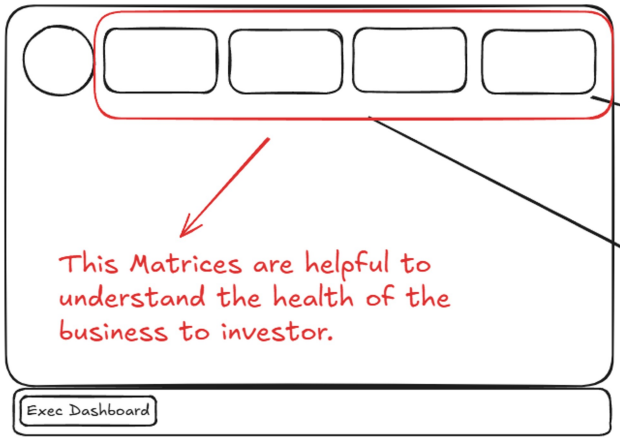
Power BI - All about DAX - Lecture 7



- C.C - Row Context [Group By] - Filtering
- Measures - Aggregation - Filter Context.

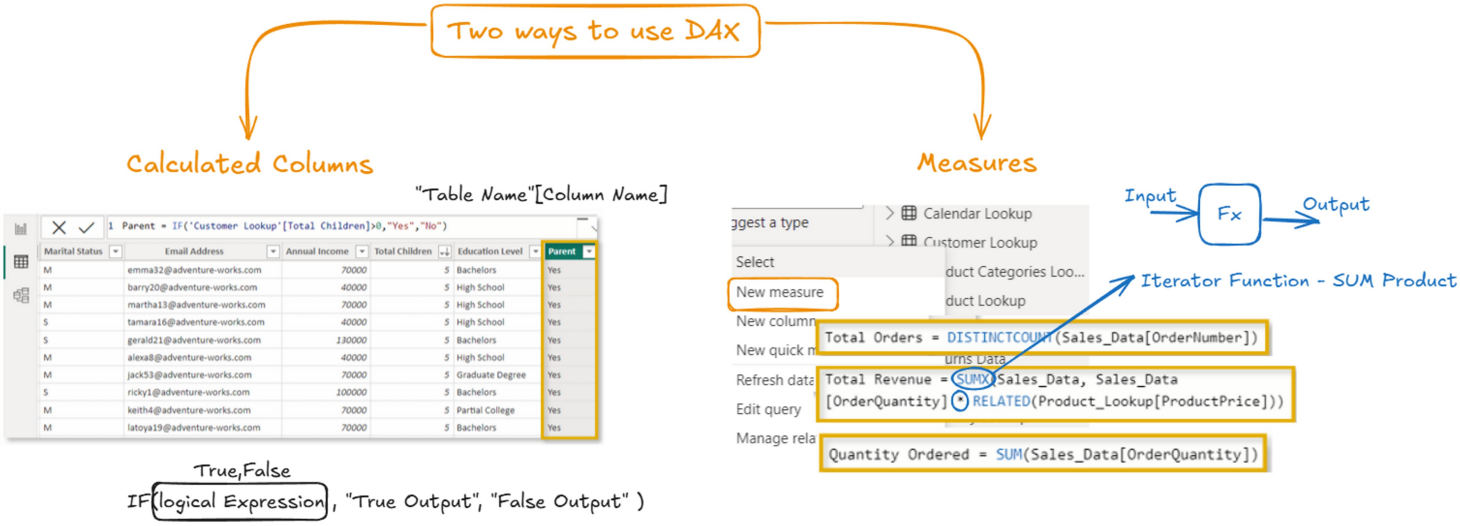
Group BY - Filtering [Distinct Value]

Aggregated Columns - measures .

MEET DAX

Data Analysis Expressions (commonly known as DAX) is the formula language that drives the Power BI front-end. With DAX, you can:

- Go beyond the capabilities of traditional spreadsheet formulas, with powerful and flexible functions built specifically to work with relational data models.
- Add calculated columns (for filtering) and measures (for aggregation) to enhance data models.

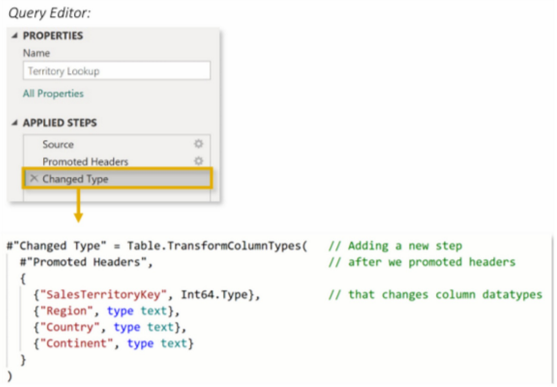


M VS. DAX

M and DAX are two distinct functional languages used within Power BI Desktop:

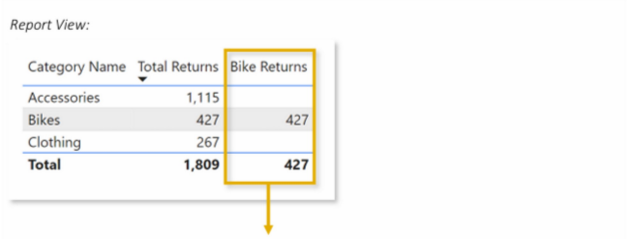
- M is used in the Power Query editor, and is designed specifically for extracting, transforming and loading data.
- DAX is used in the Power BI front-end, and is designed specifically for analyzing relational data models

M CODE




```
#"Changed Type" = Table.TransformColumnTypes(    // Adding a new step
  #"Promoted Headers",                           // after we promoted headers
  {
    {"SalesTerritoryKey", Int64.Type},           // that changes column datatypes
    {"Region", type text},
    {"Country", type text},
    {"Continent", type text}
  }
)
```

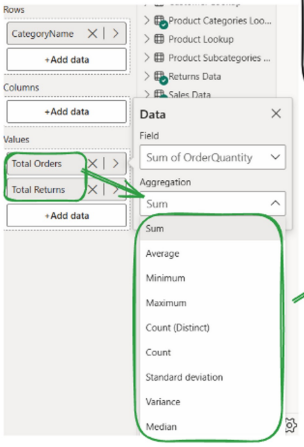
DAX




```
1 Bike Returns =
2 CALCULATE(
3     [Total Returns],                                  // Counting total returns
4     'Product Categories Lookup'[Category Name] = "Bikes"    // filtered for bikes only
5 )
```

DAX

1. Calculated Columns
2. Measures :
   - A. Implicit Measure
   - B. Explicit Measure




```
Mountain Bike Returns =
    CALCULATE([Bike Returns],
    'Product Subcategory Lookup'[Subcategory Name] =
          "Mountain Bike")
```

Implicit Measure

All Orders = CALCULATE(Measure, Filter)

Reference Measure.

CALCULATED COLUMNS

Calculated columns allow you to add new, formula-based columns to tables in a model

- Calculated columns refer to entire tables or columns (no A1-style cell references).

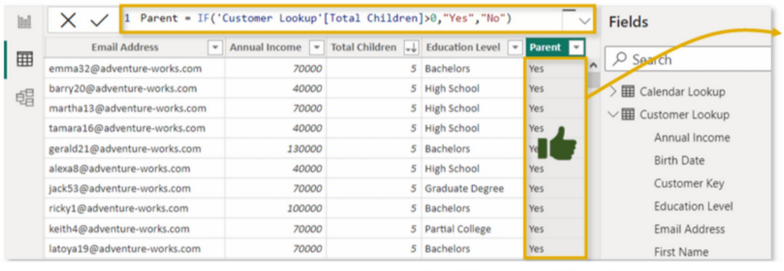
- Calculated columns generate values for each row, which are visible within tables in the Data view.

- Calculated columns understand row context; they're great for defining properties based on information in each row, but generally useless for aggregation (sum, count, etc.)

HEY THIS IS IMPORTANT!
As a rule of thumb, use calculated columns to "stamp" static, fixed values to each row in a table (or go upstream and use the Query Editor!)
DO NOT use calculated columns for aggregation – this is what measures are for!

PRO TIP:
Calculated columns are typically used for filtering & grouping data, rather than creating aggregate numerical values

EXAMPLE: CALCULATED COLUMNS

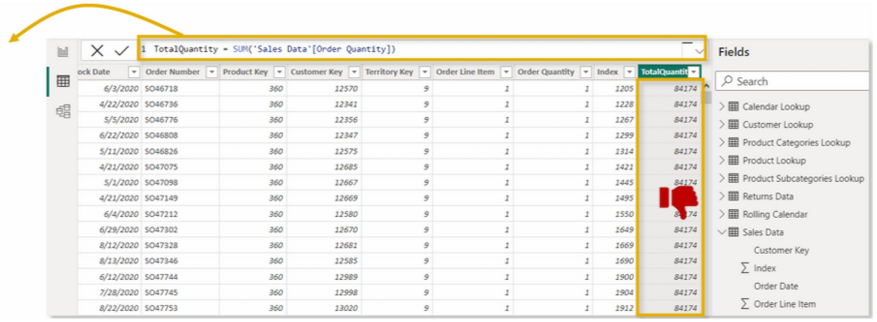


In this case we've added a calculated column named Parent, which equals "Yes" if the [Total Children] field is greater than 0, and "No" otherwise

- Since calculated columns understand row context, a new value is calculated in each row based on the value in the [Total Children] column.

- This is a valid use of calculated columns; it creates a new row "property" that we can use to filter or segment any related data within the model.

Here we're using an aggregation function (SUM) to calculate a new column named Total Quantity



- Since this is an aggregation function, the same grand total is returned in every row of the table
- This is not a valid use of calculated columns; these values are statically "stamped" onto the table and can't be filtered, sliced, etc.

DAX MEASURES

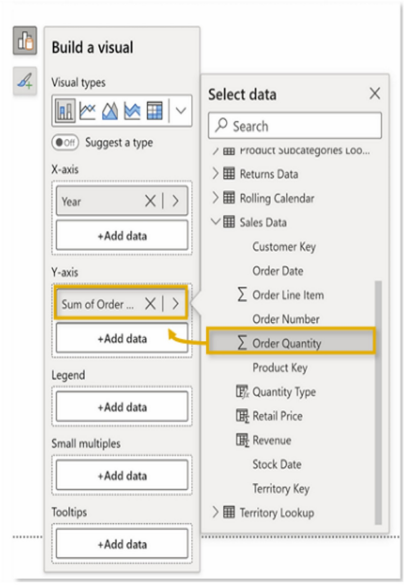
Measures are DAX formulas used to generate new calculated values

- Like calculated columns, measures reference entire tables or columns (no A1-style cell references).
- Unlike calculated columns, measures aren't visible within tables; they can only be "seen" within a visualization like a chart or matrix (similar to a calculated field in a PivotTable).
- Measures evaluate based on filter context, which means they recalculate when the fields or filters around them change.

HEY THIS IS IMPORTANT!
As a rule of thumb, use measures when a single row can't give you the answer, or when you need to aggregate values across multiple rows in a table

PRO TIP:
Use measures to create numerical, calculated values that can be analyzed in the "values" field of a report visual

IMPLICIT VS. EXPLICIT MEASURES



Example of an implicit measure

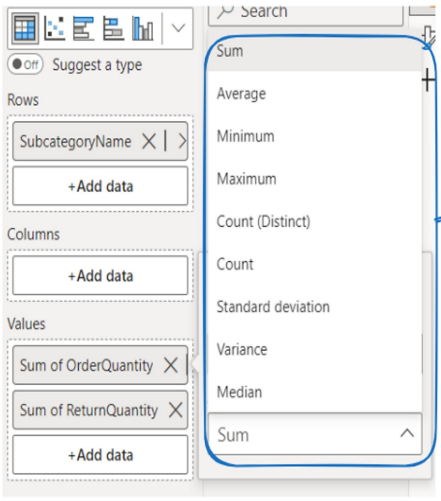
Implicit measures are created when you drag raw numerical fields into a report visual and manually select an aggregation mode (Sum, Average, Min, Max, Count, etc.)

Explicit measures are created when you actually write a DAX formula and define a new measure that can be used within the model

HEY THIS IS IMPORTANT!

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

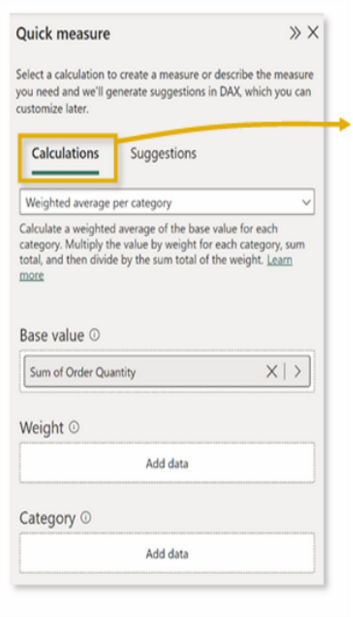
Explicit measures can be used anywhere in the report, and referenced by other DAX calculations to create "measure trees".



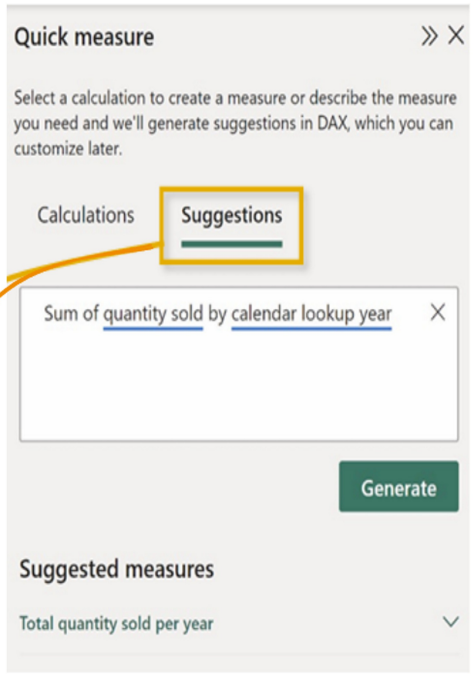
Implicit Measures.

QUICK MEASURES

Quick measures automatically create formulas based on pre-built templates or natural language prompts



Quick measure calculations can be used to build measures using predefined templates (weighted averages, percent difference, time intelligence, etc.)



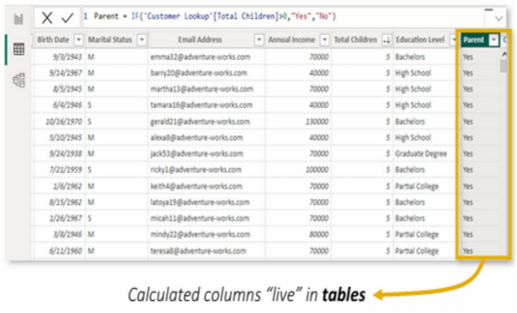
Quick measure suggestions can be used to find suggested measures based on natural language queries (i.e. "sum of quantity sold by calendar year")

PRO TIP:
Quick measures can be a great learning tool for beginners or for building more complex formulas but use them with caution; mastering DAX requires a deep understanding of the underlying theory!

RECAP: CALCULATED COLUMNS VS. MEASURES

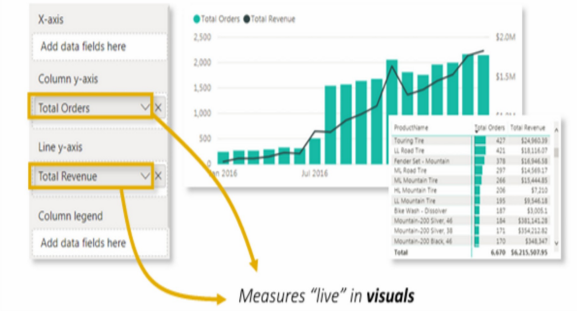
CALCULATED COLUMNS

- Values are calculated based on information from each row of a table (row context)
- Appends static values to each row in a table and stores them in the model (which increases file size)
- Recalculate on data source refresh or when changes are made to component columns
- Primarily used for filtering data in reports



MEASURES

- Values are calculated based on information from any filters in the report (filter context)
- Does not create new data in the tables themselves (doesn't increase file size)
- Recalculate in response to any change to filters within the report
- Primarily used for aggregating values in report visuals



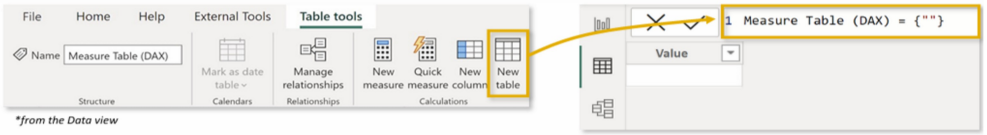
PRO TIP: MEASURE TABLES

It's a common best practice to create a dedicated table to store your measures; this will help you stay organized, find measures quickly, and allow you to group related measures into folders.

Option 1: Enter Data into Power Query (loads the table to the data model – table is visible in Power Query)

Add a table Name and click OK to load the table to the data model

Option 2: Create a calculated table using DAX directly in the model (table is not visible in Power Query)



Create new table & use a table constructor { } to add a single column

FILTER CONTEXT

Measures are evaluated based on filter context, which means that they recalculate whenever the fields or filters around them change

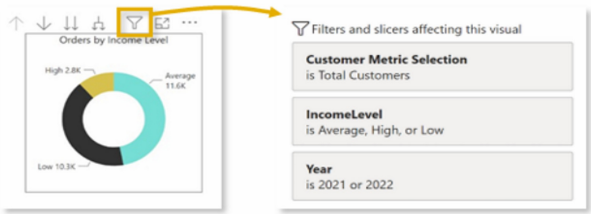
Top 10 Products	Orders	Revenue	Return %
Water Bottle - 30 oz.	3,983	$39,755	1.95%
Patch Kit/8 Patches	2,952	$13,506	1.61%
Mountain Tire Tube	2,846	$28,333	1.64%
Road Tire Tube	2,173	$17,265	1.55%
Sport-100 Helmet, Red	2,099	$73,444	3.33%
AWC Logo Cap	2,062	$35,865	1.11%
Sport-100 Helmet, Blue	1,995	$67,112	3.31%
Fender Set - Mountain	1,975	$87,041	1.36%
Sport-100 Helmet, Black	1,940	$65,262	2.68%
Mountain Bottle Cage	1,896	$38,062	2.02%
Total	**15,587**	**$465,644**	**1.85%**

For this value in the matrix (2,846), the Orders measure is calculated based on the following filter context: Products[Product Name] = "Mountain Tire Tube"

- This allows the measure to return the total order quantity for each product specifically (or whatever context the row and column labels dictate – years, countries, categories, customer names, etc.)

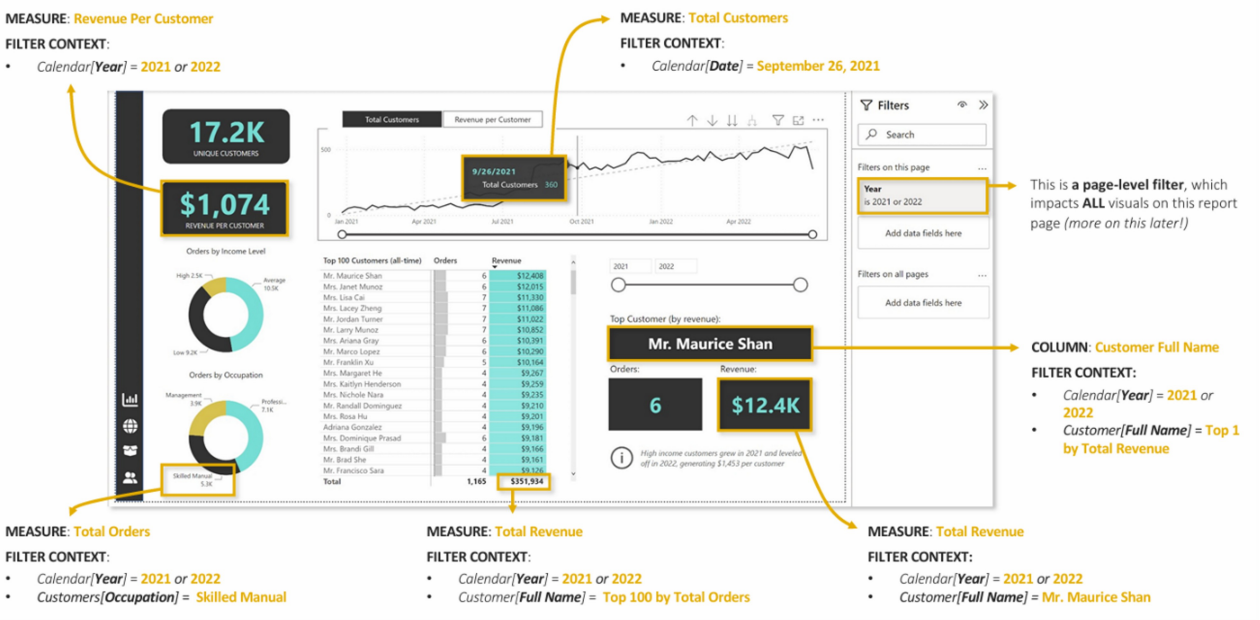
This total (15,587) does NOT calculate by summing the values above; it evaluates as an independent measure with no filter context applied

- IMPORTANT: Every measure value in a report evaluates independently (like an island) and calculates based on its own filter context

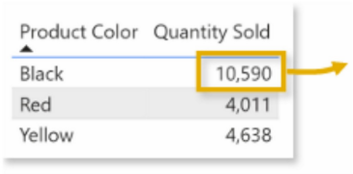


PRO TIP: Clicking the filter icon will show you the filters currently applied to a selected visual

EXAMPLE: FILTER CONTEXT



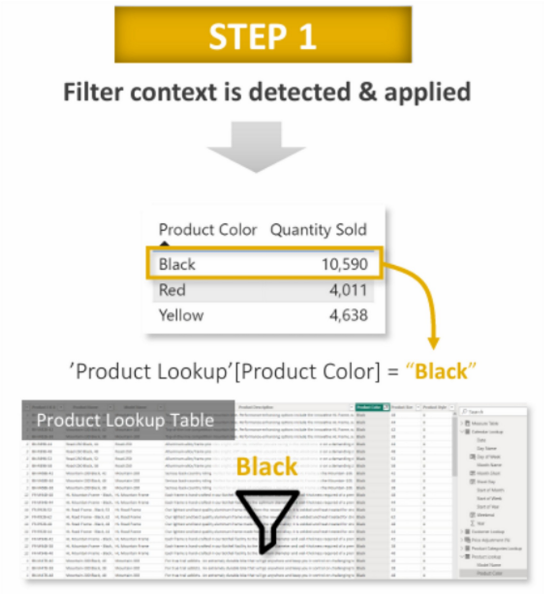
STEP-BY-STEP MEASURE CALCULATION

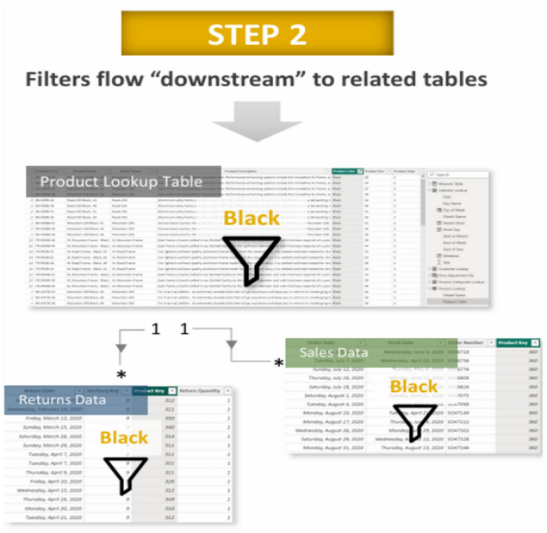


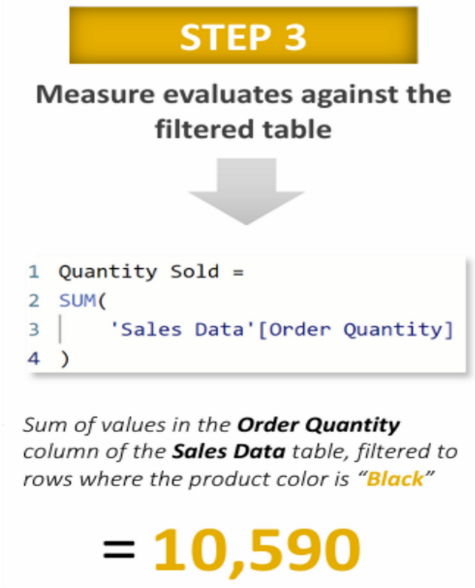
Product Color	Quantity Sold
Black	10,590
Red	4,011
Yellow	4,638

How exactly is this measure value calculated?

- NOTE: This all happens instantly behind the scenes, every time the filter context changes







DAX SYNTAX

MEASURE NAME

- Measures are always surrounded by brackets (i.e. [Total Quantity]) when referenced in formulas, so spaces are OK

Referenced TABLE NAME

Referenced COLUMN NAME

```
Total Quantity: = SUM(Transactions[quantity])
```

FUNCTION NAME

- Calculated columns don’t always use functions, but measures do:
  - In a Calculated Column, =Transactions[quantity] returns the value from the quantity column in each row (since it evaluates one row at a time).
  - In a Measure , = Transactions[quantity] will return an error since Power BI doesn’t know how to translate that as a single value – you need some sort of aggregation

This is a "fully qualified" column, since it’s preceded by the table name.

NOTE: Table names with spaces must be surrounded by single quotes:

- Without a space: Transactions[quantity]
- With a space: ’Transactions Table’[quantity]

PRO TIP:
Column references use fully qualified names (i.e. 'Table'[Column])

Measure references just use the measure name (i.e. [Measure]) and can be called by typing an open square bracket " ["

DAX OPERATORS

Arithmetic Operator	Meaning	Example
+	Addition	2 + 7
-	Subtraction	5 – 3
*	Multiplication	2 * 6
/	Division	4 / 2
^	Exponent	2 ^ 5

Comparison Operator	Meaning	Example
=	Equal to	[City]="Boston"
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
>=	Greater than or equal to	[Unit Price]>=2.5
<=	Less than or equal to	[Unit Price]<=2.5
!= <>	Not equal to	[Country]<>"Mexico"

"Important"

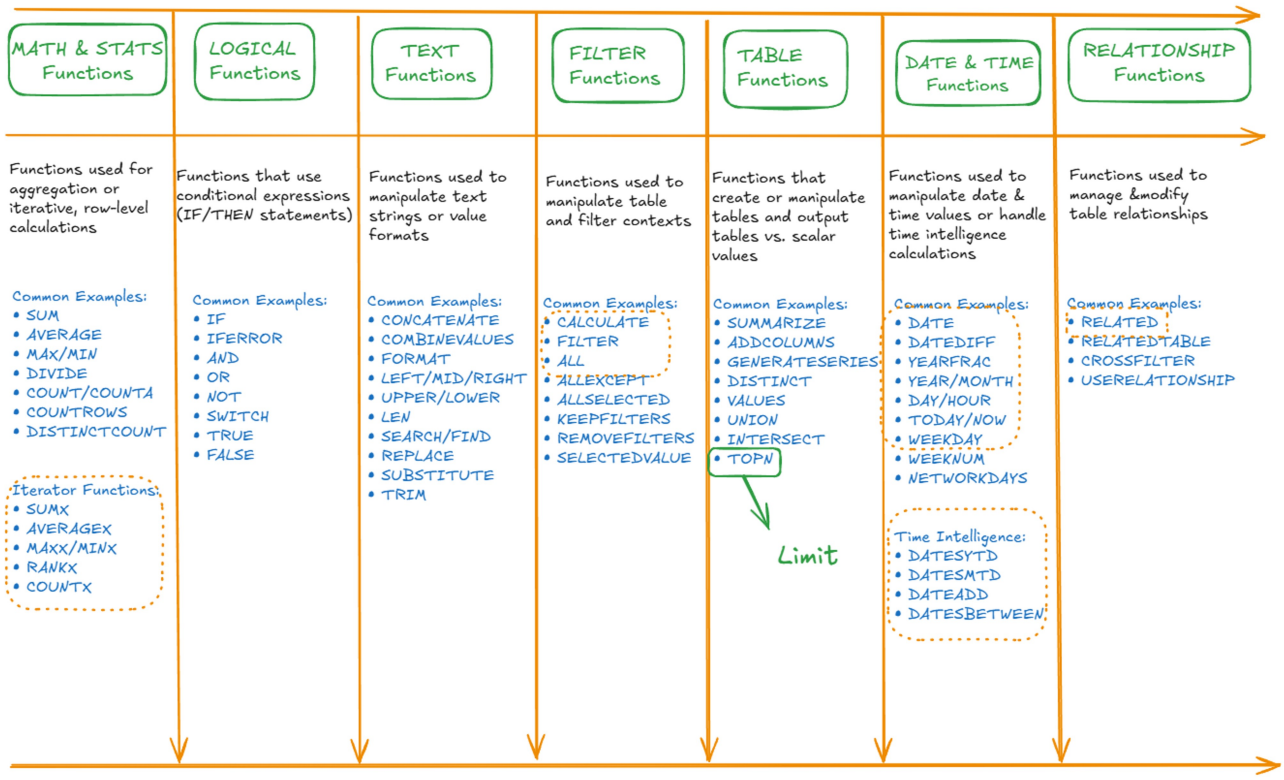
OR(logic1 , logic2)
AND(logic1 , logic2)

Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	[City] & " " & [State]
&&	Create an AND condition between two logical expressions	([State]="MA") && ([Quantity]>10)
|| *(double pipe)*	Create an OR condition between two logical expressions	([State]="MA") || ([State]="CT")
IN	Creates a logical OR condition based on a given list (using curly brackets)	'Store Lookup'[State] IN { "MA", "CT", "NY" }

SELECT * FROM Users WHERE Year = 2000 OR year = 2001
OR year = 2002 OR Year = 2003 OR year = 2004.

SELECT * FROM Users WHERE Year in (2000,2001,2002,2003,2004).

COMMON FUNCTION CATEGORIES



Set1 = {1,2,3,4,5,6}

Set2 = {4,6,7,8}

Set1 Union Set2 = {1,2,3,4,5,6,7,8}

Set1 intersect Set2 = {4,6}

10K 20K 30K 40K 50K 60K 70K 80K 90K 100K 110K 120K 130K 140K