KQL CHEATSHEET

KUSTO QUERY ESSENTIALS TO START WITH

SOURCES:
MICROSOFT LEARN
MUSTLEARNKQL
KUSTOKING

HYPERFOX

VERSION 1.1

_DATA INGESTION

```
.execute database script <|
.create-merge table T1 (Timestamp:datetime, F1:string)
.ingest async into table T1 ('https://url/file.csv.gz') with
(ignoreFirstRecord=true)
```

_FILTERS

```
| where IP_Add startswith "10.168.8" // see OPERATORS
| where * has "Kusto" and ActivityId == SubActivityId
| where isnotempty(Field1) or isnotnull(Field2)
| project-away Field1, Field2, Field3 //Remove extra columns
| distinct Field1, Field2, Field5    //Deduplicate values
| lookup Table1 on Field2  //lookup from another table1
```

_TIME

```
let min_t = toscalar(Table | summarize min(Timestamp));
let max_t = toscalar(Table | summarize max(Timestamp));
let int_t = 1h;
Table | range Field1 from min_t to max_t step int_t
| where TimeGenerated > ago(30m)  // last 30 min sample
| extend TimeGenerated1 = todatetime(TimeGenerated)
| where TimeGenerated between (min_t .. max_t)
| extend myTime = now() - totimespan("1d")
find Session.Id=="c8894" and ingestion_time()> ago(24h)
```

_STATISTICS

```
| summarize num=count(), unique=dcount(Field1) by
bin(Timestamp,1h), Field2, Field3
| summarize dcount(Field1), make_set(Field1) by Field2
| order by Field1 asc , Field2 desc   //sort can be used
| top 10 by Field1 asc    //returns top data set
```

INITIAL SEARCH

```
Table1 | sample 50   //sampling
Table1 | getschema   //data types
Table1 | count   //total rows
search "string" in ("Tbl1*","Tbl2")
```

OPERATORS

==	in	!startswith	has	has_any
=~	!in~	startswith_cs	!has	has_all
!=	!in	endswith	has_cs	
!~	!in~	!endswith	has_suffix	
	matches_regex		has_prefix	

DATA TYPES

bool | datetime | dynamic | string | int | long | real | timespan | decimal

JOIN

```
LEFT_Table_ Query
| join kind=inner
   ( RIGHT_Table_ Query )
on CommonField
```

leftanti | fullouter | leftouter | rightouter

VISUALIZATION

```
| render timechart
    with ( title= , ysplit= , kind=
```

linechart | areachart | scatterchart | barchart | columnchart | anomalychart | anomalycolumns= | legend=

_TRANSFORMATION

STRING

```
| extend IP_Net = replace_regex(via_ip,@'(\d\.168\.)(\d).+)',
@'\1\2+')
| extend StartDir =  substring(ProcessName,0,
string_size(ProcessName) - string_size(Process))
| extend sum_sign = iif(sin + cos > 0, "sum_pos", "sum_neg")
| extend bucket = case(Size <= 3, "S", Size <= 10, "M", "L")
| evaluate pivot(Field1, sum(Count))   //Pivot or Transpose
| extend Field1= strcat(Field2, " ", "world")   //concatenate
| project-rename exception = Date_Exception
| parse EventText with * "resourceName=" resourceName ",
totalSlices=" totalSlices:long *
| parse kind = regex EventText with "(.*?[a-zA-Z]*=)"
resourceName @", totalSlices=\s*\d+\s*.*?sliceNumber="
sliceNumber:long ".*?(previous)?lockTime=" lockTime
| extend AdditionalExtensions = extract_all(@"
(?P<key>\w+)=(?P<value>[a-zA-Z0-9-_:/@. ]+)",
dynamic(["key","value"]), AdditionalExtensions)
| extend extension =
tostring(parse_json(ShareLocalPathParsed).Extension)
| mv-expand newField1=Field1 to typeof(string)  //expand a list
```

NUMERICAL

```
| as hint.materialized=true T
| extend Total = toscalar(T | summarize sum(Count))
```

sumif | min | max | minif | maxif | avg

```
| extend perc50 = toscalar(T | summarize percentile(Count, 50))
| project Cnt, Total, Percentage = round(Cnt*100.0 / Total, 1)
| summarize arg_max(Field_Max, * ) by Field1
```

IPV4 HANDLING

```
| extend result = ipv4_is_match(ip1_string, ip2_string, prefix)
| evaluate ipv4_lookup(IP_Data, ip, network)
| extend result = ipv4_is_in_range(ip_address, ip_prefix)
```

KQL SAMPLE QUERIES

```
let suspiciousAccounts = datatable(account: string) [
  @"administrator",
  @"NT AUTHORITY\SYSTEM" ];
SecurityEvent | where Account in (suspiciousAccounts)
```

```
let LowActivityAccounts =
    SecurityEvent
    | summarize cnt = count() by Account
    | where cnt < 1000;
LowActivityAccounts | where Account contains "SQL"
```

```
let today = endofday(now());
let yesterday = startofday(now(-1d));
Table
| where host == "DC02.sample.net" and EventTime between
(yesterday .. today) and Name == "Controller"
| count
```

```
let geoData = materialize
(externaldata(network:string,geoname_id:string,continent_code:string,continent_name:string,
country_iso_code:string,country_name:string,is_anonymous_proxy:string,is_satellite_provider:string)
[@"https://raw.githubusercontent.com/datasets/geoip2-ipv4/master/data/geoip2-ipv4.csv"] with
(ignoreFirstRecord=true, format="csv");let ip = AzureDiagnostics
| summarize count() by clientIP_scip
| evaluate ipv4_lookup(geoData, clientIP_s, network)
| project clientIP_s, network, count_, country_iso_code, country_name
```

```
let min_t = toscalar(Table | summarize min(Timestamp));
let max_t = toscalar(Table | summarize max(Timestamp));
let dt = 1h;
Table
| make-series num=count() default=0 on Timestamp in range(min_t, max_t, dt) by
Field1
| extend (baseline, seasonal, trend, residual) = series_decompose(num, -1, 'linefit')
| render timechart with(title="", ysplit=panels)
```