Power BI
Day 6

Introduction to DAX in Power BI

INTRODUCTION TO DAX IN POWER BI

What is DAX?

- **D**ata **A**nalysis e**X**pressions
- Formula language to create calculations
  - Columns, tables, measures
- Based on Excel formulas and functions
  - e.g., `SUM()`
- Used in other Microsoft tools
  - Power Pivot and Analysis Services

DAX functions

- Predefined formulas that perform calculations on specific values called **arguments**
- **Function syntax** indicates the order of arguments expected
- Function categories
  - Aggregation - `SUM()`, `AVERAGE()`, `COUNT()`
  - Date and Time - `TODAY()`, `MONTH()`, `YEAR()`
  - Logical - `IF()`, `AND()`, `OR()`
  - Text - `CONCATENATE()`, `UPPER()`, `LEFT()`
  - And many more...
- DAX reference:
  - https://docs.microsoft.com/en-us/dax/dax-function-reference

DAX functions example

- `SUM()`
  - *Syntax:* `SUM(<column>)`
  - *Description:* Adds all the numbers in a column.
  - *One argument:* `<column>`
  - *Example:* `SUM(Sales)`

- `LEFT()`
  - *Syntax:* `LEFT(<text>, <num_chars>)`
  - *Description:* Returns the specified number of characters from the start of a text.
  - *Two arguments:* `<text>` , `<num_chars>`
  - *Example:* `LEFT('DataCamp', 4)` = "Data"

Creating calculated columns

- Expands our existing datasets without editing the source data
- Evaluates at a row level and adds a new column to an existing table
- Calculated at data load or when the data is refreshed

Creating calculated columns

- Expands our existing datasets without editing the source
- Evaluates at a row level and adds a new column to an existing table
- Calculated at data load and when the data is refreshed
- DAX example: `Price_w_tax = Price + ( Price * Tax )`

Item	Price	Tax
A	$ 20	25%
B	$ 45	0%
C	$ 100	15%

Creating calculated columns

- Expands our existing datasets without editing the source
- Evaluates at a row level and adds a new column to an existing table
- Calculated at data load and when the data is refreshed
- DAX example: `Price_w_tax = Price + ( Price * Tax )`

Item	Price	Tax	Price_w_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
C	$ 100	15%	**$115**

Creating calculated measures

- Enables complex calculations
- Aggregates multiple rows and adds a new field that can be added to visualizations
- Calculated at **query time** as you interact and filter
  - More efficient because the calculation is not run every time the table is accessed
- Two ways to create a measure
  - Write a measure from scratch
  - Use the built-in Quick Measure tool

Creating calculated measures

Item	Price	Tax	Price_w_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
C	$ 100	15%	**$115**

- `Total_price_w_tax = SUM(Price_w_tax)`
- `Total_price_w_tax = $25 + $45 + $115 = $185`

Summary

Calculated columns:

- Evaluate for each row
- Add a new column to an existing table
- Calculated at data load or when the data is refreshed

Calculated measures:

Summary

Calculated columns:

- For evaluating each row
- Add a new column to an existing table
- Calculated at data load or when the data is refreshed

Item	Price	Tax	Price_w_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
C	$ 100	15%	**$115**

Calculated measures:

- For aggregating multiple rows
- Results in another field that you can add to a visualization
- Calculated at **query time** as you interact and filter
- `Total_price_w_tax = SUM(Price_w_tax)`

Summary

Calculated columns:

- For evaluating each row
- Add a new column to an existing table
- Calculated at data load and when the data is refreshed

Item	Price	Tax	Price_w_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
C	$ 100	15%	**$115**

Calculated measures:

- For aggregating multiple rows
- Results in another field that you can add to a visualization
- Calculated at **query time** as you interact and filter
- `Total_price_w_tax = SUM(Price_w_tax)`

[1] Calculated tables will be covered later.

Adventure Works

- Sells bikes and bike-parts globally
- Table: **Sales**
  - Transactional data for each order line of a sale
  - Contains categorical data including product category

Let's practice!

INTRODUCTION TO DAX IN POWER BI

Creating calculated columns and measures

INTRODUCTION TO DAX IN POWER BI

Let's practice!

INTRODUCTION TO DAX IN POWER BI

Context in DAX formulas

INTRODUCTION TO DAX IN POWER BI

Introduction to Context

- Enables dynamic analysis where results of a formula change to reflect the selected data
- There are 3 types of context: row, filter and query

Introduction to Row Context

- "The current row"

Calculated Column

- Includes values from all columns within the current row

Item	Price	Tax	Price_with_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
C	$ 100	15%	**$115**

Introduction to Row Context

- "The current row"

Measures

- Can apply when using iterator functions which compute calculations row by row
- Iterator functions can be identified by an `X` after the function name i.e `SUMX()`
- Syntax: `SUMX(<table>, <expression>)`

Introduction to Row Context

- "The current row"

Measures

- Can apply when using iterator functions which compute calculations row by row
- Iterator functions can be identified by an `X` after the function name i.e `SUMX()`
- Syntax: `SUMX(<table>, <expression>)`

Item	Price	Tax	Price_with_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**

Introduction to Row Context

- "The current row"

Measures

- Can apply when using iterator functions which compute calculations row by row
- Iterator functions can be identified by an `X` after the function name i.e `SUMX()`
- Syntax: `SUMX(<table>, <expression>)`

Item	Price	Tax	Price_w_tax
A	$ 20	25%	**$25**
B	$ 45	0%	**$45**
Total	**-**	**-**	**$ 70**

- Example: `SUMX(Sales, Sales[Price] + (Sales[Price] * Sales[Tax]))`

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Filter context can be applied in several ways:

- Attributes in a row/column
- Via a slicer
- Through the filter pane
- In a calculated measure

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Example:

Color	Quantity
Blue	1,250
Green	200
Black	4,000

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Example:

Color	Quantity
Blue	1,250

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Example:

Color	Quantity
Blue	1,250
Green	200
Black	4,000

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Example:

	Socks	Shoes	T-shirt
Blue	200	800	250
Green	90	10	100
Black	2,000	800	1,200

Introduction to Filter Context

Filter context is a set of filters that have been applied before the calculation is carried out.

Example:

	Socks
Blue	200

Calculate Function

- Syntax: `CALCULATE(<expression>[, <filter1> [, <filter2> [, …]]])`
  - Expression: a measure or calculation to be evaluated. Must return a single value.
  - Filters:
    - Filters need to evaluate as a table
    - Filters should not clash with one another
      - `Sales[City]="London"` , `Sales[Country] <> "United Kingdom"`
    - `CALCULATE()` filters will always override filters from the visualization
- Example: `CALCULATE(SUM(Sales), Sales[Region]="EMEA")`

Let's practice!

INTRODUCTION TO DAX IN POWER BI

Creating DAX measures

INTRODUCTION TO DAX IN POWER BI

Let's practice!

INTRODUCTION TO DAX IN POWER BI

The Date table

INTRODUCTION TO DAX IN POWER BI

Working with dates

Example Date: 2020/09/20 12:52

Date and Time Functions

- `YEAR(<date>)` > 2020
- `QUARTER(<datetime>)` > 3
- `MONTH(<datetime>)` > 9

Format Function

- Weekday: `FORMAT(<date>, <"dddd">)` > Friday
- Time: `FORMAT(<date>, <"h:nn:ss">)` > "12:52:00"

Time Intelligence Functions

- `LASTDATE()`
- `DATESBETWEEN()`
- `DATEADD()`

[1] https://docs.microsoft.com/en-us/dax/format-function-dax

Working with dates

- Evaluate data in time-series to spot trends and patterns i.e seasonal performance
- Out of the box features:
  - 20+ Date and Time Functions
  - 30+ Time Intelligence Functions
  - Automatically enabled date hierarchies
    - Drill-able to year, quarter, month and day

The importance of a date table

Issues of relying on **only** dates from transactional tables:

- Gaps in dates i.e no sales made on 20th September
- Returns wrong results when using time-intelligence functions
  - No error, wrong result
  - Difficult to troubleshoot

Creating a Date Table

- A dedicated date table is highly recommended for accurate reporting using time-intelligence functions.

Benefits:

- Filter by multiple date attributes such as Year and Month
- Custom calendar view/definitions such as fiscal dates
- Use of time-intelligence features to select a time horizon (e.g Today, Yesterday, Last 30 days)

Types of Analysis:

- Revenue by Day of Week, Fiscal Performance, Public Holidays

Creating a Date table

CALENDAR()

- Syntax: `CALENDAR(<start_date>, <end_date>)`
- Returns a table with a single column 'date' that contains a continuous set of dates inclusive of the specified dates
- Example: `CALENDAR('2020-01-01', '2020-12-31')`

Creating a Date table

CALENDAR()

- Syntax: `CALENDAR(<start_date>, <end_date>)`
- Returns a table with a single column 'date' that contains a continuous set of dates inclusive of the specified dates
- Example: `CALENDAR('2020-01-01', '2020-12-31')`

Date
2020-01-01
2020-01-02
...
2020-12-31

Creating a Date table

CALENDARAUTO()

- Syntax: `CALENDARAUTO(<fiscal_year_end_month>)`
- Returns a table with a single column 'date' that automatically takes the earliest and latest date in the model and internally calls `CALENDAR()` .
- Example: `CALENDARAUTO(12)`

Creating a Date table

CALENDARAUTO()

- Syntax: `CALENDARAUTO(<fiscal_year_end_month>)`
- Returns a table with a single column 'date' that automatically takes the earliest and latest date in the model and internally calls `CALENDAR()` .
- Example: `CALENDARAUTO(12)`

Date
2020-01-01
2020-07-31
...
2020-12-31

Let's practice!

INTRODUCTION TO DAX IN POWER BI

Dates and Quick Measures

INTRODUCTION TO DAX IN POWER BI

Let's practice!

INTRODUCTION TO DAX IN POWER BI

Become a DAX master!

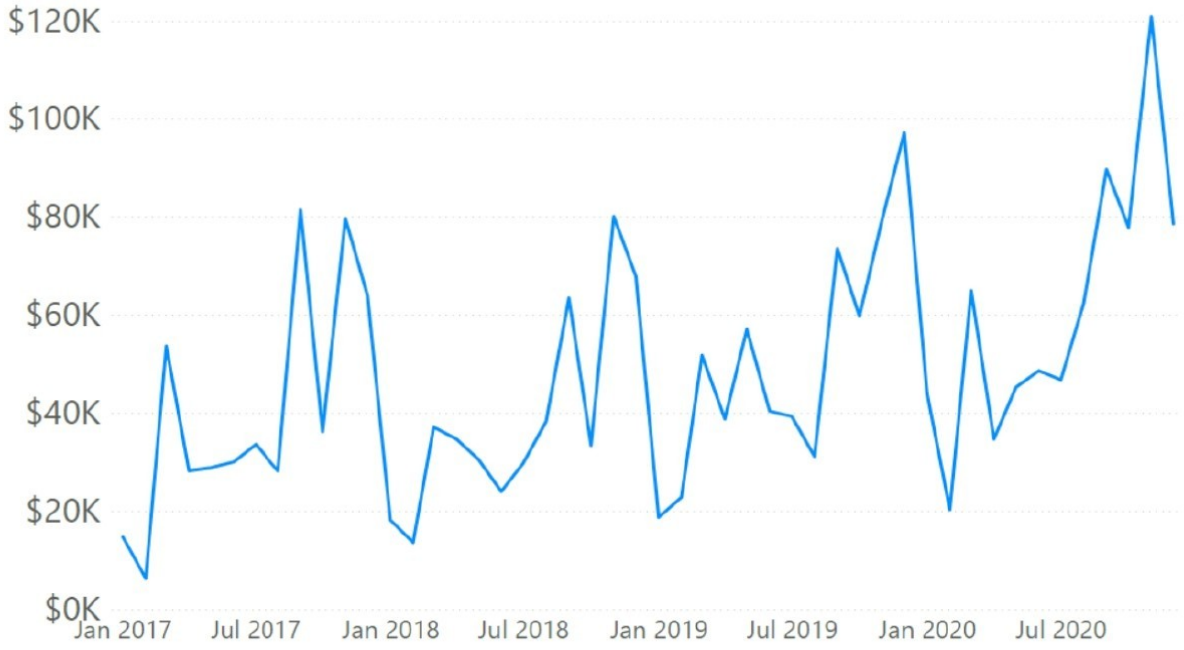
INTRODUCTION TO DAX IN POWER BI

Time intelligence functions

INTERMEDIATE DAX IN POWER BI

Time intelligence functions

- Manipulate and compare data using time periods



- Compare current period with previous period
- Estimate monthly/quarterly/yearly goals
- Many time intelligence functions exist

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*

dates	NEXTDAY
2009-07-07	2009-07-08
2009-07-08	2009-07-09
2009-07-09	2009-07-10

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*
- `SAMEPERIODLASTYEAR(<dates>)`
  - *Returns the last year*

dates	NEXTDAY	LASTYEAR
2009-07-07	2009-07-08	2008-07-07
2009-07-08	2009-07-09	2008-07-08
2009-07-09	2009-07-10	2008-07-09

- `DATESBETWEEN(<dates>, <start_date>, <end_date>)`
  - *Returns dates between start and end date*

Time intelligence in Power BI

INTERMEDIATE DAX IN POWER BI

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*
- `SAMEPERIODLASTYEAR(<dates>)`
  - *Returns the last year*

dates	NEXTDAY	LASTYEAR
2009-07-07	2009-07-08	2008-07-07
2009-07-08	2009-07-09	2008-07-08
2009-07-09	2009-07-10	2008-07-09

- `DATESBETWEEN(<dates>, <start_date>, <end_date>)`
  - *Returns dates between start and end date*

dates	DATESBETWEEN
2009-07-07	
2009-07-08	2009-07-08
2009-07-09	2009-07-09
2009-07-10	

Time intelligence functions returning a date

```
Mid Season Sales =
CALCULATE(
    SUM(Fact_Table[Sales]),
    DATESBETWEEN(Dim_Date[Date Key],
                 DATE(2014, 10, 04),
                 DATE(2014, 10, 26)
                 )
)
```

Time intelligence functions returning a date

```
TOTALYTD(<expression>, <dates> [,<filter>])
TOTALQTD(<expression>, <dates> [,<filter>])
TOTALMTD(<expression>, <dates> [,<filter>])
```

Returns the year, quarter, or month to date value of the expression.

```
Sum_YTD =
TOTALYTD(
    SUM(Fact_Table[Value]),
    Dim_Date[Date Key]
)
```

Time intelligence functions returning a date

```
TOTALYTD(<expression>, <dates> [,<filter>])
TOTALQTD(<expression>, <dates> [,<filter>])
TOTALMTD(<expression>, <dates> [,<filter>])
```

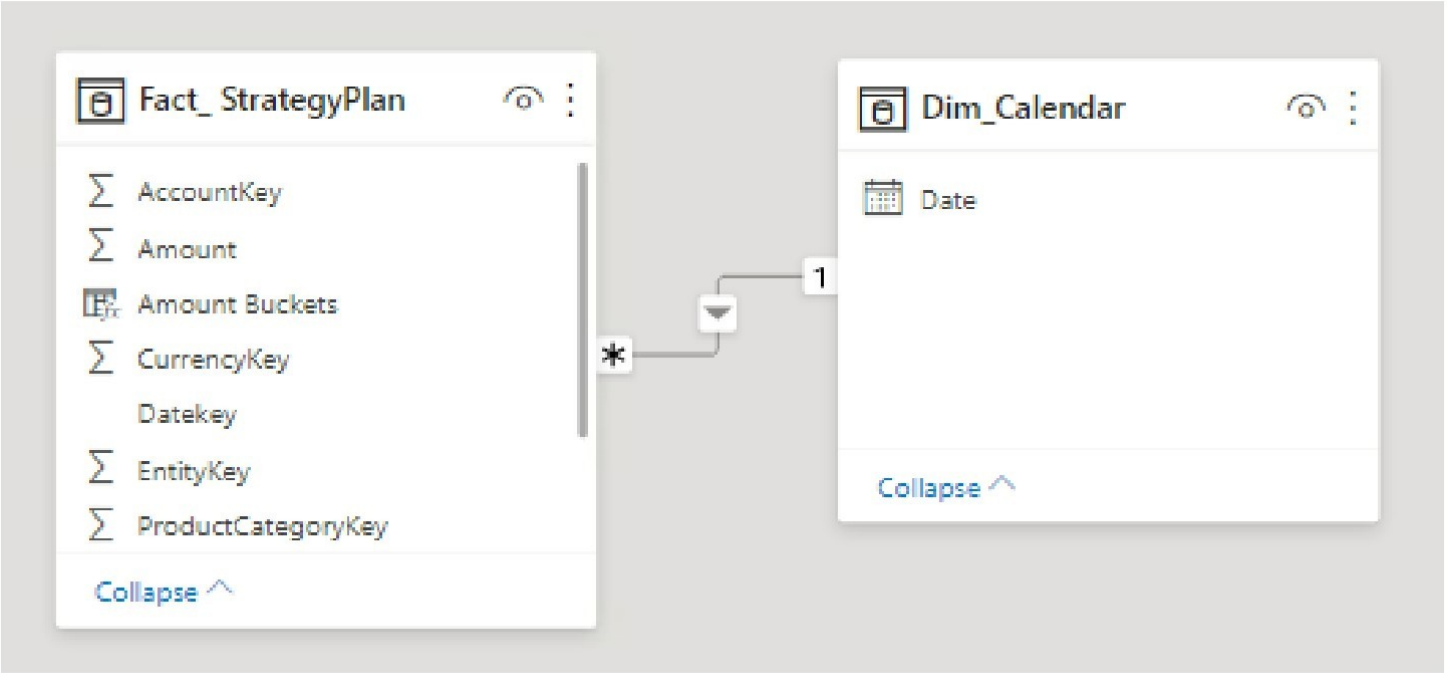
Returns the year, quarter, or month to date value of the expression.

```
Sum_YTD =
TOTALYTD(
    SUM(Fact_Table[Orders]),
    Dim_Date[Date Key]
)
```

Year	Month	Value	Sum_YTD
2021	Jan	6,532	6,532
2021	Feb	4,263	10,795
2021	Mar	1,256	12,051
Total		**12,051**	**12,051**

Best practices for time intelligence functions

- Use a separate date dimension table



A date column in the fact table could contain missing dates!

Let's practice!

INTERMEDIATE DAX IN POWER BI

DAX FUNCTION

BI Academy

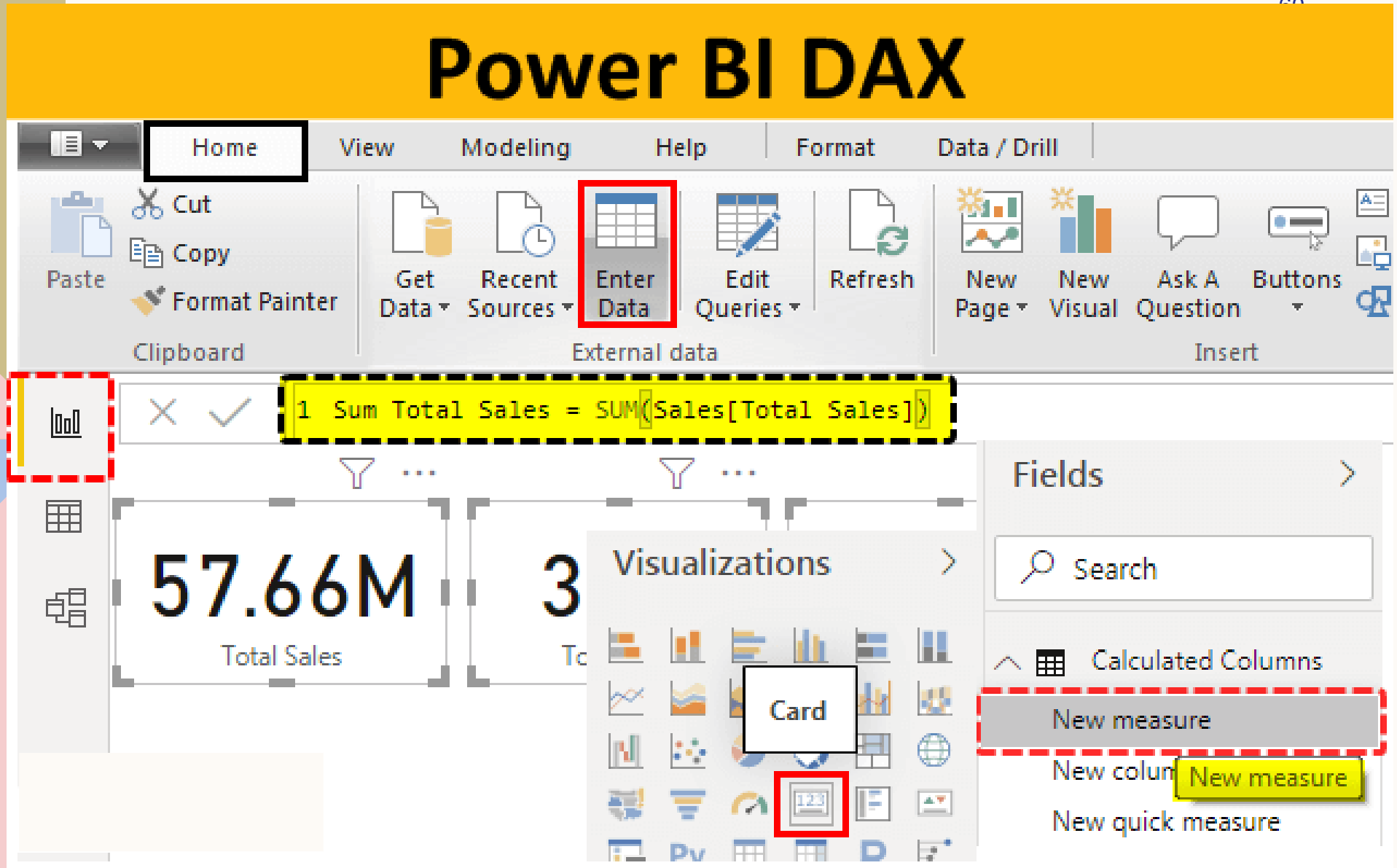
Why is DAX Important?

Learning how to create effective DAX formulas will help you get the most out of your data

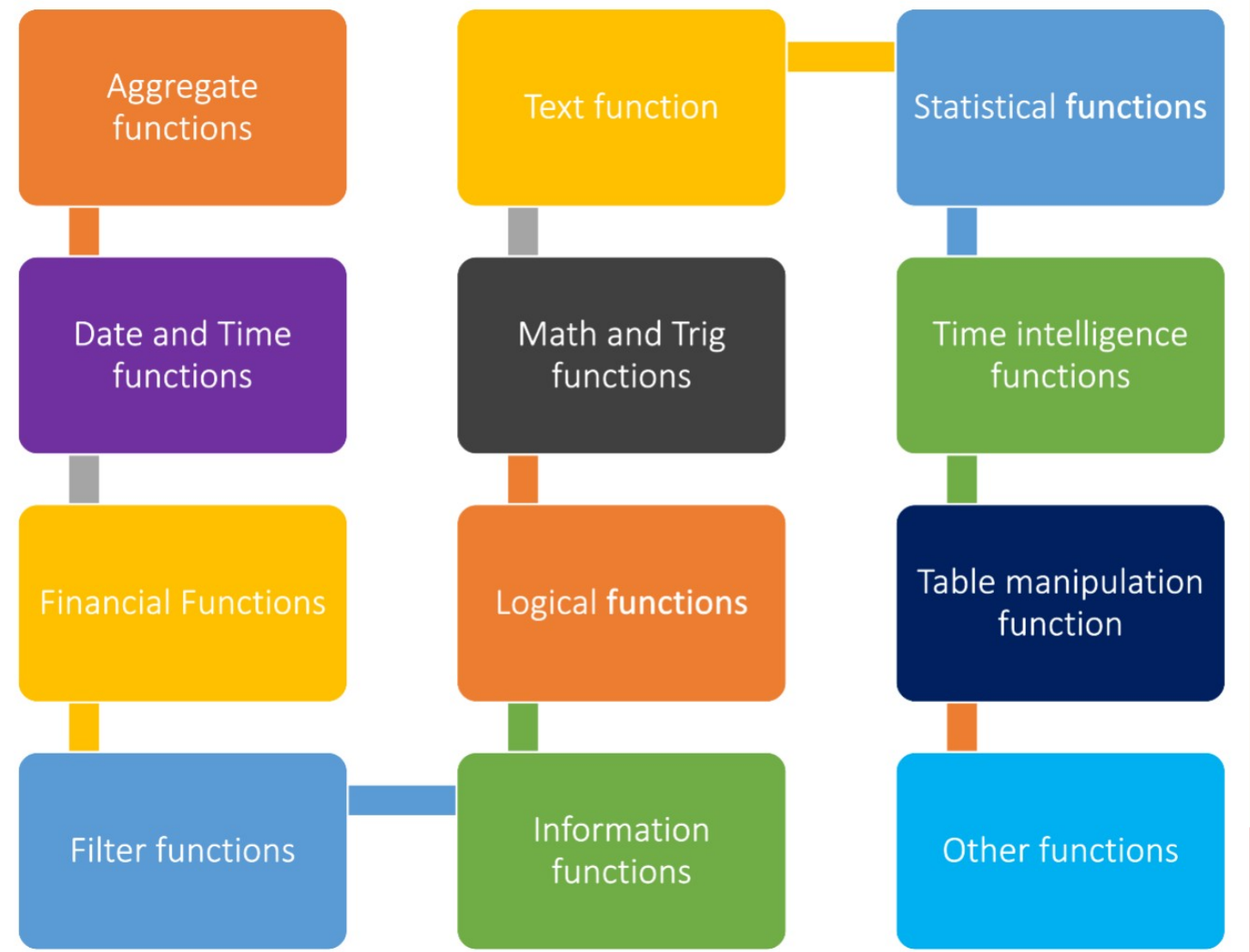
DAX formulas provides the capability to create reports that show valuable insights easy when calculation over large data is needed

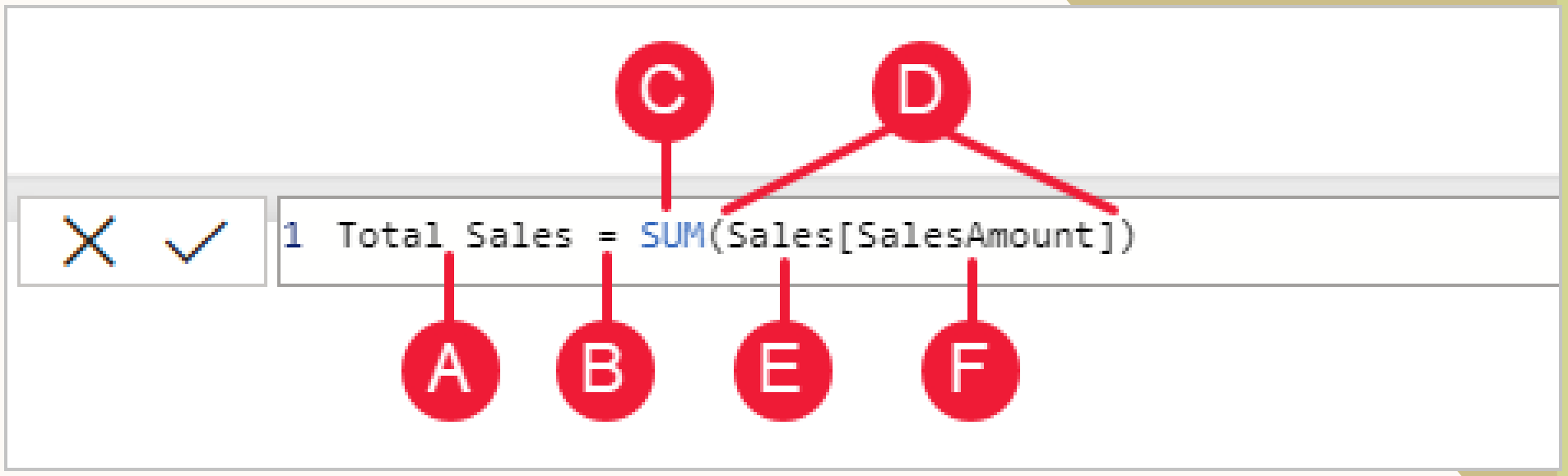
When you get the information you need, you can begin to solve real business problems easily with the help of measures

Power BI DAX


Home
View
Modeling
Help
Format
Data / Drill
Paste
Cut
Copy
Format Painter
Clipboard
Get Data
Recent Sources
Enter Data
Edit Queries
Refresh
External data
New Page
New Visual
Ask A Question
Buttons
Insert
1 Sum Total Sales = SUM(Sales[Total Sales])
57.66M
Total Sales
Visualizations
Card
Fields
Search
Calculated Columns
New measure
New colum
New measure
New quick measure

TYPES OF DAX FUNCTIONS





This formula includes the following syntax elements:

A. The measure name, Total Sales.

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

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

D. Parenthesis (), which surround an expression that contains one or more argume[illegible]ost functions require at least one argument. An argument passes a value to a function.

E. The referenced table, Sales.

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

SUM

Total Sales = SUM('master table'[Sales])

- **The DAX formula Total Sales = SUM('master table'[Sales]) calculates the sum of the 'Sales' column in the 'master table', giving you the total sales amount. You can use similar formulas to create other measures or calculated columns based on your specific analysis requirements.**

SUMX

Total Sales per Customer = SUMX(VALUES('master table'[Customer ID]), CALCULATE(SUM('master table'[Sales])))

- **The SUMX function is used to sum the result of an expression evaluated for each row in a table.**
- **Here's an example of how you might use SUMX:**
- **Let's say you want to calculate the total sales for each customer by summing the 'Sales' column for each order they made.**
- **In this example: VALUES('master table'[Customer ID]) provides a table of unique customer IDs. CALCULATE(SUM('master table'[Sales])) calculates the total sales for each customer using the context of the customer ID.**
- **This measure, when placed in a visual or a table, will give you the total sales for each unique customer.**

AVERAGE

Avg Sales per Order = AVERAGE('master table'[Sales])

is a measure that calculates the average sales amount per order in the 'master table’.
This is a straightforward calculation, and it should work as intended.
If you're encountering an error while creating a new table, it's important to understand the context in which you are trying to use this measure.

Measures are typically used in visualizations, cards, or tables to provide aggregated results. I

f you are trying to use this measure in a calculated column in a new table, it might not work as expected because measures are intended for use in the context of visualizations. Calculated columns in tables should typically use column-wise operations rather than aggregation

AVERAGEX

- Average Sales and Profit per Order =

```
AVERAGEX(
    'Master Table',
    'Master Table'[Sales] + 'Master Table'[Profit]
)
```

- This formula iterates over each row in the 'Master Table' and calculates the sum of 'Sales' and 'Profit' for each row.

- Then, it calculates the average of these sums, giving you the average sales and profit per order.

COUNT AND COUNTROWS

COUNT CUSTID=
COUNT('MASTER TABLE'[CUSTOMER ID])

NUM ORDERS = COUNTROWS('MASTER TABLE')

- Count function Counts the number of rows in the specified column that contain non-blank values.

- You can use the COUNT function to count column values, or you can use the COUNTROWS function to count table rows. Both functions will achieve the same result, providing that the counted column contains no BLANKs.

MAX

Max Sales = MAX('master table'[Sales])

This DAX formula calculates the maximum value from the 'Sales' column in the 'master table'. You can use a similar approach to create measures or calculated columns for other maximum-related analyses.

CALCULATE

Central Region Orders = CALCULATE(COUNTROWS('Location Table'), 'Location Table'[Region] = "Central")

COUNTROWS('master table'): This part of the formula counts the number of rows in the ‘location table' without any filters. CALCULATE(..., 'master table'[Region] = "Central"): This part of the formula modifies the filter context for the 'master table' by specifying that it should only consider rows where the 'Region' column is equal to "Central".

TOPN

```
Top 5 Products Sales =
CALCULATE(
    SUM('master table'[Sales]),
    TOPN(5, ALL('master table'), 'master table'[Sales], DESC))
```

- It calculates the sum of 'Sales' for the top 5 products based on their sales amounts in descending order across the entire 'master table'. Here's a brief breakdown:
- CALCULATE(SUM('master table'[Sales]), ...): This part of the formula calculates the sum of 'Sales' within the specified context.
- TOPN(5, ALL('master table'), 'master table'[Sales], DESC): This part determines the top 5 rows from the 'master table' based on the 'Sales' column in descending order.
- The ALL('master table') ensures that the calculation considers the entire table without any filters.

DISTINCT

```
count_distinct_rows =
COUNTROWS(DISTINCT('Master
Table'[Customer ID]))
```

- It uses the COUNTROWS function in combination with DISTINCT to count the number of distinct values in the 'Customer ID' column from the 'Master Table'. This will give you the count of unique customer IDs in your dataset.
- This measure will return the total number of unique customer IDs in the 'Master Table'. If you want to count distinct rows based on multiple columns, you can modify the formula accordingly.

FILTER

CREATE A NEW TABLE NOT NEW MEASURE OR COLUMN

- FilteredTable = FILTER('Master', 'Master'[Ship Mode] = "Standard Class")
- FilteredTable = FILTER('Master', 'Master'[Category] = "Office Supplies")
- FilteredTable = FILTER('Master', 'Master'[Profit] > 0)
- FilteredTable = FILTER('Master', 'Master'[Region] = "Central")
- FilteredTable = FILTER('Master', 'Master'[Sales] > 100)

- You can use FILTER to reduce the number of rows in the table that you are working with, and use only specific data in calculations. FILTER is not used independently, but as a function that is embedded in other functions that require a table as an argument.

IF

Creating a New Column to Identify Profitable Sales:

ProfitableSales = IF('master table'[Profit] > 0, "Profitable", "Not Profitable")

SalesCategory = IF('master table'[Sales] > 100, "High Sales", "Low Sales")

ShippingModeDecision = IF('master table'[Segment] = "Consumer", "Standard Class", "Express")

SWITCH

Creating a New Column:

```
Switch_Category_Profit =
SWITCH ('Orders'[Category],
    "Office Supplies", IF('Orders'[Profit] > 0, "Profitable",
"Not Profitable"),
    "Furniture", IF('Orders'[Profit] > 0, "Profitable", "Not
Profitable"),
    "Technology", IF('Orders'[Profit] > 0, "Profitable",
"Not Profitable"),
    "Other", "Uncategorized"
)
```

DIVIDE

Creating a New Measure:

Profit Margin = DIVIDE(SUM('master table'[Profit]), SUM('master table'[Sales]))

TIME INTELLIGENCE FUNCTIONS

DATA ANALYSIS EXPRESSIONS (DAX)

- Data Analysis Expressions (DAX) includes time-intelligence functions that enable you to manipulate data using time periods, including days, months, quarters, and years, and then build and compare calculations over those periods.
- Before using any time-intelligence functions, make sure to mark one of the tables containing date column as Date Table.

DATESBETWEEN

RETURNS A TABLE THAT CONTAINS A COLUMN OF DATES THAT BEGINS WITH A SPECIFIED START DATE AND CONTINUES UNTIL A SPECIFIED END DATE.

CREATE NEW MEASURE AND CROSS CHECK WITH DATE SLICER

- Customers Order Date time Intelligent function =
- CALCULATE(
- DISTINCTCOUNT('Order Table'[Order ID]),
- DATESBETWEEN(
- 'Order Table'[Order Date],
- DATE(2017, 12,20),
- MAX('Order Table'[Order Date])
-)
-)

- CALCULATE: This function is used to modify the context in which a formula is being evaluated.
- DISTINCTCOUNT('Order Table'[Order ID]): This calculates the distinct count of 'Order ID' in the modified context.
- DATESBETWEEN('Order Table'[Order Date], DATE(2017, 12, 20), MAX('Order Table'[Order Date])):
- This filters the data based on the date range specified, ensuring only data within that range is considered.

THANK YOU

BI Academy

Let's practice!

INTERMEDIATE DAX IN POWER BI