Training – Day #4

15 June 2023 04:33

#Knowledge-Objects

A user-defined entity that enriches the existing data in the Splunk platform.

You can use knowledge objects to get specific information about your data.

When you create a knowledge object, you can keep it private or you can share it with other users.

Knowledge managers manage how their organizations use knowledge objects in their Splunk Enterprise deployments.

Splunk Enterprise knowledge objects include **saved searches, event types, tags, field extractions, lookups, reports, alerts, data models, workflow actions, and fields.**

From <https://docs.splunk.com/Splexicon:Knowledgeobject>

When you run a search, Splunk software runs several operations to derive knowledge objects and apply them to events returned by the search. **Splunk software performs these operations in a specific sequence.**

https://docs.splunk.com/Documentation/Splunk/9.0.4/Knowledge/Searchtimeoperationssequence

Search-time operation order	Operation name	Configurable in Splunk Web?	Location of file configuration
1	Role-based field filtering	No	`fieldFilter-<fieldname>` in a stanza in the authorize.conf file.
2	Inline field extraction (no field transform)	Yes	`EXTRACT-<class>` in a stanza in the props.conf file.
3	Field extraction that uses a field transform	Yes	`REPORT-<class>` in a stanza in the props.conf file.
4	Automatic key-value field extraction	No	In stanzas in the props.conf file, where `KV_MODE` is set to a valid value other than `none`. If no `KV_MODE` value is specified for a stanza, it is set to `auto` by default.
5	Field aliasing	Yes	`FIELDALIAS-<class>` in a stanza in the props.conf file.
6	Calculated fields	Yes	`EVAL-<fieldname>` in a stanza in the props.conf file.
7	Lookups	Yes	`LOOKUP-<class>` in a stanza in the props.conf file.
8	Event types	Yes	In a stanza in the eventtypes.conf file.
9	Tags	Yes	In a stanza in the tags.conf file.

#Calculated Fields:

Calculated fields are fields added to events at search time that **perform calculations with the values of two or more fields already present in those events.** Use calculated fields as a shortcut for performing repetitive, long, or complex transformations using the eval command.

https://docs.splunk.com/Documentation/Splunk/9.0.4/Knowledge/definecalcfields#:~:text=Calculated%20fields%20are%20fields%20added,transformations%20using%20the%20eval%20command.

We can **create a Template where we can define eval expression only once** and we have to attach that KB when any requirement is needed.

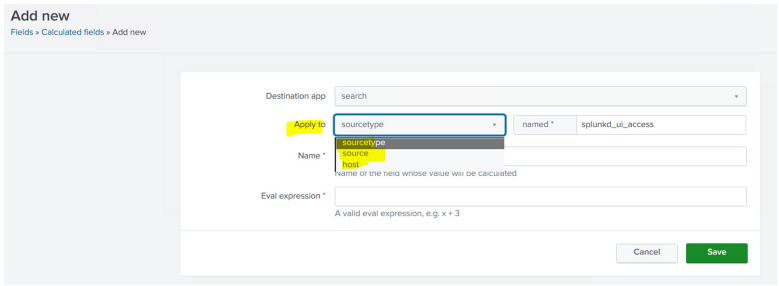
So if any change is required in expression, we have to do only on one place i.e. Template.

We can **apply Template on dataset based on (to be more precise on its attachment)** :

(1) source

(2) sourcetype

(3) host

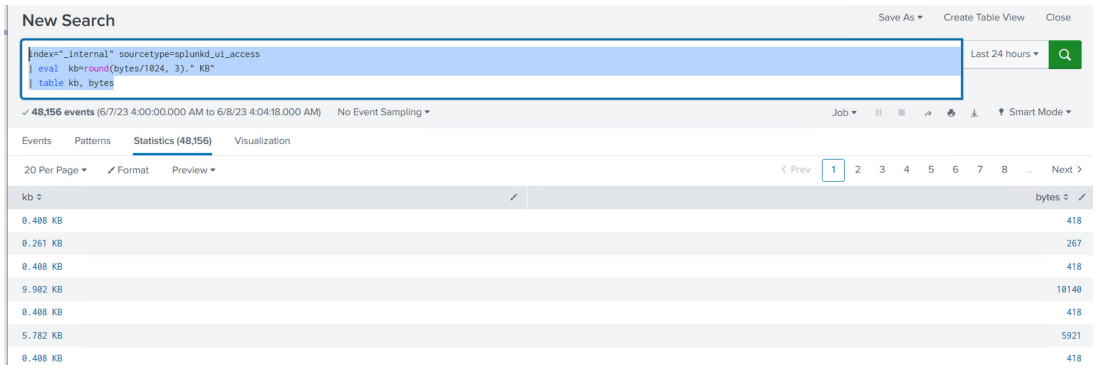


Without Calculated Field :

index="_internal" sourcetype=splunkd_ui_access

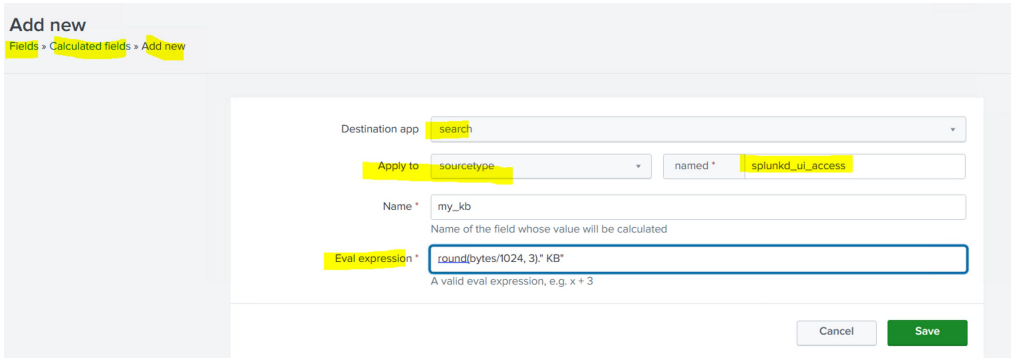
| eval kb=round(bytes/1024, 3)." KB"

| table kb, bytes

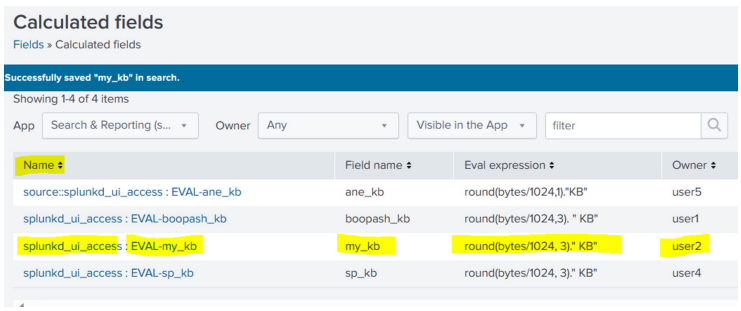


Create Calculate Filed:

Settings => Knowledge => Fields => Calculated Fields => New Calculated Fields



Name of Calculated Fields will be => ***<sourcetype/host/source>:EVAL-<Calculated Filed Name>***

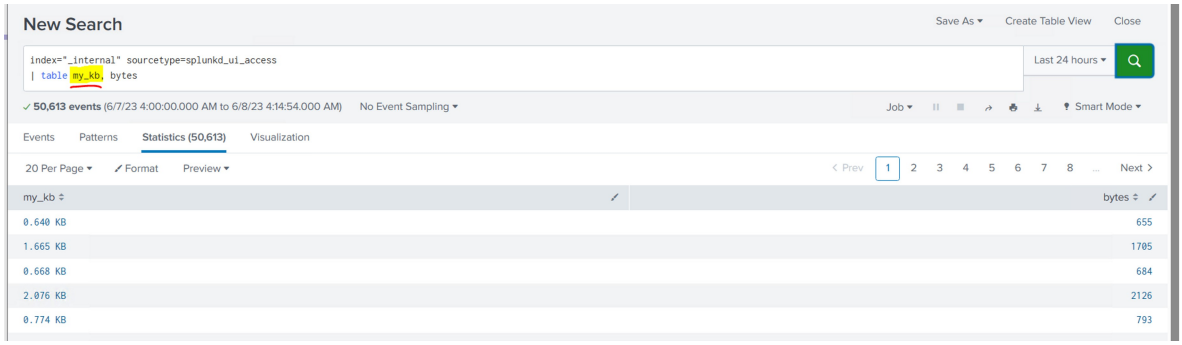


Search with template of created Calculated Fields:

index="_internal" sourcetype=splunkd_ui_access

| table my_kb, bytes

Note : No eval line is needed.

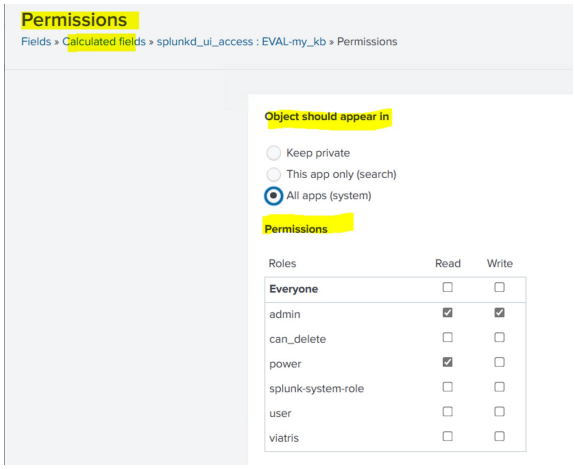


Permissions of Calculated Fields :

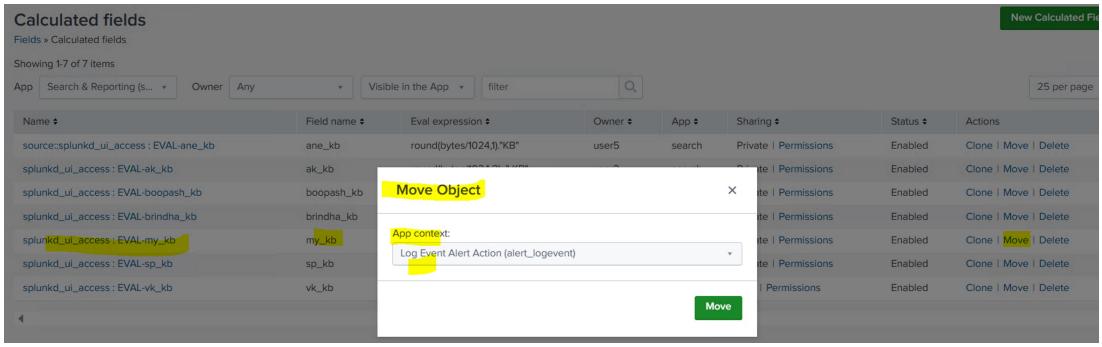
(1) ***Keep private*** => Only creator can use and edit + ***Default settings***

(2) ***This app only (search)*** => ***Give Permissions Based on Role + KB can be accessible from only search application***

(3) ***All apps (system) =>*** => ***Give Permissions Based on Role + KB can be accessible from all application***



Move => to move from one application to other application



#Tags:

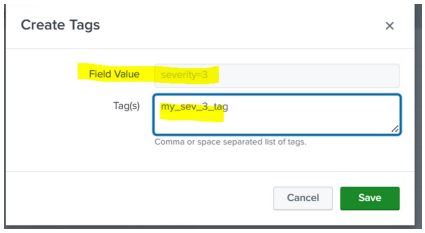
https://docs.splunk.com/Documentation/Splunk/9.0.4/SearchReference/Tags

Tag is used to **Categorize** or **Identify** something.

In Splunk, we add **Tags** to **"Field Value".**

How to create Tags:

Open Event Filed => Go to Field => Edit Tags=>



Run Query :

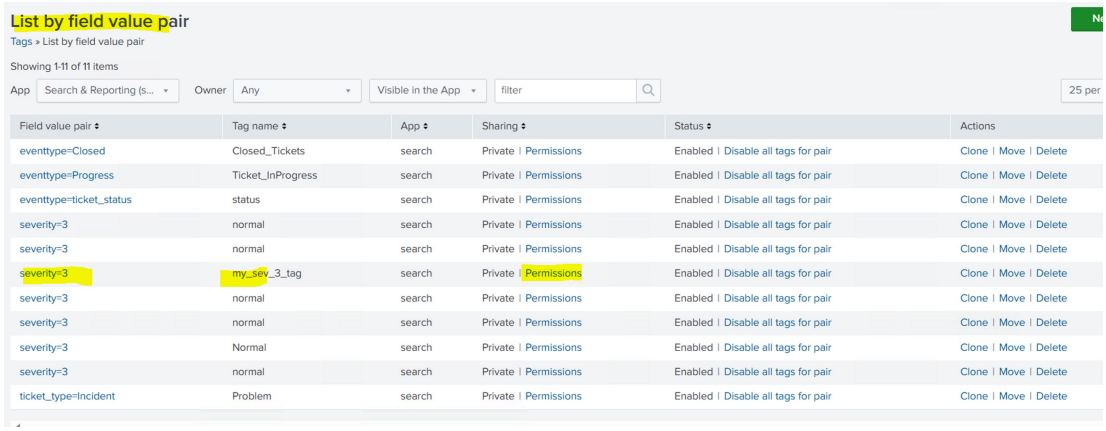
index="my_indx" source="sample_tickets.csv" tag::severity="my_sev_3_tag"

2 Field will be auto generated -

(1) tag

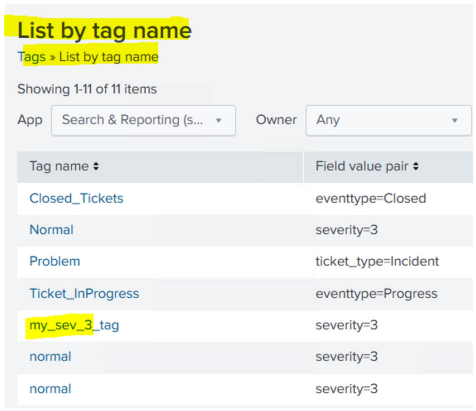
(2) tag-<field name>

Permissions:



How to Edit Condition of Tags:

Settings => Tags => List by tag name => click "Tag name" => Edit field value condition



Tag name	Field value pair
Closed_Tickets	eventtype=Closed
Normal	severity=3
Problem	ticket_type=Incident
Ticket_InProgress	eventtype=Progress
my_sev_3_tag	host=123, severity=3

#EventType:

https://docs.splunk.com/Documentation/Splunk/9.0.4/Knowledge/Abouteventtypes

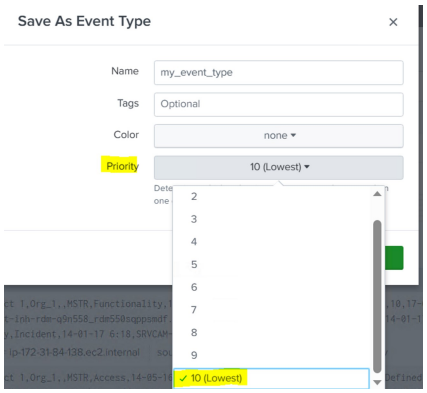
(1) Condition of **Event Categorization based on "Search Query"**

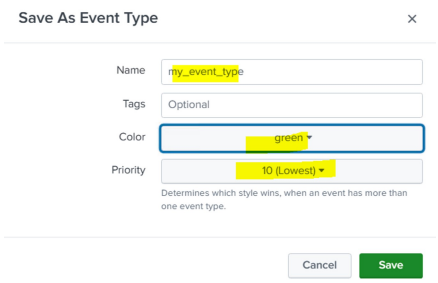
How to create Event-Type:

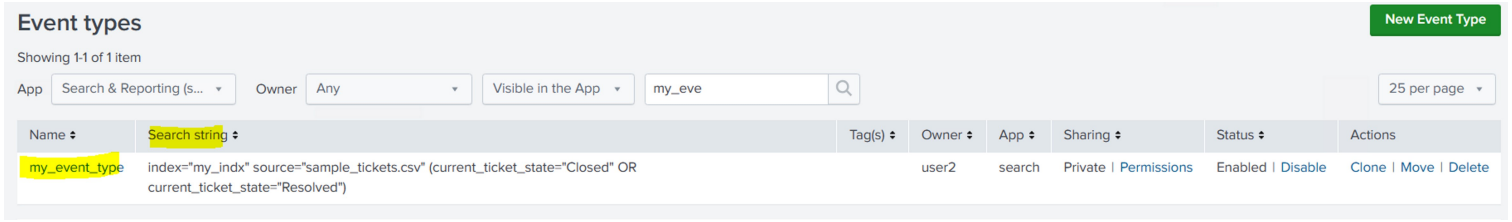
Step #1 : Run your SPL :

index="my_indx" source="sample_tickets.csv" (current_ticket_state="Closed" OR current_ticket_state="Resolved")

Step #2: Save as "Event Type"



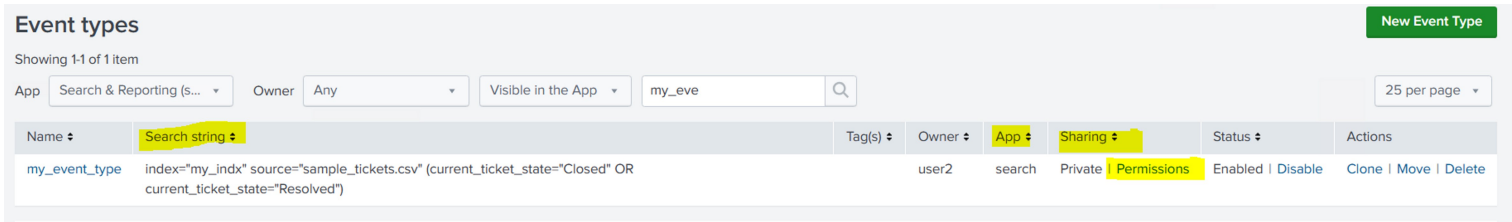




Step #3 : Use Event Type to see the results

eventtype="my_event_type"

Permissions:



#Lookups:

https://docs.splunk.com/Documentation/Splunk/9.0.4/Knowledge/Aboutlookupsandfieldactions

Lookup is a small static file.

Lookups enrich your **event data** by adding field-value combinations from lookup tables.

4 Type of lookup files :

(1) CSV lookup

(2) KV store lookup

(3) Geospatial lookup

(4) External lookup

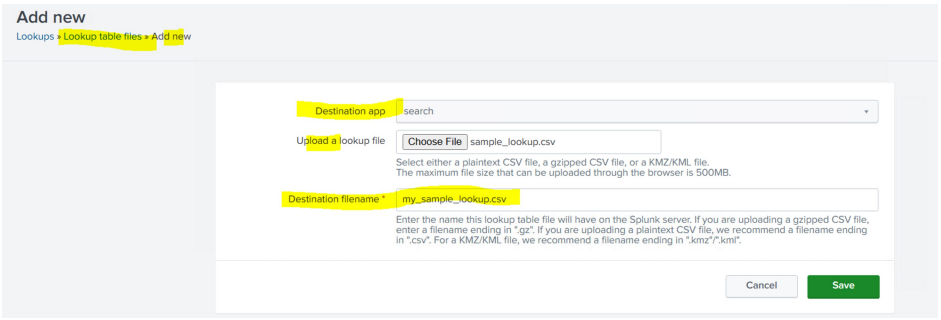
No license consumption as no index consumption. It will consume disk space on server.

How to use lookups?

Step #1 : Upload lookup in Splunk.

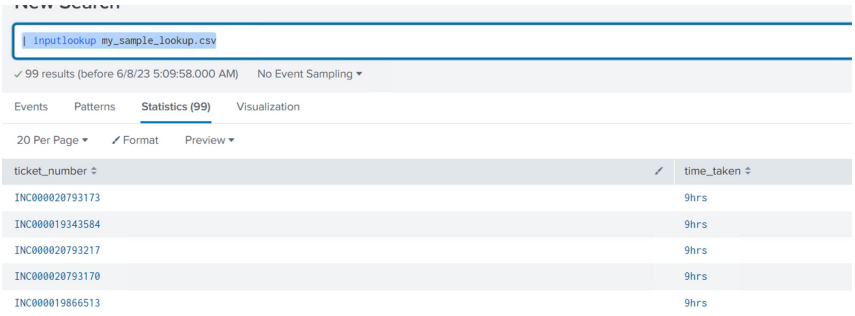
Settings => Lookups => Lookup Table files => Add =>

Note : Don't forget to put ".csv" in end of filename "Destination filename".



Step #2 : List lookup by "|inputlookup" command

|inputlookup my_sample_lookup.csv



Step #3 : Use lookup to enrich event by **2 way**s-

(1) using join

(2) using Lookup Definition => **More faster** than Join

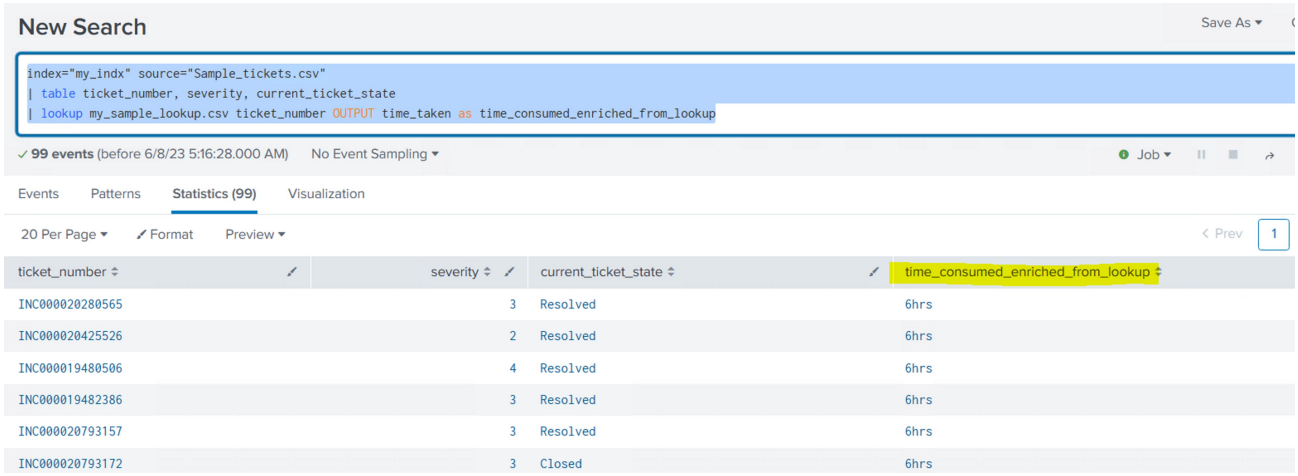
Note : If field name are not same, then use "**rename**" command to create similar name.

(1) using join

index="my_indx" source="Sample_tickets.csv"

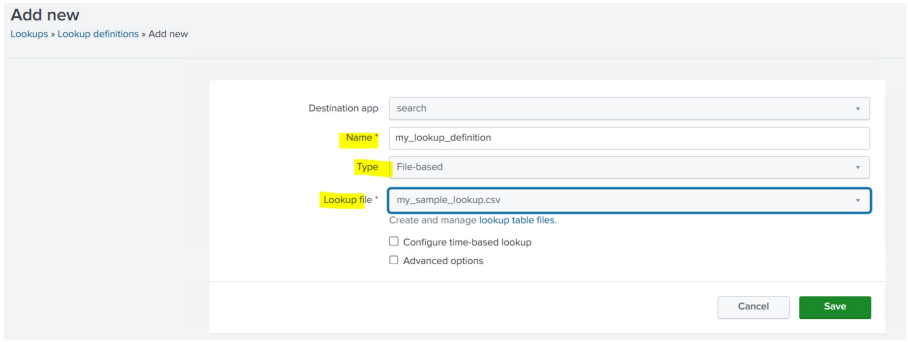
| table ticket_number, severity, current_ticket_state

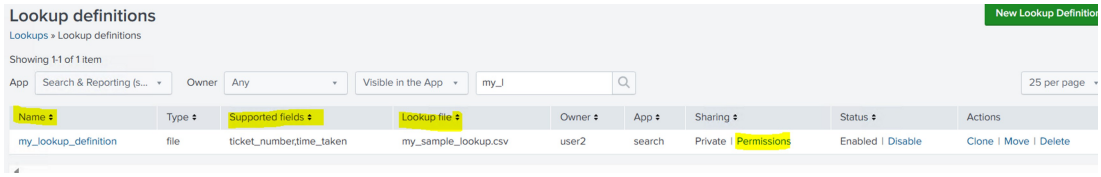
| lookup my_sample_lookup.csv ticket_number OUTPUT time_taken as time_consumed_enriched_from_lookup



(2) using lookup definitions

Settings => Lookups => Lookup Definitions => Add New

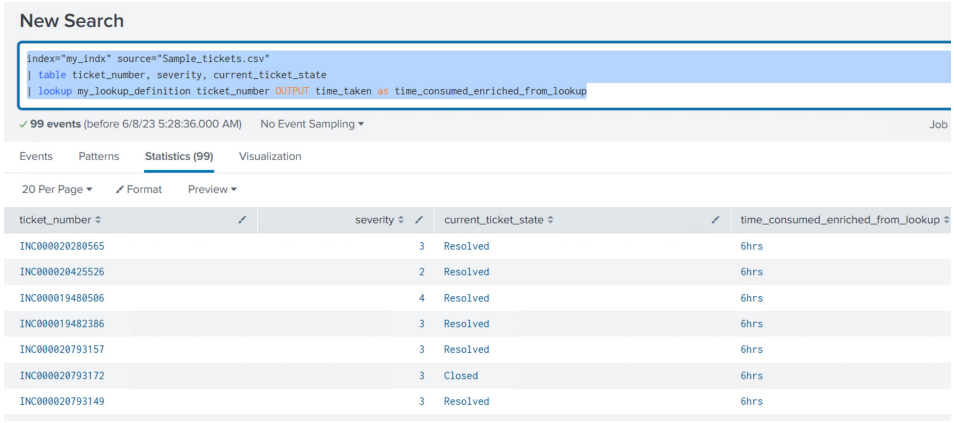




index="my_indx" source="Sample_tickets.csv"

| table ticket_number, severity, current_ticket_state

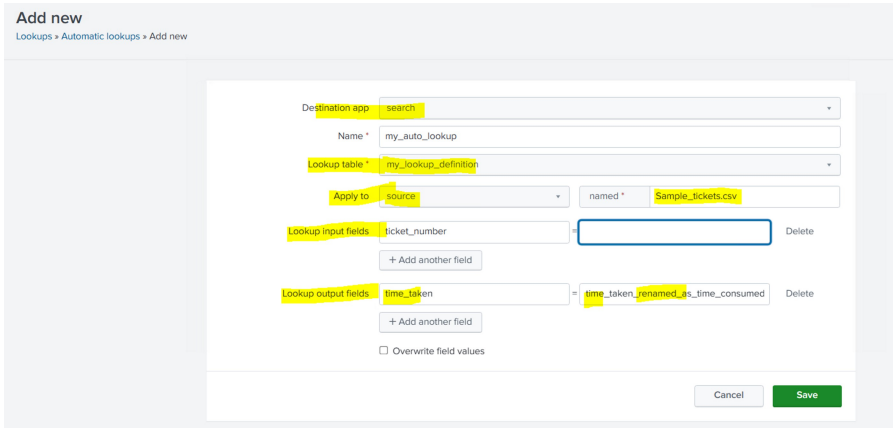
| lookup my_lookup_definition ticket_number OUTPUT time_taken as time_consumed_enriched_from_lookup



#Automatic Lookup

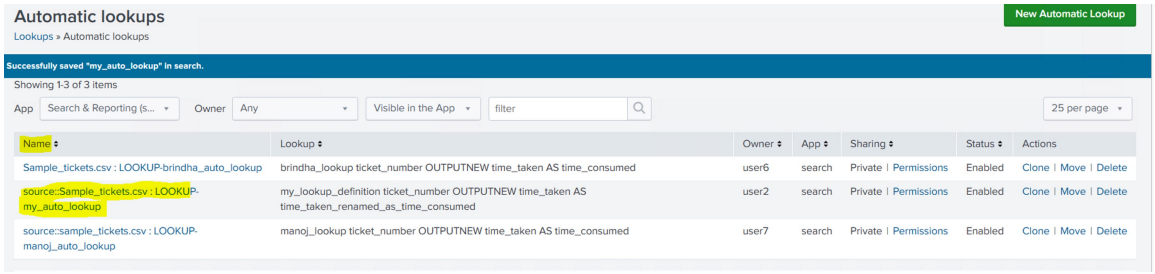
Settings => Lookups => Automatic Lookup => Add New

Lookup input fields <field name of lookup> = <field name of index, need to mention if name is not same>



Name of automatic lookup => **<Apply to>::<Apply to Value>: LOOKUP -<auto lookup name>**

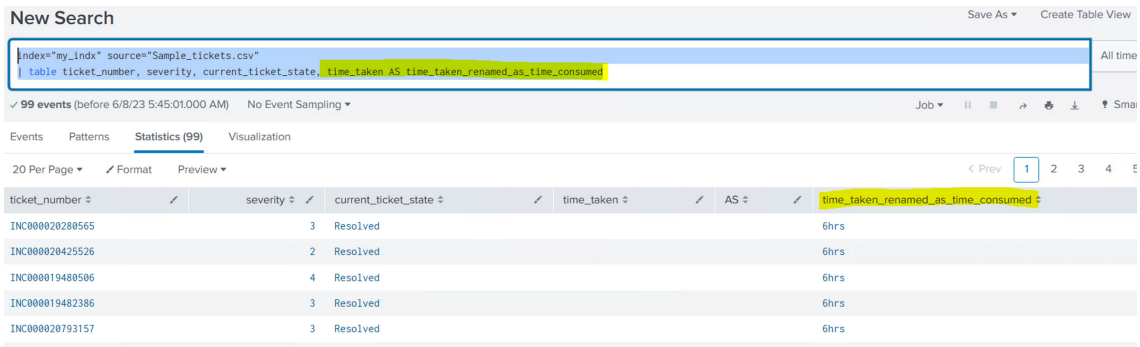
It will create **lookup command** with **OUTPUTNEW.**



Step #3: use the output field

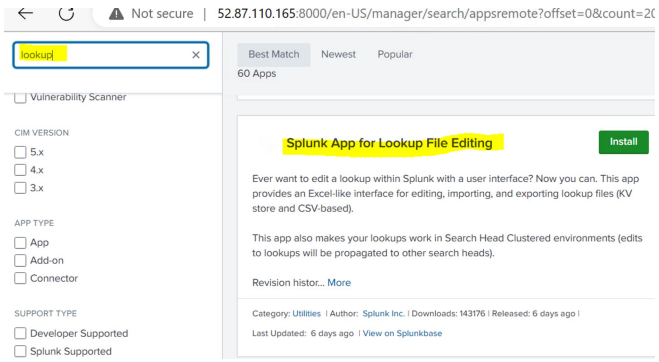
index="my_indx" source="Sample_tickets.csv"

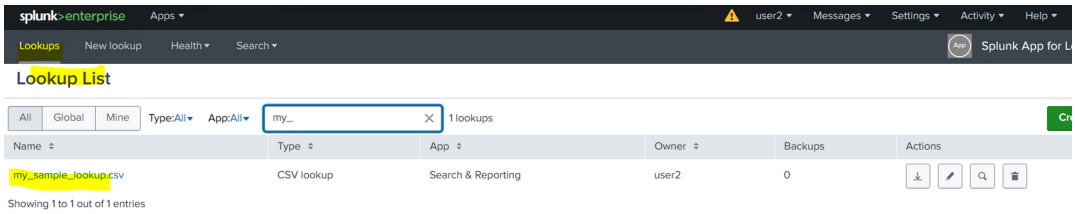
| table ticket_number, severity, current_ticket_state, time_taken AS time_taken_renamed_as_time_consumed

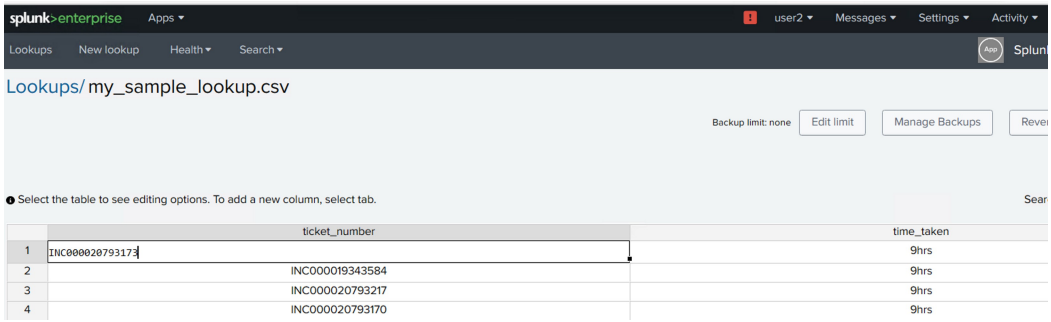


#Lookup Editor Application

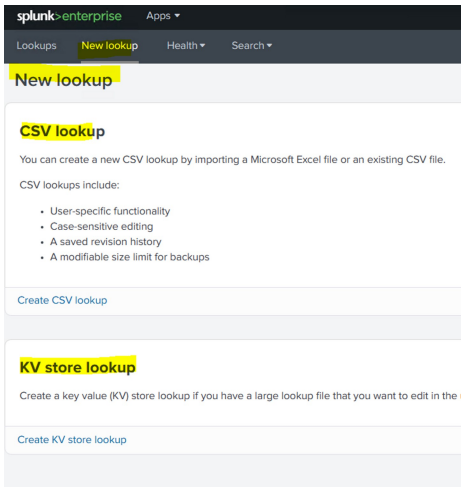
This application will make very easy to delete/modify any lookup file.

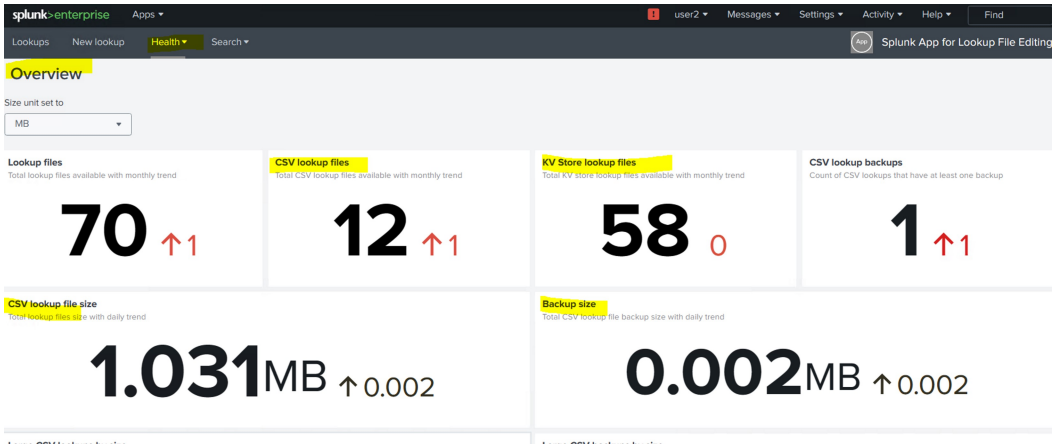


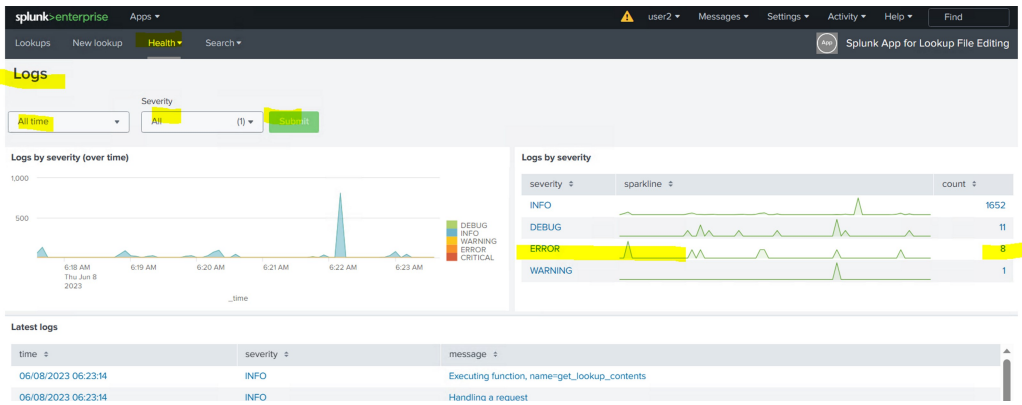




Create 2 kinds of lookup (1) CSV and (2) KV store







#outputlookup command

It is used to push data in an existing lookup.

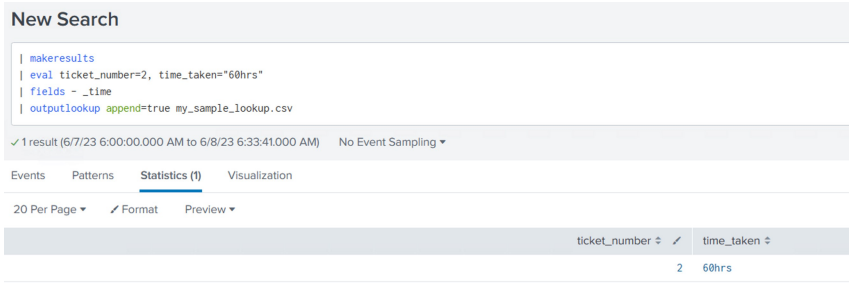
Default is override the value. So use append=true

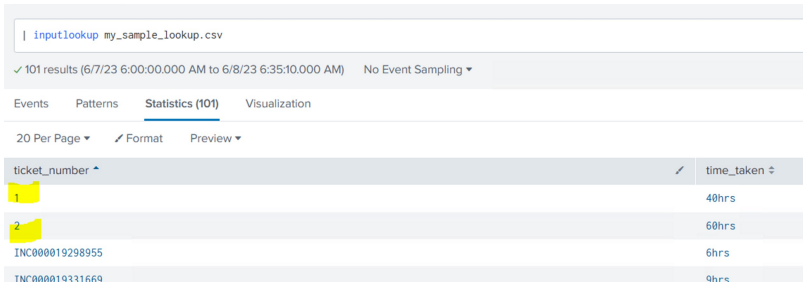
| makeresults

| eval ticket_number=1, time_taken="40hrs"

| fields - _time

| outputlookup append=true my_sample_lookup.csv





#KV store lookup

Key Value paired

Used to store **Big files** which are **Dynamic** in nature.

Values are in Stack.

#Macros

Macros are like a function.

like define a function

Call a function

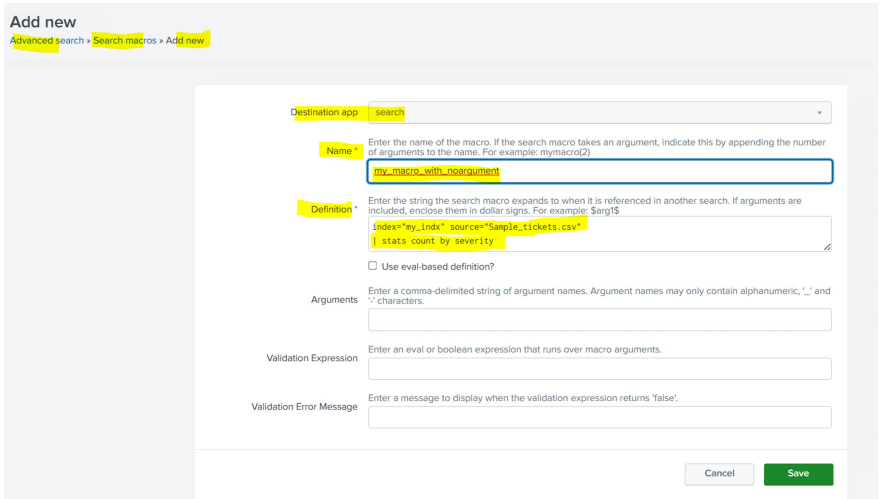
- Macro with no argument
- Macro with single argument
- Macro with multiple argument

How to create a macro ?

Settings => Advance search => Search macros => Add new

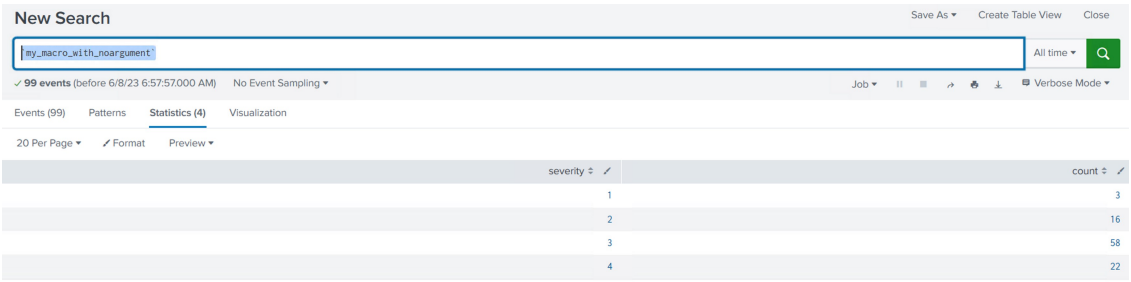
index="my_indx" source="Sample_tickets.csv"
| stats count by severity

Macro with no argument



Call a macro of no argument with tilde symbol `` :

`my_macro_with_noargument`

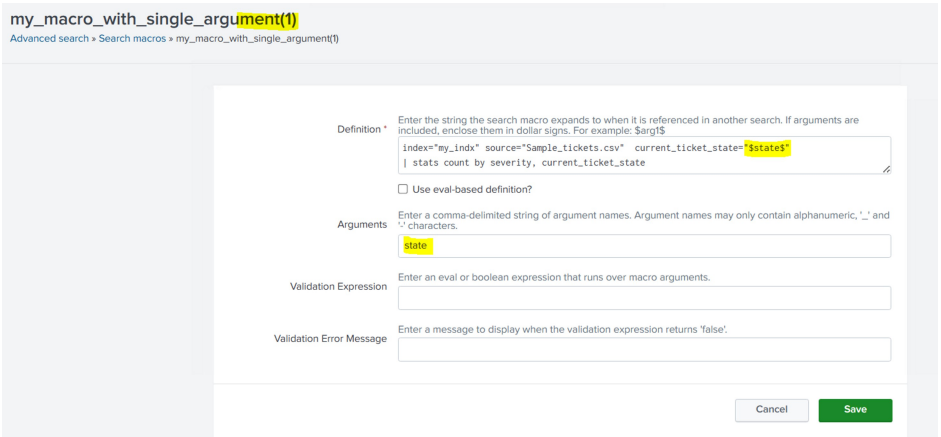


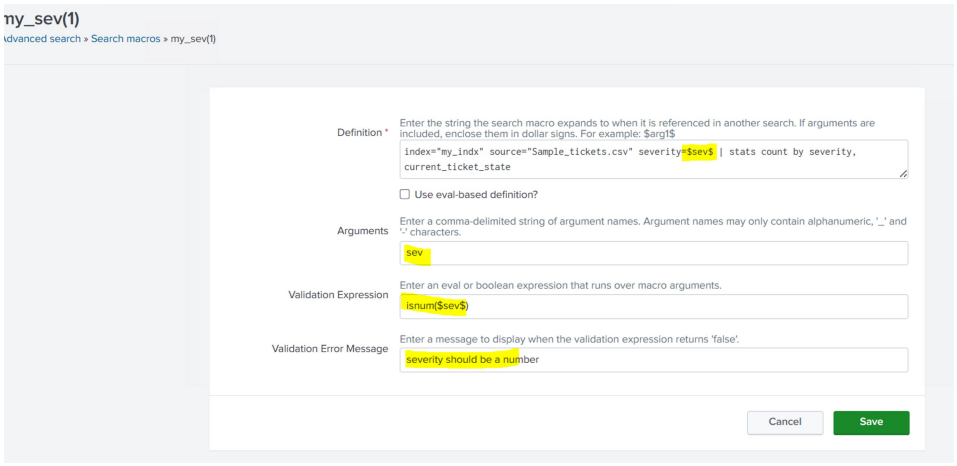
`my_macro_with_noargument` | search severity=2

Macro with single argument

Note: arguments are inside ***$$***

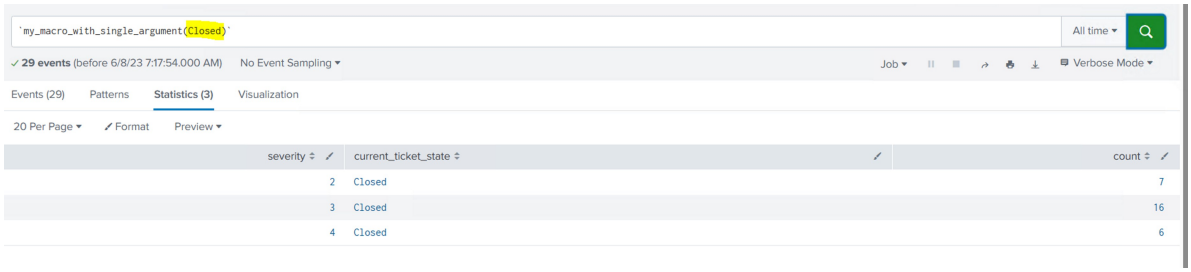
index="my_indx" source="Sample_tickets.csv" current_ticket_state="Closed"
| stats count by severity, current_ticket_state



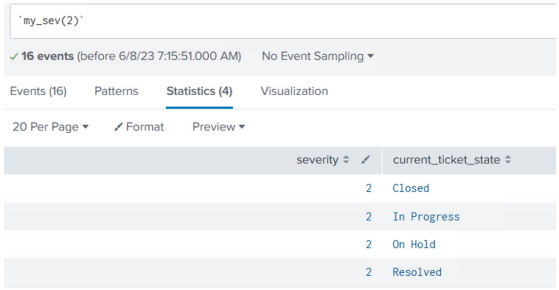


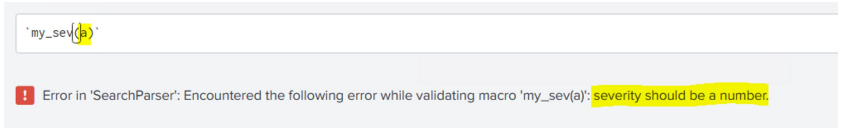
Call a macro of single argument

`my_macro_with_single_argument(Closed)`

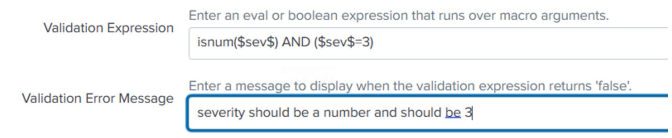


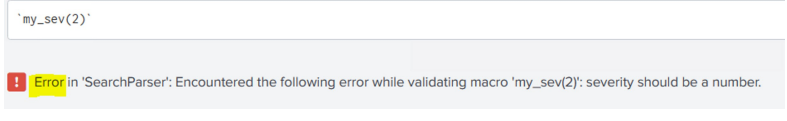
`my_sev(2)`





Validation : Isnum(sev) AND (sev >0) AND (sev<5)

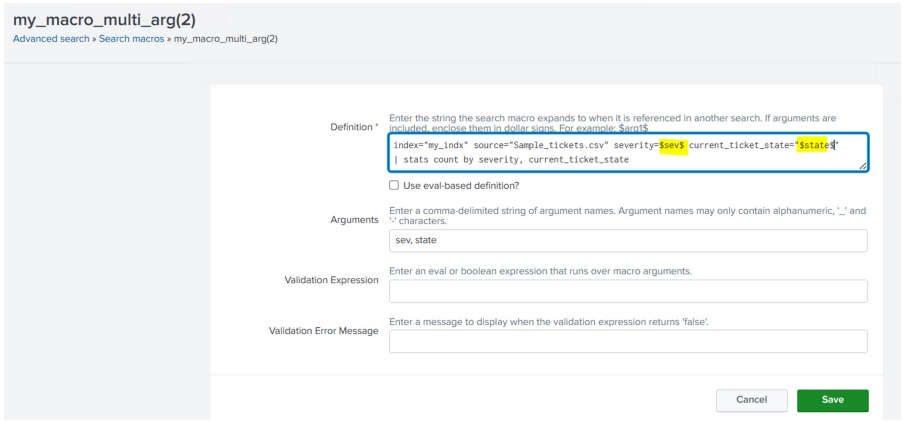




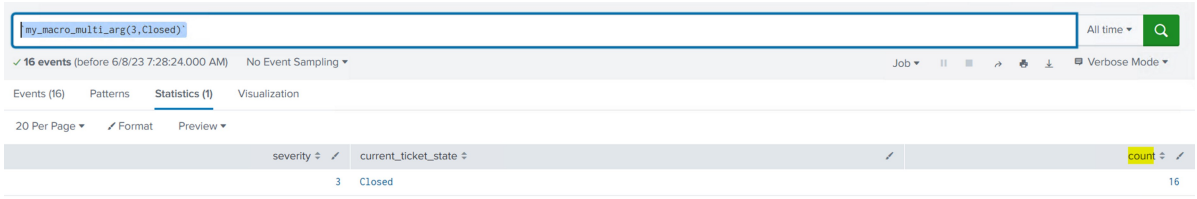
Macro with multiple argument

index="my_indx" source="Sample_tickets.csv" severity=3 current_ticket_state="Closed"
| stats count by severity, current_ticket_state

my_macro_multi_arg(2)



`my_macro_multi_arg(3,Closed)`



Changing the argument order in "**Arguments**" :