**Splunk Fields**

**1. Using Fields**

In Splunk, data is stored as events, and each event contains information that is organized into **fields**. Fields are essentially labels or categories that describe parts of an event. For example, a log line may contain a status field, a host field, or an action field.

* **Field Names vs. Values**
  + Field names are **case sensitive**.
  + Field values are **not case sensitive**.
* **Field Operators**  
  You can use operators in searches to filter results:
  + = and != work with both numbers and strings.
  + >, <, >=, and <= work only with numbers.

Example:

status=404

price>=100

These queries return only events that match the conditions.

* **Boolean Logic with Fields**  
  You can combine conditions with Boolean operators:
  + AND (implicit if you just use spaces)
  + OR (explicitly written)
  + NOT (explicitly written to exclude results)

Example:

index=security sourcetype=linux\_secure action=failure NOT (host="mail\*" OR host=www1)

* + This searches the security index, looks for events of type linux\_secure where the action is failure, and excludes hosts that begin with “mail” or are exactly “www1.”
* **Difference Between != and NOT**
  + != checks only events that have the field and compares the value.
  + NOT excludes all events with a certain value, even if some events do not have that field.

Example with three events:

status=200 user=john

status=404 user=jane

user=mary

* + status!=200 returns only the second event (because the third event does not have a status field to compare).
  + NOT status=200 returns the second and third events (it excludes only those with status=200).
* **OR Example**
* index=web (status=500 OR status=503 OR status=505)

This looks in the web index and finds events where the status is 500, 503, or 505.

* **IN Example**
* index=web status IN ("500", "503", "505")

This is a shorter way of writing multiple OR conditions.

* **Best Practice: Place fields Before stats**  
  When you use the stats command to calculate counts, sums, or averages, it is recommended to use the fields command before it. The fields command ensures only the required fields are processed, improving search performance.

Example:

index=web status IN ("500", "503", "505")

| fields status

| stats count by status

* **Including and Excluding Fields**
  + fields field1 field2 keeps only the listed fields.
  + fields - field1 removes a field from results.
  + You may also see fields +fieldname to emphasize inclusion.
* **Renaming Fields**  
  You can rename fields to make reports clearer:
* index=web status IN ("500", "503", "505")
* | stats count by status
* | rename status as "HTTP Status", count as "Number of Events"

This changes the column names in the results to make them easier to read.

**2. Using the Fields Sidebar**

When you run a search in Splunk, the Fields sidebar on the left helps you explore data more easily.

* **Selected Fields**  
  These are fields that are always shown in results (for example, \_time, \_raw, host, source, sourcetype).
* **Interesting Fields**  
  These are fields that occur in at least 20% of the events returned. They appear in the sidebar so you can quickly add them to your search.

From the sidebar, you can:

* Click on a field name to view its values.
* Click on a value to add it directly into your search query.

This makes it easier to refine searches without typing everything manually.

**3. Fields in Search Queries**

Fields are used directly in search queries to filter and refine results. Examples include:

* Exact match:
* status=200
* Inequality:
* duration>5
* Multiple conditions:
* action=login user=admin
* Boolean filters:
* NOT (host="web1" OR host="web2")
* IN operator for multiple values:
* status IN ("200", "404", "500")

The pipe symbol | is used to pass search results from one command to another, creating a pipeline of commands.

**4. Fields in Search Results**

When Splunk ingests data, it automatically extracts a set of default fields for every event. These include:

* host: The system or machine from which the event originated.
* source: The file, directory, or input source for the event.
* sourcetype: The format or type of the data (for example, syslog, apache logs, JSON).
* \_time: The timestamp of the event.
* \_raw: The full raw text of the log event.

These fields are automatically available for filtering and analysis as soon as the data is ingested.

**5. Enriching Data with Knowledge Objects**

Beyond the default fields, Splunk provides tools to enrich your data with additional knowledge objects:

* **Field Extractions**: Create custom fields by extracting values from raw data.
* **Field Aliases**: Assign alternative names to existing fields.
* **Calculated Fields (via eval)**: Use the eval command to create new fields based on existing ones.

Example with eval:

index=network sourcetype=cisco\_wsa\_squid

| stats sum(sc\_bytes) as Bytes by usage

| eval bandwidth = Bytes/1024/1024

Here, a new field called bandwidth is created by converting bytes into megabytes.

* **Lookups**: Enhance data by adding external information (for example, mapping IP addresses to geographic locations).
* **Tags and Event Types**: Organize and categorize events to make searches easier.

**Splunk - Visualization**

**1. Formatting Commands**

Formatting commands shape the data returned by Splunk searches, making it easier to view and analyze.

**fields**

* Used to include or exclude fields.
* Syntax:
  + fields field1 field2 → keeps only the specified fields.
  + fields - field1 field2 → removes both field1 and field2.
  + fields -field1 field2 → removes only field1.
* Improves search efficiency by reducing the amount of data returned.

**table**

* A transforming command used to create tabular views of results.
* Displays only the specified fields.
* Example:

index=web sourcetype=access\_combined product\_name=\*

| table JSESSIONID product\_name price

* Drops all fields except those explicitly listed.

**dedup**

* Removes duplicate events based on one or more fields.
* Splunk keeps the first unique combination and drops the rest.
* Example:

... | dedup JSESSIONID price

**addtotals**

* Adds row or column totals for numeric fields.
* Options:
  + row=true → adds a totals row.
  + col=true → adds a totals column.
  + label → customizes the totals row label.
  + labelfield → specifies the field where the label should appear.
  + fieldname → renames the totals column.
  + row=false → prevents adding a totals row.
* Example:

| addtotals col=true label="Total Sales" labelfield="product\_name" fieldname="Total By Product"

**fieldformat**

* Alters the display of field values without changing the raw data.
* Example:

... | fieldformat Total = "$" + tostring(Total, "commas")

* + Displays totals with a dollar sign and comma formatting.

**2. Visualizing Data**

Transforming commands prepare data for visualization.

**top**

* Finds the most common values in a field.
* Returns count and percentage by default.
* Default limit: 10.
* Options:
  + limit → change number of results.
  + countfield → rename count column.
  + percentfield → rename percent column.
  + showcount → show/hide count column.
  + showperc → show/hide percent column.
  + showother → include/exclude "Other".
* Example:

| top product\_name by Vendor limit=3 countfield="Number of Sales" showperc=false

**rare**

* Opposite of top.
* Finds the least common values in a field.

**stats**

* General aggregation function.
* Common functions:
  + sum
  + count
  + dc() → distinct count
  + avg
  + min
  + max
  + values
  + list
* Example:

| stats sum(price) as Total by product\_name

**chart**

* Similar to stats but optimized for two-dimensional aggregation.
* Works well with visualization tools.
* Example:

| chart sum(price) over product\_name by VendorCountry

**timechart**

* A time-aware version of chart.
* Automatically buckets events by time.
* The X-axis is always time.
* Options:
  + span → defines time intervals (e.g., 5m, 1h, 1d).
* Example:

| timechart span=1h count

**trendline**

* Adds moving averages to visualize trends.
* Types:
  + sma → Simple Moving Average (equal weights).
  + ema → Exponential Moving Average (recent values weighted more).
  + wma → Weighted Moving Average (manual linear weights).

**3. Generating Maps**

Splunk supports geographic visualization using specialized commands.

**Internal Fields**

* \_time → timestamp of the event.
* \_raw → raw event data.

**iplocation**

* Enriches IP addresses with geographic info.
* Fields added: Country, City, Region, Latitude, Longitude.

**geostats**

* Similar to stats, but geography-aware.
* Aggregates results by latitude and longitude.
* Key arguments:
  + latfield → latitude field.
  + longfield → longitude field.
  + globallimit → limit column count.

**geom**

* Draws geographic shapes (polygons) for choropleth maps.
* Works with geo lookup datasets (e.g., geo\_countries).

**Visualization Types**

* **Marker Map** → plots points.
* **Cluster Map** → groups nearby points.
* **Choropleth Map** → shades regions based on values.

**4. Single Value Visualizations**

* Display one key metric at a time.
* Often paired with formatting commands like fieldformat to improve readability.
* Example:

| stats sum(price) as TotalSales

| fieldformat TotalSales = "$" + tostring(TotalSales, "commas")

**5. Visual Formatting**

Visual formatting improves readability of charts and dashboards.

* **Column order** → determined by the order of fields in the table command.
* **Default limits** → chart command shows 10 columns by default (configurable with limit).
* **Chart overlay** → compare multiple data series in the same chart.
* **Single value visualization formatting** → add currency symbols, commas, or percentages.
* **Fieldformat vs eval**:
  + fieldformat → changes display only.
  + eval → changes actual values.