# Training – Day #3

15 June 2023 04:31

***Chart command ( continued)***

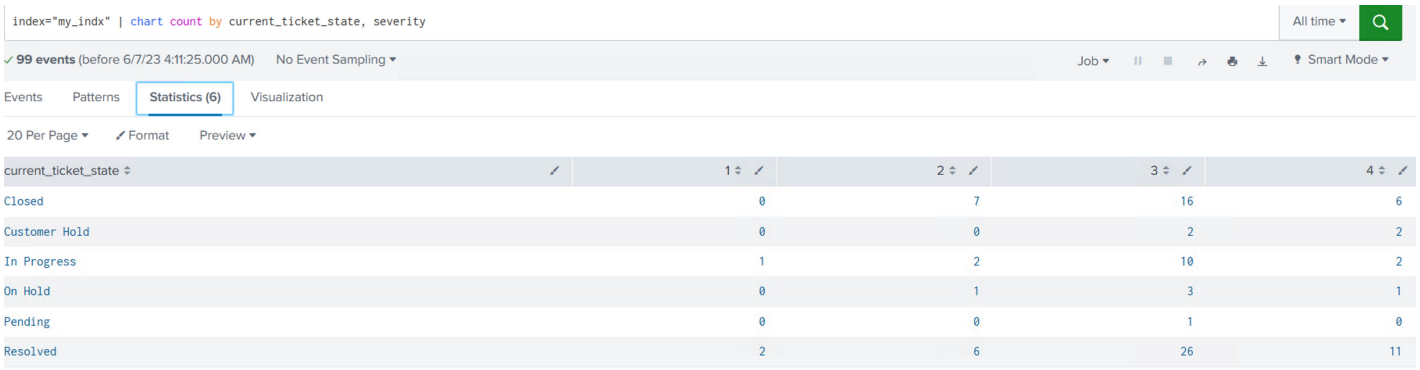
**#Chart Overlay**

It is used to show threshold line on above the current graph.

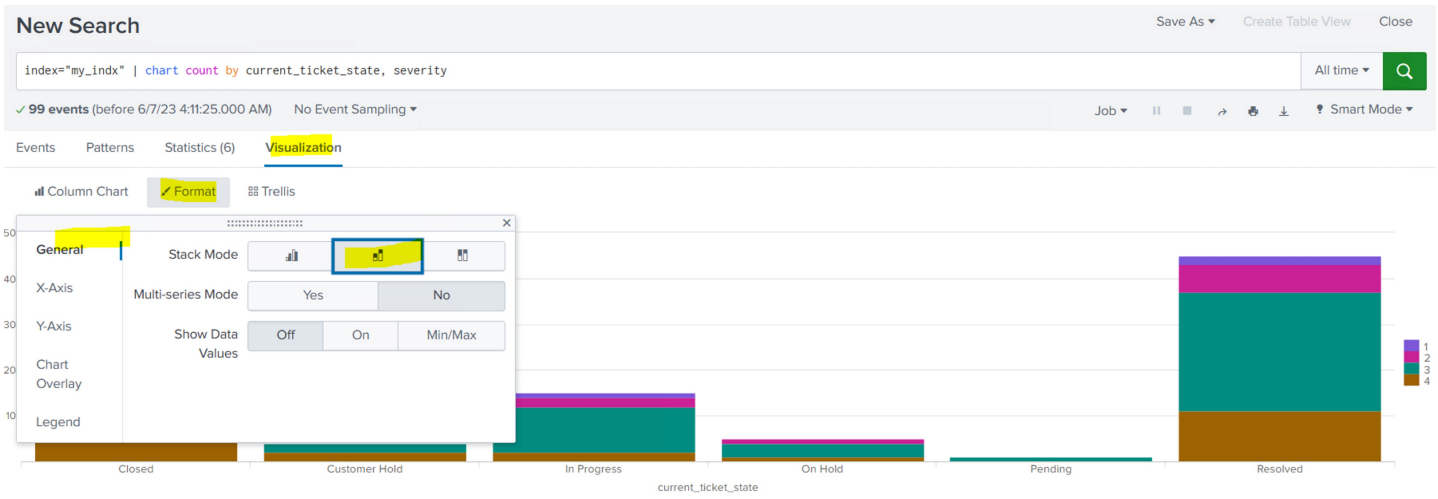
**#Stack in chart :**

Statistics will be same but in visualization, pattern will be different.

***index="my_indx" | chart count by current_ticket_state, severity***




| current_ticket_state | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Closed | 0 | 7 | 16 | 6 |
| Customer Hold | 0 | 0 | 2 | 2 |
| In Progress | 1 | 2 | 10 | 2 |
| On Hold | 0 | 1 | 3 | 1 |
| Pending | 0 | 0 | 1 | 0 |
| Resolved | 2 | 6 | 26 | 11 |



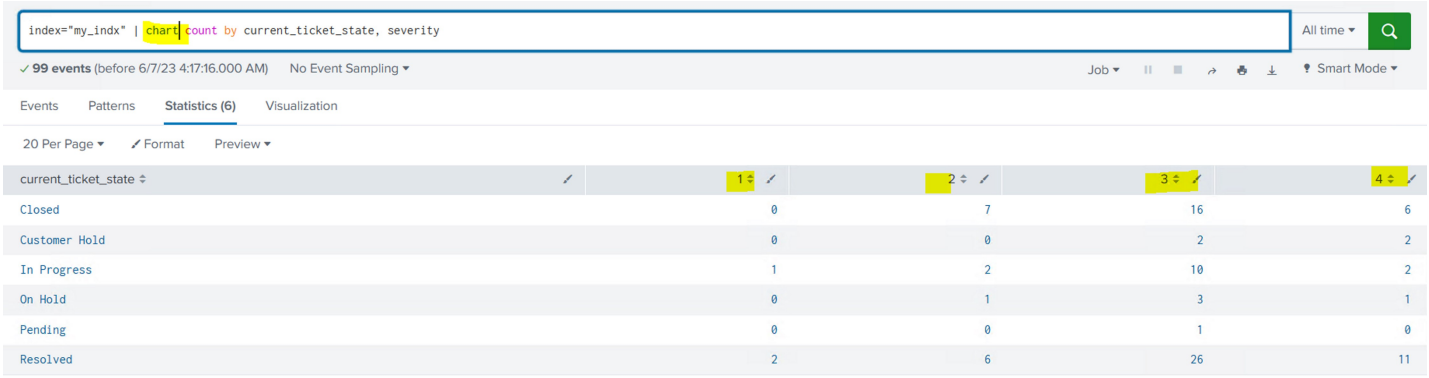

**Note : Log and stacking cannot be enabled at the same time. It will through error.**

---

**Question** : Difference between statistical output of below two queries :

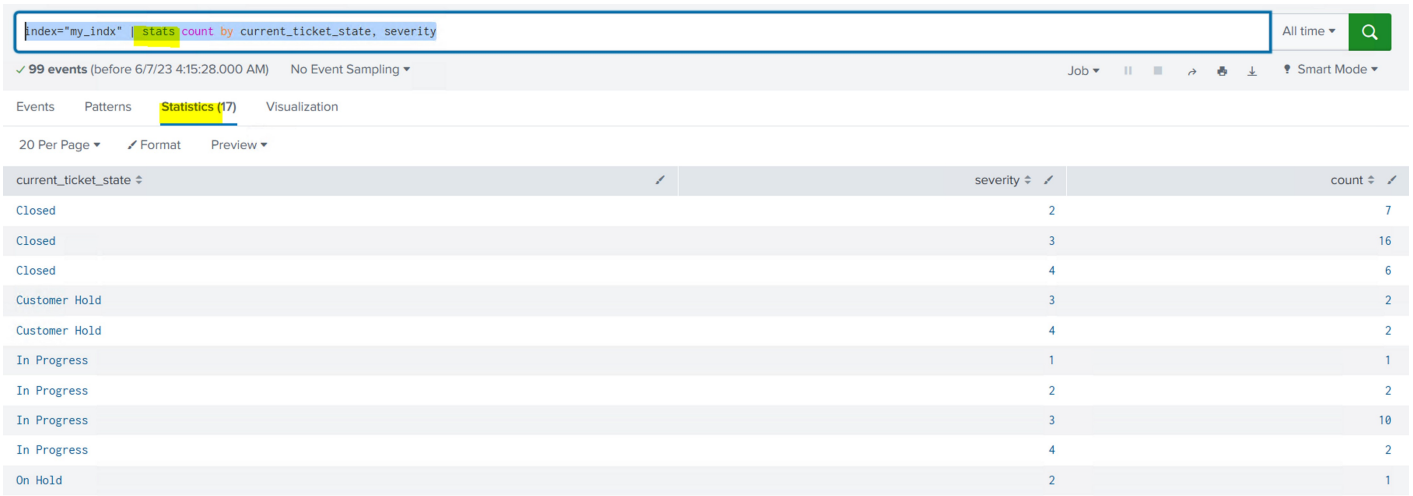
**Answer :**

**SPL1 : index="my_indx" | chart count by current_ticket_state, severity**




| current_ticket_state | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Closed | 0 | 7 | 16 | 6 |
| Customer Hold | 0 | 0 | 2 | 2 |
| In Progress | 1 | 2 | 10 | 2 |
| On Hold | 0 | 1 | 3 | 1 |
| Pending | 0 | 0 | 1 | 0 |
| Resolved | 2 | 6 | 26 | 11 |

**SPL2 : index="my_indx" | stats count by current_ticket_state, severity**



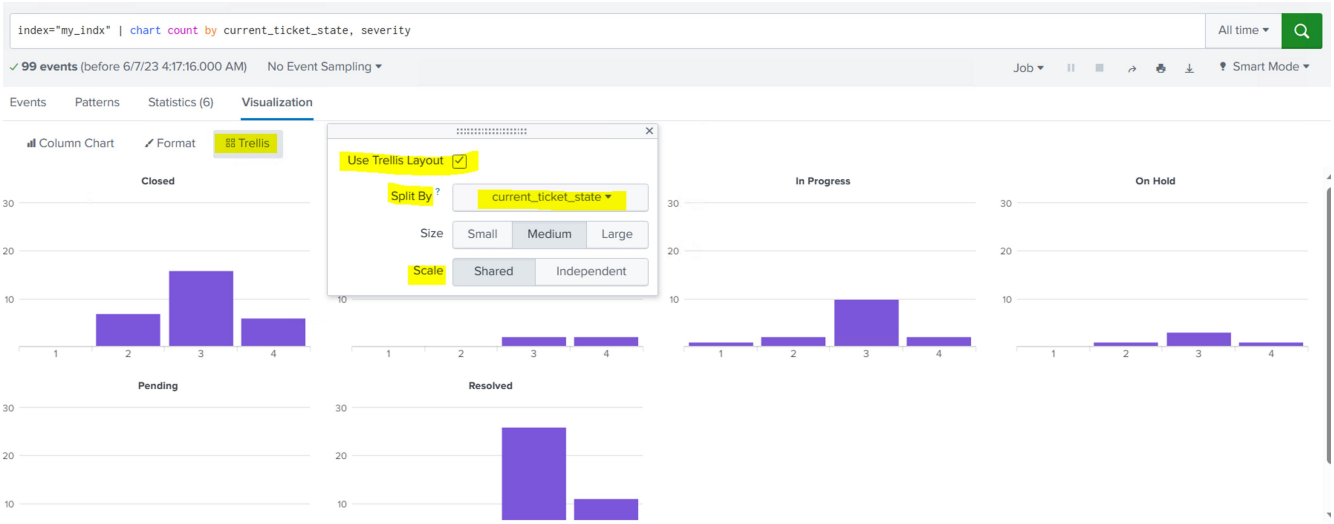

| current_ticket_state | severity | count |
|---|---|---|
| Closed | 2 | 7 |
| Closed | 3 | 16 |
| Closed | 4 | 6 |
| Customer Hold | 3 | 2 |
| Customer Hold | 4 | 2 |
| In Progress | 1 | 1 |
| In Progress | 2 | 2 |
| In Progress | 3 | 10 |
| In Progress | 4 | 2 |
| On Hold | 2 | 1 |

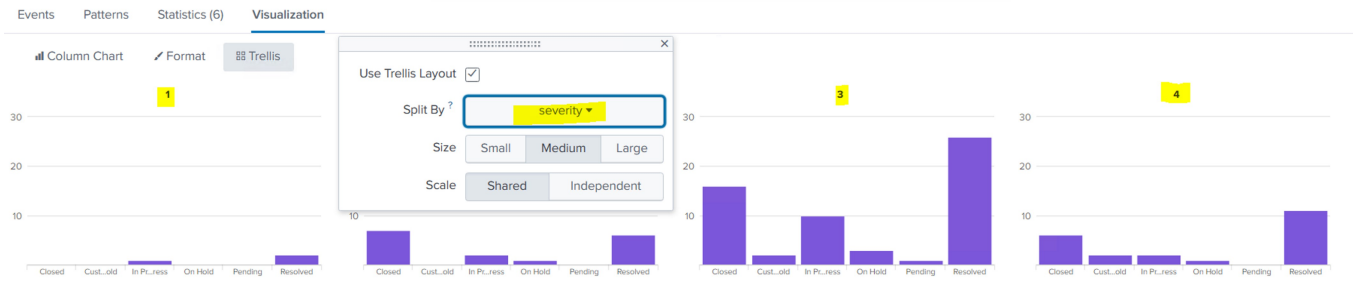
---

**#Trellis in chart :**

Trellis will convert one chart into multiple chart.

Scale can be "***Shared***" or it can be "***Independent***" for each graph.






**#Single Value Visualization :**

We have ***4 type of graph*** for Single Value visualization.

***(1) Single Value***

***(2) Radial Gauge***

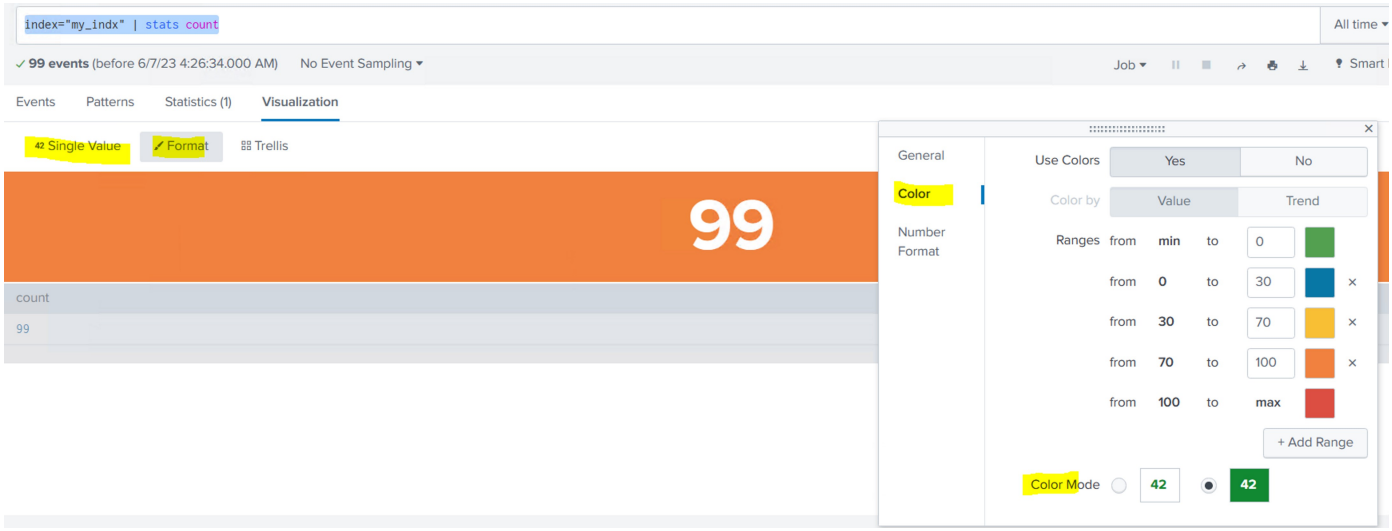
***(3) Marker Gauge***

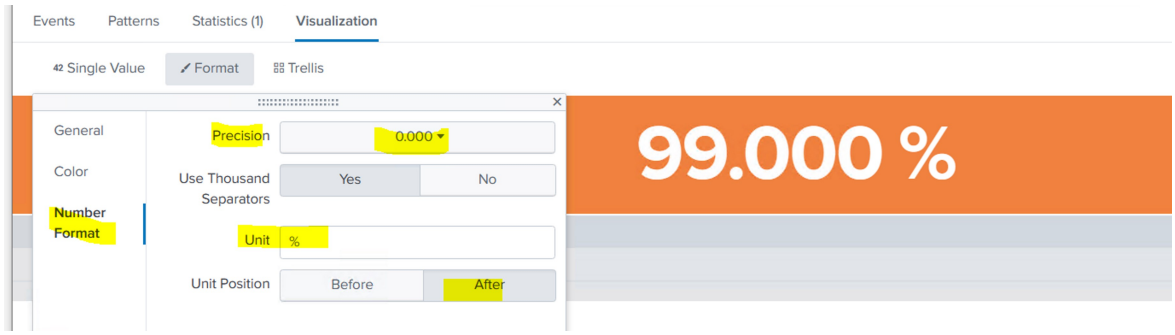
***(4) Filler Gauge***

***Note*** : Make sure "Trellis" is disabled.

***index="my_indx" | stats count***

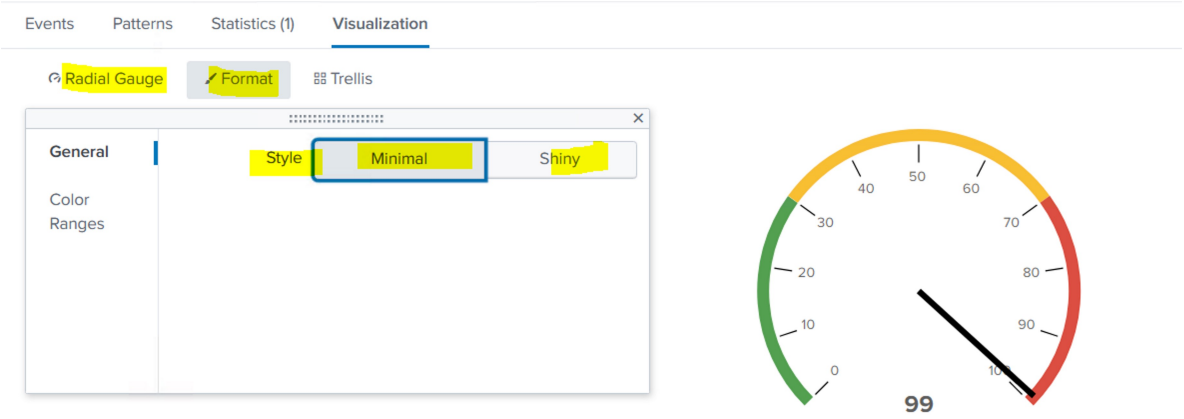
**1st way** : **Single Value:**



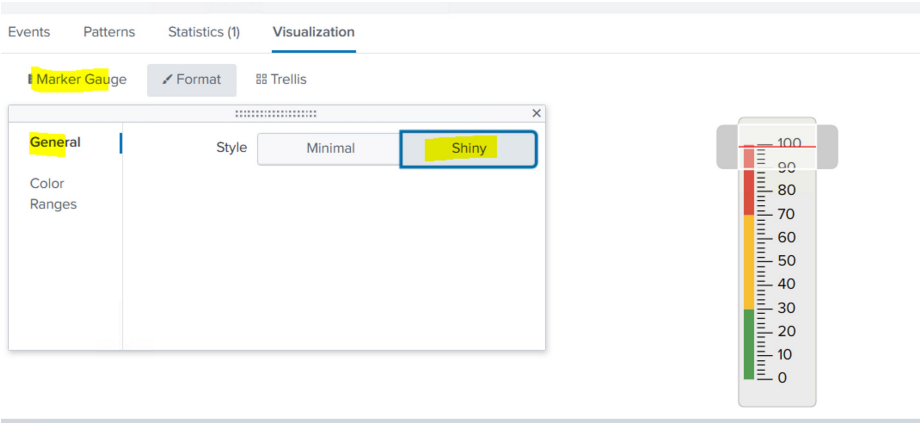
**2nd way** : **Radial Gauge:**

Two styles : **Minimal** and **Shiny**



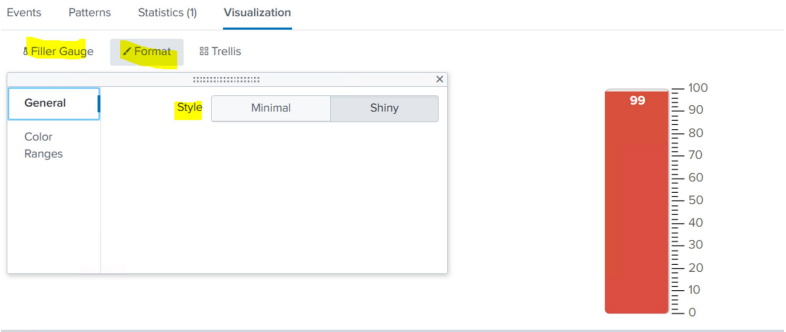

**3rd way** : **Marker Gauge:**

Two styles : **Minimal** and **Shiny**




**4th way** : **Filler Gauge:**

Two styles : **Minimal** and **Shiny**




**#Timechart :**

In timechart command, :

**X-axis** => **Default will be _time**

**Y-axis** => will be count/sum or other stats

A timechart is a statistical aggregation applied to a field to produce a chart, with **time used as the X-axis**.

You can specify a split-by field, where each distinct value of the split-by field becomes a series in the chart.

***If you use an eval expression, the split-by clause is required***.

With the **limit and agg options, you can specify series filtering.** These options are ignored if you specify an explicit where-clause.

If you set **limit=0, no series filtering occurs.**

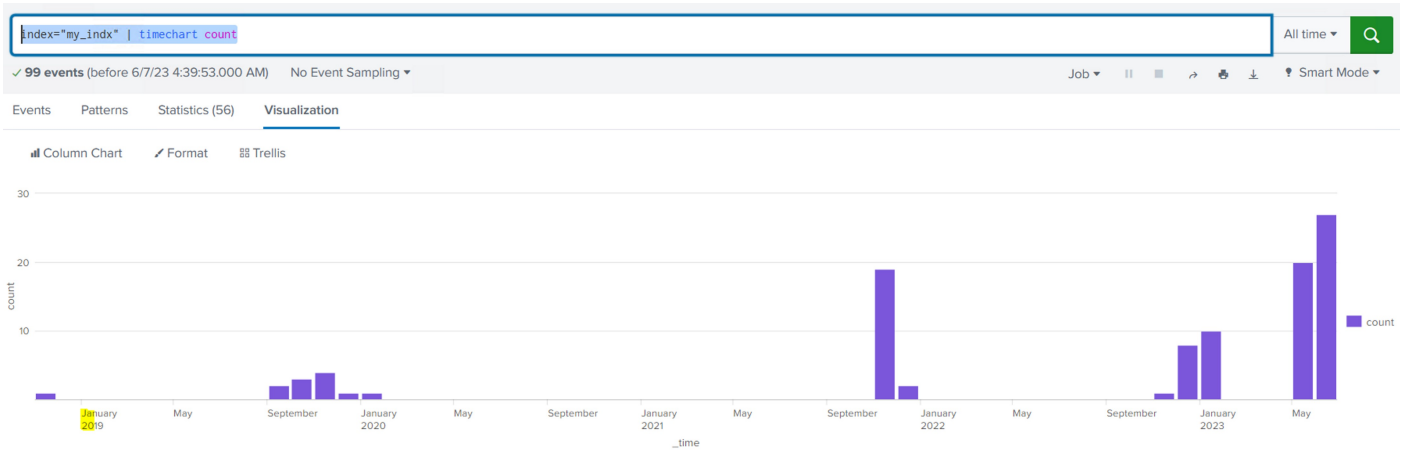
*From <https://docs.splunk.com/Documentation/Splunk/9.0.5/SearchReference/Timechart?ref=hk>*

### Default time spans

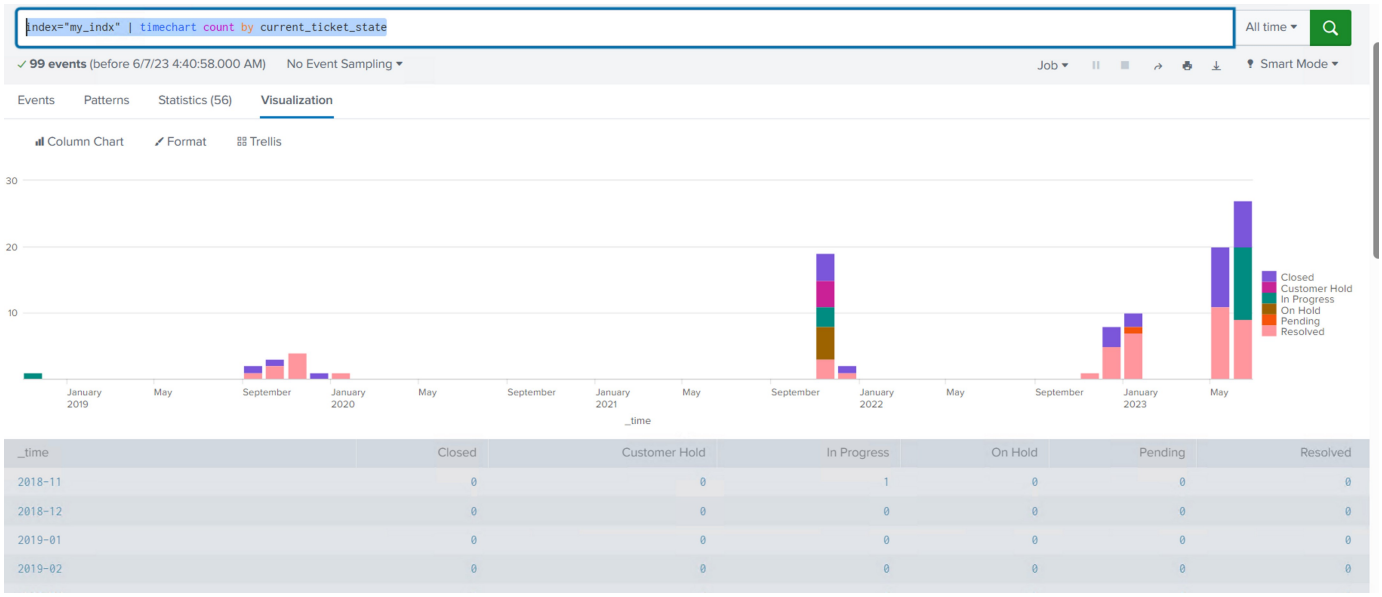
If you use the predefined time ranges in the time range picker, and do not specify the `span` argument, the following table shows the default span that is used.

| Time range | Default span |
|---|---|
| Last 15 minutes | 10 seconds |
| Last 60 minutes | 1 minute |
| Last 4 hours | 5 minutes |
| Last 24 hours | 30 minutes |
| Last 7 days | 1 day |
| Last 30 days | 1 day |
| Previous year | 1 month |

***index="my_indx" | timechart count***

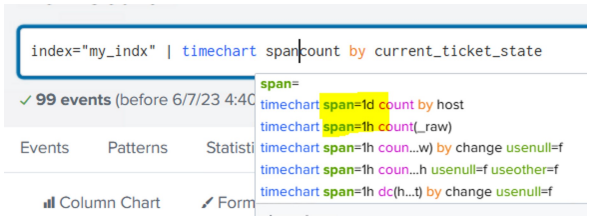



***index="my_indx" | timechart count by current_ticket_state***



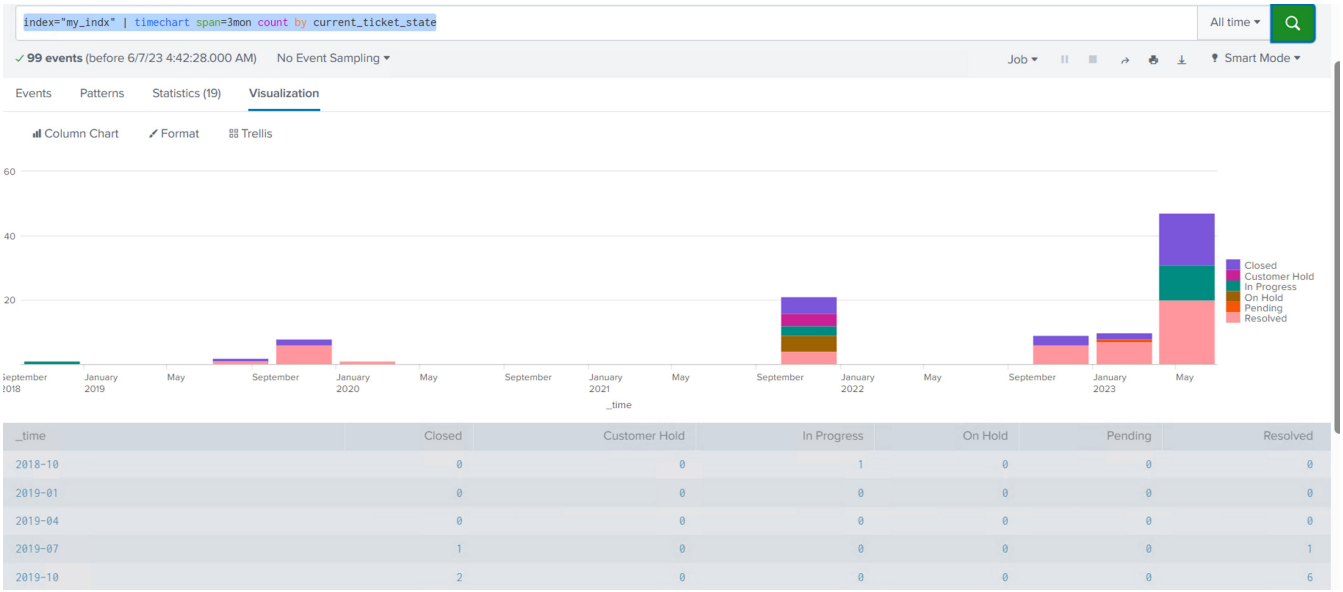

| _time | Closed | Customer Hold | In Progress | On Hold | Pending | Resolved |
|---|---|---|---|---|---|---|
| 2018-11 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2018-12 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2019-01 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2019-02 | 0 | 0 | 0 | 0 | 0 | 0 |

**Span**




| Timescale | Valid syntax | Description |
|---|---|---|
| <sec> | s \| sec \| secs \| second \| seconds | Time scale in seconds. |
| <min> | m \| min \| mins \| minute \| minutes | Time scale in minutes. |
| <hr> | h \| hr \| hrs \| hour \| hours | Time scale in hours. |
| <day> | d \| day \| days | Time scale in days. |
| <week> | w \| week \| weeks | Time scale in weeks. |
| <month> | mon \| month \| months | Time scale in months. |
| <subseconds> | us \| ms \| cs \| ds | Time scale in microseconds (us), milliseconds (ms), centiseconds (cs), or deciseconds (ds) |

***index="my_indx" | timechart span=3mon count by current_ticket_state***




| _time | Closed | Customer Hold | In Progress | On Hold | Pending | Resolved |
|---|---|---|---|---|---|---|
| 2018-10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2019-01 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2019-04 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2019-07 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2019-10 | 2 | 0 | 0 | 0 | 0 | 6 |

---

***Question*** : How to use X-axis as other timestamp value (time_submitted) instead of _time(default)

***Answer*** : ***By using eval*** to swap _time with time_submitted and then converting _time to ***EPOCH*** as after eval conversion splunk will not be able to recognize timestamp.

**Step #1** : Convert time_submitted (31-08-16 9:44) to epoch by using ***strptime()*** *and* ***eval()***

***Step #2 :*** Swap _time with time_submitted_epoch ***y using eval***()

**Step #3 :** Use **table** command to fetch fields

***Step #4 :*** use **timechart** now to print visualization

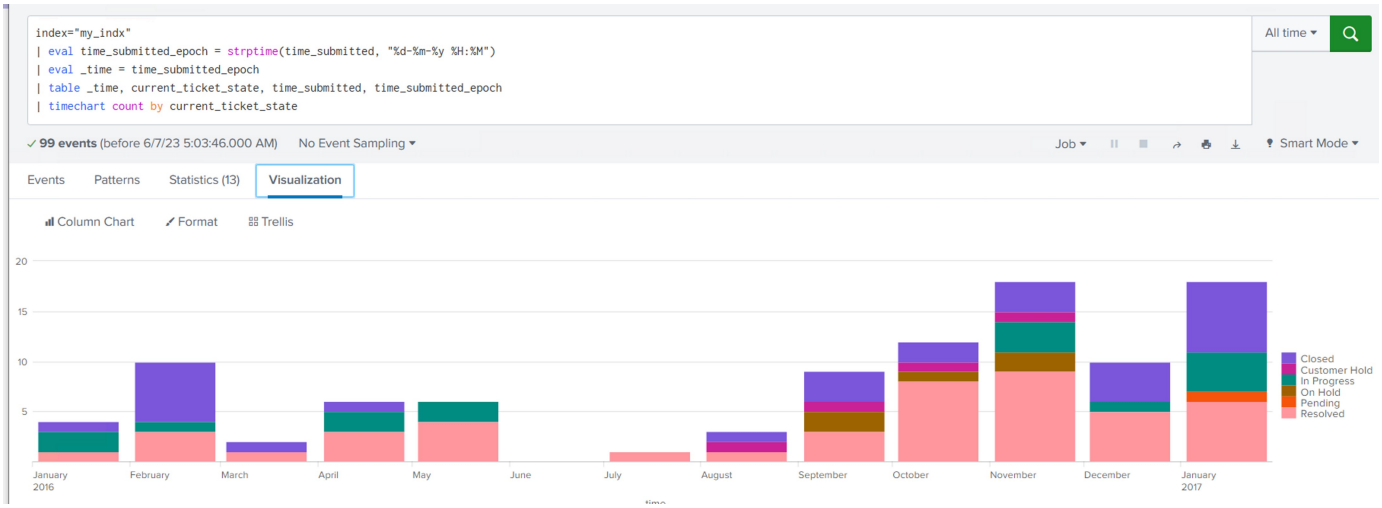
***index="my_indx"***

***| eval time_submitted_epoch = strptime(time_submitted, "%d-%m-%y %H:%M")***

***| eval _time = time_submitted_epoch***

***| table _time, current_ticket_state, time_submitted, time_submitted_epoch***

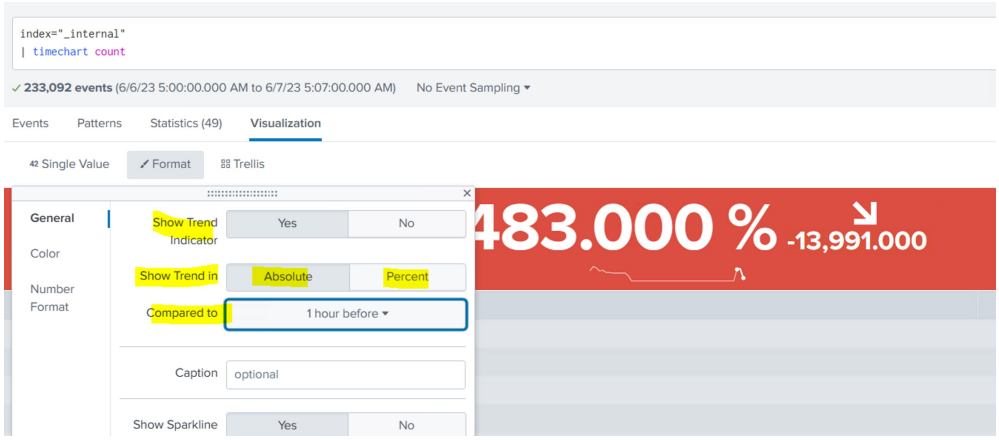
***| timechart count by current_ticket_state***

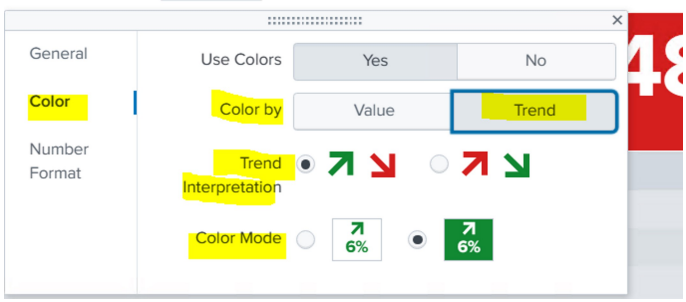


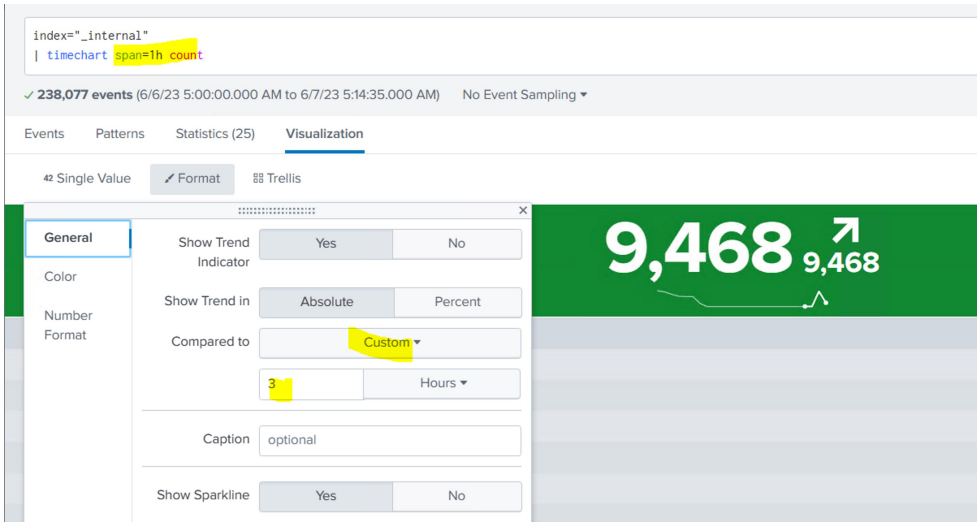

---

**#Trendline:**

Trendline will show trend compared to data of custom/specific time.




**#MAP:**

We need Latitude and Longitude values to pin any location on Map.

**source="vendors.csv" host="ip-172-31-84-138.ec2.internal" index="my_indx" sourcetype="csv"**

**| geostats latfield=VendorLatitude longfield=VendorLongitude count by VendorCountry**

Events Patterns **Statistics (2,988)** Visualization

20 Per Page ▾ Format Preview ▾ ‹ Prev 1 2 3 4 5 6 7 8 … Next ›

| geobin | Brazil | Canada | China (PRC) | France | Germany | India | Japan | OTHER | South Africa | United Kingdom | United States | latitude | longitude |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bin_id_zl_0_y_1_x_2 | | | | | | | | 1 | | | | -51.70778 | -57.82654 |
| bin_id_zl_0_y_2_x_2 | 2 | | | | | | | 1 | | | | -29.57282 | -56.64565 |
| bin_id_zl_0_y_2_x_3 | 2 | | | | | | | | | | | -22.88841 | -43.21506 |
| bin_id_zl_0_y_2_x_4 | | | | | | | | | 3 | | | -29.90986 | 25.61692 |

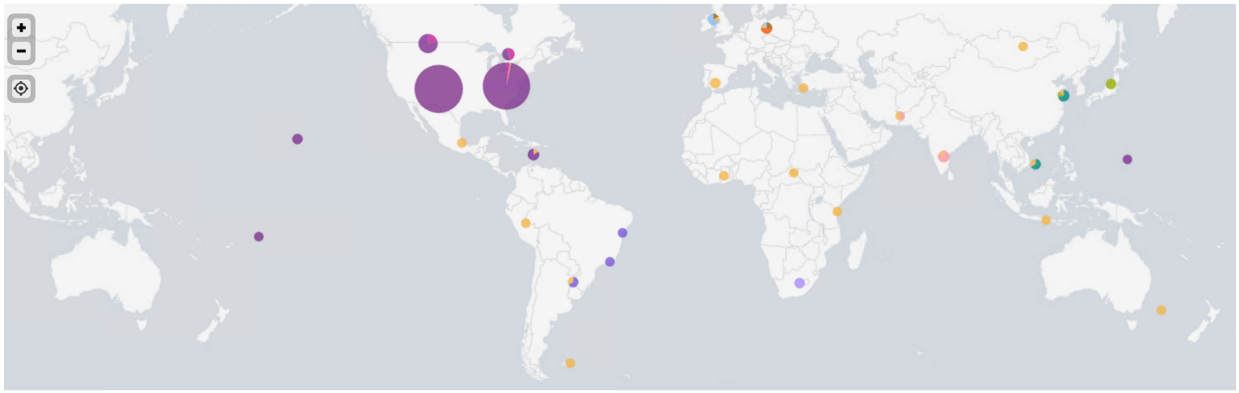
```
source="vendors.csv" host="ip-172-31-84-138.ec2.internal" index="my_indx" sourcetype="csv"
| geostats latfield=VendorLatitude longfield=VendorLongitude count by VendorCountry
```

⚠ Forced to skip results in geostats due to invalid latitude/longitude count='10'.

✓ **461 events** (before 6/7/23 5:52:23.000 AM) No Event Sampling ▾ ⚠ Job ▾

Events Patterns Statistics (2,988) **Visualization**

Cluster Map Format Trellis



**Default Geo Lookups** (in .**kmz** file ) comes with Splunk :

(1) geo_countries

(2) geo_us_states

## Lookup definitions

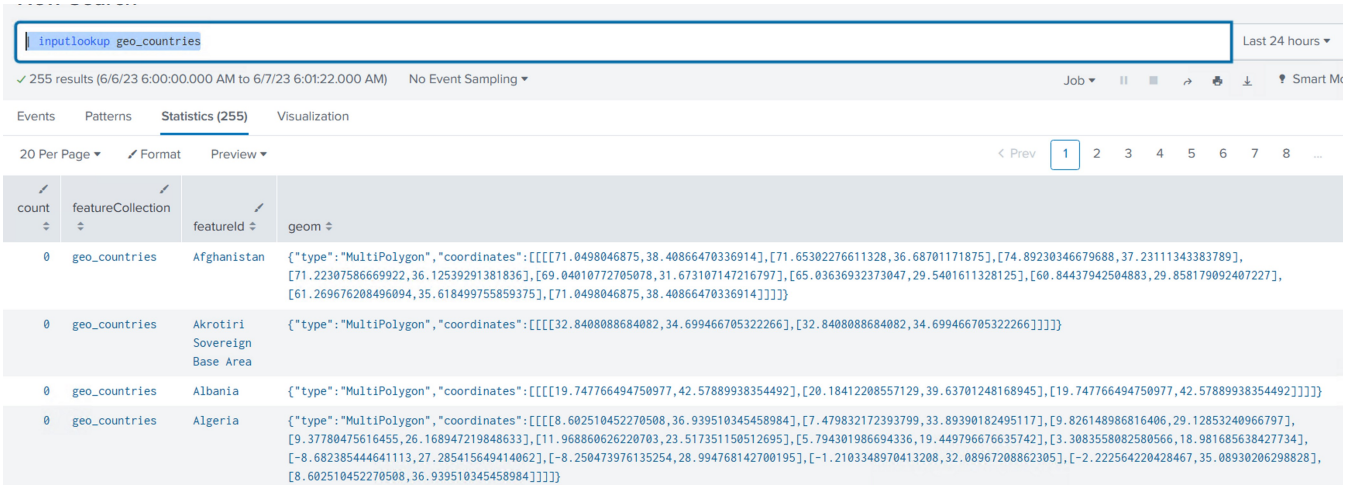
Lookups » Lookup definitions

Showing 1-4 of 4 items

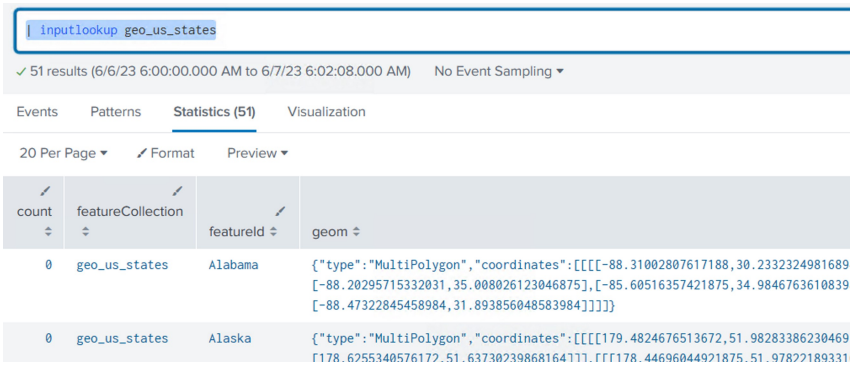
App Search & Reporting (s... ▾ Owner Any ▾ Visible in the App ▾ geo

| Name | Type | Supported fields | Lookup file | Owner | App | Sharing |
|---|---|---|---|---|---|---|
| geo_attr_countries | file | country,region_wb,region_un,subregion,continent,iso2,iso3 | geo_attr_countries.csv | No owner | search | Global \| Permissions |
| geo_attr_us_states | file | state_name,state_fips,state_code | geo_attr_us_states.csv | No owner | search | Global \| Permissions |
| geo_countries | geo | None | geo_countries.kmz | No owner | search | Global \| Permissions |
| geo_us_states | geo | None | geo_us_states.kmz | No owner | search | Global \| Permissions |

**| inputlookup geo_countries**




**| inputlookup geo_us_states**




---

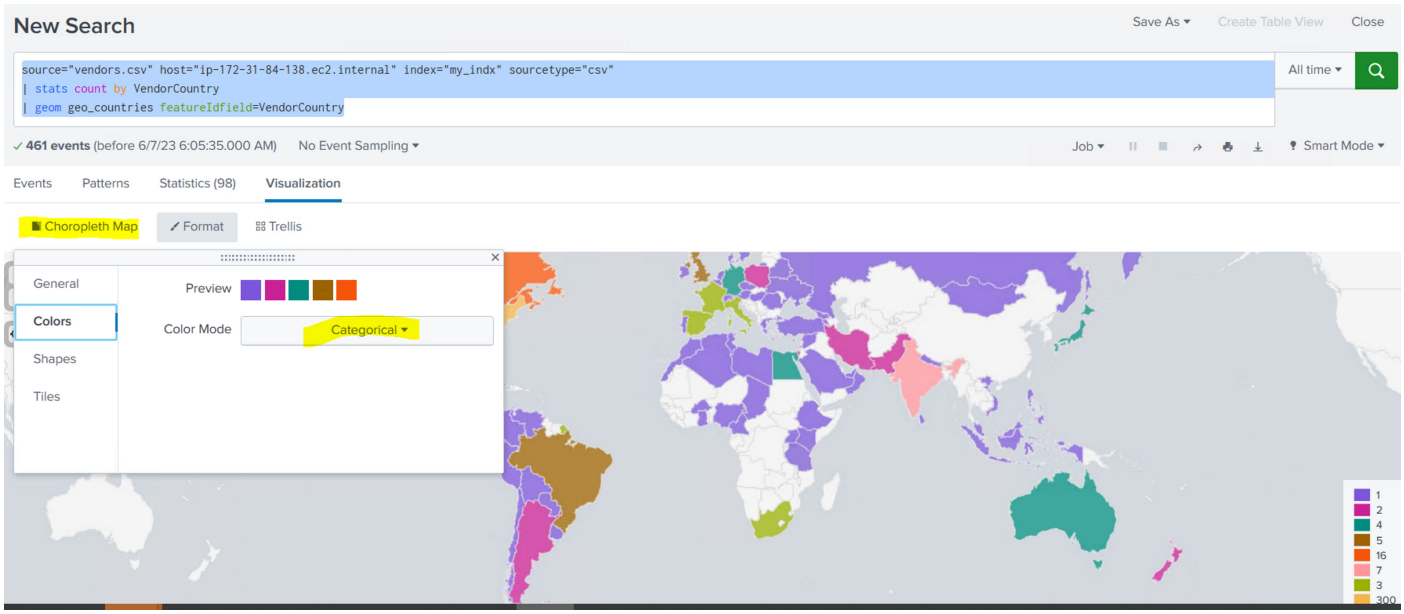
**Question** : Highlight countries based on their count in map. (**Geospatial data).**

**Answer** : using ***geom*** command and ***Choropleth*** map.

***source="vendors.csv" host="ip-172-31-84-138.ec2.internal" index="my_indx" sourcetype="csv"***

***| stats count by VendorCountry***

***| geom geo_countries featureIdfield=VendorCountry***




---

**Question** : How to convert kml file to kmz ?

**Answer** : Kml file is same as XML. Zip the kml file and give extension to .kmz

---

**#Application:**

Precheck before App installation :

(1) Space

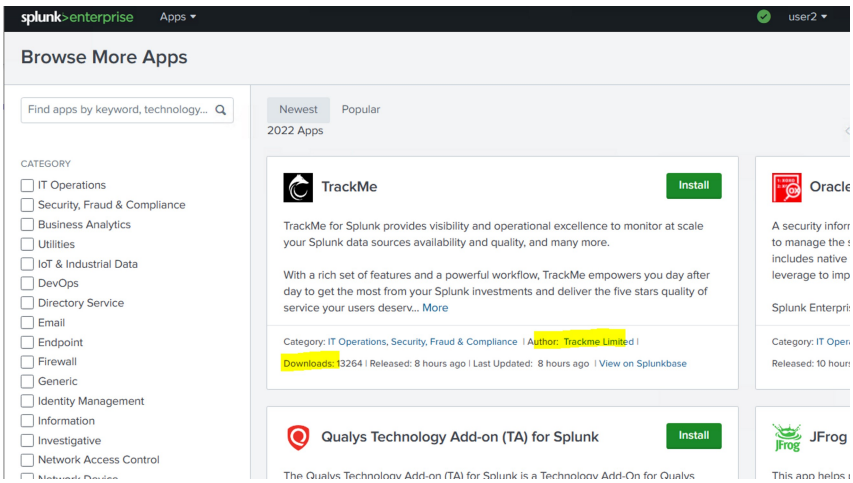
(2) Sufficient Resources

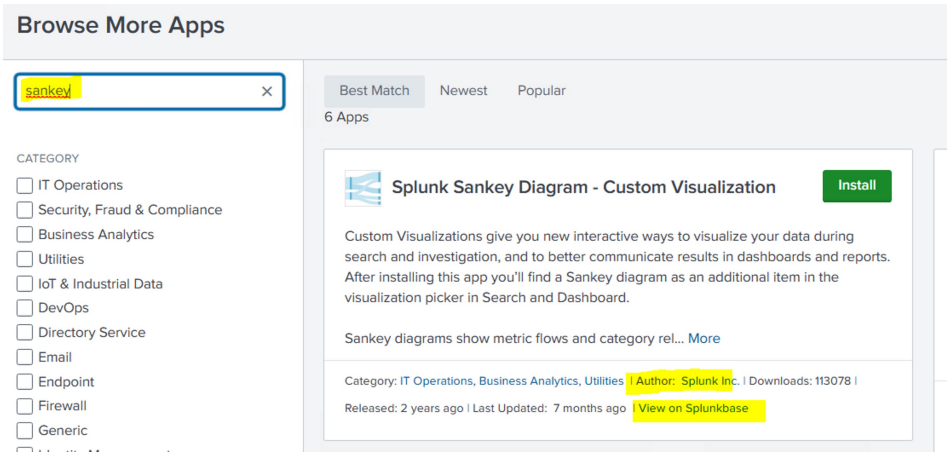
(3) Developer/**Author** (Splunk or 3rd party)

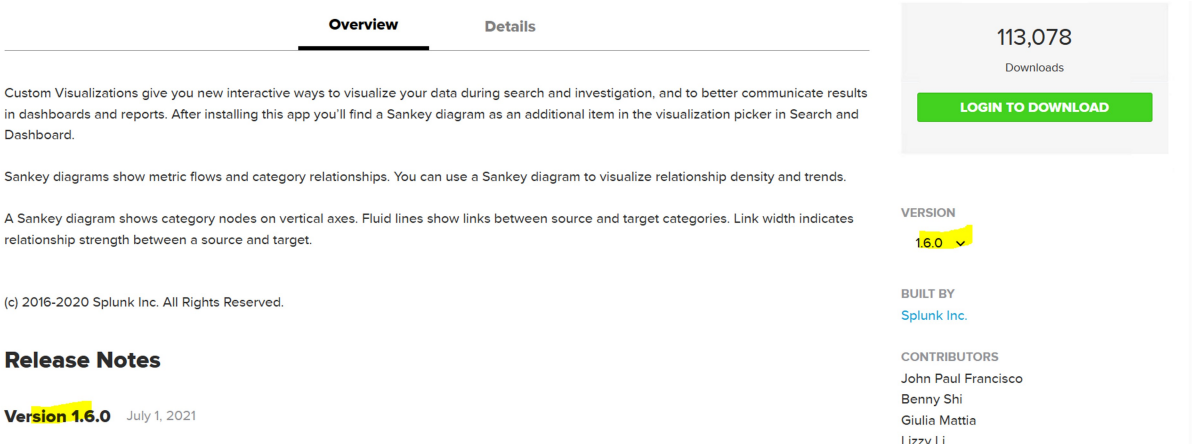
(4) Number of Downloads

(5) Out of date/Latest/Expiry

(6) Version of Splunk



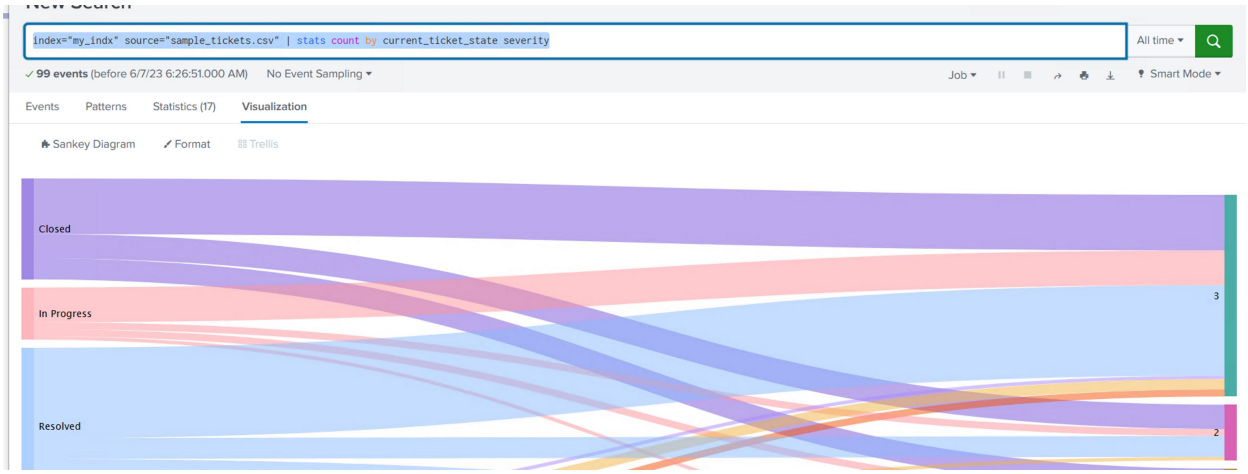


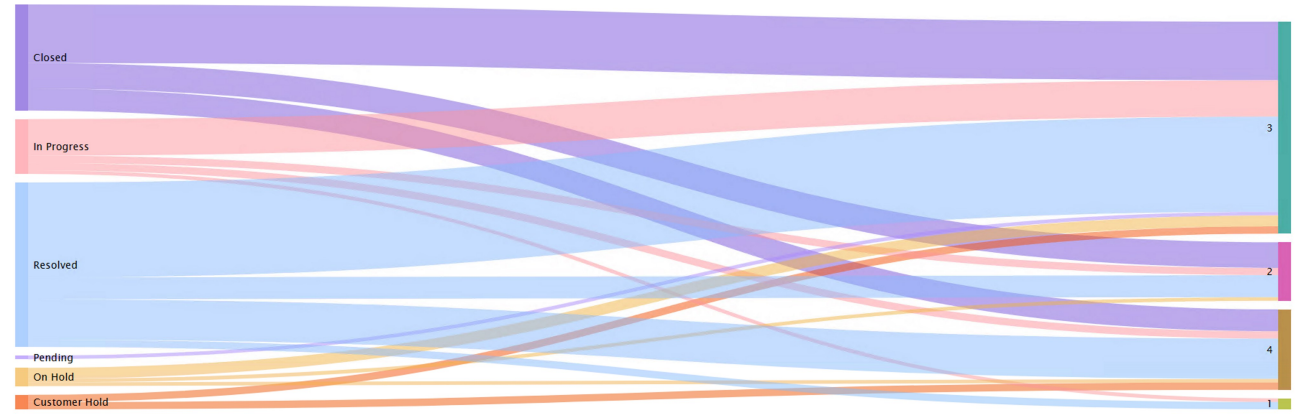




**#SanKey Diagram:**

SanKey diagram needs 2 values.

**index="my_indx" source="sample_tickets.csv" | stats count by current_ticket_state severity**






| current_ticket_state | severity | count |
|---|---|---|
| Closed | 2 | 7 |
| Closed | 3 | 16 |
| Closed | 4 | 6 |
| Customer Hold | 3 | 2 |
| Customer Hold | 4 | 2 |

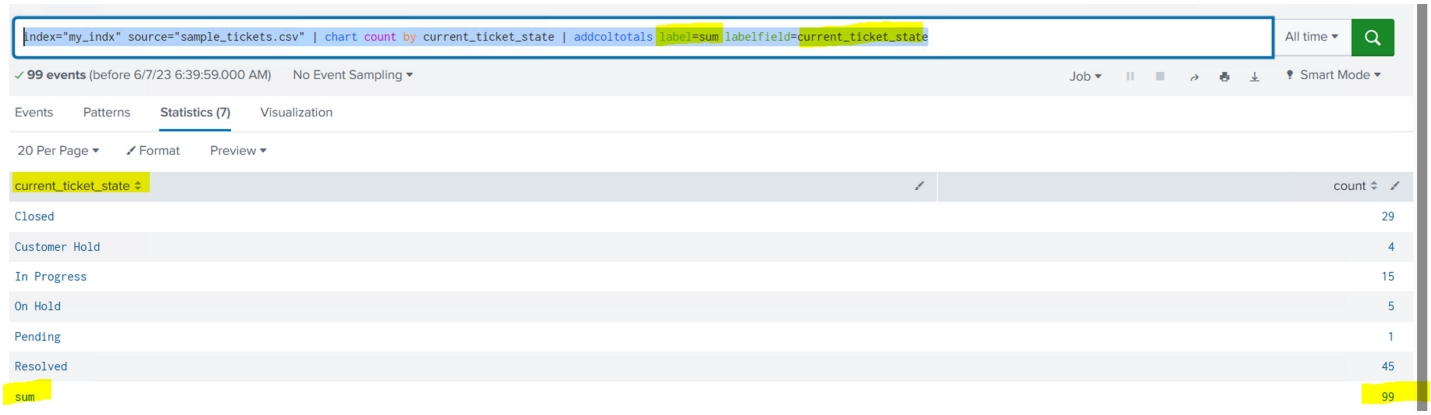
**#addcoltotals**

It does **Column-wise addition**.

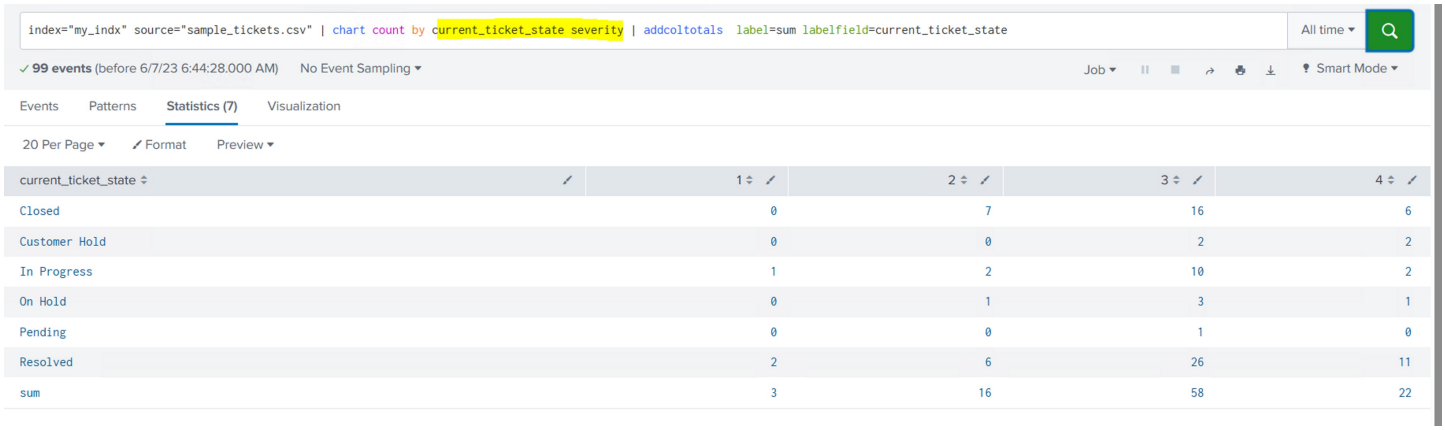
***index="my_indx" source="sample_tickets.csv" | chart count by current_ticket_state | addcoltotals label=sum labelfield=current_ticket_state***

***Label => what will be the name of total***

***Labelfield => after which field/column the label will be visible.***

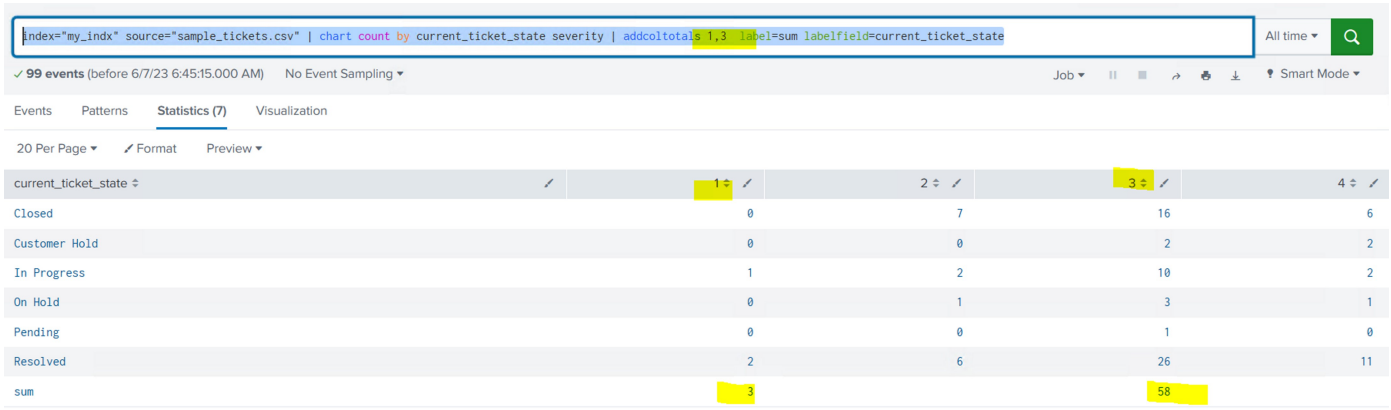



***index="my_indx" source="sample_tickets.csv" | chart count by current_ticket_state severity | addcoltotals label=sum labelfield=current_ticket_state***




| current_ticket_state | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Closed | 0 | 7 | 16 | 6 |
| Customer Hold | 0 | 0 | 2 | 2 |
| In Progress | 1 | 2 | 10 | 2 |
| On Hold | 0 | 1 | 3 | 1 |
| Pending | 0 | 0 | 1 | 0 |
| Resolved | 2 | 6 | 26 | 11 |
| sum | 3 | 16 | 58 | 22 |

***index="my_indx" source="sample_tickets.csv" | chart count by current_ticket_state severity | addcoltotals 1,3  label=sum labelfield=current_ticket_state***

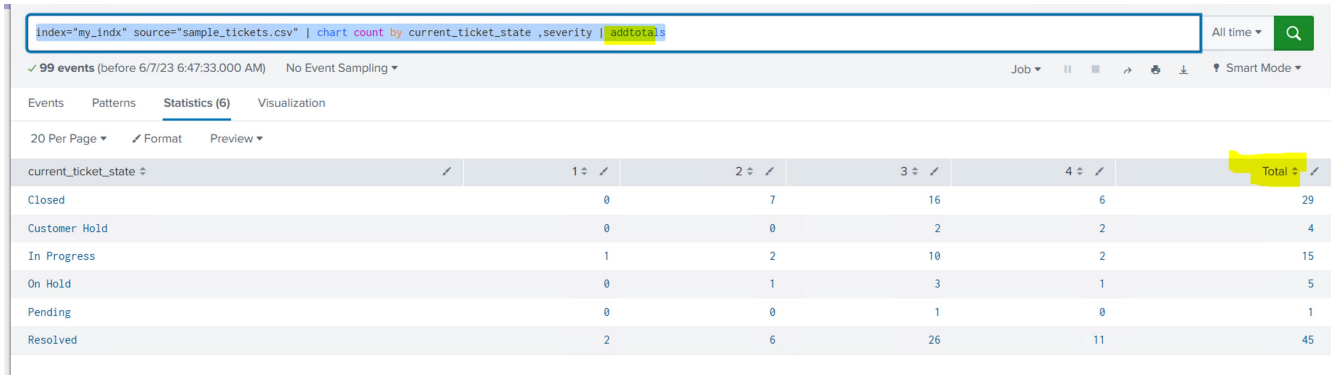



| current_ticket_state | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Closed | 0 | 7 | 16 | 6 |
| Customer Hold | 0 | 0 | 2 | 2 |
| In Progress | 1 | 2 | 10 | 2 |
| On Hold | 0 | 1 | 3 | 1 |
| Pending | 0 | 0 | 1 | 0 |
| Resolved | 2 | 6 | 26 | 11 |
| sum | 3 | | 58 | |

**#addtotals**

It does **Row-wise addition**.

***index="my_indx" source="sample_tickets.csv" | chart count by current_ticket_state ,severity | addtotals***




| current_ticket_state | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|
| Closed | 0 | 7 | 16 | 6 | 29 |
| Customer Hold | 0 | 0 | 2 | 2 | 4 |
| In Progress | 1 | 2 | 10 | 2 | 15 |
| On Hold | 0 | 1 | 3 | 1 | 5 |
| Pending | 0 | 0 | 1 | 0 | 1 |
| Resolved | 2 | 6 | 26 | 11 | 45 |

**#Rex**

**Useful websites :**

https://regex101.com/

https://www.debuggex.com/cheatsheet/regex/pcre

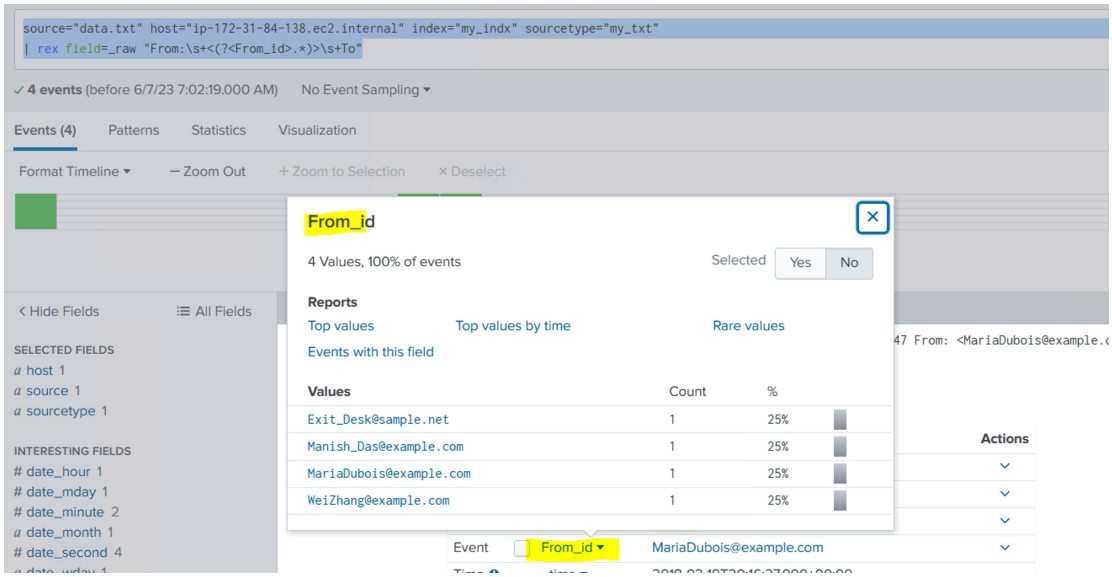
2 types of searches:

(1) **Index-time search** => at time of processing and before ingestion

(2) **Search-time search** => after ingestion

***source="data.txt" host="ip-172-31-84-138.ec2.internal" index="my_indx" sourcetype="my_txt"***

***| rex field=_raw "From:\s+<(?<From_id>.*)>\s+To"***



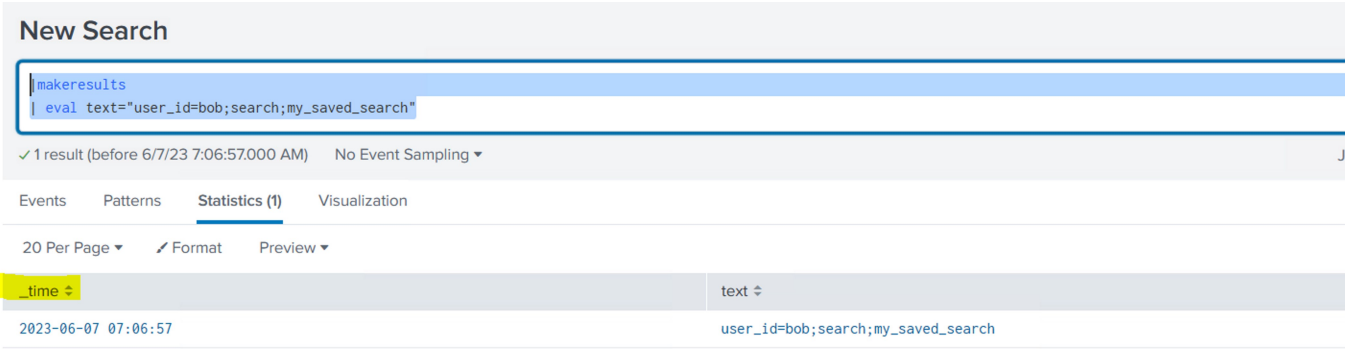

**#makeresults**

It create sample results.

**_time column will be added by default.**

***|makeresults***

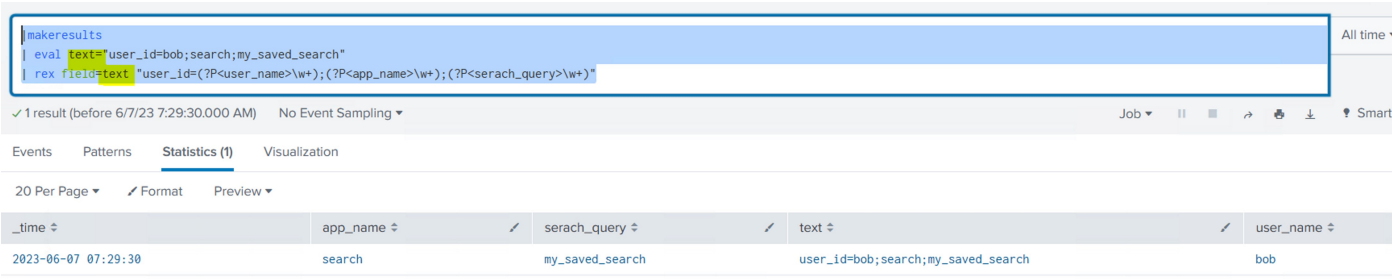
***| eval text="user_id=bob;search;my_saved_search"***




***|makeresults***

***| eval text="user_id=bob;search;my_saved_search"***

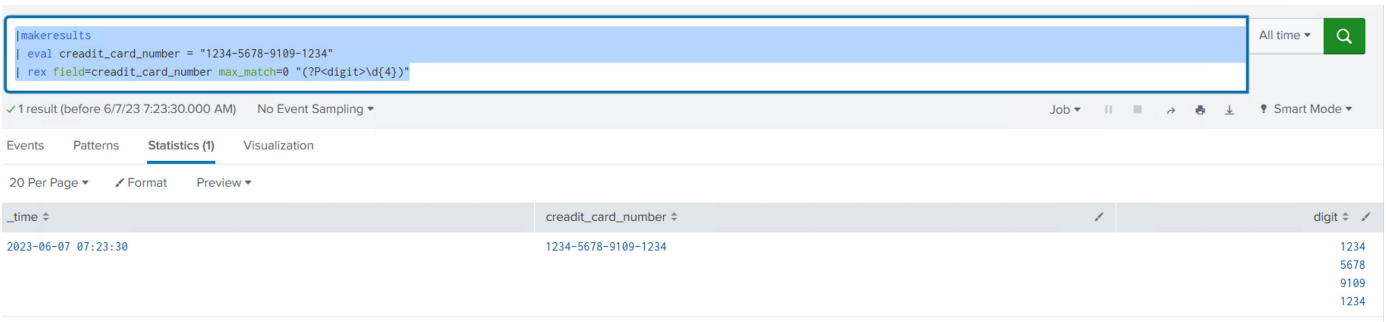
***| rex field=text "user_id=(?P<user_name>\w+);(?P<app_name>\w+);(?P<serach_query>\w+)"***




**|makeresults**

**| eval creadit_card_number = "1234-5678-9109-1234"**

**| rex field=creadit_card_number max_match=0 "(?P<digit>\d{4})"**




**|makeresults**

**| eval creadit_card_number = "1234-5678-9109-1234"**

**| rex field=creadit_card_number max_match=0 offset_field="new_field" "(?P<digit>\d{4})"**