Lab Report — Search Commands Deep Dive: Part 1

Name: Dhruvish
Date: October 24, 2025
Platform: Splunk Cloud Trial

Objective

The objective of this lab was to gain proficiency with the most frequently used **Splunk transforming search commands** — count, values(), dc(), and avg().
You practiced grouping data using by, shaping result tables, and creating safe numeric fields for averages with eval.
These foundational skills are essential for data summarization, reporting, and dashboard development in any SOC or IT analytics workflow.

Tools Used

- **Splunk Cloud Trial** (or Splunk Free)
- **Web Browser**

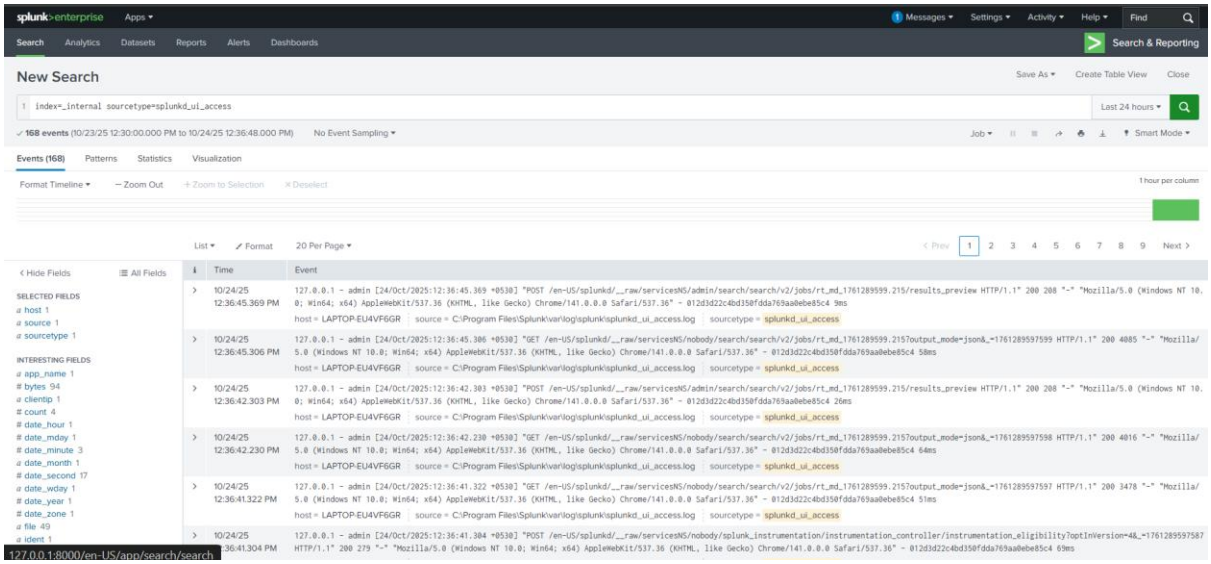
Procedure and Observations

Step 1: Pick a Dataset & Set Time

- Navigated to **Apps ▸ Search & Reporting**.
- Set the time range to **Last 24 hours**.
- Selected the preferred dataset:
- index=_internal sourcetype=splunkd_ui_access

(Fallback: index=_internal sourcetype=splunkd if needed)

- This dataset includes web-access-like fields such as user, status, uri_path, and occasionally bytes.

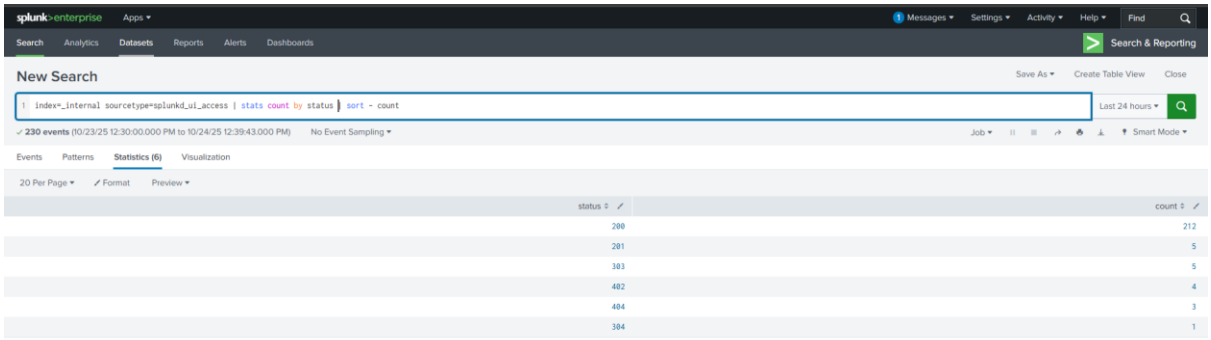


Step 2: Using count — How Many?

- Found the total number of events by sourcetype:

  index=_internal | stats count by sourcetype
- Narrowed the view to UI access logs, grouped by HTTP status:

  index=_internal sourcetype=splunkd_ui_access | stats count by status | sort - count
- Observed which status codes appeared most frequently.

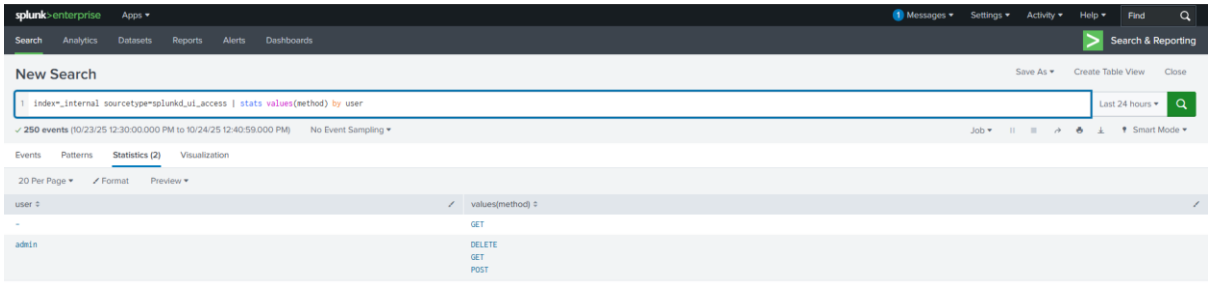


Step 3: Using values() — Unique Field Values

- Displayed all unique HTTP methods used by each user:

  index=_internal sourcetype=splunkd_ui_access | stats values(method) by user
- Listed all distinct URI paths seen per status code:

  index=_internal sourcetype=splunkd_ui_access | stats values(uri_path) as paths by status



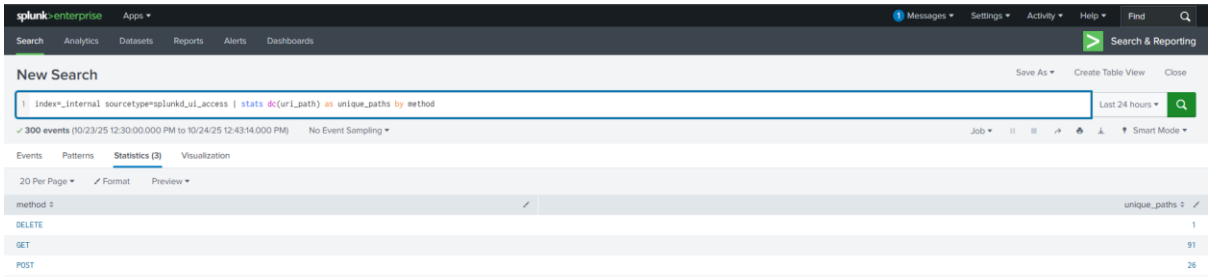
Step 4: Using dc() — Distinct Counts

- Counted distinct users per HTTP status:

  index=_internal sourcetype=splunkd_ui_access | stats dc(user) as unique_users by status

- Counted distinct URI paths used by each method:

  index=_internal sourcetype=splunkd_ui_access | stats dc(uri_path) as unique_paths by method



Step 5: Using avg() — Creating a Safe Numeric Field

Because this dataset lacks a built-in numeric metric like “response time,” created a proxy numeric value to calculate averages:

- Used event raw length as a stand-in metric:

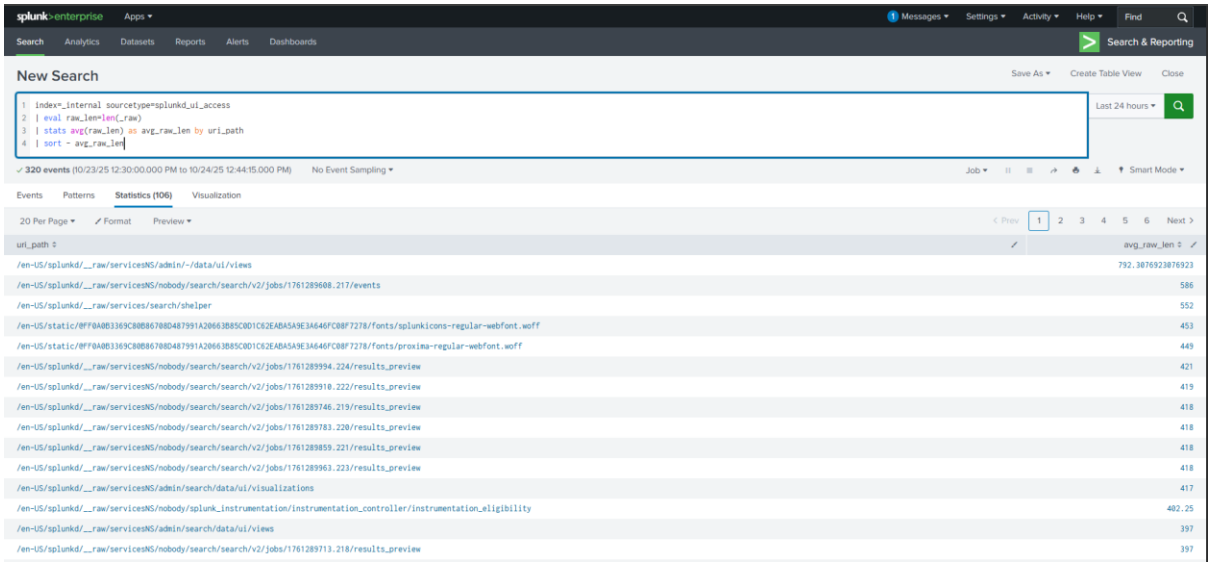
  index=_internal sourcetype=splunkd_ui_access

  | eval raw_len = len(_raw)

  | stats avg(raw_len) as avg_raw_len by uri_path

  | sort - avg_raw_len

- Observed which URIs had the largest average event size.



Step 6: Combining Multiple Metrics in One stats

Created a single stats command combining count, distinct, and average metrics for a richer overview.

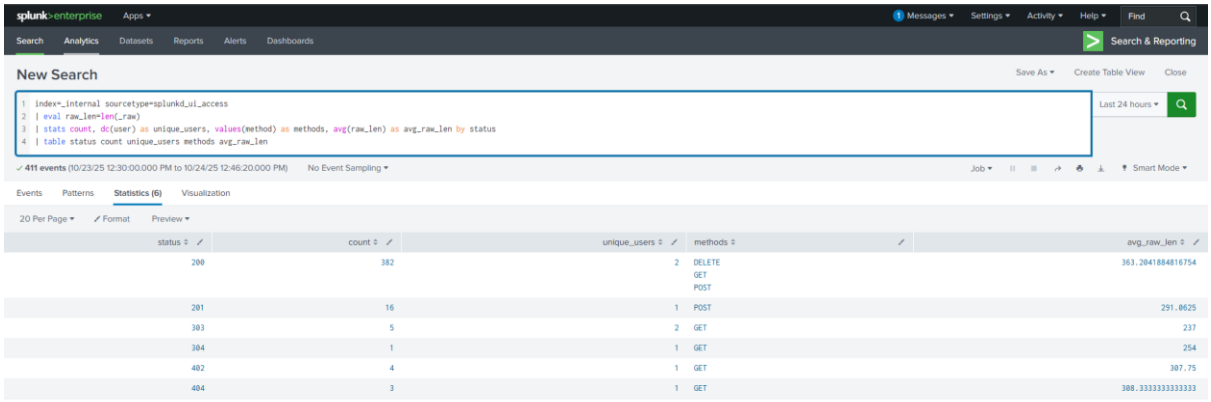
index=_internal sourcetype=splunkd_ui_access

| eval raw_len=len(_raw)

| stats count, dc(user) as unique_users, values(method) as methods, avg(raw_len) as avg_raw_len by status

| table status count unique_users methods avg_raw_len

- Produced a well-organized table summarizing activity per status code.
- This combined view is excellent for dashboards or health summaries.



Step 7: Grouping Strategy & Filters

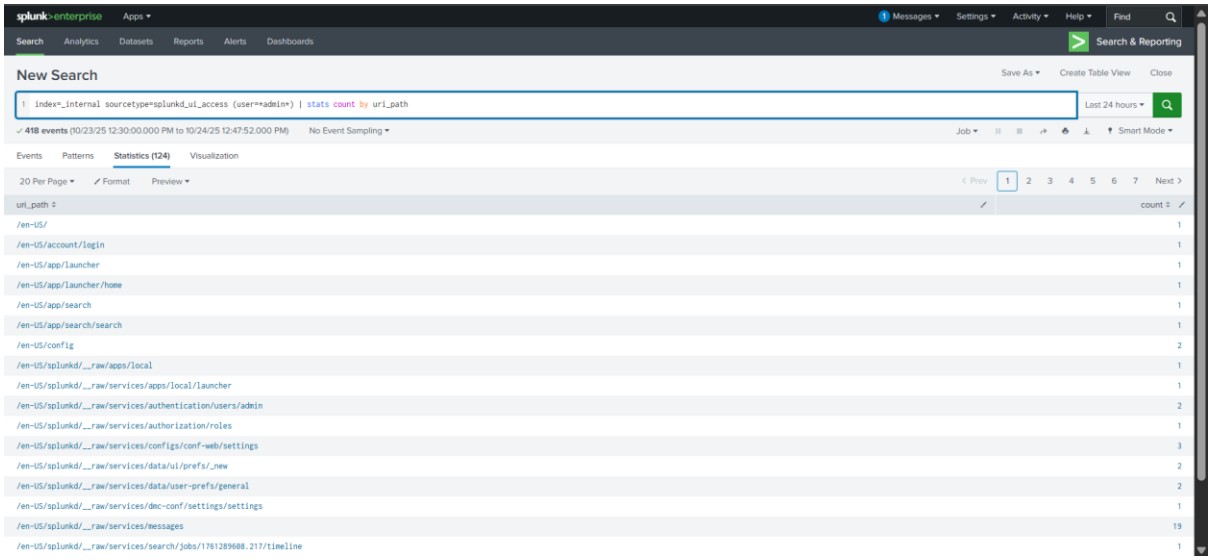
- Identified top users:

  index=_internal sourcetype=splunkd_ui_access | top user limit=5

- Performed deep-dive on a specific user (from the top list):

- ndex=_internal sourcetype=splunkd_ui_access user=<username> | stats count by uri_path
- Applied post-aggregation filtering:

  ... | stats count by uri_path | where count > 10



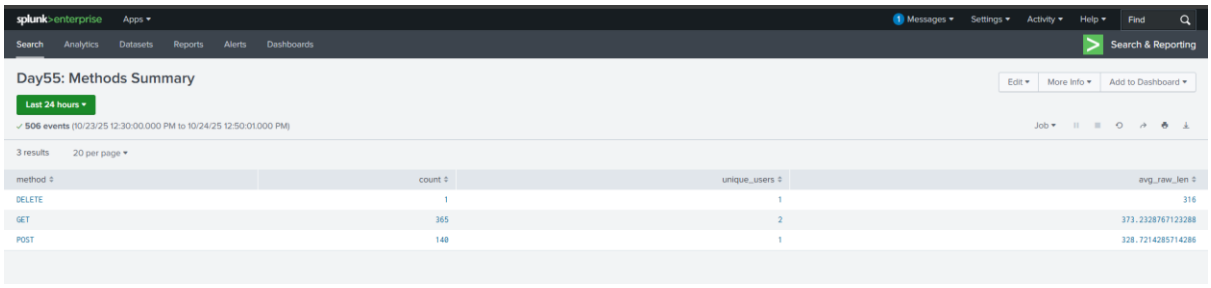
Step 8: Save a Reusable Report

- Created a reusable report summarizing methods and averages:

  index=_internal sourcetype=splunkd_ui_access

  | eval raw_len=len(_raw)

  | stats count, dc(user) as unique_users, avg(raw_len) as avg_raw_len by method
- Saved report as: **Day55: Methods Summary**.
- Added description and verified it under *Reports*.



Reflection

- **When is values() helpful vs. overwhelming?**
  values() is great for small sets (like listing HTTP methods per user), but can become unreadable when there are too many unique values (e.g., hundreds of URI paths).

- **What business question does dc() answer that count cannot?**
  count measures total volume; dc() measures **unique participation** (e.g., number of unique users or endpoints).
- **What numeric field would you average in a real environment?**
  I'd average **response_time**, **bytes**, or **duration** to monitor performance trends. If unavailable, proxy metrics like _raw length or calculated fields can still provide relative insights.

Summary

In this lab, I explored Splunk's most fundamental transforming commands:

- **count** for totals,
- **values()** for unique listings,
- **dc()** for distinct counts, and
- **avg()** for averages.

By combining these with grouping, field shaping, and report creation, I built the foundation for quantitative analysis in Splunk — a key skill for generating dashboards, alerts, and SOC insights.