Multivalue Fields – Lab Solutions Guide

Overview

Welcome to the Splunk Education lab environment. These lab exercises will test your knowledge of searching, creating, modifying, and manipulating multivalue data with multivalue commands and functions.

Scenario

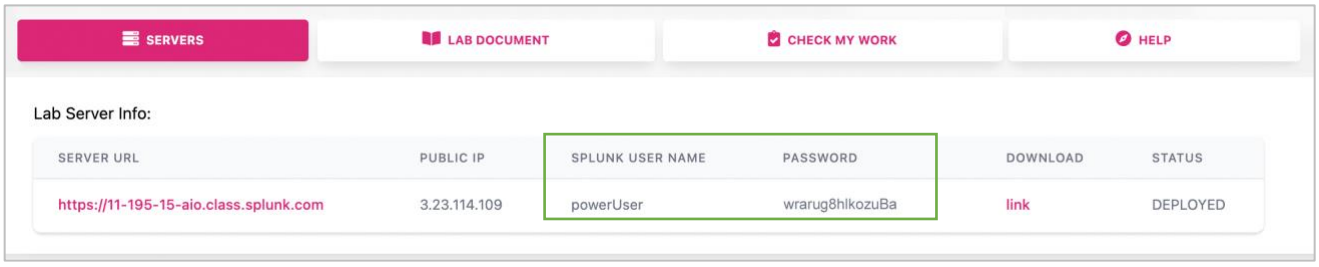
You will use data from the international video game company, Buttercup Games. A list of source types is provided below.

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
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
systems	Linux system log	**server_log**	Active Ram, Available Ram, CPU Percent Used, Free Ram, Inactive Ram, RAM Percent Used, Total Ram, Used Ram
	HTTP status code definitions	**status_definitions**	status, status_description, status_type
	AWS system data	**system_info**	CPU_CORES{}.core, CPU_CORES{}.core_percent_used, CPU_CORES{}.system, CPU_CORES{}.user, RAM.active, RAM.available, RAM.total, RAM.used, ROOT_USERS{}, SYSTEM{}

Lab Connection Info

Access labs using the server URL, user name, and password shown in your lab environment.



splunk>

Common Commands and Functions

These commands and statistical functions are commonly used in searches but may not have been explicitly discussed in the module. 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 – What are Multivalue Fields?

Description

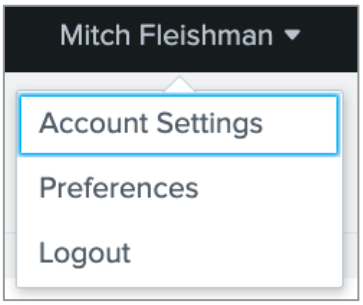
Configure the lab environment user account. Then, you will use **`spath`** to interpret self-describing data and multivalue **`stats`** functions to convert single-value fields to multivalue fields.

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 welcoming you to the lab environment. You can click **Continue to Tour** but this is not required. Click **Skip** to dismiss the pop-up 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.

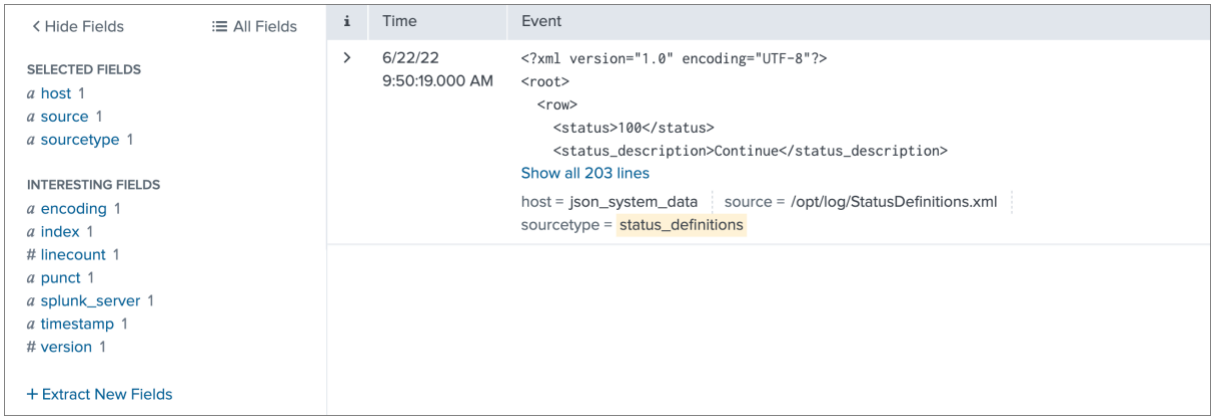
7. Navigate to ***user name*** **> Preferences.**
8. Choose your local time zone from the **Time zone** drop-down list.
9. Click **Apply**.
10. (Optional) Navigate to ***user name*** **> Preferences > SPL Editor > Search auto-format** and click on the toggle to activate auto-formatting. Then click **Apply**.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

Search auto-format disabled (default)

Search auto-format enabled

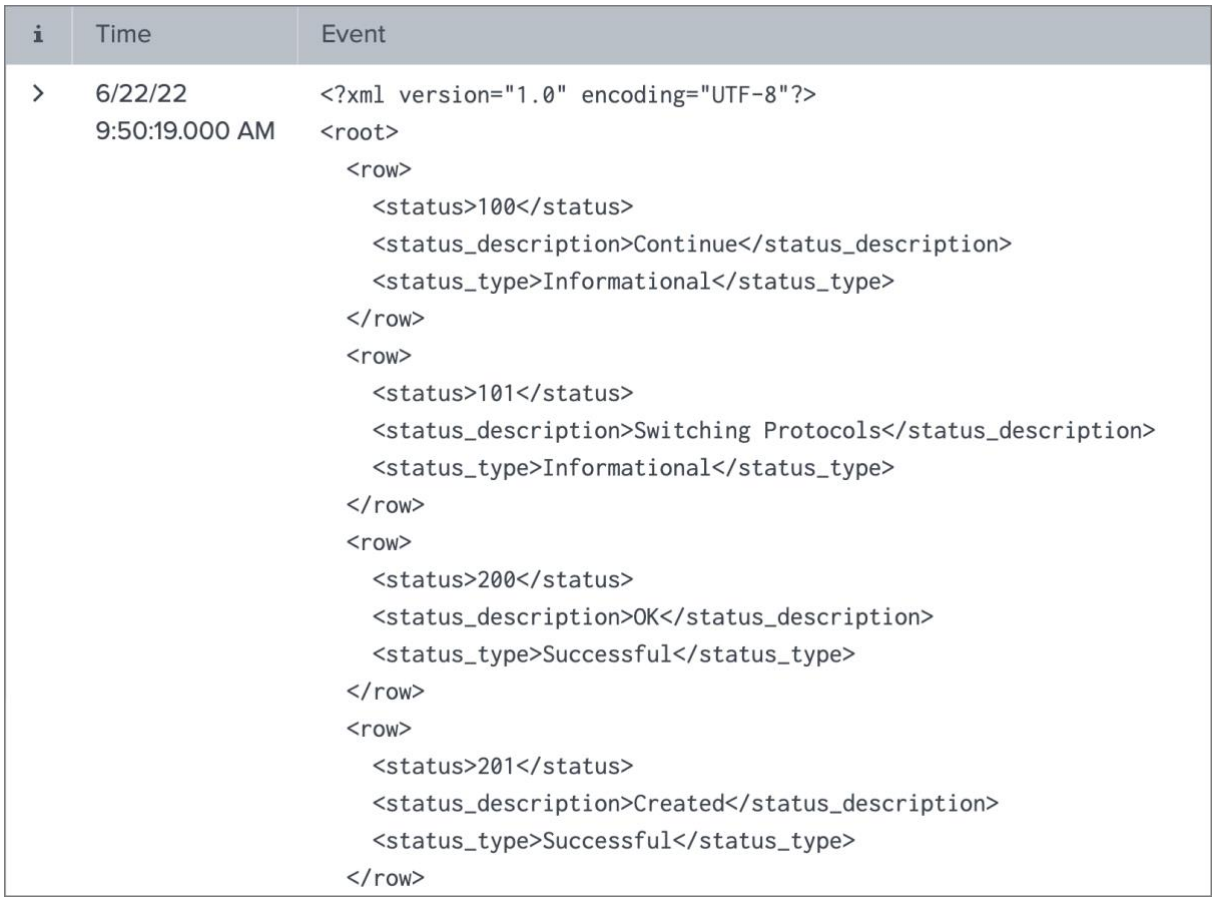
Task 2: Extract fields using the `spath` command.

11. In the top left corner of Splunk Web, select **Apps** > **Search & Reporting**. This sets the app context to the search app.
12. Search HTTP status definitions (**`index=systems sourcetype=status_definitions`**) data over **All time**.




```
index=systems sourcetype=status_definitions
```

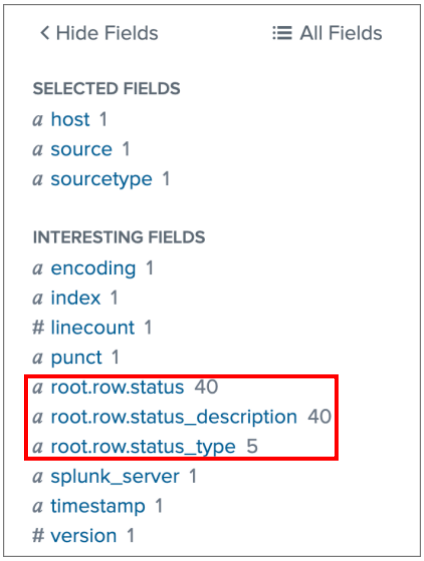
13. Expand the event details by clicking **Show all 203 lines**. Notice that this one event is in XML format.



14. Extract fields, over **All time**, by using the **`spath`** command.

```
index=systems sourcetype=status_definitions
| spath
```

15. Look at your **Interesting Fields** list. New fields should now be available.



16. Display a table containing the **`root.row.status`**, **`root.row.status_description`**, and **`root.row.status_type`** from the extracted XML.

root.row.status	root.row.status_description	root.row.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

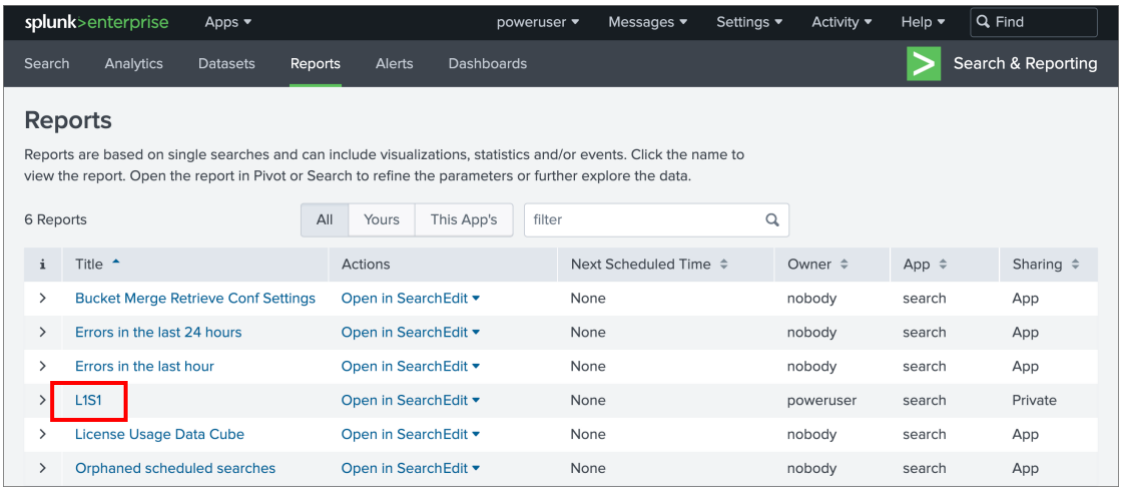
```
index=systems sourcetype=status_definitions
| spath
| table root.*
```

17. Rename the columns as **`status`**, **`status_description`**, and **`status_type`**.

status	status_description	status_type
100	Continue	Informational
101	Switching Protocols	Informational
200	OK	Successful
201	Created	Successful
202	Accepted	Successful

```
index=systems sourcetype=status_definitions
| spath
| table root.*
| rename root.row.* as *
```

18. 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.



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

Task 3: Extract fields from an XML file using the `spath` function of the `eval` command.

19. Reuse the previous search but use the **`eval`** command with the **`spath`** function to extract and create **`root.row.status`** as **`status`**, **`root.row.status_description`** as **`description`** and **`root.row.status_type`** as **`type`**. Search over **All Time** and display results in a table as shown:

```
index=systems sourcetype=status_definitions
| eval status = spath(_raw,"root.row.status"),
  description = spath(_raw,"root.row.status_description"),
  type = spath(_raw,"root.row.status_type")
| table status, description, type
```

status	description	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

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

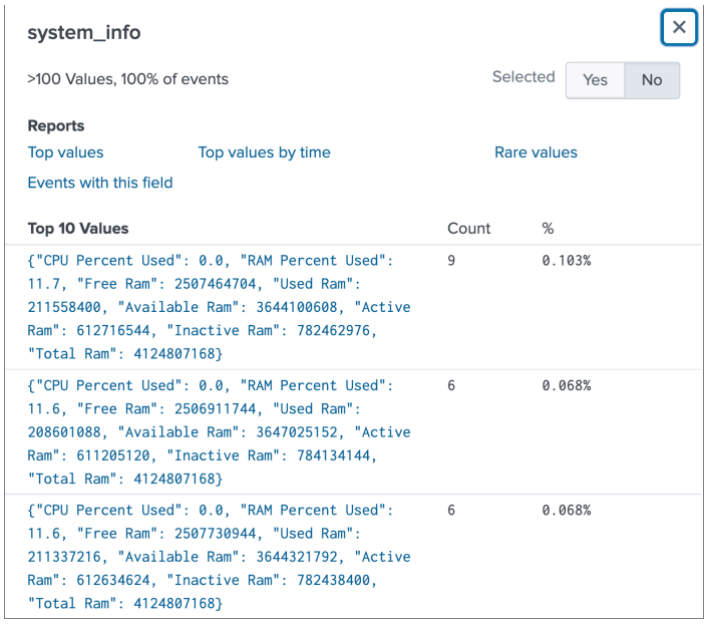
Scenario: ITOps wants to analyze performance of a Linux server based on a system log.

Task 4: Display a table showing the performance of the server over the course of the last 24 hours.

21. Search for all events in the linux system log (**`index=systems sourcetype=server_log`**) over the **Last 24 hours**.

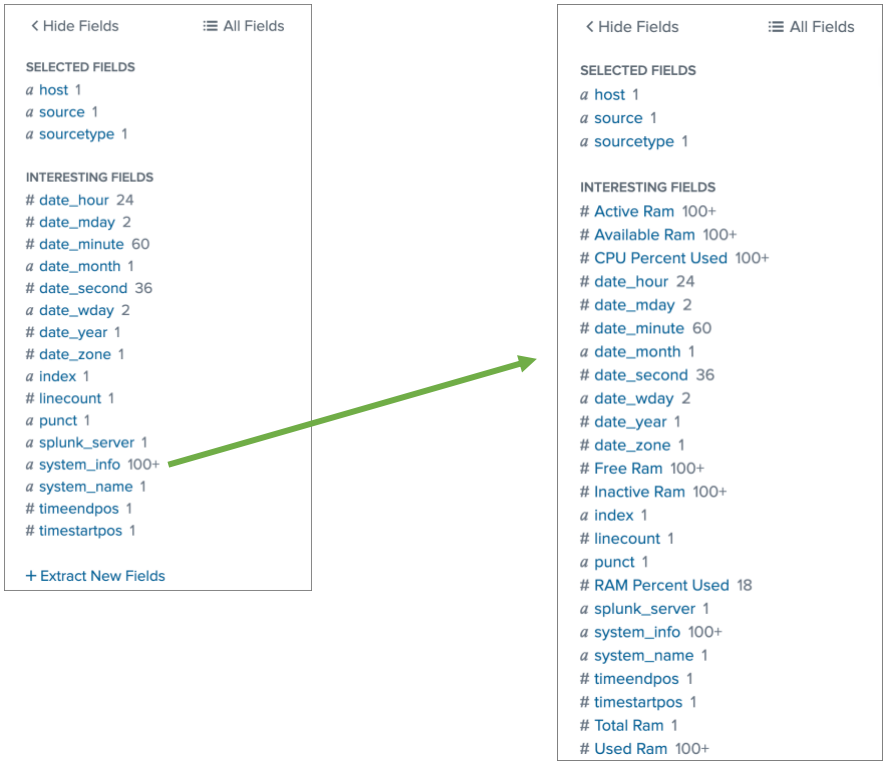
```
index=systems sourcetype=server_log
```

22. Click the **`system_info`** field in the **Interesting Fields** list. Note that the values are in JSON format.



23. Use the **`spath`** command to extract fields specifically from the **`system_info`** field. Notice the new fields that are now listed under **Interesting Fields**.

```
index=systems sourcetype=server_log
| spath input=system_info
```



24. Display a table, for the **Last 24 hours**, showing **`_time`**, **`Used Ram`**, **`Free Ram`**, **`RAM Percent Used`**, and **`CPU Percent Used`**.

```
index=systems sourcetype=server_log
| spath input=system_info
| table _time, "Used Ram", "Free Ram", "RAM Percent Used", "CPU Percent Used"
```

_time	Used Ram	Free Ram	RAM Percent Used	CPU Percent Used
2022-06-22 11:37:29	208412672	2504355840	11.6	0.5
2022-06-22 11:37:19	208162816	2504613888	11.6	33.3
2022-06-22 11:37:09	208162816	2504613888	11.6	0.0
2022-06-22 11:36:59	208166912	2504613888	11.6	20.4
2022-06-22 11:36:49	205340672	2507460608	11.5	2.5
2022-06-22 11:36:39	205340672	2507460608	11.5	12.5

NOTE: The system information extracted does not include all types of memory usage for the server (e.g., inactive RAM, wired RAM). The **`RAM Percent Used`** values represent a percentage of **total** RAM.

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

Scenario: Sales Ops wants a table displaying the number of successful online purchases during the previous week by web host and category to see which types were purchased most.

Task 5: Use a multivalue `stats` function to complete a search.

26. Complete the **`<missing>`** portion of the following search so that all **`categoryId`** and **`count`** values are listed by **`host`**. Run this search over the **Previous week**.

```
index=web sourcetype=access_combined action=purchase status=200 categoryId=*
| stats count by host, categoryId
| sort -count
| <missing>
```

host	categoryId	count
www1	STRATEGY	204
	ARCADE	130
	ACCESSORIES	92
	TEE	64
	SHOOTER	62
	SIMULATION	49
	SPORTS	37
www2	STRATEGY	192
	ARCADE	120
	ACCESSORIES	102
	TEE	85
	SIMULATION	50
	SHOOTER	43
	SPORTS	38

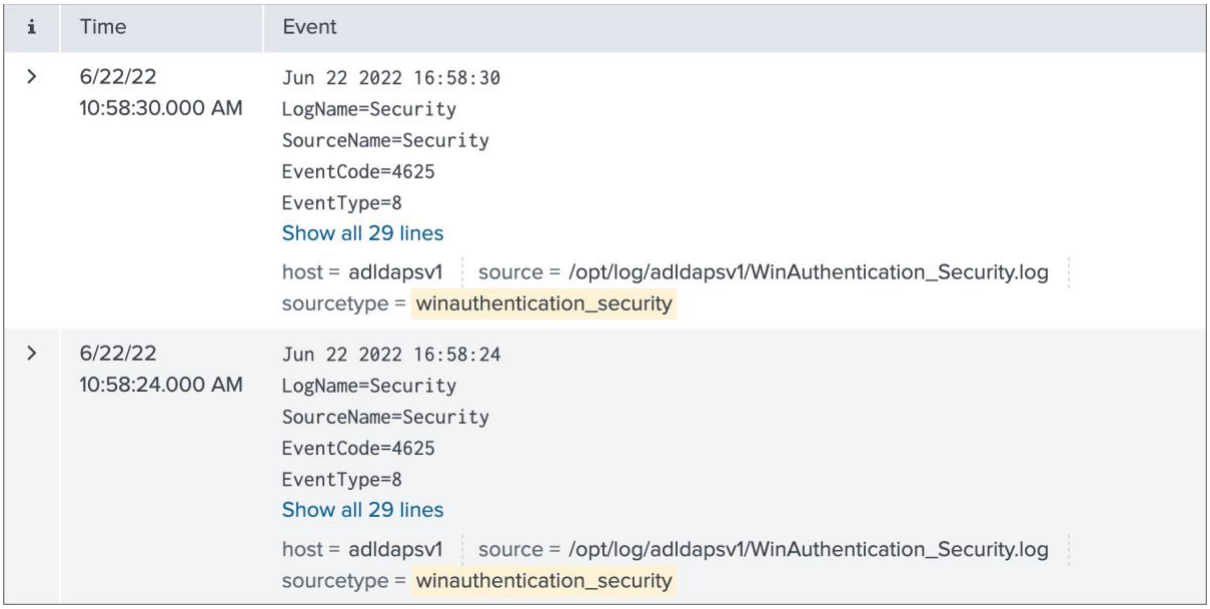
```
index=web sourcetype=access_combined action=purchase status=200 categoryId=*
| stats count by host, categoryId
| sort -count
| stats list(categoryId) as categoryId, list(count) as count by host
```

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

Scenario: ITOps wants to see all unique users active on the AD/DNS server during the last 4 hours.

Task 6: Use a multivalue `stats` function to list unique values.

28. Search Active Directory data (**`index=security sourcetype=winauthentication_security`**) for events over the **Last 4 hours**.




```
index=security sourcetype=winauthentication_security
```

29. Use a multivalue **`stats`** function to list all unique values of **`User`**. Retain "User" as the field name by using an **`as`** clause.


```
index=security sourcetype=winauthentication_security
| stats values(User) as User
```

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

Lab Exercise 2 – Create and Evaluate Multivalue Fields

Description

In this lab exercise, you will use the commands you learned in class to create and evaluate multivalue fields.

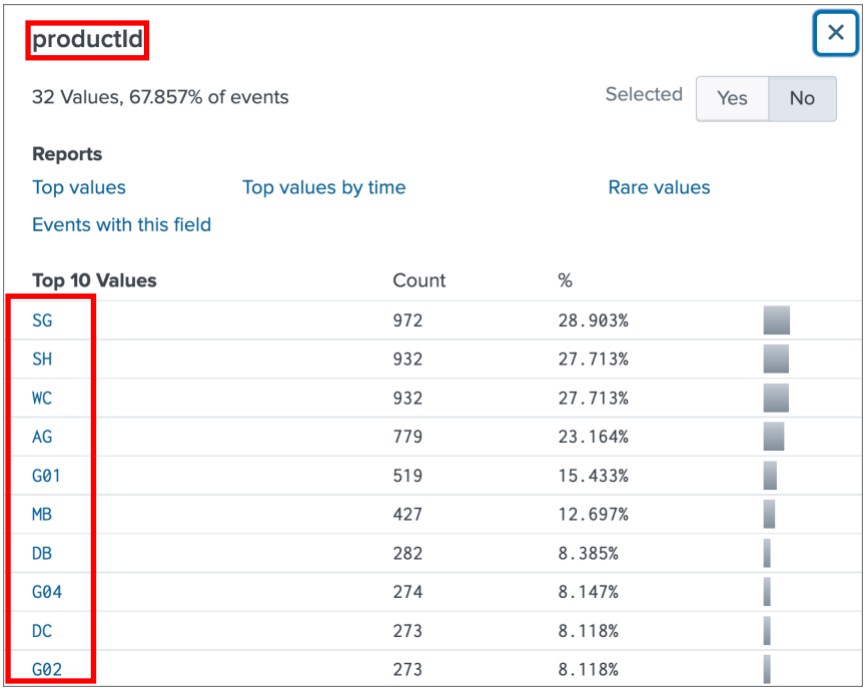
Steps

Scenario: Show a count for all products sold online yesterday by `product_name` whose `productId` contains "`SH`".

Task 1: Use makemv to convert a single-value field to a multivalue field.

NOTE: There are other ways to accomplish this search – for example, by using a **`where`** command with a **`match`** function and specifying a regex pattern. However, in this task, you will use **`makemv`** to reinforce what you learned about multivalued functions in the lecture.

1. Search all Buttercup games online sales events (**`index=web sourcetype=access_combined`**) from **Yesterday**.
   ```
   index=web sourcetype=access_combined
   ```
2. In the **Interesting Fields** list, look at how the values of **`productId`** are structured, e.g. AA-BB-CC1. Use the **`makemv`** command to split the values of **`productId`** into 3 groupings, e.g. AA, BB, CC1. In other words, use **`makemv`** to convert the single-value field **`productId`** into a multivalue field containing 3 values without the dash (**`-`**) character.
   ```
   index=web sourcetype=access_combined
   | makemv delim="-" productId
   ```
3. In the **Interesting Fields** list, you will see how the **`productId`** field has been split up.



4. Only keep results where the **productId** contains "SH" by using the **search** command with the expression, **productId = "SH"**.

```
index=web sourcetype=access_combined
| makemv delim="-" productId
| search productId = "SH"
```

NOTE: Step 5 is optional and requires knowledge of the **stats** command. You can skip this step and continue to step 6 to save your search as a report.

5. Count events by **product_name**.

product_name	count
Fire Resistance Suit of Provolone	478
Holy Blade of Gouda	490
World of Cheese	580
World of Cheese Tee	390

```
index=web sourcetype=access_combined
| makemv delim="-" productId
| search productId = "SH"
| stats count by product_name
```

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

Scenario: Security has requested that you develop a search that will identify any Buttercup Games employees who used a workstation other than their own during the last 7 days. The workstation name should be included in the saved report.

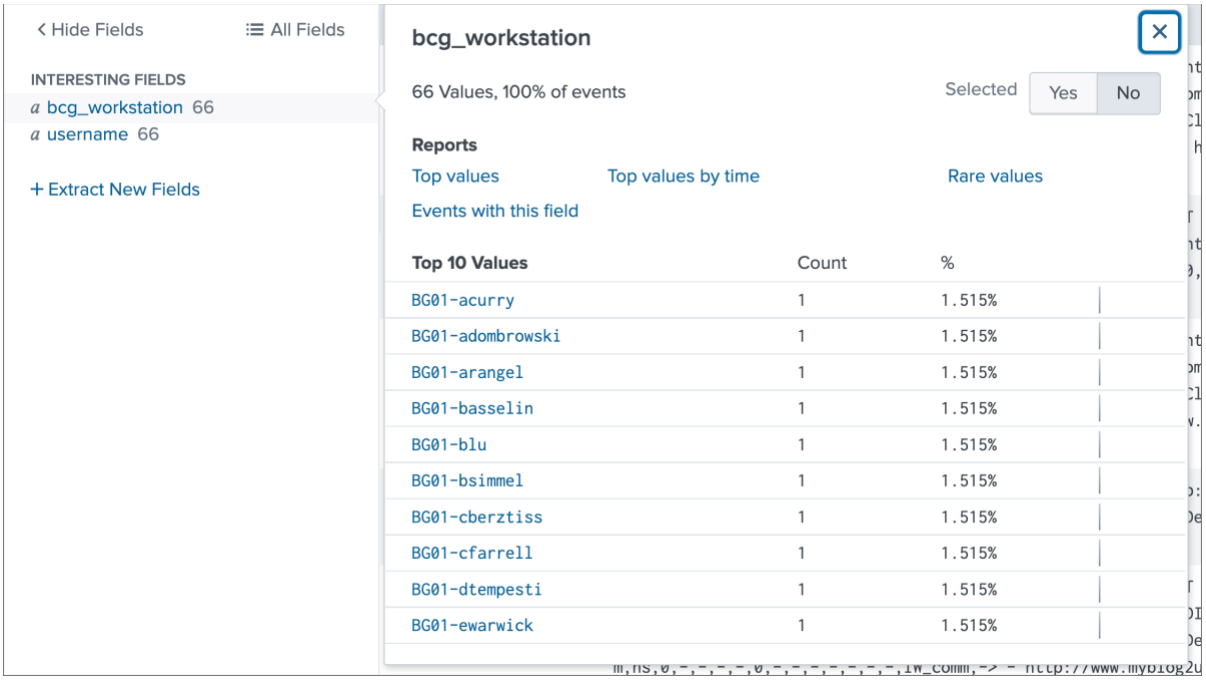
Task 2: Use multivalue `eval` functions to complete a search. Then, display results as a table and filter the search results.

NOTE: At Buttercup Games, employees are required to use ONLY their company assigned workstations (i.e., their assigned desktop or laptop computer.) Therefore, it would be unusual for an employee to use a machine belonging to another employee. Thus, *one would expect that this search would generally produce no results.*

7. Run this search over the **Last 7 days**. This search finds events from the web security appliance data and:
   a. Limits results to just **bcg_workstation** and **username** values using the **fields** command.
   b. Further filters results to just unique combinations of **bcg_workstation** and **username** using the **dedup** command.

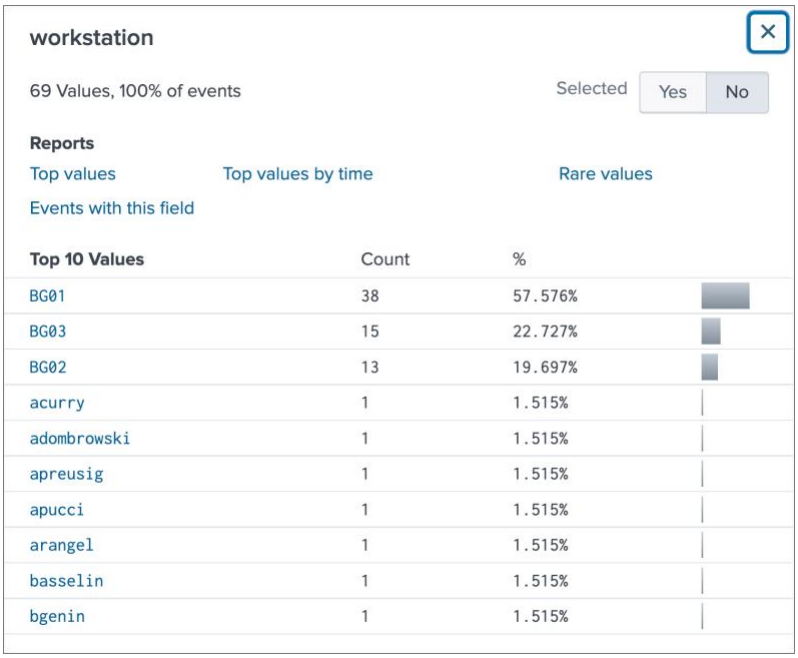
```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
```

8. Review the values for **`username`** and **`bcg_workstation`**. Notice how all the **`bcg_workstation`** values have the same naming convention: **`BG0x-username`**, for example **`BG01-acurry`**, etc.



9. Create a new multivalue field with the **`eval`** command called "workstation" whose values are made from splitting up **`bcg_workstation`** using the (**`-`**) dash character.

```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
```



10. Create a new single value field called "workstation_user" whose value is the username portion of **`workstation`**. (Hint: The function you will be using has two arguments, a multivalue field and an integer referencing the position of a value within that multivalue field's index.)

```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
| eval workstation_user = mvindex(workstation,1)
```

11. Use the **`table`** command to display **`workstation_user`** and **`username`** values side by side in a table.

username	workstation_user
jcappelletti	jcappelletti
msluis	msluis
spahkthecah	spahkthecah
svoronoff	svoronoff
acurry	acurry
fyards	fyards
gvoronoff	gvoronoff
myuan	myuan
syoungin	syoungin
pdabbeville	pdabbeville

```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
| eval workstation_user = mvindex(workstation,1)
| table username workstation_user
```

12. Use the **`where`** command to find Buttercup Games employees who used a workstation that was not their own. Since all employees are required to take mandatory security and compliance training every week and therefore, should only be using their own workstations, your search should return no results.


```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
| eval workstation_user = mvindex(workstation,1)
| table username workstation_user
| where username!=workstation_user
```

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

Challenge: Test your search.

14. Since your search returned no results, you should verify that the search worked as you intended. Simulate the scenario of a user logging into another user's workstation. To do so, edit the **<missing>** portion of this search so that the username **rroberts** will be changed to "rsanchez" but all other values will remain the same. Execute the search over the **Last 7 days**. (Hint: This search uses an **eval** function that was not discussed in the slides. Refer to the Search Manual to find an **eval** function that will evaluate the expression **username=rroberts** and return **rsanchez** if true and **username** if false.)

```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
| eval workstation_user = mvindex(workstation,1)
| table username workstation_user
| eval username = <missing>
| where username!=workstation_user
```

username	workstation_user
rsanchez	rroberts

```
index=network sourcetype=cisco_wsa_squid
| fields bcg_workstation username
| dedup bcg_workstation username
| eval workstation = split(bcg_workstation,"-")
| eval workstation_user = mvindex(workstation,1)
| table username workstation_user
| eval username = if(username="rroberts","rsanchez",username)
| where username!=workstation_user
```

15. Save your search as a report with the name **L2X**.

Lab Exercise 3 – Analyze Multivalue Fields

Description

In this lab exercise, you will use the commands you learned in class to analyze multivalue fields.

Steps

Scenario: Create a report that will display each AWS system's CPU core along with a colon-separated list of the percent used and average percent used for each core.

Task 1: Use multivalue `eval` functions to complete a search.

1. Search AWS system data (**`index=systems sourcetype=system_info`**) for systems in use during the **Last 60 minutes**. Then rename **`SYSTEM{}`** to "SYSTEM", **`CPU_CORES{}.core_percent_used`** to "core_percent_used", and **`CPU_CORES{}.core`** to "cpu_core".

```
index=systems sourcetype=system_info
| rename SYSTEM{} as SYSTEM, CPU_CORES{}.core_percent_used as core_percent_used,
  CPU_CORES{}.core as cpu_core
```

2. Create a new field called "cpu_percent_used" that concatenates **`core_percent_used`** values by a **`:`** (colon).

```
index=systems sourcetype=system_info
| rename SYSTEM{} as SYSTEM, CPU_CORES{}.core_percent_used as core_percent_used,
  CPU_CORES{}.core as cpu_core
| eval cpu_percent_used = mvjoin(core_percent_used,":")
```

3. Calculate the average of **`core_percent_used`** by **`SYSTEM`**, **`cpu_percent_used`**, and **`cpu_core`**. Name this average "average_cpu_used".

SYSTEM	cpu_percent_used	cpu_core	average_cpu_used
nix_system_idx_1	0.0:0.0	core_number_1	0
nix_system_idx_1	0.0:0.0	core_number_2	0
nix_system_idx_1	0.0:21.0	core_number_1	10.5
nix_system_idx_1	0.0:21.0	core_number_2	10.5
nix_system_idx_1	0.0:24.0	core_number_1	12
nix_system_idx_1	0.0:24.0	core_number_2	12

```
index=systems sourcetype=system_info
| rename SYSTEM{} as SYSTEM, CPU_CORES{}.core_percent_used as core_percent_used,
  CPU_CORES{}.core as cpu_core
| eval cpu_percent_used = mvjoin(core_percent_used,":")
| stats avg(core_percent_used) as average_cpu_used by SYSTEM, cpu_percent_used,
  cpu_core
```

NOTE: In this environment, the **`SYSTEM`** name value is not unique per AWS instance. The **`cpu_percent_used`** that you created a moment ago is unique per AWS instance. Therefore, to obtain the average usage per instance, you need not only **`SYSTEM`**, but also the **`cpu_percent_used`**.

4. Finally, list the values of **average_cpu_used** and **cpu_percent_used** by **SYSTEM** and **cpu_core**. Don't forget to include an **as** clause so that field names are preserved.

SYSTEM	cpu_core	average_cpu_used	cpu_percent_used
nix_system_idx_1	core_number_1	0	0.0:0.0
		7	0.0:14.0
		19	0.0:38.0
		27.5	14.0:41.0
		17.4	17.8:17.0
		21.1	21.2:21.0
		24.5	24.0:25.0
		51.85	37.0:66.7
nix_system_idx_1	core_number_2	0	0.0:0.0
		7	0.0:14.0
		19	0.0:38.0
		27.5	14.0:41.0
		17.4	17.8:17.0
		21.1	21.2:21.0
		24.5	24.0:25.0
		51.85	37.0:66.7
nix_system_idx_1_UK	core_number_1	0	0.0
		14	14.0

```
index=systems sourcetype=system_info
| rename SYSTEM{} as SYSTEM, CPU_CORES{}.core_percent_used as core_percent_used,
  CPU_CORES{}.core as cpu_core
| eval cpu_percent_used = mvjoin(core_percent_used,":")
| stats avg(core_percent_used) as average_cpu_used by SYSTEM, cpu_percent_used,
  cpu_core
| stats list(average_cpu_used) as average_cpu_used, list(cpu_percent_used) as
  cpu_percent_used by SYSTEM, cpu_core
```

SYSTEM	cpu_core	average_cpu_used	cpu_percent_used
nix_system_idx_1	core_number_1	0	0.0:0.0
		10.5	0.0:21.0
		12	0.0:24.0
		15	0.0:30.0
		17.5	0.0:35.0
		17.8	0.0:35.6
		2.5	0.0:5.0
		13.55	15.0:12.1
		24	20.0:28.0
		21	21.0:21.0

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

Challenge: Modify this search so that `CPU_CORES{}.core` has `_number` removed from each of its values. For example, `core_number_1` would become `core_1`, `core_number_2` would become `core_2`, `core_number_3` would become `core_3`, etc.)

6. Complete the **`<missing>`** portion of this search and run the search over the **Last 24 hours**.. You will be using a multivalue **`eval`** function and the **`replace`** function. (The **`replace(X,Y,Z)`** function was not discussed in the slides. This function is a text function that substitutes the string **`Z`** for every occurrence of regex string **`Y`** in string **`X`**. See the Search Manual for more information about the **`replace`** function.)

```
index=systems sourcetype=system_info
| rename SYSTEM{} as system, CPU_CORES{}.core as core
| rename CPU_CORES{}.core_percent_used as percent_used
| eval core = <missing>
| eval zip_percent_used = mvzip(core, percent_used,":")
| stats count as sum_core by system, asctime, zip_percent_used
| search zip_percent_used!=": 0.0"
| stats list(zip_percent_used) as "CPU Core Usage" by system, asctime
```

system	asctime	CPU Core Usage
nix_system_idx_1	2022-06-22T19:40:56+00:00	core_1:0.0 core_2:0.0
nix_system_idx_1	2022-06-22T19:42:16+00:00	core_1:36.0 core_2:36.0
nix_system_idx_1	2022-06-22T19:43:36+00:00	core_1:54.0 core_2:20.0
nix_system_idx_1	2022-06-22T19:44:56+00:00	core_1:62.0 core_2:53.5
nix_system_idx_1	2022-06-22T19:46:16+00:00	core_1:62.0 core_2:57.0
nix_system_idx_1	2022-06-22T19:47:36+00:00	core_1:0.0 core_2:0.0

```
index=systems sourcetype=system_info
| rename SYSTEM{} as system, CPU_CORES{}.core as core
| rename CPU_CORES{}.core_percent_used as percent_used
| eval core = mvmap(core, replace(core,"(core_number_)","core_"))
| eval zip_percent_used = mvzip(core, percent_used,":")
| stats count as sum_core by system, asctime, zip_percent_used
| search zip_percent_used!=": 0.0"
| stats list(zip_percent_used) as "CPU Core Usage" by system, asctime
```

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