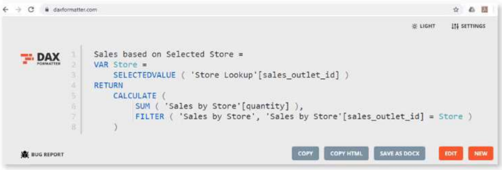
DAX_Intro

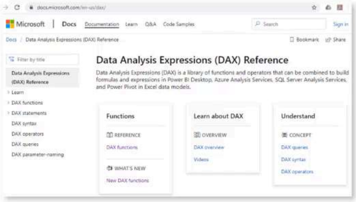
Resource

HELPFUL DAX RESOURCES

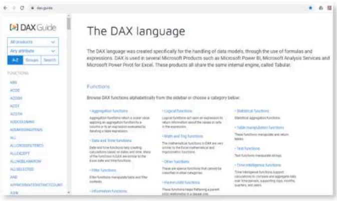


```
Sales based on Selected Store =
VAR Store =
    SELECTEDVALUE ( 'Store Lookup'[sales_outlet_id] )
RETURN
    CALCULATE (
        SUM ( 'Sales by Store'[quantity] ),
        FILTER ( 'Sales by Store', 'Sales by Store'[sales_outlet_id] = Store )
    )
```

DAX Formatter *(daxformatter.com)* by sqlbi.com is a great tool for cleaning and formatting your DAX code, with options to customize based on regional settings or personal preferences



Data Analysis Expressions (DAX) Reference



The DAX language

www.Docs.Microsoft.com/en-us/dax is the official Microsoft DAX reference guide, and a great resource for exploring DAX functions

DAX Guide *(dax.guide)* by sqlbi.com is a great resource to learn and explore DAX functions, category groups, product compatibility, and more

DAX Studio is an open source tool that allows you to write, execute and analyze DAX queries, as well as troubleshoot and optimize your code

Data Table vs Lookup Table

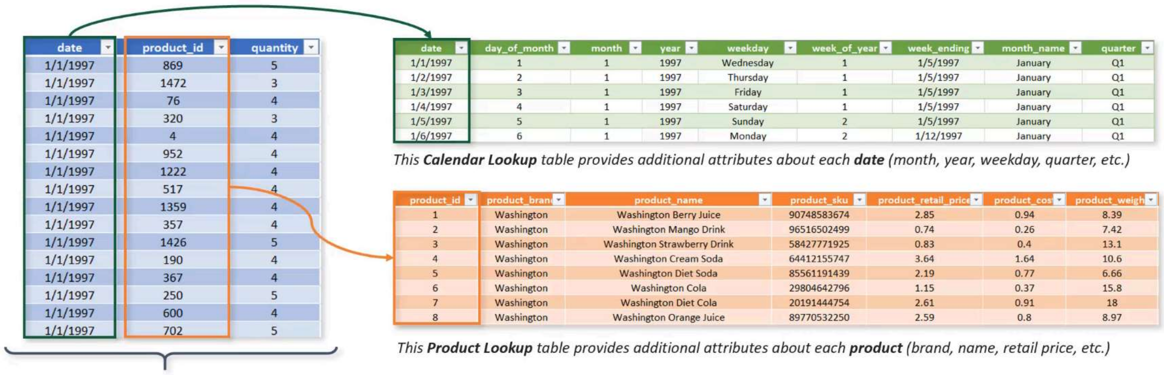
Here's a comparison between a **data table** and a **lookup table** in a tabular format:

Feature	Data Table	Lookup Table
Content	Contains actual transactional or factual or Measurable data (e.g., sales amounts, dates).	Contains dimension or descriptive data for lookup (e.g., product names, customer types).
Purpose	Stores primary business data or facts for analysis (e.g., sales, orders).	Stores reference information used to enrich or categorize data (e.g., product details, region names).
Size	Usually larger with many records (transactions, events).	Typically smaller, containing fewer records (unique categories, types).
Relationships	Often forms the "many" side in one-to-many relationships.	Forms the "one" side in one-to-many relationships.
Updates	Frequently updated as new transactions occur.	Rarely updated; mostly static data for reference.
Examples	Sales transactions, customer orders, inventory records.	Product categories, region codes, customer types.
Usage	Used for reporting and analysis of business metrics.	Used for joining with data tables to provide additional context or metadata.

DATA TABLES VS. LOOKUP TABLES

Models generally contain two types of tables: **data** (or *"fact"*) tables, and **lookup** (or *"dimension"*) tables

- **Data tables** contain measurable *values* or *metrics* about the business *(quantity, revenue, pageviews, etc.)*
- **Lookup tables** provide descriptive *attributes* about each dimension in your model *(customers, products, etc.)*



date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

*This **Calendar Lookup** table provides additional attributes about each **date** (month, year, weekday, quarter, etc.)*

product_id	product_bran	product_name	product_sku	product_retail_price	product_cos	product_weigh
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

*This **Product Lookup** table provides additional attributes about each **product** (brand, name, retail price, etc.)*

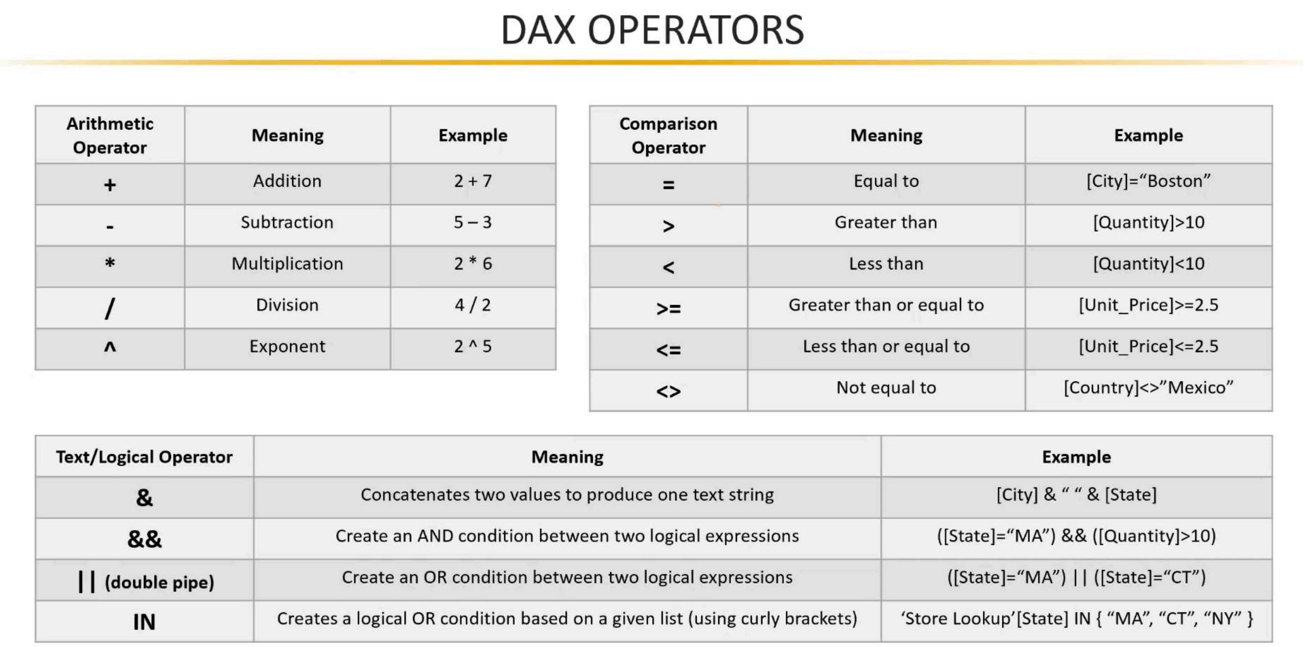
*This **Data Table** contains "**quantity**" values, and connects to lookup tables via the "**date**" and "**product_id**" columns*

Primary Key V/S Foreign Key

Here's a comparison of the **Primary Key** and **Foreign Key** in the context of Power BI:

Feature	Primary Key	Foreign Key
Purpose	Uniquely identifies each record in a table (lookup/dimension table).	Refers to the primary key in another table (data/fact table). Used to link tables together.
Table Type	Found in lookup (dimension) tables, like `Products` or `Customers`.	Found in data (fact) tables, like `Sales` or `Orders`.
Uniqueness	Values must be unique for each record (e.g., `ProductID`, `CustomerID`).	Values can repeat and refer to multiple records (e.g., multiple sales for the same `ProductID`).
Role in Relationships	Forms the "one" side of a one-to-many relationship.	Forms the "many" side of a one-to-many relationship.
Example in Power BI	`ProductID` in a `Products` table uniquely identifies each product.	`ProductID` in a `Sales` table links back to the `ProductID` in the `Products` table.
Usage in Model	Used for filtering and slicing the fact table data based on categories in the lookup table.	Used to aggregate and analyze facts (e.g., total sales) based on attributes in the related lookup table.
Power BI Relationship	Typically, you create a **one-to-many** relationship, where the primary key connects to the foreign key.	Part of a **many-to-one** relationship with the primary key in the dimension table.

DAX operators



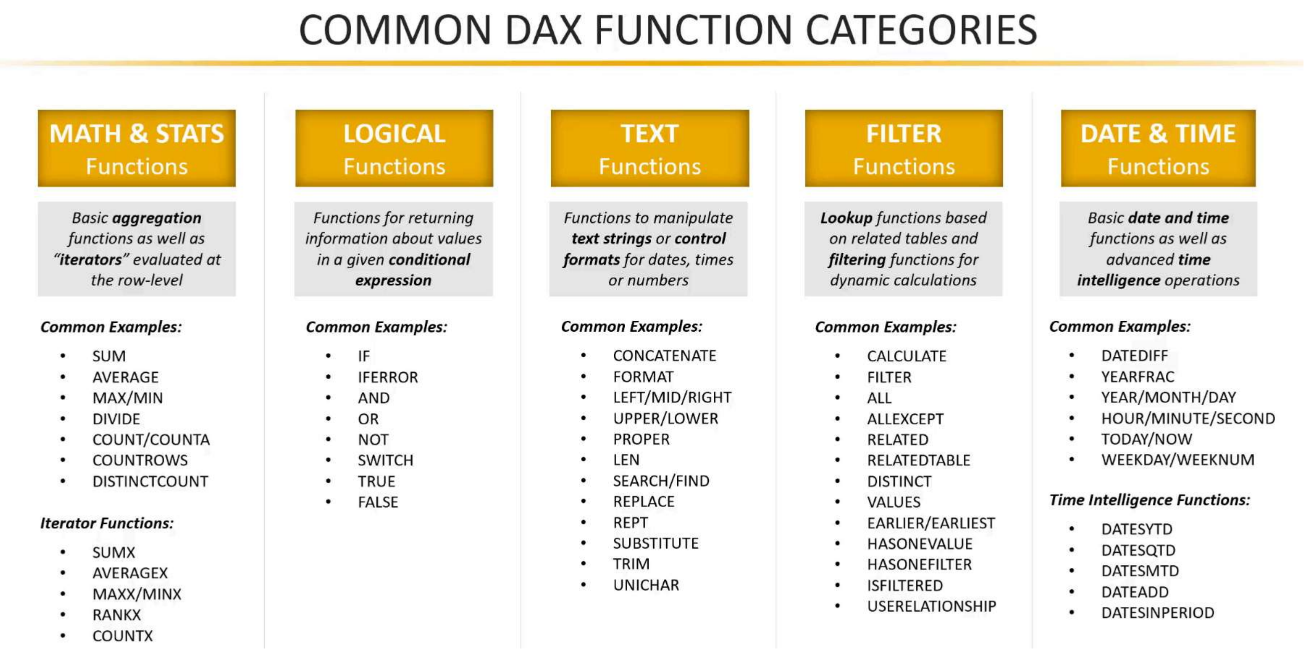
DAX OPERATORS

Arithmetic Operator	Meaning	Example
+	Addition	2 + 7
-	Subtraction	5 – 3
*	Multiplication	2 * 6
/	Division	4 / 2
^	Exponent	2 ^ 5

Comparison Operator	Meaning	Example
=	Equal to	[City]=“Boston”
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
>=	Greater than or equal to	[Unit_Price]>=2.5
<=	Less than or equal to	[Unit_Price]<=2.5
<>	Not equal to	[Country]<>”Mexico”

Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	[City] & “ “ & [State]
&&	Create an AND condition between two logical expressions	([State]=“MA”) && ([Quantity]>10)
|| (double pipe)	Create an OR condition between two logical expressions	([State]=“MA”) || ([State]=“CT”)
IN	Creates a logical OR condition based on a given list (using curly brackets)	‘Store Lookup’[State] IN { “MA”, “CT”, “NY” }

Common DAX function



COMMON DAX FUNCTION CATEGORIES

MATH & STATS Functions

*Basic **aggregation** functions as well as “**iterators**” evaluated at the row-level*

Common Examples:

- SUM
- AVERAGE
- MAX/MIN
- DIVIDE
- COUNT/COUNTA
- COUNTROWS
- DISTINCTCOUNT

Iterator Functions:

- SUMX
- AVERAGEX
- MAXX/MINX
- RANKX
- COUNTX

LOGICAL Functions

*Functions for returning information about values in a given **conditional expression***

Common Examples:

- IF
- IFERROR
- AND
- OR
- NOT
- SWITCH
- TRUE
- FALSE

TEXT Functions

*Functions to manipulate **text strings** or **control formats** for dates, times or numbers*

Common Examples:

- CONCATENATE
- FORMAT
- LEFT/MID/RIGHT
- UPPER/LOWER
- PROPER
- LEN
- SEARCH/FIND
- REPLACE
- REPT
- SUBSTITUTE
- TRIM
- UNICHAR

FILTER Functions

***Lookup** functions based on related tables and **filtering** functions for dynamic calculations*

Common Examples:

- CALCULATE
- FILTER
- ALL
- ALLEXCEPT
- RELATED
- RELATEDTABLE
- DISTINCT
- VALUES
- EARLIER/EARLIEST
- HASONEVALUE
- HASONEFILTER
- ISFILTERED
- USERELATIONSHIP

DATE & TIME Functions

*Basic **date and time** functions as well as advanced **time intelligence** operations*

Common Examples:

- DATEDIFF
- YEARFRAC
- YEAR/MONTH/DAY
- HOUR/MINUTE/SECOND
- TODAY/NOW
- WEEKDAY/WEEKNUM

Time Intelligence Functions:

- DATESYTD
- DATESQTD
- DATESMTD
- DATEADD
- DATESINPERIOD

Math & Stats Functions

Function	Syntax	Example
SUM	`SUM(<ColumnName>)`	`TotalSales = SUM(Sales[SalesAmount])`
AVERAGE	`AVERAGE(<ColumnName>)`	`AvgPrice = AVERAGE(Products[UnitPrice])`
MAX	`MAX(<ColumnName>)`	`MaxSales = MAX(Sales[SalesAmount])`
MIN	`MIN(<ColumnName>)`	`MinSales = MIN(Sales[SalesAmount])`
DIVIDE	`DIVIDE(<Numerator>, <Denominator>[, <AlternativeResult>])`	`ProfitMargin = DIVIDE(Sales[Profit], Sales[Revenue], 0)`

Function	Syntax	Example
COUNT	`COUNT(<ColumnName>)`	`TotalOrders = COUNT(Orders[OrderID])`
COUNTA	`COUNTA(<ColumnName>)`	`TotalItems = COUNTA(Inventory[ItemName])`
COUNTROWS	`COUNTROWS(<TableName>)`	`TotalRows = COUNTROWS(Customers)`
DISTINCTCOUNT	`DISTINCTCOUNT(<ColumnName>)`	`DistinctCustomers = DISTINCTCOUNT(Sales[CustomerID])`

Iterator Functions

Function	Syntax	Example
SUMX	`SUMX(<Table>, <Expression>)`	`TotalProfit = SUMX(Sales, Sales[Revenue] - Sales[Cost])`
AVERAGEX	`AVERAGEX(<Table>, <Expression>)`	`AvgProfit = AVERAGEX(Sales, Sales[Revenue] - Sales[Cost])`
MAXX	`MAXX(<Table>, <Expression>)`	`MaxProfit = MAXX(Sales, Sales[Revenue] - Sales[Cost])`
MINX	`MINX(<Table>, <Expression>)`	`MinProfit = MINX(Sales, Sales[Revenue] - Sales[Cost])`
RANKX	`RANKX(<Table>, <Expression>[, <Value>, <Order>, <Ties>])`	`RankSales = RANKX(ALL(Sales), Sales[SalesAmount])`
COUNTX	`COUNTX(<Table>, <Expression>)`	`CountProducts = COUNTX(Products, Products[Category])`

Logical Functions

Here's a **table format** with the **syntax and example** for each **logical function** mentioned in the image based on **Power BI DAX functions**:

Function	Syntax	Example
IF	`IF(<Condition>, <TrueResult>, <FalseResult>)`	`ProfitCheck = IF(Sales[Profit] > 1000, "High", "Low")`
IFERROR	`IFERROR(<Value>, <AlternateResult>)`	`SafeDivide = IFERROR(DIVIDE(Sales[Profit], Sales[Cost]), 0)`
AND	`AND(<Condition1>, <Condition2>)`	`CheckStatus = IF(AND(Sales[Profit] > 1000, Sales[Cost] < 500), "Good", "Bad")`
OR	`OR(<Condition1>, <Condition2>)`	`CheckDiscount = IF(OR(Sales[Discount] > 10, Sales[Profit] > 500), "Valid", "Invalid")`
NOT	`NOT(<Condition>)`	`NotProfitable = IF(NOT(Sales[Profit] > 0), "Loss", "Profit")`
SWITCH	`SWITCH(<Expression>, <Value1>, <Result1>, ..., <ElseResult>)`	`CategoryCheck = SWITCH(Products[Category], "Electronics", "E1", "Clothing", "C1", "Other")`

Function	Syntax	Example
TRUE	`TRUE()`	`AlwaysTrue = IF(TRUE(), "Yes", "No")`
FALSE	`FALSE()`	`AlwaysFalse = IF(FALSE(), "No", "Yes")`

Text Functions

Here's a **table format** with the **syntax and example** for each **text function** mentioned in the image based on **Power BI DAX functions**:

Function	Syntax	Example
CONCATENATE	`CONCATENATE(<Text1>, <Text2>)`	`FullName = CONCATENATE(Employee[FirstName], Employee[LastName])`
FORMAT	`FORMAT(<Value>, <FormatString>)`	`FormattedDate = FORMAT(Sales[Date], "DD-MM-YYYY")`
LEFT	`LEFT(<Text>, <NumberOfCharacters>)`	`FirstThree = LEFT(Customer[PhoneNumber], 3)`
MID	`MID(<Text>, <StartPosition>, <NumberOfCharacters>)`	`MiddlePart = MID(Product[SKU], 2, 3)`
RIGHT	`RIGHT(<Text>, <NumberOfCharacters>)`	`LastTwo = RIGHT(Customer[PhoneNumber], 2)`
UPPER	`UPPER(<Text>)`	`UpperCaseName = UPPER(Employee[FirstName])`
LOWER	`LOWER(<Text>)`	`LowerCaseName = LOWER(Employee[LastName])`
PROPER	`PROPER(<Text>)`	`ProperName = PROPER(Customer[FullName])`
LEN	`LEN(<Text>)`	`NameLength = LEN(Customer[FullName])`
SEARCH	`SEARCH(<FindText>, <WithinText>[, <StartPosition>])`	`SearchResult = SEARCH("Sales", Sales[Comment], 1)`
FIND	`FIND(<FindText>, <WithinText>[, <StartPosition>, <NotFoundValue>])`	`FindResult = FIND("A", Customer[Name])`
REPLACE	`REPLACE(<OldText>, <Start>, <NumChars>, <NewText>)`	`CorrectedPhone = REPLACE(Customer[Phone], 4, 3, "123")`
REPT	`REPT(<Text>, <NumberOfTimes>)`	`RepeatedText = REPT("-", 5)`
SUBSTITUTE	`SUBSTITUTE(<Text>, <OldText>, <NewText>[, <InstanceNum>])`	`CorrectedText = SUBSTITUTE(Customer[Name], "Inc.", "LLC")`
TRIM	`TRIM(<Text>)`	`CleanName = TRIM(Customer[FullName])`
UNICHAR	`UNICHAR(<Number>)`	`UnicodeChar = UNICHAR(169)`

Filter Functions

Here's a **table format** with the **syntax and example** for each **filter function** mentioned based on **Power BI DAX functions**:

Function	Syntax	Example
CALCULATE	`CALCULATE(<Expression>, <Filter1>, <Filter2>, ...)`	`TotalSalesFiltered = CALCULATE(SUM(Sales[SalesAmount]), Sales[Region] = "North")`
FILTER	`FILTER(<Table>, <Expression>)`	`FilteredSales = FILTER(Sales, Sales[SalesAmount] > 1000)`
ALL	`ALL(<Table/Column>)`	`AllProducts = CALCULATE(SUM(Sales[SalesAmount]), ALL(Products))`
ALLEXCEPT	`ALLEXCEPT(<Table>, <Column1>, <Column2>, ...)`	`SalesWithoutCategory = CALCULATE(SUM(Sales[SalesAmount]), ALLEXCEPT(Sales, Sales[Category]))`
RELATED	`RELATED(<Column>)`	`ProductName = RELATED(Products[ProductName])`
RELATEDTABLE	`RELATEDTABLE(<Table>)`	`TotalOrders = COUNTROWS(RELATEDTABLE(Orders))`
DISTINCT	`DISTINCT(<Column>)`	`DistinctRegions = DISTINCT(Sales[Region])`
VALUES	`VALUES(<Column>)`	`RegionValues = VALUES(Sales[Region])`
EARLIER	`EARLIER(<Column>[, <Number>)`	`SalesRank = RANKX(FILTER(Sales, Sales[SalesAmount] < EARLIER(Sales[SalesAmount])), Sales[SalesAmount])`
EARLIEST	`EARLIEST(<Column>[, <Number>])`	`EarliestSaleDate = EARLIEST(Sales[SaleDate])`
HASONEVALUE	`HASONEVALUE(<Column>)`	`IsSingleRegion = IF(HASONEVALUE(Sales[Region]), "Yes", "No")`
HASONEFILTER	`HASONEFILTER(<Column>)`	`IsFiltered = HASONEFILTER(Sales[Category])`
ISFILTERED	`ISFILTERED(<Column>)`	`CheckFilter = IF(ISFILTERED(Sales[Category]), "Filtered", "Not Filtered")`
USERELATIONSHIP	`USERELATIONSHIP(<Column1>, <Column2>)`	`TotalSalesWithAltRel = CALCULATE(SUM(Sales[SalesAmount]), USERELATIONSHIP(Sales[OrderDate], Calendar[Date]))`

Date & Time function

Here's a **table format** with the **syntax and example** for each **date and time function** based on **Power BI DAX functions**:

Function	Syntax	Example
DATEDIFF	`DATEDIFF(<StartDate>, <EndDate>, <Interval>)`	`DaysBetween = DATEDIFF(Orders[OrderDate], Orders[ShipDate], DAY)`
YEARFRAC	`YEARFRAC(<StartDate>, <EndDate>)`	`FractionYear = YEARFRAC(Orders[StartDate], Orders[EndDate])`

The DATEDIFF function is used to calculate the difference between two dates and return the result in the specified interval, such as days, months, quarters, or years.

The YEARFRAC function can calculate the fraction of the year between two dates.

(2023-01-01 to 2023-07-01), the fraction of the year calculated is approximately 0.5 (since it spans about six months).

Function	Syntax	Example
YEAR	YEAR(<Date>)	OrderYear = YEAR(Orders[OrderDate])
MONTH	MONTH(<Date>)	OrderMonth = MONTH(Orders[OrderDate])
DAY	DAY(<Date>)	OrderDay = DAY(Orders[OrderDate])
HOUR	HOUR(<DateTime>)	OrderHour = HOUR(Orders[OrderDateTime])
MINUTE	MINUTE(<DateTime>)	OrderMinute = MINUTE(Orders[OrderDateTime])
SECOND	SECOND(<DateTime>)	OrderSecond = SECOND(Orders[OrderDateTime])
TODAY	TODAY()	CurrentDate = TODAY()
NOW	NOW()	CurrentDateTime = NOW()
WEEKDAY	WEEKDAY(<Date>[, <ReturnType>])	DayOfWeek = WEEKDAY(Orders[OrderDate], 1)
WEEKNUM	WEEKNUM(<Date>[, <ReturnType>])	OrderWeek = WEEKNUM(Orders[OrderDate], 1)
DATESYTD	DATESYTD(<DatesColumn>[, <YearEndDate>])	SalesYTD = CALCULATE(SUM(Sales[SalesAmount]), DATESYTD(Calendar[Date]))
DATESQTD	DATESQTD(<DatesColumn>)	SalesQTD = CALCULATE(SUM(Sales[SalesAmount]), DATESQTD(Calendar[Date]))
DATESMTD	DATESMTD(<DatesColumn>)	SalesMTD = CALCULATE(SUM(Sales[SalesAmount]), DATESMTD(Calendar[Date]))
DATEADD	DATEADD(<DatesColumn>, <NumberOfIntervals>, <Interval>)	SalesLastYear = CALCULATE(SUM(Sales[SalesAmount]), DATEADD(Calendar[Date], -1, YEAR))
DATESINPERIOD	DATESINPERIOD(<DatesColumn>, <StartDate>, <NumberOfIntervals>, <Interval>)	SalesInPeriod = CALCULATE(SUM(Sales[SalesAmount]), DATESINPERIOD(Calendar[Date], TODAY(), -30, DAY))

The DATESQTD function is used to return a date range from the start of the current quarter up to the specified date

The DATEADD function in Power BI DAX is used to shift a set of dates by a specified number of intervals.

The DATESINPERIOD function is used to return a date range that includes all dates within a specified period relative to a given start date

Calculated Column v/s Measure

Feature	Calculated Columns	Measures
Definition	A column added to a table that contains values calculated row by row using DAX.	A dynamic calculation performed at the time of report viewing using DAX.
Calculation Scope	Computed for each row in the table and stored in the data model.	Computed dynamically based on the context of the visual or report.
Storage	Values are stored in the table as part of the data model.	Not stored; recalculated each time they are used in a visual.
Performance Impact	Can impact performance, especially with large tables, as they increase the model size.	More efficient in large datasets, as they calculate only when needed.

Feature	Calculated Columns	Measures
Row-Level or Aggregate	Works at the row level (e.g., row-by-row calculations).	Works at an aggregate level based on the context (e.g., total sales, average quantity).
Usage in Filters/Slicers	Can be used as fields in slicers and filters, just like any other column.	Cannot be used as slicers or filters directly; they can only be displayed in visuals.
Recalculation Frequency	Calculated once when the data model is refreshed or when the column is created.	Recalculated dynamically whenever a visual or report using the measure is refreshed.
Examples	Concatenating `FirstName` & `LastName` into a `FullName` column.	Summing up `SalesAmount` in a visual to calculate total sales.
Memory Usage	Increases memory usage as calculated columns are stored in the model.	Low memory usage since measures are not stored but recalculated.
Best Use Case	Best for row-by-row calculations and adding extra fields to a table (e.g., `Profit = Revenue - Cost`).	Best for aggregate or summary calculations in visuals (e.g., `Total Sales`, `Average Price`).
Examples of DAX	`Profit = [Revenue] - [Cost]` (row-by-row calculation)	`TotalSales = SUM([SalesAmount])` (aggregate calculation)

Filter Context v/s Row context

Feature	Filter Context	Row Context
Definition	The set of filters applied to a data model, either through slicers, filters, or visuals, affecting the calculation results.	The context of a single row where calculations are evaluated for each row individually in a table.
Scope	Applies to multiple rows or the entire dataset, depending on how filters are applied.	Applies to one row at a time, performing calculations row by row.
Use Case	Used for aggregations and summarizations in visuals (e.g., calculating the total or average of a column for a subset of data).	Used for row-by-row calculations, such as calculated columns or iterating over rows with functions like `SUMX` or `FILTER`.
How It's Triggered	Triggered by slicers, filters, or groupings in Power BI visuals or DAX expressions like `CALCULATE`.	Triggered when evaluating a DAX formula for each row in a table (e.g., when using iterators like `SUMX`, `FILTER`, or in calculated columns).
Example of Use	When you filter a report to only show data for the year 2023, the filter context limits the data to that year, and calculations like `SUM(SalesAmount)` reflect this filter.	When calculating `Profit = [Revenue] - [Cost]` for each row in a table, the DAX formula uses row context to compute the value row by row.
DAX Functionality	Functions like `CALCULATE` can modify filter context by adding or removing filters.	Functions like `SUMX` or `FILTER` operate within row context, iterating over rows in a table.

Feature	Filter Context	Row Context
Key DAX Functions	`CALCULATE` , `FILTER` , `ALL` , `VALUES` , `REMOVEFILTERS`	`SUMX` , `AVERAGEX` , `FILTER` , `RELATED` , `EARLIER`
Example DAX Formula	`CALCULATE(SUM(SalesAmount), Year = 2023)` sums `SalesAmount` but only for rows where `Year` equals 2023.	`TotalProfit = SUMX(Sales, Sales[Revenue] - Sales[Cost])` calculates profit for each row in the `Sales` table and then sums the results.
Visualization Impact	Filter context changes the data that appears in visuals based on user interaction (e.g., filtering by year, product category).	Row context affects calculations within each row of a table (e.g., calculating row-level profit or discount).
Typical Scenario	Aggregating total sales for a specific year or region based on filters applied.	Calculating `Discount` for each individual sale by subtracting a percentage from `Price` .