Basic Calculations

1. SUM

Calculates the sum of a column.

TotalSales = SUM(Sales[SalesAmount])

2. AVERAGE

Calculates the average value of a column.

AverageSales = AVERAGE(Sales[SalesAmount])

3. COUNT

Counts the number of rows in a column that are not empty.

NumberOfSales = COUNT(Sales[SalesAmount])

4. COUNTA

Counts the number of non-empty values in a column.

CountNonEmptyValues = COUNTA(Sales[ProductName])

5. MIN

Finds the minimum value in a column.

MinSalesAmount = MIN(Sales[SalesAmount])

6. MAX

Finds the maximum value in a column.

MaxSalesAmount = MAX(Sales[SalesAmount])

7. IF

Performs a logical test and returns different values based on the result.

SalesCategory = IF(Sales[SalesAmount] > 1000, "High", "Low")

8. CALCULATE

Changes the context in which data is evaluated and allows for more complex aggregations.

TotalSales2023 = CALCULATE(SUM(Sales[SalesAmount]), Sales[Year] = 2023)

9. DATEADD

Shifts a date by a specified number of intervals.

PreviousMonthSales = CALCULATE(SUM(Sales[SalesAmount]), DATEADD(Sales[Date], -1, MONTH))

10. FORMAT

Formats a value according to a specified format string.

FormattedSales = FORMAT(SUM(Sales[SalesAmount]), "$#,##0.00")

11. DISTINCTCOUNT

Counts the number of unique values in a column.

UniqueProducts = DISTINCTCOUNT(Sales[ProductID])

12. RELATED

Fetches a related value from another table.

ProductCategory = RELATED(Product[Category])

13. ALL

Removes all filters from a table or column.

TotalSalesAllYears = CALCULATE(SUM(Sales[SalesAmount]), ALL(Sales[Year]))

14. FILTER

Returns a table that has been filtered according to a specified condition.

HighValueSales = FILTER(Sales, Sales[SalesAmount] > 1000)

15. SUMX

Calculates the sum of an expression evaluated for each row in a table.

TotalProfit = SUMX(Sales, Sales[SalesAmount] - Sales[Cost])

16. AVERAGEX

Calculates the average of an expression evaluated for each row in a table.

```
AverageProfit = AVERAGEX(Sales, Sales[SalesAmount] - Sales[Cost])
```

17. EARLIER

Accesses data from an earlier row context within a nested calculation.

```
RunningTotal =
CALCULATE(
   SUM(Sales[SalesAmount]),
   FILTER(
      Sales,
      Sales[Date] <= EARLIER(Sales[Date])
   )
)
```

18. CONTAINS

Checks if a table contains a specified value.

```
ProductExists = CONTAINS(Product, Product[ProductID], Sales[ProductID])
```

19. SWITCH

Evaluates an expression against a list of values and returns the corresponding result.

```
SalesCategory = SWITCH(
   TRUE(),
   Sales[SalesAmount] > 1000, "High",
   Sales[SalesAmount] > 500, "Medium",
   "Low"
)
```

20. ISBLANK

Checks if a value is blank.

```
CheckBlank = IF(ISBLANK(Sales[SalesAmount]), "No Sales", Sales[SalesAmount])
```

21. LOOKUPVALUE

Returns a value for a specified column from a table where a condition is met.

```
ProductName = LOOKUPVALUE(Product[ProductName], Product[ProductID],
Sales[ProductID])
```

22. VALUES

Returns a one-column table that contains the distinct values from a column.

DistinctYears = VALUES(Sales[Year])

23. ALLSELECTED

Removes context filters from columns and rows but keeps filters applied by slicers or visuals.

SelectedSales = CALCULATE(SUM(Sales[SalesAmount]),
ALLSELECTED(Sales[ProductID]))

24. RANKX

Ranks items in a table based on a specified expression.

SalesRank = RANKX(ALL(Sales[ProductID]), SUM(Sales[SalesAmount]), , DESC)

25. USERELATIONSHIP

Specifies an inactive relationship to be used in a calculation.

SalesByAlternateDate = CALCULATE(SUM(Sales[SalesAmount]),
USERELATIONSHIP(Sales[AlternateDate], Calendar[Date]))