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

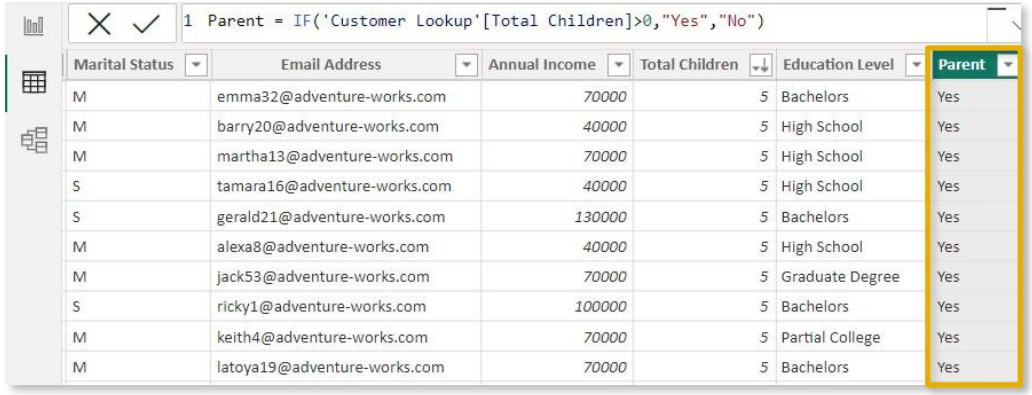
MEET DAX

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

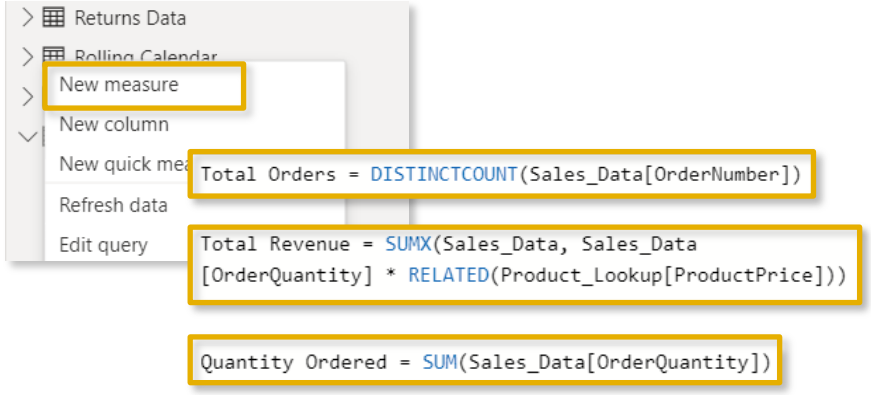
- Go beyond the capabilities of traditional spreadsheet formulas, with powerful and flexible functions built specifically to work with relational data models
- Add **calculated columns** *(for filtering)* and **measures** *(for aggregation)* to enhance data models

Two ways to use DAX

Calculated Columns



Measures



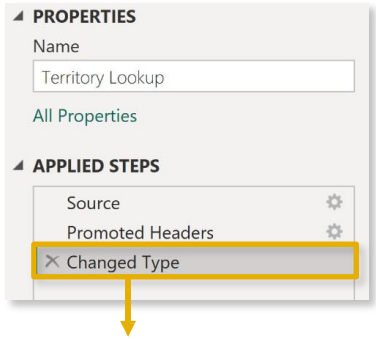
M VS. DAX

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**

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

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!

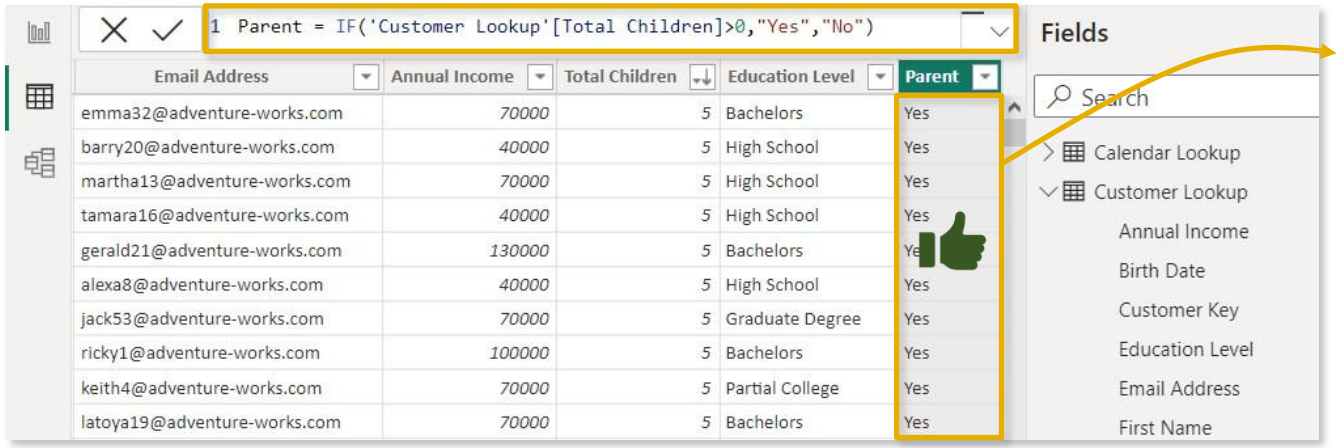
As a rule of thumb, use calculated columns to "stamp" static, fixed values to each row in a table (*or go upstream and use the Query Editor!*)

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

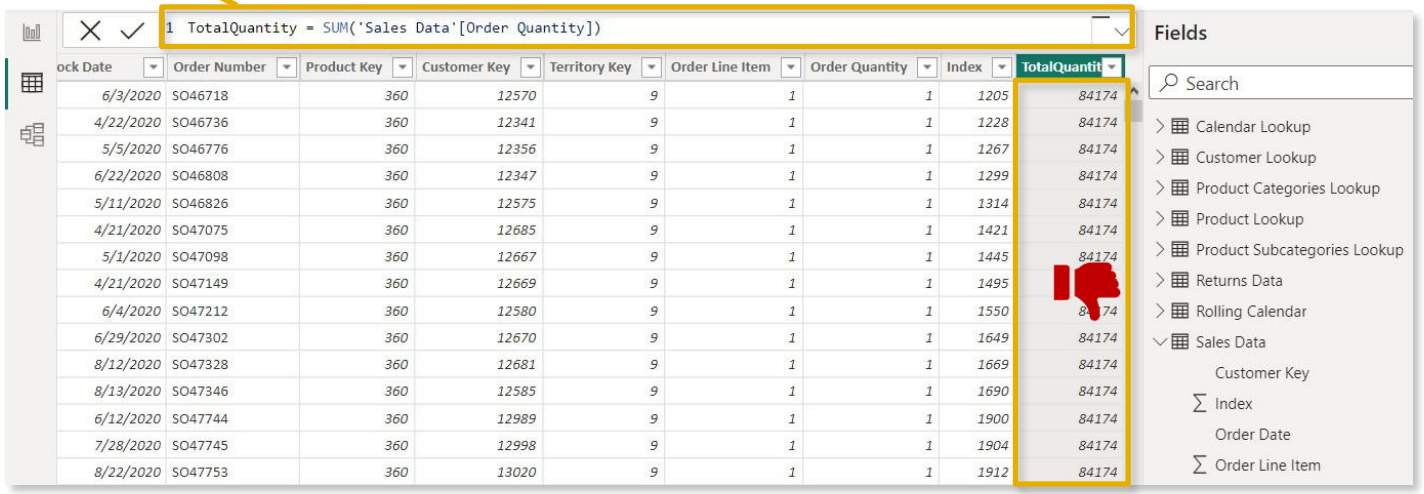


In this case we've added a **calculated column** named **Parent**, which equals "***Yes***" if the [Total Children] field is greater than 0, and "***No***" otherwise

- 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 **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.



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

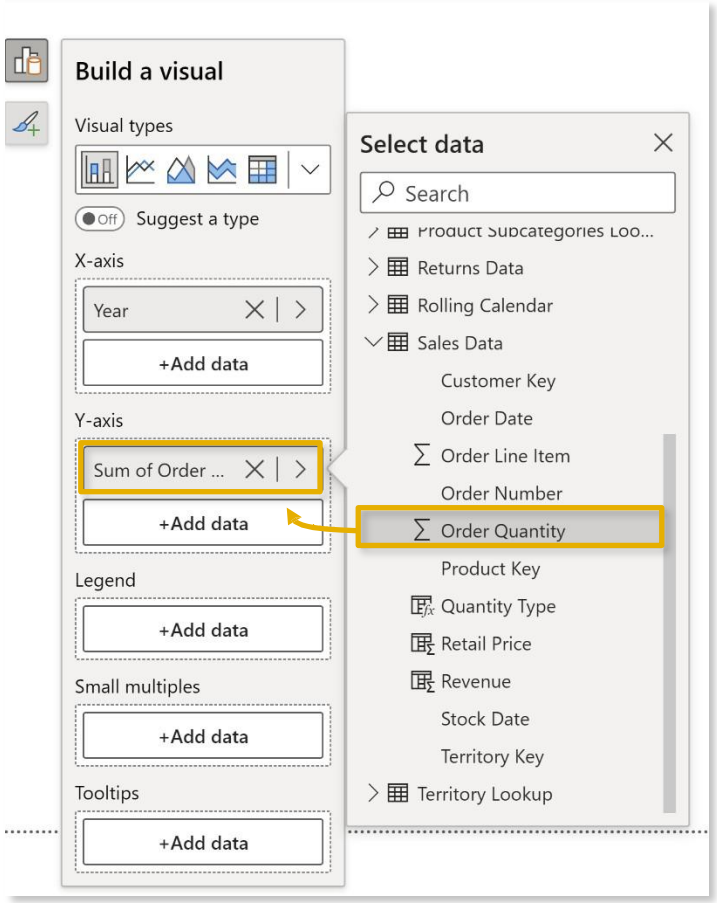
HEY THIS IS IMPORTANT!

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

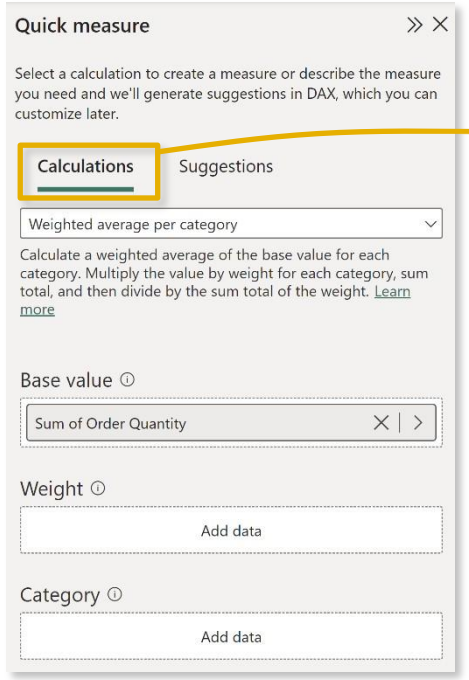
HEY THIS IS IMPORTANT!

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"

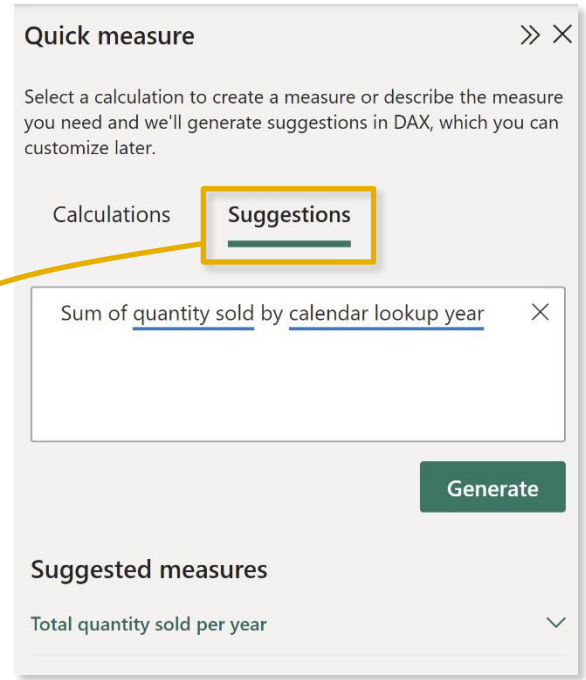
QUICK MEASURES

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



Quick measure **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")*



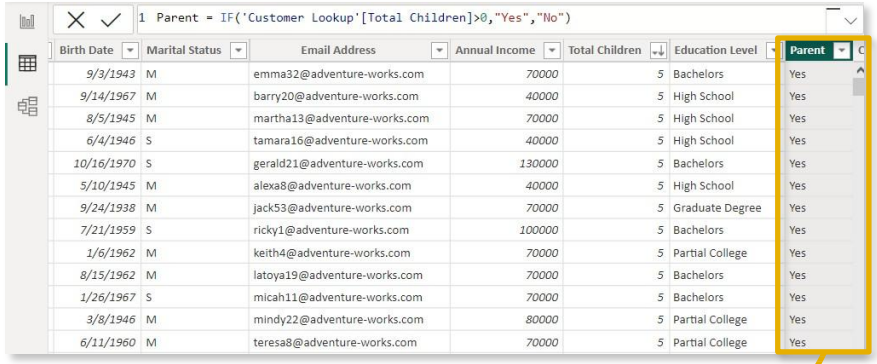
PRO TIP:

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

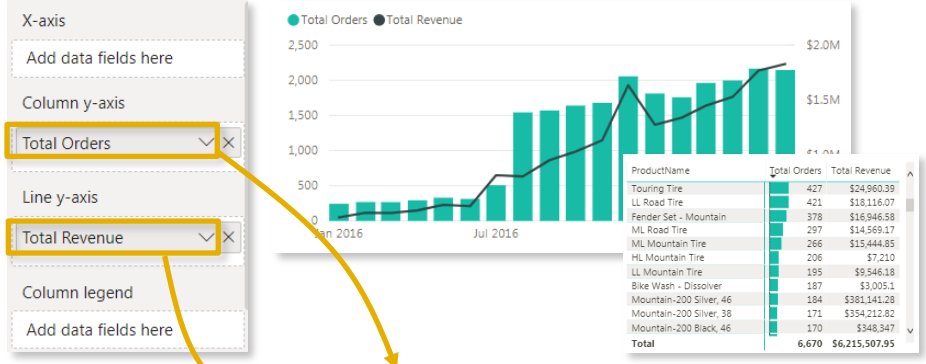
- Values are calculated based on information from each row of a table (**row context**)
- Appends static values to each row in a table and stores them in the model (*which increases file size*)
- Recalculate on data source refresh or when changes are made to component columns
- Primarily used for **filtering**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



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

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

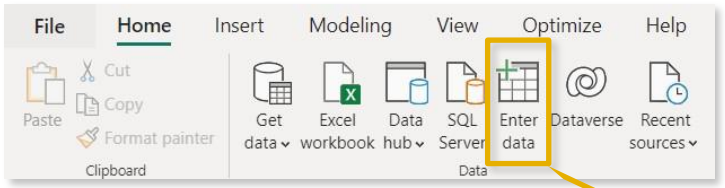


*Measures "live" in **visuals***

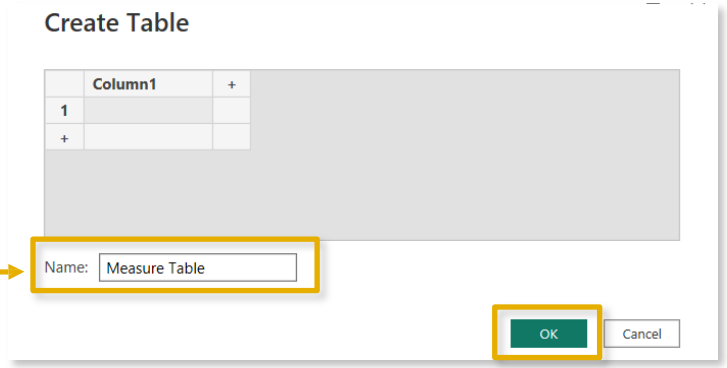
PRO TIP: MEASURE TABLES

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)

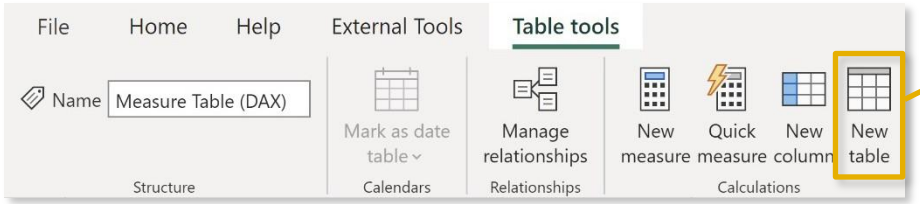


**from the Report view*

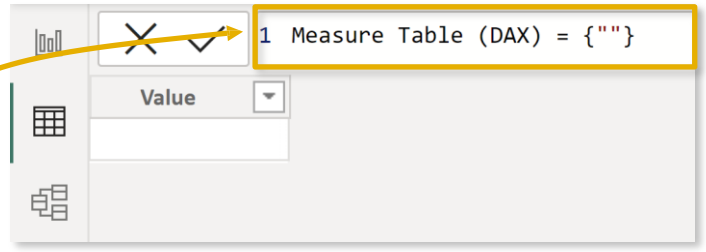


Add a table Name and **click OK** to load the table to the data model

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



**from the Data view*



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%**

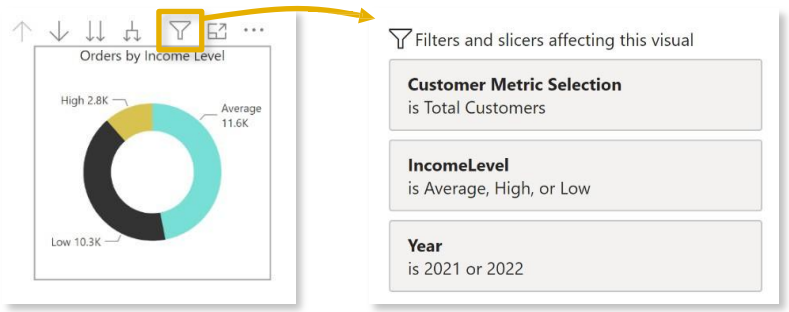
For this value in the matrix (2,846), the **Orders** measure is calculated based on the following filter context: *Products[**Product Name**] = "**Mountain Tire Tube**"*

- 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

MEASURE: Revenue Per Customer

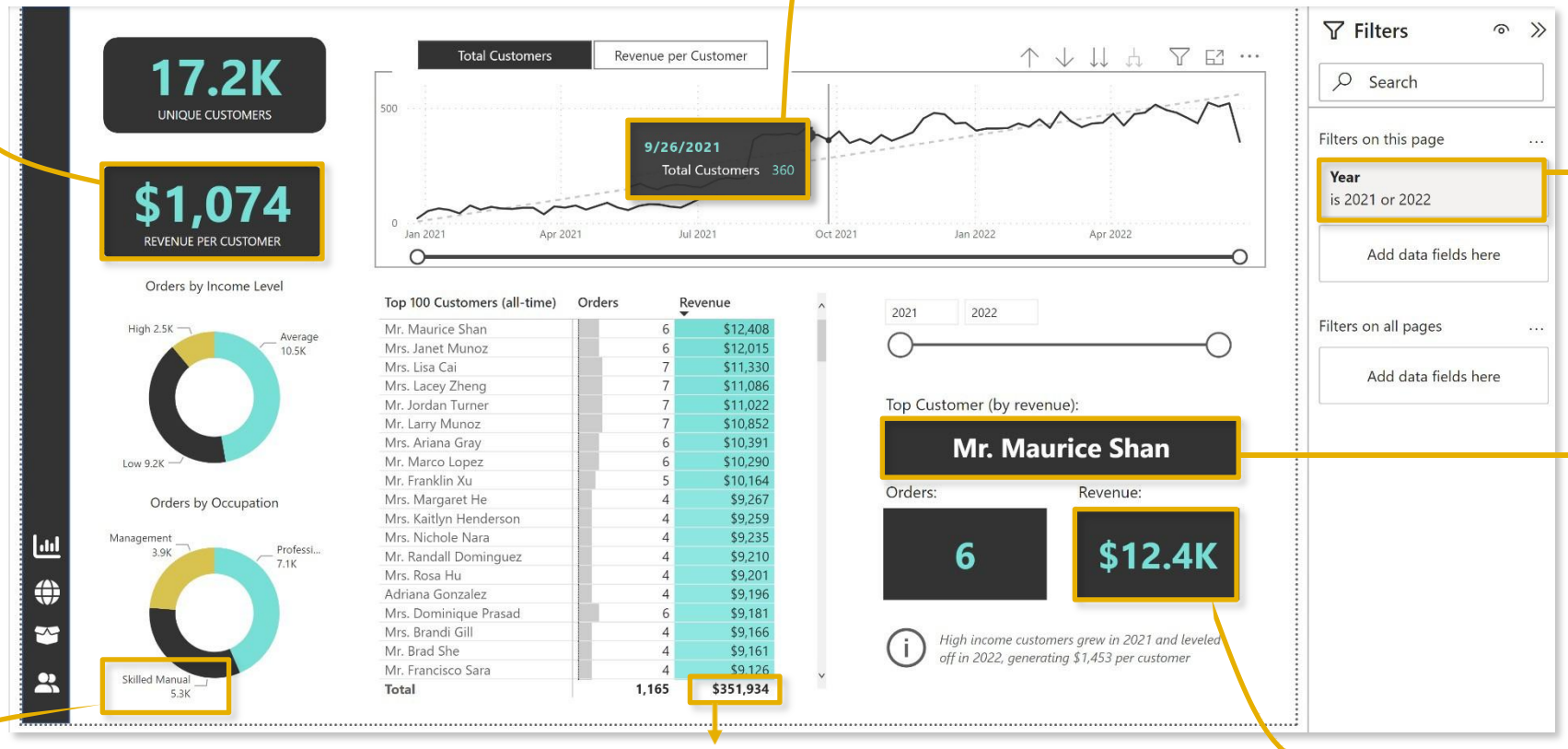
FILTER CONTEXT:

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

MEASURE: Total Customers

FILTER CONTEXT:

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



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?

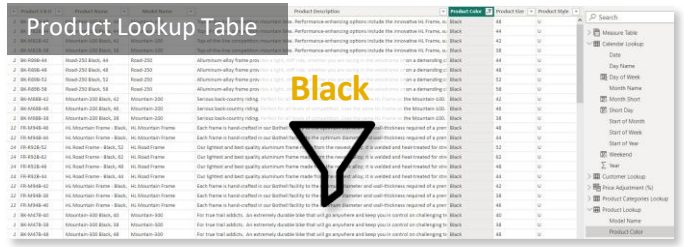
- **NOTE**: This all happens *instantly* behind the scenes, every time the filter context changes

STEP 1

Filter context is detected & applied

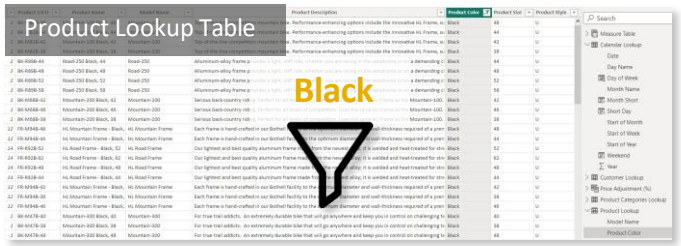
Product Color	Quantity Sold
Black	10,590
Red	4,011
Yellow	4,638

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



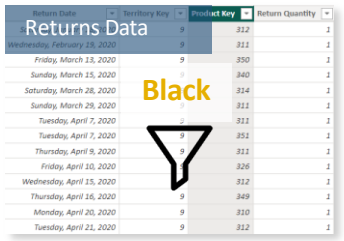
STEP 2

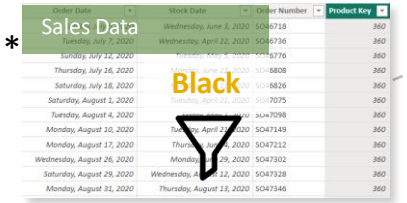
Filters flow "downstream" to related tables



1 1

* *





STEP 3

Measure evaluates against the filtered table

```
1 Quantity Sold =
2 SUM(
3     'Sales Data'[Order Quantity]
4 )
```

*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

```
1 Total Customers =
2 DISTINCTCOUNT(
3     'Sales Data'[Customer Key]
4 )
```

```
1 Return Rate =
2 DIVIDE(
3     [Quantity Returned],
4     [Quantity Sold],
5     "No Sales"
6 )
```

BASIC LOGICAL FUNCTIONS

IF	Checks if a given condition is met and returns one value if the condition is TRUE, and another if the condition is FALSE	=**IF**(LogicalTest, ResultIfTrue, *[ResultIfFalse]*)
IFERROR	Evaluates an expression and returns a specified value if it returns an error, otherwise returns the expression itself	=**IFERROR**(Value, ValueIfError)
SWITCH	Evaluates an expression against a list of values and returns one of multiple possible expressions	=**SWITCH**(Expression, Value1, Result1, …, *[Else]*)
AND	Checks whether both arguments are TRUE to return TRUE, otherwise returns FALSE	=**AND**(Logical1, Logical2)
OR	Checks whether any argument is TRUE to return TRUE, otherwise returns FALSE	=**OR**(Logical1, Logical2)

***Note**: Use the **&&** and **||** operators to include more than two conditions*

SWITCH

SWITCH Evaluates an expression against a list of values and returns one of multiple possible expressions

=**SWITCH**(Expression, Value1, Result1, …, *[Else]*)

Any **DAX expression** that returns a single scalar value, evaluated multiples times

Examples:

- *Calendar[Month ID]*
- *'Product Lookup'[category]*

List of **values** produced by the expression, each paired with a result to return for rows/cases that match

Examples:

=**SWITCH**(*Calendar[Month ID],*
1, "January",
2, "February"

Value returned if the expression doesn't match any value argument

PRO TIP

SWITCH(TRUE) is a common DAX pattern to replace multiple nested IF statements

ASSIGNMENT: LOGICAL FUNCTIONS

NEW MESSAGE

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

Subject: **Customer segmentation fields**

Hey there!

Ethan has been working with the DS team on a customer segmentation analysis, and came back to us with a few requests.

Could you please add some new columns in our customer table to identify "priority" customers, segment customers based on income level, and group some of the education categories?

I've attached the logic to use, but reach out with any questions!

-Dianne

Reply Forward

Key Objectives

1. Create a calculated column in the Customer Lookup table named **Customer Priority**:
   - If the customer is a parent and has an annual income > $100,000, Customer Priority = **Priority**
   - Otherwise, Customer Priority = **Standard**
2. Create a calculated column in the Customer Lookup table named **Income Level**:
   - If annual income is >= $150,000, **Very High**
   - If annual income is >= $100,000, **High**
   - If annual income is >= $50,000, **Average**
   - Otherwise, Income Level = **Low**

ASSIGNMENT: LOGICAL FUNCTIONS

NEW MESSAGE

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

Subject: **Customer segmentation fields**

Hey there!

Ethan has been working with the DS team on a customer segmentation analysis, and came back to us with a few requests.

Could you please add some new columns in our customer table to identify "priority" customers, segment customers based on income level, and group some of the education categories?

I've attached the logic to use, but reach out with any questions!

-Dianne

Reply Forward

Key Objectives

BONUS: Use a SWITCH function* to create another column named **Education Category**:

- If EducationLevel is High School or Partial High School, Education Category **= High School**
- If EducationLevel is Bachelors or Partial College, Education Category = **Undergrad**
- If EducationLevel is Graduate Degree, Education Category = **Graduate**

**You can use the "data groups" tool to do this too!*

SOLUTION: LOGICAL FUNCTIONS

NEW MESSAGE

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

Subject: **Customer segmentation fields**

Hey there!

Ethan has been working with the DS team on a customer segmentation analysis, and came back to us with a few requests.

Could you please add some new columns in our customer table to identify “priority” customers, segment customers based on income level, and group some of the education categories?

I’ve attached the logic to use, but reach out with any questions!

-Dianne

Reply Forward

Solution Preview

```
Customer Priority =
IF(
    'Customer Lookup'[AnnualIncome] > 100000 &&
    'Customer Lookup'[Is Parent?] = "Yes",
    "Priority",
    "Standard"
)
```

```
Income Level =
IF('Customer Lookup'[AnnualIncome] >= 150000, "Very High",
IF('Customer Lookup'[AnnualIncome] >= 100000, "High",
IF('Customer Lookup'[AnnualIncome] >= 50000, "Average",
"Low")))
```

```
Education Category =
SWITCH('Customer Lookup'[EducationLevel],
"High School","High School",
"Partial High School","High School",
"Bachelors","Undergrad",
"Partial College","Undergrad",
"Graduate Degree","Graduate")
```

TEXT FUNCTIONS

LEN	Returns the number of characters in a string	=**LEN**(Text)
CONCATENATE	Joins two text strings into one	=**CONCATENATE**(Text1, Text2)
UPPER/LOWER	Converts a string to upper or lower case	=**UPPER/LOWER** (Text)
LEFT/RIGHT/MID	Returns a number of characters from the start/middle/end of a text string	=**LEFT/RIGHT**(Text, *[NumChars]*) =**MID**(Text, StartPosition, NumChars)
SUBSTITUTE	Replaces an instance of existing text with new text in a string	=**SUBSTITUTE**(Text, OldText, NewText, *[InstanceNumber]*)
SEARCH	Returns the position where a specified string or character is found, reading left to right	=**SEARCH**(FindText, WithinText, *[StartPosition], [NotFoundValue]*)

***Note**: Use the **&** operator as a shortcut, or to combine more than two strings*

ASSIGNMENT: TEXT

NEW MESSAGE

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

Subject: **Couple random requests**

Good morning!

Hoping you can help with a couple quick updates to the model:

1) Ethan wants to make the month abbreviations ALL CAPS to make them more readable in our reports.

2) The product team asked us to break out the SKU category into its own field, which we can define as any characters before the first hyphen ("-") in the ProductSKU column.

Thanks, reach out with any questions!

Reply Forward

Key Objectives

1. Update the **Month Short** column in the Calendar Lookup table to extract and capitalize the first 3 characters of the month name

2. Create a new column in the Product Lookup table named **SKU Category**, to return any number of characters before the first hyphen in the ProductSKU column

SOLUTION: TEXT

NEW MESSAGE

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

Subject: **Couple random requests**

Good morning!

Hoping you can help with a couple quick updates to the model:

1) Ethan wants to make the month abbreviations ALL CAPS to make them more readable in our reports.

2) The product team asked us to break out the SKU category into its own field, which we can define as any characters before the first hyphen (“-”) in the ProductSKU column.

Thanks, reach out with any questions!

Reply Forward

Solution Preview

```
1 Month Short =
2 UPPER(
3     LEFT(
4         'Calendar Lookup'[Month Name],
5         3
6     )
7 )
```

```
1 SKU Category =
2 LEFT(
3     'Product Lookup'[Product SKU],
4     SEARCH(
5         "-",
6         'Product Lookup'[Product SKU]
7     )
8     -1
9 )
```

BASIC DATE & TIME FUNCTIONS

TODAY/NOW	Returns the current date or exact time	=**TODAY/NOW**()
DAY/MONTH/YEAR	Returns the day of the month (1-31), month of the year (1-12), or year of a given date	=**DAY/MONTH/YEAR**(Date)
HOUR/MINUTE/ SECOND	Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value	=**HOUR/MINUTE/SECOND**(Datetime)
WEEKDAY/ WEEKNUM	Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year	=**WEEKDAY/WEEKNUM**(Date, *[ReturnType]*)
EOMONTH	Returns the date of the last day of the month, +/- a specified number of months	=**EOMONTH**(StartDate, Months)
DATEDIFF	Returns the difference between two dates, based on a given interval (day, hour, year, etc.)	=**DATEDIFF**(Date1, Date2, Interval)

ASSIGNMENT: DATE & TIME

NEW MESSAGE

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

Subject: **Customer birth years**

Hey there, super easy one for you.

The customer segmentation project got me wondering if there are any interesting patterns or insights based on customer age.

Could you please add a field in our customer table to extract only the year from the birthdate field?

Thanks!
-Dianne

Reply Forward

Key Objectives

1. Create a new column in the Customer Lookup table named **Birth Year**, to extract only the year from the BirthDate column

SOLUTION: DATE & TIME

NEW MESSAGE

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

Subject: **Customer birth years**

Hey there, super easy one for you.

The customer segmentation project got me wondering if there are any interesting patterns or insights based on customer age.

Could you please add a field in our customer table to extract only the year from the birthdate field?

Thanks!
-Dianne

Reply Forward

Solution Preview

```
1 Birth Year =
2 YEAR(
3     'Customer Lookup'[BirthDate]
4 )
```

RELATED

RELATED() Returns related values in each row of a table based on relationships with other tables

=**RELATED**(ColumnName)

The **column** from a related table containing the values you want to retrieve

Examples:

- *'Product Lookup'[Product Name]*
- *'Territory Lookup'[Country]*

HEY THIS IS IMPORTANT!

RELATED works like a **VLOOKUP** function in Excel – it uses the relationship between tables (*defined by primary and foreign keys*) to pull values from one table into a new column of another.

Since this function requires row context, it can only be used as a **calculated column** or as part of an **iterator function** that cycles through all rows in a table (*FILTER, SUMX, MAXX, etc.*)

PRO TIP:

Instead of using RELATED to create extra columns (which increases file size), **nest it within measures like FILTER or SUMX**

CALCULATE

CALCULATE()

Evaluates an expression in a context that is modified by filters

=**CALCULATE**(Expression, *[Filter1], [Filter2],…*)

Name of an **existing measure** or a **DAX formula** for a valid measure

Examples:

- *[Total Orders]*
- *SUM('Returns Data'[Return Quantity])*

A Boolean (True/False) expression or a table expression that defines a filter

Note: these require fixed values or aggregation functions that return a scalar value (you cannot create filters based on measures)

Examples:

- *'Territory Lookup'[Country] = **"USA"***
- *Calendar[Year] <> **MAX**(Calendar[Year])*

PRO TIP:

Think of CALCULATE as a **filter modifier**; it allows you to overrule existing report filters and "force" new filter context

EXAMPLE: CALCULATE

```
1 Red Sales = CALCULATE( [Quantity Sold], 'Product Lookup'[Product Color] = "Red" )
```

Here we've defined a new measure named **Red Sales**, which evaluates the **Quantity Sold** measure under a filter context where the product color is "**Red**"

Product Color	Quantity Sold	Red Sales
Black	10,590	4,011
Multi	5,756	4,011
Red	4,011	4,011
Silver	3,257	4,011
Total	**23,614**	**4,011**

*Note how we see the **the same repeated values** for each product color, and even the total!*

HEY THIS IS IMPORTANT!

The CALCULATE function modifies and overrules any competing filter context!

In this matrix, the "Black" row has competing filter context: Product Color = **Black** (from the row label) and Product Color= "**Red**" (from the CALCULATE function)

Both can't be true at the same time, so the "**Red**" filter from CALCULATE takes priority

EXAMPLE: CALCULATE

CALCULATE

Filters are modified by CALCULATE

[**Product Color**] = "Red"

If the measure being evaluated contains a **CALCULATE** function, filter context is *overwritten* between **Step 1** & **Step 2**

STEP 1

Filter context is detected & applied

Product Color	Quantity Sold	Red Sales
Black	10,590	4,011
Red	4,011	4,011
Silver	3,257	4,011

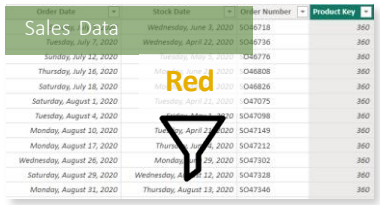
'Product Lookup'[**Product Color**] = "Black"

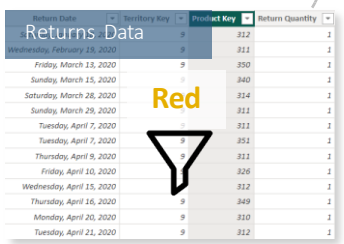
STEP 2

Filters flow "downstream" to related tables

1 1

* *





STEP 3

Measure evaluates against the filtered table

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

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

= 4,011

DAX MEASURE TOTALS

Measure totals may seem incorrect or inconsistent depending on how they are calculated, because they **don't simply add up the visible values in the report**

*Total Returns look right, but shouldn't Total Orders be **37,888**??*

-Anonymous confused man

Category Name	Total Returns	Total Orders
Accessories	1,115	16,983
Bikes	427	13,929
Clothing	267	6,976
Total	**1,809**	**25,164**

PRO TIP:

Understand EXACTLY how your measures calculate and **what they are designed to measure**

```
1 Total Orders =
2 DISTINCTCOUNT(
3     'Sales Data'[Order Number]
4 )
```

[Total Orders] counts **distinct orders** in the Sales Data table

Order Date	Stock Date	Order Number	Product Key
Thursday, June 30, 2022	Thursday, April 07, 2022	SO74140	568
Thursday, June 30, 2022	Friday, March 04, 2022	SO74140	477
Thursday, June 30, 2022	Monday, May 30, 2022	SO74140	223
Thursday, June 30, 2022	Friday, April 29, 2022	SO74141	604
Thursday, June 30, 2022	Wednesday, May 04, 2022	SO74141	471
Thursday, June 30, 2022	Monday, May 30, 2022	SO74142	383
Thursday, June 30, 2022	Friday, March 18, 2022	SO74142	490
Thursday, June 30, 2022	Tuesday, March 15, 2022	SO74143	479
Thursday, June 30, 2022	Friday, April 08, 2022	SO74143	606
Thursday, June 30, 2022	Tuesday, March 22, 2022	SO74143	477
Thursday, June 30, 2022	Thursday, June 02, 2022	SO74143	462
Thursday, June 30, 2022	Monday, April 25, 2022	SO74144	574
Thursday, June 30, 2022	Sunday, April 24, 2022	SO74144	220
Thursday, June 30, 2022	Monday, March 14, 2022	SO74145	561
Thursday, June 30, 2022	Tuesday, June 14, 2022	SO74146	584
Thursday, June 30, 2022	Friday, March 18, 2022	SO74147	605
Thursday, June 30, 2022	Sunday, May 29, 2022	SO74147	538
Thursday, June 30, 2022	Thursday, March 24, 2022	SO74147	490

Table: Sales Data (56,046 rows) Column: Order Numbe (25,164 distinct values)

Order **SO74144** included **two products**: a bike and a helmet.

That counts as **1** distinct order for the Total and **1** distinct order for BOTH **Accessories** & **Bikes**

With no filter context, there are **25,164** total distinct orders

ASSIGNMENT: CALCULATE

NEW MESSAGE

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

Subject: **URGENT: Bike returns**

Hey there,

Apparently George (our Product VP) has been speaking with some of the store managers, and they've raised concerns about the number of bike returns they are seeing recently.

Can you please create a measure to calculate total returns for bikes specifically, and let me know what you see? Volume alone won't tell the full story, so let's calculate the return *rate* for bikes as well, and see how it's trending before responding to George.

Need this ASAP – thank you!

Reply Forward

Key Objectives

1. Create a new measure named **Bike Returns** to calculate the total quantity of bikes returned
2. Create a matrix to show **Bike Returns** (values) by **Start of Month** (rows). What do you notice about the volume of bike returns over time?
3. Create a new measure named **Bike Sales** to calculate the total quantity of bikes sold, and add it to the matrix. What do you notice?
4. Create a new measure named **Bike Return Rate** using either CALCULATE or DIVIDE, and add it to the matrix
5. How would you respond to the Product VP's concerns about rising bike returns?

SOLUTION: CALCULATE

NEW MESSAGE

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

Subject: **URGENT: Bike returns**

Hey there,

Apparently George (our Product VP) has been speaking with some of the store managers, and they've raised concerns about the number of bike returns they are seeing recently.

Can you please create a measure to calculate total returns for bikes specifically, and let me know what you see? Volume alone won't tell the full story, so let's calculate the return *rate* for bikes as well, and see how it's trending before responding to George.

Need this ASAP – thank you!

Reply Forward

Solution Preview

```
1 Bike Returns =
2 CALCULATE(
3     [Total Returns],
4     'Product Categories Lookup'[Category Name] = "Bikes"
5 )
```

```
1 Bike Sales =
2 CALCULATE(
3     [Quantity Sold],
4     'Product Categories Lookup'[Category Name] = "Bikes"
5 )
```

```
1 Bike Return Rate =
2 CALCULATE(
3     [Return Rate],
4     'Product Categories Lookup'[Category Name] = "Bikes"
5 )
```

(Solution continued on next slide)

SOLUTION: CALCULATE

NEW MESSAGE

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

Subject: **URGENT: Bike returns**

Hey there,

Apparently George (our Product VP) has been speaking with some of the store managers, and they've raised concerns about the number of bike returns they are seeing recently.

Can you please create a measure to calculate total returns for bikes specifically, and let me know what you see? Volume alone won't tell the full story, so let's calculate the return *rate* for bikes as well, and see how it's trending before responding to George.

Need this ASAP – thank you!

Reply Forward

Solution Preview

6/1/2021	8	312	2.564%
7/1/2021	12	506	2.372%
8/1/2021	14	485	2.887%
9/1/2021	22	575	3.826%
10/1/2021	26	612	4.248%
11/1/2021	25	688	3.634%
12/1/2021	26	1038	2.505%
1/1/2022	14	766	1.828%
2/1/2022	22	806	2.730%
3/1/2022	27	888	3.041%
4/1/2022	38	956	3.975%
5/1/2022	36	1116	3.226%
6/1/2022	34	1157	2.939%
Total	**429**	**13929**	**3.080%**

The volume of bike returns has risen over time, but so has the number of bikes being sold.

When we look at the rate of returns as a percent of sales, we don't see a concerning trend.

ALL

ALL Returns all rows in a table, or all values in a column, ignoring any filters that have been applied

=**ALL**(Table *or* Column, *[Column2], [Column3],…*)

The **table** or **column** that you want to clear filters on

Examples:

- *Transactions*
- *Products[Category]*

Additional columns that you want to clear filters on (optional)

- Cannot specify columns if your first parameter is a **table**
- All columns must include the **table name** and come from the **same table**

Examples:

- *'Customer Lookup'[City], 'Customer Lookup'[Country]*
- *Products[Product Name]*

PRO TIP:

Instead of adding filter context, **the ALL function removes it**. This is often used in "**% of Total**" calculations, when the denominator needs to remain fixed regardless of filter context.

ASSIGNMENT: CALCULATE & ALL

NEW MESSAGE

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

Subject: **Return analysis follow-up**

Hey again,

Thanks for the quick turnaround on that bike return analysis – crisis averted!

That got me thinking about how we could start analyzing the return data in our reports. Could you please help me create two new measures, one to calculate ALL returns (regardless of filter context), and another that divides Total Returns by All Returns?

That should allow us to see the % of returns by different products and product categories.

Reply Forward

Key Objectives

1. Create a new measure named **All Returns** to calculate the total number of returns, regardless of filter context
2. Create a new measure named **% of All Returns** that divides Total Returns by All Returns
3. Create a matrix to show % of All Returns (values) by product Category Name (rows). Which category accounts for the largest percentage of returns? The smallest?

SOLUTION: CALCULATE & ALL

NEW MESSAGE

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

Subject: **Return analysis follow-up**

Hey again,

Thanks for the quick turnaround on that bike return analysis – crisis averted!

That got me thinking about how we could start analyzing the return data in our reports. Could you please help me create two new measures, one to calculate ALL returns (regardless of filter context), and another that divides Total Returns by All Returns?

That should allow us to see the % of returns by different products and product categories.

Reply Forward

Solution Preview

```
All Returns =
CALCULATE(
    [Total Returns],
    ALL(
        'Returns Data'
    )
)
```

```
% of All Returns =
DIVIDE(
    [Total Returns],
    [All Returns]
)
```

Category Name	% of All Returns
Bikes	23.60%
Clothing	14.76%
Accessories	61.64%
Total	**100.00%**

FILTER

FILTER — Returns a table that represents a subset of another table or expression

=**FILTER**(Table, FilterExpression)

Table to be filtered

Examples:

- *Territory Lookup*
- *Customer Lookup*

A Boolean (True/False) filter expression to be evaluated for each row of the table

Examples:

- *'Territory Lookup'[Country] = "**USA**"*
- *Calendar[Year] = **1998***
- *Products[Price] > **[Overall Avg Price]***

HEY THIS IS IMPORTANT!

FILTER is used to add new filter context, and can handle **more complex filter expressions** than CALCULATE (*by referencing measures, for example*)

Since FILTER returns an entire table, it's often **nested within other functions**, like CALCULATE or SUMX

PRO TIP:

Since FILTER **iterates through each row in a table**, it can be slow and computationally expensive; only use FILTER if a simple CALCULATE function won't get the job done!

ITERATOR FUNCTIONS

Iterator (or "**X**") **functions** allow you to loop through the same expression on each row of a table, then apply some sort of aggregation to the results (SUM, MAX, etc.)

=**SUMX**(Table, Expression)

Aggregation to apply to calculated rows*

Examples:

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

Table in which the expression will be evaluated

Examples:

- *Sales*
- *FILTER(Sales, RELATED(Products[Category])="Clothing")*

Expression to be evaluated for each row of the given table

Examples:

- *[Total Orders]*
- *Sales[Retail Price] * Sales[Quantity]*

PRO TIP:

Imagine that iterator functions **add a temporary new column** to a table, calculate a value in each row based on the given expression, then aggregate the values within that temporary column (similar to **SUMPRODUCT** in Excel)

In this example we're looking at **SUMX, but other iterator functions follow a similar syntax*

ASSIGNMENT: ITERATORS

NEW MESSAGE

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

Subject: **Profit calculation – HELP!**

Hey,

Ethan asked for a quick analysis of company profit over the past few years, but I'm struggling with the calculation.

We need a measure that multiplies order quantity by product cost, but I'd like to do it without adding redundant columns to our Sales table.

Could you take a stab at this please?

-Dianne

Reply Forward

Key Objectives

1. Create a new measure named **Total Cost** that multiplies the order quantities in the Sales Data table by the product cost in the Product Lookup table, then calculates the sum
2. Create a new measure named **Total Profit** (revenue minus cost)
3. Create a matrix to show Total Profit (values) by Year (rows). How much profit has AdventureWorks earned so far in 2022?

SOLUTION: ITERATORS

NEW MESSAGE

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

Subject: **Profit calculation – HELP!**

Hey,

Ethan asked for a quick analysis of company profit over the past few years, but I'm struggling with the calculation.

We need a measure that multiplies order quantity by product cost, but I'd like to do it without adding redundant columns to our Sales table.

Could you take a stab at this please?

-Dianne

Reply Forward

Solution Preview

```
1 Total Cost =
2 SUMX(
3     'Sales Data',
4     'Sales Data'[Order Quantity]
5     *
6     RELATED(
7         'Product Lookup'[Product Cost]
8     )
9 )
```

```
1 Total Profit =
2 [Total Revenue] - [Total Cost]
```

Year	Total Profit
2020	$2,601,606
2021	$3,967,023
2022	$3,888,952
Total	**$10,457,581**

TIME INTELLIGENCE

Time Intelligence patterns are used to calculate common date-based comparisons

Performance To-Date

=**CALCULATE**(Measure, **DATESYTD**(Calendar[Date]))

*Use **DATESYTD** for Years, **DATESQTD** for Quarters, **DATESMTD** for Months*

Previous Period

=**CALCULATE**(Measure, **DATEADD**(Calendar[Date], -1, **MONTH**))

*Select an interval (**DAY**, **MONTH**, **QUARTER**, or **YEAR**) and the # of intervals to compare (e.g. previous month, rolling 10-day)*

Running Total

=**CALCULATE**(Measure,
DATESINPERIOD(Calendar[Date], **MAX**(Calendar[Date]), -10, **DAY**))

PRO TIP:

To calculate a **moving average**, use the running total calculation above and **divide by the number of intervals**

ASSIGNMENT: TIME INTELLIGENCE

NEW MESSAGE

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

Subject: **Time Intelligence Measures**

Hey there, need a big favor!

The leadership team has been asking a lot of questions about month-over-month and year-over-year comparisons, and I've been pulling the numbers pretty manually.

Could you please add the following list of measures, to make these metrics easier to track and share with stakeholders?

Thank you!

-Dianne

Reply Forward

Key Objectives

Add the following measures to the model:

1. **Previous Month Returns**
2. **Previous Month Orders**
3. **Previous Month Profit**
4. **Order Target** (10% increase over previous month)
5. **Profit Target** (10% increase over previous month)
6. **90-day Rolling Profit**

SOLUTION: TIME INTELLIGENCE

NEW MESSAGE

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

Subject: **Time Intelligence Measures**

Hey there, need a big favor!

The leadership team has been asking a lot of questions about month-over-month and year-over-year comparisons, and I've been pulling the numbers pretty manually.

Could you please add the following list of measures, to make these metrics easier to track and share with stakeholders?

Thank you!

-Dianne

Reply Forward

Solution Preview

```
1 Previous Month Orders =
2 CALCULATE(
3     [Total Orders],
4     DATEADD(
5         'Calendar Lookup'[Date],
6         -1,
7         MONTH
8     )
9 )
```

```
1 90-day Rolling Profit =
2 CALCULATE(
3     [Total Profit],
4     DATESINPERIOD(
5         'Calendar Lookup'[Date],
6         LASTDATE(
7             'Calendar Lookup'[Date]
8         ),
9         -90,
10        DAY
11    )
12 )
```

```
1 Order Target =
2 [Previous Month Orders] * 1.1
```

DAX BEST PRACTICES

Know when to use calculated columns vs. measures

- *Use calculated columns for filtering, and measures for aggregating values*

Use explicit measures, even for simple calculations

- *Explicit measures can be referenced anywhere, and nested within other measures*

Use fully-qualified column references in measures

- *This makes your DAX more readable, and differentiates column references from measure references*

Move column calculations “upstream” when possible

- *Adding calculated columns at the source or in Power Query improves report speed and efficiency*

Minimize the use of “expensive” iterator functions

- *Use iterators with caution, especially if you are working with large tables or complex models*