Leveraging Lookups and Subsearches – Lab Solutions Guide

Overview

Welcome to the Splunk Education lab environment. These lab exercises will test your knowledge of lookup commands and subsearches.

Scenario

You will use data from the international video game company, Buttercup Games. A list of source types is provided below.

NOTE: This is a lab environment driven by data generators with obvious limitations. This is not a production environment. Screenshots approximate what you should see, not the **exact** output.

Index	Type	Sourcetype	Interesting Fields
web	Online sales	**access_combined**	action, bytes, categoryId, clientip, itemId, JSESSIONID, price, productId, product_name, referer, referer_domain, sale_price, status, user, useragent
security	Active Directory	**winauthentication_security**	LogName, SourceName, EventCode, EventType, User
	Badge reader	**history_access**	Address_Description, Department, Device, Email, Event_Description, First_Name, last_Name, Rfid, Username
	Web server	**linux_secure**	action, app, dest, process, src_ip, src_port, user, vendor_action
sales	Retail sales	**vendor_sales**	categoryId, product_name, productId, sale_price, Vendor, VendorCity, VendorCountry, VendorID, VendorStateProvince
network	Email security data	**cisco_esa**	dcid, icid, mailfrom, mailto, mid
	Web security appliance data	**cisco_wsa_squid**	action, cs_method, cs_mime_type, cs_url, cs_username, sc_bytes, sc_http_status, sc_result_code, severity, src_ip, status, url, usage, x_mcafee_virus_name, x_wbrs_score, x_webcat_code_abbr
	Firewall data	**cisco_firewall**	bcg_ip, dept, Duration, fname, IP, lname, location, rfid, splunk_role, splunk_server, Username

splunk>

Common Commands and Functions

These commands and statistical functions are commonly used in searches but may not have been explicitly discussed in the course. Please use this table for quick reference. Click on the hyperlinked SPL (Search Processing Language) to be taken to the Search Manual for that command or function.

SPL	Type	Description	Example
`sort`	command	Sorts results in descending or ascending order by a specified field. Can limit results to a specific number.	*Sort the first 100 `src_ip` values in descending order* `| sort 100 -src_ip`
`where`	command	Filters search results using eval-expressions.	*Return events with a `count` value greater than 30* `| where count > 30`
`rename`	command	Renames one or more fields.	*Rename `SESSIONID` to 'The session ID'* `| rename SESSIONID as "The session ID"`
`fields`	command	Keeps (+) or removes (-) fields from search results.	*Remove the `host` field from the results* `| fields - host`
`stats`	command	Calculates aggregate statistics over the results set.	*Calculate the total sales, i.e. the sum of `price` values.* `| stats sum(price)`
`eval`	command	Calculates an expression and puts the resulting value into a new or existing field.	*Concatenate `first_name` and `last_name` values with a space to create a field called "`full_name`"* `| eval full_name=first_name." ".last_name`
`table`	command	Returns a table.	*Output `vendorCountry`, `vendor`, and `sales` values to a table* `| table vendorCountry, vendor, sales`
`sum()`	statistical function	Returns the sum of the values of a field. Can be used with **`stats`**, **`timechart`**, and **`chart`** commands.	*Calculate the sum of the `bytes` field* `| stats sum(bytes)`
`count` or `count()`	statistical function	Returns the number of occurrences of all events or a specific field. Can be used with **`stats`**, **`timechart`**, and **`chart`** commands.	*Count all events as "events" and count all events that contain a value for `action` as "action"* `| stats count as events, count(action) as action`

Refer to the Search Reference Manual for a full list of commands and functions.

Lab Exercise 1 – Using Lookup Commands

Description

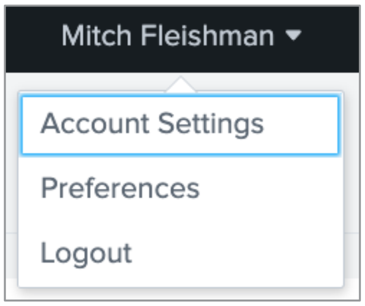
Configure the lab environment user account. Then, use **`inputlookup`**, **`lookup`**, and **`outputlookup`** commands to call on and create lookups in search.

Steps

Task 1: Log into Splunk and change the account name and time zone.

Set up your lab environment to fit your time zone. This allows the instructor to track your progress and assist you if necessary.

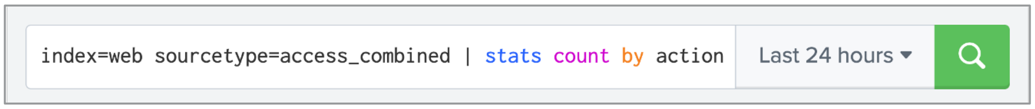
1. Log into your Splunk lab environment using the username and password provided to you.
2. You may see a pop-up window welcoming you to the lab environment. You can click **Continue to Tour** but this is not required. Click **Skip** to dismiss the window.
3. Click on the username you logged in with (at the top of the screen) and then choose **Account Settings** from the drop-down menu.
4. In the **Full name** box, enter your first and last name.
5. Click **Save**.
6. Reload your browser to reflect the recent changes to the interface. (This area of the web interface will be referred to as ***user name***.)



After you complete step 6, you will see your name in the web interface.

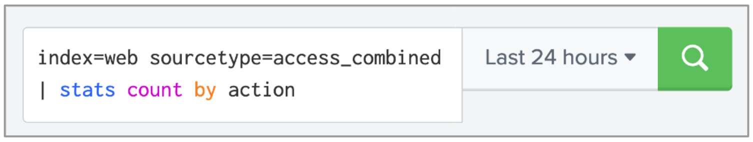
NOTE: Sometimes there is a delay in executing an action like saving in the UI or returning results of a search. Please allow the UI a few minutes to execute your action.

7. Navigate to ***user name*** **> Preferences.**
8. Choose your local time zone from the **Time zone** drop-down menu.
9. Click **Apply**.
10. (Optional) Navigate to ***user name*** **> Preferences > SPL Editor > Search auto-format** and click on the toggle to activate auto-formatting. Then click **Apply**. When the pipe character (**|**) is used in search, the SPL Editor will automatically begin the pipe on a new line.



```
index=web sourcetype=access_combined | stats count by action
```

Search auto-format disabled.



```
index=web sourcetype=access_combined
| stats count by action
```

Search auto-format enabled.

splunk>

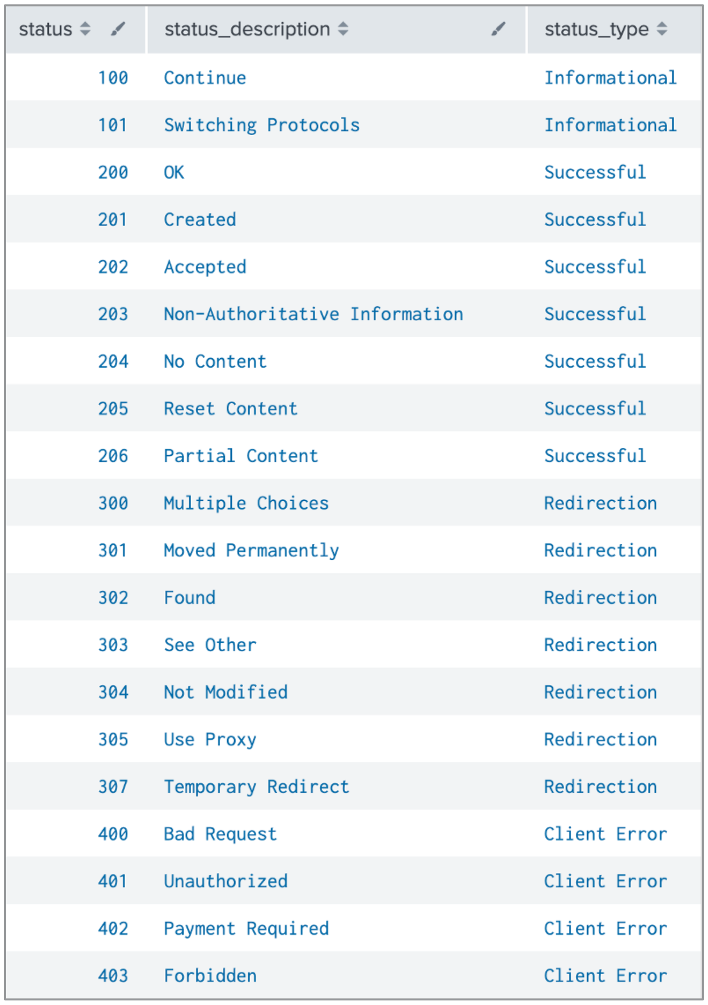
Scenario: You provided your knowledge manager with a CSV containing HTTP statuses, status descriptions, and status types. Your knowledge manager just informed you that the lookup was uploaded.

Task 2: Verify that a lookup has been uploaded correctly.

11. Your lab environment is configured to take you to the **Search & Reporting** app within Splunk. (Also called the "search" app.) Confirm you are in the correct app by clicking **Apps** in the top left corner. You should see **Search & Reporting** highlighted. If you do not, click on **Search & Reporting**.
12. Your knowledge manager provided you with the following information about the **`status_definitions.csv`** lookup. Use the **`inputlookup`** command to verify that the file-based lookup has been correctly uploaded.

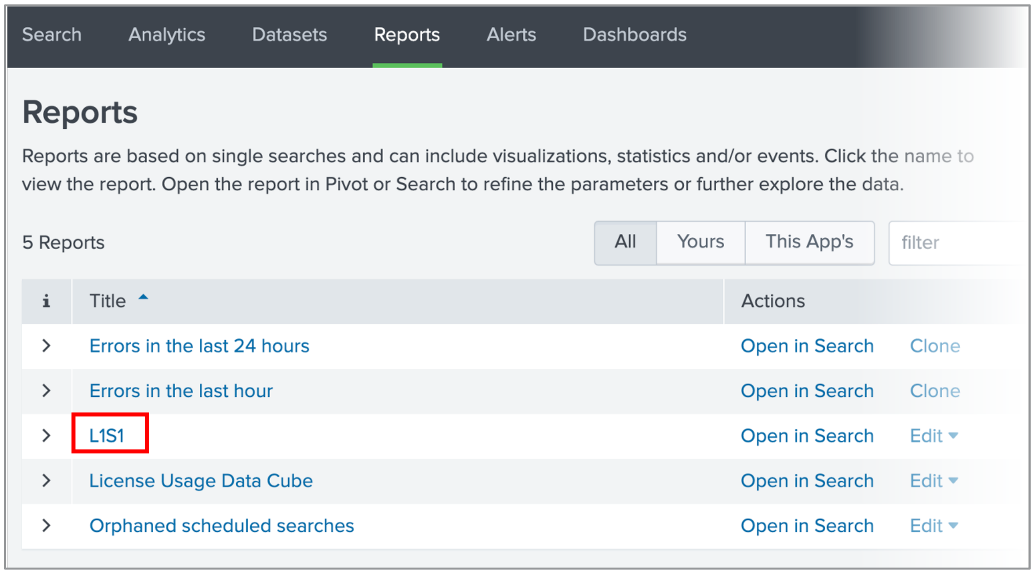
    filename: status_definitions.csv
    definition name: status_definitions_lookup
    lookup type: file-based

    You can view the contents of status_definitions.csv with either of these searches:
    ```
    | inputlookup status_definitions.csv
    | inputlookup status_definitions_lookup
    ```
13. Confirm that your search output matches the lookup contents:



status	status_description	status_type
100	Continue	Informational
101	Switching Protocols	Informational
200	OK	Successful
201	Created	Successful
202	Accepted	Successful
203	Non-Authoritative Information	Successful
204	No Content	Successful
205	Reset Content	Successful
206	Partial Content	Successful
300	Multiple Choices	Redirection
301	Moved Permanently	Redirection
302	Found	Redirection
303	See Other	Redirection
304	Not Modified	Redirection
305	Use Proxy	Redirection
307	Temporary Redirect	Redirection
400	Bad Request	Client Error
401	Unauthorized	Client Error
402	Payment Required	Client Error
403	Forbidden	Client Error

14. Save your search as a report with the name **L1S1**.
    a. Click **Save As** > **Report**
    b. For **Title**, enter L1S1.
    c. **Save**.
    d. You can **View** your report or exit out of the **Your Report Has Been Created** window by clicking the **X** in the upper-right corner.
    e. You can access your saved reports using the **Reports** tab in the application bar.
    f. Re-initialize the search window by clicking **Search** in the application bar. Perform this step each time you save a search to avoid accidentally editing your recently saved search.



Reports

Reports are based on single searches and can include visualizations, statistics and/or events. Click the name to view the report. Open the report in Pivot or Search to refine the parameters or further explore the data.

5 Reports

i	Title	Actions	
>	Errors in the last 24 hours	Open in Search	Clone
>	Errors in the last hour	Open in Search	Clone
>	L1S1	Open in Search	Edit ▾
>	License Usage Data Cube	Open in Search	Edit ▾
>	Orphaned scheduled searches	Open in Search	Edit ▾

*Your recently saved **L1S1** report will be visible in the **Reports** tab.*

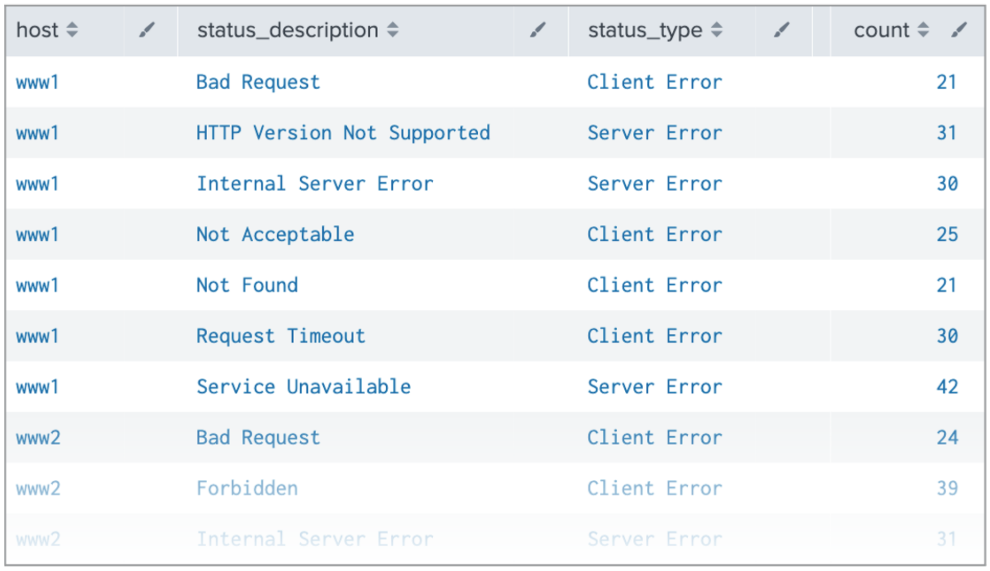
Task 3: Use the `status_definitions.csv` lookup in a search.

15. The following search needs to find events from the online sales data that do not have a **`status`** of **`200`**. (This represents unsuccessful events and is written as **`status!=200`**.) However, the search is missing the **`lookup`** command.

```
index=web sourcetype=access_* status!=200
| stats count by host, status_description, status_type
```

    a. Run the search above over the **Last 24 hours**. You should receive an error or **No results found.**
    b. Use the **`lookup`** command to add the **`status_description`** and **`status_type`** fields from the **`status_definitions_lookup`**. Then, run the search again.

```
index=web sourcetype=access_* status!=200
| lookup status_definitions_lookup status OUTPUT status_description, status_type
| stats count by host, status_description, status_type
```



host	status_description	status_type	count
www1	Bad Request	Client Error	21
www1	HTTP Version Not Supported	Server Error	31
www1	Internal Server Error	Server Error	30
www1	Not Acceptable	Client Error	25
www1	Not Found	Client Error	21
www1	Request Timeout	Client Error	30
www1	Service Unavailable	Server Error	42
www2	Bad Request	Client Error	24
www2	Forbidden	Client Error	39
www2	Internal Server Error	Server Error	31

16. Save your search as a report with the name **L1S2.**

Scenario: SecOps wants a report of known users who have been browsing "Uncategorized URLs" over the last 24 hours.

Task 4: Use information from the `knownusers.csv` and `status_definitions.csv` lookups to complete a search that will generate a report of users who have accessed uncategorized URLs over the last 24 hours. The report should include the users' departments, the URL accessed, and the associated http status and description.

17. This task uses information from the **`status_definitions.csv`** lookup and the **`knownusers.csv`** lookup. Your knowledge manager has provided you with the following information about the **`knownusers.csv`** lookup. Use the **`inputlookup`** command to explore the **`knownusers.csv`** file-based lookup.

    filename: knownusers.csv
    definition name: none
    lookup type: file-based

```
| inputlookup knownusers.csv
```

HINT: You may find it helpful to have both lookups available to reference for this task. Right-click on **Search** in the application bar (next to **Datasets**, **Reports**, etc.) and click "Open Link in New Tab." Run an **`inputlookup`** search on **`status_definitions.csv`**. Repeat these steps for **`knownusers.csv`**.

18. Complete the **`<missing>`** portions of the following search:

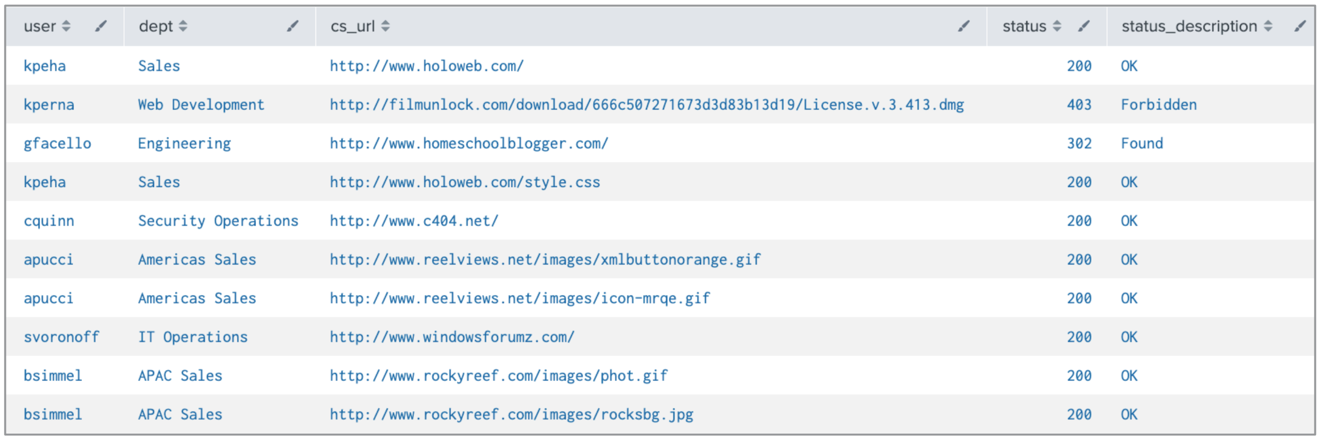
```
index=network sourcetype=cisco_wsa_squid x_webcat_code_full="Uncategorized URLs"
| lookup knownusers.csv <missing>
| lookup status_definitions.csv <missing>
| search user=*
| table user, dept, cs_url, status, status_description
```

a. The first lookup should use **knownusers.csv** to retrieve **user** values for all matching **username** values in the events. (Hint: You will need to use **user as username** for your lookup. This tells Splunk to match the values of **user** from the lookup against the values of **username** from the event data.)

b. The second lookup should use **status_definitions.csv** to retrieve **status_description** values for all matching **status** values in the events.

c. Run the search over the **Last 24 hours**.

```
index=network sourcetype=cisco_wsa_squid x_webcat_code_full="Uncategorized URLs"
| lookup knownusers.csv user as username OUTPUT user
| lookup status_definitions.csv status OUTPUT status_description
| search user=*
| table user, dept, cs_url, status, status_description
```



user	dept	cs_url	status	status_description
kpeha	Sales	http://www.holoweb.com/	200	OK
kperna	Web Development	http://filmunlock.com/download/666c507271673d3d83b13d19/License.v.3.413.dmg	403	Forbidden
gfacello	Engineering	http://www.homeschoolblogger.com/	302	Found
kpeha	Sales	http://www.holoweb.com/style.css	200	OK
cquinn	Security Operations	http://www.c404.net/	200	OK
apucci	Americas Sales	http://www.reelviews.net/images/xmlbuttonorange.gif	200	OK
apucci	Americas Sales	http://www.reelviews.net/images/icon-mrqe.gif	200	OK
svoronoff	IT Operations	http://www.windowsforumz.com/	200	OK
bsimmel	APAC Sales	http://www.rockyreef.com/images/phot.gif	200	OK
bsimmel	APAC Sales	http://www.rockyreef.com/images/rocksbg.jpg	200	OK

Example of final output.

19. Save your search as a report with the name **L1S3**.

Scenario: Sales would like a map of retail sales in Canada by province over the previous week.

Task 5: Use the geospatial lookup file, canada.kml, to return a choropleth map of Canadian retail sales by province during the previous week.

20. The knowledge manager has uploaded and defined the **canada.kml** geospatial lookup and provided you with the following information. Use this info to create a search that will display the contents of the lookup. You should see the geospatial lookup output displayed as a table with the following fields: **count**, **featureCollection**, **featureId**, and **geom**.

    filename: canada.kml
    definition name: canada_prov
    lookup type: geospatial

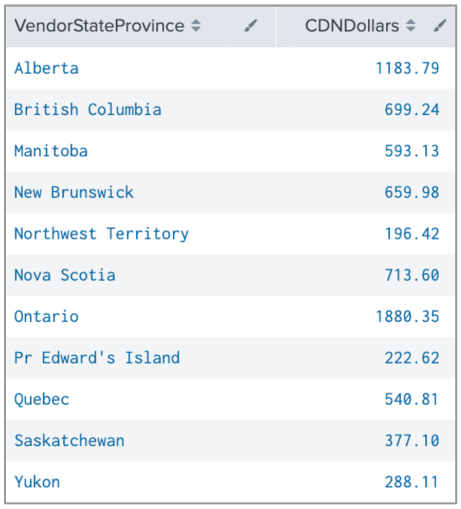
```
| inputlookup canada_prov
```

    Using `inputlookup` with the lookup filename only works for .csv and .csv.gz files. If you tried to search for `| inputlookup canada.kml`, you would have received an error message.

21. Open a second search browser window. The following search calculates total sales from Canada in Canadian dollars. Complete the **<missing>** portions of the **geom** command so that the results of this search are correlated with the **canada.kml** lookup. (Hint: The **geom** command must use the geospatial lookup definition name and the **featureIdField** should be a field with values that are present in the

events and in the lookup.)

```
index=sales sourcetype=vendor_sales VendorCountry=Canada
| stats sum(price) as USDollars by VendorStateProvince
| eval CDNDollars = round(USDollars*1.31,2)
| fields - USDollars
| geom <missing> featureIdField=<missing>
```

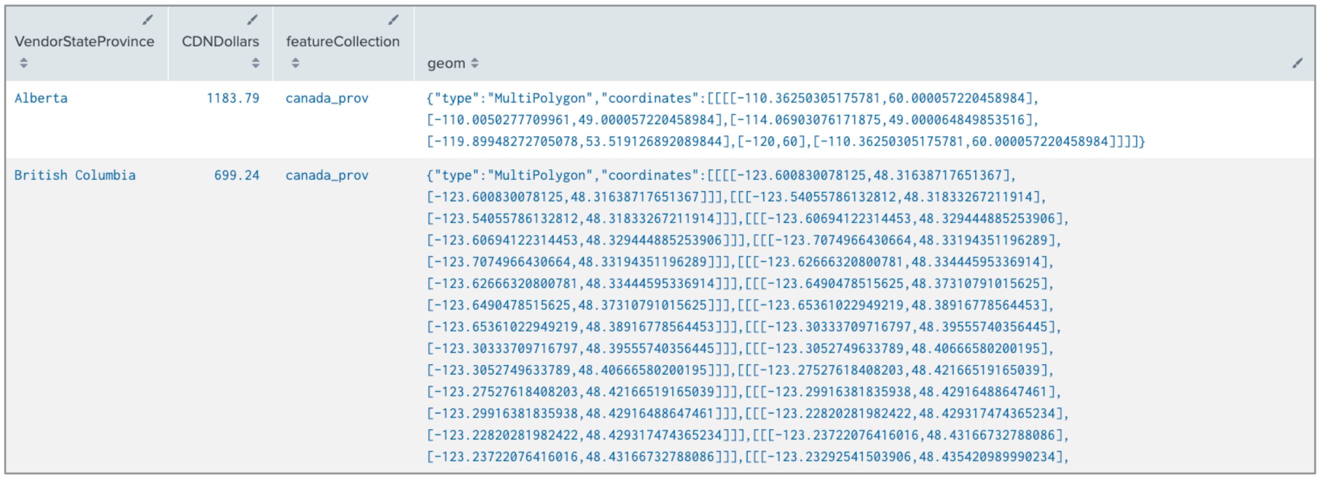


VendorStateProvince	CDNDollars
Alberta	1183.79
British Columbia	699.24
Manitoba	593.13
New Brunswick	659.98
Northwest Territory	196.42
Nova Scotia	713.60
Ontario	1880.35
Pr Edward's Island	222.62
Quebec	540.81
Saskatchewan	377.10
Yukon	288.11

Output of the first 4 lines of this search.

```
index=sales sourcetype=vendor_sales VendorCountry=Canada
| stats sum(price) as USDollars by VendorStateProvince
| eval CDNDollars = round(USDollars*1.31,2)
| fields - USDollars
| geom canada_prov featureIdField=VendorStateProvince
```

22. Run the search over the **Previous week** and confirm that your output looks like the table below.

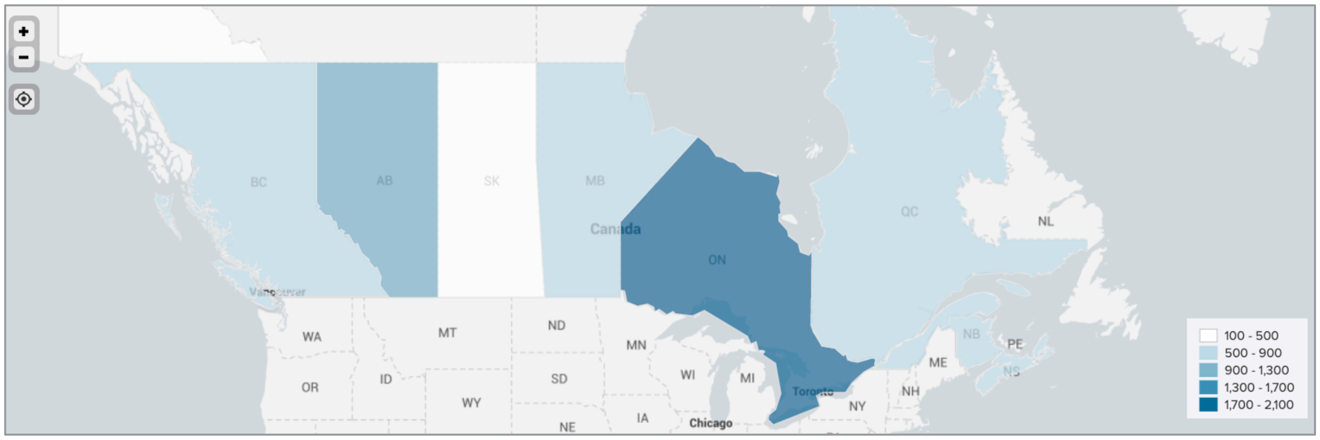


VendorStateProvince	CDNDollars	featureCollection	geom
Alberta	1183.79	canada_prov	{"type":"MultiPolygon","coordinates":[[[[-110.36250305175781,60.000057220458984],[-110.0050277709961,49.000057220458984],[-114.06903076171875,49.000064849853516],[-119.8994827270507 8,53.519126892089844],[-120,60],[-110.36250305175781,60.000057220458984]]]]}
British Columbia	699.24	canada_prov	{"type":"MultiPolygon","coordinates":[[[[-123.600830078125,48.31638717651367],[-123.600830078125,48.31638717651367]]],[[[-123.54055786132812,48.31833267211914],[-123.54055786132812,48.31833267211914]]],[[[-123.60694122314453,48.329444885253906],[-123.60694122314453,48.329444885253906]]],[[[-123.7074966430664,48.33194351196289],[-123.7074966430664,48.33194351196289]]],[[[-123.62666320800781,48.33444595336914],[-123.62666320800781,48.33444595336914]]],[[[-123.6490478515625,48.37310791015625],[-123.6490478515625,48.37310791015625]]],[[[-123.65361022949219,48.38916778564453],[-123.65361022949219,48.38916778564453]]],[[[-123.30333709716797,48.39555740356445],[-123.30333709716797,48.39555740356445]]],[[[-123.3052749633789,48.40666580200195],[-123.3052749633789,48.40666580200195]]],[[[-123.27527618408203,48.42166519165039],[-123.27527618408203,48.42166519165039]]],[[[-123.29916381835938,48.42916488647461],[-123.29916381835938,48.42916488647461]]],[[[-123.22820281982422,48.429317474365234],[-123.22820281982422,48.429317474365234]]],[[[-123.23722076416016,48.43166732788086],[-123.23722076416016,48.43166732788086]]],[[[-123.23292541503906,48.435420989990234],

23. Click the **Visualization** tab and change the visualization to **Choropleth Map**.
24. Under the **Format** tab:
    a. Set **Latitude** to **53**.
    b. Set **Longitude** to **-92**.
    c. Set **Zoom** to **4**.

d. Set **Color Mode** to **Sequential**.

e. Set **Maximum Color** to **006D9C**.



25. Save your search as a report with the name **L1S4**.

Scenario: TechOps wants to be able to search for web server errors coming from www.buttercupgames.com that are associated with unsuccessful purchases.

Task 6: Troubleshoot this search and then output results to a lookup with the `outputlookup` command.

26. This search is not returning the desired results. Troubleshoot the **`lookup`** command expression.

```
index=web sourcetype=access_combined status!=200
  referer_domain=http://www.buttercupgames.com
| lookup status_definitions.csv status_description as Description OUTPUT status
  status_type
| stats count by host, status_description, status_type, clientip
| stats list(status_description) as status_description, list(status_type) as
  status_type, list(host) as host, list(count) as "count" by clientip
```

This search does not work because the chosen lookup field to match against the search results, `Description`, is not a common field between the `access_combined` sourcetype and the `status_definitions.csv` lookup. You will need to find a common field between the `access_combined` sourcetype and the `status_definitions.csv` lookup to use as your lookup field.

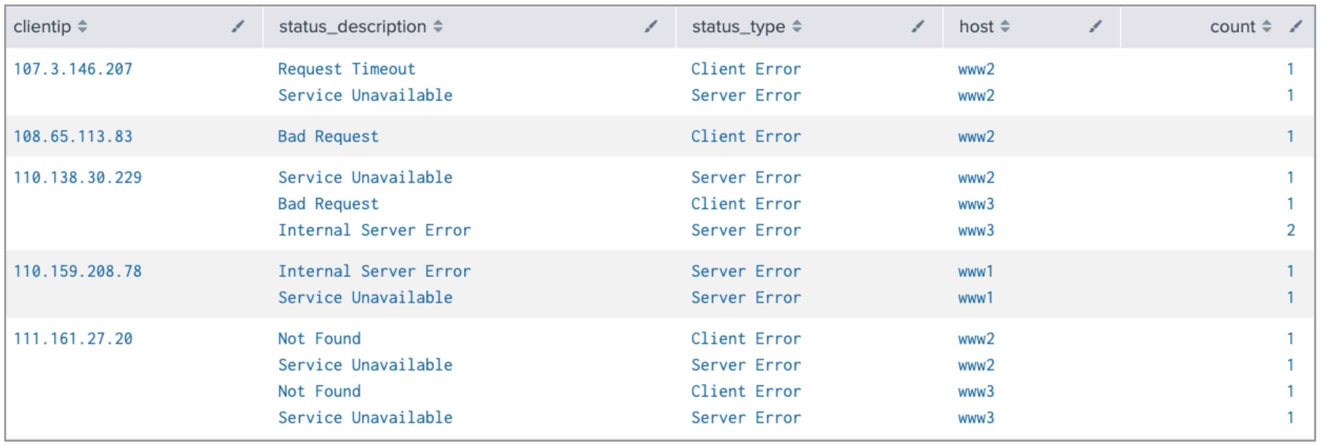
You can find this field by running the basic search in one browser tab and the lookup contents in another browser tab and comparing the fields shown in the Field Sidebar. Alternatively, you can run this search:

```
index=web sourcetype=access_combined
|fieldsummary
|fields field
|append
 [|inputlookup status_definitions.csv
  |fieldsummary
  |fields field]
|stats count by field
|where count>=2
```

This search puts the fields from the `access_combined` sourcetype and the fields from the `status_definitions.csv` lookup into one column. (The fields from the `status_definitions.csv` lookup will be appended, i.e. added, to the bottom.) Then, the `stats` command will count how many times each field appears and the `where` command will find the fields that appeared twice. When you run this search, you will get only one result, the `status` field. This is the only field that appears twice because it exists in the `access_combined` sourcetype and the `status_definitions.csv` lookup.

Now that you have your common field, you can edit the search:

```
index=web sourcetype=access_combined status!=200
  referer_domain=http://www.buttercupgames.com
| lookup status_definitions.csv status OUTPUT status_description
  status_type
| stats count by host, status_description, status_type, clientip
| stats list(status_description) as status_description, list(status_type) as
  status_type, list(host) as host, list(count) as "count" by clientip
```

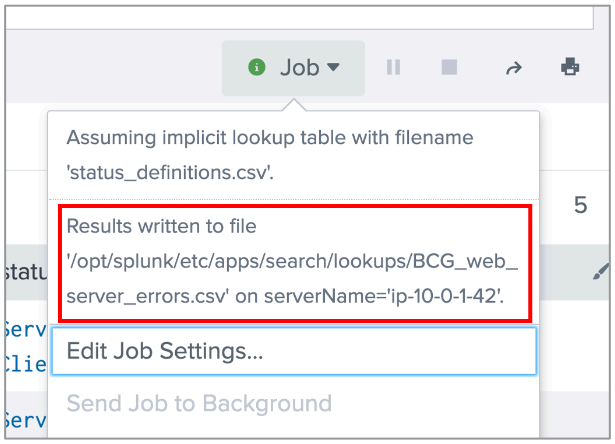


clientip	status_description	status_type	host	count
107.3.146.207	Request Timeout Service Unavailable	Client Error Server Error	www2 www2	1 1
108.65.113.83	Bad Request	Client Error	www2	1
110.138.30.229	Service Unavailable Bad Request Internal Server Error	Server Error Client Error Server Error	www2 www3 www3	1 1 2
110.159.208.78	Internal Server Error Service Unavailable	Server Error Server Error	www1 www1	1 1
111.161.27.20	Not Found Service Unavailable Not Found Service Unavailable	Client Error Server Error Client Error Server Error	www2 www2 www3 www3	1 1 1 1

27. Output the results of this search to a lookup called **`BCG_web_server_errors.csv`**. Make sure the lookup is created in the same app the search is being run. Run the search over the **Last 24 hours**.

```
index=web sourcetype=access_combined status!=200
  referer_domain=http://www.buttercupgames.com
| lookup status_definitions.csv status OUTPUT status_description
  status_type
| stats count by host, status_description, status_type, clientip
| stats list(status_description) as status_description, list(status_type) as
  status_type, list(host) as host, list(count) as "count" by clientip
| outputlookup BCG_web_server_errors.csv createinapp=true
```

28. Confirm that the **`BCG_web_server_errors.csv lookup`** has been created by clicking on **Job**.



Assuming implicit lookup table with filename 'status_definitions.csv'.

Results written to file '/opt/splunk/etc/apps/search/lookups/BCG_web_server_errors.csv' on serverName='ip-10-0-1-42'.

29. Save your search as a report with the name **L1S5**.

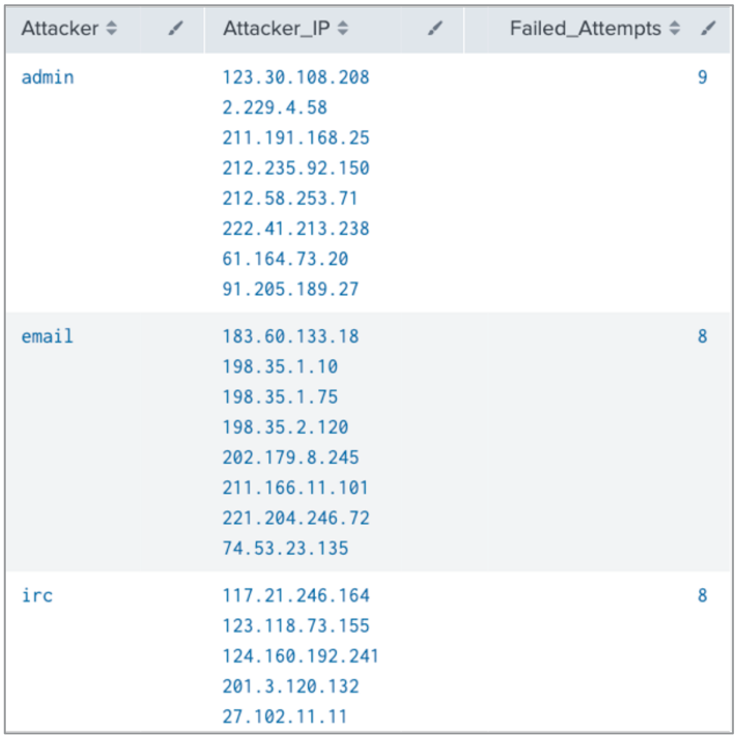
Challenge: Filter a search by excluding values from a lookup.

Scenario: SecOps is finding an increase in penetration attempts. Find *unknown* users with more than 3 failed logins within the last 24 hours.

30. Complete the **`<missing>`** portion of the **`lookup`** expression in this search. The search should exclude known users from the final results. Keep a few things in mind:
    a. Both the **`linux_secure`** data and the **`knownusers.csv`** lookup file use the same field name for **`user`**. Therefore, the **`user`** field from the **`linux_secure`** data has been renamed to **`user_from_events`** before using the **`lookup`** and **`search`** commands.
    b. The **`search`** command filters search results. The **`<missing>`** portion of the **`search`** expression is a field name.
    c. The remainder of the search performs statistical aggregations on the results and further manipulates the data to achieve the scenario goal.
    d. The search should be run over the **Last 24 hours**.

```
index=security sourcetype=linux_secure fail*
| rename user as user_from_events
| lookup <missing>
| search NOT <missing>=*
| stats count by user_from_events, src_ip
| stats values(src_ip) as Attacker_IP, sum(count) as Failed_Attempts by
  user_from_events
| rename user_from_events as Attacker
| search Failed_Attempts > 3
| sort -Failed_Attempts
```

```
index=security sourcetype=linux_secure fail*
| rename user as user_from_events
| lookup knownusers.csv user as user_from_events OUTPUT user
| search NOT user=*
| stats count by user_from_events, src_ip
| stats values(src_ip) as Attacker_IP, sum(count) as Failed_Attempts by
  user_from_events
| rename user_from_events as Attacker
| search Failed_Attempts > 3
| sort -Failed_Attempts
```



Attacker	Attacker_IP	Failed_Attempts
admin	123.30.108.208 2.229.4.58 211.191.168.25 212.235.92.150 212.58.253.71 222.41.213.238 61.164.73.20 91.205.189.27	9
email	183.60.133.18 198.35.1.10 198.35.1.75 198.35.2.120 202.179.8.245 211.166.11.101 221.204.246.72 74.53.23.135	8
irc	117.21.246.164 123.118.73.155 124.160.192.241 201.3.120.132 27.102.11.11	8

Example of final output.

31. Save your search as a report with the name **L1X.**

splunk>

Lab Exercise 2 – Adding a Subsearch

Description

Create subsearches to manipulate search input.

Steps

Scenario: Marketing and Sales would like to know how many times multiplayer games were "viewed" on the website during the "Multiplayer Madness" event this past Saturday.

Task 1: Use a subsearch and a lookup to filter search results.

1. Your knowledge manager provided you with the following information. These lookups contain information about the products sold by Buttercup Games. This task requires events from the following games: SIM Cubicle, Dream Crusher, Mediocre Kingdoms, Puppies vs. Zombies, Manganiello Bros., Final Sequel, Benign Space Debris, and Curling 2014. Use the **`inputlookup`** command to find the correct lookup and verify its contents.

   filename: products.csv
   definition name: product_lookup
   description: code, category ID, price, product ID, and sale price of all products
   lookup type: file-based

   filename: sp_products.csv
   definition name: none
   description: list of single player games
   lookup type: file-based

   filename: mp_products.csv
   definition name: none
   description: list of multiplayer games
   lookup type: file-based

   ```
   | inputlookup mp_products.csv
   ```

2. This search is looking back to Saturday (**`earliest=@w6 latest=@w7`**) for all events involving a "view" action in the web sales index. Then, the search transforms and sorts the data to show which game was viewed the most. Replace the **`<missing>`** portion of the basic search with a subsearch so that only events involving multiplayer games are returned.

   ```
   index=web sourcetype=access_combined action="view" earliest=@w6 latest=@w7 <missing>
   | stats count(action) as "viewed" by product_name
   | sort -viewed
   ```

   ```
   index=web sourcetype=access_combined action="view" earliest=@w6 latest=@w7
     [inputlookup mp_products.csv]
   | stats count(action) as "viewed" by product_name
   | sort -viewed
   ```

   This subsearch will also work if you don't have a lookup. However, it runs significantly slower.

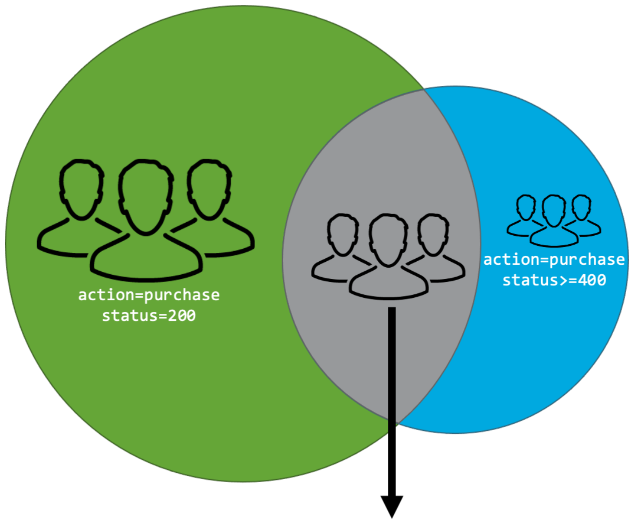
   ```
   index=web sourcetype=access_combined action="view" earliest=@w6 latest=@w7
      [search index=web sourcetype=access_combined product_name="Dream Crusher" OR
       product_name="SIM Cubicle" OR product_name="Mediocre Kingdoms" OR
       product_name="Final Sequel" OR product_name="Manganiello Bros." OR
   ```

```
        product_name="Puppies vs. Zombies" OR product_name="Curling 2014" OR
        product_name="Benign Space Debris"
       | fields product_name]
    | stats count(action) as "viewed" by product_name
    | sort -viewed
```

3. Save your search as a report with the name **L2S1**.

Task 2: Combine two searches into a single search. The resulting search should find the average and median sales totals for `clientips` who have experienced problems making a purchase (`action=purchase status>=400`) but still managed to complete a successful web order during the previous week.

4. This Venn diagram represents the components of this search: the results of the outer search (green), the results of the inner search (blue), and the results of the outer search filtered by the results of the inner search (grey).



The data from these customers (`clientips`) will undergo statistical transformations to get average and median sales totals

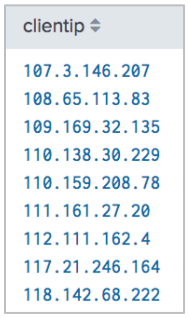
Based on the task description, we want to perform statistical transformations on the data represented by the grey inner section—the customers that experienced problems with a purchase (**`action=purchase status>=400`**) yet still completed a successful online sales order (**`action=purchase status=200`**) over the previous week.

Answer these questions about the inner and outer searches:

a. TRUE or FALSE: The inner search (blue) will look for customers who did not experience issues with their online purchase.
FALSE; The inner search will be filtering based on `status>=400` and therefore, will retrieve events where customers experienced purchase errors.

b. TRUE or FALSE: The outer search (green) will look for successful purchase events but only return events from customers that appeared in the results of the inner search.
FALSE; The outer search (green) will only return events from customers who experienced a successful purchase. The grey section contains the customers who experienced successful purchases and appeared in the results of the inner search. This only happens when the outer and inner searches are combined. This is what you will do in this task.

5. Which of these searches provides the desired results of the inner search?



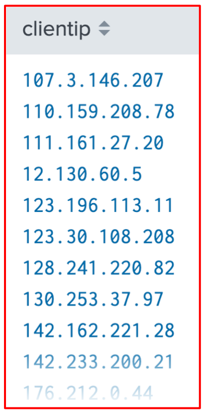
clientip
107.3.146.207
108.65.113.83
109.169.32.135
110.138.30.229
110.159.208.78
111.161.27.20
112.111.162.4
117.21.246.164
118.142.68.222

Desired results of inner search.

Search 1
```
index=web sourcetype=access_combined status=200 action=purchase
| stats sum(sale_price) as sales_sum by clientip
| stats avg(sales_sum) as avg_sales, median(sales_sum) as median_sales
```

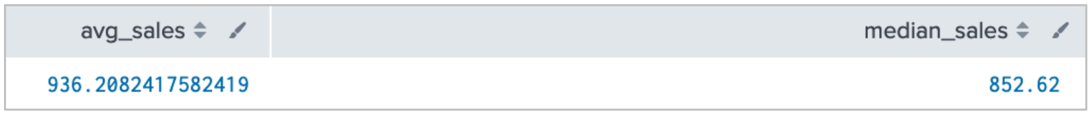
Search 2
```
index=web sourcetype=access_combined status=200 OR status=400 action=purchase
| stats sum(sale_price) as sales_sum by clientip
| stats avg(sales_sum) as avg_sales, median(sales_sum) as median_sales
```

Search 3
```
search index=web sourcetype=access_combined status>=400 action=purchase
| stats values(clientip) as clientip
```
This is the correct inner search. This search finds all unsuccessful web sales purchase events and outputs a list of unique **clientips**. This search, when used as a subsearch, will act as a filter on the outer search.



clientip
107.3.146.207
110.159.208.78
111.161.27.20
12.130.60.5
123.196.113.11
123.30.108.208
128.241.220.82
130.253.37.97
142.162.221.28
142.233.200.21
176.212.0.44

6. Which of these searches provides the desired results of the outer search? (Note: If you run these searches, remove the **[<subsearch>]** placeholder, otherwise you will receive an error.)

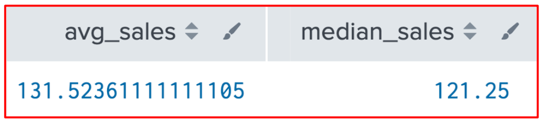


avg_sales	median_sales
936.2082417582419	852.62

Desired results of outer search.

Search 1
```
index=web sourcetype=access_combined status=200 action=purchase
  [<subsearch>]
| stats sum(sale_price) as sales_sum by clientip
| stats avg(sales_sum) as avg_sales, median(sales_sum) as median_sales
```

This is the correct outer search. The results of the subsearch will be added to the basic search. Therefore, the basic search will look for all successful purchase events from the web sales server that match the **`clientip`** values returned by the subsearch/inner search. These events are transformed by the **`stats`** command to return average and median sales.



avg_sales	median_sales
131.52361111111105	121.25

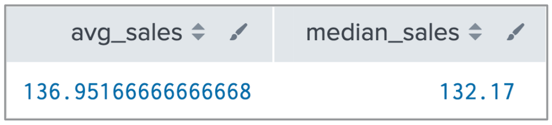
Search 2

```
index=web sourcetype=access_combined status=200 OR status=400 action=purchase
  [<subsearch>]
| stats sum(sale_price) as sales_sum by clientip
| stats avg(sales_sum) as avg_sales, median(sales_sum) as median_sales
```

Search 3

```
search index=web sourcetype=access_combined status>=400 action=purchase
  [<subsearch>]
| stats values(clientip) as clientip
```

7. Combine the inner and outer search to create your final search. Run this search over the **Previous week**.



avg_sales	median_sales
136.95166666666668	132.17

```
index=web sourcetype=access_combined status=200 action=purchase
  [search index=web sourcetype=access_combined status>=400 action=purchase
   | stats values(clientip) as clientip]
| stats sum(sale_price) as sales_sum by clientip
| stats avg(sales_sum) as avg_sales, median(sales_sum) as median_sales
```

8. Save your search as a report with the name **L2S2.**

Lab Exercise 3 – Using the `return` Command

Description

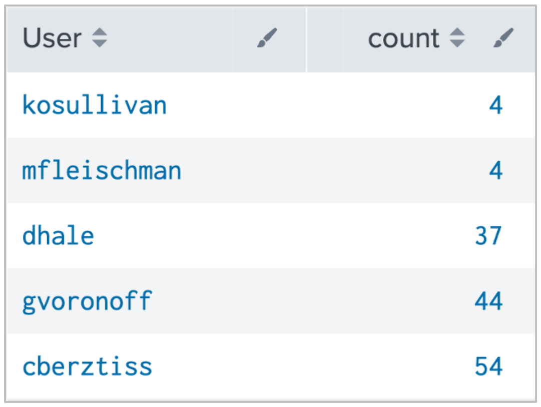
Use the **`return`** command to control output from a search and a subsearch.

Steps

Task 1: Return search results as key value pairs.

1. A coworker has asked you to help create a subsearch for a report. You have created a search that normalizes **`username`** and **`Username`** values in the **`network`** data and finds the top 5 most active users. Complete the **`<missing>`** portion of the search so that **`User`** values are returned as key-value pairs. Run the search over the **Last 24 hours**.

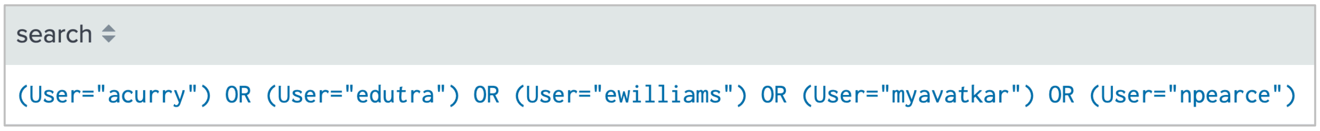
```
index=network
| eval User=coalesce(username,Username)
| stats count by User
| sort 5 -count
| <missing>
```



User	count
kosullivan	4
mfleischman	4
dhale	37
gvoronoff	44
cberztiss	54

Before the ***`return`*** *command.*

```
index=network
| eval User=coalesce(username,Username)
| stats count by User
| sort 5 -count
| return 5 User
```



```
search
(User="acurry") OR (User="edutra") OR (User="ewilliams") OR (User="myavatkar") OR (User="npearce")
```

After the ***`return`*** *command.*

2. Save your search as a report with the name **L3S1**.

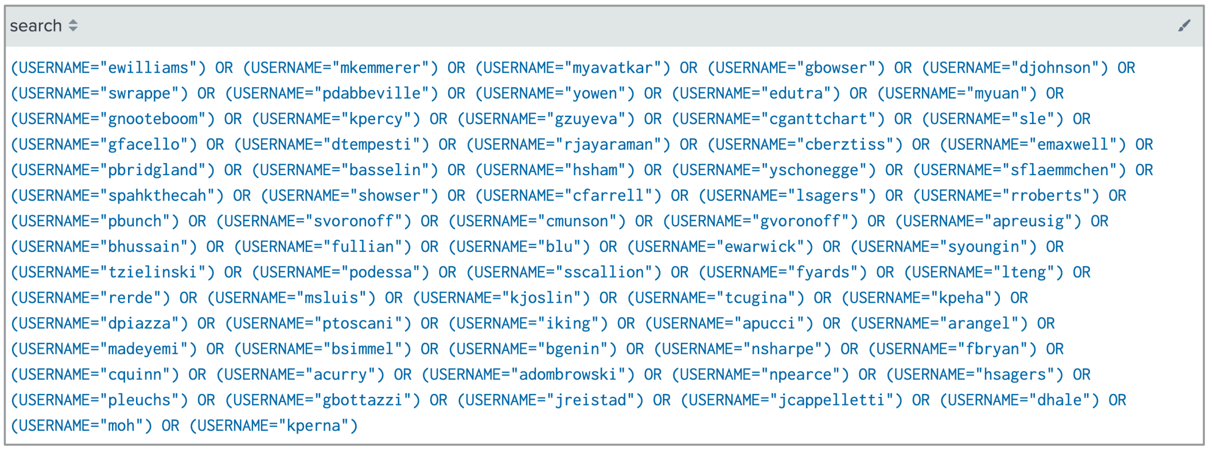
splunk>

Scenario: SecOps wants to know which employees have entered invalid passwords over the last 7 days.

Task 2: Filter search input by returning key-value pairs from the `employees.csv` lookup. Count instances of "failed password" by employee usernames.

3. Your knowledge manager has provided you with the following information about the **`employees.csv`** lookup. Create a search that will open **`employees.csv`** and return all **`USERNAME`** values as key-value pairs. (Hint: Use the **`inputlookup`** command with the **`employees.csv`** lookup to find out how many rows of data exist in the lookup file. The number of rows will match the number of results returned. Then, use this number with the **`return`** command.)

   filename: employees.csv
   definition name: employee_lookup
   lookup type: file-based



```
(USERNAME="ewilliams") OR (USERNAME="mkemmerer") OR (USERNAME="myavatkar") OR (USERNAME="gbowser") OR (USERNAME="djohnson") OR
(USERNAME="swrappe") OR (USERNAME="pdabbeville") OR (USERNAME="yowen") OR (USERNAME="edutra") OR (USERNAME="myuan") OR
(USERNAME="gnooteboom") OR (USERNAME="kpercy") OR (USERNAME="gzuyeva") OR (USERNAME="cganttchart") OR (USERNAME="sle") OR
(USERNAME="gfacello") OR (USERNAME="dtempesti") OR (USERNAME="rjayaraman") OR (USERNAME="cberztiss") OR (USERNAME="emaxwell") OR
(USERNAME="pbridgland") OR (USERNAME="basselin") OR (USERNAME="hsham") OR (USERNAME="yschonegge") OR (USERNAME="sflaemmchen") OR
(USERNAME="spahkthecah") OR (USERNAME="showser") OR (USERNAME="cfarrell") OR (USERNAME="lsagers") OR (USERNAME="rroberts") OR
(USERNAME="pbunch") OR (USERNAME="svoronoff") OR (USERNAME="cmunson") OR (USERNAME="gvoronoff") OR (USERNAME="apreusig") OR
(USERNAME="bhussain") OR (USERNAME="fullian") OR (USERNAME="blu") OR (USERNAME="ewarwick") OR (USERNAME="syoungin") OR
(USERNAME="tzielinski") OR (USERNAME="podessa") OR (USERNAME="sscallion") OR (USERNAME="fyards") OR (USERNAME="lteng") OR
(USERNAME="rerde") OR (USERNAME="msluis") OR (USERNAME="kjoslin") OR (USERNAME="tcugina") OR (USERNAME="kpeha") OR
(USERNAME="dpiazza") OR (USERNAME="ptoscani") OR (USERNAME="iking") OR (USERNAME="apucci") OR (USERNAME="arangel") OR
(USERNAME="madeyemi") OR (USERNAME="bsimmel") OR (USERNAME="bgenin") OR (USERNAME="nsharpe") OR (USERNAME="fbryan") OR
(USERNAME="cquinn") OR (USERNAME="acurry") OR (USERNAME="adombrowski") OR (USERNAME="npearce") OR (USERNAME="hsagers") OR
(USERNAME="pleuchs") OR (USERNAME="gbottazzi") OR (USERNAME="jreistad") OR (USERNAME="jcappelletti") OR (USERNAME="dhale") OR
(USERNAME="moh") OR (USERNAME="kperna")
```

```
| inputlookup employees.csv
| return 72 USERNAME
```

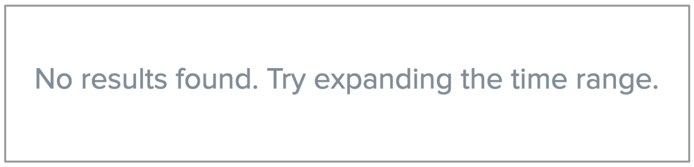
4. This search looks for "failed password" events in the **`security`** index. Filter the search input by adding the subsearch you created in the previous step. Then, run the search over the **Last 7 days**. What results were returned?

```
index=security "failed password"
| stats count by user
```

```
index=security "failed password"
  [| inputlookup employees.csv
   | return 72 USERNAME]
| stats count by user
```

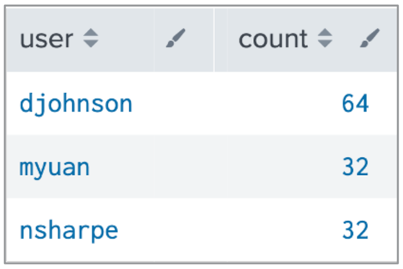
   No results were returned. See step 5 for an explanation.

5. The search is not working because the subsearch is returning **`USERNAME`** values while the outer search is aggregating on **`user`** values. Fix the search and run over the **Last 7 days**. (Hint: No additional pipes need to be added to the search.)


No results found. Try expanding the time range.

Before editing the search.

```
index=security "failed password"
    [| inputlookup employees.csv
    | return 72 user=USERNAME]
| stats count by user
```



user	count
djohnson	64
myuan	32
nsharpe	32

After editing the search.

6. Save your search as a report with the name **L3S2**.