Introduction to DAX

What is DAX?

DAX stands for **Data Analysis eXpressions**, and it is the formula language used in **Power BI**. **DAX** is also found in other offerings from **Microsoft**, such as **Power Pivot** and **SSAS Tabular**.

DAX is a functional language, which means the full executed code is contained inside a function.

There are two primary calculations you can create using DAX

- Calculated Columns
- Measures

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.

DAX Table and Column Name syntax

Whether you're creating a New Column or Measure, it's important to know the general format of table names in DAX.

'Table Name'[ColumnName]

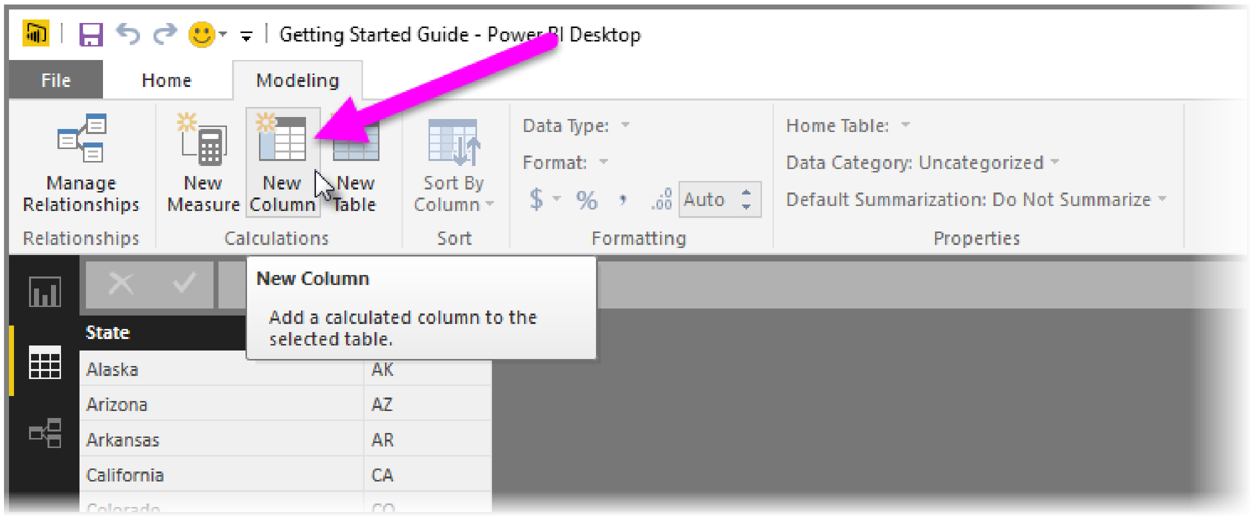
If there are spaces in the table name (as shown above), the single quotes around the table name are mandatory. If the table name has no spaces, the single quotes can be omitted, so the syntax looks like the following. Column names must always include the square brackets.

TableName[ColumnName]

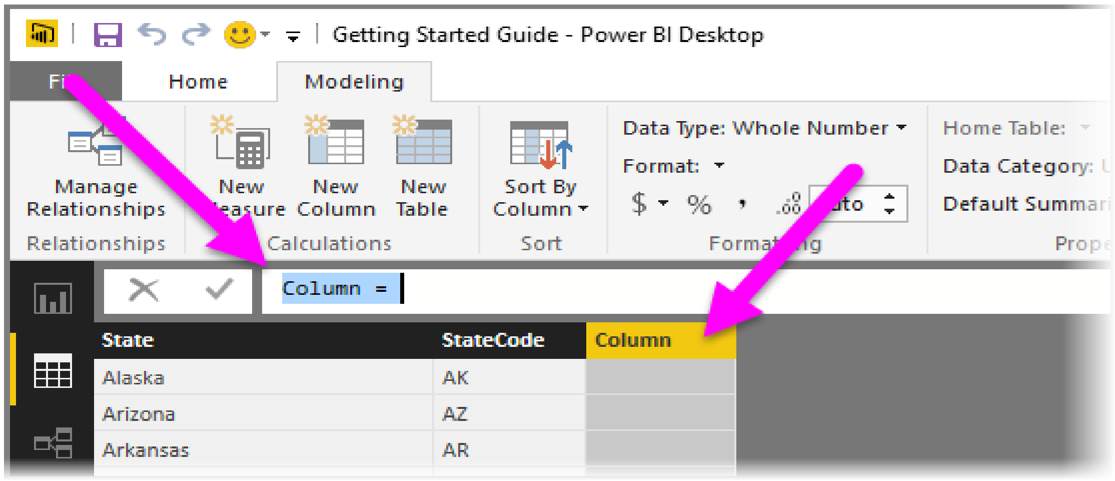
Creating Calculated Columns

Calculated Columns are useful when you want a calculation for every row in your table.

You can create Calculated Columns in Power BI Desktop by selecting New Column from the Modeling tab. It's best to be in Data view (rather than Report or Relationships view), since you can see the **New Column** created and the Formula Bar is populated and ready for your DAX formula.



Once you select the New Column button, the Formula Bar is populated with a basic column name (which you change to suit your formula, of course) and the = operator, and the new column appears in the data grid, as shown in the following image.



The required elements for a calculated column are the following

- A new column name
- At least one function or expression

Calculated Columns Examples

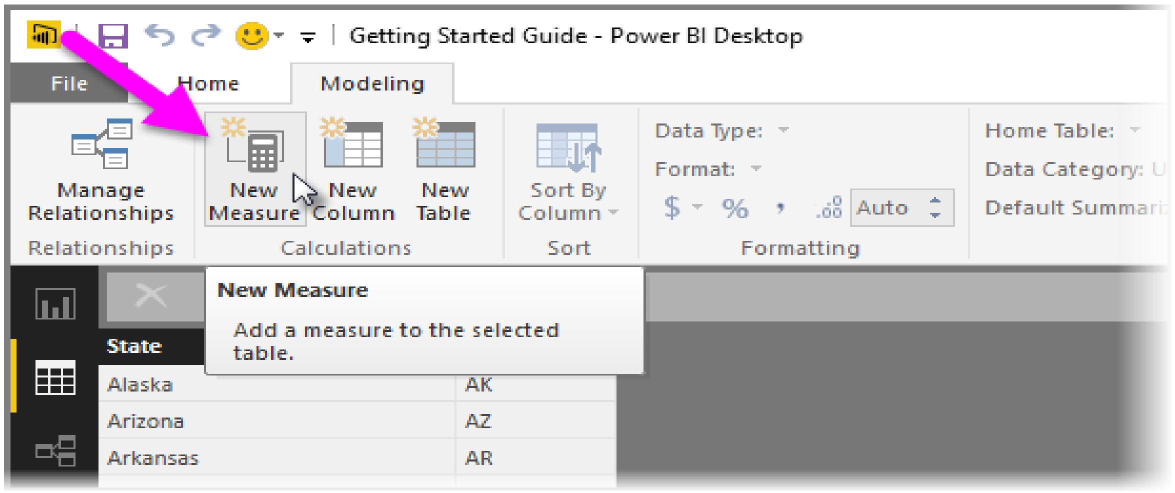
1. Load the Orders table into Power BI

2. Create the following Calculated Columns

a) Unit Price = Sales / Quantity
b) Cost Price = Sale - Profit
C) Quantity Type = IF (Orders [Quantity] = 1, "Single Item", "Multiple Items")
d) State and Country = CONCATENATE (Orders [State], CONCATENATE (", ", Orders [Country]))

Creating Measures

Use a calculated measure when you are calculating percentages or ratios, or you need complex aggregations. To create a measure using a DAX formula, select the New Measure button from the Modeling tab. Again, it's best to be in the Data view of Power BI Desktop since it shows the Formula Bar and makes it easy to write your DAX formula.

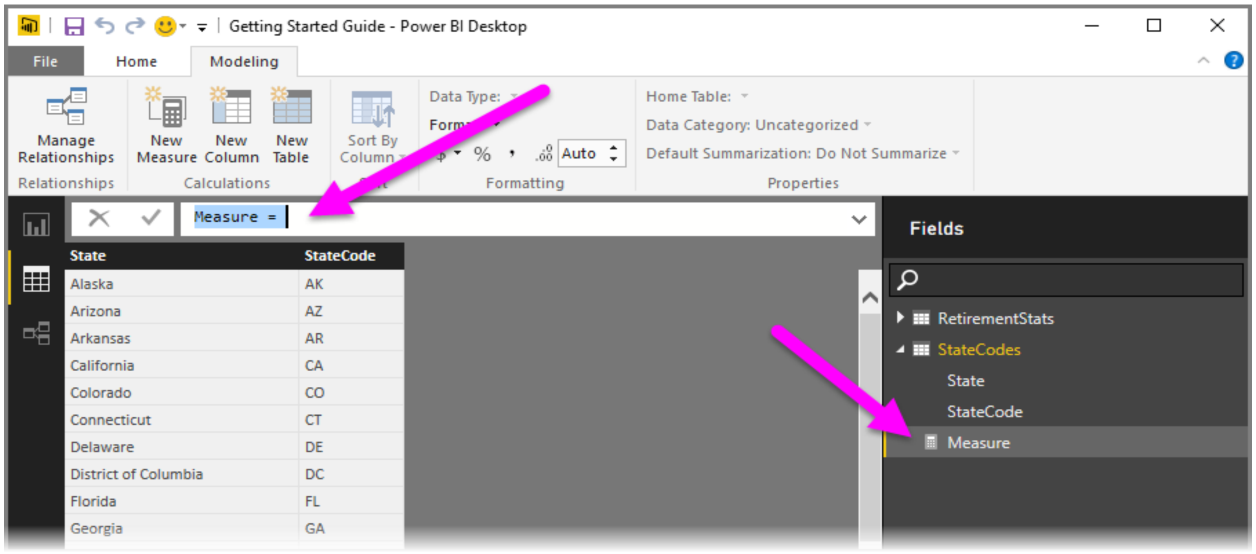


New Measure

Add a measure to the selected table.

State	
Alaska	AK
Arizona	AZ
Arkansas	AR

With measures, you see a new measure icon appear in the Fields pane with the name of the measure. The Formula Bar is again populated with the name of your DAX formula (this time, with your measure).



```
Measure =
```

State	StateCode
Alaska	AK
Arizona	AZ
Arkansas	AR
California	CA
Colorado	CO
Connecticut	CT
Delaware	DE
District of Columbia	DC
Florida	FL
Georgia	GA

The required elements for a calculated measure are the same as they are for a calculated column

- A new measure name
- At least one function or expression

Calculated Measures Examples

- Total Sales = SUM(Orders[Sales])
- Total Profit = SUM(Orders[Profit])
- Average Sales = AVERAGE(Orders[Sales])

Implicit Vs Explicit Measures

Implicit Measures are created when you drag raw numerical fields (like Sales, Profit, Quantity) into the values field well of a visual and manually select the aggregation mode (Sum, Average, Min/Max, etc.).

Explicit measures are created by actually entering DAX functions like below

- Sum Of Sales = SUM(Orders[Sales])

Implicit measures are only accessible within the specific visualization in which it was created, and cannot be referenced elsewhere.

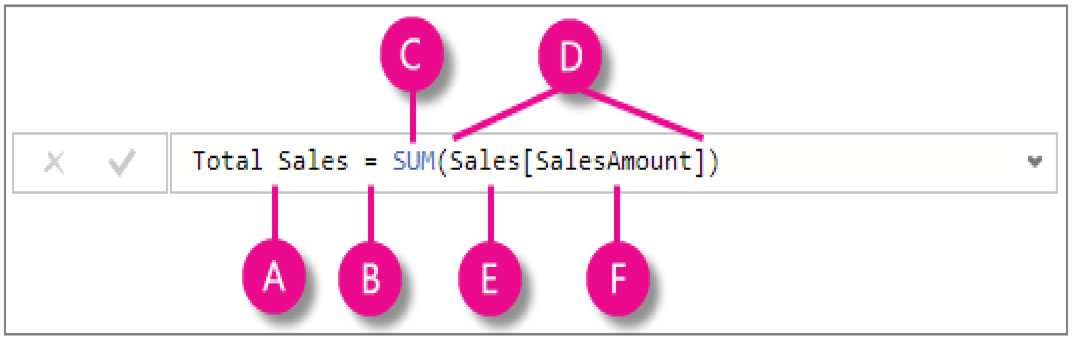
Explicit measures can be used anywhere in the report and referenced with other DAX calculations.

- Sum Of Sales = SUM (Orders[Sales])
- % Sales = **[Sum Of Sales]/**CALCULATE(**[Sum Of Sales],**ALL(Orders))

Calculated Columns Vs Measures

Calculated Columns	Measures
Values are calculated based on information for each row of a Table	Values are calculated based on information from any filters in the report or the fields used in the visualizations
Appends Static values to each row in a table and stores them in the model which increases the file size	Does not create new data in the tables which doesn't increases the file size
Recalculate on data source refresh or when changes are made to component columns which are used to derive Calculated Columns	Recalculate in response to any change to filters within the report or when we add another dimension to the visualization.
Primarily used as rows, columns, slicers or filters in Visualizations	Almost always used within the values field of a visuals
Calculated Column Name Should be unique at Table Level	Measure Name Should be unique thought the Model

DAX Syntax & Operators



```
Total Sales = SUM(Sales[SalesAmount])
```

This formula includes the following syntax elements

A) The measure name Total Sales.

B) The equals sign operator (=) indicates the beginning of the formula. When calculated, it will return a result.

C) The DAX function SUM adds up all of the numbers in the Sales [SalesAmount] column. You'll learn more about functions later.

D) Parenthesis () surround an expression containing one or more arguments. All functions require at least one argument. An argument passes a value to a function.

E) The referenced table Sales.

F) The referenced column [SalesAmount] in the Sales table. With this argument, the SUM function knows on which column to aggregate a SUM.

DAX Operators

The Data Analysis Expression (DAX) language uses operators to create expressions that compare values, perform arithmetic calculations, or work with strings.

Types of Operators

There are four different types of operators → arithmetic, comparison, text concatenation, and logical.

Arithmetic Operators

To perform basic mathematical operations such as addition, subtraction, or multiplication, combine numbers, and produce numeric results, use the following arithmetic operators.

Arithmetic operator	Meaning	Example
+ (plus sign)	Addition	3+3
- (minus sign)	Subtraction or sign	3-1
* (asterisk)	Multiplication	3*3
/ (forward slash)	Division	39/3
^ (caret)	Exponentiation	16^4

Comparison Operators

You can compare two values with the following operators. When two values are compared by using these operators, the result is a logical value, either TRUE or FALSE.

Comparison operator	Meaning	Example
=	Equal to	[Region] = "USA"
>	Greater than	[Sales Date] > "Jan 1 2009"
<	Less than	[Sales Date] < "Jan 1 2009"
>=	Greater than or equal to	[Amount] >= 20000
<=	Less than or equal to	[Amount] <= 100
<>	Not equal to	[Region] <> "USA"

Text Concatenation Operator

Use the ampersand (&) to join, or concatenate, two or more text strings to produce a single piece of text.

Text operator	Meaning	Example
& (ampersand)	Connects, or concatenates, two values to produce one continuous text value	[State] & ", " & [Country]

EX: State and Country = CONCATENATE (Orders [State], CONCATENATE (", ", Orders [Country]))

Or

State and Country = Orders [State] & ", " & Orders [Country]

Logical Operators

Use logical operators (&&) and (||) to combine expressions to produce a single result.

Logical operator	Meaning	Example
&& (double ampersand)	Creates an AND condition between two expressions that each have a Boolean result. If both expressions return TRUE, the combination of the expressions also returns TRUE; otherwise the combination returns FALSE.	([Region] = "France") && ([BikeBuyer] = "yes"))
|| (double pipe symbol)	Creates an OR condition between two logical expressions. If either expression returns TRUE, the result is TRUE; only when both expressions are FALSE is the result FALSE.	(([Region] = "France") || ([BikeBuyer] = "yes"))
IN	Creates a logical OR condition between each row being compared to a table. Note: the table constructor syntax uses curly braces.	'Product'[Color] IN { "Red", "Blue", "Black" }