CALCULATED FIELDS WITH DAX

CALCULATED FIELDS WITH DAX

In this section we'll use **Data Analysis Expressions (DAX)** to add calculated columns & measures to our model, and introduce topics like row & filter context, iterators and more

TOPICS WE'LL COVER:

- DAX 101
- Columns & Measures
- Row & Filter Context
- DAX Syntax
- Common Functions
- Calculate
- Iterators
- Time Intelligence

GOALS FOR THIS SECTION:

- Introduce DAX fundamentals and learn when to use calculated columns and measures
- Understand the difference between row context and filter context, and how they impact DAX calculations
- Learn DAX formula syntax, basic operators and common function categories *(math, logical, text, date/time, filter, etc.)*
- Explore nested functions, and more complex topics like iterators and time intelligence patterns

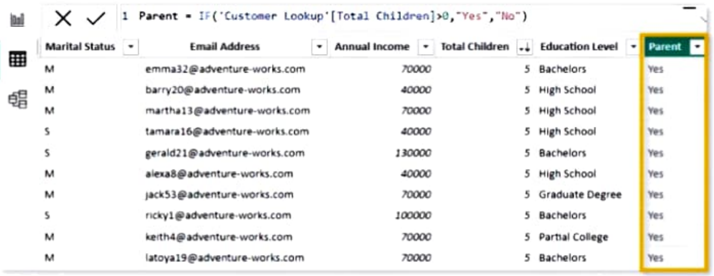
MEET DAX

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

Two ways to use DAX

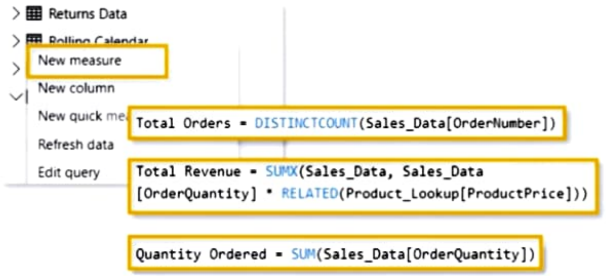
Calculated Columns



```
Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

Marital Status	Email Address	Annual Income	Total Children	Education Level	Parent
M	emma32@adventure-works.com	70000	5	Bachelors	Yes
M	barry20@adventure-works.com	40000	5	High School	Yes
M	martha13@adventure-works.com	70000	5	High School	Yes
S	tamara16@adventure-works.com	40000	5	High School	Yes
S	gerald21@adventure-works.com	130000	5	Bachelors	Yes
M	alexa8@adventure-works.com	40000	5	High School	Yes
M	jack53@adventure-works.com	70000	5	Graduate Degree	Yes
S	ricky1@adventure-works.com	100000	5	Bachelors	Yes
M	keith4@adventure-works.com	70000	5	Partial College	Yes
M	latoya19@adventure-works.com	70000	5	Bachelors	Yes

Measures



```
Total Orders = DISTINCTCOUNT(Sales_Data[OrderNumber])

Total Revenue = SUMX(Sales_Data, Sales_Data
[OrderQuantity] * RELATED(Product_Lookup[ProductPrice]))

Quantity Ordered = SUM(Sales_Data[OrderQuantity])
```

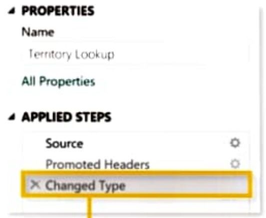
M VS. DAX

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

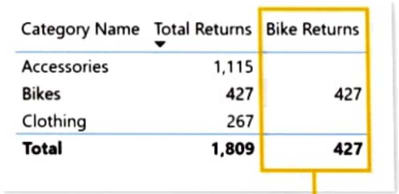
Query Editor:



```
#"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

Report View:



Category Name	Total Returns	Bike Returns
Accessories	1,115	
Bikes	427	427
Clothing	267	
Total	**1,809**	**427**

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

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!

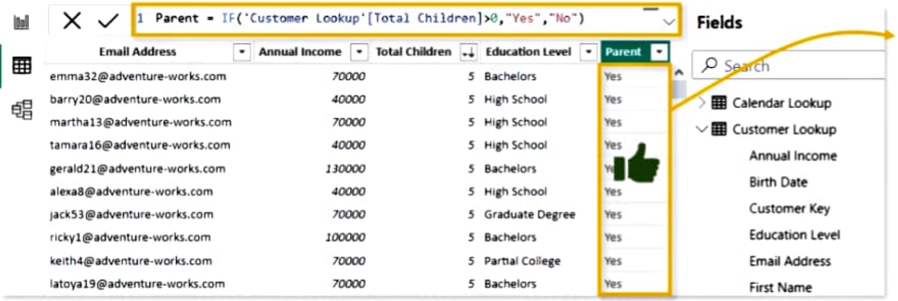
As a rule of thumb, use calculated columns to "stamp" static, fixed values to each row in a table (*or go upstream and use the Query Editor!*)

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



```
Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

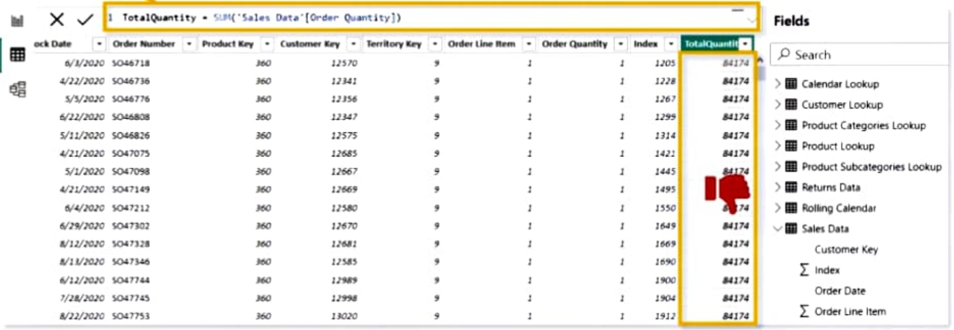
Email Address	Annual Income	Total Children	Education Level	Parent
emma32@adventure-works.com	70000	5	Bachelors	Yes
barry20@adventure-works.com	40000	5	High School	Yes
martha13@adventure-works.com	70000	5	High School	Yes
tamara16@adventure-works.com	40000	5	High School	Yes
gerald21@adventure-works.com	130000	5	Bachelors	Ye
alexa8@adventure-works.com	40000	5	High School	Yes
jack53@adventure-works.com	70000	5	Graduate Degree	Yes
ricky1@adventure-works.com	100000	5	Bachelors	Yes
keith4@adventure-works.com	70000	5	Partial College	Yes
latoya19@adventure-works.com	70000	5	Bachelors	Yes

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 **TotalQuantity**

- 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.



```
TotalQuantity = SUM('Sales Data'[Order Quantity])
```

ock Date	Order Number	Product Key	Customer Key	Territory Key	Order Line Item	Order Quantity	Index	TotalQuantit
6/3/2020	SO46718	360	12570	9	1	1	1205	84174
4/22/2020	SO46736	360	12341	9	1	1	1228	84174
5/5/2020	SO46776	360	12356	9	1	1	1267	84174
6/22/2020	SO46808	360	12347	9	1	1	1299	84174
5/11/2020	SO46826	360	12575	9	1	1	1314	84174
4/21/2020	SO47075	360	12685	9	1	1	1421	84174
5/1/2020	SO47098	360	12667	9	1	1	1445	84174
4/21/2020	SO47149	360	12669	9	1	1	1495	[illegible]
6/4/2020	SO47212	360	12580	9	1	1	1550	[illegible]
6/29/2020	SO47302	360	12670	9	1	1	1649	84174
8/12/2020	SO47328	360	12681	9	1	1	1669	84174
8/13/2020	SO47346	360	12585	9	1	1	1690	84174
6/12/2020	SO47744	360	12989	9	1	1	1900	84174
7/28/2020	SO47745	360	12998	9	1	1	1904	84174
8/22/2020	SO47753	360	13020	9	1	1	1912	84174

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

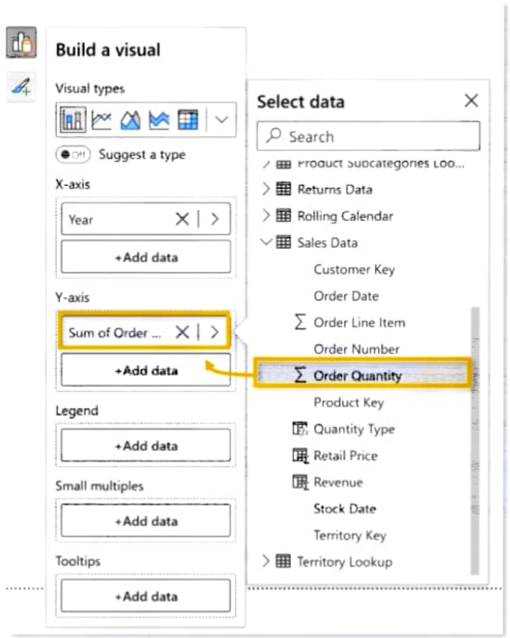
HEY THIS IS IMPORTANT!

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

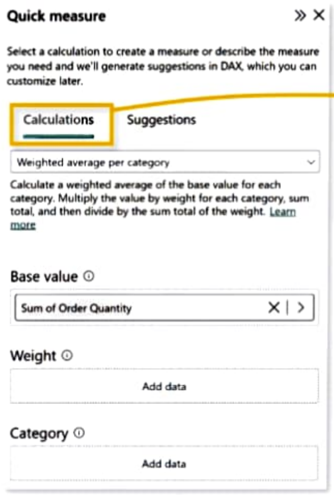
HEY THIS IS IMPORTANT!

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"

QUICK MEASURES

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



Quick measure

Select a calculation to create a measure or describe the measure you need and we'll generate suggestions in DAX, which you can customize later.

Weighted average per category

Calculate a weighted average of the base value for each category. Multiply the value by weight for each category, sum total, and then divide by the sum total of the weight. Learn more

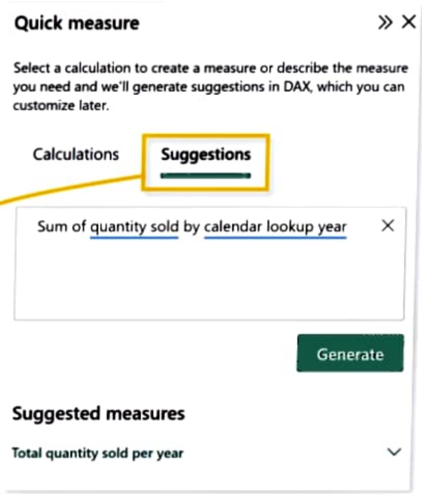
Base value: Sum of Order Quantity

Weight: Add data

Category: Add data

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")*



Quick measure

Select a calculation to create a measure or describe the measure you need and we'll generate suggestions in DAX, which you can customize later.

Sum of quantity sold by calendar lookup year

Suggested measures

Total quantity sold per 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**!

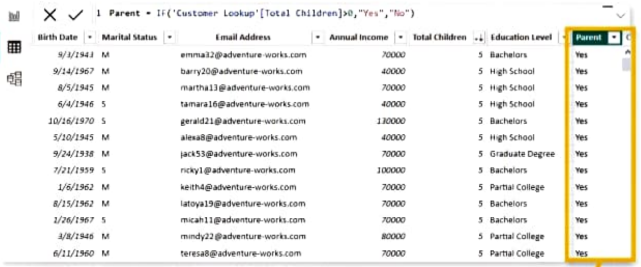
RECAP: CALCULATED COLUMNS VS. MEASURES

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

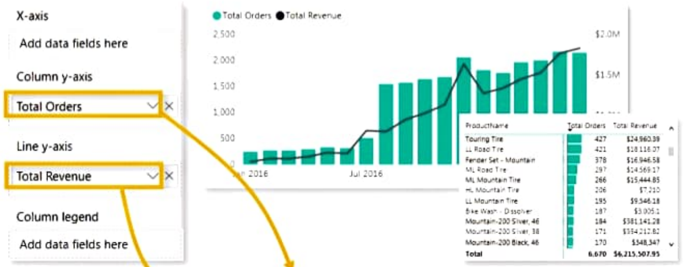
- Values are calculated based on information from any filters in the report (**filter context**)
- Does not create new data in the tables themselves (*doesn't increase file size*)
- Recalculate in response to any change to filters within the report
- Primarily used for **aggregating values** in report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



```
1 Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

Birth Date	Marital Status	Email Address	Annual Income	Total Children	Education Level	Parent
9/3/1943	M	emma32@adventure-works.com	70000	5	Bachelors	Yes
9/14/1967	M	barry20@adventure-works.com	40000	5	High School	Yes
8/5/1945	M	martha13@adventure-works.com	70000	5	High School	Yes
6/4/1946	S	tamara16@adventure-works.com	40000	5	High School	Yes
10/16/1970	S	gerald21@adventure-works.com	130000	5	Bachelors	Yes
5/10/1945	M	alexa8@adventure-works.com	40000	5	High School	Yes
9/24/1938	M	jack53@adventure-works.com	70000	5	Graduate Degree	Yes
7/21/1959	S	ricky1@adventure-works.com	100000	5	Bachelors	Yes
1/6/1962	M	keith4@adventure-works.com	70000	5	Partial College	Yes
8/15/1962	M	latoya19@adventure-works.com	70000	5	Bachelors	Yes
1/26/1967	S	micah11@adventure-works.com	70000	5	Bachelors	Yes
3/8/1946	M	mindy22@adventure-works.com	80000	5	Partial College	Yes
6/11/1960	M	teresa8@adventure-works.com	70000	5	Partial College	Yes

Calculated columns "live" in ***tables***



X-axis
Add data fields here
Column y-axis
Total Orders
Line y-axis
Total Revenue
Column legend
Add data fields here

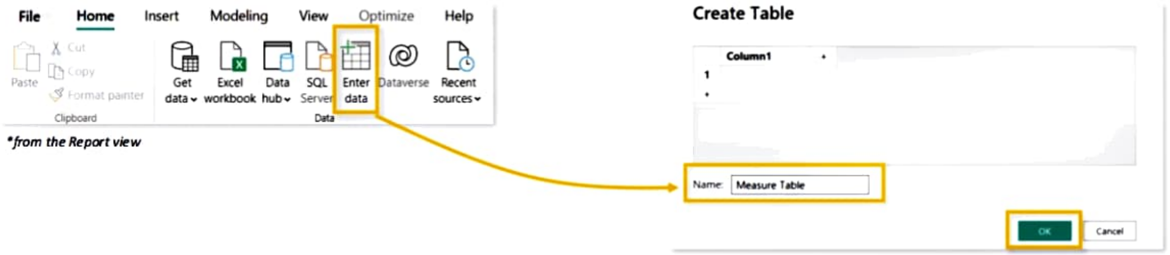
ProductName	Total Orders	Total Revenue
Touring Tire	427	$24,960.39
LL Road Tire	421	$18,116.07
Fender Set - Mountain	378	$16,946.58
ML Road Tire	297	$14,569.17
ML Mountain Tire	266	$15,444.85
HL Mountain Tire	206	$7,210
LL Mountain Tire	195	$9,546.18
Bike Wash - Dissolver	187	$3,005.1
Mountain-200 Silver, 46	184	$381,141.28
Mountain-200 Silver, 38	171	$354,212.82
Mountain-200 Black, 46	170	$348,347
Total	**6,670**	**$6,215,507.95**

Measures "live" in ***visuals***

PRO TIP: MEASURE TABLES

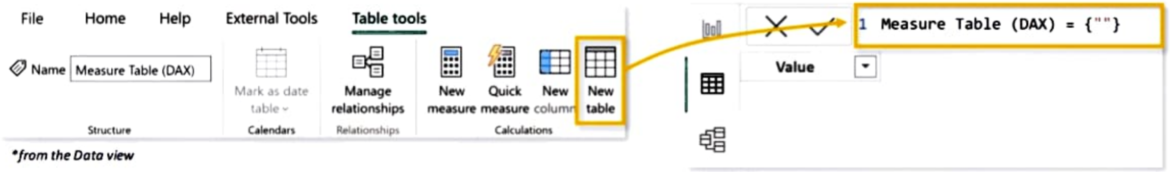
It's a common best practice to **create a dedicated table to store your measures**; this will help you stay organized, find measures quickly, and allow you to group related measures into folders

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)

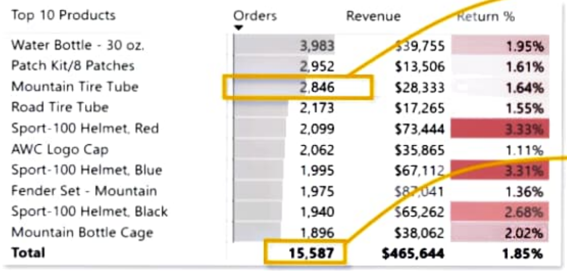


```
1 Measure Table (DAX) = {""}
```

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%**

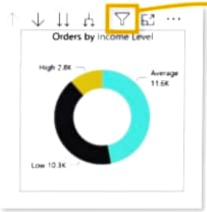
For this value in the matrix (2,846), the **Orders** measure is calculated based on the following filter context: *Products[**Product Name**] = "**Mountain Tire Tube**"*

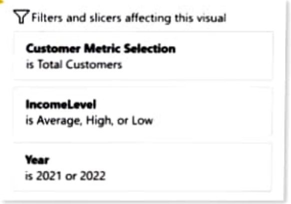
- This allows the measure to return the total order quantity for each product specifically *(or whatever context the row and column labels dictate – years, countries, categories, customer names, etc.)*

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





Filters and slicers affecting this visual

Customer Metric Selection
is Total Customers

IncomeLevel
is Average, High, or Low

Year
is 2021 or 2022

EXAMPLE: FILTER CONTEXT

MEASURE: Revenue Per Customer

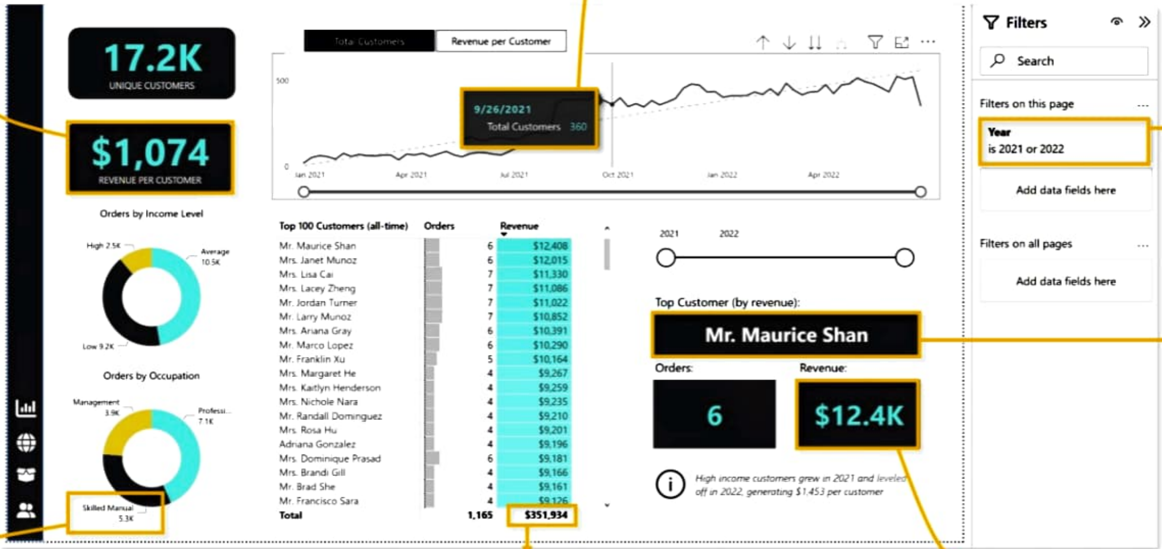
FILTER CONTEXT:

- *Calendar[**Year**]* = 2021 *or* 2022

MEASURE: Total Customers

FILTER CONTEXT:

- *Calendar[**Date**]* = September 26, 2021



Top 100 Customers (all-time)	Orders	Revenue
Mr. Maurice Shan	6	$12,408
Mrs. Janet Munoz	6	$12,015
Mrs. Lisa Cai	7	$11,330
Mrs. Lacey Zheng	7	$11,086
Mr. Jordan Turner	7	$11,022
Mr. Larry Munoz	7	$10,852
Mrs. Ariana Gray	6	$10,391
Mr. Marco Lopez	6	$10,290
Mr. Franklin Xu	5	$10,164
Mrs. Margaret He	4	$9,267
Mrs. Kaitlyn Henderson	4	$9,259
Mrs. Nichole Nara	4	$9,235
Mr. Randall Dominguez	4	$9,210
Mrs. Rosa Hu	4	$9,201
Adriana Gonzalez	4	$9,196
Mrs. Dominique Prasad	6	$9,181
Mrs. Brandi Gill	4	$9,166
Mr. Brad She	4	$9,161
Mr. Francisco Sara	4	$9,126
Total	**1,165**	**$351,934**

High income customers grew in 2021 and leveled off in 2022, generating $1,453 per customer

This is **a page-level filter**, which impacts **ALL** visuals on this report page *(more on this later!)*

COLUMN: Customer Full Name

FILTER CONTEXT:

- *Calendar[**Year**]* = 2021 *or* 2022
- *Customer[**Full Name**]* = Top 1 by Total Revenue

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[**Year**]* = 2021 *or* 2022
- *Customers[**Occupation**]* = Skilled Manual

MEASURE: Total Revenue

FILTER CONTEXT:

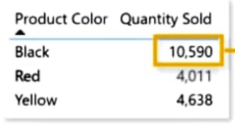
- *Calendar[**Year**]* = 2021 *or* 2022
- *Customer[**Full Name**]* = Top 100 by Total Orders

MEASURE: Total Revenue

FILTER CONTEXT:

- *Calendar[**Year**]* = 2021 *or* 2022
- *Customer[**Full Name**]* = Mr. Maurice Shan

STEP-BY-STEP MEASURE CALCULATION



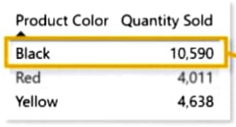
Product Color	Quantity Sold
Black	10,590
Red	4,011
Yellow	4,638

How *exactly* is this measure value calculated?

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

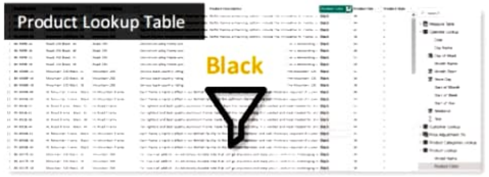
STEP 1

Filter context is detected & applied



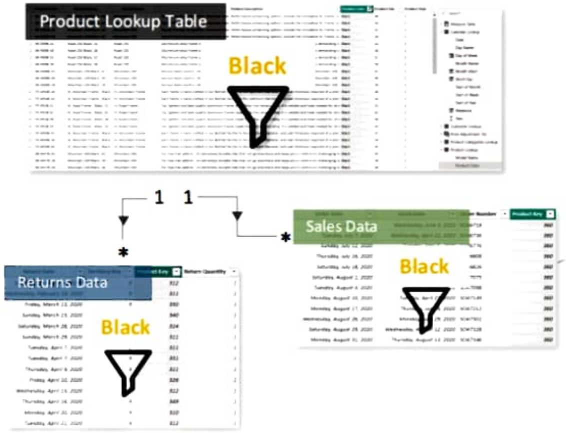
Product Color	Quantity Sold
Black	10,590
Red	4,011
Yellow	4,638

'Product Lookup'[Product Color] = "Black"



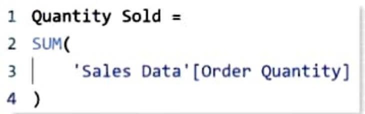
STEP 2

Filters flow "downstream" to related tables



STEP 3

Measure evaluates against the filtered table



```
Quantity Sold =
SUM(
    'Sales Data'[Order Quantity]
)
```

*Sum of values in the **Order Quantity** column of the **Sales Data** table, filtered to rows where the product color is "Black"*

= 10,590

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"

Pay attention to these!

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" }

Head to* ***https://learn.microsoft.com* *for more information about DAX syntax, operators, troubleshooting, etc.*

COMMON FUNCTION CATEGORIES

MATH & STATS Functions

*Functions used for **aggregation** or iterative, row-level calculations*

Common Examples:

- SUM
- AVERAGE
- MAX/MIN
- DIVIDE
- COUNT/COUNTA
- COUNTROWS
- DISTINCTCOUNT

Iterator Functions:

- SUMX
- AVERAGEX
- MAXX/MINX
- RANKX
- COUNTX

LOGICAL Functions

*Functions that use **conditional expressions** (IF/THEN statements)*

Common Examples:

- IF
- IFERROR
- AND
- OR
- NOT
- SWITCH
- TRUE
- FALSE

TEXT Functions

*Functions used to manipulate **text strings** or **value formats***

Common Examples:

- CONCATENATE
- COMBINEVALUES
- FORMAT
- LEFT/MID/RIGHT
- UPPER/LOWER
- LEN
- SEARCH/FIND
- REPLACE
- SUBSTITUTE
- TRIM

FILTER Functions

*Functions used to **manipulate table** and **filter contexts***

Common Examples:

- CALCULATE
- FILTER
- ALL
- ALLEXCEPT
- ALLSELECTED
- KEEPFILTERS
- REMOVEFILTERS
- SELECTEDVALUE

TABLE Functions

*Functions that **create** or **manipulate tables** and output tables vs. scalar values*

Common Examples:

- SUMMARIZE
- ADDCOLUMNS
- GENERATESERIES
- DISTINCT
- VALUES
- UNION
- INTERSECT
- TOPN

DATE & TIME Functions

*Functions used to manipulate **date & time values** or handle time intelligence calculations*

Common Examples:

- DATE
- DATEDIFF
- YEARFRAC
- YEAR/MONTH
- DAY/HOUR
- TODAY/NOW
- WEEKDAY
- WEEKNUM
- NETWORKDAYS

Time Intelligence:

- DATESYTD
- DATESMTD
- DATEADD
- DATESBETWEEN

RELATIONSHIP Functions

*Functions used to **manage & modify table relationships***

Common Examples:

- RELATED
- RELATEDTABLE
- CROSSFILTER
- USERELATIONSHIP

***Note**: *This is NOT a comprehensive list. DAX contains more than 250 different functions!*

BASIC MATH & STATS FUNCTIONS

SUM	Evaluates the sum of a column	=SUM(ColumnName)
AVERAGE	Returns the average (arithmetic mean) of all the numbers in a column	=AVERAGE(ColumnName)
MAX	Returns the largest value in a column or between two scalar expressions	=MAX(ColumnNameOrScalar1, *[Scalar2]*)
MIN	Returns the smallest value in a column or between two scalar expressions	=MIN(ColumnNameOrScalar1, *[Scalar2]*)
DIVIDE	Performs division and returns the alternate result (or blank) if DIV/0	=DIVIDE(Numerator, Denominator, *[AlternateResult]*

COUNTING FUNCTIONS

COUNT	Counts the number of non-empty cells in a column (excluding Boolean values)	=COUNT(ColumnName)
COUNTA	Counts the number of non-empty cells in a column (including Boolean values)	=COUNTA(ColumnName)
DISTINCTCOUNT	Counts the number of distinct values in a column	=DISTINCTCOUNT(ColumnName)
COUNTROWS	Counts the number of rows in the specified table, or a table defined by an expression	=COUNTROWS([Table])

ASSIGNMENT: MATH & STATS

NEW MESSAGE

From: **Dianne A. Xu** *(Senior Analyst)*

Subject: **Help with a few measures**

Hey there, excited to start working with you!

I'll need to pull some high-level metrics from our model to share with leadership, and I could use some help with the calculations.

For now, could you please create one measure to calculate the total number of distinct customers, and a second measure that we can use to calculate return rate (quantity returned / quantity sold)? Thank you!

-Dianne

Reply Forward

Key Objectives

1. Create a measure named **Total Customers**, to calculate the number of distinct AdventureWorks customers who made a transaction
2. Create a measure named **Return Rate**, defined as quantity returned divided by quantity sold

SOLUTION: MATH & STATS

NEW MESSAGE

From: **Dianne A. Xu** *(Senior Analyst)*

Subject: **Help with a few measures**

Hey there, excited to start working with you!

I'll need to pull some high-level metrics from our model to share with leadership, and I could use some help with the calculations.

For now, could you please create one measure to calculate the total number of distinct customers, and a second measure that we can use to calculate return rate (quantity returned / quantity sold)? Thank you!

-Dianne

Reply   Forward

Solution Preview

```
Total Customers =
DISTINCTCOUNT(
        'Sales Data'[Customer Key]
)
```

```
Return Rate =
DIVIDE(
        [Quantity Returned],
        [Quantity Sold],
        "No Sales"
)
```