SPL2 Command Quick Reference

The following commands are supported in SPL2.

Command	Description
apply	Used in conjunction with the `fit` command for Machine Learning analysis.
bin	Puts continuous numerical values into discrete sets, or bins. **Example:** Return the average for a field for a specific time span. Bin the search results using a 5 minute time span on the `_time` field. Return the average "thruput" of each "host" for each 5 minute time span. `...| bin span=5m _time | stats avg(thruput) by _time, host`
dedup	Removes the events that contain an identical combination of values for the fields that you specify. **Example:** Remove duplicates of results with the same `host` value. `... | dedup host`
eval	Calculates an expression and puts the resulting value into a search results field. **Example:** Create a new field that contains the result of a calculation. Create a new field called `velocity` in each event. Calculate the velocity by dividing the values in the `distance` field by the values in the `time` field. `... | eval velocity=distance/time` **Example:** Use the `if` function to analyze field values. Create a new field called `error` in each event. Using the if function, set the value in the error field to `OK` if the status value is 200. Otherwise set the error field value to `Problem`. `... | eval error = if(status == 200, "OK", "Problem")`
fields	Keeps or removes fields from search results based on the list of fields that you specify. **Example:** Specify a list of fields to include in the search results. Return only the host and src fields from the search results. `... | fields host, src`
fit	Used in conjunction with the `apply` command for Machine Learning analysis.
from	Retrieves data from a dataset, such as an index, metric index, lookup, view, or job. **Example:** Return data from the main index for the last 5 minutes. Group the results by host. Calculate the sum of the bytes field. Return the sum and the host fields where the sum of the bytes is greater than I MB. `| FROM main WHERE earliest=-5m@m AND latest=@m` `GROUP BY host` `SELECT sum(bytes) AS sum, host` `HAVING sum > 1024*1024`

head

Returns the first N number of specified results in search order.

Example: Stop searching when a null value is encountered. This example returns results while `action=purchase` or the `action` field does not exist in the results (`null=true`). A maximum of 50 results are returned.

```
...| head while (action="purchase") null=true 50
```

into

Sends results to a dataset that is writable, a dataset sink. Appends or replaces the dataset sink in the search data pipeline.

Example: Append the search results to the `mytable` dataset, which is a lookup kind of dataset.

```
... | into mode=append mytable
```

join

Combines the results from two datasets by using one or more common fields.

Example: Join datasets on fields that have the same name. Combine the results from a search with the `vendors` dataset. The data is joined on the `product_id` field, which is common to both datasets.

```
... | join left=L right=R where L.product_id=R.product_id vendors
```

lookup

Invokes field value lookups.

Example: Put corresponding information from a lookup dataset into your events.

Appends the data returned from your search results with the data in the `users` lookup dataset using the `uid` field. For search results that contains a `uid` field, the value in that field is matched with the `uid` field in the `users` lookup dataset. The `username` and `department` fields from the `users` lookup dataset are appended to each search result. If the search results already have the `username` and `department` fields, the OUTPUTNEW argument only fills in missing values in those fields.

```
... | lookup users uid OUTPUTNEW username, department
```

mvexpand

Expands the values of a multivalue field into separate events, one event for each value in the multivalue field.

Example: Expand the values in the `myfield` field.

```
... | mvexpand myfield
```

rename

Renames one or more fields.

Example: Rename a field with special characters. Rename the `ip-add` field to `IPAddress`. Field names that contain anything other than a-z, A-Z, 0-9, or "_", need single-quotation marks.

```
... | rename 'ip-add' AS IPAddress
```

reverse

Reverses the order of the search results.

Example:

```
…| reverse
```

rex	Use to either extract fields using regular expression named groups, or replace or substitute characters in a field using sed expressions. **Example:** Extract values from a field using a <regex-expression>. Extract "user", "app" and "SavedSearchName" from a field called `savedsearch_id` in scheduler.log events. `... | rex field=savedsearch_id "(?<user>w+);(?<app>w+);(?<SavedSearchName>w+)"` If the contents of the field is `savedsearch_id=bob;search;my_saved_search` then this rex command syntax extracts `user=bob`, `app=search`, and `SavedSearchName=my_saved_search`.

search	Retrieve events from indexes or filter the results of a previous search command in the pipeline.
	Example: Search for a field-value pair for a specific source IP (src).
	`| search src="192.0.2.0"`
	Example: Search for multiple field-value pairs with boolean and comparison operators. This example searches for events with code values of either 10, 29, or 43 and any `host` that is not "localhost", and an `xqp` value that is greater than 5.
	`| search (code=10 OR code=29 OR code=43) host!="localhost" xqp>5`

select	See the from command.
sort	Sorts all of the results by the specified fields. **Example:** This example sorts the results first by the `lastname` field in ascending order and then by the `firstname` field in descending order. `... | sort lastname, -firstname`
stats	Calculates aggregate statistics such as average, count, and sum, over the results set. **Example:** Takes the incoming result set and calculates the sum of the `bytes` field and groups the sums by the values in the `host` field. `... | stats sum(bytes) BY host`
streamstats	Adds a cumulative statistical value to each search result as each result is processed. **Example:** Use a <by-clause> to add a running count to search results. This search uses the `host` field to reset the count. For each search result, a new field is appended with a count of the results based on the `host` value. The count is cumulative and includes the current result. `... | streamstats count() BY host`
thru	Writes data to a writeable dataset and then passes the same data to the next command in the search string. With the `thru` command you can append or replace data in a dataset. **Example:** Appends all the incoming search result set to the `actions` dataset. Those same search results are also passed into the `eval` command. `... | thru actions | eval field=<expr>`

timechart	Creates a time series chart with corresponding table of statistics. **Example:** For each minute, calculate the average value of the `CPU` field for each `host`. `... | timechart span=1m avg(CPU) BY host`
timewrap	Compare data over a specific time period, such as day-over-day or month-over-month, or multiple time periods, such as a two week period over another two week period. **Example:** Display a timechart that has a span of 1 day for each count in a week over week comparison table. Each table column, which is the series, is 1 week of time. `... | timechart count span=1d | timewrap 1week`
union	Merges the results from two or more datasets into one dataset. One dataset can be piped into the union command and merged with a second dataset. **Example:** The following example merges events from the customers, orders, and vendors datasets. You must separate the dataset names with a comma. `| union customers, orders, vendors` **Example:** The following example appends the current results of the main search with the tabular results of errors from the subsearch. `... | stats count() BY category1 | union [search error | stats count() BY category2]`
where	Filters search results based on the outcome of a Boolean expression. **Example:** Use the `like` comparison operator similar to a wildcard. This example returns all results where the `ipaddress` field contains values that start with "192.". `... | where like(ipaddress, "192.")`