



DAX for Power BI

Sample manual - first two chapters



Wise Owl
Training

TABLE OF CONTENTS (1 of 4)

1	INTRODUCTION TO DAX IN POWER BI	Page
1.1	DAX in Power BI	6
	<i>How DAX is Used 1 – Calculated Columns</i>	6
	<i>How DAX is Used 2 – Measures</i>	6
	<i>How DAX is Used 3 – Calculated Tables</i>	7
	<i>Choosing the Type of Calculation</i>	7
	<i>Where Else is DAX Used?</i>	7
1.2	The Construct-a-Creature Database	8
	<i>The Tables and Relationships</i>	8
1.3	Getting Help with DAX	9
	<i>Getting Help within Power BI</i>	9
	<i>Other Sources of Help</i>	9

2	BASIC CALCULATED COLUMNS	Page
2.1	Calculated Columns	10
	<i>Key Features of Calculated Columns</i>	10
	<i>Creating a Calculated Column</i>	10
	<i>Entering a Formula</i>	11
	<i>Calculated Column Properties</i>	12
	<i>Using Calculated Columns in Visuals</i>	12
2.2	Referencing Columns and Tables	13
	<i>Qualifying Column Names</i>	13
	<i>Table Names</i>	13
	<i>Referencing Columns in Other Tables</i>	14
	<i>The Related Function</i>	14
2.3	Editing DAX Code	15
	<i>Multiple Lines and Indenting</i>	15
	<i>Comments</i>	16
	<i>Keyboard Shortcuts</i>	16

3	WORKING WITH DATA TYPES	Page
3.1	DAX Data Types	17
	<i>Viewing a Column's Data Type</i>	17
3.2	Working with Numbers	18
	<i>Basic Arithmetic</i>	18
	<i>Controlling the Calculation Order</i>	18
	<i>Safely Dividing Numbers</i>	19
	<i>Numeric Functions</i>	20
3.3	Working with Text	21
	<i>Writing Text in Calculations</i>	21
	<i>Concatenating Text</i>	21
3.4	Text Functions	22
	<i>Finding and Extracting Text</i>	22
	<i>Replacing Text</i>	22
	<i>Generating Text</i>	23
	<i>Converting and Formatting Text</i>	23
3.5	Working with Dates	24
	<i>Entering Date and Time Values</i>	24
	<i>Returning the Current Date and Time</i>	24
	<i>Calculating Date and Time Values</i>	25
	<i>Calculating the Difference Between Dates</i>	25
	<i>Extracting Date Parts</i>	26
	<i>Formatting Dates</i>	26

4	CONDITIONAL FUNCTIONS	Page
4.1	The IF Function	27
	<i>Testing a Single Condition</i>	27
	<i>Comparison Operators</i>	27
	<i>The IN Operator</i>	27
	<i>Nesting IF Functions</i>	28
	<i>Combining Logical Tests</i>	28
	<i>The NOT Operator</i>	28
4.2	Working with Blanks	29
	<i>Producing a Blank</i>	29
	<i>Blank Arithmetic</i>	29
	<i>Testing for Blanks</i>	29
	<i>The COALESCE Function</i>	30
4.3	Testing for Errors	31
	<i>The ISERROR and IFERROR Functions</i>	31
	<i>Avoiding Error Functions</i>	31
4.4	The SWITCH Function	32
	<i>A Simple SWITCH Function</i>	32
	<i>Logical Tests in a SWITCH Function</i>	32

TABLE OF CONTENTS (2 of 4)

5	BASIC MEASURES	Page
5.1	Introduction to Measures	33
	<i>Measures vs. Calculated Columns</i>	33
	<i>Implicit Measures</i>	33
5.2	Creating a Measure	34
	<i>Adding a Measure to a Table</i>	34
	<i>Formatting Measures</i>	35
	<i>Displaying a Measure in a Visual</i>	35
	<i>Referencing Measures</i>	36
5.3	Filter Context	37
	<i>What is Filter Context?</i>	37
	<i>How DAX Applies Filter Context</i>	38
5.4	Measures Tables	39
	<i>Creating a Separate Measures Table</i>	39
	<i>Moving Measures</i>	40
5.5	Quick Measures	41
	<i>Creating a Quick Measure</i>	41
	<i>Editing a Quick Measure</i>	42
	<i>Using a Quick Measure</i>	42

6	AGGREGATION FUNCTIONS	Page
6.1	Aggregating Column Values	43
	<i>Basic Aggregation Functions</i>	43
	<i>Functions for Counting</i>	44
	<i>Dealing with Boolean Values</i>	44
6.2	Aggregating Expressions	45
	<i>The AggregateX Functions</i>	45
6.3	Iterators and Row Context	46
	<i>A Reminder of Filter Context</i>	46
	<i>Row Context in Iterator Functions</i>	47
	<i>The Final Result</i>	47
	<i>How to Spot Iterators</i>	47

7	THE CALCULATE FUNCTION	Page
7.1	Introducing the CALCULATE Function	48
	<i>Expressions in the CALCULATE Function</i>	48
7.2	Adding New Filters	49
	<i>Basic Filter Expressions</i>	49
	<i>Adding Multiple Filters</i>	50
	<i>Filter Arguments and Filter Context</i>	50
	<i>Multiple Columns in Filter Arguments</i>	51
7.3	Replacing Filters	52
	<i>Replacing an Existing Filter</i>	52
	<i>Comparing Differently Filtered Measures</i>	53
	<i>Dealing with Blank Values</i>	53
7.4	Keeping Filters	54
	<i>The Problem with the Default Behaviour</i>	54
	<i>The KEEPFILTERS Function</i>	55
	<i>Using the VALUES Function</i>	55
7.5	Removing Filters	56
	<i>Removing Every Filter</i>	56
	<i>Using the ALL Function</i>	57
	<i>Comparing Filtered and Unfiltered Values</i>	57
	<i>Removing Filters from Specific Fields</i>	58
	<i>An Issue with Sort-By Fields</i>	59
	<i>Removing Filters from a Specific Table</i>	59
7.6	Special Filter Removal Functions	60
	<i>The ALLEXCEPT Function</i>	60
	<i>The ALLSELECTED Function</i>	61

8	VARIABLES	Page
8.1	Introduction to Variables	62
	<i>Using Variables in Measures</i>	62
8.2	How Variables are Evaluated	63
	<i>Lazy Evaluation</i>	63
	<i>DAX Variables are Constants</i>	63
8.3	Debugging with Variables	64
	<i>Returning Different Variables</i>	64
8.4	Nesting Variables	65
	<i>Variables in Functions</i>	65
	<i>Variable Scope</i>	66

TABLE OF CONTENTS (3 of 4)

9	THE FILTER FUNCTION	Page
9.1	Introduction to the FILTER Function	67
	<i>A Basic FILTER Example</i>	67
	<i>Using the CALCULATE Function</i>	67
	<i>How CALCULATE and FILTER are Related</i>	68
	<i>Using Multiple Filters</i>	68
	<i>Using Variables</i>	68
9.2	FILTER vs. CALCULATE	69
	<i>Referencing Multiple Fields</i>	69
	<i>Using Fields from Different Tables</i>	69
	<i>Referring to Measures</i>	70
	<i>Replacing Filters</i>	71
9.3	The CALCULATETABLE Function	72
	<i>Using CALCULATETABLE</i>	72

10	FILTERS AND RELATIONSHIPS	Page
10.1	Relationships and Filter Direction	73
	<i>The Problem with Relationships</i>	73
	<i>Changing the Cross Filter Direction</i>	74
	<i>Solving the Problem using Filters</i>	75
10.2	Cross Filter Direction in Measures	76
	<i>The CROSSFILTER Function</i>	76
	<i>Using Single and Both Filter Directions Simultaneously</i>	77
	<i>Multiple CROSSFILTER Functions</i>	77

11	CONTEXT TRANSITION	Page
11.1	What is Context Transition?	78
	<i>Row and Filter Context</i>	78
11.2	Context Transition in Calculated Columns	79
	<i>Row Context in Calculated Columns</i>	79
	<i>Performing Context Transition</i>	79
	<i>Implicit Context Transition</i>	80
	<i>The RELATEDTABLE Function</i>	80
11.3	Context Transition in Measures	81
	<i>Row Context in Measures</i>	81
	<i>Context Transition in Measures</i>	82
	<i>The Effect of Filter Context</i>	82
	<i>Removing Filters</i>	83
11.4	Ranking Values	84
	<i>The RANKX Function</i>	84
	<i>Ranking in Calculated Columns</i>	84
	<i>Context Transition in Calculated Columns</i>	85
	<i>Ranking in Measures</i>	85

12	TIME INTELLIGENCE	Page
12.1	Introduction to Time Intelligence	86
	<i>Calendar Tables</i>	86
	<i>The Date Column</i>	87
	<i>Referring to Calendar Tables</i>	87
12.2	Time Intelligence Functions	88
	<i>General Time Intelligence Functions</i>	88
	<i>Using the DATEADD Function</i>	88
	<i>How DATEADD Works</i>	89
	<i>Using the DATESINPERIOD Function</i>	89
	<i>Using the Current Date</i>	90
	<i>Using Specific Dates</i>	90
12.3	To Date Calculations	91
	<i>Returning Date Ranges</i>	91
	<i>Calculating Running Totals</i>	91
	<i>Total To Date Functions</i>	92
	<i>Easier Running Total Calculations</i>	92
	<i>Specifying Year End Dates</i>	93
	<i>Calculating Life to Date Values</i>	93
12.4	Next and Previous Periods	94
	<i>Next and Previous Period Functions</i>	94
	<i>Comparing Entire Previous Years</i>	95
	<i>Comparing Parts of Previous Years</i>	96
12.5	Period Start and End Dates	97
	<i>Period Start and End Functions</i>	97
	<i>Start and End Dates</i>	98
	<i>Opening and Closing Balances</i>	99
	<i>First and Last Non-Blank Dates</i>	99
	<i>First and Last Non-Blank Values</i>	100
	<i>Non-Blank Opening Balances</i>	100
12.6	Moving Averages	101
	<i>Calculating a Moving Average</i>	101

TABLE OF CONTENTS (4 of 4)

13	CUSTOM CALENDARS	Page	14	DYNAMIC MEASURES	Page
13.1	Why use Custom Calendars?	102	14.1	Dynamic Labels	114
	<i>Disabling Automatic Calendars</i>	102		<i>Why use Dynamic Labels?</i>	114
13.2	Creating a Custom Calendar	103	14.2	Returning a Single Value	115
	<i>The CALENDARAUTO and CALENDAR Functions</i>	103		<i>The VALUES Function</i>	115
	<i>Adding Extra Columns</i>	104		<i>Testing for a Single Value</i>	116
	<i>Financial Years</i>	105		<i>The SELECTEDVALUE Function</i>	116
13.3	Finishing the Calendar	106	14.3	Concatenating Values	117
	<i>Marking as a Date Table</i>	106		<i>The CONCATENATEX Function</i>	117
	<i>Changing Default Aggregations</i>	106		<i>More Complex Expressions</i>	117
	<i>Setting Sort-By Columns</i>	107	14.4	Filtered Values	118
	<i>Creating Hierarchies</i>	107		<i>Testing for Filtered Values</i>	118
	<i>Hiding Fields</i>	108		<i>Testing for Cross Filtered Values</i>	118
	<i>Creating a Relationship</i>	108		<i>Selecting the Top N Rows</i>	119
13.4	Using a Custom Calendar	109	14.5	Disconnected Slicers	120
	<i>Creating Visuals</i>	109		<i>Creating a Disconnected Table</i>	120
	<i>Time Intelligence Functions</i>	109		<i>Creating a Disconnected Slicer</i>	121
13.5	Multiple Date Fields	110		<i>Referencing the Selected Value</i>	121
	<i>Using Multiple Calendars</i>	110	14.6	Formatting with Measures	122
	<i>Using a Single Calendar</i>	111		<i>Calculating Colours with Measures</i>	122
	<i>Changing the Active Relationship</i>	111		<i>Using Measures in Conditional Formatting</i>	123
	<i>The USERRELATIONSHIP Function</i>	112		<i>Choosing Colours with Slicers</i>	123
13.6	Special Dates	113			

CHAPTER 1 - INTRODUCTION TO DAX IN POWER BI

1.1 DAX in Power BI

In Power BI you can use the *DAX (Data Analysis eXpressions)* language to create *calculated columns, measures and tables*. You can see an example of each below.

How DAX is Used 1 – Calculated Columns

A *calculated column* is like a formula in an Excel table. The results of a calculated column are stored in the data model.

SaleDate	ProductId	CentreId	Quantity	Price	Sale value
21 July 2023	10	101	1	£3.99	£3.99
21 July 2023	12	178	2	£6.99	£13.98
21 July 2023	13	207	1	£3.95	£3.95
21 July 2023	13	342	4	£3.95	£15.80

This calculated column gives the value for each row of a sales table by multiplying the price of an item by the quantity sold.

As we will see in this manual, a calculated column is evaluated for each row of a table; DAX uses the *row context* to access the correct values for each separate calculation.

How DAX is Used 2 – Measures

A *measure* is a formula which calculates a value when you place it in a visual. The visual provides a *filter context* which tells the measure which values it can use in the calculation.

```

1 Sum of sale value = SUMX(
2   Sales,
3   Sales[Price] * Sales[Quantity]
4 )
    
```

RegionName	Air	Land	Water	Total
East Anglia	1,268.71	13,640.28	2,050.91	16,959.90
East Midlands	2,609.87	26,289.48	5,033.67	33,933.02
London	4,899.45	54,018.62	9,136.50	68,054.57
North	2,874.62	25,980.55	4,334.49	33,189.66
North West	7,950.17	69,751.18	12,614.71	90,316.06
South East	10,697.97	102,171.04	17,507.12	130,376.13
South West	2,616.48	23,769.80	4,506.69	30,892.97
West Midlands	4,702.33	49,116.82	8,878.73	62,697.88
Yorkshire & Humberside	6,231.78	47,396.27	8,509.78	62,137.83
Total	43,851.38	412,134.04	72,572.60	528,558.02

This measure multiplies the price of an item by the quantity sold for each row of a sales table, then sums the results.

When we place the measure in a visual (a matrix in this case) the measure is calculated for each data point in the visual.

The filter context controls which values the measure can access. This cell gives the total sales value for products in the **Water** environment sold in the **South West** region.

How DAX is Used 3 – Calculated Tables

Although most of the tables in your Power BI data model will be created by importing data, you can also use DAX to calculate tables. It's common to use this technique to create custom calendars.

```

1 Sales Calendar = ADDCOLUMNS(
2     CALENDAR(
3         DATE(YEAR(MIN(Sales[SalesDate])), 1, 1),
4         DATE(YEAR(MAX(Sales[SalesDate])), 12, 31)
5     ),
6     "Year", YEAR([Date]),
7     "Month", MONTH([Date]),
8     "Month Name", FORMAT([Date], "MMMM"),
9     "Day", DAY([Date])
10 )


```

Date	Year	Month	Month Name	Day
01/01/2020 00:00:00	2020	1	January	1
02/01/2020 00:00:00	2020	1	January	2
03/01/2020 00:00:00	2020	1	January	3

This DAX expression creates a calculated table which contains a range of dates related to the sales in our database.

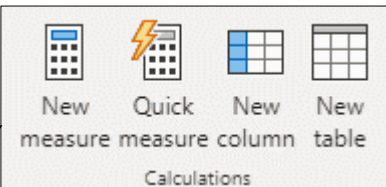
Choosing the Type of Calculation

Power BI provides several ways to create calculated columns, measures and tables. One way is to click the relevant tool on the **Home** tab of the ribbon while in the **Data** view.



Select the **Data view** tool on the left of the Power BI window.

The **Calculations** group on the **Home** tab of the ribbon has a button for each type of DAX calculation.



New measure Quick measure New column New table

Calculations

Where Else is DAX Used?

In addition to Power BI, you can write DAX in the following applications:

Application	Description
<i>Power Pivot</i>	Power Pivot is an add-in for Excel which allows you to combine data from multiple sources and present this in a pivot table or chart.
<i>SQL Server Analysis Services (SSAS) Tabular</i>	SSAS Tabular Model allows you to combine data from lots of different data sources, apply security to it to control who sees what and then allow employees of your organisation to share the resulting data model.

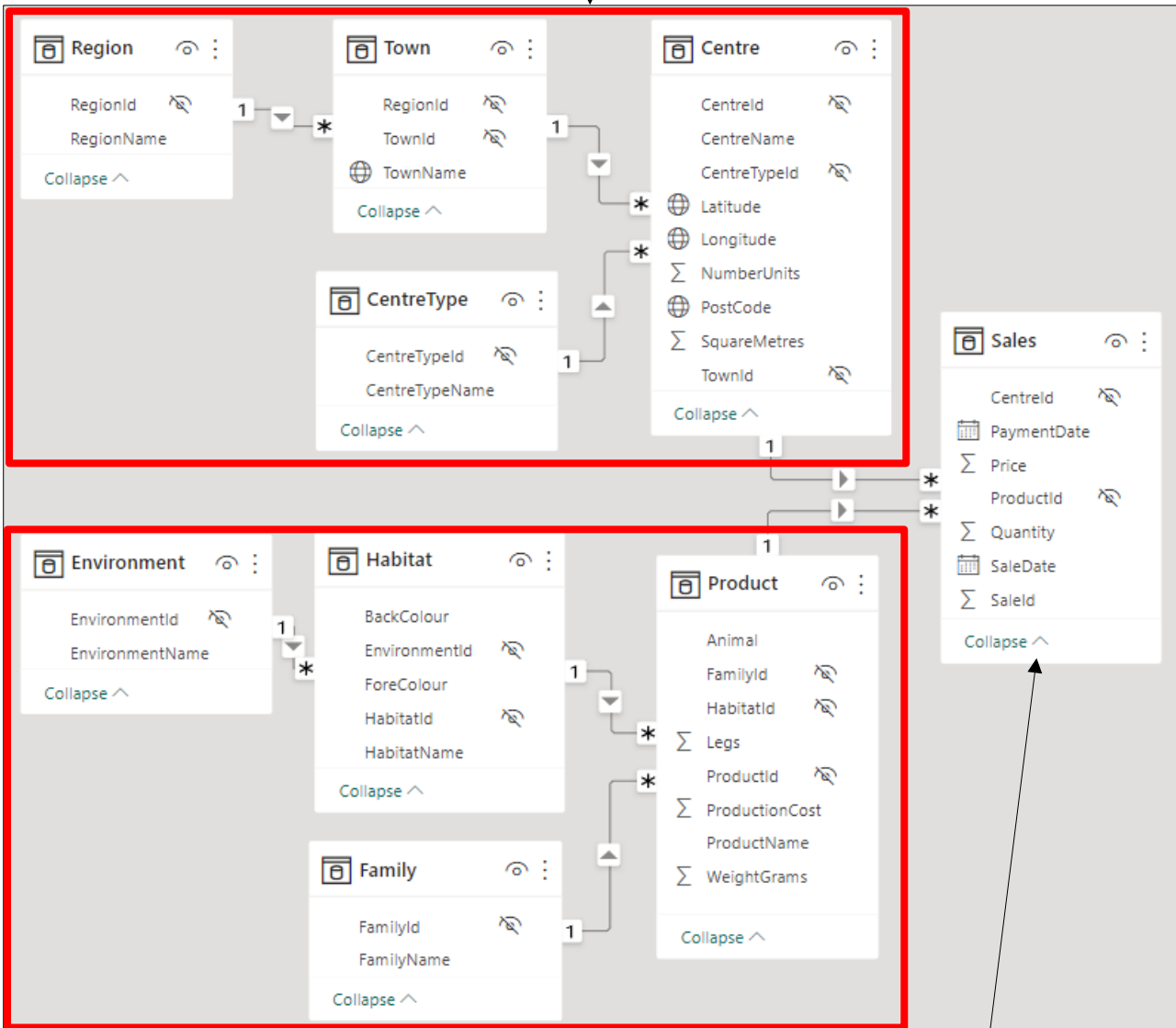
1.2 The Construct-a-Creature Database

This courseware uses data from the (fictitious) Wise Owl subsidiary **Construct-a-Creature** (a retail chain loosely modelled on Build-a-Bear, but with a wider range of animals available for purchase).

The Tables and Relationships

You can see the tables and relationships of the database in the diagram below:

There are four tables representing a geographical dimension to do with where sales took place. These tables give the shopping centre, type of centre, town and region.



There are also four tables representing a product dimension. These give details of the product sold (for example, a frog is an amphibian which lives in a fresh water habitat in a watery environment).

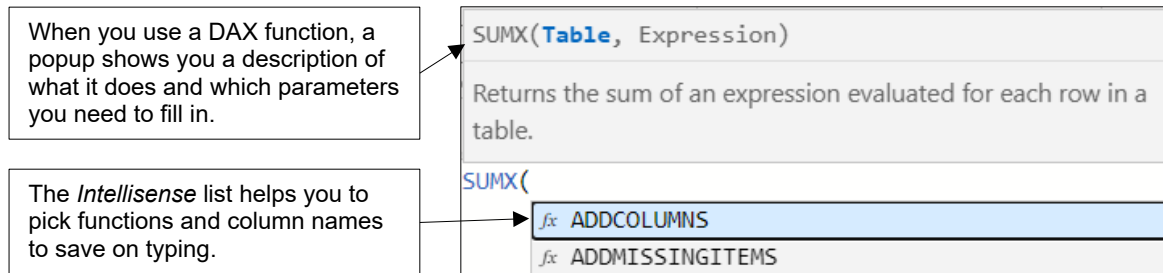
The **Sales** table stores how many of each product were sold in each transaction.

1.3 Getting Help with DAX

The amount of help available for DAX both within Power BI and from third parties has increased dramatically since the product was first released.

Getting Help within Power BI

You'll see various popups appear to help you as you write DAX in Power BI.



Other Sources of Help

You can find more descriptive help for DAX on a range of websites, as shown in the table below:

Website	Description	URL
<i>Microsoft DAX Reference</i>	Microsoft's official documentation for DAX functions. It's somewhat dry but a useful technical reference.	https://learn.microsoft.com/en-us/dax/
<i>Power BI Community</i>	A Microsoft forum in which you can post questions about any aspect of Power BI and rely on other members to provide answers.	https://community.fabric.microsoft.com/t5/Microsoft-Power-BI-Community/ct-p/powerbi
<i>SQLBI</i>	A third-party site maintained by Marco Russo and Alberto Ferrari. The site contains lots of free resources to help you with DAX.	https://www.sqlbi.com/
<i>DAX Guide</i>	A third-party alternative to Microsoft's DAX Reference created by the Italians. This site fleshes out the detail of DAX functions and provides links to helpful articles which describe in more detail how the functions work.	https://dax.guide/
<i>Wise Owl</i>	The Wise Owl website contains lots of free resources including videos, blogs and exercises to help you with learning DAX.	https://www.wiseowl.co.uk/resources/

CHAPTER 2 - BASIC CALCULATED COLUMNS

2.1 Calculated Columns

A *calculated column* is a type of calculation you can create using DAX in Power BI. This chapter shows you the basics of writing DAX using calculated columns.

Key Features of Calculated Columns

You can see some of the key features of calculated columns in the table below:

Feature	Description
<i>Created in data tables</i>	You create a calculated column in a table in the data model. The calculated column can refer directly to any column in the same table.
<i>Calculated immediately</i>	A calculated column produces its results as soon as you enter it. The values are updated whenever the data model is refreshed.
<i>Stores data in the model</i>	A calculated column stores its results in the data model. Each calculated column you create increases the storage space required by the model.
<i>Uses row context</i>	The expression in a calculated column is evaluated for each row in the table. The row context provides the expression with access to values on the same row in the table.

Creating a Calculated Column

You can create a calculated column in any of the three Power BI views but, if you want to see the results of your calculation, it's best to select the **Data** view.

Select the **Data view** button on the left of the Power BI window.

You can right-click on any column in the table and choose **New column** to add a calculated column.

You can choose **New column** from the **Table tools** or **Home** tabs of the ribbon to add a calculated column.

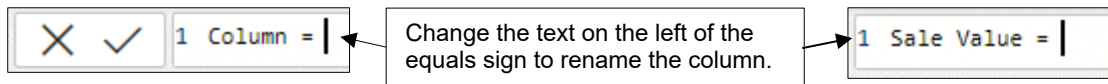
You can right-click the table in the **Data** panel and choose **New column** to add a calculated column.

Quantity	Price	PaymentDate
1	Sort ascending	
1	Sort descending	
1	Clear sort	
1	Clear filter	
1	Clear all filters	
1	Copy	
1	Copy table	
2	New measure	
2	New column	
2	Refresh data	
2		
2		

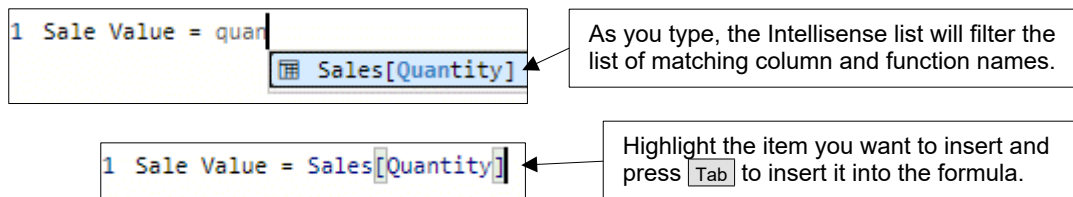
Entering a Formula

After choosing to create a calculated column you can enter your DAX code in the formula bar below the ribbon. The example below divides one column by another to create a new value:

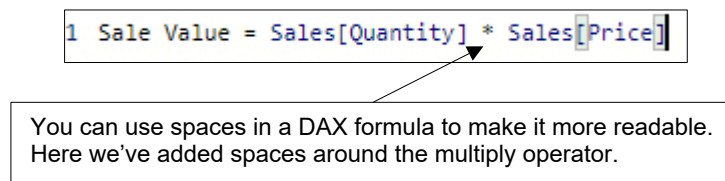
- 1) Start by giving the column a sensible name.



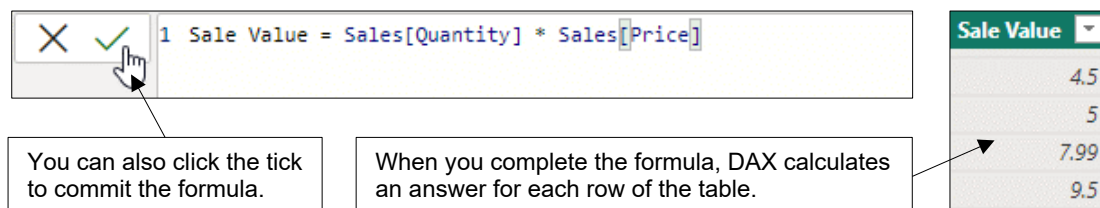
- 2) To reference a column in the same table, simply begin typing the column name.



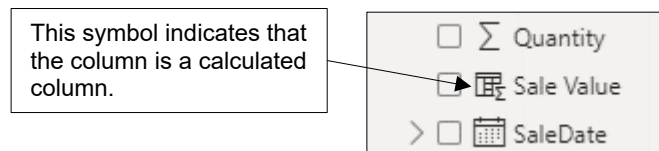
- 3) Type in an operator then reference the next column.



- 4) Press **Enter** to commit the formula.



Your calculated column will appear in the table in the **Data** pane with a special symbol to indicate its status.



Calculated Column Properties

Once you've created a calculated column, you can modify it in the same way as any other column in your data model, as shown in the diagram below.

The diagram illustrates the 'Column tools' ribbon in Power BI Desktop, divided into three main sections:

- Structure:** Contains the 'Name' field (set to 'Sale Value') and the 'Data type' dropdown (set to 'Decimal number'). A callout states: "You can apply formatting to the column using these tools."
- Formatting:** Includes the 'Format' dropdown (set to 'General'), currency symbols (\$, %), and a 'Auto' dropdown. A callout states: "Select the column in the **Data** pane. You can then use the **Column tools** tab in the ribbon to modify its properties."
- Properties:** Features the 'Summarization' dropdown (set to 'Sum') and the 'Data category' dropdown (set to 'Uncategorized'). A callout states: "You can change the default aggregation function used when you add the column to a visual using this drop-down list." Another callout states: "You can assign a data category to the column using this drop-down list."

Using Calculated Columns in Visuals

You can use a calculated column to populate visuals in your report, just as for any other column in your data model.

The diagram shows a bar chart titled "Sale Value by Centre Size" and its configuration pane. The chart displays three bars for 'Small', 'Medium', and 'Large' centre sizes, with the x-axis representing the 'Sum of Sale Value' (0.0M, 0.2M, 0.4M). The configuration pane shows:

- Y-axis:** 'Centre Size' (a calculated column).
- X-axis:** 'Sum of Sale Value' (a calculated column).
- Legend:** 'Add data fields here'.
- Small multiples:** 'Add data fields here'.

Callouts explain the usage of calculated columns in visuals:

- "If the calculation produces text, you can use it to populate category fields in a visual. Here we've used a calculated column called **Centre Size** to group the values in a chart."
- "You can assign numeric calculated columns to value fields in a visual. Here we're using the **Sale Value** calculated column to create a sum of sale value."

2.2 Referencing Columns and Tables

This section shows you various ways to refer to columns and tables in your DAX formulae.

Qualifying Column Names

When you select a column name from the Intellisense list, it will be automatically qualified with the name of the table to which it belongs. You don't have to include the table name, however.

```
1 Average Unit Size = Centre[SquareMetres] / Centre[NumberUnits]
```

```
1 Average Unit Size = [SquareMetres] / [NumberUnits]
```

These two formulae give the same result.

Although you don't always have to include the table name when referencing a column, it makes sense to do so for the following reasons:

- If you always include the table name you don't have to remember when it is or isn't required.
- It allows you to spot when you're referring to a column as opposed to, say, a measure.
- When a column name exists in multiple tables it makes the reference unambiguous.

Table Names

If the name of a table contains spaces, or it conflicts with another DAX keyword, you must enclose the table name in single quotes.

```
1 Sale value = 'Sales Table'[Price] * 'Sales Table'[Quantity]
```

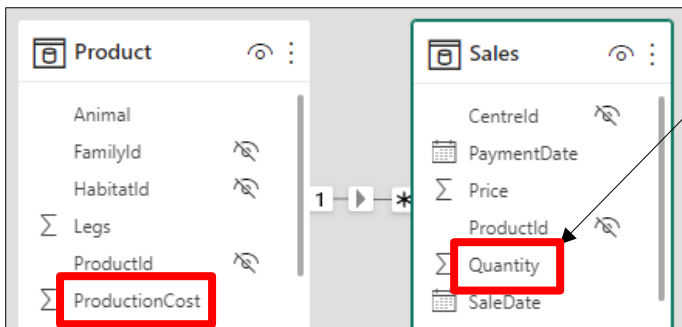
Sales Table must be enclosed in single quotes due to the space in the table name.



You can always enclose a table name in single quotes, even when it isn't required.

Referencing Columns in Other Tables

In a calculated column you can only directly reference other columns which belong to the same table. This is a problem when your calculation needs to refer to columns in other tables!



We'd like to create a calculated column in the **Sales** table which multiplies the **Quantity** by the **ProductionCost** in the **Product** table.

```
1 Sale cost = Sales[Quantity] * Product[ProductionCost]
```

! A single value for column 'ProductionCost' in table 'Product' cannot be

When we reference the **ProductionCost** column directly, the formula results in an error.

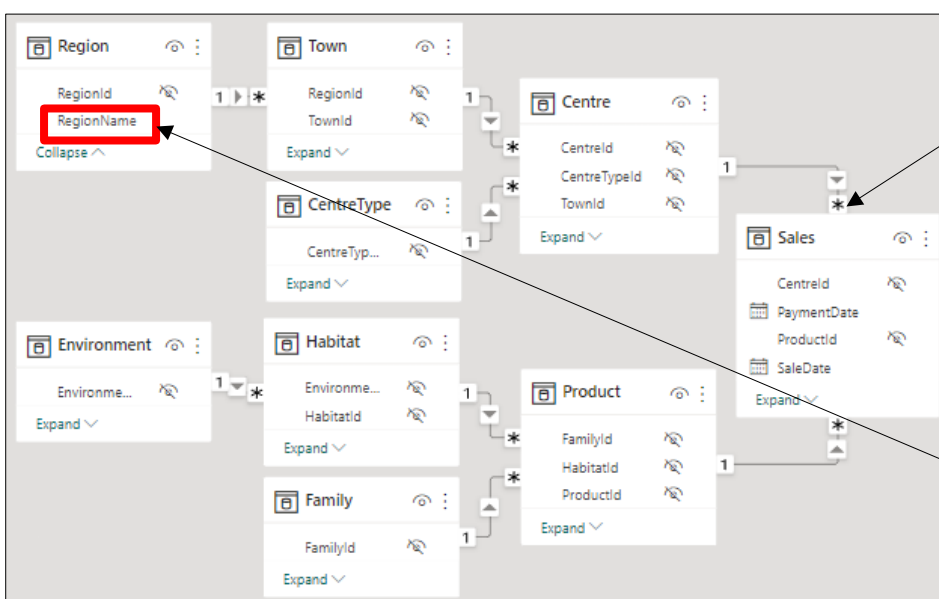
The Related Function

The key to solving the above problem is that the **Product** and **Sales** tables have a relationship. This allows us to use the **RELATED** function to reference a column in a related table.

```
1 Sale cost = Sales[Quantity] * RELATED(Product[ProductionCost])
```

We can successfully reference the **ProductionCost** column by using the **RELATED** function.

You can reference columns from a table at the **1** end of a relationship, regardless of how many steps it takes. In our model, the **Sales** table sits at the many end of every relationship:



A calculated column in the **Sales** table can reference any other column in the data model using the **RELATED** function.

In the **Sales** table you could return the **RegionName** with this formula:

```
=RELATED(
  Region[RegionName]
)
```

2.3 Editing DAX Code

Although you can write a DAX formula as a continuous stream of code, there are several things you can do to make your code more readable.

```
1 Sale Value % of Total = IF(NOT(ISBLANK(SUMX(Sales,Sales[Price]*Sales
  [Quantity]))),DIVIDE(SUMX(Sales,Sales[Price]*Sales[Quantity]),
  CALCULATE(SUMX(Sales,Sales[Price]*Sales[Quantity]),REMOVEFILTERS()))))
```

Laying out a formula like this makes it very difficult to work with!



It's not important to understand what the code in this section does – instead, focus on the techniques used to make it more readable.

Multiple Lines and Indenting

You can break a formula onto multiple lines and add tab spaces to make it more readable. You can use the following keys to add new lines and indenting to a formula:

Key	What it does
Shift + Enter or Alt + Enter	Adds a new line, and a tab level if appropriate.
Tab	Indents the highlighted lines one tab space.
Shift + Tab	Outdents the highlighted lines one tab space.

You can see an example of a formula with new lines and indenting in the diagram below:

When you use a function (in this case **DIVIDE**), open the round brackets then start a new line and add a tab space.

Close the brackets for a function at the same indent level as the line on which the brackets were opened. Use the vertical grey lines to help you put the brackets in the right place.

Each new function that you use should have its arguments indented one extra tab level.

All of the arguments for a single function should be at the same tab level.

Use a comma at the end of a line to separate one argument from the next.

```
1 Sale Value % of Total = DIVIDE(
2     SUMX(
3         Sales,
4         Sales[Price]*Sales[Quantity]
5     ),
6     CALCULATE(
7         SUMX(
8             Sales,
9             Sales[Price]*Sales[Quantity]
10        ),
11        REMOVEFILTERS()
12    )
13 )
```

Comments

You can use *comments* to annotate your code. You can add a comment at the end of any line after the = sign in a formula.





<pre> 1 Sale Value % of Total = //divide filtered sales by all sales 2 DIVIDE(3 SUMX(--calculate filtered sales 4 Sales, 5 Sales[Price]*Sales[Quantity] 6), 7 /* 8 remove filters 9 to get unfiltered sales 10 */ 11 CALCULATE(</pre>	<p>You can type // to begin adding a comment, followed by the comment text.</p>
<p>You can also begin a comment using -- rather than //.</p>	
<p>Start a multi-line comment with /* and end it with */ as shown here.</p>	

Keyboard Shortcuts

You can use a range of keyboard shortcuts to help you edit your DAX code. You can see some of these in the table below:

Key	What it does
Ctrl + G	Goto the specified line number.
Alt + ↑ / Alt + ↓	Move the line of code up / down.
Shift + Alt + ↑ / Shift + Alt + ↓	Copy the line of code up / down.
Ctrl + Shift + \	Jump to the paired bracket.
Alt + Left mouse click	Add a text cursor at the clicked position.
Ctrl + L	Select the line of code.
Ctrl + Shift + L	Select all occurrences of the current selection.
Ctrl + F2	Select all occurrences of the current word.
Ctrl + /	Comment/uncomment the line of code.
Ctrl + = / Ctrl + -	Zoom in / zoom out.
Ctrl + Space Bar	Show the Intellisense list.
Ctrl + I	Show and hide tooltips.
Ctrl + J	Expand and collapse the formula bar.

WHAT WE DO

					
	ONLINE TRAINING	MANCHESTER OR LONDON	AT YOUR OFFICE	BESPOKE CONSULTANCY	
OFFICE 365	Microsoft Excel	✓	✓	✓	✓
	VBA macros	✓	✓	✓	✓
	Office Scripts	✓		✓	
	Microsoft Access				✓
POWER PLATFORM	Power BI and DAX	✓	✓	✓	✓
	Power Apps	✓		✓	
	Power Automate	✓	✓	✓	✓
SQL SERVER	Reporting Services	✓	✓	✓	✓
	Report Builder	✓		✓	✓
	Integration Services	✓	✓	✓	✓
	Analysis Services	✓		✓	
CODING LANGUAGES	SQL	✓	✓	✓	✓
	Visual C#	✓	✓	✓	✓
	Python	✓	✓	✓	✓



