Hands-On Splunk Lab Report

*Note: Splunk, Visualize, Analyze, and Initalize data.

Andrew Michael Flores
B.B.A – Cyber Security
University of Texas at San Antonio
San Antonio, USA
aflores.cyber@gmail.com

***Abstract*— Splunk is a data analytics platform designed to search, analyze, and visualize machine-generated data. It supports the creation of statistical tables, reports, and visualizations to derive insights from large datasets. Common data sources include application logs, transaction records, configuration files, authentication logs, and fault data. Data ingestion begins with the installation of a forwarder on the host system, which transmits data to indexers. Indexers store the incoming data in structured repositories known as indexes, which are further organized into buckets containing the indexed files. This architecture enables users to perform efficient searches and real-time analysis on the ingested data.**

Keywords—indexes, forwarders, repositories, buckets, Boolean, indexer, wildcard, String, sourcetype, time range, fields, tables, parse,

I. Introduction

In this lab, I focused on gaining hands-on experience with Splunk. I began by setting up a Splunk instance and creating an admin role. I then proceeded to create additional users and assign appropriate roles to them. Next, I practiced indexing data, which enabled me to access new indexes for use in searches and reports. Furthermore, I worked on basic searching and reporting techniques. I started with simple keyword searches and gradually advanced to more complex queries using wildcards and Boolean expressions to filter and refine results more effectively.

II. Creating Users and Setting Roles

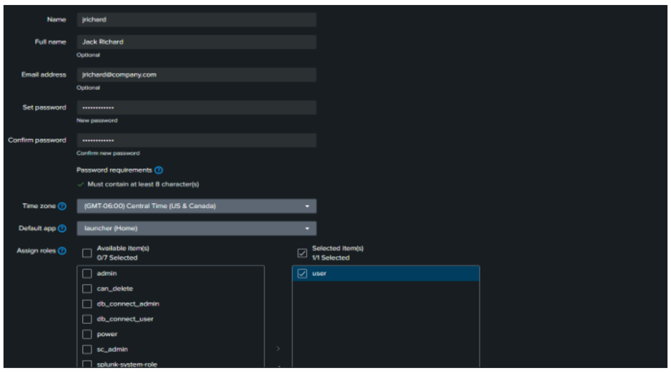
A. *Pre-defined roles*

When a Splunk instance is first launched, it includes three pre-defined roles: admin, power, and user. The user role can perform basic tasks such as running searches, creating knowledge objects, analyzing data, and generating visualizations.

However, it has limited access to configuration and system-level settings. The power role includes all the capabilities of the user role, with the added ability to share knowledge objects and access some additional features useful for more advanced analysis. The admin role has the highest level of access. In addition to all the capabilities of the power and user roles, an admin can manage system-wide settings, including managing data inputs, setting up forwarding, configuring index clustering, and other administrative tasks.

B. *Demonstration*

To begin the lab, I started by creating two roles in my Splunk deployment. My goal was to create users with different roles to verify their respective capabilities. I began by creating a user named Jack Richard, filling in the necessary information, and following in suit with the creation of another user named Cole Gilstrap. With Cole Gilstrap I had deselected the role user and assigned the power user in order to test the differences between all three predefined roles.

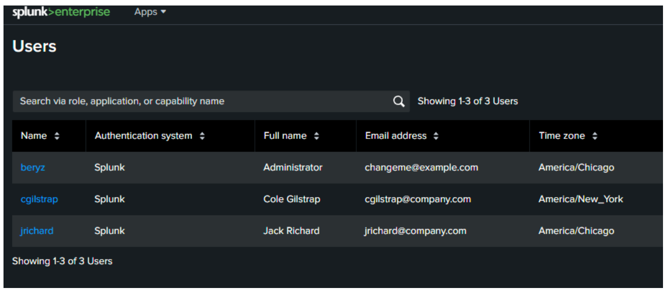


Name: jrichard
Full name: Jack Richard
Optional
Email address: jrichard@company.com
Optional
Set password: ••••••••••••
New password
Confirm password: ••••••••••••
Confirm new password
Password requirements
✓ Must contain at least 8 character(s)
Time zone: (GMT-06:00) Central Time (US & Canada)
Default app: launcher (Home)
Assign roles:
Available item(s) 0/7 Selected
- ☐ admin
- ☐ can_delete
- ☐ db_connect_admin
- ☐ db_connect_user
- ☐ power
- ☐ sc_admin
- ☐ splunk-system-role

Selected item(s) 1/1 Selected
- ☑ user

Figure 1: Creating a new user named Jack Richard and assigning the user role in Splunk.

After creating the user role for Jack Richard I followed the same procedure for Gilstrap, but instead I assigned the power role. From here I accessed the user section from the administrator account and saw that all the users were created.



Users

Search via role, application, or capability name
Showing 1-3 of 3 Users

Name	Authentication system	Full name	Email address	Time zone
beryz	Splunk	Administrator	changeme@example.com	America/Chicago
cgilstrap	Splunk	Cole Gilstrap	cgilstrap@company.com	America/New_York
jrichard	Splunk	Jack Richard	jrichard@company.com	America/Chicago

Showing 1-3 of 3 Users

Figure 2: List of users currently configured in Splunk, including their assigned roles and status.

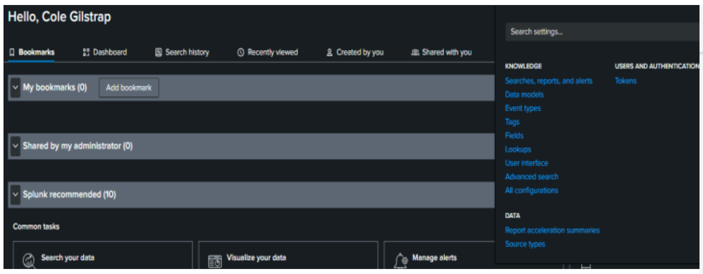


Figure 3: Logged in as Cole Gilstrap (assigned the Power role), with limited permissions under the Settings tab compared to the admin role, but with more functionality than the standard User role.

In conclusion, in this part of the lab I focused on creating and managing users within a Splunk instance by assigning them different roles to observe their respective permissions and capabilities. I began by creating two custom users, Jack Richard and Cole Gilstrap, with distinct roles to test the access levels associated with each. By logging in as each user, I was able to verify the differences in permissions between standard users, power users, and administrators. Additionally, I confirmed that the users were successfully created by reviewing the Users section under the Settings tab in Splunk Enterprise.

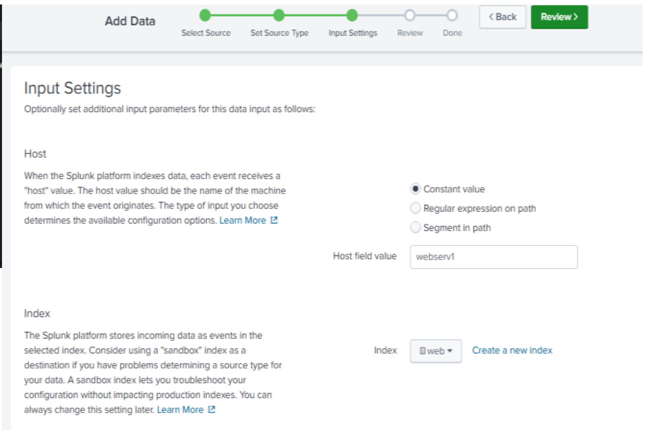
III. Indexing Data into Splunk

In this section of the lab, the focus was on indexing data into Splunk. To begin, I downloaded the tutorial dataset provided for this exercise. Through this process, I learned that there are three primary methods for ingesting data into Splunk: Upload, Monitor, and Forward.

There are three main methods for indexing data into Splunk: Upload, Monitor, and Forward. The Upload method is used to index data only once and is suitable for uploading local files directly from a computer. Monitor is designed to continuously observe files and directories, as well as network ports, for incoming data. This method works with data located on the same Splunk Enterprise instance and is particularly useful for testing scenarios. Lastly, the Forward method is commonly used in production environments. It involves installing forwarders on remote machines to send data to the Splunk instance over receiving ports, enabling centralized data collection from distributed sources.

A. Search & Reporting

When indexing the data into the Splunk instance, I utilized the Search & Reporting app. When initiating the data input process, Splunk prompted me to Select "Source". I proceeded by dragging the access.log file from webserver1 into the interface. Upon ingestion, I observed that the access.log file was automatically assigned to the source metadata field and categorized under the default source type of access_combined_wcookie.



Input Settings

Optionally set additional input parameters for this data input as follows:

Host

When the Splunk platform indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. Learn More

[✓ Constant value] Regular expression on path Segment in path

Host field value: webserv1

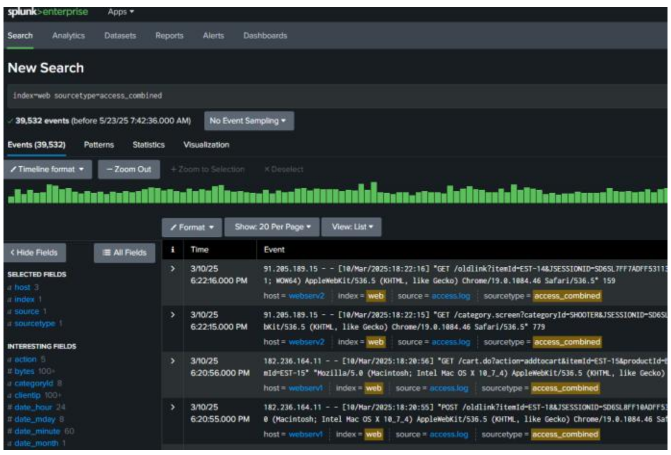
Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More

Index: web — Create a new index

Figure 4: Setting the Host field value to webserv1. Also, creating a new index titled web

Since the file originated from webserv1, I set the Host field value to match the source system name. This ensures accurate identification of the data's origin, which is essential for effective search, filtering, and correlation in Splunk. Additionally, instead of storing the data in the default main index, I created a new index named web to store the file. This approach promotes better data organization, simplifies management, and enables more targeted analysis of web server logs. I followed this procedure to align with industry best practices, as such measures are essential for maintaining clarity and scalability in real-world environments.



```
index=web sourcetype=access_combined
✓ 39,532 events (before 5/23/25 7:42:36.000 AM)
```

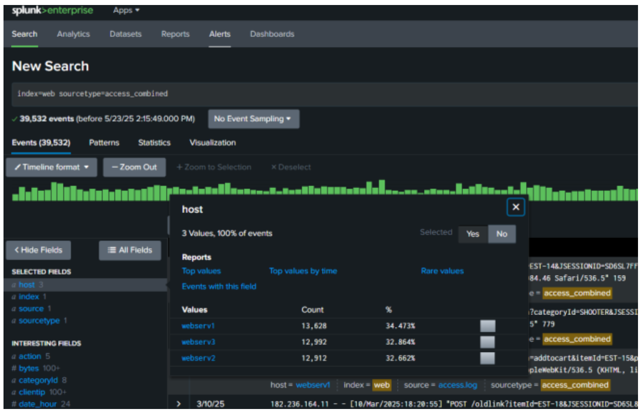
SELECTED FIELDS: host 3, index 1, source 1, sourcetype 1

INTERESTING FIELDS: action 5, bytes 100+, categoryId 8, clientip 100+, date_hour 24, date_mday 8, date_minute 60, date_month 1

```
i  Time                 Event
>  3/10/25              91.205.189.15 - - [10/Mar/2025:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF5311
   6:22:16.000 PM       1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159
                        host = webserv2 | index = web | source = access.log | sourcetype = access_combined
>  3/10/25              91.205.189.15 - - [10/Mar/2025:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL
   6:22:15.000 PM       bKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779
                        host = webserv2 | index = web | source = access.log | sourcetype = access_combined
>  3/10/25              182.236.164.11 - - [10/Mar/2025:18:20:56] "GET /cart.do?action=addtocart&itemId=EST-15&productId=
   6:20:56.000 PM       mId=EST-15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko)
                        host = webserv1 | index = web | source = access.log | sourcetype = access_combined
>  3/10/25              182.236.164.11 - - [10/Mar/2025:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8FF10ADFF5
   6:20:55.000 PM       0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Saf
                        host = webserv1 | index = web | source = access.log | sourcetype = access_combined
```

Figure 5: All the retrieved events originate from the web index that was previously created. The host values corresponding to each event accurately reflect their respective source systems (e.g., webserv1, webserv2, etc.), confirming that the data was indexed and categorized correctly.

To narrow down the search, I removed the Host and source. Instead, I retained the index and source type parameter. Here I used "index = web sourcetype=access_combined". This query retrieves all events stored in the web index with the specified source type, allowing me to analyze data from all three web servers efficiently.

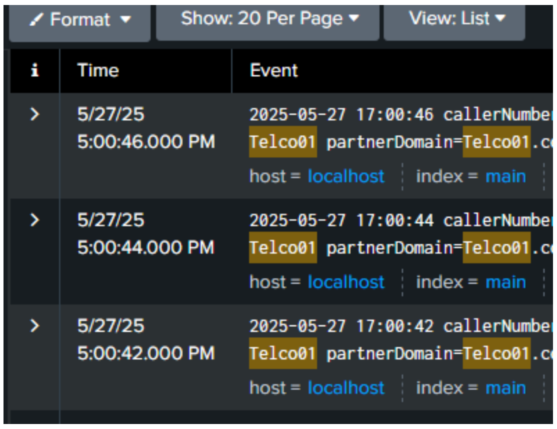


```
New Search
index=web sourcetype=access_combined
✓ 39,532 events (before 5/23/25 2:15:49.000 PM)   No Event Sampling
Events (39,532)   Patterns   Statistics   Visualization

host
3 Values, 100% of events
Values      Count     %
webserv1    13,628    34.473%
webserv3    12,992    32.864%
webserv2    12,912    32.662%

host = webserv1 | index = web | source = access.log | sourcetype = access_combined
3/10/25  182.236.164.11 - - [10/Mar/2025:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8
```

Figure 6: All the retrieved events originate from the web index that was previously created. The host values corresponding to each event accurately reflect their respective source systems (e.g., webserv1, webserv2, etc.), confirming that the data was indexed and categorized correctly.

IV. Basic Searching in Splunk

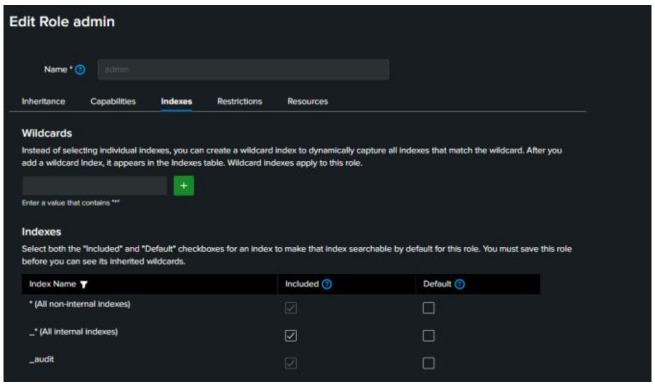
With the data now being indexed into the Splunk instance, I proceeded to begin exploring search functionality. Initially, I performed an all-time search for events containing specific keywords. My objective was to identify which indexes were storing relevant information. I began by querying a few keywords (invalid, amanda, Macintosh, and Telco01). Of these, only Telco01 returned results.



i	Time	Event
>	5/27/25 5:00:46.000 PM	2025-05-27 17:00:46 callerNumbe Telco01 partnerDomain=Telco01.c host = localhost index = main
>	5/27/25 5:00:44.000 PM	2025-05-27 17:00:44 callerNumbe Telco01 partnerDomain=Telco01.c host = localhost index = main
>	5/27/25 5:00:42.000 PM	2025-05-27 17:00:42 callerNumbe Telco01 partnerDomain=Telco01.c host = localhost index = main

Figure 7: Using the search head to search for Telco01 in the default index.

Here Telco01 is generating results because it originates from the event generator data that I ingested into the main index. Since the main index is searchable by default, searching for Telco01 in the Search & Reporting app returned results as expected. Upon further investigation, I realized that the other keywords I used were associated with data stored in different indexes. However, the admin role did not have the web and security indexes set as searchable by default. To address this, I updated the role settings to make those indexes searchable by default.



Edit Role admin

Name * admin

Inheritance | Capabilities | Indexes | Restrictions | Resources

Wildcards

Instead of selecting individual indexes, you can create a wildcard index to dynamically capture all indexes that match the wildcard. After you add a wildcard index, it appears in the Indexes table. Wildcard indexes apply to this role.

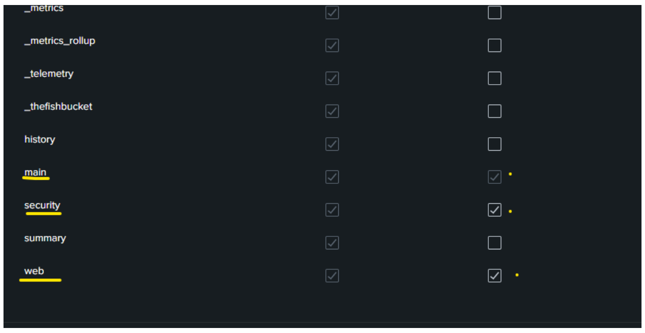
Enter a value that contains "*"

Indexes

Select both the "Included" and "Default" checkboxes for an index to make that index searchable by default for this role. You must save this role before you can see its inherited wildcards.

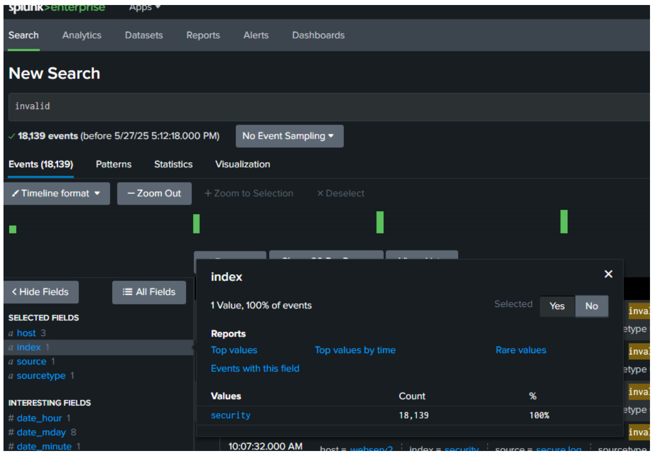
Index Name	Included	Default
* (All non-internal indexes)	✓	☐
_* (All internal indexes)	✓	☐
_audit	✓	☐

Figure 8: Editing the Role admin Indexes that are searchable by default.



_metrics	✓	☐
_metrics_rollup	✓	☐
_telemetry	✓	☐
_thefishbucket	✓	☐
history	✓	☐
main	✓	✓
security	✓	✓
summary	✓	☐
web	✓	✓

Figure 9: Including the created indexes web & security for the admin role

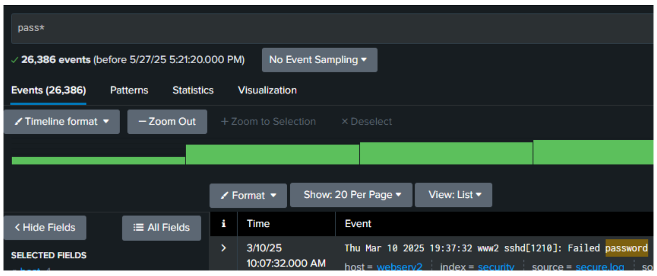


```
New Search
invalid
✓ 18,139 events (before 5/27/25 5:12:18.000 PM)   No Event Sampling
Events (18,139)   Patterns   Statistics   Visualization

index
1 Value, 100% of events
Values      Count     %
security    18,139    100%

10:07:32.000 AM   host = webserv2 | index = security | source = secure.log | sourcetype
```

Figure 10: Invalid is now searchable on the admin account since the Security index is now part of the default indexes searchable.

A. Wildcard

To match characters in a string in Splunk, I tested the use of the wildcard character *. For example, I searched for pass* to retrieve results that begin with the prefix “pass,” such as “password,” “passcode,” and “passphrase.” This helped me identify related terms and patterns in the indexed data more efficiently.


```
pass*
```
✓ 26,386 events (before 5/27/25 5:21:20.000 PM)

i	Time	Event
>	3/10/25 10:07:32.000 AM	Thu Mar 10 2025 19:37:32 www2 sshd[1210]: Failed password host = webserv2 index = security source = secure.log

Figure 11: When searched only password showed up in the results.

B.   Boolean

Boolean operators can be used in Splunk searches to refine and control query results. The most commonly used Boolean operators are AND, OR, and NOT. During my research, I learned that the AND operator is implied between terms by default. For example, the search query failed password is interpreted by Splunk as failed AND password, returning results that contain both terms. The OR operator is used to search for events that contain at least one of multiple terms.

For example, error OR failure will return events that contain either the word "error" or the word "failure," or both. This is useful when you want to capture related events that may be labeled differently. The NOT operator is used to exclude certain terms from the search results. For instance, login NOT failed will return events that contain the word "login" but exclude any that also contain the word "failed." This allows for more targeted searches by filtering out unwanted data. To test this, I used the following inputs in the Search head.

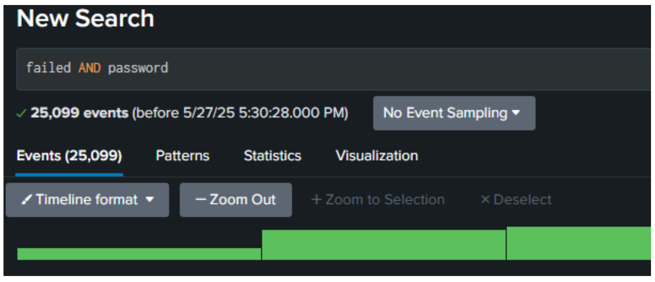

```
failed AND password
```
✓ 25,099 events (before 5/27/25 5:30:28.000 PM)

Figure 12: Using the AND Boolean operator. This search will search for failed and password.

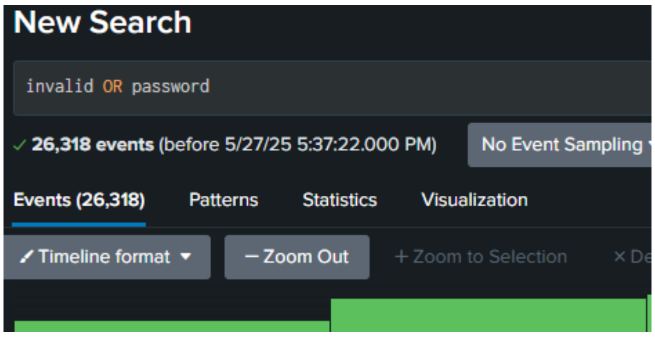

```
invalid OR password
```
✓ 26,318 events (before 5/27/25 5:37:22.000 PM)

Figure 13: Using the OR Boolean operator. This search looked at for Events containg invalid or password.

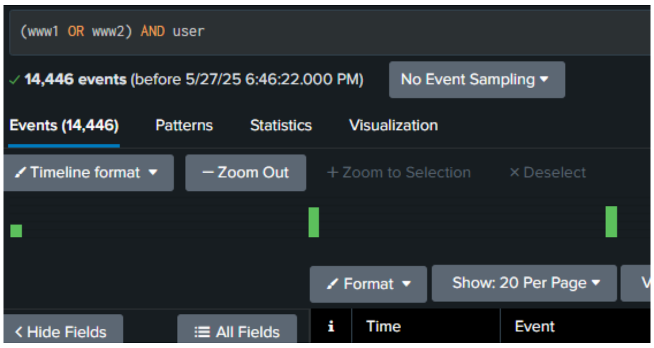
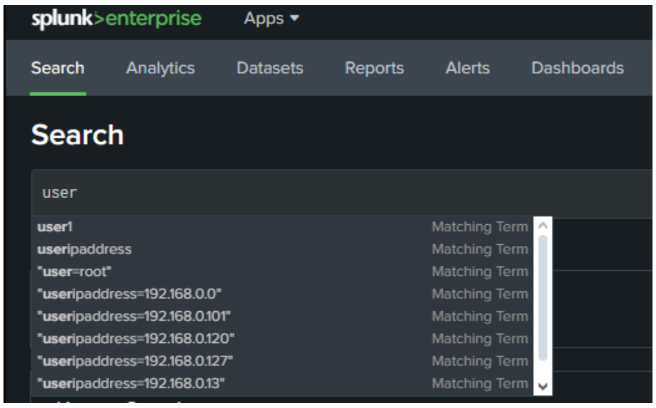

```
(www1 OR www2) AND user
```
✓ 14,446 events (before 5/27/25 6:46:22.000 PM)

Figure 14: Combining both the OR & AND Boolean operator. To search for www1 or www2 and user keyword.

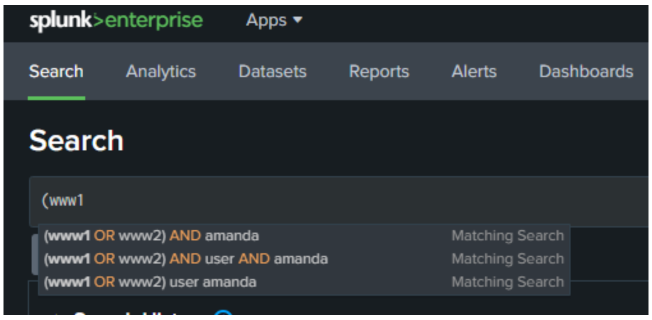
C.   Search Assistant

The search assistant is a tool in Splunk that helps with writing searches by providing selections to complete search strings. This is done by providing keywords within your data, matching searches used in recent search history, or showing list of commands after first pipe (|).


```
user
```

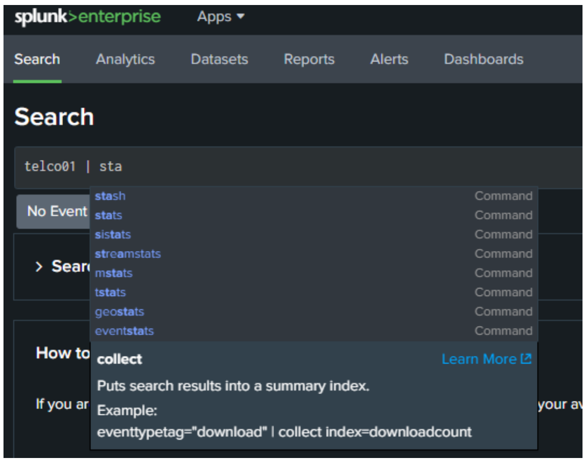
user1	Matching Term
useripaddress	Matching Term
"user=root"	Matching Term
"useripaddress=192.168.0.0"	Matching Term
"useripaddress=192.168.0.101"	Matching Term
"useripaddress=192.168.0.120"	Matching Term
"useripaddress=192.168.0.127"	Matching Term
"useripaddress=192.168.0.13"	Matching Term

Figure 15: The Search assistant tool showing matching terms within the Splunk index.


```
(www1
```

(www1 OR www2) AND amanda	Matching Search
(www1 OR www2) AND user AND amanda	Matching Search
(www1 OR www2) user amanda	Matching Search

Figure 16: The Search Assistant tool displaying matching searches based on recent query history. If needed, a matching search can be selected and executed directly from the suggestions.



```
telco01 | sta
```

stash	Command
stats	Command
sistats	Command
streamstats	Command
mstats	Command
tstats	Command
geostats	Command
eventstats	Command

collect Learn More

Puts search results into a summary index.

Example:

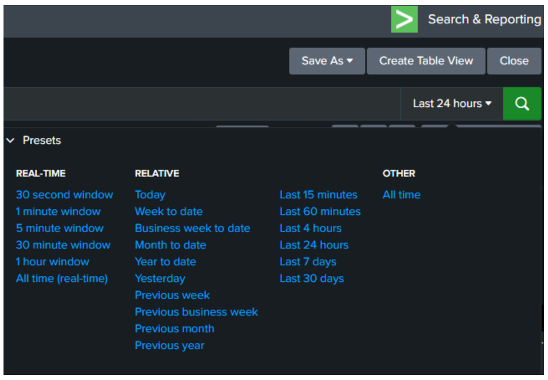
eventtypetag="download" | collect index=downloadcount

Figure 17: Search assistant also shows commands that can used with descriptions attached.

D. Setting Search Time Range

While performing searches and generating reports in Splunk, users can select from a range of preset time frames. Selecting an appropriate time range is important because parsing and indexing large volumes of data can take time. Using relative and precise time windows help ensure faster results and improve the accuracy of search queries by focusing only on relevant data.

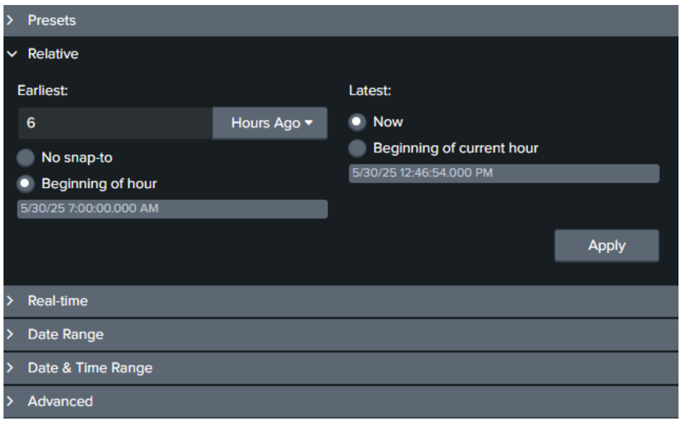
To begin this section, I selected the drop-down menu for the Time Range Picker and reviewed the available preset time ranges. I observed real-time options such as 1-minute, 5-minute, and 30-minute windows, which allow for continuous monitoring of incoming data. In addition, I explored relative time frames like "Today," "Yesterday," "Last 60 minutes," and "Last 24 hours." These presets help users efficiently define the scope of their searches, ensuring more accurate and relevant results.



REAL-TIME	RELATIVE		OTHER
30 second window	Today	Last 15 minutes	All time
1 minute window	Week to date	Last 60 minutes	
5 minute window	Business week to date	Last 4 hours	
30 minute window	Month to date	Last 24 hours	
1 hour window	Year to date	Last 7 days	
All time (real-time)	Yesterday	Last 30 days	
	Previous week		
	Previous business week		
	Previous month		
	Previous year		

Figure 18: Preset values in the Search & Repoting application.

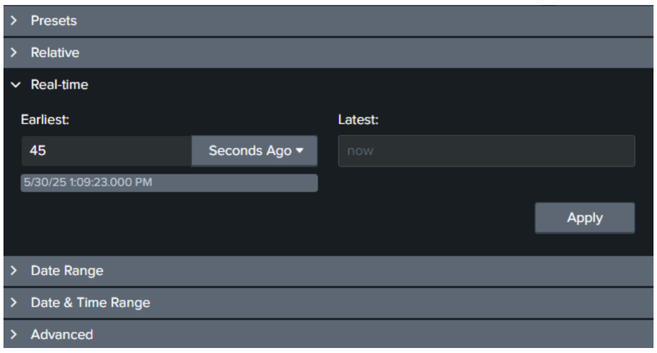
Furthermore, to test the capabilities of the Time Range Picker, I explored whether it was possible to define custom time ranges. I noticed that "6 hours ago" was not included as a predefined option. By navigating to the Relative time section, I was able to manually specify a custom time range setting the earliest time to 6 hours ago and the latest time to "now." This flexibility allows for more precise searches to keep the parsing time at a minimum.



Earliest: 6 Hours Ago
No snap-to
Beginning of hour [✓]
5/30/25 7:00:00.000 AM

Latest: Now [✓]
Beginning of current hour
5/30/25 12:46:54.000 PM

Figure 19: Using the Relative section in the Time Search Picker to keep the searches more narrow and precise.

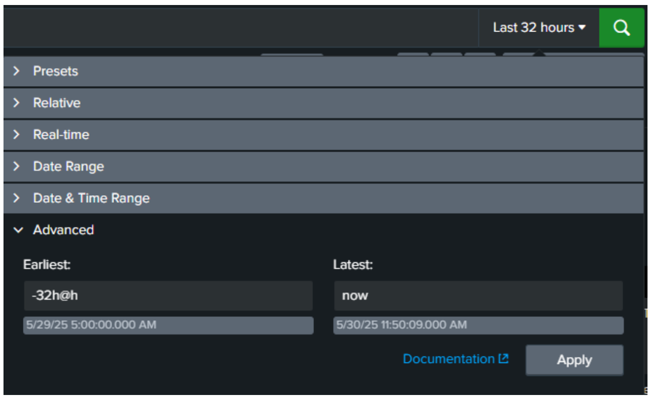


Earliest: 45 Seconds Ago
5/30/25 1:09:23.000 PM

Latest: now

Figure 20: Similar to Figure 19, the Real-time drop-down was selected to specify a custom search window. In this instance, a 45-second time range was manually chosen, as it was not available among the default presets.

In this final instance, I aimed to define a specific time range for my search by using the Advanced section of the Time Range Picker. I wanted to apply a time snap-back feature to control the starting point of the search. To do this, I set the Earliest time to -32h@h, which instructs Splunk to begin the search 32 hours prior to the Latest time, snapping to the start of the hour. The @h syntax rounds the time down to the nearest hour, removing the minutes and seconds. In this case, the search began at 5/29/2025 @ 5:00 AM and ran through to the current time, 5/30/2025 @ 11:50:09 AM.



Earliest: -32h@h
5/29/25 5:00:00.000 AM

Latest: now
5/30/25 11:50:09.000 AM

Figure 21: Setting a custom search range using the Advanced tab in the Search & Reporting application. When configured correctly, the top-right corner reflects the updated search range, displaying "32 hours."

E. Events Timeline

The Events Timeline in Splunk allows users to visually select and analyze specific time ranges within their search results. For example, in this case, the search was set to review data from the last 60 minutes. The Events Timeline displayed the frequency of events, with each column representing a one-minute interval. This visualization helps quickly identify trends, spikes, or irregularities in event activity over the selected time period.

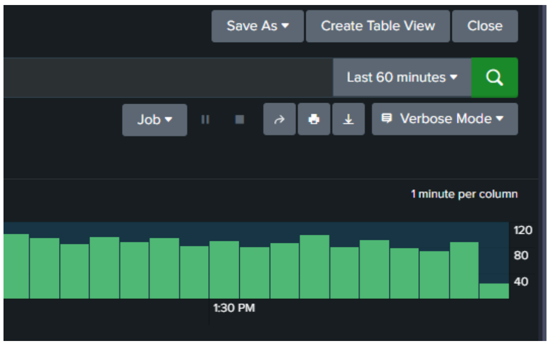
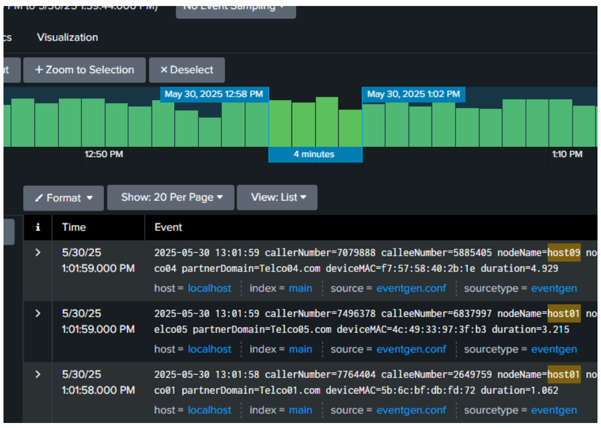


Figure 22: Showcasing the Events Timeline with the search range set to the last 60 minutes. The timeline displays event frequency using a legend where each column represents a 1-minute interval.

In this example, I selected the Last 60 Minutes as the initial time range, then used the Events Timeline to further narrow the scope by selecting and dragging over 4 minutes of the range. This allowed me to isolate and focus the search on a more specific window of recent activity to examine detailed event behavior.

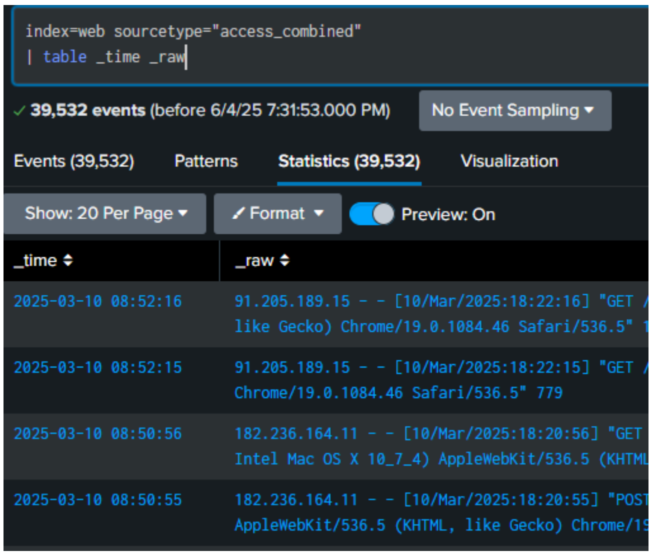


i	Time	Event
>	5/30/25 1:01:59.000 PM	2025-05-30 13:01:59 callerNumber=7079888 calleeNumber=5885405 nodeName=host09 no co04 partnerDomain=Telco04.com deviceMAC=f7:57:58:40:2b:1e duration=4.929 host = localhost index = main source = eventgen.conf sourcetype = eventgen
>	5/30/25 1:01:59.000 PM	2025-05-30 13:01:59 callerNumber=7496378 calleeNumber=6837997 nodeName=host01 no elco05 partnerDomain=Telco05.com deviceMAC=4c:49:33:97:3f:b3 duration=3.215 host = localhost index = main source = eventgen.conf sourcetype = eventgen
>	5/30/25 1:01:58.000 PM	2025-05-30 13:01:58 callerNumber=7764404 calleeNumber=2649759 nodeName=host01 no co01 partnerDomain=Telco01.com deviceMAC=5b:6c:bf:db:fd:72 duration=1.062 host = localhost index = main source = eventgen.conf sourcetype = eventgen

Figure 23: Selecting and dragging over a 4-minute segment within the Events Timeline to narrow the search range.

V. Using Fields in Searches

Now that I had some familiarity with basic searching in Splunk I wanted to take a step further by attempting to do field searches. To begin, I accessed the web index and filtered the results using the source type access_combined. To enhance the output and focus on key details, I appended the command | table _time _raw to the search. This produced a statistics table that displayed each event alongside its corresponding timestamp, allowing for a clearer view of when each event occurred.



```
index=web sourcetype="access_combined"
| table _time _raw
```

✓ 39,532 events (before 6/4/25 7:31:53.000 PM) No Event Sampling

Events (39,532) Patterns Statistics (39,532) Visualization

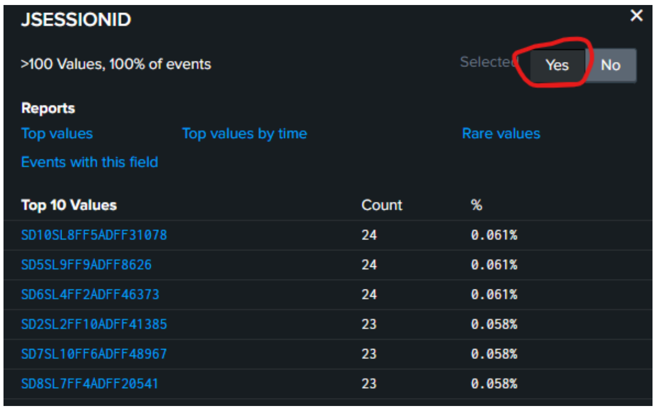
Show: 20 Per Page Format Preview: On

_time	_raw
2025-03-10 08:52:16	91.205.189.15 - - [10/Mar/2025:18:22:16] "GET / like Gecko) Chrome/19.0.1084.46 Safari/536.5" 1
2025-03-10 08:52:15	91.205.189.15 - - [10/Mar/2025:18:22:15] "GET / Chrome/19.0.1084.46 Safari/536.5" 779
2025-03-10 08:50:56	182.236.164.11 - - [10/Mar/2025:18:20:56] "GET Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML
2025-03-10 08:50:55	182.236.164.11 - - [10/Mar/2025:18:20:55] "POST AppleWebKit/536.5 (KHTML, like Gecko) Chrome/1

Figure 24: Retrieving information under the web index using the access_combined source type.

A. Using Fields Sidebar

In Splunk, fields are automatically categorized as interesting if they appear in at least 20% of the returned events. With this in mind, I began exploring field selection by choosing one of these interesting fields, JSESSIONID. I added it to the Selected Fields section by clicking "Yes." The field was marked with an alphabetical (a) icon, indicating it contains alphanumeric values, as opposed to the # symbol used for numeric fields. Once selected, JSESSIONID was displayed alongside each event, making it easier to track and analyze session identifiers within the dataset.



JSESSIONID

>100 Values, 100% of events Selected [✓ Yes] No

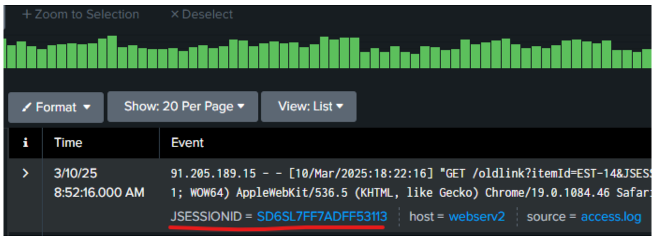
Reports

Top values Top values by time Rare values

Events with this field

Top 10 Values	Count	%
SD10SL8FF5ADFF31078	24	0.061%
SD5SL9FF9ADFF8626	24	0.061%
SD6SL4FF2ADFF46373	24	0.061%
SD2SL2FF10ADFF41385	23	0.058%
SD7SL10FF6ADFF48967	23	0.058%
SD8SL7FF4ADFF20541	23	0.058%

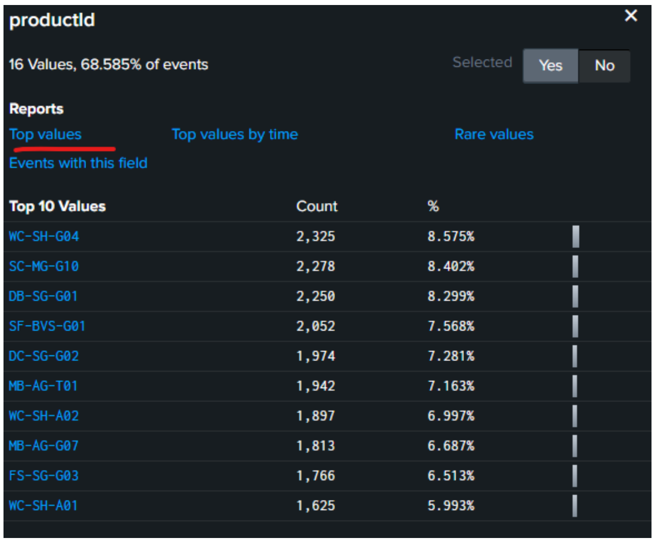
Figure 25: Changing JSESSIONID from an interesting field to a selected field.



i	Time	Event
>	3/10/25 8:52:16.000 AM	91.205.189.15 - - [10/Mar/2025:18:22:16] "GET /oldlink?itemId=EST-14&JSES 1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safar JSESSIONID = SD6SL7FF7ADFF53113 host = webserv2 source = access.log

Figure 26: JESSIONID now showing up as a selected field under the event section.

Lastly, I wanted to explore one of the reporting features using an interesting field. For example, with the productId field, I selected the Top Values option to generate a report displaying the top 20 most frequently occurring product IDs.


productId

16 Values, 68.585% of events

Selected Yes No

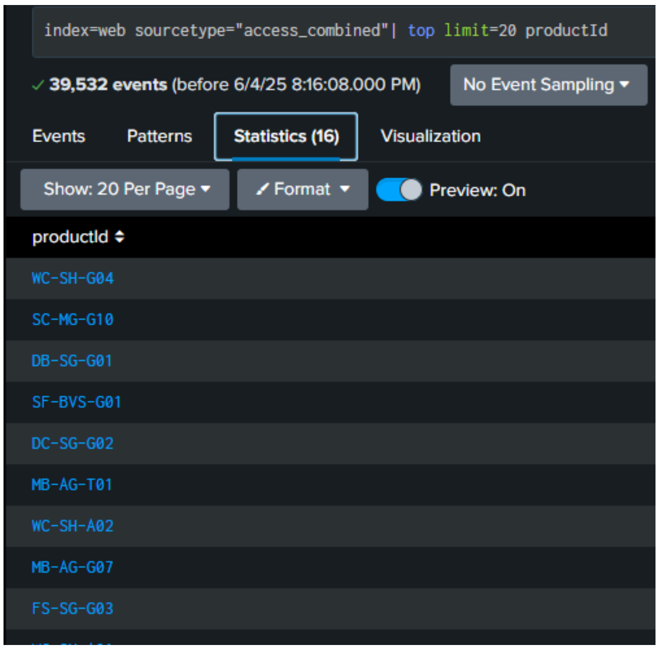
Reports

Top values Top values by time Rare values

Events with this field

Top 10 Values	Count	%
WC-SH-G04	2,325	8.575%
SC-MG-G10	2,278	8.402%
DB-SG-G01	2,250	8.299%
SF-BVS-G01	2,052	7.568%
DC-SG-G02	1,974	7.281%
MB-AG-T01	1,942	7.163%
WC-SH-A02	1,897	6.997%
MB-AG-G07	1,813	6.687%
FS-SG-G03	1,766	6.513%
WC-SH-A01	1,625	5.993%

Figure 27: Selecting the Top Values option to display the Top values in productid, which reveals the 20 most frequently occurring entries.


```
index=web sourcetype="access_combined"| top limit=20 productId
```

✓ 39,532 events (before 6/4/25 8:16:08.000 PM)

No Event Sampling

Events Patterns Statistics (16) Visualization

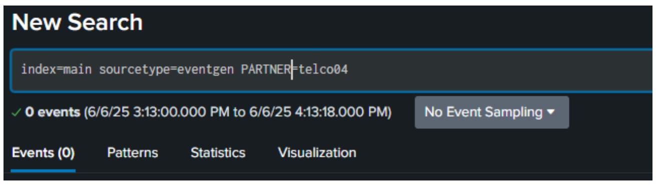
Show: 20 Per Page Format Preview: On

productId
WC-SH-G04
SC-MG-G10
DB-SG-G01
SF-BVS-G01
DC-SG-G02
MB-AG-T01
WC-SH-A02
MB-AG-G07
FS-SG-G03

Figure 28: Showcasing the Top 20 values in productid.

B. *Using Fields in Searches*

To test whether field names in Splunk are case-sensitive, I conducted a simple search experiment. I began by querying the field partner in all lowercase, which successfully parsed and returned relevant events. However, when I repeated the same search using the uppercase field name PARTNER, no results were returned. This confirmed that field names in Splunk are indeed case-sensitive, and proper casing must be used to retrieve accurate results.


New Search

```
index=main sourcetype=eventgen PARTNER=telco04
```

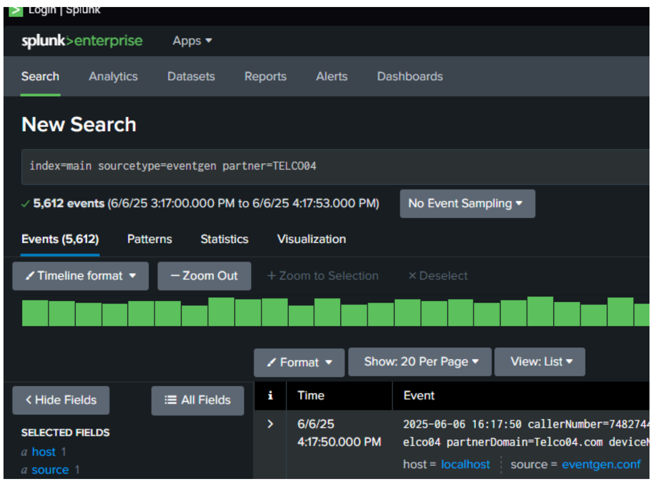
✓ 0 events (6/6/25 3:13:00.000 PM to 6/6/25 4:13:18.000 PM)

No Event Sampling

Events (0) Patterns Statistics Visualization

Figure 29: Using the fieldtype PARTNER

Likewise, I also wanted to test if field values were also case sensitive. To do this I tried changing the field value telco04 from lowercase to uppercase. For example, I ran the following search.



New Search

```
index=main sourcetype=eventgen partner=TELCO04
```

✓ 5,612 events (6/6/25 3:17:00.000 PM to 6/6/25 4:17:53.000 PM)

No Event Sampling

Events (5,612) Patterns Statistics Visualization

Timeline format Zoom Out + Zoom to Selection × Deselect

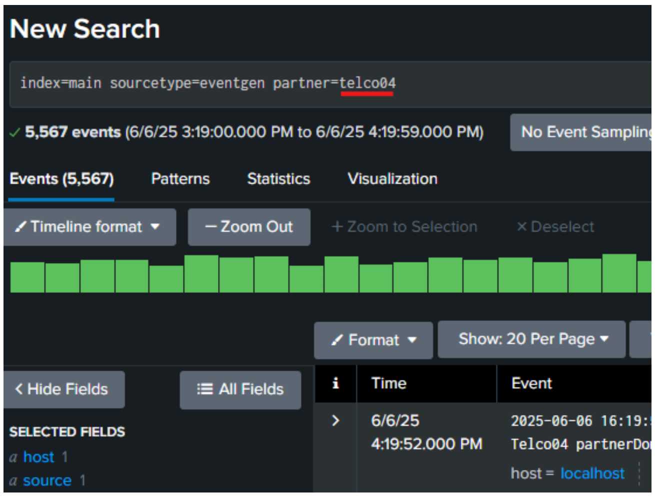
Format Show: 20 Per Page View: List

Hide Fields All Fields

SELECTED FIELDS
a host 1
a source 1

i	Time	Event
>	6/6/25 4:17:50.000 PM	2025-06-06 16:17:50 callerNumber=748274 elco04 partnerDomain=Telco04.com device host = localhost source = eventgen.conf

Figure 30: Searching with the field value telco04 being capitalized instead of lowercase.


New Search

```
index=main sourcetype=eventgen partner=telco04
```

✓ 5,567 events (6/6/25 3:19:00.000 PM to 6/6/25 4:19:59.000 PM)

No Event Samplin

Events (5,567) Patterns Statistics Visualization

Timeline format Zoom Out + Zoom to Selection × Deselect

Format Show: 20 Per Page

Hide Fields All Fields

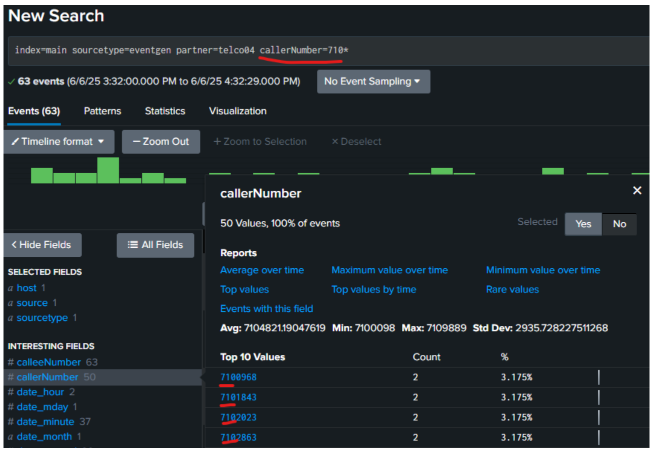
SELECTED FIELDS
a host 1
a source 1

i	Time	Event
>	6/6/25 4:19:52.000 PM	2025-06-06 16:19: Telco04 partnerDo host = localhost

Figure 31: Searching with the field value telco04 being lowercase.

With the search completed, I was able to confirm that field names in Splunk are case-sensitive. Any variation in casing such as uppercase vs. lowercase will result in the field not being recognized, and no results being returned. On the contrary, field types can have some changes such as uppercase and lowercase.

With the foundational concepts clarified, I wanted to take my search a step further by applying several techniques learned throughout the lab. For this search, I aimed to return events that met the following criteria: they belonged to the main index, used the eventgen sourcetype, had a field value of partner=telco04, and included a callerNumber field that started with the digits 710. This search combined multiple filters, field-value matching, and the use of a wildcard to demonstrate more advanced querying capabilities in Splunk.



```
index=main sourcetype=eventgen partner=telco04 callerNumber=710*
```

63 events (6/6/25 3:32:00.000 PM to 6/6/25 4:32:29.000 PM)

callerNumber

50 Values, 100% of events

Avg: 7104821.19047619 Min: 7100098 Max: 7109889 Std Dev: 2935.728227511268

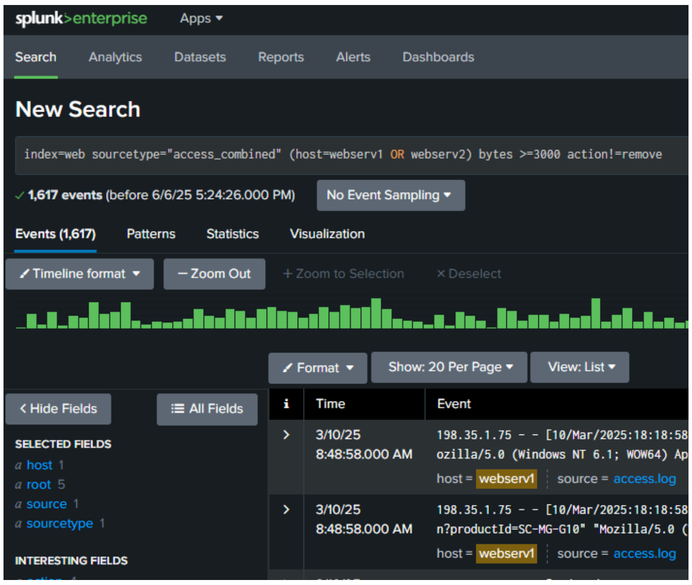
Top 10 Values	Count	%
7100968	2	3.175%
7101843	2	3.175%
7102023	2	3.175%
7102863	2	3.175%

Figure 32: Using multiple fields in the search header.

As shown in Figure 32, I was able to successfully retrieve data from the specified fields using the constructed search. To further validate the results, I clicked on the callerNumber field within the event data to confirm that each value indeed began with 710, ensuring the search criteria were accurately applied.

C. *Comparison Operators*

In this section, I continued working with multiple fields in my searches while also incorporating Boolean operators to add more precision and flexibility. I used a variety of operators such as <, >, <=, >=, OR, AND, and != to create more complex queries.



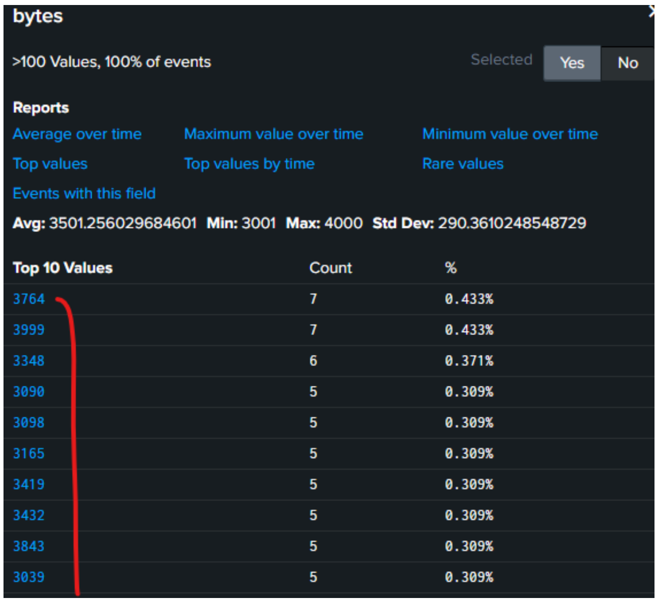
```
index=web sourcetype="access_combined" (host=webserv1 OR webserv2) bytes >=3000 action!=remove
```

1,617 events (before 6/6/25 5:24:26.000 PM)

```
3/10/25          198.35.1.75 - - [10/Mar/2025:18:18:58
8:48:58.000 AM   ozilla/5.0 (Windows NT 6.1; WOW64) Ap
                 host = webserv1   source = access.log

3/10/25          198.35.1.75 - - [10/Mar/2025:18:18:58
8:48:58.000 AM   n?productId=SC-MG-G10" "Mozilla/5.0 (
                 host = webserv1   source = access.log
```

Figure 33: Using the following: index=web sourcetype="access_combined" (host=webserv1 OR webserv2) bytes >=3000 action!=remove in the search header.

In this search I specified exactly where I wanted to pull data from the events. This search looked inside the index web and the sourcetype access_combined while looking at a host=webserv1 or webserv2 with bytes being greater than or equal to 3000 and inside the action field that is not equal to remove. I knew the information was correctly returned when I looked into each of the following fields.



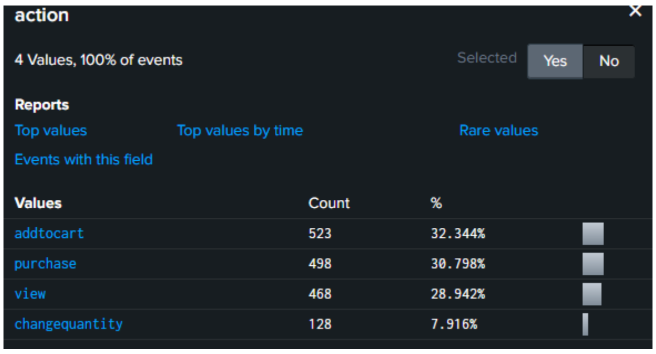
bytes

>100 Values, 100% of events

Avg: 3501.256029684601 Min: 3001 Max: 4000 Std Dev: 290.3610248548729

Top 10 Values	Count	%
3764	7	0.433%
3999	7	0.433%
3348	6	0.371%
3090	5	0.309%
3098	5	0.309%
3165	5	0.309%
3419	5	0.309%
3432	5	0.309%
3843	5	0.309%
3039	5	0.309%

Figure 34: The bytes field containing values greater than 3000.

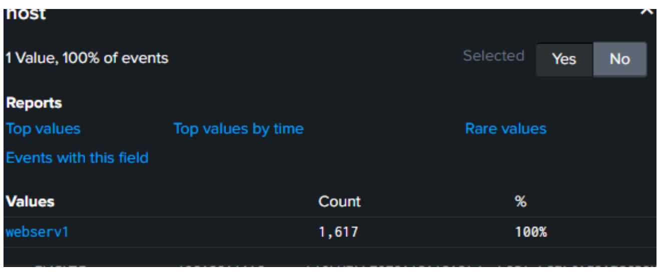


action

4 Values, 100% of events

Values	Count	%
addtocart	523	32.344%
purchase	498	30.798%
view	468	28.942%
changequantity	128	7.916%

Figure 35: The action field does not contain remove.

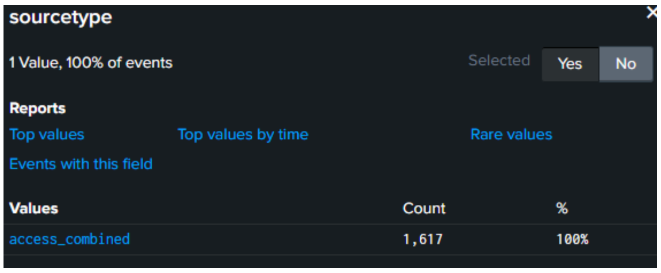


host

1 Value, 100% of events

Values	Count	%
webserv1	1,617	100%

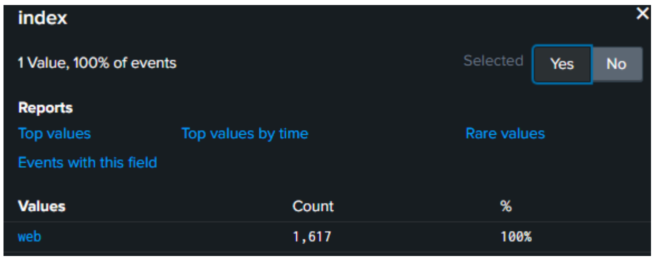
Figure 36: The host field containg webserv1 as it only needed to have webserv1 or webserv2 to fufill the condition.



1 Value, 100% of events

Values	Count	%
access_combined	1,617	100%

Figure 37: The sourcetype field is equal to access_combined.



1 Value, 100% of events

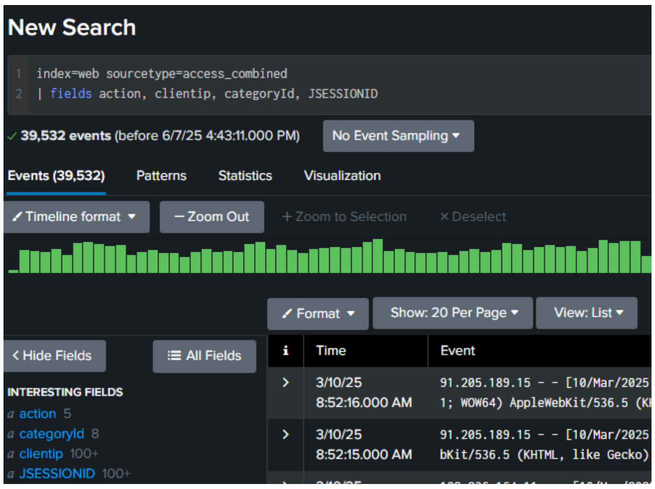
Values	Count	%
web	1,617	100%

Figure 38: The index field is equal to web.

VI. Search Language Fundamentals

A. *Field Command*

Fields in Splunk can be leveraged to perform a wide range of searches and data manipulations. To become more familiar with using the pipe (|) command, I began by applying the fields command. This allowed me to filter and limit the list of fields displayed in the search results, making the output more focused and easier to analyze. By specifying only the fields of interest, I was able to streamline the results and reduce visual clutter.



```
index=web sourcetype=access_combined
| fields action, clientip, categoryId, JSESSIONID
```

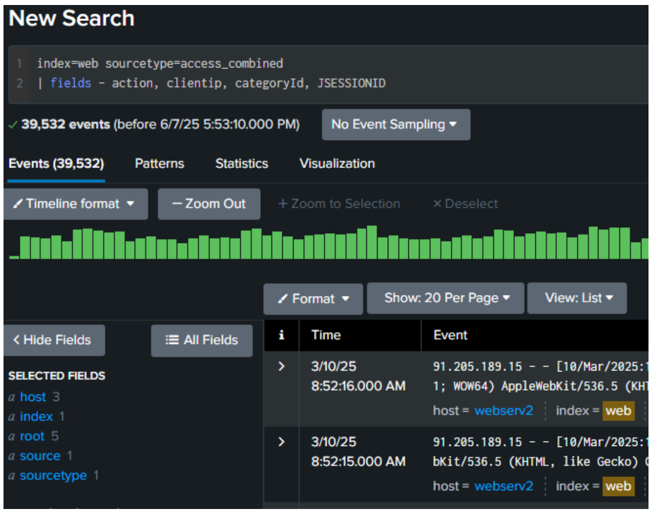
✓ 39,532 events (before 6/7/25 4:43:11.000 PM)

INTERESTING FIELDS
a action 5
a categoryId 8
a clientip 100+
a JSESSIONID 100+

i	Time	Event
>	3/10/25 8:52:16.000 AM	91.205.189.15 - - [10/Mar/2025 1; WOW64) AppleWebKit/536.5 (K
>	3/10/25 8:52:15.000 AM	91.205.189.15 - - [10/Mar/2025 bKit/536.5 (KHTML, like Gecko)

Figure 39: Applying the fields command to filter and display only the specified fields in the search results. In this example, the fields shown are: action, clientip, categoryId, and JSESSIONID.

On the contrary I also wanted to remove the same specified fields as it may be needed to help with parsing since it could take longer when working with more data. To do this I used a – to specific the removal of these fields.



```
index=web sourcetype=access_combined
| fields - action, clientip, categoryId, JSESSIONID
```

✓ 39,532 events (before 6/7/25 5:53:10.000 PM)

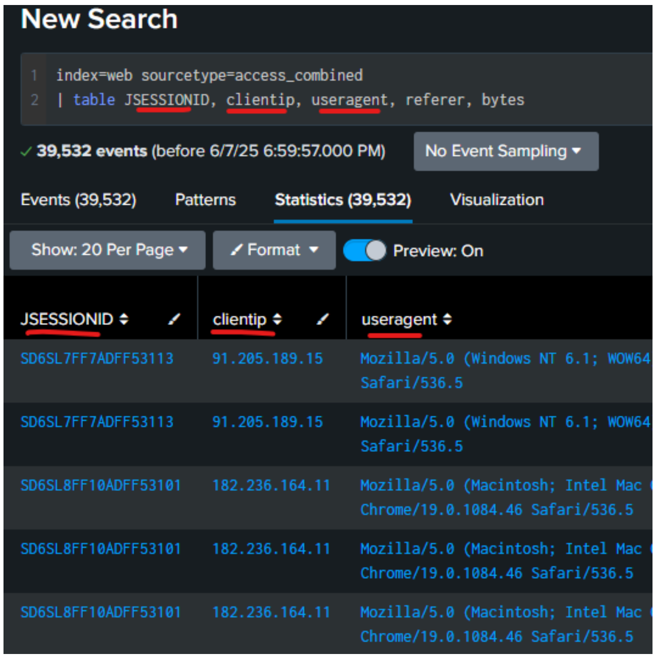
SELECTED FIELDS
a host 3
a index 1
a root 5
a source 1
a sourcetype 1

i	Time	Event
>	3/10/25 8:52:16.000 AM	91.205.189.15 - - [10/Mar/2025: 1; WOW64) AppleWebKit/536.5 (KH host = webserv2 index = web
>	3/10/25 8:52:15.000 AM	91.205.189.15 - - [10/Mar/2025: bKit/536.5 (KHTML, like Gecko) host = webserv2 index = web

Figure 40: Using the fields command with a minus sign (-) to remove specific fields from the search results. In this example, the excluded fields are: action, clientip, categoryId, and JSESSIONID.

B. *Table Command*

To analyze the data in a more structured format, I used the table command to create a statistical table displaying the selected fields: JSESSIONID, clientip, useragent, referrer, and bytes. The table command organizes the results into rows and columns, where each row represents a single event and each column corresponds to one of the specified fields. The fields appear in the order listed in the command, making it easier to interpret and compare data across events.



```
index=web sourcetype=access_combined
| table JSESSIONID, clientip, useragent, referer, bytes
```

✓ 39,532 events (before 6/7/25 6:59:57.000 PM)

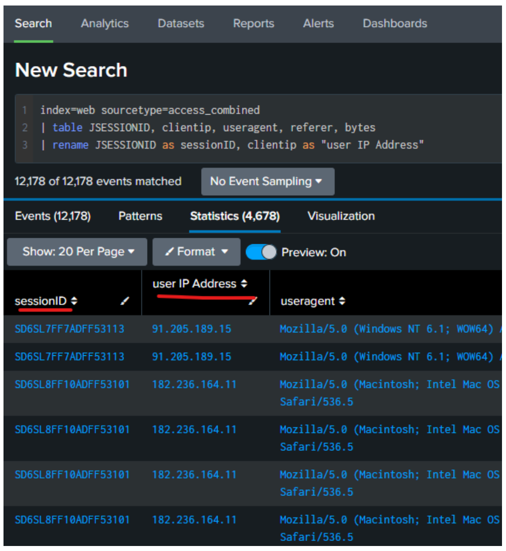
JSESSIONID	clientip	useragent
SD6SL7FF7ADFF53113	91.205.189.15	Mozilla/5.0 (Windows NT 6.1; WOW64 Safari/536.5
SD6SL7FF7ADFF53113	91.205.189.15	Mozilla/5.0 (Windows NT 6.1; WOW64 Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac Chrome/19.0.1084.46 Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac Chrome/19.0.1084.46 Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac Chrome/19.0.1084.46 Safari/536.5

Figure 41: table showing the specified fields.

This command can be useful when you want to focus on specific fields of interest without the distraction of additional data. It helps streamline the output, making it easier to analyze patterns, compare values across events, and prepare data for reporting or visualization. The table command is especially valuable when presenting results to stakeholders who need clear, readable information without the complexity of raw event data.

C. *Rename Command*

The rename command in Splunk is used to assign new, more readable or meaningful names to existing fields. In the screenshot below, two fields were renamed: JSESSIONID was changed to sessionID, and clientip was renamed to user IP address. This can be particularly helpful for improving clarity in reports or aligning field names with business terminology.



```
index=web sourcetype=access_combined
| table JSESSIONID, clientip, useragent, referer, bytes
| rename JSESSIONID as sessionID, clientip as "user IP Address"
```

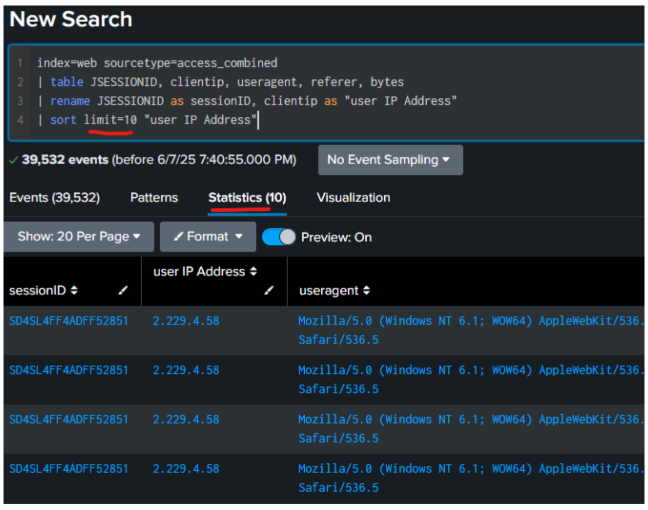
12,178 of 12,178 events matched

sessionID	user IP Address	useragent
SD6SL7FF7ADFF53113	91.205.189.15	Mozilla/5.0 (Windows NT 6.1; WOW64)
SD6SL7FF7ADFF53113	91.205.189.15	Mozilla/5.0 (Windows NT 6.1; WOW64)
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac OS Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac OS Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac OS Safari/536.5
SD6SL8FF10ADFF53101	182.236.164.11	Mozilla/5.0 (Macintosh; Intel Mac OS Safari/536.5

Figure 42: The table and rename command being used to change the JSESSIONID to sessionID and clientip to user IP address.

D. *Sort Command*

The sort command in Splunk arranges search results in ascending order by default, unless a descending order is explicitly specified using a minus sign (-). In this instance, I used the sort command to organize the results by user IP addresses and applied a limit to display only the top 10 results.



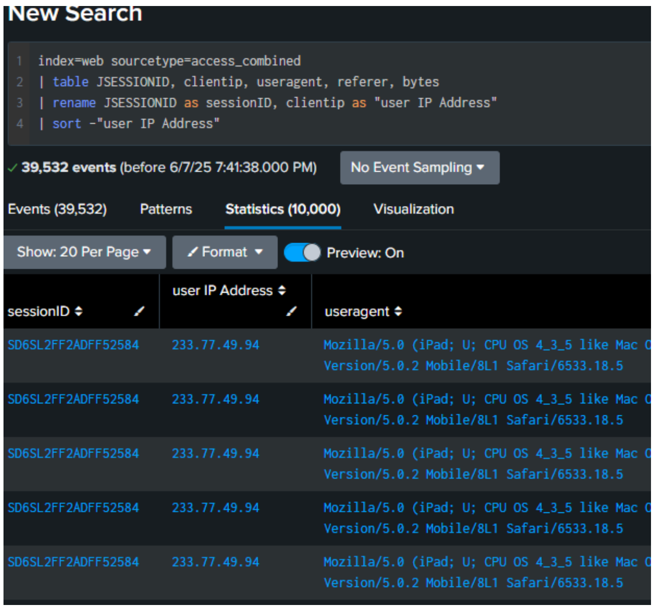
```
index=web sourcetype=access_combined
| table JSESSIONID, clientip, useragent, referer, bytes
| rename JSESSIONID as sessionID, clientip as "user IP Address"
| sort limit=10 "user IP Address"
```

✓ 39,532 events (before 6/7/25 7:40:55.000 PM)

sessionID	user IP Address	useragent
SD4SL4FF4ADFF52851	2.229.4.58	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536. Safari/536.5
SD4SL4FF4ADFF52851	2.229.4.58	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536. Safari/536.5
SD4SL4FF4ADFF52851	2.229.4.58	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536. Safari/536.5
SD4SL4FF4ADFF52851	2.229.4.58	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536. Safari/536.5

Figure 43: Sorting user IP addresses in ascending order and giving it a limit of 10 statistics.

However, in some cases, it may be more beneficial to display the statistics in descending order, such as when identifying the highest values first. To achieve this, I added a minus sign (-) in front of the user IP address field within the sort command. This reversed the order of the results, showing the most frequent or relevant IP addresses at the top of the table.



```
index=web sourcetype=access_combined
| table JSESSIONID, clientip, useragent, referer, bytes
| rename JSESSIONID as sessionID, clientip as "user IP Address"
| sort -"user IP Address"
```

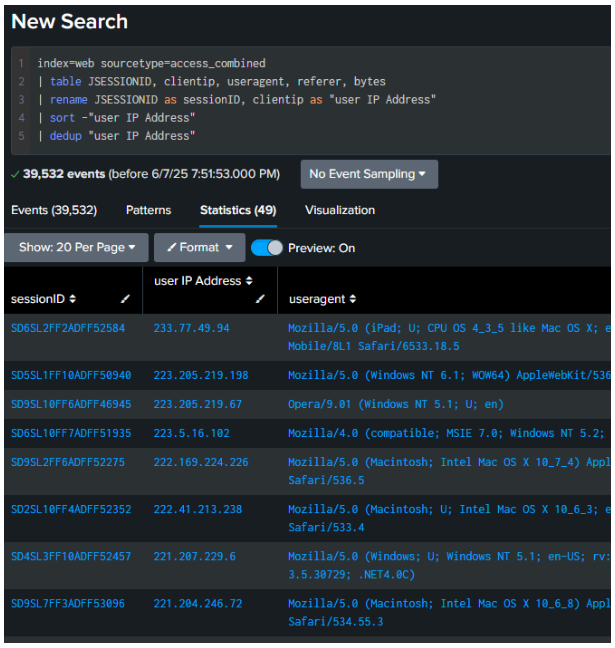
✓ 39,532 events (before 6/7/25 7:41:38.000 PM)

sessionID	user IP Address	useragent
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac O Version/5.0.2 Mobile/8L1 Safari/6533.18.5
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac O Version/5.0.2 Mobile/8L1 Safari/6533.18.5
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac O Version/5.0.2 Mobile/8L1 Safari/6533.18.5
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac O Version/5.0.2 Mobile/8L1 Safari/6533.18.5
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac O Version/5.0.2 Mobile/8L1 Safari/6533.18.5

Figure 44: Sorting user IP addresses in decending order.

E. *Dedup Command*

The dedup command in Splunk is used to remove duplicate events that contain identical values in specified fields. For example, I observed that the IP addresses in the results table appeared multiple times, making the data harder to interpret. To streamline the output and improve readability, I applied the dedup command to eliminate these duplicates, ensuring that each IP address was displayed only once based on the selected field.



```
index=web sourcetype=access_combined
| table JSESSIONID, clientip, useragent, referer, bytes
| rename JSESSIONID as sessionID, clientip as "user IP Address"
| sort -"user IP Address"
| dedup "user IP Address"
```

✓ 39,532 events (before 6/7/25 7:51:53.000 PM)

sessionID	user IP Address	useragent
SD6SL2FF2ADFF52584	233.77.49.94	Mozilla/5.0 (iPad; U; CPU OS 4_3_5 like Mac OS X; e Mobile/8L1 Safari/6533.18.5
SD5SL1FF10ADFF50940	223.205.219.198	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536
SD9SL10FF6ADFF46945	223.205.219.67	Opera/9.01 (Windows NT 5.1; U; en)
SD6SL10FF7ADFF51935	223.5.16.102	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2;
SD9SL2FF6ADFF52275	222.169.224.226	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) Appl Safari/536.5
SD2SL10FF4ADFF52352	222.41.213.238	Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_3; e Safari/533.4
SD4SL3FF10ADFF52457	221.207.229.6	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv: 3.5.30729; .NET4.0C)
SD9SL7FF3ADFF53096	221.204.246.72	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) Appl Safari/534.55.3

Figure 45: Using the dedup command to remove duplicates from the user IP Address field.

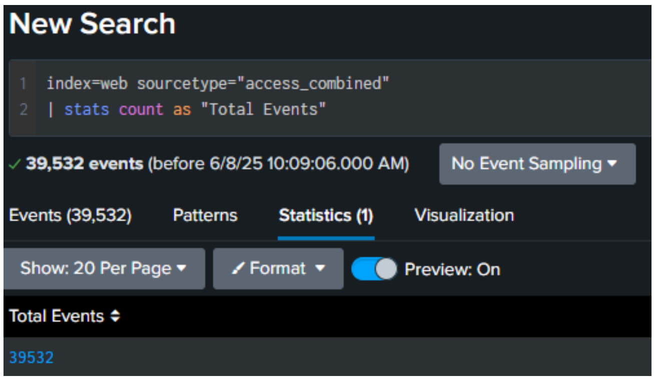
VII. Basic Transforming Commands

Transforming commands in Splunk are used to order search results into a statistics table. Examples of transforming commands are table, stats, top, chart, time chart, and rare command.

A. Stats Command

The stats command in Splunk is used to perform statistical calculations on data retrieved from a search. It allows you to summarize and analyze large volumes of data efficiently. When using the stats command, it's important to apply statistical functions such as count, distinct_count (or dc), sum, avg, list, values, max, and min. These functions enable you to calculate metrics like totals, averages, unique values, and ranges, helping uncover meaningful insights within your dataset.

To start off I wanted to test out the statistical functions starting with count. The count function can be used to return the number of events for current search.

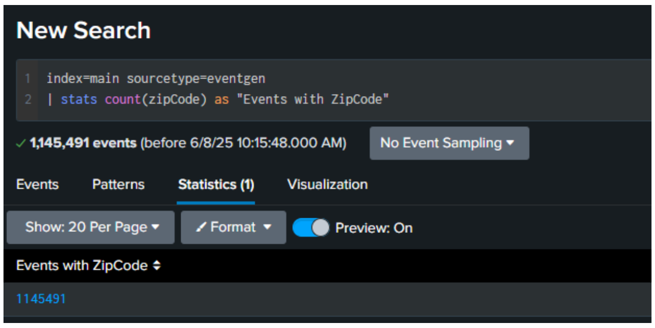


```
index=web sourcetype="access_combined"
| stats count as "Total Events"
```

✓ 39,532 events (before 6/8/25 10:09:06.000 AM)

Total Events
39532

Figure 46: Using the count statistical function to count the events in this search and also renamed the count field to "Total Events" using the as clause.



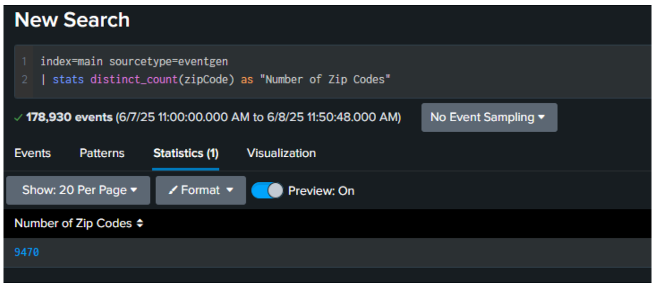
```
index=main sourcetype=eventgen
| stats count(zipCode) as "Events with ZipCode"
```

✓ 1,145,491 events (before 6/8/25 10:15:48.000 AM)

Events with ZipCode
1145491

Figure 47: In this example, I used the count function as an argument within the stats command to calculate the total number of events that contain the zipCode field. To improve readability in the results, I also applied the AS clause to rename the output column to "Events with ZipCode."

Next, I tested the stats command using the distinct_count function, abbreviated as dc. This function calculates the number of unique values for a specified field. In this case, I applied it to the zipCode field to determine how many unique zip codes were present in the result set. By using distinct_count, duplicate entries are excluded, providing an accurate count of distinct values within the data.



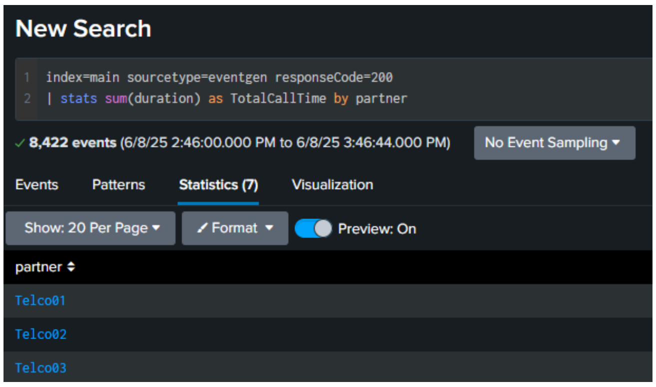
```
index=main sourcetype=eventgen
| stats distinct_count(zipCode) as "Number of Zip Codes"
```

✓ 178,930 events (6/7/25 11:00:00.000 AM to 6/8/25 11:50:48.000 AM)

Number of Zip Codes
9470

Figure 48: Applying the distinct_count (dc) function in the stats command to calculate the number of unique values in the zipCode field. This ensures that only distinct zip codes are counted, excluding duplicates from the result set.

I also tested the sum function within the stats command, which calculates the total of all values in a specified numeric field. In this instance, I used it to sum the values in the duration field. Additionally, I grouped the results by the partner field and renamed the output column to TotalCallTime.

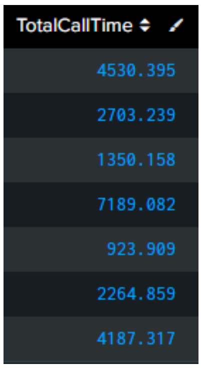


```
index=main sourcetype=eventgen responseCode=200
| stats sum(duration) as TotalCallTime by partner
```

✓ 8,422 events (6/8/25 2:46:00.000 PM to 6/8/25 3:46:44.000 PM)

partner
Telco01
Telco02
Telco03

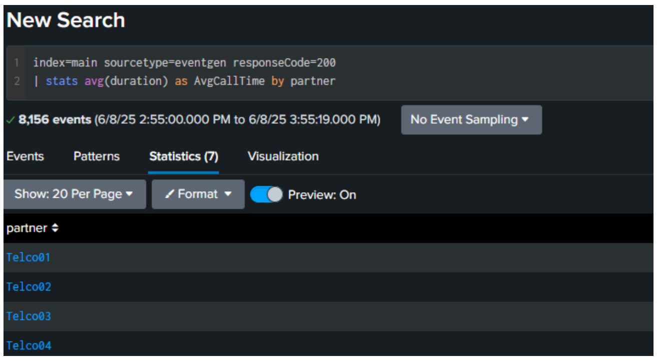
Figure 49: Using the sum fuction to summarize the TotalCallTime.



TotalCallTime
4530.395
2703.239
1350.158
7189.082
923.909
2264.859
4187.317

Figure 50: TotalCallTime summarized.

Furthermore, I tested the avg function within the stats command to calculate the average call duration. To do this, I replaced the sum function with avg and applied it to the duration field. I also renamed the output column to AvgCallTime for clarity.



New Search

```
index=main sourcetype=eventgen responseCode=200
| stats avg(duration) as AvgCallTime by partner
```

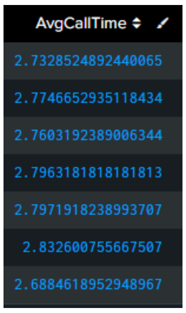
✓ 8,156 events (6/8/25 2:55:00.000 PM to 6/8/25 3:55:19.000 PM) No Event Sampling

Events Patterns Statistics (7) Visualization

Show: 20 Per Page Format Preview: On

partner
Telco01
Telco02
Telco03
Telco04

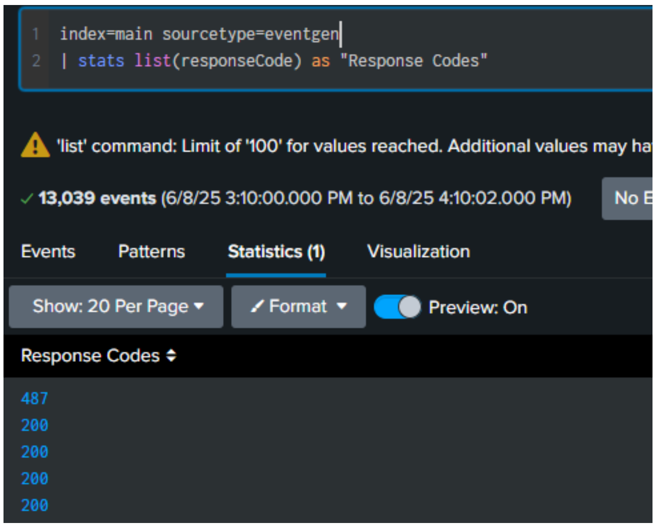
Figure 51: Using the avg function to get the average duration for all events and rename as AvgCallTime and grouped by the partner field.



AvgCallTime
2.7328524892440065
2.7746652935118434
2.7603192389006344
2.7963181818181813
2.7971918238993707
2.832600755667507
2.6884618952948967

Figure 52: AvgCallTime with the averages

Lastly, I explored the use of the list and values functions within the stats command. To begin, I used the list function to display all values in the responseCode field, including duplicates, and renamed the resulting column to Response Codes. In addition, I applied the values function to the same field to return only the unique response codes. This allowed me to compare both functions and understand how they handle duplicate versus distinct values in the data.



```
index=main sourcetype=eventgen
| stats list(responseCode) as "Response Codes"
```

'list' command: Limit of '100' for values reached. Additional values may ha

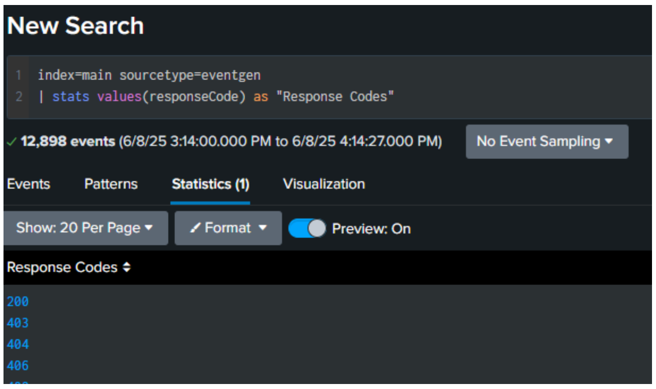
✓ 13,039 events (6/8/25 3:10:00.000 PM to 6/8/25 4:10:02.000 PM) No E

Events Patterns Statistics (1) Visualization

Show: 20 Per Page Format Preview: On

Response Codes
487 200 200 200 200

Figure 53: Listing all values in the responseCode.



New Search

```
index=main sourcetype=eventgen
| stats values(responseCode) as "Response Codes"
```

✓ 12,898 events (6/8/25 3:14:00.000 PM to 6/8/25 4:14:27.000 PM) No Event Sampling

Events Patterns Statistics (1) Visualization

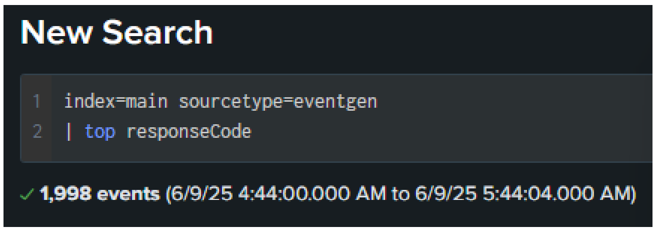
Show: 20 Per Page Format Preview: On

Response Codes
200 403 404 406

Figure 54: Applying the values function in the stats command to display only unique values in the responseCode field.

B. Top Command

The top command returns a statistical table showing the most common values for a specified field. By default, it displays the field name, the count of each value's occurrences, and the percentage those values represent out of the total. Additionally, the top command returns the top 10 results unless specified otherwise, making it useful for quickly identifying frequently occurring field values in a dataset.

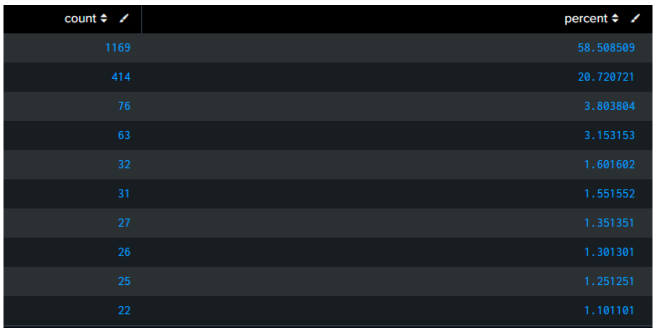


New Search

```
index=main sourcetype=eventgen
| top responseCode
```

✓ 1,998 events (6/9/25 4:44:00.000 AM to 6/9/25 5:44:04.000 AM)

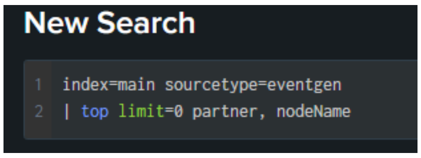
Figure 55: Using the top command in the search header.



count	percent
1169	58.508509
414	20.720721
76	3.803804
63	3.153153
32	1.601602
31	1.551552
27	1.351351
26	1.301301
25	1.251251
22	1.101101

Figure 56: Results of using the top command. Showing the count and percentage produced.

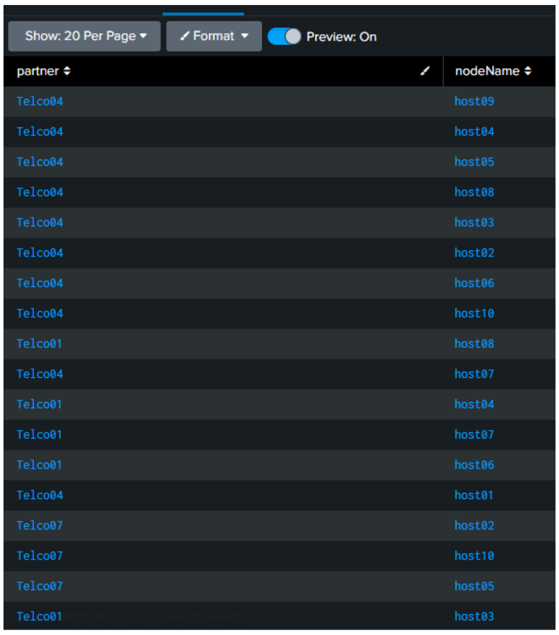
For the final part of this section, I wanted to display more than just the default top 10 results returned by the top command. To do this, I utilized the limit option. By setting limit=0, I was able to return all values for the specified field, rather than restricting the output to only the top 10.



New Search

```
index=main sourcetype=eventgen
| top limit=0 partner, nodeName
```

Figure 57: Using the top command with the limit function.

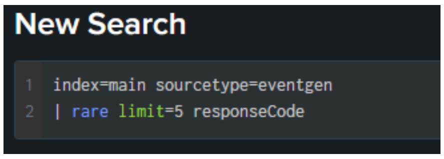

Show: 20 Per Page
Format
Preview: On

partner	nodeName
Telco04	host09
Telco04	host04
Telco04	host05
Telco04	host08
Telco04	host03
Telco04	host02
Telco04	host06
Telco04	host10
Telco01	host08
Telco04	host07
Telco01	host04
Telco01	host07
Telco01	host06
Telco04	host01
Telco07	host02
Telco07	host10
Telco07	host05
Telco01	host03

Figure 58: Applying limit=0 with the top command to display all values in the specified field, rather than restricting the output to the top 10 results.

C. *Rare Command*

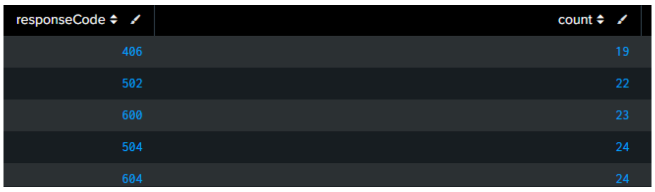
The rare command in Splunk returns a statistical table that highlights the least common values for a specified field. Similar to the top command, it provides the field name, count, and percentage, but focuses on values that occur infrequently within the dataset. This is especially useful for identifying anomalies or outliers.



New Search

```
index=main sourcetype=eventgen
| rare limit=5 responseCode
```

Figure 59: Using the rare command to find the five least common values in the responseCode field.



responseCode	count
406	19
502	22
600	23
504	24
604	24

Figure 60: Reults of the rare command showing the five least common values and their associated count.

D. *Formatting Statistics Tables*

In Splunk, you can apply formatting settings either to the entire statistics table or to individual columns. To explore this feature, I began by modifying the color formatting for the partner field. Located in the top-right corner of each column is a paintbrush icon, which provides customization options such as color schemes. Additionally, there is a number formatting section that allows for rounding and other numerical adjustments, enabling better visual clarity and presentation of data.

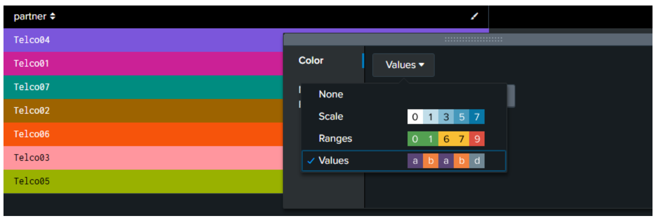
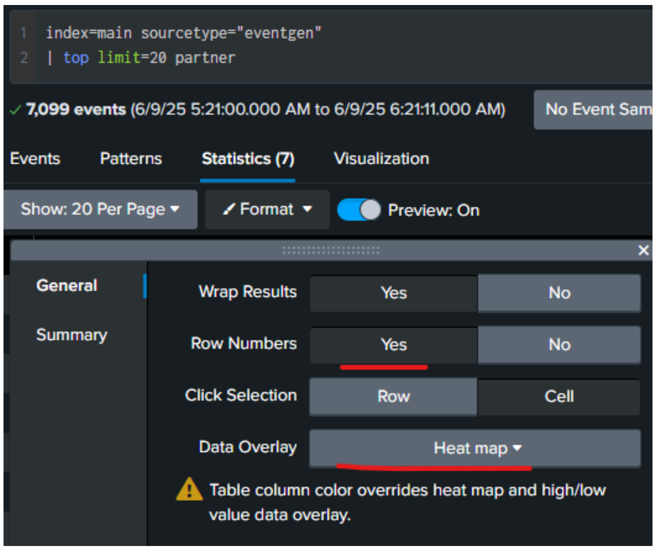


Figure 61: Changing the color of the partner field to values.



```
index=main sourcetype="eventgen"
| top limit=20 partner
```

✓ 7,099 events (6/9/25 5:21:00.000 AM to 6/9/25 6:21:11.000 AM)

Wrap Results: Yes | No

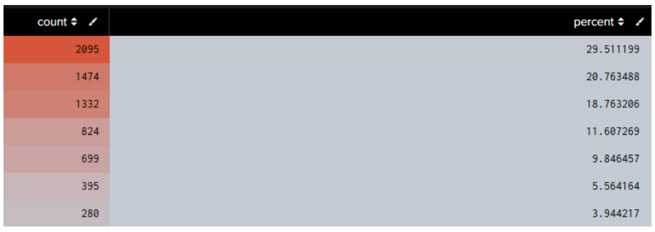
Row Numbers: [✓ Yes] | No

Click Selection: Row | Cell

Data Overlay: Heat map

Table column color overrides heat map and high/low value data overlay.

Figure 62: Changing the entire stats table by enabling row numbers and a heat map in the Data Overlay section.

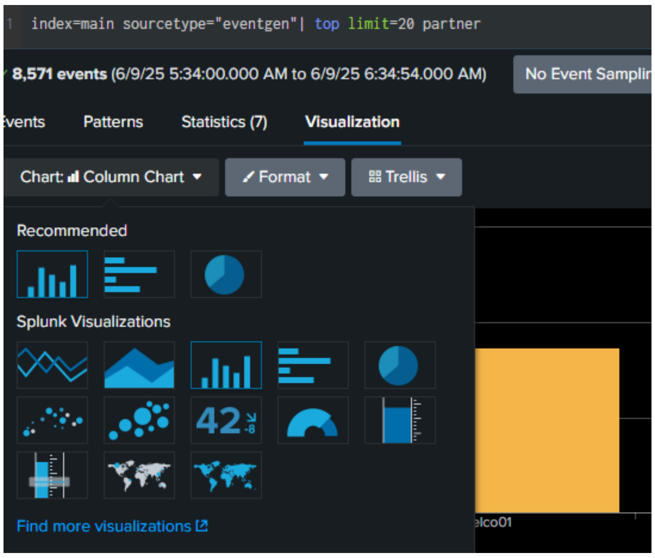


count	percent
2095	29.511199
1474	20.763488
1332	18.763206
824	11.607269
699	9.846457
395	5.564164
280	3.944217

Figure 63: Results of the statistics formatting changes, showcasing a heatmap applied to the partner field for enhanced visual analysis.

E. *Formatting Visualizations*

By default, Splunk selects a basic visualization for your data, but this can be changed based on your analysis needs. For example, you can switch to a column chart, pie chart, or other visualization types to better represent the data. This flexibility is important when trying to highlight patterns, trends, or distributions that may not be immediately obvious in a raw table format.



```
index=main sourcetype="eventgen"| top limit=20 partner
```

8,571 events (6/9/25 5:34:00.000 AM to 6/9/25 6:34:54.000 AM)

Figure 64: Selecting the different charts for the partner field.

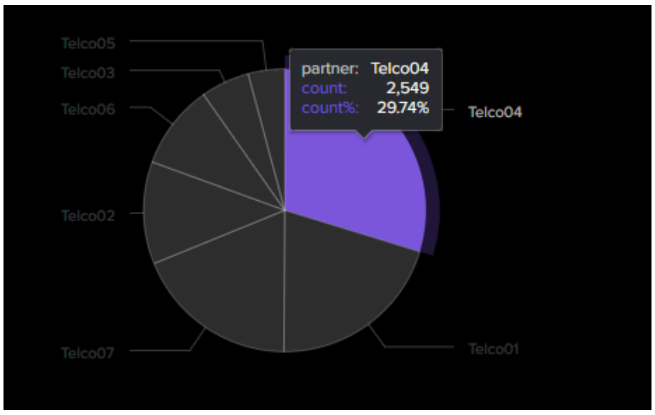


Figure 65: Selecting a pie chart as the visualization type to represent data distribution more clearly.

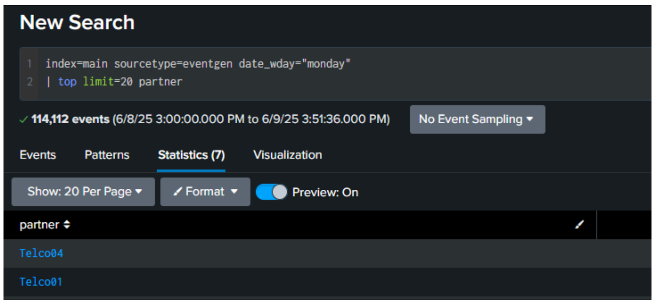
VIII. Creating Reports and Dashboards

Reports in Splunk are saved searches that preserve the results of events, statistical, or visualization-based queries. As previously mentioned in the lab, a search job remains in Splunk for only 10 minutes by default unless it is explicitly shared, in which case it remains available for up to one week. However, to retain search results beyond that time frame, users can create saved reports. These reports are especially useful for ongoing monitoring of key metrics across various domains, including security, IT operations, application performance, and business analytics.

Dashboards in Splunk are powerful tools used to present insights from data through visual indicators driven by underlying searches. They enable real-time monitoring of critical metrics across various domains, including IT operations, Security Information and Event Management (SIEM), application performance, and business analytics. Dashboards provide an intuitive and customizable interface for tracking key performance indicators and facilitating informed decision-making.

A. *Creating a Report*

As a best practice, it is recommended to use a consistent naming convention when creating knowledge objects in Splunk. A common format includes: <Group_Name>_<Object_Type>_<Description>. This approach improves organization, simplifies collaboration, and enhances searchability within large environments. In this example, I followed this convention to create a report focused on displaying the top 20 counts for the partner field.



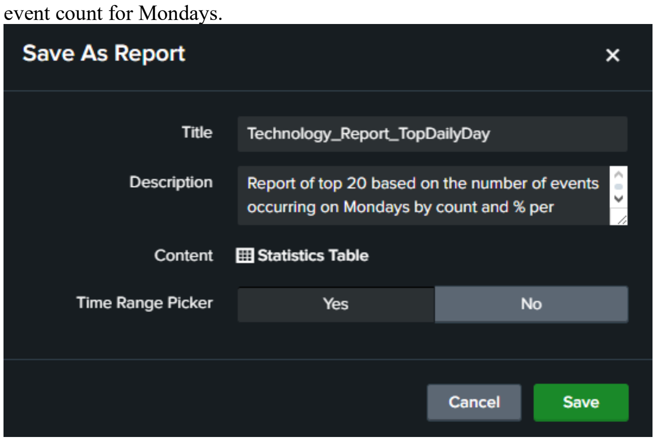
New Search

```
index=main sourcetype=eventgen date_wday="monday"
| top limit=20 partner
```

✓ 114,112 events (6/8/25 3:00:00.000 PM to 6/9/25 3:51:36.000 PM)

partner
Telco04
Telco01

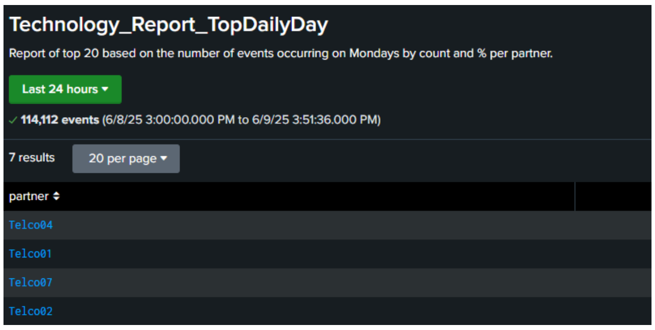
Figure 66: Executing a search to display the top 20 partner values based on event count for Mondays.



Save As Report

Title	Technology_Report_TopDailyDay
Description	Report of top 20 based on the number of events occurring on Mondays by count and % per
Content	Statistics Table
Time Range Picker	Yes [✓ No]

Figure 67: Creating a report for fig. 66.


Technology_Report_TopDailyDay

Report of top 20 based on the number of events occurring on Mondays by count and % per partner.

✓ 114,112 events (6/8/25 3:00:00.000 PM to 6/9/25 3:51:36.000 PM)

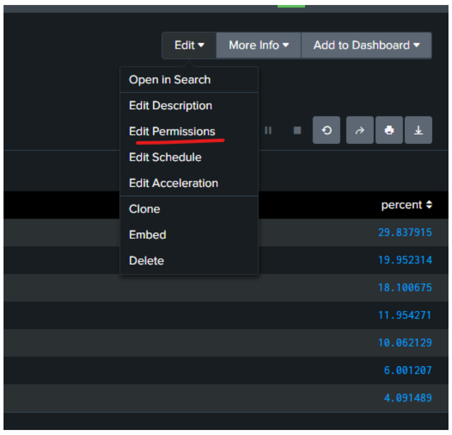
7 results

partner
Telco04
Telco01
Telco07
Telco02

Figure 68: Generated report displaying the top 20 partner values by count for events occurring on Mondays.

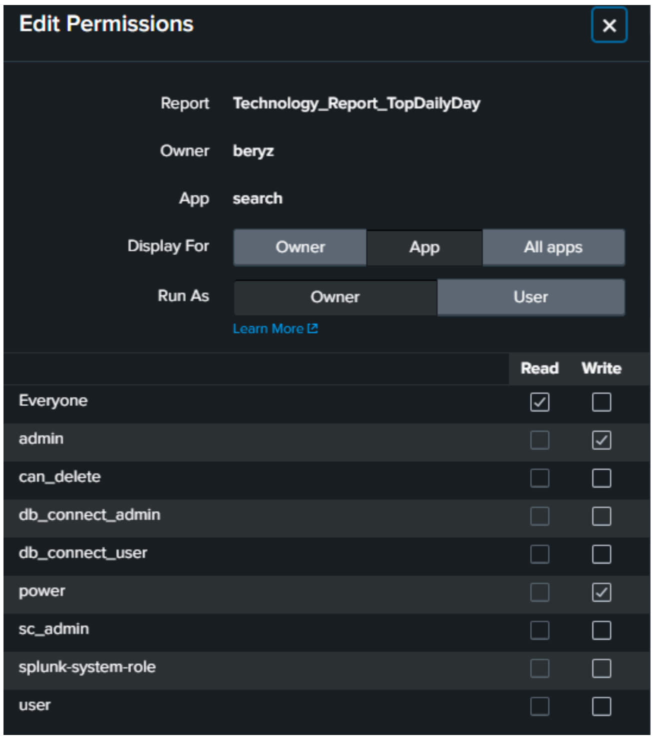
B. Managing Reports

With the report successfully created, I proceeded to manage its permissions by assigning specific access roles. By default, a newly created report in Splunk is only accessible to its owner. To make it available to others, I shared the report as a knowledge object. In this case, I configured the permissions so that all users could view the report, while only users with the power role were granted write access, allowing them to make modifications.



percent
29.837915
19.952314
18.100675
11.954271
10.062129
6.001207
4.091489

Figure 69: Selecting the edit drop down selections to edit the permissions of the knowledge object.


Edit Permissions

Report Technology_Report_TopDailyDay

Owner beryz

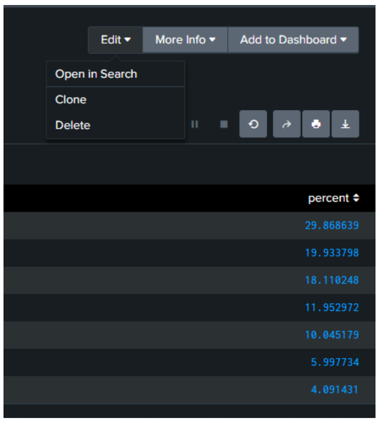
App search

Display For: Owner | App | All apps

Run As: Owner | User

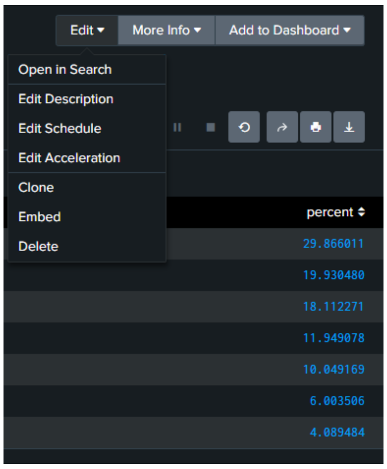
	Read	Write
Everyone	✓	
admin		✓
can_delete		
db_connect_admin		
db_connect_user		
power		✓
sc_admin		
splunk-system-role		
user		

Figure 70: Editing the permissions for the Technology_Report_TopDailyDay so that everyone is able to read the report but only power users and above are able to write and make changes into the report.



percent
29.868639
19.933798
18.110248
11.952972
10.045179
5.997734
4.091431

Figure 71: While logged in as Jack Richard (user role), I confirmed that I was unable to make changes to the report. This validated that users with the user role and roles with fewer privileges do not have write access, effectively enforcing the principle of least privilege.

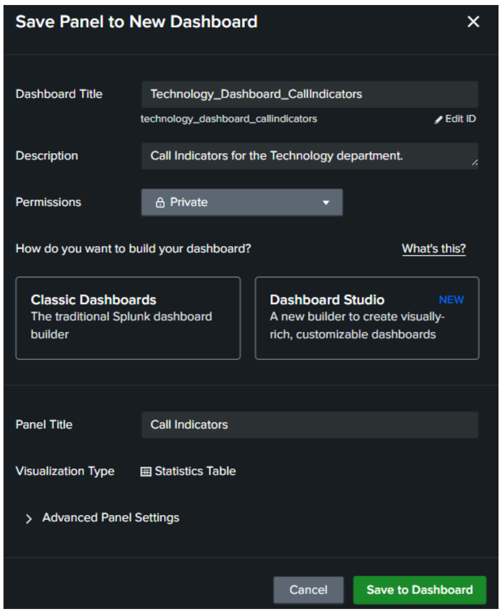


percent
29.866011
19.930480
18.112271
11.949078
10.049169
6.003506
4.089484

Figure 72: While logged in as Cole Gilstrap (Power user role), I confirmed that he had write access to the report but could not modify owner-level permissions for the knowledge object, maintaining appropriate access boundaries.

C. *Creating a dashboard*

In this section, I aimed to create a dashboard focused on call indicators by utilizing the top command on the fields deviceMAC, userIPAddress, and callerNumber.


Save Panel to New Dashboard

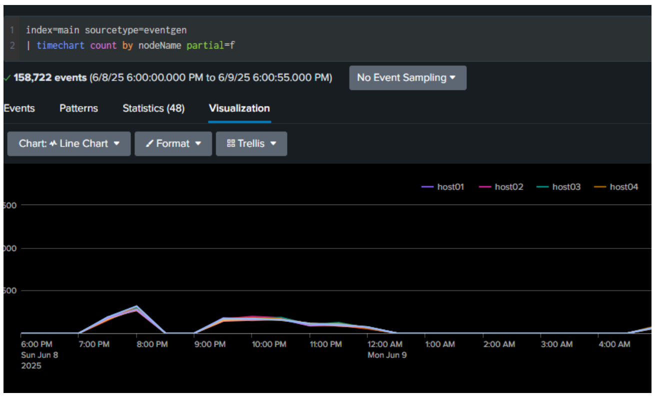
Dashboard Title: Technology_Dashboard_CallIndicators
technology_dashboard_callindicators
Description: Call Indicators for the Technology department.
Permissions: Private
How do you want to build your dashboard?

What's this?
Edit ID
Classic Dashboards
The traditional Splunk dashboard builder
Dashboard Studio NEW
A new builder to create visually-rich, customizable dashboards

Panel Title: Call Indicators
Visualization Type: Statistics Table

Advanced Panel Settings
Cancel
Save to Dashboard

Figure 73: Creating a dashboard for call Indicators.



```
index=main sourcetype=eventgen
| timechart count by nodeName partial=f
```

✓ 158,722 events (6/8/25 6:00:00.000 PM to 6/9/25 6:00:55.000 PM)

Figure 74: Creating a new visualizations to add to the dashboard Technology_Dashboard_CallIndicators.

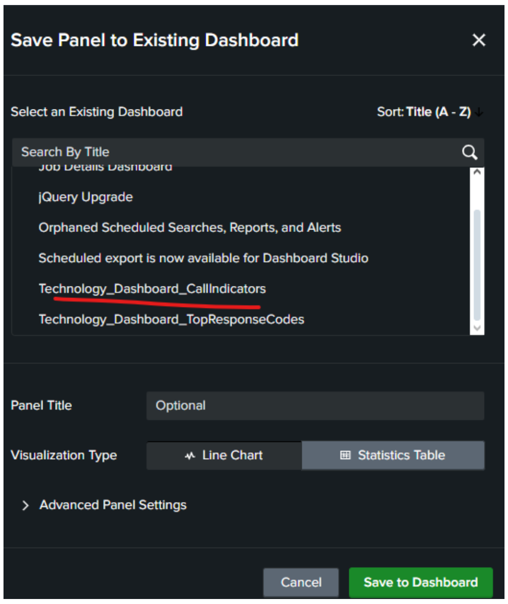
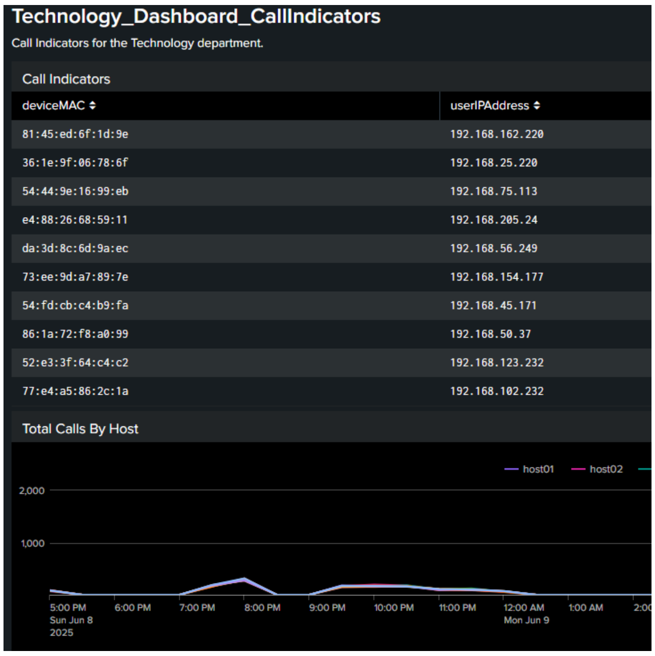

Save Panel to Existing Dashboard
Select an Existing Dashboard
Sort: Title (A - Z)
Search By Title
Job Details Dashboard
jQuery Upgrade
Orphaned Scheduled Searches, Reports, and Alerts
Scheduled export is now available for Dashboard Studio
Technology_Dashboard_CallIndicators
Technology_Dashboard_TopResponseCodes
Panel Title
Optional
Visualization Type
Line Chart
Statistics Table
Advanced Panel Settings
Cancel
Save to Dashboard

Figure 75: Saved the panel to the existing dashboard.



Technology_Dashboard_CallIndicators

Call Indicators for the Technology department.

Call Indicators

deviceMAC	userIPAddress
81:45:ed:6f:1d:9e	192.168.162.220
36:1e:9f:06:78:6f	192.168.25.220
54:44:9e:16:99:eb	192.168.75.113
e4:88:26:68:59:11	192.168.205.24
da:3d:8c:6d:9a:ec	192.168.56.249
73:ee:9d:a7:89:7e	192.168.154.177
54:fd:cb:c4:b9:fa	192.168.45.171
86:1a:72:f8:a0:99	192.168.50.37
52:e3:3f:64:c4:c2	192.168.123.232
77:e4:a5:86:2c:1a	192.168.102.232

Total Calls By Host

Figure 76: Dashboard display showcasing two distinct search results, each represented in its own panel with appropriately labeled titles.

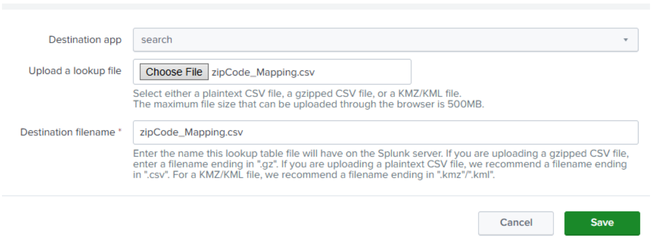
IX. Creating and Using Lookups

A lookup is a method used to enrich data by adding new fields from an external lookup table. One common type is the CSV (file-based) lookup, which uses external CSV files. These are typically applied to small, static datasets and are the most widely used lookup type in Splunk. Another type is the KV (Key-Value) Store Collection lookup, which retrieves data from a KV store collection. This method is well-suited for large lookup tables that contain frequently changing data. External or scripted lookups provide additional flexibility by allowing data enrichment through

Python scripts or binary executables, rather than relying on file-based sources. Lastly, geospatial lookups are used to enrich data with geographic information, such as boundaries of mapped regions including countries, states, or counties.

A. Creating a CSV Lookup

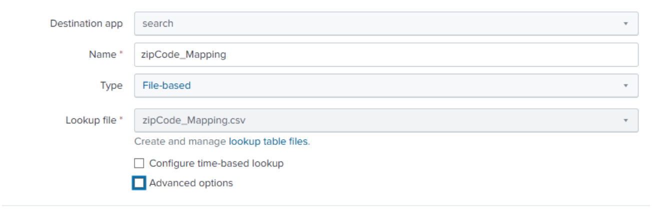
Within the Settings menu under the Knowledge section, there is an option for Lookups. When selected, it prompts you to upload a lookup file and specify a destination filename. To keep the process straightforward, I uploaded a CSV file named zipCode_Mapping.csv and used the same name for the destination file.



Destination app	search
Upload a lookup file	Choose File zipCode_Mapping.csv
	Select either a plaintext CSV file, a gzipped CSV file, or a KMZ/KML file. The maximum file size that can be uploaded through the browser is 500MB.
Destination filename *	zipCode_Mapping.csv
	Enter the name this lookup table file will have on the Splunk server. If you are uploading a gzipped CSV file, enter a filename ending in ".gz". If you are uploading a plaintext CSV file, we recommend a filename ending in ".csv". For a KMZ/KML file, we recommend a filename ending in ".kmz"/".kml".

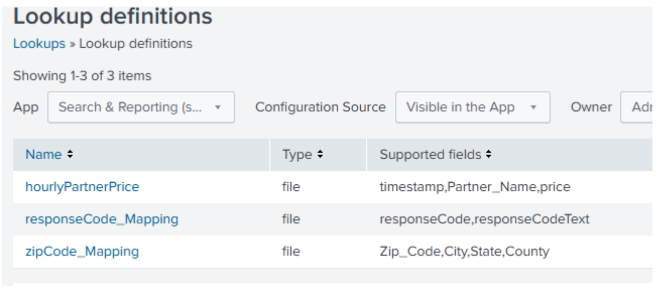
Figure 77: Uploading and Creating the Lookup File (zipCode_Mapping.csv) in the Lookups Configuration Page.

To use a lookup table in Splunk, the lookup file must first be uploaded, followed by the creation of a lookup definition. In this next section, I created a lookup definition.



Destination app	search
Name *	zipCode_Mapping
Type	File-based
Lookup file *	zipCode_Mapping.csv
	Create and manage lookup table files.
	☐ Configure time-based lookup
	☐ Advanced options

Figure 78: Defining the Lookup by Creating a Lookup Definition for zipCode_Mapping.csv



Lookup definitions

Lookups » Lookup definitions

Showing 1-3 of 3 items

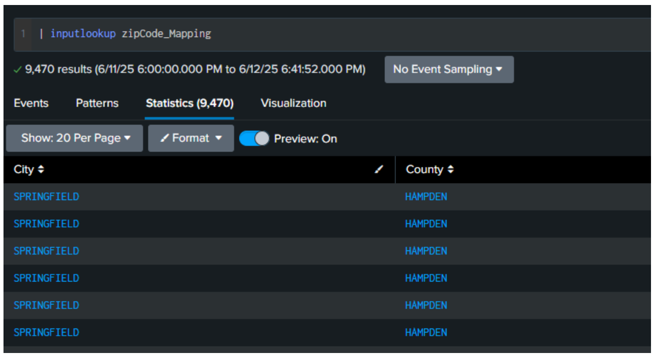
App Search & Reporting (s...
Configuration Source Visible in the App
Owner Adr

Name	Type	Supported fields
hourlyPartnerPrice	file	timestamp,Partner_Name,price
responseCode_Mapping	file	responseCode,responseCodeText
zipCode_Mapping	file	Zip_Code,City,State,County

Figure 79: List of lookup defintions created in the instance.

A. Inputlookup Command

The inputlookup command is used to review and validate the contents of a lookup table. In this example, I used the pipe character to run the following search: | inputlookup zipCode_Mapping, which displays the contents of the zipCode_Mapping.csv file.



```
| inputlookup zipCode_Mapping
```

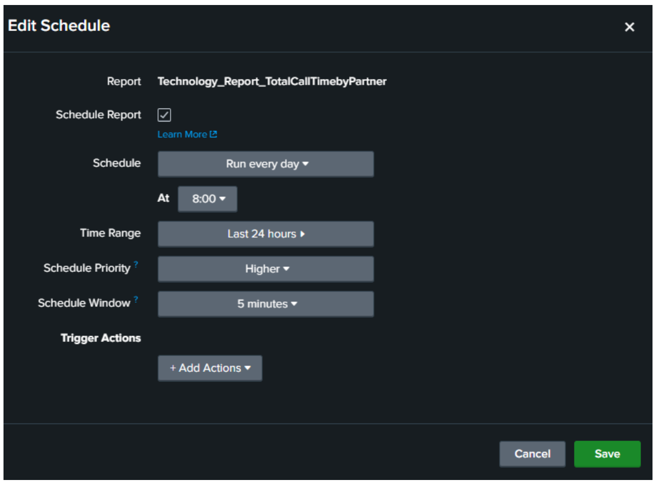
✓ 9,470 results (6/11/25 6:00:00.000 PM to 6/12/25 6:41:52.000 PM)

City	County
SPRINGFIELD	HAMPDEN
SPRINGFIELD	HAMPDEN
SPRINGFIELD	HAMPDEN
SPRINGFIELD	HAMPDEN
SPRINGFIELD	HAMPDEN
SPRINGFIELD	HAMPDEN

Figure 80: Displaying the contents of zipCode_Mapping.

X. Creating Scheduled Reports

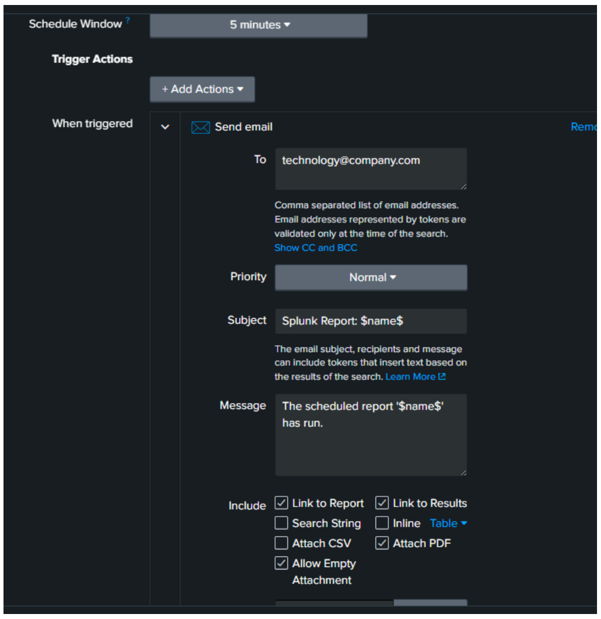
Scheduled reports in Splunk are useful for automating the delivery of search results at defined intervals. A common use case is to keep IT teams informed of key metrics. In this example, I created a report titled Technology_Report_TotalCallTimebyPartner and configured it to run every morning at 8:00 AM with a 24-hour time range and assigned it as a high schedule priority, with a 5-minute schedule window.



Edit Schedule

Report	Technology_Report_TotalCallTimebyPartner
Schedule Report	✓
Schedule	Run every day
At	8:00
Time Range	Last 24 hours
Schedule Priority	Higher
Schedule Window	5 minutes
Trigger Actions	+ Add Actions

Figure 81: Configuring the scheduled report with a time range of the last 24 hours, set to run daily at 8:00 AM, with a schedule window of 5 minutes and a priority level set to "Higher."


Schedule Window: 5 minutes

Trigger Actions

+ Add Actions

When triggered: Send email

To: technology@company.com

Comma separated list of email addresses. Email addresses represented by tokens are validated only at the time of the search.

Show CC and BCC

Priority: Normal

Subject: Splunk Report: $name$

The email subject, recipients and message can include tokens that insert text based on the results of the search. Learn More

Message: The scheduled report '$name$' has run.

Include: ☑ Link to Report ☑ Link to Results ☐ Search String ☐ Inline Table ☐ Attach CSV ☑ Attach PDF ☑ Allow Empty Attachment

Figure 82: Sending the report to an email.

References

[1] "Online Courses - Learn Anything, On Your Schedule | Udemy," *Udemy*, 2025. https://www.udemy.com/course/the-complete-splunk-core-certified-user-course-splk-1001/learn/lecture/40793984?start=0#overview (accessed May 27, 2025).

[2] Use fields to search. (2025). Retrieved from https://docs.splunk.com/Documentation/Splunk/9.4.2/SearchTutorial/Usefieldstosearch