[Power BI Calculated Columns and Measures] [cheatsheet]

Basic Arithmetic

- Addition: `Total = [Revenue] + [Costs]`
- Subtraction: `Profit = [Revenue] - [Costs]`
- Multiplication: `Total Cost = [Quantity] * [Unit Price]`
- Division: `Profit Margin = DIVIDE([Profit], [Revenue], 0)`
- Exponentiation: `Squared Value = [Value] ^ 2`

Aggregation Functions

- Sum: `Total Sales = SUM(Sales[Amount])`
- Average: `Avg Sales = AVERAGE(Sales[Amount])`
- Count: `Customer Count = COUNT(Customers[CustomerID])`
- Distinct Count: `Unique Products = DISTINCTCOUNT(Products[ProductID])`
- Min: `Lowest Price = MIN(Products[Price])`
- Max: `Highest Price = MAX(Products[Price])`

Text Functions

- Concatenate: `Full Name = Employees[FirstName] & " " & Employees[LastName]`
- Left: `First Letter = LEFT(Products[ProductName], 1)`
- Right: `Last 3 Chars = RIGHT(Orders[OrderID], 3)`
- Len: `Name Length = LEN(Customers[CustomerName])`
- Upper: `Upper Case Name = UPPER(Products[ProductName])`
- Lower: `Lower Case Name = LOWER(Products[ProductName])`
- Proper: `Proper Case Name = PROPER(Customers[CustomerName])`
- Trim: `Trimmed Name = TRIM(Products[ProductName])`
- Substitute: `Fixed Phone = SUBSTITUTE(Customers[Phone], "-", "")`

Date and Time Functions

- Year: `Order Year = YEAR(Orders[OrderDate])`
- Month: `Order Month = MONTH(Orders[OrderDate])`
- Day: `Order Day = DAY(Orders[OrderDate])`
- Quarter: `Order Quarter = QUARTER(Orders[OrderDate])`
- Week Number: `Order Week = WEEKNUM(Orders[OrderDate])`
- Week Day: `Order Weekday = WEEKDAY(Orders[OrderDate])`

By: Waleed Mousa

- Date Difference: Days Since Order = DATEDIFF(Orders[OrderDate], TODAY(), DAY)
- End of Month: Month End = EOMONTH(Orders[OrderDate], 0)
- Date Add: Due Date = DATEADD(Orders[OrderDate], 30, DAY)
- Is After: Is Recent = Orders[OrderDate] > DATE(2023, 1, 1)

Conditional Statements

- Simple IF: Status = IF(Products[Stock] > 0, "In Stock", "Out of Stock")
- Nested IF: Price Category = IF([Price] < 10, "Low", IF([Price] < 50, "Medium", "High"))
- SWITCH: Day Type = SWITCH(WEEKDAY(Orders[OrderDate]), 1, "Weekend", 7, "Weekend", "Weekday")
- AND: Is Valid = AND(Products[Price] > 0, Products[Stock] > 0)
- OR: Needs Attention = OR(Products[Price] = 0, Products[Stock] = 0)
- NOT: Is Not Valid = NOT([Is Valid])

Time Intelligence Functions

- Year-to-Date: YTD Sales = TOTALYTD(SUM(Sales[Amount]), 'Date'[Date])
- Quarter-to-Date: QTD Sales = TOTALQTD(SUM(Sales[Amount]), 'Date'[Date])
- Month-to-Date: MTD Sales = TOTALMTD(SUM(Sales[Amount]), 'Date'[Date])
- Previous Year: PY Sales = CALCULATE([Total Sales], SAMEPERIODLASTYEAR('Date'[Date]))
- Year-over-Year Growth: YoY Growth = DIVIDE([Total Sales] - [PY Sales], [PY Sales], 0)
- Moving Annual Total: MAT Sales = CALCULATE([Total Sales], DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -1, YEAR))
- Previous Month: PM Sales = CALCULATE([Total Sales], PREVIOUSMONTH('Date'[Date]))
- Month-over-Month Growth: MoM Growth = DIVIDE([Total Sales] - [PM Sales], [PM Sales], 0)
- Rolling 3-Month Average: 3M Avg Sales = AVERAGEX(DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -2, MONTH), [Total Sales])

Statistical Functions

- Median: Median Price = MEDIAN(Products[Price])
- Mode: Most Common Price = MODE(Products[Price])

By: Waleed Mousa

- Percentile: 90th Percentile Price = PERCENTILE.EXC(Products[Price], 0.9)
- Standard Deviation: Price StDev = STDEV.S(Products[Price])
- Variance: Price Variance = VAR.S(Products[Price])
- Correlation: Price-Sales Correlation = CALCULATE(CORRELATION(Products[Price], Sales[Amount]))

Ranking Functions

- Rank: Sales Rank = RANKX(ALL(Products), [Total Sales],, DESC)
- Dense Rank: Dense Sales Rank = RANKX(ALL(Products), [Total Sales],, DESC, Dense)
- Percent Rank: Sales Percentile = PERCENTRANK.EXC(ALL(Products[Total Sales]), [Total Sales])

Filter Functions

- All: Overall Avg Sales = AVERAGE(ALL(Sales[Amount]))
- AllExcept: Category Avg Sales = AVERAGE(ALLEXCEPT(Sales, Sales[Category]))
- Filter: High Value Sales = CALCULATE([Total Sales], FILTER(Sales, Sales[Amount] > 1000))
- TopN: Top 5 Products = CALCULATE([Total Sales], TOPN(5, ALL(Products), [Total Sales], DESC))

Iterator Functions

- SUMX: Total Revenue = SUMX(Sales, Sales[Quantity] * Sales[UnitPrice])
- AVERAGEX: Avg Order Value = AVERAGEX(Orders, RELATED(Sales[Amount]))
- MAXX: Highest Sales = MAXX(Sales, Sales[Amount])
- MINX: Lowest Sales = MINX(Sales, Sales[Amount])

Parent-Child Functions

- PATH: Employee Path = PATH(Employees[EmployeeID], Employees[ManagerID])
- PATHITEM: Manager Level 2 = PATHITEM([Employee Path], 2)
- PATHITEMREVERSE: Direct Manager = PATHITEMREVERSE([Employee Path], 1)
- PATHCONTAINS: Has Manager = PATHCONTAINS([Employee Path], Employees[ManagerID])

Error Handling

By: Waleed Mousa

- IFERROR: Safe Division = IFERROR(DIVIDE([Revenue], [Costs]), 0)
- ISBLANK: Has Sales = NOT(ISBLANK([Total Sales]))
- ISEMPTY: Has Orders = NOT(ISEMPTY(Orders))
- HASONEVALUE: Single Product Selected = HASONEVALUE(Products[ProductName])

Text Analysis

- FIND: Contains A = IF(FIND("a", LOWER(Products[ProductName]), 1, -1) > 0, "Yes", "No")
- SEARCH: Position of Space = SEARCH(" ", Customers[CustomerName], 1, -1)
- UNICHAR: Star Rating = REPT(UNICHAR(9733), Products[Rating])
- UNICODE: First Char Code = UNICODE(LEFT(Products[ProductName], 1))

Advanced Calculations

- Running Total: Running Total Sales = CALCULATE(SUM(Sales[Amount]), FILTER(ALL('Date'), 'Date'[Date] <= MAX('Date'[Date])))
- Cumulative Percentage: Cumulative % = DIVIDE(CALCULATE([Total Sales], FILTER(ALL('Date'), 'Date'[Date] <= MAX('Date'[Date]))), [Overall Total Sales])
- Pareto Analysis: Pareto = DIVIDE(CALCULATE([Total Sales], FILTER(ALL(Products), Products[Total Sales] >= EARLIER(Products[Total Sales]))), [Overall Total Sales])
- Market Share: Market Share % = DIVIDE([Total Sales], CALCULATE([Total Sales], ALL(Products)))
- Contribution to Parent: Category Contribution % = DIVIDE([Total Sales], CALCULATE([Total Sales], ALL(Products[Category])))
- Year-over-Year Comparison: YoY Comparison = SWITCH(TRUE(), [YoY Growth] > 0.1, "High Growth", [YoY Growth] > 0, "Growth", [YoY Growth] = 0, "Flat", "Decline")

Window Functions

- Moving Average: 3-Day Moving Avg = AVERAGEX(DATESINPERIOD('Date'[Date], LASTDATE('Date'[Date]), -2, DAY), [Daily Sales])
- Cumulative Sum: Cumulative Sales = CALCULATE(SUM(Sales[Amount]), FILTER(ALL('Date'), 'Date'[Date] <= MAX('Date'[Date])))
- Percent of Running Total: % of Running Total = DIVIDE([Total Sales], [Cumulative Sales])
- Difference from Previous: Sales Diff = [Total Sales] - CALCULATE([Total Sales], PREVIOUSDAY('Date'[Date]))

By: Waleed Mousa

- Rolling Year Comparison: Rolling Year Diff = [MAT Sales] - CALCULATE([MAT Sales], DATEADD('Date'[Date], -1, YEAR))

Advanced Time Intelligence

- Custom Year-to-Date: Custom YTD = CALCULATE([Total Sales], DATESYTD('Date'[Date], "6-30"))
- Fiscal Year Calculations: Fiscal YTD = CALCULATE([Total Sales], DATESYTD('Date'[Date], "7-1"))
- Semi-Annual Periods: Half Year = IF(MONTH('Date'[Date]) <= 6, "H1", "H2")
- Week-over-Week Comparison: WoW Change = [Total Sales] - CALCULATE([Total Sales], DATEADD('Date'[Date], -7, DAY))
- Last N Periods: Last 3 Months Sales = CALCULATE([Total Sales], DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -2, MONTH))

Data Categorization

- Sales Bracket: Sales Bracket = SWITCH(TRUE(), [Total Sales] >= 1000000, "Large", [Total Sales] >= 100000, "Medium", "Small")
- Age Group: Age Group = SWITCH(TRUE(), Customers[Age] >= 60, "Senior", Customers[Age] >= 40, "Middle-aged", Customers[Age] >= 20, "Young Adult", "Youth")
- Product Performance: Performance Category = SWITCH(TRUE(), [YoY Growth] > 0.2, "High Performer", [YoY Growth] > 0, "Growing", [YoY Growth] > -0.1, "Stable", "Declining")

Data Quality Checks

- Completeness Check: Is Complete = AND(NOT(ISBLANK([Sales])), NOT(ISBLANK([Costs])))
- Range Check: Is Valid Price = AND([Price] >= 0, [Price] <= 1000)
- Format Check: Is Valid Email = SEARCH("@", Customers[Email], 1, -1) > 0
- Consistency Check: Is Consistent = [Sales] >= [Costs]

KPI Calculations

- Gross Profit Margin: GPM = DIVIDE([Gross Profit], [Total Sales], 0)
- Customer Lifetime Value: CLV = DIVIDE([Total Sales], DISTINCTCOUNT(Customers[CustomerID]))
- Customer Acquisition Cost: CAC = DIVIDE([Marketing Spend], [New Customers])

By: Waleed Mousa

- Return on Investment: `ROI = DIVIDE([Net Profit], [Total Investment], 0)`
- Debt-to-Equity Ratio: `D/E Ratio = DIVIDE([Total Liabilities], [Total Equity], 0)`

Advanced Time Intelligence

- Sliding Window: `6M Sliding Window = CALCULATE([Total Sales], DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -5, MONTH))`
- Parallel Period: `Parallel Period Sales = CALCULATE([Total Sales], PARALLELPERIOD('Date'[Date], -1, YEAR))`
- Custom Period Comparison: `Q4 vs Q2 = DIVIDE(CALCULATE([Total Sales], 'Date'[QuarterNo] = 4), CALCULATE([Total Sales], 'Date'[QuarterNo] = 2)) - 1`
- Rolling Forecast: `12M Forecast = CALCULATE([Total Sales], DATEADD('Date'[Date], 1, YEAR)) * (1 + [YoY Growth])`

Advanced Statistical Measures

- Z-Score: `Sales Z-Score = DIVIDE([Total Sales] - AVERAGE([Total Sales]), STDEV.S([Total Sales]))`
- Confidence Interval: `CI Upper = AVERAGE([Sales]) + 1.96 * STDEV.S([Sales]) / SQRT(COUNT([Sales]))`
- Moving Correlation: `12M Rolling Correlation = CORRELATIONX(DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -11, MONTH), [Sales], [Marketing Spend])`
- Exponential Moving Average: `EMA = CALCULATE(0.2 * [Sales] + 0.8 * CALCULATE([EMA], PREVIOUSDAY('Date'[Date])))`

Advanced Ranking and Segmentation

- Percentile Bucketing: `Sales Percentile Bucket = SWITCH(TRUE(), [Sales Percentile] <= 0.2, "Bottom 20%", [Sales Percentile] <= 0.4, "20-40%", [Sales Percentile] <= 0.6, "40-60%", [Sales Percentile] <= 0.8, "60-80%", "Top 20%")`
- Relative Ranking: `Relative Sales Rank = RANKX(ALLSELECTED(Products), [Total Sales],, DESC)`
- ABC Analysis: `ABC Category = SWITCH(TRUE(), [Cumulative Sales %] <= 0.7, "A", [Cumulative Sales %] <= 0.9, "B", "C")`

Inter-table Calculations

By: Waleed Mousa

- Average Basket Size: `Avg Basket Size = DIVIDE([Total Sales], DIVIDE(COUNTROWS(Sales), DISTINCTCOUNT(Sales[OrderID])))`
- Product Penetration: `Product Penetration % = DIVIDE(CALCULATE(DISTINCTCOUNT(Sales[CustomerID]), FILTER(Sales, Sales[ProductID] = EARLIER(Products[ProductID]))), DISTINCTCOUNT(Sales[CustomerID]))`
- Customer Segment Performance: `Segment Performance = DIVIDE(CALCULATE([Total Sales], RELATEDTABLE(CustomerSegment)), CALCULATE([Total Sales], ALL(CustomerSegment)))`

Advanced Conditional Formatting

- Dynamic Threshold: `Performance Indicator = SWITCH(TRUE(), [Sales] > [Target] * 1.1, 3, [Sales] > [Target], 2, [Sales] > [Target] * 0.9, 1, 0)`
- Gradient Scale: `Temperature = 1 - ([Value] - [Min Value]) / ([Max Value] - [Min Value])`

Complex Business Rules

- Tiered Discounting: `Discount % = SWITCH(TRUE(), [Order Quantity] >= 100, 0.15, [Order Quantity] >= 50, 0.1, [Order Quantity] >= 20, 0.05, 0)`
- Dynamic Pricing: `Adjusted Price = [Base Price] * (1 - [Discount %]) * IIF([Stock] < 10, 1.1, 1)`
- Loyalty Points: `Points Earned = ROUNDDOWN([Total Sales] * IIF(Customers[Tier] = "Gold", 0.05, IIF(Customers[Tier] = "Silver", 0.03, 0.01)), 0)`

Text Analytics

- Sentiment Score: `Sentiment = SWITCH(TRUE(), CONTAINSSTRING([Review], "excellent") || CONTAINSSTRING([Review], "great"), 2, CONTAINSSTRING([Review], "good") || CONTAINSSTRING([Review], "nice"), 1, CONTAINSSTRING([Review], "poor") || CONTAINSSTRING([Review], "bad"), -1, 0)`
- Word Count: `Word Count = LEN([Text]) - LEN(SUBSTITUTE([Text], " ", "")) + 1`

Financial Calculations

By: Waleed Mousa

- Compound Annual Growth Rate: `CAGR = POWER(DIVIDE(LASTNONBLANK([Value], [Year]), FIRSTNONBLANK([Value], [Year])), 1 / ([Last Year] - [First Year] + 1)) - 1`
- Days Sales Outstanding: `DSO = DIVIDE([Accounts Receivable], [Total Sales]) * 365`
- Working Capital: `Working Capital = [Current Assets] - [Current Liabilities]`
- Debt Service Coverage Ratio: `DSCR = DIVIDE([EBITDA], [Total Debt Service])`

Forecasting and Predictive Measures

- Simple Linear Regression: `Sales Forecast = AVERAGEX(Sales, Sales[Amount]) + (MAX('Date'[DateKey]) - AVERAGE('Date'[DateKey])) * DIVIDE(SUMX(Sales, (Sales[Amount] - AVERAGE(Sales[Amount])) * ('Date'[DateKey] - AVERAGE('Date'[DateKey]))), SUMX(Sales, POWER('Date'[DateKey] - AVERAGE('Date'[DateKey]), 2)))`
- Seasonal Adjustment: `Seasonally Adjusted Sales = DIVIDE([Total Sales], AVERAGE(CALCULATETABLE(VALUES('Date'[MonthNo]))))`
- Holt-Winters Forecasting: `HW Forecast = [Level] + [Trend] * [Period] + [Seasonal Factor]`

Advanced Error Handling and Data Quality

- Multi-condition Error Check: `Data Quality Flag = IF(OR(ISBLANK([Sales]), [Sales] < 0, [Sales] > 1000000), "Check Required", "OK")`
- Fuzzy Matching: `Potential Duplicate = IF(MINX(FILTER(Customers, Customers[CustomerID] <> EARLIER(Customers[CustomerID])), SQRTSUMX(Customers, POWER(UNICODE(MID(Customers[Name], [Char], 1)) - UNICODE(MID(EARLIER(Customers[Name]), [Char], 1)), 2))) < 5, "Yes", "No")`

Dynamic Measures

- Measure Switch: `Selected Measure = SWITCH(SELECTEDVALUE(MeasureSelector[Measure]), "Sales", [Total Sales], "Profit", [Total Profit], "Units", [Total Units], BLANK())`
- Dynamic Time Calculation: `Dynamic Time Calc = CALCULATE([Total Sales], DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -[Selected Periods], [Selected Time Grain]))`

Geographic Calculations

By: Waleed Mousa

- Distance Calculation: `Distance = ACOS(SIN(RADIANS(Stores[Latitude])) * SIN(RADIANS(Customers[Latitude])) + COS(RADIANS(Stores[Latitude])) * COS(RADIANS(Customers[Latitude])) * COS(RADIANS(Stores[Longitude] - Customers[Longitude]))) * 6371`
- Geohash: `Geohash = CONCATENATE(REPT(UNICHAR(97 + ROUNDDOWN(([Latitude] + 90) / 180 * 26, 0)), 3), REPT(UNICHAR(97 + ROUNDDOWN(([Longitude] + 180) / 360 * 26, 0)), 3))`

Anomaly Detection

- Outlier Flag: `Is Outlier = IF(ABS(([Value] - AVERAGE([Value])) / STDEV.S([Value])) > 3, "Yes", "No")`
- Anomaly Score: `Anomaly Score = ABS(([Value] - AVERAGE([Value])) / STDEV.S([Value]))`

Advanced Set Analysis

- Market Basket Analysis: `Co-occurrence = DIVIDE(CALCULATE(DISTINCTCOUNT(Sales[OrderID]), FILTER(ALL(Products), Products[ProductID] <> EARLIER(Products[ProductID]))), DISTINCTCOUNT(Sales[OrderID]))`
- Customer Overlap: `Shared Customers = DIVIDE(COUNTROWS(INTERSECT(CALCULATETABLE(VALUES(Customers[CustomerID]), Products[Category] = "A"), CALCULATETABLE(VALUES(Customers[CustomerID]), Products[Category] = "B"))), COUNTROWS(UNION(CALCULATETABLE(VALUES(Customers[CustomerID]), Products[Category] = "A"), CALCULATETABLE(VALUES(Customers[CustomerID]), Products[Category] = "B"))))`

By: Waleed Mousa