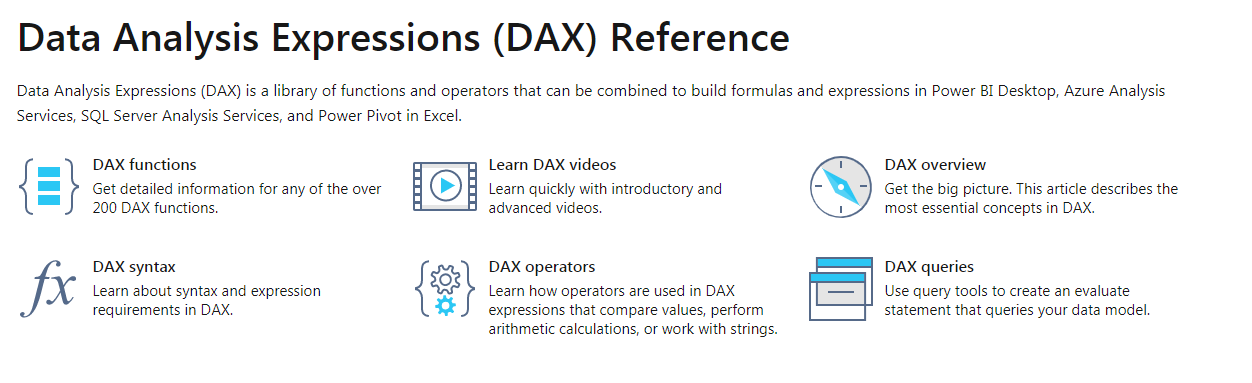
Power BI article

**Power BI as a Tool for Business Intelligence:** <https://towardsdatascience.com/power-bi-as-a-tool-for-business-intelligence-237b478c3b84>

<https://docs.microsoft.com/en-us/visualstudio/products/> - Filter functions

# data science and Power BI

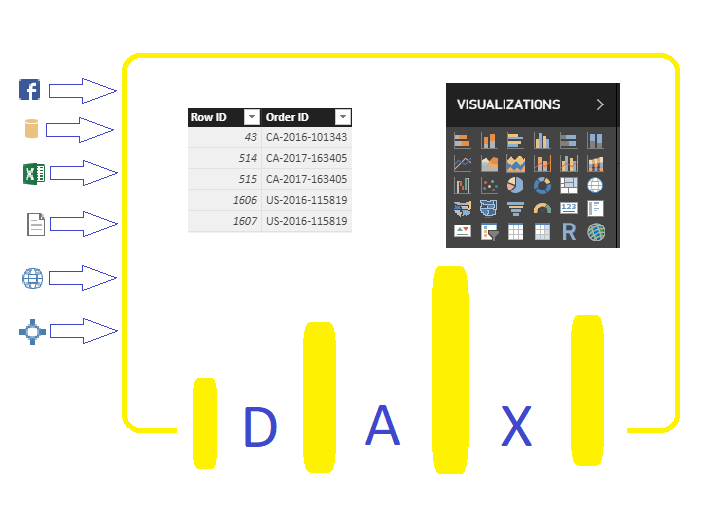
# Data Analysis Expressions (DAX) Reference



<https://towardsdatascience.com/introducing-dax-data-analysis-expressions-dca49cb68501>

**D**ata **A**nalysis E**x**pressions are a collection of functions that can be used to perform a task and return one or more values. Although this sounds very similar to any other programming language, DAX is only a formula or a query language. DAX was developed around 2009 by Microsoft to be used with Microsoft’s PowerPivot, which at that time was available as an Excel (2010) add-in. It is extremely popular today as it is now the language of choice for Power BI and is supported by Tabular SSAS as well. Since it is widely being used in Power BI, I have concentrated on DAX for Power BI in this article but, can be applied to other applicable tools.

Power BI is a Business Intelligence tool that consumes data from hundreds of data sources, preps the data and produces beautiful interactive reports and dashboards that deliver compelling insights. It provides a suite of services to bring the data in, build dashboards or reports and share it with the required folks. Although Power BI is relatively new, its core in a way has been in the making for a long time. It sits on the shoulders of SSAS (SQL Server Analysis Services), Power Pivot, Power View and Power Query — some of which were available as a free Excel add-in.



DAX usability in Power BI Desktop

Within the entire suite of Power BI, DAX is mostly used to transform the data model imported into Power BI and/or to add new measures for visualizations. These functions are used to create Calculated Columns or Measures to help with reporting.

DAX is also used in securing the reports and dashboards that are published by providing Row-Level Security (RLS) at the data model level.

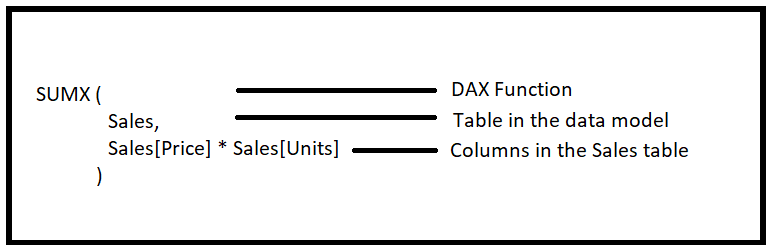
**Calculated Columns**

Calculated columns are created within the data model and are calculated row by row at the column level and are stored within the data model and become part of the table in which it resides. Calculated columns also consume more memory. A calculated column is usually represented as Table[Column].

**Measures**

Measures, as the name suggests are aggregates and DAX helps with creating additional measures based on the data model to help with reporting/visuals. These DAX functions run on the fly, also known as dynamic aggregation and hence are not stored with the data model. Since the measures are calculated at the time of query, they consume additional CPU. A measure is simply represented as [Measure].

**Syntax**

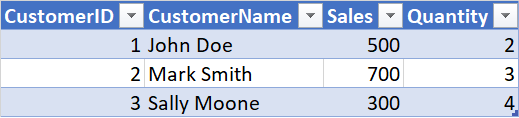


Syntax of a DAX function

**Aggregation Functions**

DAX aggregation functions aggregate a column or expression over rows in a table and are used in Measures. DAX provides a lot of in-built aggregate functions and here’s a [reference list](https://msdn.microsoft.com/en-us/query-bi/dax/statistical-functions-dax).

**Example:**



Customer Table

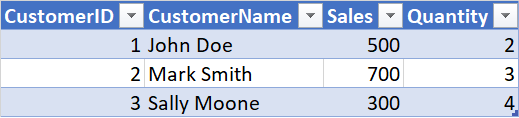
Average Sales=AVERAGE(Customer[Sales])

The Average Sales measure would now return 500.

**‘X’ Aggregation Functions**

X aggregation functions iterate and aggregate over rows in a table, the same as an aggregate function — the only difference being X version of the aggregate function aggregates over an expression instead of column in a table.

**Example:**



Customer Table

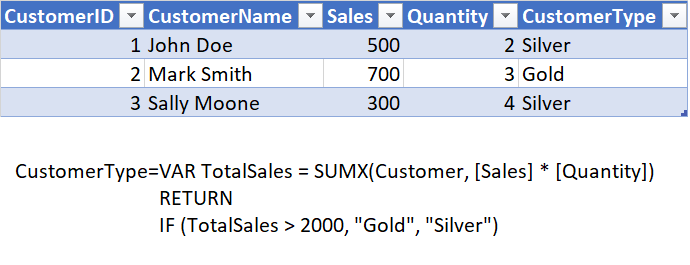
Average Sales=AVERAGEX(Customer, [Sales] \* [Quantity])

The Average Sales measure would now return 1433.34.

**VAR Function**

A named variable stores the result of an expression which can later be passed to other expressions as a parameter. Variables reduce the redundancy in the code and also help in performance by avoiding the need to create a new calculated column.

**Example:**



An example of variable in DAX

**Time Intelligence Functions**

Time Intelligence functions are extremely useful when it comes to Business Intelligence and also saves time in performing complex calculations in simple functions. These functions work on date and time fields to produce a wide array of calculations based on different levels of time period. Here’s a [list](https://msdn.microsoft.com/en-us/query-bi/dax/time-intelligence-functions-dax) of Time Intelligence functions.

**Text Functions**

Text functions work on columns in a table to concatenate, search or manipulate the string and return the whole or a part of a string. Here’s a [list](https://msdn.microsoft.com/en-us/query-bi/dax/text-functions-dax?f=255&MSPPError=-2147217396) of text functions.

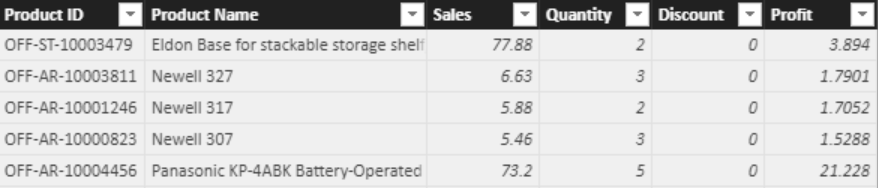
**Table Functions**

Table functions return full tables instead of values. They are typically used in conjunction with other functions to iterate over each row in the result table returned.

Imagine you have a group of related tables and need to perform calculations based on two or more tables. Some of the table functions are most useful in such cases. Here’s a [list](https://msdn.microsoft.com/en-us/query-bi/dax/filter-functions-dax) of table functions.

The function **FILTER**returns a subset of the current table based on the expression provided and returns a table as a result.

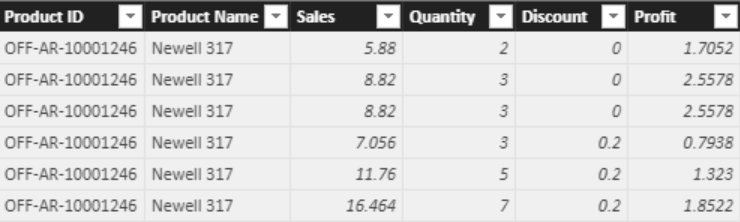
**Example:**



Orders Tables

SuperStore = FILTER (Orders, Orders[Product Name] = “Newell 317”)

This creates a new table by the name SuperStore with just Newell 317 products in them as shown below.



SuperStore Table

The function **RELATEDTABLE**returns all the rows in the table that are related to the current table.

**Example:**



Customer Table

Count of Orders= COUNTROWS(RELATEDTABLE(‘SuperStore’))

**Outro**

While this is just a brief introduction to DAX, there are a ton of other useful functions in DAX that will make the life of an analyst or a developer a lot easier.

Here are some excellent resources to learn DAX

* DAX Training Videos at [SQLBI](https://www.sqlbi.com/training/dax/)
* DAX on [TutorialsPoint](https://www.tutorialspoint.com/dax_functions/index.htm" \t "_blank)
* [Guy in a cube](https://www.youtube.com/results?search_query=guy+in+a+cube+DAX)

## [Towards Data Science](https://towardsdatascience.com/?source=post_sidebar--------------------------post_sidebar-)

#### **Sharing concepts, ideas, and codes.**

[Follow](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fintroducing-dax-data-analysis-expressions-dca49cb68501&source=post_sidebar--------------------------follow_sidebar-)

#### **91**

* [Dax](https://towardsdatascience.com/tag/dax)
* [Power Bi](https://towardsdatascience.com/tag/power-bi)
* [Business Intelligence](https://towardsdatascience.com/tag/business-intelligence)
* [Data Analysis](https://towardsdatascience.com/tag/data-analysis)
* [Excel](https://towardsdatascience.com/tag/excel)

#### **91 claps**

WRITTEN BY

## [Sharad Vijalapuram](https://towardsdatascience.com/@sharadvm?source=follow_footer--------------------------follow_footer-)

[Fllow](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fintroducing-dax-data-analysis-expressions-dca49cb68501&source=follow_footer-a669406c3f54-------------------------follow_footer-)

#### **Business Intelligence and Data Analysis professional with 6+ years of sound experience designing and developing BI solutions and storytelling.**