DAX Cheat Sheet

Draphony

1. Basic Aggregations

```
// Returns the sum of all values in a column.
SUM(column_name)

// Returns the average of values in a column.
AVERAGE(column_name)

// Returns the number of non-empty values in a column.
COUNT(column_name)

// Returns the maximum value in a column.
MAX(column_name)

// Returns the minimum value in a column.
MIN(column_name)

// Safely divides two numbers, returns alternate result if denominator is 0.
DIVIDE(numerator, denominator, alternate_result)
```

2. Iterator Functions

```
// Calculates the sum of an expression evaluated for each row in a table.
SUMX(table, expression)

// Calculates the average of an expression evaluated for each row in a table.
AVERAGEX(table, expression)

// Returns the maximum value of an expression evaluated for each row in a table.
MAXX(table, expression)

// Returns the minimum value of an expression evaluated for each row in a table.
MINX(table, expression)

// Returns the count of non-blank results from evaluating an expression for each row.
COUNTX(table, expression)
```

3. Logical Functions

```
// Returns one value if the condition is true, otherwise returns another value.
IF(condition, true_result, false_result)

// Evaluates an expression and returns different results based on matching values.
SWITCH(expression, value1, result1, value2, result2, ...)

// Returns TRUE if both conditions are true.
AND(condition1, condition2)

// Returns TRUE if either condition is true.
OR(condition1, condition2)

// Reverses the result of a logical condition (TRUE becomes FALSE, and vice versa).
NOT(logic)

// Returns the value if there is no error; otherwise returns the specified alternate value.
IFERROR(value, value_if_error)
```

4. Time Intelligence

```
// Shifts dates by the specified number of intervals (e.g., days, months).
DATEADD(dates, number_of_intervals, interval)

// Returns dates between two given dates.
DATESBETWEEN(dates, date1, date2)

// Calculates the year-to-date total for a given measure.
TOTALYTD(SUM(column), date_column)

// Returns the dates from the beginning of the year to the current date.
DATESYTD(date_column)

// Returns the set of dates in the same period of the previous year.
SAMEPERIODLASTYEAR(date_column)
```

5. Date Functions

```
// Returns the current date.
TODAY()

// Returns the current date and time.
NOW()

// Returns the year from a date.
YEAR(date)

// Returns the month from a date.
MONTH(date)

// Returns the day of the month from a date.
DAY(date)

// Returns a table with a single column of dates between start_date and end_date.
CALENDAR(start_date, end_date)

// Creates a date value from the specified year, month, and day.
DATE(year, month, day)

// Returns the difference between two dates in the specified interval (e.g., days, months)
DATEDIFF(date1, date2, interval)

// Returns the day of the week for a date.
WEEKDAY(date, return_type)

// Returns the week number for a date.
WEEKNUM(date, return_type)
```

6. Mathematical Functions

```
// Rounds a number to the specified number of digits.
ROUND(number, num_digits)

// Safely divides two numbers, returns alternate result if division by zero occurs.
DIVIDE(numerator, denominator, alternateResult)

// Returns the remainder after dividing a number by a divisor.
MOD(number, divisor)

// Returns the rank of a value in a table based on an expression.
RANKX(table, expression, value, [order], [ties])

// Returns the kth percentile excluding the value k.
PERCENTILE.EXC(column, k)

// Returns the kth percentile including the value k.
PERCENTILE.INC(column, k)
```

7. Filtering and Context

```
// Returns a table filtered by a condition.
FILTER(table, condition)

// Changes the context of the calculation by applying filters.
CALCULATE(expression, filter1, filter2, ...)

// Ignores all filters applied to a table or column.
ALL(table)

// Returns all rows, excluding any blank rows.
ALLNOBLANKROW(table, column_name1, column_name2, ...)

// Ignores all filters except for the specified columns.
ALLEXCEPT(table, column1, column2, ...)

// Removes all filters from the specified table or columns.
REMOVEFILTERS(table, column_name1, column_name2, ...)
```

8. Text Functions

```
// Joins two text strings into one.
CONCATENATE(text1, text2)

// Returns the starting position of a substring within a text string.
FIND(find_text, within_text, start_num, not_found_value)

// Formats a value according to the specified format string.
FORMAT(value, format)

// Returns the length of a text string.
LEN(text)

// Converts a text string to lowercase.
LOWER(text)

// Converts a text string to uppercase.
UPPER(text)

// Removes extra spaces from text.
TRIM(text)

// Returns the leftmost characters from a text string.
LEFT(text, num_chars)

// Returns the rightmost characters from a text string.
RIGHT(text, num_chars)

// Searches for a substring within a text string.
SEARCH(find_text, within_text, start_num, not_found_value)

// Replaces occurrences of old_text with new_text.
SUBSTITUTE(text, old_text, new_text, instance_num)

// Replaces part of a text string, based on position and length.
REPLACE(old_text, start_position, num_chars, new_text)
```

9. Table Functions

```
// Groups a table by specified columns and returns aggregated results.
SUMMARIZE(table, groupby_column1, "name1", expression1, ...)

// Returns a table that contains unique values from a column or set of columns.
DISTINCT(table)

// Adds calculated columns to a table.
ADDCOLUMNS(table, "new_column_name", expression)

// Returns a table with selected columns.
SELECTCOLUMNS(table, name1, expression1, name2, expression2)

// Returns the Cartesian product of two tables.
CROSSJOIN(table1, table2)

// Groups a table by specified columns and applies expressions to each group.
GROUPBY(table, groupby_column_name, column_name, expression)

// Returns a table that contains the common rows from two tables.
INTERSECT(left_table, right_table)

// Returns the natural inner join of two tables.
NATURALINNERJOIN(left_table, right_table)

// Returns the natural left outer join of two tables.
NATURALLEFTOUTERJOIN(left_table, right_table)

// Returns the rows from the first table that are not in the second table.
EXCEPT(left_table, right_table)

// Returns the union of two tables (rows from both tables).
UNION(table1, table2)
```