# SQL to Kusto query translation

Kusto supports subset of SQL language. See the list of SQL known issues (../api/tds/sqlknownissues.html) for the full list of unsupported features.

Primary language to interact with Kusto is KQL (Kusto Query Language), and in order to make transition and learning experience easier, you can use Kusto service to translate SQL queries to KQL. This can be achieved by sending SQL query to Kusto services prefixing it with 'EXPLAIN' verb.

For example:

```
[Click to run the query] (https://help.kusto.windows.net:443/Samples?query=H4sIAAAAA
EXPLAIN
SELECT COUNT_BIG(*) as C FROM StormEvents
```

| Query |
| --- |
| StormEvents |

# SQL to Kusto cheat sheet

The table below shows sample queries in SQL and thier KQL equivalients.

| Category | SQL Query | Kusto Query |
| --- | --- | --- |
| Select data from table | `SELECT * FROM dependencies` | `dependencies` |
| -- | `SELECT name, resultCode FROM dependencies` | `dependencies \| project name, resultCode` |
| -- | `SELECT TOP 100 * FROM dependencies` | `dependencies \| take 100` |
| Null evaluation | `SELECT * FROM dependencies WHERE resultCode IS NOT NULL` | `dependencies \| where isnotnull(resultCode)` |
| Comparison operatots (date) | `SELECT * FROM dependencies WHERE timestamp > getdate()-1` | `dependencies \| where timestamp > ago(1d)` |
| -- | `SELECT * FROM dependencies WHERE timestamp BETWEEN ... AND ...` | `dependencies \| where timestamp > datetime(2016-10-01) and timestamp <= datetime(2016-11-01)` |

| Category | SQL Query | Kusto Query |
|---|---|---|
| Comparison operators (string) | `SELECT * FROM dependencies`<br>`WHERE type = "Azure blob"` | `dependencies`<br>`\| where type == "Azure blob"` |
| -- | `-- substring`<br>`SELECT * FROM dependencies`<br>`WHERE type like "%blob%"` | `// substring`<br>`dependencies`<br>`\| where type contains "blob"` |
| -- | `-- wildcard`<br>`SELECT * FROM dependencies`<br>`WHERE type like "Azure%"` | `// wildcard`<br>`dependencies`<br>`\| where type startswith "Azure"`<br>`// or`<br>`dependencies`<br>`\| where type matches regex "^Azure.*"` |
| Comparison (boolean) | `SELECT * FROM dependencies`<br>`WHERE !(success)` | `dependencies`<br>`\| where success == "False"` |
| Distinct | `SELECT DISTINCT name, type FROM dependencies` | `dependencies`<br>`\| summarize by name, type` |
| Grouping, Aggregation | `SELECT name, AVG(duration) FROM dependencies`<br>`GROUP BY name` | `dependencies`<br>`\| summarize avg(duration) by name` |
| Column aliases, Extending | `SELECT operationName as Name, AVG(duration) as AvgD FROM dependencies`<br>`GROUP BY name` | `dependencies`<br>`\| summarize AvgD = avg(duration) by operationName`<br>`\| project Name = operationName, AvgD` |
| Ordering | `SELECT name, timestamp FROM dependencies`<br>`ORDER BY timestamp ASC` | `dependencies`<br>`\| project name, timestamp`<br>`\| order by timestamp asc nulls last` |
| Top n by measure | `SELECT TOP 100 name, COUNT(*) as Count FROM dependencies`<br>`GROUP BY name`<br>`ORDER BY Count DESC` | `dependencies`<br>`\| summarize Count = count() by name`<br>`\| top 100 by Count desc` |
| Union | `SELECT * FROM dependencies`<br>`UNION`<br>`SELECT * FROM exceptions` | `union dependencies, exceptions` |

| Category | SQL Query | Kusto Query |
| --- | --- | --- |
| -- | SELECT * FROM dependencies<br>WHERE timestamp > ...<br>UNION<br>SELECT * FROM exceptions<br>WHERE timestamp > ... | dependencies<br>\| where timestamp > ago(1d)<br>\| union<br>(exceptions<br>\| where timestamp > ago(1d)) |
| Join | SELECT * FROM dependencies<br>LEFT OUTER JOIN exception<br>ON dependencies.operation_Id = exceptions.operation_Id | dependencies<br>\| join kind = leftouter<br>(exceptions)<br>on $left.operation_Id == $right.operation_Id |
| Nested queries | SELECT * FROM dependencies<br>WHERE resultCode ==<br>(SELECT TOP 1 resultCode FROM dependencies<br>WHERE resultId = 7<br>ORDER BY timestamp DESC) | dependencies<br>\| where resultCode == toscalar(<br>dependencies<br>\| where resultId == 7<br>\| top 1 by timestamp desc<br>\| project resultCode) |
| Having | SELECT COUNT(*) FROM dependencies<br>GROUP BY name<br>HAVING COUNT(*) > 3 | dependencies<br>\| summarize Count = count() by name<br>\| where Count > 3 |