KQL

Common Operators	
print Time=now(), Message=“hello”	Outputs single row with one or more scalar expressions
… | **count**	Counts records in input table (e.g. T)
… | **take** 10	Get few records - convenient to start get familiar with the data, No actual order ensured
… | **where** col1 > ago(1) and UserId = ‘abdcdef’	Filtering on a specific field
… | **search** “abc”	Will search all columns in the Perf table for the value (not case sensitive by default)
… | **project** Col1, Col2, …	Chooses some columns
… | **project-away** Col1, Col2, …	Removes some columns
…| **extend** NewCol1=Col1+Col2	Extend creates a calculated column and adds to the result set
… | **top** 10 by count_ desc/asc	returns the first N rows of the dataset when the dataset is sorted by
… | **sort** by Col1 desc	Sort the rows of the input table into order by one or more columns
… | **summarize** count(), dcount(Id) by Col1, Col2	Groups the rows according to the aggregations columns (by)
… | **distinct** Col1, Col2	Produces a table with the distinct combination of the provided columns of the input table
… | **join** (…) on Key1, Key2	Merges the rows of two tables to form a new table by matching values of the specified column(s) from each table. Kusto supports a full range of join types: fullouter, inner, innerunique, leftanti, leftantisemi, leftouter, leftsemi, rightanti, rightantisemi, rightouter, rightsemi
FactTable | **lookup** kind=leftouter DimTable on col1, col2	extends the columns of a fact table with values looked-up in a dimension table
… | **union** Tab1, Tab2	Takes two or more tables and returns the rows of all of them
… | **render** timechart	Renders results as a graphical output
… | **mv-expand** Col1,Col2 …	Turn dynamic arrays to rows (multi-value expansion)
… | **parse** Col1 with <pattern>…	Take care of unstructured data
… | extend C1 = **range**(1, 8, 3)	Generates a dynamic array holding a series of equally-spaced values
… | **make-series** sum(col1) default=0, avg(col1) default=0 on timestamp from datetime(2016-01-01) to datetime(2016-01-10) step 1d by col2	Create series of specified aggregated values along specified axis
let name = "Free"; … | where CounterName == name	binds a name to an expression
range x from 1 to 10 step 1 | **as** T1	Binds a name to the operator's input tabular expression
… | **invoke** foo(param1, param2)	invokes lambda that receives the source of invoke as tabular parameter argument
…| **evaluate** pluginName (Arg1, arg2)	operator is a tabular operator that provides the ability to invoke query language extensions known as **plugins**
Common Functions	
… | where Timestamp > **ago**(1h)	ago returns a time in the past, using the current time as a starting point d – days, h – hours, m – minutes, s – seconds, ms – milliseconds, tick – nanosecond, microsecond – microseconds
… | extend col2 = **format_datetime**(col1, "y-M-d"),	format_datetime allows you to return specific date formats format_datetime(TimeGenerated, "MM/dd/yyyy HH:mm:ss.ffff") // f/F can also be used for subSs.
… | extend col1 = **case**(col1 < 10, "Critical", "You're OK!")	Evaluates a list of predicates, and returns the first result expression whose predicate is satisfied, or the final else expression