

Result Modification – Lab Exercises Guide

Overview

Welcome to the Splunk Education lab environment. These lab exercises will test your knowledge of various commands to manipulate results and normalize 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
	Active Directory	winauthentication_security	LogName, SourceName, EventCode, EventType, User
security	Badge reader	history_access	Address_Description, Department, Device, Email, Event_Description, First_Name, last_Name, Rfid, Username
	Business Intelligence server	sales_entries	AcctCode, CustomerID, TransactionID
sales	Retail sales	vendor_sales	categoryId, product_name, productId, sale_price, Vendor, VendorCity, VendorCountry, VendorID, VendorStateProvince
	Email security data	cisco_esa	dcid, icid, mailfrom, mailto, mid
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

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 <code>src_ip</code> 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 <code>SESSIONID</code> to 'The session ID' rename SESSIONID as "The session ID"
fields	command	Keeps (+) or removes (-) fields from search results.	Remove the <code>host</code> 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 <code>first_name</code> and <code>Last_name</code> values with a space to create a field called "full_name" eval full_name=first_name." ".last_name
table	command	Returns a table.	Output <code>vendorCountry</code> , <code>vendor</code> , and <code>sales</code> 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 <code>action</code> 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 – Manipulating Output

Description

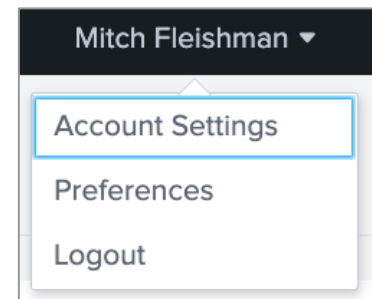
Configure the lab environment user account. Then, use the **xyseries** **untable**, and **bin** commands to manipulate output.

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 automatically begins the pipe on a new line.



Search auto-format disabled (default)



Search auto-format enabled

11. Navigate to the **Search & Reporting** app in the **Apps** dropdown.

Scenario: Networking needs to know the percentage of HTTP server errors that occurred on the e-commerce servers over the previous week.

Task 2: Use xyseries to create a chart-like search.

12. This search outputs the percentage of HTTP server errors that occurred on the e-commerce servers (`index=web sourcetype=access_combined`) grouped into 1-day time spans. Run this search over the **Previous week**.

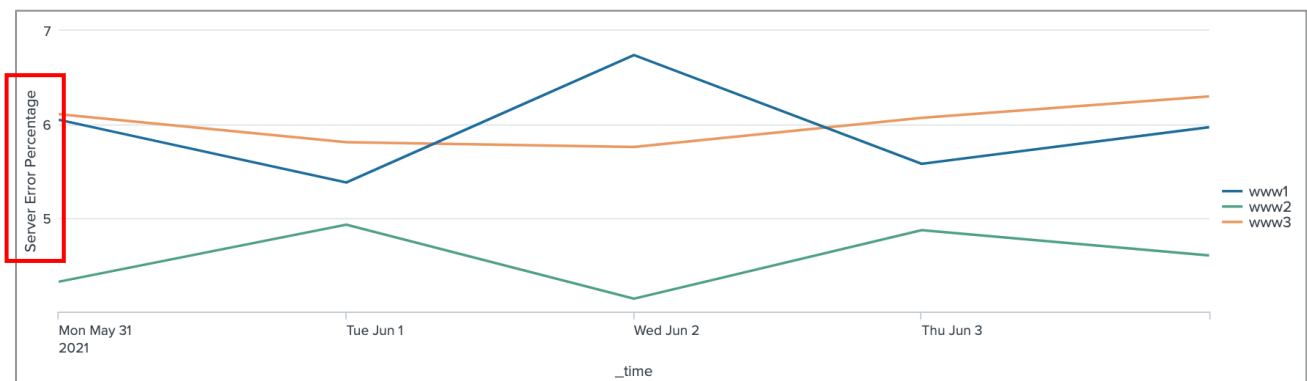
```
index=web sourcetype=access_combined
| bin _time span=1d
| stats count as total, count(eval(status>=500 AND status<=600)) as errors
  by host, _time
| eval percent = round((errors/total)*100,2)
```

host	_time	total	errors	percent
www1	2021-05-31	2843	172	6.05
www1	2021-06-01	2827	152	5.38
www1	2021-06-02	2895	195	6.74
www1	2021-06-03	3387	189	5.58
www1	2021-06-04	2678	160	5.97
www2	2021-05-31	2616	113	4.32
www2	2021-06-01	2657	131	4.93

13. Use the `xyseries` command to manipulate the `_time`, `host`, and `percent` fields into chart-like output. The resulting table should be formatted like the screenshot:

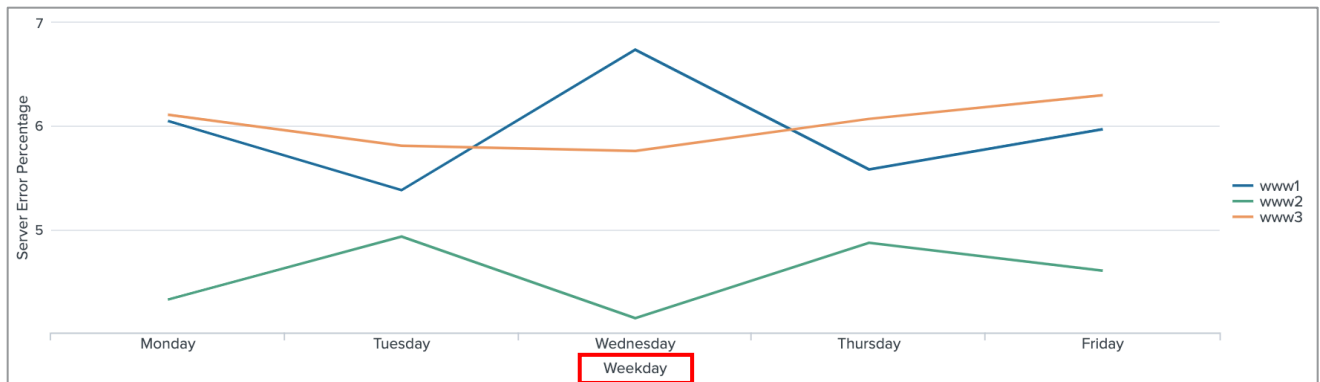
_time	www1	www2	www3
2021-05-31	6.05	4.32	6.11
2021-06-01	5.38	4.93	5.81
2021-06-02	6.74	4.14	5.76
2021-06-03	5.58	4.87	6.07
2021-06-04	5.97	4.60	6.30

14. Navigate to the **Visualization** tab and visualize results as a **Line chart**. Click **Format > Y-axis** and choose **Custom** from the **Title** drop-down menu. Label the **Y-axis** "Server Error Percentage".



NOTE: Step 15 is optional and requires knowledge of **rename** and **eval** commands. You can skip these steps and follow step 16 to save your search as a report.

15. Rename **_time** to "Weekday" and then format values to display the full day of the week. (Hint: The format variable for a full weekday is **%A**.)



16. Save your search as a report with the name **L1S1**.

- Click **Save As > Report**
- For **Title**, enter L1S1.
- Save.**
- 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.
- You can access your saved reports using the **Reports** tab in the application bar.
- Re-initialize the search window by clicking **Search** in the application bar.

Search
Analytics
Datasets
Reports
Alerts
Dashboards

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 All Yours This App's filter

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.

Scenario: IT would like to know the average bytes consumption of TCP_MISS requests from the web-security appliance data per website usage type over the last 24 hours.

Task 3: Convert chart-like output into a stats-like report.

17. This search finds events where an object had to be downloaded from an origin server (**action=TCP_MISS**) and calculates the average bytes consumed during this process for each website usage type. Run this search over the **Last 24 hours**. Are the values of **averages** sorted in descending order?

```
index=network sourcetype=cisco_wsa_squid action=TCP_MISS
| chart avg(sc_bytes) as averages over action by usage
| sort -averages
```

18. Modify the search so that the values of **averages** can be sorted. Then sort **averages** in descending order.

action	usage	averages
TCP_MISS	Personal	25747.24
TCP_MISS	Business	20156.25
TCP_MISS	Borderline	18246.2
TCP_MISS	Unknown	320

NOTE: Step 19 is optional. You can skip this step and continue to step 20.

19. Round the values of averages and list values by adding the following **eval** and **stats** commands to your search.

```
| eval averages = round(averages,0)
| stats list(usage) as usage, list(averages) as averages by action
```

action	usage	averages
TCP_MISS	Personal	25747
	Business	20156
	Borderline	14854
	Unknown	320

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

Scenario: Sales wants to know the 5-worst selling products last week and, of those products, the 3 most active customers by IP address.

Task 4: Use untable and xyseries to complete the search scenario.

21. This first objective of this scenario (find the 5 worst-selling products on the e-commerce servers by IP address) has been completed for you. Run this search over the **Previous week**.

```
index=web sourcetype=access_combined action=purchase status=200
| chart sum(price) by product_name, clientip limit=0
| addtotals
| sort 5 Total
| fields - Total
```

22. To find the top 3 most-active buyers of these products, you'll need to reorganize the results so that the **clientip** and **product_name** values are flipped. This allows you to calculate statistics about each customer's spending and find the top 3 most active **clientips**. Begin by converting your table into **stats**-like output with columns of data in this order: **product_name**, **clientip**, **sum**.

product_name	clientip	sum
Puppies vs. Zombies	107.3.146.207	4.99
Puppies vs. Zombies	110.138.30.229	14.97
Puppies vs. Zombies	111.161.27.20	4.99
Puppies vs. Zombies	117.21.246.164	9.98
Puppies vs. Zombies	118.142.68.222	4.99
Puppies vs. Zombies	12.130.60.5	9.98

23. Transform your **stats**-like output into chart-like output. Each row of data should represent the sales events associated with one **clientip**. The resulting table should be formatted like the screenshot:

clientip	Fire Resistance Suit of Provolone	Holy Blade of Gouda	Manganiello Bros. Tee	Puppies vs. Zombies	World of Cheese Tee
107.3.146.207	7.98	17.97			9.99
108.65.113.83	3.99	5.99			
109.169.32.135	3.99		29.97		9.99
110.138.30.229	3.99	5.99	9.99	4.99	9.99
110.159.208.78			9.99		
111.161.27.20		5.99			

24. Now, the data is structured so you can get a total sum for each **clientip**. Repeat lines 3 – 5 but modify the **sort** command so that it finds the top 3 **clientips**.

clientip	Fire Resistance Suit of Provolone	Holy Blade of Gouda	Manganiello Bros. Tee	Puppies vs. Zombies	World of Cheese Tee
87.194.216.51	11.97	17.97	19.98	19.96	39.96
94.229.0.20	3.99	11.98	19.98	9.98	39.96
91.210.104.143	3.99	11.98	19.98	9.98	29.97

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

Lab Exercise 2 – Modifying Results Sets

Description

Modify result sets by adding data with the **appendpipe** command and calculating statistics with the **eventstats** and **streamstats** commands.

Steps

Scenario: Sales Ops wants a table showing retail sales over the last 24 hours by category and product name with total sales for each category.

Task 1: Complete this search with the appendpipe command.

1. Modify this search so that the results of the first **stats** command are preserved and the **totalSales** for each **categoryId** is listed at the end of the search. Run the search over the **Last 24 hours**.

```
index=sales sourcetype=vendor_sales
| stats sum(price) as sales by categoryId, product_name
| stats sum(sales) as totalSales by categoryId
```

categoryId	product_name	sales	totalSales
ACCESSORIES	Fire Resistance Suit of Provolone	19.95	
ACCESSORIES	Holy Blade of Gouda	11.98	
ARCADE	Benign Space Debris	74.97	
ARCADE	Manganiello Bros.	119.97	
ARCADE	Orvil the Wolverine	199.95	
SHOOTER	World of Cheese	124.95	
SIMULATION	SIM Cubicle	139.93	
SPORTS	Curling 2014	39.98	
STRATEGY	Dream Crusher	199.95	
STRATEGY	Final Sequel	124.95	
STRATEGY	Mediocre Kingdoms	24.99	
STRATEGY	Puppies vs. Zombies	34.93	
TEE	Manganiello Bros. Tee	39.96	
TEE	World of Cheese Tee	29.97	
ACCESSORIES			31.93
ARCADE			394.89
SHOOTER			124.95
SIMULATION			139.93
SPORTS			39.98
STRATEGY			384.82

Your results should look similar to this screenshot after modifying the search.

2. Sort results in ascending order by **categoryId** so that **totalSales** values appear as the last event for each **categoryId**.

categoryId	product_name	sales	totalSales
ACCESSORIES	Fire Resistance Suit of Provolone	19.95	
ACCESSORIES	Holy Blade of Gouda	11.98	
ACCESSORIES			31.93
ARCADE	Benign Space Debris	74.97	
ARCADE	Manganiello Bros.	119.97	
ARCADE	Orvil the Wolverine	199.95	
ARCADE			394.89
SHOOTER	World of Cheese	124.95	
SHOOTER			124.95
SIMULATION	SIM Cubicle	139.93	
SIMULATION			139.93
SPORTS	Curling 2014	39.98	
SPORTS			39.98
STRATEGY	Dream Crusher	199.95	
STRATEGY	Final Sequel	124.95	
STRATEGY	Mediocre Kingdoms	24.99	
STRATEGY	Puppies vs. Zombies	34.93	
STRATEGY			384.82
TEE	Manganiello Bros. Tee	39.96	
TEE	World of Cheese Tee	29.97	

NOTE: Step 3 is optional and requires knowledge of the `eval` command. You can skip this step and follow step 4 to save your search as a report.

3. Modify your results by adding a description in the `sales` column that displays, "Category Subtotal".

categoryId	product_name	sales	totalSales
ACCESSORIES	Fire Resistance Suit of Provolone	19.95	
ACCESSORIES	Holy Blade of Gouda	11.98	
ACCESSORIES		Category Subtotal	31.93
ARCADE	Benign Space Debris	74.97	
ARCADE	Manganiello Bros.	119.97	
ARCADE	Orvil the Wolverine	199.95	
ARCADE		Category Subtotal	394.89
SHOOTER	World of Cheese	124.95	
SHOOTER		Category Subtotal	124.95
SIMULATION	SIM Cubicle	139.93	
SIMULATION		Category Subtotal	139.93
SPORTS	Curling 2014	39.98	
SPORTS		Category Subtotal	39.98
STRATEGY	Dream Crusher	199.95	
STRATEGY	Final Sequel	124.95	
STRATEGY	Mediocre Kingdoms	24.99	
STRATEGY	Puppies vs. Zombies	34.93	
STRATEGY		Category Subtotal	384.82
TEE	Manganiello Bros. Tee	39.96	
TEE	World of Cheese Tee	29.97	

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

Scenario: The retail sales manager wants to identify the retail products with lower-than-average sales across all products for the previous week.

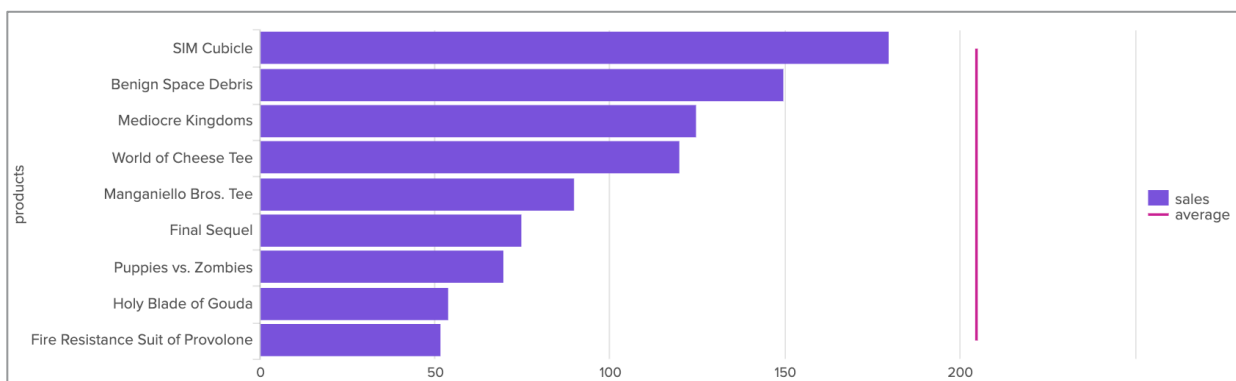
Task 2: Fill in the missing portions of a search.

5. Complete the `<missing>` portions of the search so that:
 - a. The `stats` command calculates the total sales of each product using the `price` field.
 - b. The `eventstats` command finds the average total sales across all products.
 - c. The `where` command filters search results for those events where the average total sales of all products was greater than the total sale of a specific product (Hint: You will compare the values of two fields.)
 - d. The `sort` command sorts results in descending order by `sales`.
 - e. Run the search over the **Previous week**.

```
index=sales sourcetype = vendor_sales
| stats <missing> as sales by product_name
| eventstats <missing> as average
| where <missing>
| sort -sales
```

product_name	sales	average
Benign Space Debris	4448.22	6967.801428571429
Curling 2014	4417.79	6967.801428571429
Manganiello Bros. Tee	3746.25	6967.801428571429
World of Cheese Tee	3196.80	6967.801428571429
Puppies vs. Zombies	2105.78	6967.801428571429
Holy Blade of Gouda	2000.66	6967.801428571429
Fire Resistance Suit of Provolone	1584.03	6967.801428571429

6. Switch to the **Visualization** tab and display the results as a **Bar Chart**. Display **average** as a chart overlay by clicking **Format > Chart Overlay** and typing in or choosing the **average** field for **Overlay**. Click **X-Axis** and choose **Custom** for **Title** then type "products".



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

Scenario: HR wants a report of the 3 most active users over the last month who were using the network for non-business activities. The report should include the username, event count by usage, total event count by user, and a ranking of their usage in descending order.

Task 3: Use streamstats to rank employees by their network activity.

- This search finds all non-business events from the web security appliance data. The **stats** command counts how many events are associated with each unique combination of **username** and **usage**. The calculations generated by the **stats** command are listed in the **count** column and then the results are sorted in descending order by the **sort** command. Run this search over the **Last 30 days**.

```
index=network sourcetype=cisco_wsa_squid usage!=Business
| stats count by username, usage
| sort -count
```

username	usage	count
erde	Personal	62
pleuchs	Personal	42
tzielinski	Personal	41
hsham	Personal	39
myavatkar	Personal	33
hsham	Unknown	26

- Use the **streamstats** command to create a field called "rank" that tallies each time a **username** value is encountered. Re-execute your search. You should notice how the rank values change as you scroll through the results. For example, in the screenshot below you will see that **acurry** is assigned two **rank** values. The **streamstats** command assigned a **rank** value of **1** the first time it encountered **acurry**. When the **streamstats** command encountered **acurry** a second time, it assigned a **rank** value of **2**. (Scroll through results to see the value for **rank** change for different values of **username**.)

username	usage	count	rank
mkemmerer	Personal	145	1
gfacello	Personal	114	1
acurry	Unknown	102	1
tzielinski	Personal	94	1
cganttchart	Personal	93	1
acurry	Personal	91	2
blu	Personal	82	1
cberztiss	Personal	80	1

NOTE: Step 10 is optional and requires knowledge of the **stats** and **sort** commands. You can skip this step and follow step 11 to save your search as a report.

10. Pipe search results to the **stats** command and list **rank**, **usage**, and **count** values and a "total" calculation using **sum(count)** by **username**. Use the **sort** command on the **total** field to limit your results to the top 3 **usernames**. (Hint: Use the **as** clause to keep column field names consistent.)

username	rank	usage	count	total
acurry	1	Unknown	102	195
	2	Personal	91	
	3	Violation	2	
mkemmerer	1	Personal	145	148
	2	Borderline	1	
	3	Unknown	1	
	4	Violation	1	
tzielinski	1	Personal	94	142
	2	Unknown	47	
	3	Borderline	1	

11. Save your search as a report with the name **L2S3**.

Lab Exercise 3 – Modifying Field Values

Description

Use the **foreach** command and **eval** text functions to modify multiple fields at once.

Steps

Scenario: The retail sales manager wants you to edit a sales report so that all product names are uppercase and all numerical values match the format \$x,xxx.

Task 1: Use the foreach command to format multiple fields at once.

1. Edit this search so that:

- All **product_name** values are uppercase.
- All **average** and **sales** values are in the USD currency format \$x,xxx.
- Run the search over the **Previous week**.

```
index=sales sourcetype=vendor_sales
| stats sum(price) as sales by product_name, categoryId
| eventstats avg(sales) as average
| where average > sales
| sort -sales
```

product_name	categoryId	sales	average
BENIGN SPACE DEBRIS	ARCADE	\$3,273.69	\$5,162.71
CURLING 2014	SPORTS	\$2,758.62	\$5,162.71
MANGANIELLO BROS. TEE	TEE	\$2,647.35	\$5,162.71
WORLD OF CHEESE TEE	TEE	\$2,407.59	\$5,162.71
PUPPIES VS. ZOMBIES	STRATEGY	\$1,556.88	\$5,162.71
FIRE RESISTANCE SUIT OF PROVOLONE	ACCESSORIES	\$1,260.84	\$5,162.71
HOLY BLADE OF GOUDA	ACCESSORIES	\$1,192.01	\$5,162.71

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

Lab Exercise 4 – Normalizing with eval

Description

Use `eval` functions to normalize data.

Steps

Scenario: SecOps wants to know the 5 employees who were most active on the network during the previous business week.

Task 1: The employee username data from the network index is not normalized. Use the `eval` command to normalize fields from two different sourcetypes.

1. Complete the `<missing>` portions of the search so that:
 - a. Values of `username` and `Username` are normalized under a new field called "User"
 - b. The `stats` command counts events by `User`.
 - c. Results are sorted so that the 5 most active users are listed.
 - d. Run the search over the **Previous business week**.

```
index=network sourcetype=cisco_firewall OR sourcetype=cisco_wsa_squid)
| <missing>
| stats <missing>
| <missing>
```

User	count
jreistad	106
gbrowser	87
gzuyeva	76
pbunch	76
yschonegge	76

2. Save your search as a report with the name **L4S1**.

Scenario: Security Operations believes that an individual without legitimate credentials may have gained access to the premises. Therefore, they would like to verify the usernames of everyone who has badged into the building over the last 7 days.

Task 2: Create a new field from two string fields using the `substr` and `lower` functions. Then, use the `case` function to assign "Good Username" and "Bad Username" values.

3. This search finds all badge reader events from the `security` index. Then, the `where` command limits results to events where the `Last_Name` field is not null and the `dedup` command removes duplicate `Username` values in the events. Use the `table` command to output a table with `First_Name`, `Last_Name`, and `Username` values. Run the search over the **Last 7 days**.

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

```
| where isnotnull(Last_Name)
| dedup Username
```

First_Name	Last_Name	Username
Daniil	Piazza	dpiazza
Amanda	Curry	acurry
Suzanne	Flaemmchen	sflaemmchen
Pat	Leuchs	pleuchs
Sergei	Voronoff	svoronoff
Robert	Roberts	rroberts
Placido	Toscani	ptoscani
Rao	Jayaraman	rjayaraman

NOTE: Ignore any search reference cycle error messages that may appear when this search is run. The correct data will still be displayed.

- Create a new field called "AltUsername". Use the **substr** function to create this field. The resulting values will contain the first letter of **First_Name** concatenated with the full value of **Last_Name**.

First_Name	Last_Name	Username	AltUsername
Daniil	Piazza	dpiazza	DPiazza
Amanda	Curry	acurry	ACurry
Suzanne	Flaemmchen	sflaemmchen	SFlaemmchen
Pat	Leuchs	pleuchs	PLeuchs
Sergei	Voronoff	svoronoff	SVoronoff
Robert	Roberts	rroberts	RRoberts
Placido	Toscani	ptoscani	PToscani
Rao	Jayaraman	rjayaraman	RJayaraman

- Convert **AltUsername** values to lowercase. (Note: This can be done in the same pipe.)

First_Name	Last_Name	Username	AltUsername
Ramzi	Erde	rerde	rerde
Phyllis	Bunch	pbunch	pbunch
Allen	Pucci	apucci	apucci
Guoxiang	Nooteboom	gnooteboom	gnooteboom
Yurij	Schonegge	yschonegge	yschonegge

- Use the **eval** command to create a new field called "CheckUsername" and move the **table** command:

- a. Pipe results to the eval command and the case function to create a field called **CheckUsername** that will check the values of **AltUsername** against the values of **Username**:

```
| eval CheckUsername = case(AltUsername=Username,"Good Username",
AltUsername!=Username,"Bad Username")
```

If the values are the same, a value of "Good Username" is assigned, if the values are not the same, a value of "Bad Username" is assigned.

- b. Move and revise the **table** command so that the resulting table contains **Username**, **AltUsername**, and **CheckUsername** in this order.

Username ↕	AltUsername ↕	CheckUsername ↕
dpiazza	dpiazza	Good Username
acurry	acurry	Good Username
sflaemmchen	sflaemmchen	Good Username
pleuchs	pleuchs	Good Username
svoronoff	svoronoff	Good Username
rroberts	rroberts	Good Username

7. Sort the table so that any events with a bad username appear at the top.

Username ↕	AltUsername ↕	CheckUsername ↕
pbridgland	cbridgland	Bad Username
kpercy	kpercy	Good Username
mkemmerer	mkemmerer	Good Username
iking	iking	Good Username
kjoslin	kjoslin	Good Username
fullian	fullian	Good Username

8. Save your search as a report with the name **L4S2**.

NOTE: The following challenge exercise is optional and requires knowledge of the **stats**, **eval**, and **fields** commands.

CHALLENGE: The retail sales manager wants to know the number of sales, the average sale price, and the total sales for each category over the previous business week in retail stores.

- Run the search below over the **Previous business week**. Notice that **appendpipe** is not generating results. Why is this happening? (Hint: Use the **typeOf** function to see how Splunk is interpreting the data. For information on how to use this function, see the [Search Reference Manual](#).)

```
index=sales sourcetype=vendor_sales
| stats count(price) as NumberofSales, avg(price) as AverageSales, sum(price) as
  TotalSales by categoryId
| eval AverageSales = "$".toString(AverageSales,"commas"),
  TotalSales="$".toString(TotalSales,"commas")
| appendpipe
  [stats sum(TotalSales) as Total
   | eval TotalSales = "$".toString(Total,"commas")
   | eval AverageSales = "Grand Total"
   | fields - Total]
```

With the information gained from the previous step, rewrite the search to fulfill the scenario request.

categoryid	NumberofSales	AverageSales	TotalSales
ACCESSORIES	758	\$4.76	\$3,608.42
ARCADE	722	\$36.67	\$26,472.78
SHOOTER	455	\$24.99	\$11,370.45
SIMULATION	449	\$19.99	\$8,975.51
SPORTS	246	\$19.99	\$4,917.54
STRATEGY	1550	\$23.34	\$36,184.50
TEE	659	\$9.99	\$6,583.41
	Grand Total		\$98,112.61

- Save your search as a report with the name **L4X**.