KQL Syntax Guide

We divide the KQL syntax into two main categories: CLI Commands and Statements.

CLI Commands

Commands have similar functionality of shell commands. You can use commands to explore the metadata of KQL CLI.

LIST TOPICS

List the available KQL topics in the system. A KQL topics represents a topic in Kafka along with metadata such as the format of the data in the Kafka topic. Currently KQL supports JSON, Avro and csv formats for data in a topic.

LIST STREAMS

List the available streams in KQL. A stream in KQL is a KQL topic along with a schema that specifies the structure of the data in the topic.

LIST TABLES

List the available tables in KQL. A table is KQL is a KQL topic along with a schema and a state store.

DESCRIBE <stream/table name>

Describe the schema of the stream/table with the given name.

SHOW QUERIES

Shows the list of continuous queries that are currently running in KQL. The schema information includes the list of columns with their type.

TERMINATE <query-id>

Terminates a continuous query with the given query id.

HELP

Shows the above commands and their descriptions.

EXPORT CATALOG TO

Exports the current catalog to a JSON file with the given path. Usage:

```
kql> export catalog to '<path to the output
file>'
```

Example:

```
kql> export catalog to
'/tmp/kql_catalog.json'
```

PRINT

Prints a sample of contents in a KQL topic. This is done via a simple streams app that runs for 3 seconds. Usage:

```
kql> print <topic name> [<INTERVAL|SAMPLE>
<number>]
```

You can specify an interval number so the output only includes the records that fall in the sampling rate. For instance, the following example will print every 10th message in the given topic:

```
kql> print orders sample
10
```

KQL Statements

CREATE TOPIC

Creates a new KQL topic with the given properties.

```
CREATE TOPIC topic_name WITH ( property_name = expression [, ...]
)
```

The properties that need to be set:

- *KAFKA_TOPIC*: The name of kafka topic associated with the new KQL topic. This is a required property.
- *FORMAT*: Specifies the format in which the topic data is serialized in. Currently, KQL supports *json*, *avro* and *csv*. This is a required property.
- AVROSCHEMAFILE: The path to the avro schema file. If the format us set to avro, avroschemafile will be a required property.

Examples:

```
kql> CREATE TOPIC orders_topic WITH (format = 'avro',
avroschemafile='/Users/hojjat/avro_order_schema.avro',kafka_topic='orders_kafka_topic_avro');

kql> CREATE TOPIC users_topic WITH (format = 'json', kafka_topic='users_kafka_topic_json');
```

CREATE STREAM

Create a new KQL stream with the specified columns and properties.

```
CREATE STREAM stream_name (  { column_name data_type} [, ...] ) WITH ( property_name = expression [, ...]
)
```

The supported column data types currently are BOOELAN(BOOL), INTEGER(INT), BIGINT(LONG), DOUBLE and VARCHAR (STRING).

The properties that need to be set:

- *TOPICNAME*: The name of the KQL topic that this streams is built upon. The KQL topic should already exist in the catalog. This is a required property.
- *KEY*: The name of the column that is the key. This is a required property.

Examples:

```
kql> CREATE STREAM orders (ordertime bigint, orderid varchar, itemid varchar, orderunits double) WITH
(topicname = 'orders_topic' , key='ordertime');

kql> CREATE STREAM pageview (viewtime bigint, userid varchar, pageid varchar) WITH (topicname =
'pageview_topic',key='viewtime');
```

CREATE TABLE

Create a new KQL table with the specified columns and properties.

```
CREATE TABLE table_name (  { column_name data_type} [, ...] ) WITH ( property_name = expression [, ...]
)
```

The supported column data types currently are BOOELAN(BOOL), INTEGER(INT), BIGINT(LONG), DOUBLE and VARCHAR (STRING).

The properties that need to be set:

- *TOPICNAME*: The name of the KQL topic that this streams is built upon. The KQL topic should already exist in the catalog. This is a required property.
- *KEY*: The name of the column that is the key. This is a required property.
- *STATESTORE*: The name of the state store that is used for the materialized ktable. This is a required property.

Example:

```
kql> CREATE TABLE users (usertime bigint, userid varchar, regionid varchar, gender varchar) WITH
(topicname = 'users_topic', key='userid', statestore='user_statestore');
```

SELECT

Selects rows from a KQL stream or table. The result of this statement will be printed out in the console. To stop the continuous query type "***CLOSE***"

```
SELECT select_expr [, ...] FROM from_item [, ...] [ WHERE condition ]
```

where *from_item* is one of the following:

```
table_name [ [ AS ]
alias]
```

```
from_item LEFT JOIN from_item ON
join_condition
```

Example:

```
SELECT * FROM orders WHERE orderunits > 5
;
```

CREATE STREAM AS SELECT

Create a new KQL stream along with the corresponding KQL topic and kafka topic and stream the result of the SELECt query into the topic.

```
CREATE STREAM stream_name [WITH ( property_name = expression [, ...] )] AS SELECT  select_expr [, ...]
FROM from_item [, ...] [ WHERE condition ]
```

The WITH section can be used to set the properties for the result KQL topic. The properties that can be set are as the following:

- *KAFKA_TOPIC*: The name of KQL topic and the corresponding kafka topic associated with the new KQL stream. If not set the name of the stream will be used as default.
- *FORMAT*: Specifies the format in which the result topic data is serialized in. Currently, KQL supports *json*, *avro* and *csv*. If not set the same format of the input stream will be used.
- AVROSCHEMAFILE: The path to write the avro schema file for the result. If the output format is avro, avroschemafile will be set. If not set the generated schema file will be written to "/tmp/" folder with the name of the stream as the file name.

Examples:

```
CREATE STREAM bigorders_json WITH (format = 'json', kafka_topic='bigorders_topic') AS SELECT * FROM
orders WHERE orderunits > 5 ;

CREATE STREAM enrichedpageview_female AS SELECT users.userid AS userid, pageid, regionid, gender FROM
pageview LEFT JOIN users ON pageview.userid = users.userid WHERE gender = 'FEMALE';

CREATE STREAM enrichedpageview_female_8 AS SELECT userid, pageid, regionid FROM enrichedpageview_female
WHERE pageid LIKE '%8';
```

Functions

KQL provides a set of internal functions that can use used in query expressions. The following is the list of the currently available functions:

Function name	Description	Example
`LCASE`	Convert a string into lower case.	`LCASE(col1)`
`UCASE`	Convert a string into upper case.	`UCASE(col1)`
`SUBSTRING`	Return the substring with the start and end indexes	`SUBSTRING(col1, 2, 5)`
`CONCAT`	Concatenate two strings	`CONCAT(col1, '_hello')`
`TRIM`	Trim the spaces from the beginning and end of a string	`TRIM(col1)`
`LEN`	The length of an string	`LEN(col1)`
`ABS`	The absolute value of a value.	`ABS(col3)`
`CEIL`	The ceiling of a value.	`CEIL(col3)`
`FLOOR`	The floor of a value.	`FLOOR(col3)`
`ROUND`	Round a value.	`ROUND(col3)`
`RANDOM`	Return a random value between 0 and 1.0	`RANDOM()`