Introduction to DAX

Importance of DAX in Power BI

Data Analysis Expressions (DAX) is a collection of functions, operators, and constants used in Power BI (as well as in other Microsoft data tools like Power Pivot and SQL Server Analysis Services). It allows you to create new information from the data already in your model.

Why is DAX important?

- **Rich Data Calculations**: DAX provides more advanced and complex calculations than standard Excel formulas. It's specifically designed for data modeling and reporting.
- **Dynamic Aggregations**: DAX can dynamically aggregate data based on filters applied, enabling interactive reports and dashboards.
- **Advanced Analysis**: With DAX, you can perform time-based analysis, compare data across different periods, and calculate growth percentages, among other advanced data analysis tasks.
- **Enhanced Data Models**: DAX helps you create calculated columns and measures, adding new data derived from existing data to your model.

Difference between DAX and Other Formula Languages

DAX vs Excel Formulas:

- **Context Awareness**: DAX formulas consider the context of the data in Power BI reports, such as filters and slicers, which Excel formulas do not.
- **Relationship Handling**: DAX can naturally navigate and use relationships between tables, whereas, in Excel, you'd have to use functions like VLOOKUP or INDEX/MATCH.
- **Recalculation**: DAX formulas are recalculated when data refreshes, ensuring that measures always reflect current data.

DAX vs. SQL:

- While SQL is used primarily for querying databases and returning sets of rows, DAX is used for analysis, calculating metrics, and KPIs based on data already loaded into your model.
- DAX operates on a row-by-row basis when it needs to, much like SQL, but it is primarily designed to work with data tables and return single scalar values.

Basic Concepts and Terminology

- **Calculated Columns**: You add columns to existing tables in the Power BI model using DAX formulas. The values are calculated once (when the data is loaded or refreshed) and stored in the model.
  Example: Total Sales = 'transaction' [Unit Price] * 'transaction' [Quantity]

- **Measures**: Measures are calculations used in reporting and analysis, calculated on the fly based on the current context. They are not stored in the data model.

- **DAX Functions**: These are pre-defined formulas that perform calculations using the data in your tables. Examples include SUM(), COUNT(), AVERAGE(), MIN(), MAX(), and more complex functions like CALCULATE(), ALL(), FILTER().

- **DAX Operators**: These are symbols or words that specify the type of calculation or comparison to be performed. Examples include + (addition), **-** (subtraction), * (multiplication), / (division), **&&** (logical AND), || (logical OR).

Basic Arithmetic Operations

These are fundamental operations performed on numerical data:

Create Cost Column by assuming that each unit cost = 70% of the unit price

Unit Cost = 'Transaction'[UnitPrice] * 0.7

1. **Addition**:
   - Formula: **[Column1] + [Column2]**
   - Example: **Complete Unit Cost = 'Transaction'[Unit Cost] + 'Transaction'[UnitPrice] * 'Transaction'[TaxRate]**
2. **Multiplication**:
   - Formula: **[Column1] * [Column2]**
   - Example: **Total Cost = 'Transaction' [Complete Unit Cost] * 'Transaction' [Quantity]**

3. **Subtraction**:
   - Formula: **[Column1] - [Column2]**
   - Example: **Total Profit = 'Transaction' [totalSales] - 'Transaction' [Total Cost]**

4. **Division**:

    - Formula: **[Column1] / [Column2]**
    - Example: **Unit Profit = 'Transaction' [Total Profit] / 'Transaction' [Quantity]**
    - Note: When dividing, consider using the DIVIDE function instead to handle division by zero more gracefully: **Unit Profit = DIVIDE('Transaction' [Total Profit], 'Transaction' [Quantity])**
5. **Percentage**:
    - Formula: **[Part] / [Total]**
    - Example: **Profit Percentage = 'Transaction' [Unit Profit] / 'Transaction' [Unit Price]**

POWER: Raises a number to the power of another number.
SQRT: Square root of a number.
DIVIDE: Safe division allows a fallback value if you're dividing by zero.
ROUND: Rounds a number to a specified number of decimals.

Logical Tests

These operations allow you to perform tests on your data and return results based on logical conditions:

1. **IF Statements**:
    - Formula: **IF([Logical Test], [Result if True], [Result if False])**
    - Example: **Profitable = IF('Transaction' [Unit Profit] > 0, "Yes", "No")**
2. **AND Logic**:
    - Formula: **IF([Condition1] && [Condition2], [Result if True], [Result if False])**
    - Example: **Large Profitable Sale = IF('Transaction'[unitPrice] > 10000 && [Profit Percentage] > 0.1, "Yes", "No")**
3. **OR Logic**:
    - Formula: **IF([Condition1] || [Condition2], [Result if True], [Result if False])**
    - Example: **High Quantity or Value = IF('Transaction' [Quantity] > 100 || 'Transaction' [Total Sales] > 500, "Yes", "No")**

Examples:

Identify if a transaction included a high quantity of items (let's say 3 or more).

High Quantity Purchase = IF('Transaction' [Quantity] >= 3, "Yes", "No")

Check if a transaction is high value ($1000 or more) and high quantity (5 items or more).

High Value and Quantity = IF('Transaction' [Quantity] >= 5 && 'Transaction' [Quantity] * 'Transaction' [UnitPrice] >= 1000, "Yes", "No")

Check if a transaction is either high value ($1000 or more) or high quantity (5 items or more).

Large Value or Quantity = IF('Transaction' [Quantity] >= 5 || 'Transaction' [Quantity] * 'Transaction' [UnitPrice] >= 1000, "Yes", "No")