Statistical Processing Lab Solutions Exercises

Overview

Welcome to the Splunk Education lab environment. These lab exercises will test your knowledge of common transforming commands, modifying results with the **`eval`** command and formatting data.

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`	Badge reader	**`history_access`**	Address_Description, Department, Device, Email, Event_Description, First_Name, last_Name, Rfid, Username
	Active Directory	**`winauthentication_security`**	LogName, SourceName, EventCode, EventType, User
	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`	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
`games`	Game logs	**`SimCubeBeta`**	date_hour, date_mday, date_minute, date_month, date_second, data_wday, data_year, date_zone, eventtype, index, linecount, punct, splunk_server, timeendpos, timestartpos

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 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 – Transforming Data

Description

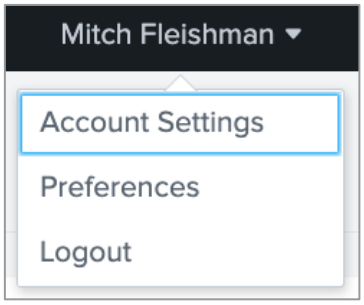
Configure the lab environment user account. Then, transform data using the **`chart`**, **`timechart`**, **`top`**, and **`stats`** commands.

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 also 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 can be delays in executing an action like saving in the UI or returning results of a search. If you are experiencing a delay, 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 begin the pipe on a new line.

Search auto-format disabled (default)

Search auto-format enabled

Scenario: The Network team wants to add a dashboard panel that displays internet usage over the last 24 hours.

Task 2: Complete a search with the `timechart` command to create a multi-series visualization.

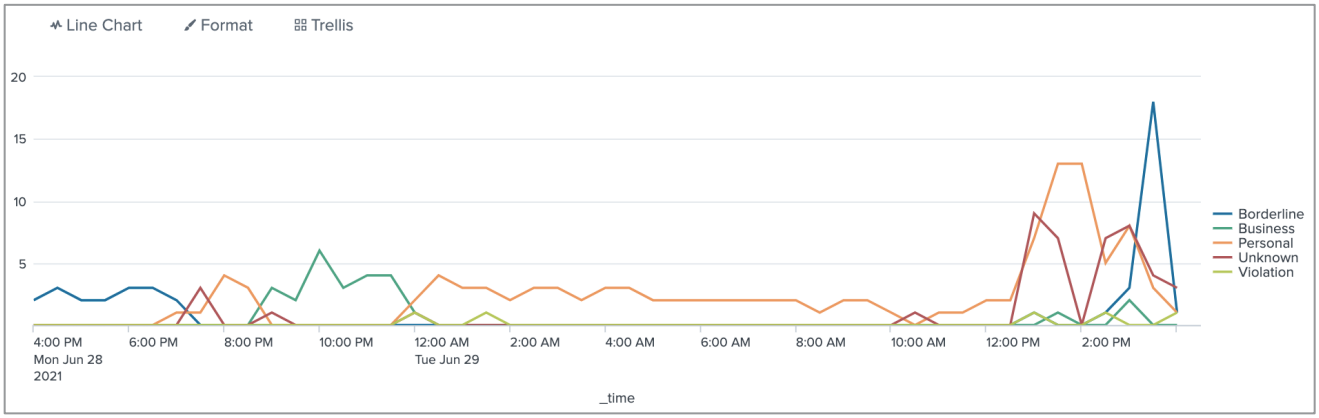
11. In the top left corner of Splunk Web, select **Apps** > **Search & Reporting**. This sets our app context to the search app.
12. Count **`usage`** events from the web security appliance data by completing the **`<missing>`** portion of the search with the **`timechart`** command. Run the search over the **Last 24 hours**.

```
index=network sourcetype=cisco_wsa_squid
| <missing>
```

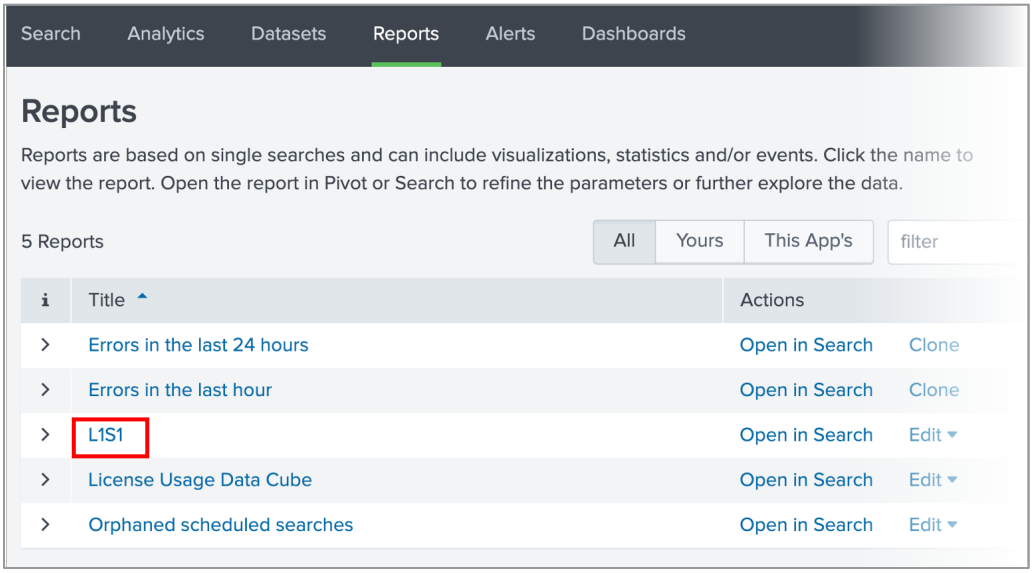
_time	Borderline	Business	Personal	Unknown	Violation
2021-06-28 16:00:00	2	0	0	0	0
2021-06-28 16:30:00	3	0	0	0	0
2021-06-28 17:00:00	2	0	0	0	0
2021-06-28 17:30:00	2	0	0	0	0
2021-06-28 18:00:00	3	0	0	0	0
2021-06-28 18:30:00	3	0	0	0	0
2021-06-28 19:00:00	2	0	1	0	0
2021-06-28 19:30:00	0	0	1	3	0

```
index=network sourcetype=cisco_wsa_squid
| timechart count by usage
```

13. Visualize results as a **Line Chart**.



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.



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

Scenario: Security wants to add a dashboard panel that displays the top 10 IPs associated with "Accepted" and "Failed" events on the web server.

Task 3: Complete a search with the `chart` command to create a multi-series visualization.

15. Re-initialize the search window by clicking **Search** in the application bar. This step should be done every time you save a report so that you do not accidentally overwrite a previous report.
16. Complete the **`<missing>`** portion of the search with the **`chart`** command so that the output displays a count of events for each **`vendor_action`** value by **`src_ip`**. Run the search over the **Last 24 hours**.

```
index=security sourcetype=linux_secure vendor_action!="session opened"
| <missing>
```

vendor_action	10.1.10.172	10.2.10.163	10.3.10.46	124.160.192.241	175.45.177.42	204.107.141.240	212.114.31.255	212.67.31.255	217.132.169.69	223.213.255.255	OTHER
Accepted	0	6	11	0	3	2	1	0	0	1	2
Failed	20	47	18	19	17	26	38	20	19	25	212

Both of these solutions are valid:

```
index=security sourcetype=linux_secure vendor_action!="session opened"
| chart  count over vendor_action by src_ip
```

```
index=security sourcetype=linux_secure vendor_action!="session opened"
| chart  count by vendor_action src_ip
```

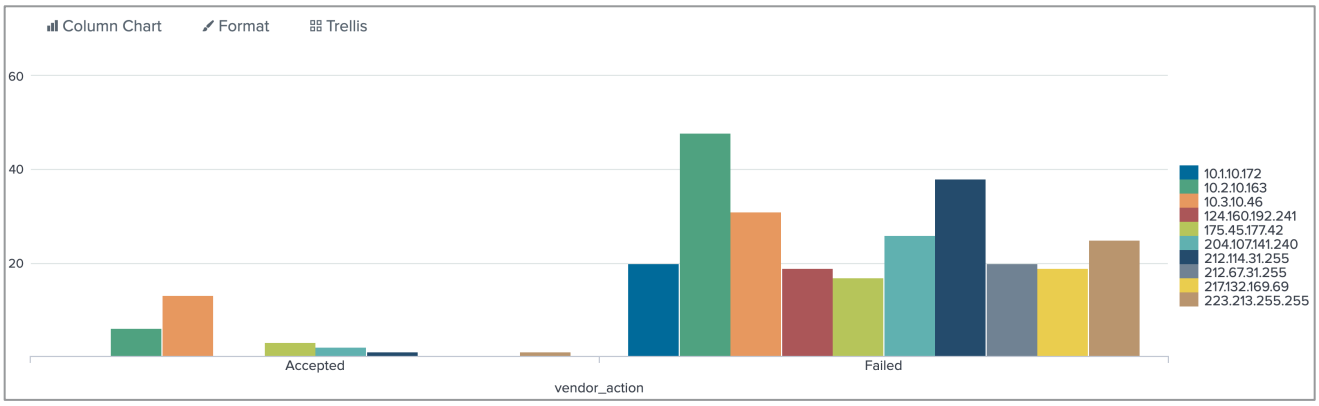
17. Revise the **`chart`** command so that only the top 10 **`src_ip`** values are shown and there is no **`OTHER`** column.

vendor_action	10.1.10.172	10.2.10.163	10.3.10.46	124.160.192.241	175.45.177.42	204.107.141.240	212.114.31.255	212.67.31.255	217.132.169.69	223.213.255.255
Accepted	0	6	11	0	3	2	1	0	0	1
Failed	20	48	26	19	17	26	38	20	19	25

```
index=security sourcetype=linux_secure vendor_action!="session opened"
| chart  count over vendor_action by src_ip useother=f
```

There is no need to use `limit=10` because by default, the `chart` command displays the top 10 values and then groups the remaining values in an `OTHER` category. If the scenario had requested any other number of `src_ip` values, then you would have used the `limit` and `useother` options together.

18. Navigate to the **Visualization** tab and view your results as a **Column Chart**.



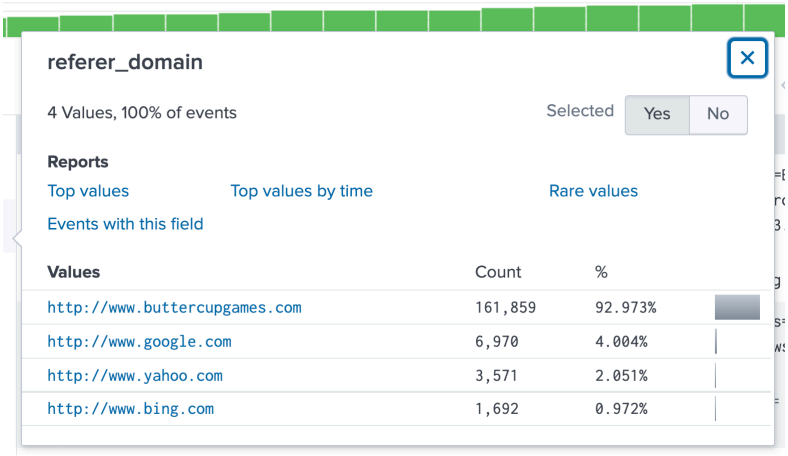
19. Save your search as a report with the name **L1S2**.

Scenario: Sales and Marketing want to know the two most popular referrer domains our website users are coming from.

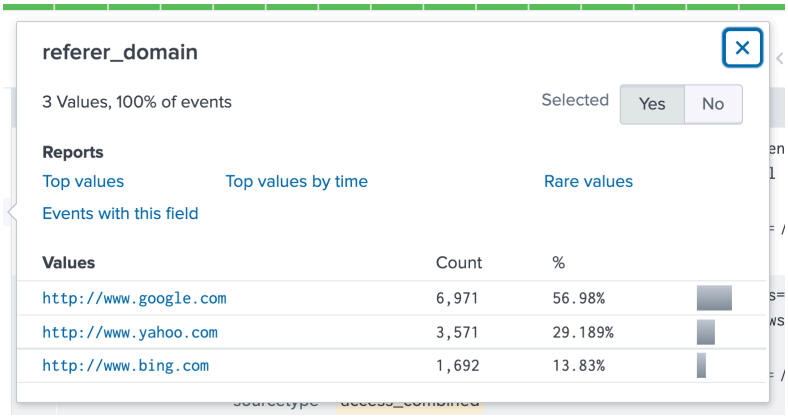
Task 4: Use the `top` command to identify which domains website visitors are using.

20. This search finds all events from online sales data. However, Sales is only interested in external domains. Edit this search so that only events where the **`referer_domain`**, i.e. the domain of the website that a visitor clicked on that led to the http request for a specific product, is not **`http://www.buttercupgames.com`**. Run the search over the **Last 30 days**.

```
index=web sourcetype=access_combined
```



The values of `referer_domain` before removing `http://www.buttercupgames.com`



The values of `referer_domain` after removing `http://www.buttercupgames.com`

```
index=web sourcetype=access_combined referer_domain!=http://www.buttercupgames.com
```

21. Use the **`top`** command to generate a table that shows the top 2 **`referer_domain`** values, the number of events associated with each of these values, and a percentage of events where these values occur.

referer_domain	count	percent
http://www.google.com	6971	56.980546
http://www.yahoo.com	3571	29.189145

```
index=web sourcetype=access_combined referer_domain!=http://www.buttercupgames.com
| top limit=2 referer_domain
```

The `top` command, by default, will generate a table containing a count and a percentage of frequency for the top 10 values of a specified field. The `limit` option is used in this solution to only show the top 2 values of `referer_domain`.

22. Edit your search to remove the **`percent`** column.

referer_domain	count
http://www.google.com	6972
http://www.yahoo.com	3571

Both of these solutions are valid:

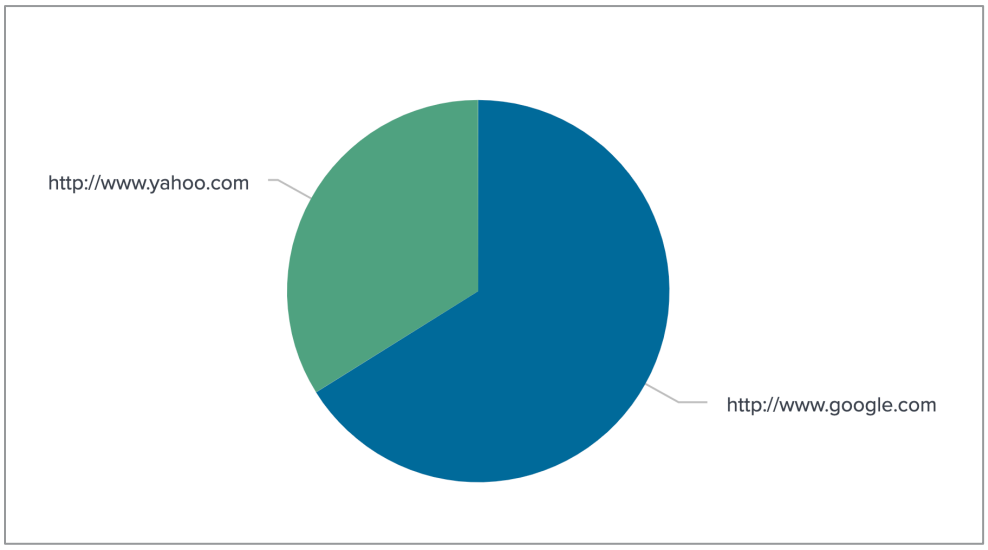
```
index=web sourcetype=access_combined referer_domain!=http://www.buttercupgames.com
| top limit=2 showperc=f referer_domain
```

```
index=web sourcetype=access_combined referer_domain!=http://www.buttercupgames.com
| top limit=2 referer_domain
| fields - percent
```

The `showperc` option can be included before or after `referer_domain`. However, writing your SPL so the options for a command are grouped together makes your search easier to read.

NOTE: Step 23 is optional and can be skipped. Continue to step 24 to save your search as a report.

23. Visualize your results as a **Pie Chart**.



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

Scenario: Facilities needs to know how many people are accessing the Buttercup Games offices daily.

Task 5: Use the `stats` command to count badge swipes at the Buttercup Games offices in San Francisco, Boston, and London.

25. Search the badge reader data (**`index=security sourcetype=history_access`**) over the **Last 24 hours**.

```
index=security sourcetype=history_access
```

26. Investigate the data and the fields in the **Interesting Fields** list. Find the field that contains the office location values, e.g. "San Francisco", "Boston", and "London." Use the **`stats`** command to count events by this field.

Address_Description	count
Boston	70
London	98
San Francisco	203

```
index=security sourcetype=history_access
| stats count by Address_Description
```

27. Revise your **`stats`** command to display a distinct count of **`Username`** values by office location.

Address_Description	dc(Username)
Boston	14
London	16
San Francisco	39

```
index=security sourcetype=history_access
| stats dc(Username) by Address_Description
```

28. Rename the count field to "Badged-in Employees."

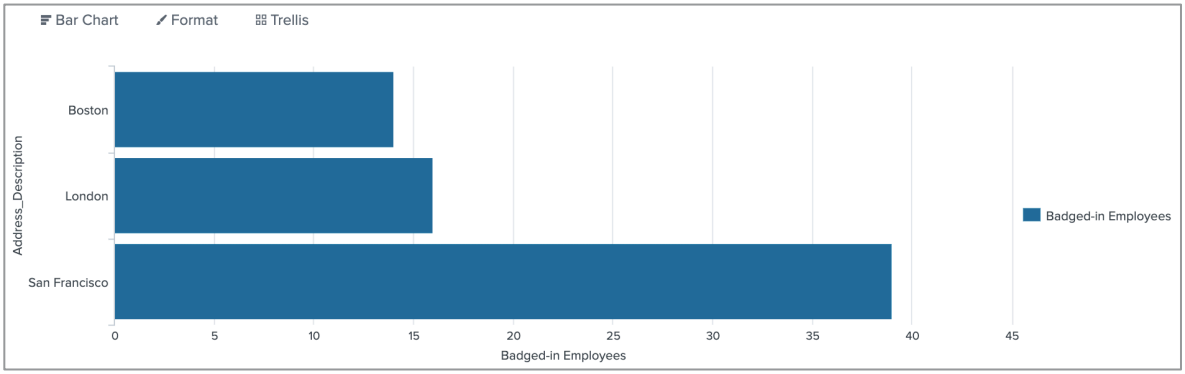
Address_Description	Badged-in Employees
Boston	14
London	16
San Francisco	39

```
index=security sourcetype=history_access
| stats dc(Username) as "Badged-in Employees" by Address_Description
```

You could also use the `rename` command:

```
index=security sourcetype=history_access
| stats dc(Username) by Address_Description
| rename dc(Username) as "Badged-in Employees"
```

29. Visualize your results as a **Bar Chart**.



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

OPTIONAL TASK 1: Security wants to identify the types of content employees are viewing while on the network. Specifically, they want to know the rare content types as these can potentially be malicious. Use the `rare` command to identify uncommon content types employees are accessing while on the internal network.

31. Search security web appliance events (**`index=network sourcetype=cisco_wsa_squid`**) and find the 3 most uncommon **`cs_mime_type`**, i.e. media type, values. Run your search over the **Last 24 hours**.

cs_mime_type	count	percent
application/x-shockwave-flash	1	0.666667
application/octet-stream	3	2.000000
text/css	4	2.666667

```
index=network sourcetype=cisco_wsa_squid
| rare limit=3 cs_mime_type
```

32. Save your search as a report with the name **L1X1**.

OPTIONAL TASK 2: Sales wants to know the 5 best-selling products for North American vendors over the previous week. Complete a search with the `chart` command to create a multi-series visualization.

33. Complete the **`<missing>`** portion of this search with the **`chart`** command so that the output displays a count of events for each **`VendorCountry`**. Run the search over the **Previous week**. (Note: The basic search contains **`VendorID<4000`** because in our environment the **`VendorID`**s for North American countries are 1000 – 2999 for USA and 3000 – 3999 for Canada.)

```
index=sales sourcetype=vendor_sales VendorID<4000
| <missing>
```

VendorCountry	count
Canada	205
United States	3569

```
index=sales sourcetype=vendor_sales VendorID<4000
| chart count by VendorCountry
```

34. Split your data by **`product_name`** to see a count of each product sold in USA and Canada.

VendorCountry	Dream Crusher	Final Sequel	Fire Resistance Suit of Provolone	Holy Blade of Gouda	Manganiello Bros.	Manganiello Bros. Tee	Puppies vs. Zombies	SIM Cubicle	World of Cheese	World of Cheese Tee	OTHER
Canada	16	19	8	20	17	6	4	20	26	5	64
United States	364	257	324	221	263	225	400	334	390	248	543

Both solutions are valid:

```
index=sales sourcetype=vendor_sales VendorID<4000
| chart count by VendorCountry product_name
```

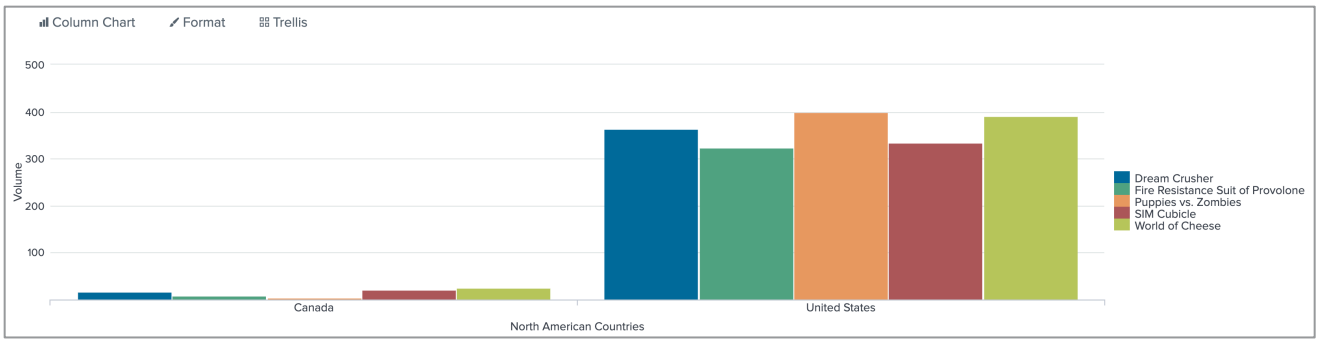
```
index=sales sourcetype=vendor_sales VendorID<4000
| chart count over VendorCountry by product_name
```

35. Finally, edit the **`chart`** command so that only the top 5 best-selling products are displayed without an **`OTHER`** category.

VendorCountry	Dream Crusher	Fire Resistance Suit of Provolone	Puppies vs. Zombies	SIM Cubicle	World of Cheese
Canada	16	8	4	20	26
United States	364	324	400	334	390

```
index=sales sourcetype=vendor_sales VendorID<4000
| chart count over VendorCountry by product_name limit=5 useother=f
```

36. Visualize your results as a **Column Chart**. Use the **Format** tab to add custom X and Y-axis labels:
    a. **X-Axis** > **Title**: Choose **Custom** and then enter: North American Countries
    b. **Y-Axis** > **Title**: Choose **Custom** and then enter: Volume



37. Save your search as a report with the name **L1X2**.

Lab Exercise 2 – Manipulating Data with `eval` Command

Description

Use **`eval`** functions to manipulate search results.

Steps

Scenario: Sales wants to know the total events, average price, and total price for each action performed by visitors to the online store during the previous week.

Task 1: Use the `stats` command and the `eval` command to transform and manipulate event data.

1. Create a search that performs the following calculations and modifications on online sales data (**`index=web sourcetype=access_combined`**) over the **Previous week**. Your search, including the basic search, should be between 4 and 9 lines long.
    a. Calculate the total events by **`action`**.
    b. Calculate the average **`price`** and sum of **`price`** by each **`action`**.
    c. Rename the count, average, and sum fields as "Total Events", "Average Price", and "Total Amount", respectively.
    d. Round **`Total Amount`** and **`Average Price`** values to two decimal places.
    e. Sort **`Total Amount`** in descending order.

action	Total Events	Average Price	Total Amount
addtocart	777	21.57	15445.84
view	704	21.52	12740.08
purchase	795	21.81	8659.03
changequantity	192	21.32	3560.33
remove	192	21.25	3506.35

Different searches can fulfill this task. These three solutions are all valid.

Solution 1: Least amount of lines

```
index=web sourcetype=access_combined
| stats count as "Total Events", avg(price) as "Average Price", sum(price) as "Total
Amount" by action
| eval "Total Amount"=round('Total Amount',2), "Average Price"=round('Average
Price',2)
| sort -"Total Amount"
```

Solution 2: One line for each search requirement presented in step 1, a-e

```
index=web sourcetype=access_combined
| stats count, avg(price), sum(price) by action
| rename count as "Total Events", avg(price) as "Average Price", sum(price) as "Total
Amount"
| eval "Total Amount"=round('Total Amount',2), "Average Price"=round('Average
Price',2)
| sort -"Total Amount"
```

Solution 3: One line per requirement (longest) with exception for the stats command

```
index=web sourcetype=access_combined
| stats count, avg(price), sum(price) by action
| rename count as "Total Events"
| rename avg(price) as "Average Price"
| rename sum(price) as "Total Amount"
| eval "Total Amount"=round('Total Amount',2)
| eval "Average Price"=round('Average Price',2)
| sort -"Total Amount"
```

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

Scenario: Networking wants to know the daily volume (in MB) handled by all Buttercup Games online sales servers over the previous week.

Task 2: Chart daily volume with `timechart` and use `eval` to convert bytes to megabytes.

3. Search online sales data (**`index=web sourcetype=access_combined`**) over the **Previous week**. Find the numeric field that represents how many bytes were transferred during each http request, i.e. each event. This isn't a trick question. This field is simply called **`bytes`**!
4. Use the **`timechart`** command with the **`sum`** function to calculate the total bytes consumed each day. Use the **`as`** clause to name this calculation "bytes."

_time	bytes
2021-06-20	7878087
2021-06-21	7898523
2021-06-22	8472696
2021-06-23	8626130
2021-06-24	9237105
2021-06-25	9729660
2021-06-26	9730813

```
index=web sourcetype=access_combined
| timechart sum(bytes) as bytes
```

The best command for this calculation is **`timechart`**. By default, **`timechart`** will use a span of 1 day when searching over a 5-day to 100-day period. The default time span will change with different time ranges. For more information, see **Default time spans** in the Search Reference manual.

Use **`span`** if you plan on sharing a report with **`timechart`** and want to preserve a specific time span.

5. Use the **`eval`** command to:
    a. Create a new field called "megabytes."
    b. Convert **`bytes`** to megabytes with the calculation: **`bytes/(1024*1024)`**.
    c. Round the result of this calculation to 2 decimal places.

_time	bytes	megabytes
2021-06-20	7878087	7.51
2021-06-21	7898523	7.53
2021-06-22	8472696	8.08
2021-06-23	8626130	8.23
2021-06-24	9237105	8.81
2021-06-25	9729660	9.28
2021-06-26	9730813	9.28

```
index=web sourcetype=access_combined
| timechart sum(bytes) as bytes
| eval megabytes = round((bytes/(1024*1024)),2)
```

6. Rewrite your search so that your **`eval`** command uses the **`round`** and **`pow`** functions to convert **`bytes`** to **`megabytes`**

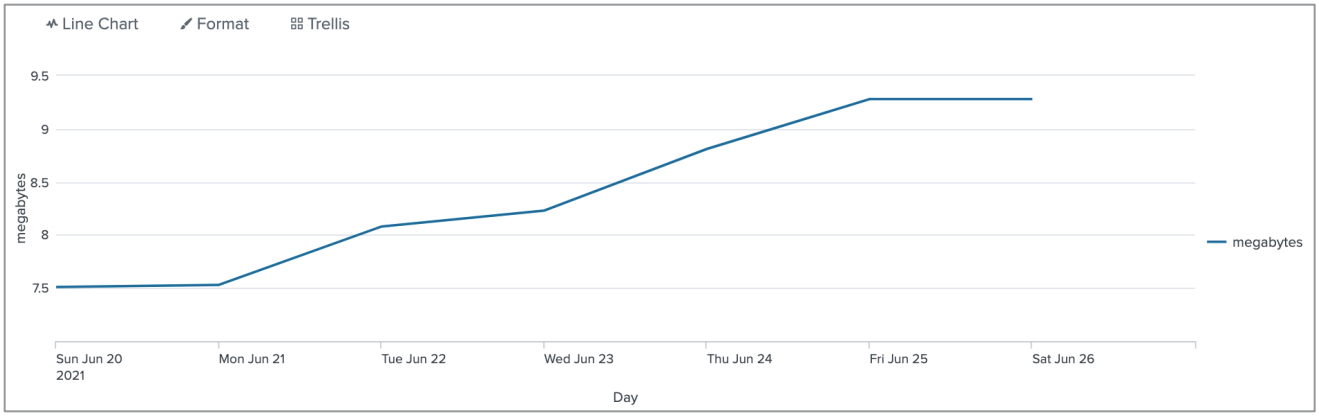
```
index=web sourcetype=access_combined
| timechart sum(bytes) as bytes
| eval megabytes = round(bytes/pow(1024,2),2)
```

7. Remove the **`bytes`** field.

_time	megabytes
2021-06-20	7.51
2021-06-21	7.53
2021-06-22	8.08
2021-06-23	8.23
2021-06-24	8.81
2021-06-25	9.28
2021-06-26	9.28

```
index=web sourcetype=access_combined
| timechart sum(bytes) as bytes
| eval megabytes = round(bytes/pow(1024,2),2)
| fields - bytes
```

8. Visualize your results as a **Line Chart** and rename the **X-axis** to "Day."



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

OPTIONAL TASK: Networking wants to know the total number of GET and POST requests and the ratio of GET to POST requests for each web server over the last 4 hours. Edit the search to round the values of `Ratio`.

10. Edit this search so that the values of **`Ratio`** are rounded to two decimal places. Run the modified search over the **Last 4 hours**.

```
index=web sourcetype=access_combined
| chart count over host by method
| eval Ratio = GET/POST
```

```
index=web sourcetype=access_combined
| chart count over host by method
| eval Ratio = round(GET/POST,2)
```

host	GET	POST	Ratio
www1	961	579	1.6597582037996546
www2	1042	716	1.4553072625698324
www3	1125	672	1.6741071428571428

Before modifying the search.

host	GET	POST	Ratio
www1	961	579	1.66
www2	1042	716	1.46
www3	1125	672	1.67

After modifying the search.

11. Save your search as a report with the name **L2X1**.

CHALLENGE: The Sim Cubicle Beta team needs help randomly generating phone numbers for characters in the game. Use the `random` function to generate fake phone numbers for players.

12. This search looks for all events from the beta phase of the new upcoming game, Sim Cubicle. Then, the **`dedup`** command removes any duplicate values for **`CharacterName`**. Complete the **`<missing>`** portion of this search so that random phone numbers are generated for each **`CharacterName`**. The phone numbers should be in the format **`555-xxxx`** where the last 4 digits contain any number from 0 to 9. Run the search over **All Time**.

```
index=games sourcetype=SimCubeBeta
| dedup CharacterName
| eval phoneNumber = <missing>
| table CharacterName phoneNumber
```

CharacterName	phoneNumber
fenster	555-8667
Dodah	555-5484
gumby	555-8426
juran	555-3241
rexar	555-6428
themancalleddonkey	555-9330

```
index=games sourcetype=SimCubeBeta
| dedup CharacterName
| eval phoneNumber = "555"."-".(random() % 10).(random() % 10).(random() % 10)
  .(random() % 10)
| table CharacterName phoneNumber
```

13. Save your search as a report with the name **L2X2**.

Lab Exercise 3 – Formatting Data

Description

Format search results with **`sort`** and **`rename`** commands and use your knowledge from the previous 2 lab exercises to fulfill scenario requests.

Steps

Scenario: Sales wants to know which one-hour intervals over the last 24 hours have Buttercup Games online sales been twice as profitable as sales in retail stores.

Task 1: Use `timechart` and `sort` commands to create a report that shows the hours where web sales were twice as much as retail sales, sorted in descending order.

1. The provided search pulls successful purchase events from the online sales data (**`index=web sourcetype=access_combined action=purchase status=200`**) and all recorded sales entries from the retail sales data (**`index=sales sourcetype=vendor_sales`**.) Calculate the sum of **`price`** values from these events, grouped into one-hour increments, and split by **`index`**. Run you search over the **Last 24 hours**.

   ```
   (index=web sourcetype=access_combined action=purchase status=200) OR (index=sales sourcetype=vendor_sales)
   ```

_time	sales	web
2021-06-28 13:00	113.91	285.87
2021-06-28 14:00	142.91	360.84
2021-06-28 15:00	185.92	300.87
2021-06-28 16:00	111.92	425.76
2021-06-28 17:00	127.92	391.80
2021-06-28 18:00	210.91	306.83

   ```
   (index=web sourcetype=access_combined action=purchase status=200) OR (index=sales sourcetype=vendor_sales)
   | timechart span=1h sum(price) by index
   ```

   Note: We separate online sales and retail sales data into two different indexes in our environment. If all sales data was kept in the same index, then we would want to split the results of timechart by sourcetype.

2. Use the **where** command to only keep events where the web sales values are more than twice as much as retail sales values:

```
| where web > sales*2
```

_time	sales	web
2021-06-28 13:00	113.91	285.87
2021-06-28 14:00	142.91	360.84
2021-06-28 16:00	111.92	425.76
2021-06-28 17:00	127.92	391.80
2021-06-28 19:00	149.93	548.79
2021-06-28 22:00	187.92	378.80

```
(index=web sourcetype=access_combined action=purchase status=200) OR (index=sales
sourcetype=vendor_sales)
| timechart span=1h sum(price) by index
| where web > sales*2
```

3. Sort results in descending order based on the **web** sales values.

_time	sales	web
2021-06-29 01:00	169.94	602.73
2021-06-28 19:00	149.93	548.79
2021-06-29 00:00	159.93	480.78
2021-06-29 02:00	138.94	434.82
2021-06-28 16:00	111.92	425.76
2021-06-29 03:00	122.94	418.81

```
(index=web sourcetype=access_combined action=purchase status=200) OR (index=sales
sourcetype=vendor_sales)
| timechart span=1h sum(price) by index
| where web > sales*2
| sort - web
```

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

Scenario: Sales wants to know which products had online sales of more than $15,000 during the last 30 days.

Task 2: Fulfill the scenario request using `stats`, `eval`, `sort`, and `rename` commands.

5. Search for all successful purchase events from the online sales data (**`index=web sourcetype=access_combined status=200 action=purchase`**) over the **Last 30 days**.

```
index=web sourcetype=access_combined action=purchase status=200
```

6. Calculate the sum of **price** values as "sales" by each **product_name**.

product_name	sales
Benign Space Debris	8296.68
Curling 2014	6236.88
Dream Crusher	19755.06
Final Sequel	10495.80
Fire Resistance Suit of Provolone	1959.09
Holy Blade of Gouda	2443.92

```
index=web sourcetype=access_combined action=purchase status=200
| stats sum(price) as sales by product_name
```

7. Limit results to products with over $15,000 in sales by using the **where** command. Refer to the Common Commands and Functions page at the beginning of this document for the **where** command syntax.

product_name	sales
Dream Crusher	19755.06
Manganiello Bros.	18355.41
Orvil the Wolverine	15076.23

```
index=web sourcetype=access_combined action=purchase status=200
| stats sum(price) as sales by product_name
| where sales > 15000
```

8. Round **sales** values to the nearest whole number.

product_name	sales
Dream Crusher	19755
Manganiello Bros.	18355
Orvil the Wolverine	15076

```
index=web sourcetype=access_combined action=purchase status=200
| stats sum(price) as sales by product_name
| where sales > 15000
| eval sales = round(sales,0)
```

9. Sort values in descending order and rename **product_name** as "Best Sellers" and **sales** as "Total Revenue."

Best Sellers	Total Revenue
Dream Crusher	19755
Manganiello Bros.	18355
Orvil the Wolverine	15076

```
index=web sourcetype=access_combined action=purchase status=200
| stats sum(price) as sales by product_name
| where sales > 15000
| eval sales = round(sales,0)
| sort - sales
| rename product_name as "Best Sellers", sales as "Total Revenue"
```

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

OPTIONAL TASK: ITOps wants to see the two most common status codes for each of the web servers. Use the `top` command to identify common status codes by web server.

11. Search online sales data (**`index=web sourcetype=access_combined`**) and find the top 2 **`status`** code values during the **Last 24 hours**.

status	count	percent
200	537	87.601958
404	17	2.773246

```
index=web sourcetype=access_combined
| top limit=2 status
```

12. Edit your search so that results are split by the web server **`host`** values. Your results should display the number of events and the percentage of events that the top 2 **`status`** code values appear in for each web server.

host	status	count	percent
www1	200	153	84.530387
www1	404	7	3.867403
www2	200	154	88.000000
www2	503	5	2.857143
www3	200	231	89.534884
www3	505	7	2.713178

```
index=web sourcetype=access_combined
| top limit=2 status by host
```

13. Remove the **`count`** field with the **`fields`** command.

host	status	percent
www1	200	84.530387
www1	404	3.867403
www2	200	88.461538
www2	503	2.747253
www3	200	89.534884
www3	505	2.713178

```
index=web sourcetype=access_combined
| top limit=2 status by host
| fields - count
```

You can achieve this result using the `showcount` option of the `top` command. This option controls the appearance of the `count` column and accepts Boolean arguments (`t`/`true`/`1` or `f`/false/0.) See the Search Reference Manual for more information.

```
index=web sourcetype=access_combined
| top limit=2 showcount=f status by host
```

14. Sort results in ascending order by **`host`** and in descending order by **`percent`**.

host	status	percent
www1	200	85.353535
www1	404	3.535354
www2	200	88.461538
www2	503	2.747253
www3	200	89.534884
www3	505	2.713178

```
index=web sourcetype=access_combined
| top limit=2 showcount=f status by host
| sort host, -percent
```

15. Save your search as a report with the name **L3X**.