KQL

Kusto Query Language

Complete Cheat Sheet

Beginner → Intermediate → Advanced → Master

Azure Data Explorer • Log Analytics • Microsoft Sentinel

Covers: Filtering • Aggregations • Joins • Time Series • Threat Detection • Analytics Rules

● BEGINNER

1. KQL FUNDAMENTALS

The Pipeline Mental Model

KQL works like a pipeline. Data flows left-to-right through **|** (pipe) operators. Each step filters or transforms the data.

```
// Table → filter → transform → output
SignIns
| where TimeGenerated > ago(1h)
| project UserPrincipalName, IPAddress, ResultType
| sort by TimeGenerated desc
```

■ *Always put your TimeGenerated filter FIRST — it limits data scanned and makes queries much faster.*

Time Filters — ago()

```
// Relative time — most common in log analysis
| where TimeGenerated > ago(1h) // last 1 hour
| where TimeGenerated > ago(24h) // last 24 hours
| where TimeGenerated > ago(7d) // last 7 days
| where TimeGenerated > ago(30m) // last 30 minutes
// Absolute time range
| where TimeGenerated between (datetime(2024-01-01) .. datetime(2024-01-31))
```

Filtering with where

```
// Equality
| where ResultType == '0'
// Not equal
| where ResultType != '0'
// Multiple conditions (AND)
| where ResultType != '0' and Location == 'Russia'
// OR condition
| where Location == 'Russia' or Location == 'Nigeria'
// Match a list of values
| where Location in ('Russia', 'Nigeria', 'China', 'Iran')
// Exclude values
| where Location !in ('United States', 'United Kingdom')
// Contains text (slower)
| where UserPrincipalName contains 'admin'
// Has whole word (faster — prefer this)
| where UserPrincipalName has 'admin'
// Starts with / ends with
| where UserPrincipalName startswith 'svc'
// Multiple terms — has_any / has_all
| where AppDisplayName has_any ('Teams', 'SharePoint', 'Portal')
```

Selecting Columns — project

```
// Pick only the columns you need
| project TimeGenerated, UserPrincipalName, IPAddress, ResultType
// Rename columns while projecting
| project Time=TimeGenerated, User=UserPrincipalName, IP=IPAddress
// Keep all columns EXCEPT some
| project-away AttackPattern, IsInteractive
```

```
// Add a new calculated column
| extend IsFailed = iff(ResultType != '0', true, false)
```

Sorting and Limiting

```
| sort by TimeGenerated desc // newest first
| sort by TimeGenerated asc // oldest first
| top 10 by TimeGenerated desc // top 10 most recent
| take 50 // any 50 rows (no order)
```

Essential Operators Quick Reference

Operator	Description	Example
`where`	Filter rows	| where EventID == 4625
`project`	Select columns	| project User, IP
`extend`	Add new column	| extend IsFail = ResultType != '0'
`sort / order by`	Sort results	| sort by count_ desc
`top`	Top N rows	| top 10 by count_
`take / limit`	Return N rows	| take 100
`count`	Count all rows	| count
`distinct`	Unique values	| distinct UserPrincipalName
`summarize`	Aggregate / group by	| summarize count() by User
`render`	Visualise as chart	| render timechart

● INTERMEDI
ATE

2. AGGREGATIONS & SUMMARIZE

summarize — The Most Important KQL Operator

```
// Basic count
SignIns | summarize count()
// Count per group (like SQL GROUP BY)
SignIns | summarize EventCount = count() by UserPrincipalName
// Multiple aggregations at once
SignIns
| summarize
TotalLogins = count(),
FailedLogins = countif(ResultType != '0'),
SuccessLogins = countif(ResultType == '0'),
UniqueIPs = dcount(IPAddress)
by UserPrincipalName
| sort by TotalLogins desc
```

Aggregation Functions

Function	Description	Example
`count()`	Total row count	summarize count()
`countif(cond)`	Count rows matching condition	countif(ResultType!='0')
`dcount(col)`	Distinct / unique count	dcount(IPAddress)
`sum(col)`	Sum of a column	sum(Duration)
`avg(col)`	Average value	avg(ResponseTime)
`min(col)`	Minimum value	min(TimeGenerated)
`max(col)`	Maximum value	max(TimeGenerated)
`make_set(col)`	List of unique values	make_set(Location)
`make_list(col)`	List of all values	make_list(IPAddress)
`stdev(col)`	Standard deviation	stdev(Duration)
`percentile(col,n)`	Nth percentile	percentile(Latency,95)

Time Bucketing with bin()

bin() groups timestamps into fixed time windows — essential for trend analysis.

```
// Events per hour
SignIns
| summarize Events = count() by bin(TimeGenerated, 1h)
| render timechart
// Failed vs successful per day
SignIns
| summarize
Failed = countif(ResultType != '0'),
Success = countif(ResultType == '0')
```

```
by bin(TimeGenerated, 1d)
| render timechart
```

let — Variables and Reusable Logic

```
// Store values in variables
let timeframe = 24h;
let threshold = 10;
let badCountries = dynamic(['Russia','Nigeria','China','Iran','Belarus']);

SignIns
| where TimeGenerated > ago(timeframe)
| where Location in (badCountries)
| summarize Attempts = count() by UserPrincipalName, IPAddress
| where Attempts > threshold
| sort by Attempts desc
```

■ *Use let to define thresholds and time windows at the top of your query — makes tuning analytics rules much easier.*

String Functions

Function	Description	Example
`tostring(x)`	Convert to string	tostring(EventID)
`toint(x)`	Convert to integer	toint(ResultType)
`tolower(x)`	Lowercase	tolower(UserPrincipalName)
`toupper(x)`	Uppercase	toupper(Location)
`strlen(x)`	String length	strlen(CommandLine)
`substring(x,i,n)`	Extract substring	substring(UPN, 0, 5)
`split(x,delim)`	Split string to array	split(UPN, '@')
`strcat(a,b)`	Concatenate strings	strcat(User, ' - ', IP)
`replace_string()`	Find and replace	replace_string(col,'old','new')
`extract(regex,1,x)`	Extract with regex	extract(@'(\d+\.\d+)', 1, IP)

● ADVANCED

3. JOINS, UNIONS & MULTI-TABLE QUERIES

join — Combining Two Tables

```
// inner join — only rows that match in BOTH tables
SignIns
| where ResultType == '0' // successful logins
| join kind=inner (
SignIns
| where ResultType != '0' // failed logins
| summarize FailCount = count() by UserPrincipalName
| where FailCount > 5
) on UserPrincipalName
| project TimeGenerated, UserPrincipalName, IPAddress, FailCount
// Result: accounts that failed 5+ times then succeeded = compromise signal
```

Join Types Quick Reference

Join Kind	Returns	Use Case
`inner`	Rows matching in both tables	Find overlap between two events
`leftouter`	All left rows + matching right rows	Enrich alerts with user details
`rightouter`	All right rows + matching left rows	Find unmatched alerts
`fullouter`	All rows from both tables	Complete comparison
`leftanti`	Left rows NOT in right table	Find logins with no prior fails
`rightanti`	Right rows NOT in left table	Find events with no resolution
`leftsemi`	Left rows that have match in right	Filter by existence

union — Combine Multiple Tables

```
// Stack two tables together (like SQL UNION ALL)
union SignIns, SecurityEvent
| where TimeGenerated > ago(24h)
| summarize count() by Type, bin(TimeGenerated, 1h)
| render timechart
// Union with wildcard — all tables starting with 'Security'
union Security*
| where TimeGenerated > ago(1h)
| summarize count() by Type
```

mv-expand — Expand Arrays into Rows

```
// make_set() creates arrays — mv-expand unpacks them
SignIns
| summarize Countries = make_set(Location) by UserPrincipalName
| mv-expand Countries
| project UserPrincipalName, Country = tostring(Countries)
```

parse — Extract Values from Unstructured Text

```
// Extract fields from free-text log messages
SignIns
| parse ResultDescription with 'Error: ' ErrorCode ' - ' ErrorMsg
| project UserPrincipalName, ErrorCode, ErrorMsg
// Using regex extract
| extend Domain = extract(@'@(.+)$', 1, UserPrincipalName)
```

iff & case — Conditional Columns

```
// iff — simple if/else
| extend Status = iff(ResultType == '0', 'Success', 'Failed')
// case — multiple conditions (like switch/case)
| extend RiskLabel = case(
RiskLevelDuringSignIn == 'high', 'CRITICAL',
RiskLevelDuringSignIn == 'medium', 'WARNING',
RiskLevelDuringSignIn == 'low', 'INFO',
'NORMAL')
```

● MASTER

4. SENTINEL THREAT DETECTION QUERIES

Brute Force Detection

```
let timeframe = 1h;
let threshold = 10;
SignIns
| where TimeGenerated > ago(timeframe)
| where ResultType != '0'
| summarize FailedAttempts = count() by UserPrincipalName, IPAddress, Location
| where FailedAttempts > threshold
| sort by FailedAttempts desc
```

Password Spray Detection

```
// One IP targeting many accounts with few attempts each
SignIns
| where TimeGenerated > ago(1h)
| where ResultType != '0'
| summarize
AccountsTargeted = dcount(UserPrincipalName),
TotalAttempts = count()
by IPAddress, Location
| where AccountsTargeted > 5 and TotalAttempts < 100
| sort by AccountsTargeted desc
```

MFA Fatigue Detection

```
// Attacker spams MFA requests hoping user approves
SignIns
| where TimeGenerated > ago(1h)
| where ResultType == '500121' // MFA denied by user
| summarize MFADenials = count() by UserPrincipalName, IPAddress
| where MFADenials > 5
| sort by MFADenials desc
```

Impossible Travel Detection

```
// Same user logged in from 2+ countries
SignIns
| where TimeGenerated > ago(24h)
| where ResultType == '0'
| summarize
Countries = make_set(Location),
CountryCount = dcount(Location),
IPs = make_set(IPAddress)
by UserPrincipalName
| where CountryCount > 1
| sort by CountryCount desc
```

Account Compromise — Success After Many Failures

```
let failed =
SignIns
| where TimeGenerated > ago(1h)
| where ResultType != '0'
| summarize FailCount = count() by UserPrincipalName
| where FailCount > 5;
SignIns
| where TimeGenerated > ago(1h)
| where ResultType == '0'
| join kind=inner failed on UserPrincipalName
| project TimeGenerated, UserPrincipalName, IPAddress, Location, FailCount
| sort by FailCount desc
```

Sign-Ins from High-Risk Countries

```
let watchlist = dynamic(['Russia','Nigeria','China','North Korea','Iran','Belarus']);
SignIns
| where TimeGenerated > ago(24h)
| where Location in (watchlist)
| summarize
Attempts = count(),
Users = make_set(UserPrincipalName)
by Location, IPAddress
| sort by Attempts desc
```

● MASTER

5. SENTINEL ANALYTICS RULES

Anatomy of a Sentinel Analytics Rule

```
// This is the KQL query you paste into the Analytics Rule editor
let timeframe = 1h;
let threshold = 10;
SignIns
| where TimeGenerated > ago(timeframe)
| where ResultType != '0'
| summarize FailedAttempts = count() by
UserPrincipalName, IPAddress, Location
| where FailedAttempts > threshold
// In the rule wizard:
// Run every: 1 hour
// Lookup data from: Last 1 hour
// Alert threshold: Is greater than 0
// Map entities: Account = UserPrincipalName, IP = IPAddress
```

■ *Map entities in the rule wizard so Sentinel can link incidents to user profiles and IP intelligence automatically.*

● INTERMEDIATE

6. VISUALISATIONS — render

Chart Types

Chart Type	Best Used For	KQL
`timechart`	Events over time / trends	| render timechart
`barchart`	Compare categories	| render barchart
`columnchart`	Compare values side by side	| render columnchart
`piechart`	Show proportions	| render piechart
`scatterchart`	Correlate two values	| render scatterchart
`areachart`	Volume over time	| render areachart

```
// Full example — login trend chart
SignIns
| where TimeGenerated > ago(7d)
| summarize
Failed = countif(ResultType != '0'),
Success = countif(ResultType == '0')
by bin(TimeGenerated, 1d)
| render timechart with (title='Daily Login Trend')
```

● INTERMEDIATE

7. COMMON SENTINEL TABLES REFERENCE

Table	Contains	Key Columns
`SigninLogs`	Azure AD / Entra sign-ins	UserPrincipalName, IPAddress, ResultType
`AuditLogs`	Azure AD changes	OperationName, InitiatedBy, TargetResources
`SecurityEvent`	Windows Security events	EventID, Account, Computer, LogonType
`CommonSecurityLog`	Firewall / IDS (CEF)	DeviceAction, SourceIP, DestinationIP
`Syslog`	Linux system logs	SyslogMessage, Computer, Facility
`AzureActivity`	Azure subscription actions	OperationName, Caller, ResourceGroup
`OfficeActivity`	Microsoft 365 events	Operation, UserId, ClientIP
`DeviceEvents`	MDE endpoint events	ActionType, DeviceName, InitiatingProcess
`DeviceProcessEvents`	Process creation (MDE)	FileName, ProcessCommandLine, AccountName
`DeviceNetworkEvents`	Network connections (MDE)	RemoteIP, RemoteUrl, ActionType
`SecurityAlert`	All Defender/Sentinel alerts	AlertName, AlertSeverity, Entities
`ThreatIntelIndicator`	Threat intel IOCs	IndicatorType, ThreatType, NetworkIP
`AADRiskyUsers`	Risky user detections	UserPrincipalName, RiskLevel, RiskDetail
`IdentityLogonEvents`	Identity-based logons (M365D)	AccountUpn, IPAddress, LogonType

● MASTER

8. QUICK QUERY TEMPLATES

Universal Investigation Template

```
// Copy this skeleton for any investigation
let timeframe = 24h;
let target = 'user@domain.com';
TableName
| where TimeGenerated > ago(timeframe) // 1. Time filter FIRST
| where SomeColumn == target // 2. Narrow scope
| project col1, col2, col3 // 3. Keep useful columns
| summarize count() by col1 // 4. Aggregate
| sort by count_ desc // 5. Rank results
| take 100 // 6. Limit output
```

Threat Hunting Starter Queries

```
// Who logged in outside business hours? (before 7am or after 8pm)
SignIns
| where TimeGenerated > ago(7d)
| extend HourUTC = datetime_part('hour', todatetime(TimeGenerated))
| where HourUTC < 7 or HourUTC > 20
| where ResultType == '0'
| summarize AfterHoursLogins = count() by UserPrincipalName, HourUTC
| sort by AfterHoursLogins desc
// New user accounts created in last 24h
AuditLogs
| where TimeGenerated > ago(24h)
| where OperationName == 'Add user'
| project TimeGenerated, InitiatedBy, TargetResources
// Service accounts logging in interactively (unusual)
SignIns
| where TimeGenerated > ago(24h)
| where UserPrincipalName startswith 'svc'
| where IsInteractive == true
| project TimeGenerated, UserPrincipalName, IPAddress, AppDisplayName
```

● ADVANCED

9. PERFORMANCE & BEST PRACTICES

	Best Practice
■ DO	Put TimeGenerated filter FIRST in every query
■ DO	Use has instead of contains for whole-word text search
■ DO	Use dcount() for approximate distinct counts (faster than count(distinct))
■ DO	Filter early — reduce rows before joining or summarizing
■ DO	Use let statements to make queries readable and reusable

■ DO	Name your summarize columns (e.g. count() as Events, not just count())
■ AVOID	Running queries without a time filter — scans all data
■ AVOID	Using contains when has will do — contains is a full scan
■ AVOID	Joining large tables without filtering both sides first
■ AVOID	Using order by before summarize — sort at the very end
■ AVOID	Smart/curly quotes — always use straight quotes ' or "

Keyboard Shortcuts in ADX Web UI

Shortcut	Action
`Shift + Enter`	Run selected query only
`F5`	Run entire query
`Ctrl + Space`	Trigger autocomplete
`Ctrl + K + C`	Comment selected lines
`Ctrl + /`	Toggle comment
`Ctrl + Z`	Undo
`Ctrl + Shift + F`	Format / pretty-print query

KQL Cheat Sheet — Azure Data Explorer · Log Analytics · Microsoft Sentinel | Covers: Beginner → Intermediate → Advanced → Master | Practice dataset: SignIns table from generated CSV