Power BI

DAX

Lecturer: Chan Sophal

Group 4:

- **Hoy Seiha**
- **Pov Phearum**
- **Ngang Puthsabbos**
- **Mony Sovann**

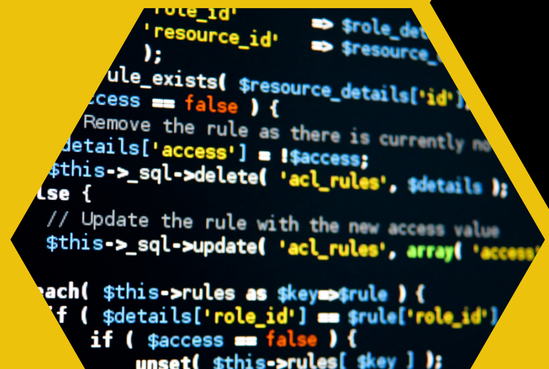


Table of Contents

DAX Pattern

“A pattern is a general, reusable solution to a commonly occurring business problem.”

01 Time Intelligence Calculation

A time intelligence calculation refers to any calculation that involves time.

There are four time intelligence calculations:

1. Period-to-date
2. Period-over-period
3. Period-to-date-growth
4. Moving annual total/average

Calendars are not the same

Does your year start on the 1st of January?

What is the granularity on your needs?

What is your month-over-month comparison?

Do you exactly 12 months? 13..

Predefined DAX functions

- Predefined DAX time intelligence functions
- Standard-time calculations
- Easy to users, but with limitations

Custom Calculations

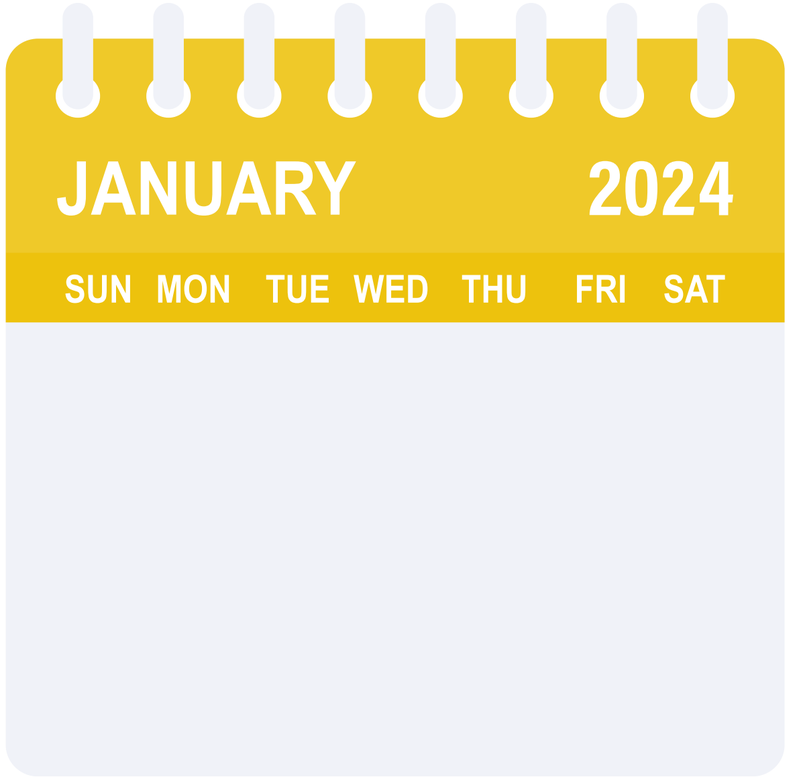
- Custom time intelligence calculations
- Month, weeks calculations
- Harder DAX code, but more flexibility

1.1 Standard-related Time

The Standard time-related calculations pattern is implemented using regular DAX time intelligence functions.

- Based on the regular Gregorian calendar
- to use any time intelligence calculation, you need a well-formed date table
- Granularity: at a day level

1.2 Month-related Calculations



This pattern describes how to compute month-related calculations such as year-to-date, same period last year, and percentage growth using a month granularity.

- Perform using custom DAX code
- Granularity: at the month level
- E.g: a year with a virtual 13th month

1.3 Week-related Calculations

This pattern describes how to compute week-related calculations, such as year-to-date, same period last year, and percentage growth using a week granularity.

- Perform using custom DAX code
- Granularity: at the day level
- weeks are kings: weeks define months and years

1.4 Custom Calculations

This pattern does not rely on DAX's built-in time intelligence functions.

- Perform using custom DAX code
- Complexity: at the day level
- Maximum flexibility comes with maximum complexity
- works with the Gregorian calendar, but is data-driven

2 Semi-addictive

These patterns help you perform calculations that are not fully additive, such as averages, balances.

- First and last date
- First and last date with data
- Opening and closing balance
- Growth in period

2.1 First and last date

it can only be adopted in the few scenarios where the dataset always contains data at the beginning and at the end of each time period.

```
Balance LastDate :=
CALCULATE (
    SUM ( Balances[Balance] ),
    LASTDATE ( 'Date'[Date] )    -- Use FIRSTDATE for Balance FirstDate
)
```

Year	Quarter	Month	Katie Jordan	Luis Bonifaz	Maurizio Macagno
CY 2019				**1,823.00**	**1,750.00**
CY 2020	Q1	January	1,687.00	1,470.00	1,500.00
		February	2,812.00	2,450.00	2,500.00
		March	3,737.00	3,430.00	3,500.00
		Total	**3,737.00**	**3,430.00**	**3,500.00**
	Q2	April	2,250.00	1,960.00	2,000.00
		May	2,025.00	1,764.00	1,800.00
		June	2,700.00	2,352.00	2,400.00
		Total	**2,700.00**	**2,352.00**	**2,400.00**
	Q3	July	3,600.00	3,136.00	3,200.00
		August	5,062.00	4,410.00	4,500.00
		September	2,812.00	2,450.00	2,500.00
		Total	**2,812.00**	**2,450.00**	**2,500.00**
	Q4	October		1,960.00	2,000.00
		Total			
	Total				
Total					

2.2 First and last date with data

In this pattern, the formula searches the last date for which there is data in the current filter context. Therefore, instead of finding the last date in the Date table, it searches for the last date in the Balances table.

```
Balance LastDateWithData :=
VAR MaxBalanceDate =
    CALCULATE (
        MAX ( Balances[Date] ),    -- Use MIN for Balance FirstDateWithData
        ALLEXCEPT (
            Balances,              -- Remove filters from the Balances expanded table
            'Date'                 -- but not from the date
        )
    )
VAR Result =
    CALCULATE (
        SUM ( Balances[Balance] ),
        'Date'[Date] = MaxBalanceDate
    )
RETURN
    Result
```

Year	Quarter	Month	Katie Jordan	Luis Bonifaz	Maurizio Macagno	**Total**
⊞ **CY 2019**				**1,823.00**	**1,750.00**	**3,573.00**
⊟ CY 2020	⊟ Q1	January	1,687.00	1,470.00	1,500.00	**4,657.00**
		February	2,812.00	2,450.00	2,500.00	**7,762.00**
		March	3,737.00	3,430.00	3,500.00	**10,667.00**
		Total	**3,737.00**	**3,430.00**	**3,500.00**	**10,667.00**
	⊟ Q2	April	2,250.00	1,960.00	2,000.00	**6,210.00**
		May	2,025.00	1,764.00	1,800.00	**5,589.00**
		June	2,700.00	2,352.00	2,400.00	**7,452.00**
		Total	**2,700.00**	**2,352.00**	**2,400.00**	**7,452.00**
	⊟ Q3	July	3,600.00	3,136.00	3,200.00	**9,936.00**
		August	5,062.00	4,410.00	4,500.00	**13,972.00**
		September	2,812.00	2,450.00	2,500.00	**7,762.00**
		Total	**2,812.00**	**2,450.00**	**2,500.00**	**7,762.00**
	⊟ Q4	October		1,960.00	2,000.00	**3,960.00**
		November			1,850.00	**1,850.00**
		Total			**1,850.00**	**1,850.00**
	Total				**1,850.00**	**1,850.00**
Total					**1,850.00**	**1,850.00**

2.3 Opening and closing balance

The previous calculations to compute a measure for the last date of a period can be used to compute the closing balance; depending on the requirements, you can choose the right technique

```
Opening :=
VAR PreviousClosingDate =
    DATEADD ( FIRSTDATE ( 'Date'[Date] ), -1, DAY )
VAR Result =
    CALCULATE ( SUM ( Balances[Balance] ), PreviousClosingDate )
RETURN
    Result
```

```
Closing :=
CALCULATE (
    SUM ( Balances[Balance] ),
    LASTDATE ( 'Date'[Date] )
)
```

		Name	Katie Jordan		Luis Bonifaz		**Total**	
Year	Quarter	Month	Opening	Closing	Opening	Closing	**Opening**	**Closing**
⊞ **CY 2019**						**1,823.00**		**1,823.00**
⊟ CY 2020	⊟ Q1	January		1,687.00	1,823.00	1,470.00	**1,823.00**	**3,157.00**
		February	1,687.00	2,812.00	1,470.00	2,450.00	**3,157.00**	**5,262.00**
		March	2,812.00	3,737.00	2,450.00	3,430.00	**5,262.00**	**7,167.00**
		Total		**3,737.00**	**1,823.00**	**3,430.00**	**1,823.00**	**7,167.00**
	⊟ Q2	April	3,737.00	2,250.00	3,430.00	1,960.00	**7,167.00**	**4,210.00**
		May	2,250.00	2,025.00	1,960.00	1,764.00	**4,210.00**	**3,789.00**
		June	2,025.00	2,700.00	1,764.00	2,352.00	**3,789.00**	**5,052.00**
		Total	**3,737.00**	**2,700.00**	**3,430.00**	**2,352.00**	**7,167.00**	**5,052.00**
	⊟ Q3	July	2,700.00	3,600.00	2,352.00	3,136.00	**5,052.00**	**6,736.00**
		August	3,600.00	5,062.00	3,136.00	4,410.00	**6,736.00**	**9,472.00**
		September	5,062.00	2,812.00	4,410.00	2,450.00	**9,472.00**	**5,262.00**
		Total	**2,700.00**	**2,812.00**	**2,352.00**	**2,450.00**	**5,052.00**	**5,262.00**
	⊟ Q4	October	2,812.00		2,450.00	1,960.00	**5,262.00**	**1,960.00**
		November			1,960.00		**1,960.00**	
		Total	**2,812.00**		**2,450.00**		**5,262.00**	
	Total				**1,823.00**		**1,823.00**	
Total								

2.4 Growth in period

A useful application of this pattern is to compute the variation of a measure over a selected time period.

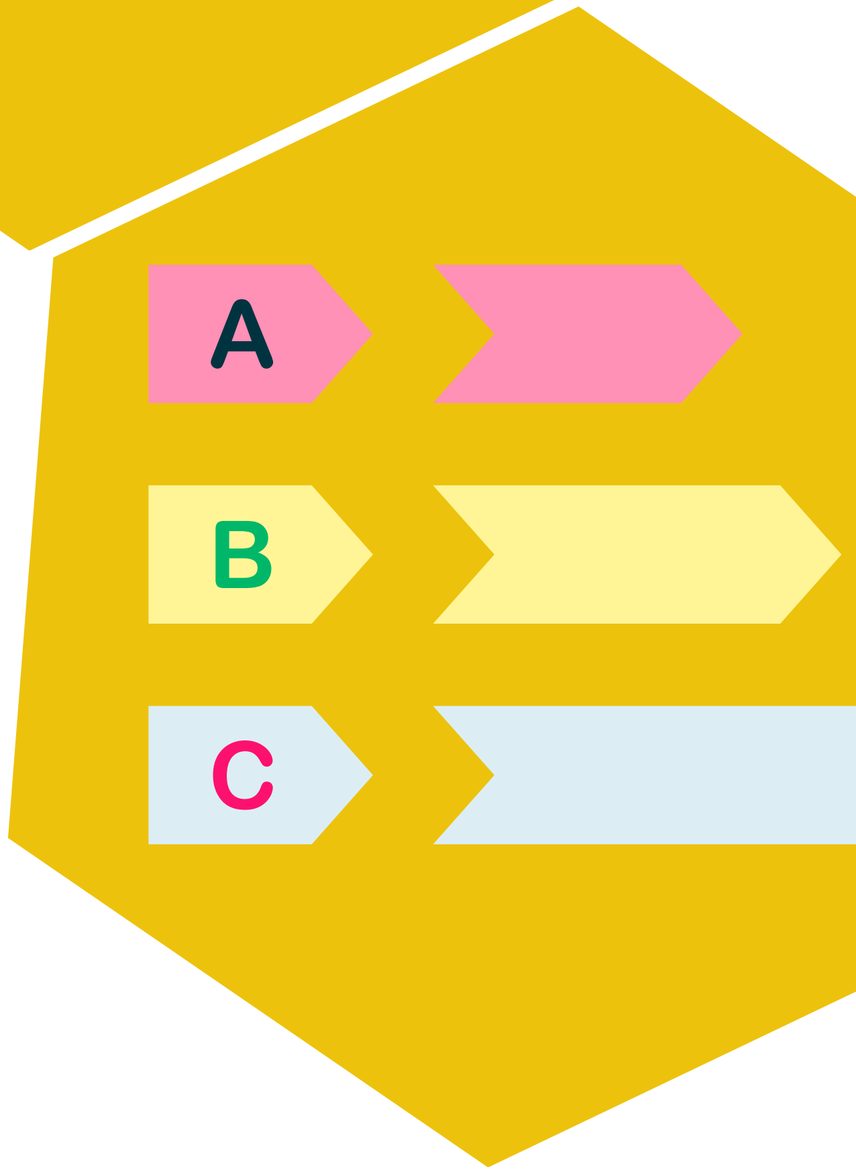
```
Growth :=
VAR Opening = [Opening] -- Use Opening Ever if required
VAR Closing = [Closing] -- Use Closing Ever if required
VAR Delta =
    IF (
        NOT ISBLANK ( Opening ) && NOT ISBLANK ( Closing ),
        Closing - Opening
    )
VAR Result =
    IF ( Delta <> 0, Delta )
RETURN
    Result
```

Year	Quarter	Month	Katie Jordan	Luis Bonifaz	Maurizio Macagno	Total
CY 2020	Q1	January	-443.00	-353.00	-250.00	**-1,046.00**
		February	1,125.00	980.00	1,000.00	**3,105.00**
		March	925.00	980.00	1,000.00	**2,905.00**
		Total	**1,607.00**	**1,607.00**	**1,750.00**	**4,964.00**
	Q2	April	-1,487.00	-1,470.00	-1,500.00	**-4,457.00**
		May	-225.00	-196.00	-200.00	**-621.00**
		June	675.00	588.00	600.00	**1,863.00**
		Total	**-1,037.00**	**-1,078.00**	**-1,100.00**	**-3,215.00**
	Q3	July	900.00	784.00	800.00	**2,484.00**
		August	1,462.00	1,274.00	1,300.00	**4,036.00**
		September	-2,250.00	-1,960.00	-2,000.00	**-6,210.00**
		Total	**112.00**	**98.00**	**100.00**	**310.00**
	Q4	October		-490.00	-500.00	**-990.00**
		November		-147.00	-150.00	**-297.00**
		Total		**-637.00**	**-650.00**	**-1,287.00**
	Total		**682.00**	**-10.00**	**100.00**	**772.00**
Total						

03 ABC Classifications

The ABC classification pattern classifies entities based on values, grouping entities together that contribute to a certain percentage of the total.

- A: 70% of the total sales
- B: up the next 20% of sales
- C: products representing the last 10% of sales



Static ABC classification

assigns a class to each product statically, so that the class of a product does not change depending on the filters being applied to the report

Snapshot ABC classification

a method used in inventory management to categorize items based on their importance or value to the business

Dynamic ABC classification

computes the class of each product dynamically, based on the report filters

Relationship in Dax

"A critical part of creating relationships between different tables to enable effective data analysis and modeling."

There are 4 types of relationship in DAX

Related

Bidirectional cross-filter

Related Table

User Relatioship

Related

A function that allows us to retrieve a single column value from a related table.

Sales Table:

SaleID	ProductID	SaleAmount
1	101	500
2	102	750
3	101	300
4	103	600

Products Table:

ProductID	ProductName
101	Product A
102	Product B
103	Product C

Related CONT.

```
Product Name = RELATED(Products[ProductName])
```

Sales Table:

SaleID	ProductID	SaleAmount	Product Name (Calculated Column)
1	101	500	Product A
2	102	750	Product B
3	101	300	Product A
4	103	600	Product C

Relatedtable

A function that allows us to retrieve a table related to a specified table through a defined relationship.

Sales Table:

OrderID	ProductID	Quantity	Revenue
1	101	5	50.00
2	102	3	30.00
3	103	2	20.00
4	101	4	40.00

Product Table:

ProductID	ProductName
101	Product A
102	Product B
103	Product C

Relatable CONT.

```
Total Revenue by Product =
SUMX(
    Sales,
    Sales[Quantity] * RELATED(Product[ProductName])
)
```

ProductName	Total Revenue
Product A	90.00
Product B	90.00
Product C	40.00

Cross-filter

A relationship between two tables in a data model where both tables can filter each other.

SalesID	ProductID	Quantity	Date
1	101	10	2023-10-01
2	102	5	2023-10-02
3	103	12	2023-10-03
4	101	8	2023-10-04

ProductID	ProductName
101	Product A
102	Product B
103	Product C

Cross-filter CONT.

```
TotalQuantityFiltered =
VAR SelectedProduct = SELECTEDVALUE('Product'[ProductID])
RETURN
    CALCULATE(
        [TotalQuantity],
        FILTER(
            'Product',
            'Product'[ProductID] = SelectedProduct
        )
    )
```

ProductName	TotalQuantityFiltered
Product A	18
Product B	5
Product C	12

Userelationship

Userelationship function allows you to temporarily override active relationships in your DAX expressions. It specifies which relationship should be used for a particular calculation.


THANK
YOU