ES|QL Query Commands and Functions

Overview

Elasticsearch Query Language (ES|QL) is a powerful, SQL-like query language used to interact with Elasticsearch. It enables users to retrieve, filter, aggregate, and transform data efficiently. This document provides a comprehensive list of ES|QL commands and functions, including their descriptions, syntax, use cases, grammar, and examples.

---

1. SOURCE COMMANDS

ES|QL supports the following source commands:

FROM

Description:

Defines the index or data source from which records will be queried.

Syntax:

FROM index_name

Example:

FROM logs

ROW

Description:

Creates a row with specified values.

Syntax:

ROW field1 = value1, field2 = value2

Example:

ROW name = 'John Doe', age = 30

SHOW

Description:

Displays available indices, functions, or metadata.

Syntax:

SHOW tables

Example:

SHOW indices

---

2. PROCESSING COMMANDS

ES|QL supports the following processing commands:

DISSECT

Description:

Parses a string field into multiple fields using a simple pattern.

Syntax:

DISSECT field PATTERN

Example:

FROM logs

| DISSECT message "%{timestamp} %{loglevel} %{message}"

DROP

Description:

Removes specific fields from records.

Syntax:

DROP field1, field2

Example:

FROM logs

| DROP sensitive_data

ENRICH

Description:

ENRICH enables you to add data from existing indices as new columns using an enrich policy. Refer to Data Enrichment for information about setting up a policy.

Syntax:

ENRICH policy [ON match_field] [WITH [new_name1 =]field1, [new_name2 =]field2, ...]

Parameters:

- **policy**: The name of the enrich policy. You need to create and execute the enrich policy first.
- **mode**: The mode of the enrich command in cross-cluster ES|QL. See enrich across clusters.
- **match_field**: The match field. ENRICH uses its value to look for records in the enrich index. If not specified, the match will be performed on the column with the same name as the match_field defined in the enrich policy.

- **fieldX**: The enrich fields from the enrich index that are added to the result as new columns. If a column with the same name as the enrich field already exists, the existing column will be replaced by the new column. If not specified, each of the enrich fields defined in the policy is added. A column with the same name as the enrich field will be dropped unless the enrich field is renamed.
- **new_nameX**: Enables you to change the name of the column that's added for each of the enrich fields. Defaults to the enrich field name. If a column has the same name as the new name, it will be discarded. If a name (new or original) occurs more than once, only the rightmost duplicate creates a new column.

Example:

FROM network_logs

| ENRICH ip_address USING geo_data

EVAL

Description:

Creates or modifies fields based on expressions.

Syntax:

EVAL new_field = expression

Example:

FROM users

| EVAL full_name = CONCAT(first_name, ' ', last_name)

GROK

Description:

Extracts data from text fields using patterns.

Syntax:

GROK field PATTERN

Example:

```
FROM logs

| GROK message "%{TIMESTAMP_ISO8601:timestamp} %{WORD:level}
%{GREEDYDATA:msg}"
```

KEEP

Description:

Retains only specified fields.

Syntax:

```
KEEP field1, field2
```

Example:

```
FROM transactions

| KEEP user_id, transaction_amount
```

LIMIT

Description:

Restricts the number of returned results.

Syntax:

```
LIMIT number
```

Example:

```
FROM logs

| LIMIT 100
```

MV_EXPAND

Description:

Expands multi-valued fields into separate rows.

Syntax:

MV_EXPAND field

Example:

FROM products

| MV_EXPAND tags

RENAME

Description:

Renames a field.

Syntax:

RENAME old_field AS new_field

Example:

FROM contacts

| RENAME email AS contact_email

SORT

Description:

Sorts records based on a field.

Syntax:

SORT field [ASC|DESC]

Example:

FROM events

| SORT timestamp DESC

STATS

Description:

Performs aggregations on fields.

Syntax:

STATS aggregation(field) [BY field1, field2, ...]

Example:

FROM sales

| STATS AVG(revenue) BY region

WHERE

Description:

Filters records based on a condition.

Syntax:

WHERE condition

Example:

FROM logs

| WHERE status_code == 500

---

3. FUNCTIONS

ES|QL supports a wide range of functions, including:

Aggregate Functions:

AVG

Description: Computes the average of numeric values.

Syntax:

STATS AVG(field)

Example:

FROM employees | STATS AVG(salary)

COUNT

Description: Returns the number of occurrences.

Syntax:

STATS COUNT(field)

Example:

FROM logs | WHERE status_code == 200 | STATS COUNT(*)

MAX

Description: Returns the maximum value.

Syntax:

STATS MAX(field)

Example:

FROM employees | STATS MAX(salary)

Conditional Functions and Expressions:

CASE

Description: Performs conditional evaluations.

Syntax:

EVAL new_field = CASE WHEN condition THEN result ELSE default END

Example:

FROM users | EVAL category = CASE WHEN age > 18 THEN 'Adult' ELSE 'Minor' END

Date and Time Functions:

NOW

Description: Returns the current timestamp.

Syntax:

NOW()

Example:

FROM logs | EVAL current_time = NOW()

String Functions:

CONCAT

Description: Concatenates multiple strings.

Syntax:

EVAL new_field = CONCAT(string1, string2)

Example:

FROM users | EVAL full_name = CONCAT(first_name, ' ', last_name)

Type Conversion Functions:

TO_STRING

Description: Converts values to string format.

Syntax:

EVAL new_field = TO_STRING(field)

Example:

FROM users | EVAL age_string = TO_STRING(age)

Operators:

IN

Description: Checks if a value exists in a list.

Syntax:

WHERE field IN (value1, value2, ...)

Example:

FROM users | WHERE role IN ('admin', 'editor')

This document provides a structured list of ES|QL commands and functions. More advanced operations and optimizations can be explored in further Elasticsearch documentation.