Remember, the basic structure of a DAX measure is:

MeasureName = Expression

Where:

- MeasureName is the name you give to your measure.
- Expression is the DAX formula that calculates the result.

---

1. Basic Aggregation Functions (Relevant to Experiment 7 - Summarization)

- **SUM:** Adds all the numbers in a column.

MySumMeasure = SUM(TableName[ColumnName])

- **AVERAGE:** Calculates the arithmetic mean of all numbers in a column.

MyAverageMeasure = AVERAGE(TableName[ColumnName])

- **COUNT:** Counts the number of rows in a column that are not blank. (Can count numbers or text).

MyCountMeasure = COUNT(TableName[ColumnName])

- **COUNTA:** Counts the number of rows in a column that are not empty. (Similar to COUNT but handles different data types).

MyCountAMeasure = COUNTA(TableName[ColumnName])

- **DISTINCTCOUNT:** Counts the number of distinct values in a column.

MyDistinctCountMeasure = DISTINCTCOUNT(TableName[ColumnName])

- **MAX:** Returns the largest numeric value in a column, or the latest date.

MyMaxMeasure = MAX(TableName[ColumnName])

- **MIN:** Returns the smallest numeric value in a column, or the earliest date.

MyMinMeasure = MIN(TableName[ColumnName])

---

2. CALCULATE Function (Core to Experiments 5 & 6)

Evaluates an expression in a context modified by filters. This is one of the most powerful and frequently used DAX functions.

- **Syntax:**

MyCalculatedMeasure = CALCULATE(<expression>[, <filter1>[, <filter2>[, ...]]])

    - <expression>: Typically an aggregation measure (e.g., SUM(TableName[SalesAmount]) or another measure like [Total Sales]).
    - <filterN>: A boolean expression or a table expression that defines a filter.

- **Example with a simple filter:**

SalesForUSA = CALCULATE(SUM(Sales[Amount]), Sales[Country] = "USA")

- **Example using another measure as the expression:**

TotalSales = SUM(Sales[Amount])

SalesForElectronics = CALCULATE([TotalSales], Products[Category] = "Electronics")

---

3. FILTER Function (Often used with CALCULATE - Relevant to Experiment 5)

Returns a table that has been filtered. It's an iterator, meaning it evaluates a condition row by row.

- **Syntax:**

FILTER(<table>, <filterExpression>)

    - <table>: A table or a table expression.
    - <filterExpression>: A Boolean expression to be evaluated for each row of the table.

- **Example used within CALCULATE:**

```
SalesHighMarginProducts =
CALCULATE(
  SUM(Sales[Amount]),
  FILTER(
    Products,
    Products[Margin] > 0.40
  )
```

)

---

4. ALL Function Family (Core to Experiment 6)

These functions are used to remove filters from the filter context.

- **ALL(TableName):** Removes all filters from the specified table.

PercentOfTotalSales = DIVIDE(SUM(Sales[Amount]), CALCULATE(SUM(Sales[Amount]), ALL(Sales)))

- **ALL(TableName[ColumnName1], TableName[ColumnName2], ...):** Removes filters from specified columns in a table.

SalesAllRegionsForCurrentProduct = CALCULATE(SUM(Sales[Amount]), ALL(Sales[Region]))

- **ALLSELECTED(TableName[ColumnName] or TableName):** Removes context filters from columns and rows in the current query,[1] while retaining all other context filters or explicit filters. Useful for visual totals that respect slicers outside the visual.

SalesForSelectedCategories = CALCULATE(SUM(Sales[Amount]), ALLSELECTED(Products[Category]))

- **ALLEXCEPT(TableName, TableName[ColumnToKeepFilter1], TableName[ColumnToKeepFilter2], ...):** Removes all context filters in the table except for filters that have been applied to the specified columns.

SalesForCurrentProductCategoryAllElse =

CALCULATE(

  SUM(Sales[Amount]),

  ALLEXCEPT(Products, Products[Category])

)

(This calculates total sales, keeping only the filter on Products[Category] active from the Products table and removing all other filters from that table.)

---

5. Iterator Functions (e.g., SUMX, AVERAGEX - Generally Useful, can enhance Summarization in Experiment 7)

These functions iterate over each row of a given table and evaluate an expression for each row. Then they perform an aggregation on the results of those expressions.

- **SUMX(<table>, <expression>):**

TotalRevenue = SUMX(Sales, Sales[Quantity] * Sales[UnitPrice])

*(Calculates Quantity * UnitPrice for each row in Sales and then sums up these results.)*

- **AVERAGEX(<table>, <expression>):**

AverageTransactionValue = AVERAGEX(Sales, Sales[Quantity] * Sales[UnitPrice])

---

6. SUMMARIZE Function (For creating calculated tables - Relevant to Experiment 7)

Returns a summary table for the requested totals over a set of groups. *Note: While SUMMARIZE creates tables, the aggregate functions within it are similar to measure expressions.*

- **Syntax:**

SUMMARIZE(

<table>,

<groupBy_columnName1>[, <groupBy_columnName2>]...,

[<nameX>, <expressionX>]...

)

    - **Example (This is for a calculated table, not a measure):** <!-- end list -->

SalesSummaryByRegion =

SUMMARIZE(

Sales,

Sales[Region],

"TotalSalesAmount", SUM(Sales[Amount]),

"NumberOfTransactions", COUNT(Sales[TransactionID])

)

---

7. DIVIDE Function (Good Practice)

Performs division and provides an alternative value if division by zero occurs (or handles BLANK() denominator).

- **Syntax:**

DIVIDE(<numerator>, <denominator>[, <alternateResult>])

    - <alternateResult> is optional; if omitted and division by zero occurs, it returns BLANK().
- **Example:**

ProfitMargin = DIVIDE(SUM(Sales[Profit]), SUM(Sales[Revenue]), 0)

(Returns 0 if total revenue is zero or blank)

---

8. Logical Functions (e.g., IF - Generally Useful)

- **IF(<logical_test>, <value_if_true>, <value_if_false>):** Checks a condition, and returns one value if TRUE, otherwise returns another value.

SalesPerformance =

IF(

  SUM(Sales[Amount]) > 100000,

  "High Performance",

  "Standard Performance"

)