DAX

Dr. Ernesto Lee

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
- 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:

- 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

Context in DAX Formulas

Introduction to Context

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

Introduction to Row Context

- "The current row"

Introduction to Row Context

- "The current row"

Calculated Column

- Includes values from all columns within the current row

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 identi ed by an `X` a er 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 identi ed by an `X` a er 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 lters that have been applied before the calculation is carried out.

Filter context can be applied in several ways:

- A ributes in a row/column
- Via a slicer
- Through the lter pane
- In a calculated measure

Introduction to Filter Context

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

Example:

	Socks	Shoes	T-shirt
Blue	200	800	250
Green	90	10	100
Black	2,00 0	800	1,200

Introduction to Filter Context

Filter context is a set of lters 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()` lters will always override lters from the visualization
- Example: `CALCULATE(SUM(Sales), Sales[Region]="EMEA")`

The Date Table

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

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

Working with dates

- Evaluate data in time-series to spot trends and pa erns 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 -- hard to troubleshoot
  -

Creating a Date Table

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

Bene ts:

- Filter by multiple date a ributes such as Year and Month
- Custom calendar view/de nitions such as scal 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 speci ed 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 speci ed 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