Course 3 (Week 3)

Power BI - II

Advanced Power BI: Dashboard creation, using DAX for data analysis.

What’s the most random talent or skill you have?

Skills Covered

- Create visually engaging and interactive Power BI dashboards.
- Master advanced DAX expressions for powerful analysis.
- Use time intelligence functions to analyze trends over time.
- Manage data models and relationships for multi-source analysis.
- Optimize DAX performance for faster and scalable dashboards.

Learning Outcomes

- Learners will be able to:

1. Build interactive Power BI dashboards that drive insights.
2. Write DAX expressions to enhance dashboard analytics.
3. Perform context-sensitive data analysis using advanced DAX.
4. Utilize time intelligence for comparative trend analysis.
5. Apply performance optimization techniques to Power BI dashboards

Objectives for today

1. Design dynamic and interactive Power BI dashboards.
2. Apply advanced DAX functions for data analysis.
3. Connect and analyze data from multiple sources with data modeling.
4. Implement time-based calculations for actionable insights.
5. Optimize dashboard performance for efficiency and scalability.

Previously

Hands-on activity over a dataset

- Importing Data
- Data Transformation
- Creating Visualizations
- Exporting Reports

Interactive Dashboarding with Power BI

Sections

Section 1

Section 2

Section 3

Section 4

Section 5

Interactive Dashboarding

- Interactive dashboards are visually compelling, user-friendly interfaces that allow users to interact with data dynamically

- **Components of an Interactive Dashboard:**
- **Visuals** Charts, graphs, tables, maps
- **Interactivity** Filters, slicers, buttons
- **Usability** Clear design, purposeful placement, and responsive elements that guide users to explore insights.

Best Practices (for best UX design)

- **Simplify Your Design: (Hick's Law)**
  Less is more. Use clean, simple visuals. Too much data or too many visuals can overwhelm the user.

- **Focus on Key Metrics: (Pareto's Principle)**
  Choose the most important KPIs and metrics for the dashboard. Prioritize actionable insights.

- **Consistency is Key: (Consistency and Standards)**
  Use consistent color schemes, fonts, and icons to enhance readability and reduce cognitive load.

Best Practices (for best UX design)

- **Use Filters and Slicers Effectively: (User control & freedom)**
  Give users the ability to interact with different dimensions without cluttering the layout.

- **Contextual Navigation: (Progressive Disclosure)**
  Design dashboards in a way that helps users navigate the story.
  Use a logical flow of information, starting with **high-level insights** and enabling users to drill down for more details.

Interactive Visual Elements

1. **Slicers**

   These are a visual control that allow users to filter data easily by categories like Time, Region, Product Category, etc. They make data exploration interactive without leaving the dashboard.

- **Example** : Adding a slicer for "Year" allows users to dynamically change the view of the dashboard for a specific year.

Interactive Visual Elements

2. Filters

Filters allow you to narrow down the scope of data being displayed in the dashboard. You can apply filters at a visual level, page level, or report level.

- **Example** : Filtering data for **Region** or **Product Type** so that all visuals are updated based on the selected filter.

Interactive Visual Elements

3. Dynamic Titles

Create dynamic titles that change based on user selections to give a more personalized experience using DAX.

- **Example** : Show “Sales for [Region]” as a dynamic title

```
Dynamic_Title
= "Sales for " & SELECTEDVALUE(Sales[Region])
```

DAX in Power Bi

Sections

Section 1

Section 2

Section 3

What is DAX?

- DAX (Data Analysis Expressions) is a formula language
- Used in Power BI, Power Pivot, and SQL Server Analysis Services (SSAS)
- To perform data analysis and create calculated columns, measures, and custom tables.

Importance of DAX in Power BI and Data Analysis

- Enhanced Data Modeling:
  go beyond basic aggregations and apply complex business logic within your data model.
- Real-time Insights:
  create dynamic measures that update automatically based on filters and interactions within dashboards.
- Data Relationships:
  provides the flexibility to analyze data from multiple perspectives, e.g., comparing sales data by product categories or regions.

DAX is essential for advanced data modeling, applying complex logic, creating dynamic/real-time insights, and analyzing data within dashboards.

DAX Syntax and Structure

- **Basic Formula:**
  All DAX formulas begin with an equals sign (=), similar to Excel. However, DAX works with tables and columns rather than individual cells
  Example: = SUM(Sales[Amount])
- **Referencing Columns:**
  Columns are referenced using table and column names inside square brackets.
  Example: Sales[Amount] refers to the "Amount" column in the "Sales" table.

Common DAX Functions

Aggregation Functions

SUM: Adds up all the values in a column.

```
= SUM(Sales[Amount])
```

AVERAGE: Returns the average of the values in a column.

```
= AVERAGE(Sales[Amount])
```

Common DAX Functions

Counting Functions

COUNT: Counts the number of values in a column.

```
= COUNT(Sales[OrderID])
```

DISTINCTCOUNT: Counts distinct values in a column.

```
= DISTINCTCOUNT(Sales[CustomerID])
```

Calculated Columns and Measures

Calculated Columns:

Evaluated row-by-row in a table. Stored physically in the data model.

Example: A "Profit" column calculated by:

```
Sales[Amount] - Sales[Cost]
```

Creating Simple Calculated Columns

Go to the "Data" view in Power BI, select your table, and click on "New Column."

Write a formula like:

```
Profit = Sales[Amount] - Sales[Cost]
```

Calculated Measures

Measures:

Calculated on demand and adjust dynamically based on filters.

Do not physically exist in the data model;
they are evaluated only in visualizations.

Example: A measure for total sales calculated by
SUM(Sales[Amount])

Creating Basic Measures

Go to the "Modeling" tab and click on "New Measure."

Write a formula to create a dynamic measure for total sales:

```
Total Sales = SUM(Sales[Amount])
```

Hands-on Activity 1

Scenario: Sales Data Analysis

Task 1: Create Basic Measures

Total Sales: Total Sales = SUM(Sales[Amount])

Average Sales = AVERAGE(Sales[Amount])

Distinct Customers = DISTINCTCOUNT(Sales[CustomerID])

Task 2: Create a Calculated Column

Profit = Sales[Amount] - Sales[Cost]

DAX: Intermediate

Sections

Section 1

Section 2

Section 3

Context in DAX - Row Context

- Think of row context like a situation where DAX is looking at **one row at a time** in a table.
- For example, if you create a calculated column, DAX will process each row in the table one by one.
- Sales[Amount] * Sales[Quantity]
  DAX will take the values from each row and do that multiplication for each row separately.
- It's like going through a list, one item at a time, and calculating for that item alone.

Context in DAX - Filter Context

- Filter is what happens when you apply filters to your data in Power BI (like filtering by region, product, or date)
- When you create a measure (a dynamic calculation), DAX considers the filters applied to the data (e.g., "East" region, "Electronics" category), and calculates based on those filters.

FILTER Function

- This function in DAX allows you to apply specific conditions to a table. It helps you retrieve only the rows that meet certain criteria.
- Think of it like saying, “I only want to look at rows where the product price is greater than 100” or “I only want to see data for customers from the 'East' region.”

```
High_Price_Sales
= CALCULATE(SUM(Sales[Amount]), FILTER(Sales, Sales[Price] > 100))
```

Another Example

- This function in DAX allows you to apply specific conditions to a table. It helps you retrieve only the rows that meet certain criteria.
- Think of it like saying, “I only want to look at rows where the product price is greater than 100” or “I only want to see data for customers from the 'East' region.”

```
Sales_East_Electronics
= CALCULATE(   SUM(Sales[Amount]),    FILTER(Sales, Sales[Region] =
"East" &&  Sales[Category] = "Electronics"))
```

Hands-on Activity 2

Scenario: Sales Data Analysis

Task: Context-Based Analysis

Calculate total sales for specific regions

Calculate sales for the “Technology" category in the "East"

More Advanced DAX Techniques

Aggregation functions

SUMX

- Performs a row-by-row calculation and then sums the result. It's used when you need to calculate values for each row before aggregating.
- Example: Calculate the total sales (price * quantity) for each order, then sum up the results.

```
Total_Sales = SUMX(Sales, Sales[Quantity] * Sales[Price])
```

Aggregation functions

AVERAGEX

- Similar to **SUMX**, but calculates the average instead of the sum.
- Example: Calculate the average sales per order

```
Average_Sales_Per_Order
= AVERAGEX(Sales, Sales[Quantity] * Sales[Price])
```

Iterative Functions

AVERAGE **SUM**

- operate directly on a column of values in a table.

Relationships and Related Functions

Relationships

- Relationships between tables allow you to pull data from one table to another
(e.g., linking **Sales** with **Products**)
- Example: Retrieve the product category from a related **Products** table

```
Product_Category = RELATED(Products[Category])
```

Best Practices

Best Practices for Efficient DAX

- Avoid complex DAX formulas when possible; keep them simple and readable.
- Use variables to store intermediate results for better readability and efficiency.
- If you don’t need to iterate row-by-row, use simple aggregation functions like SUM or AVERAGE
- Avoid using too many calculated columns in large datasets, as they can increase memory usage.
- Pulling large sets of related data can slow down calculations, so use this RELATEDTABLE selectively.

Resources

- Power BI Gallery - Themes Gallery - Microsoft Fabric Community
- Free to use themes and resources - PowerBI.tips

That’s a wrap

Any Questions?