Working With Time – Lab Guide

Overview

Welcome to the Splunk Education lab environment. These lab exercises will familiarize you with working with time through searches.

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
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
	Email security data	**cisco_esa**	dcid, icid, mailfrom, mailto, mid

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 – Searching with Time

Description

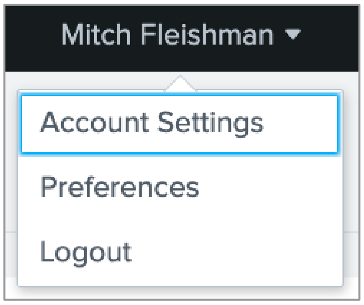
Configure the lab environment user account. Then, use the `bin` command to group search results.

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.

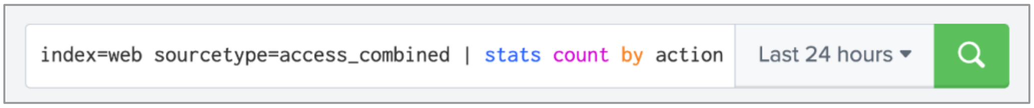
1. Log into your Splunk lab environment using the username and password provided to you.
2. You may see a pop-up window welcoming you to the lab environment. You can click **Continue to Tour** but this is not required. Click **Skip** to dismiss the window.
3. Click on the username you logged in with (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.

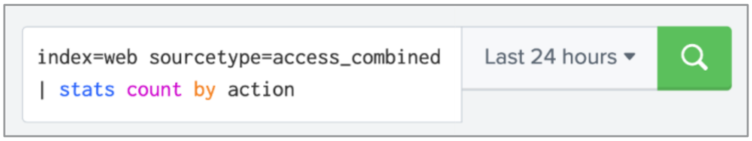
NOTE: Sometimes there can be delays in executing an action like saving in the UI or returning results of a search. If you are experiencing a delay, please allow the UI a few minutes to execute your action.

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 automatically begin the pipe on a new line.



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

Search auto-format disabled (default)



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

Search auto-format enabled

Task 2: Assume today is Friday and the current time is exactly 9:43:00 AM. Answer the following questions to test your knowledge of the `earliest` and `latest` time modifiers.

11. Provide the time modifiers that would satisfy the search scenarios. The first answer is provided to you.
    a. Return results from today that occurred from 9:13 AM to right now.
    `earliest=-30m latest=now()`
    b. Return results from yesterday only.

    c. Return results from 9:13 AM on Thursday to 9:13 today.

    d. Return results from Sunday at noon to the beginning of Wednesday.

    e. What would using **`earliest=-1h@h`** return?

    f. What would using **`earliest=@w7 latest=@d`** return?

Scenario: The facilities team wants to know how many employees from each department are badging into the building during each hour of the day.

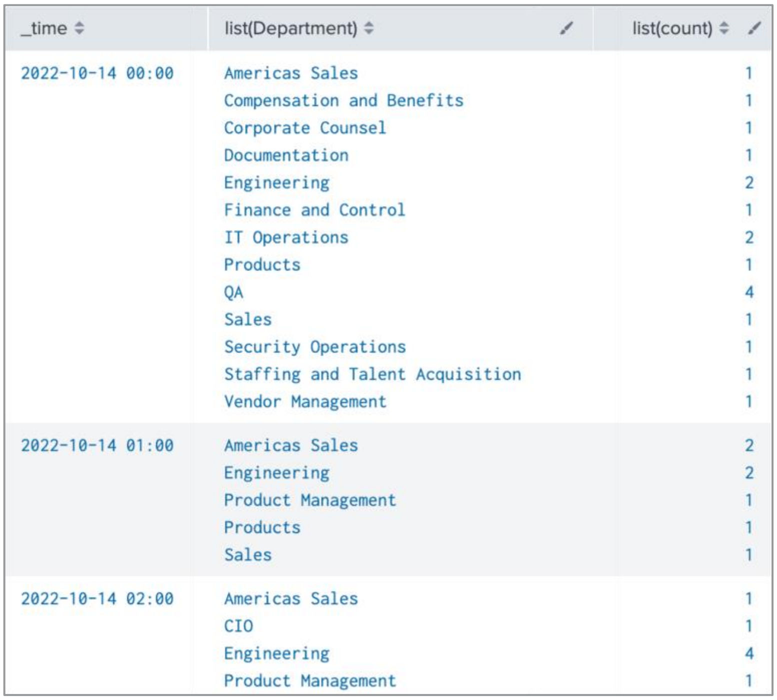
Task 3: Use the `bin` command to group results into 1-hour bins.

12. Navigate to the **Search & Reporting** app in the application bar.
13. Search badge reader data (**`index=security sourcetype=history_access`**) for all events that occurred today.

14. Use the **`bin`** command to group results into 1-hour bins. What happens to the timestamp for each event?

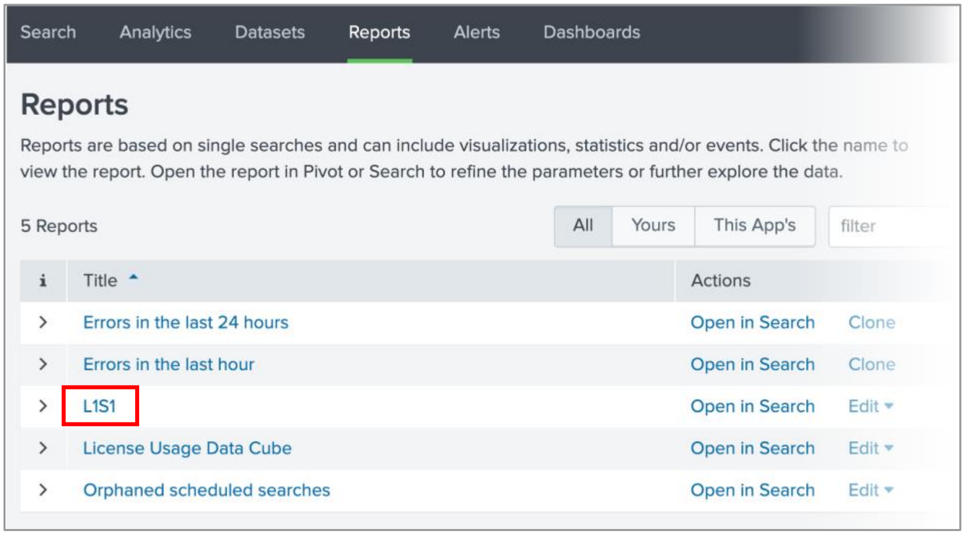
15. Pipe your results to the following stats commands. The first **`stats`** command counts all events by each unique combination of hour and **`Department`**. The second **`stats`** command lists all Departments and their count values for each hour.

```
| stats count by Department, _time
| stats list(Department), list(count) by _time
```



_time	list(Department)	list(count)
2022-10-14 00:00	Americas Sales Compensation and Benefits Corporate Counsel Documentation Engineering Finance and Control IT Operations Products QA Sales Security Operations Staffing and Talent Acquisition Vendor Management	1 1 1 1 2 1 2 1 4 1 1 1 1
2022-10-14 01:00	Americas Sales Engineering Product Management Products Sales	2 2 1 1 1
2022-10-14 02:00	Americas Sales CIO Engineering Product Management	1 1 4 1

16. 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.



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

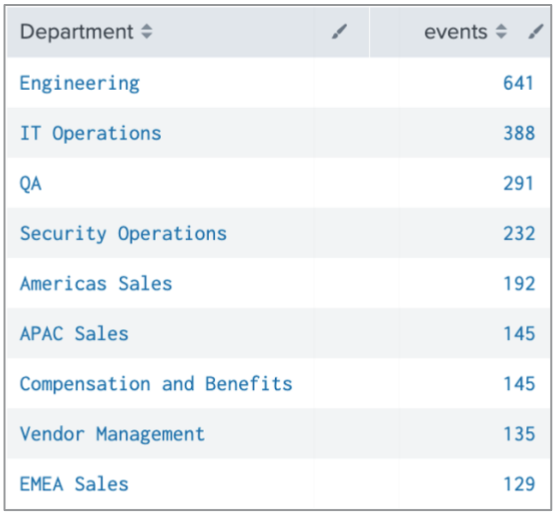
i	Title	Actions	
>	Errors in the last 24 hours	Open in Search	Clone
>	Errors in the last hour	Open in Search	Clone
>	L1S1	Open in Search	Edit
>	License Usage Data Cube	Open in Search	Edit
>	Orphaned scheduled searches	Open in Search	Edit

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

CHALLENGE: Facilities wants to know access events (badge swipes) by employees per department during the previous business week grouped into ranges of 100.

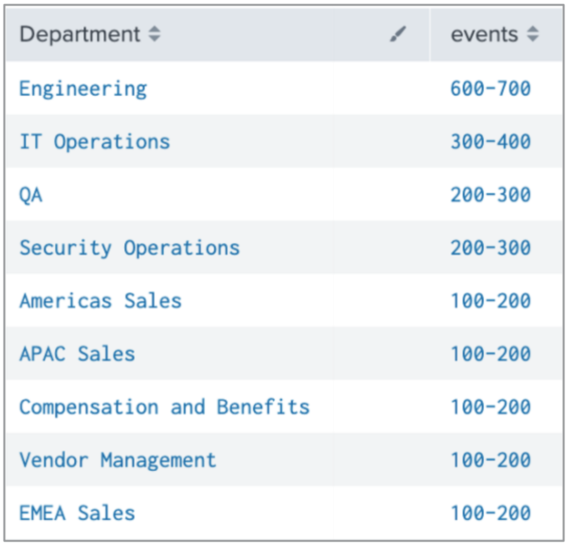
17. Re-initialize the search window by clicking **Search** in the application bar. This step should be done every time you save a report so that you do not accidentally overwrite a previous report.
18. This search finds, counts, and sorts all "Access" events from the badge reader and groups results by department. Run this search over the **Previous business week**.

```
index=security sourcetype=history_access Event_Description="Access"
| stats count as events by Department
| sort -events
```



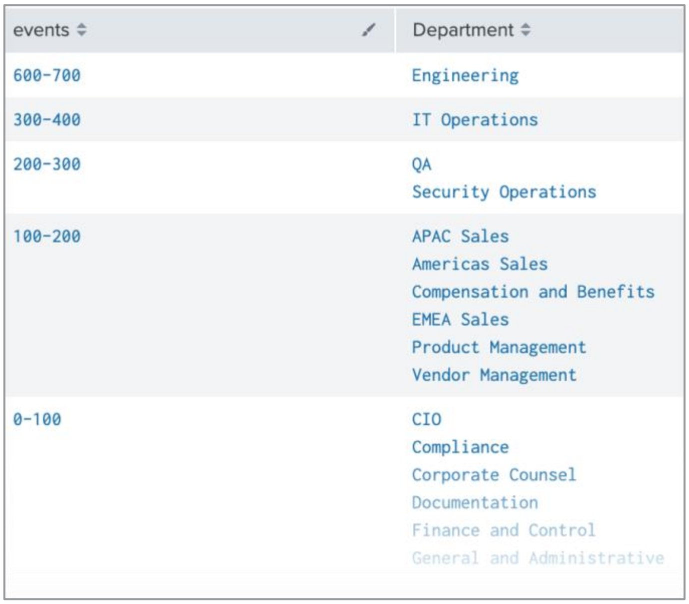
Department	events
Engineering	641
IT Operations	388
QA	291
Security Operations	232
Americas Sales	192
APAC Sales	145
Compensation and Benefits	145
Vendor Management	135
EMEA Sales	129

19. Fulfill the scenario request by grouping `events` values into ranges of 100.



Department	events
Engineering	600-700
IT Operations	300-400
QA	200-300
Security Operations	200-300
Americas Sales	100-200
APAC Sales	100-200
Compensation and Benefits	100-200
Vendor Management	100-200
EMEA Sales	100-200

20. Make the report easier to read by listing `Department` values for each `events` range with the `stats` command. (Note: The location of the `sort` command will determine if your sorting is conserved.)



events	Department
600-700	Engineering
300-400	IT Operations
200-300	QA Security Operations
100-200	APAC Sales Americas Sales Compensation and Benefits EMEA Sales Product Management Vendor Management
0-100	CIO Compliance Corporate Counsel Documentation Finance and Control General and Administrative

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

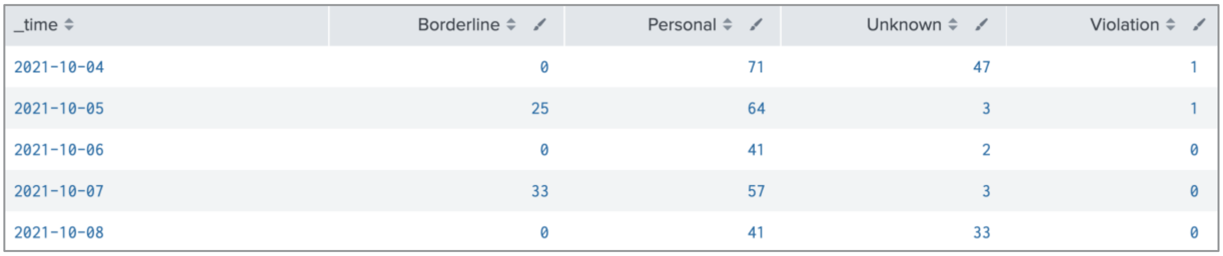
Lab Exercise 2 – Formatting Time & Using Time Commands

Scenario: The Network team would like to see the non-business activity from the web security appliance that was logged during the previous business week.

Task 1: Use the `timechart` command to group events into 1-day increments. Then, format your results with the `eval` command and create a visualization.

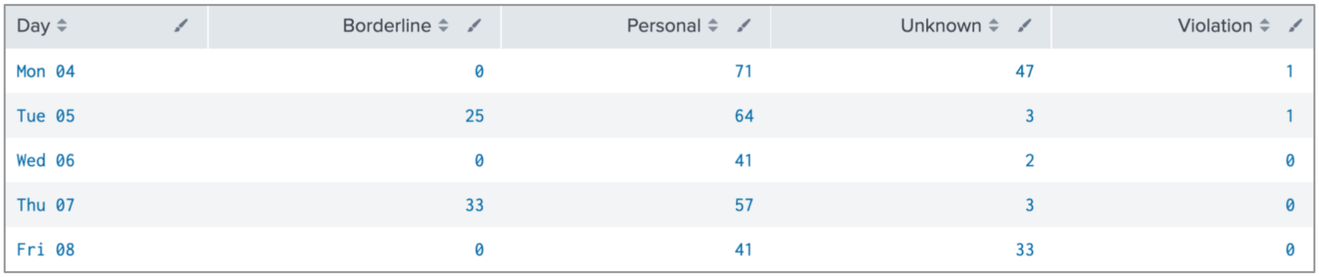
NOTE: For this scenario, “business week” is defined as Monday – Friday.

1. Search web security appliance data (**`index=network sourcetype=cisco_wsa_squid`**) for non-business activity, i.e., **`usage`** values other than **`Business`** (**`usage!=Business)`** during the previous business week.
2. Use the **`timechart`** command to count events by **`usage`** and group the results into 1-day segments.



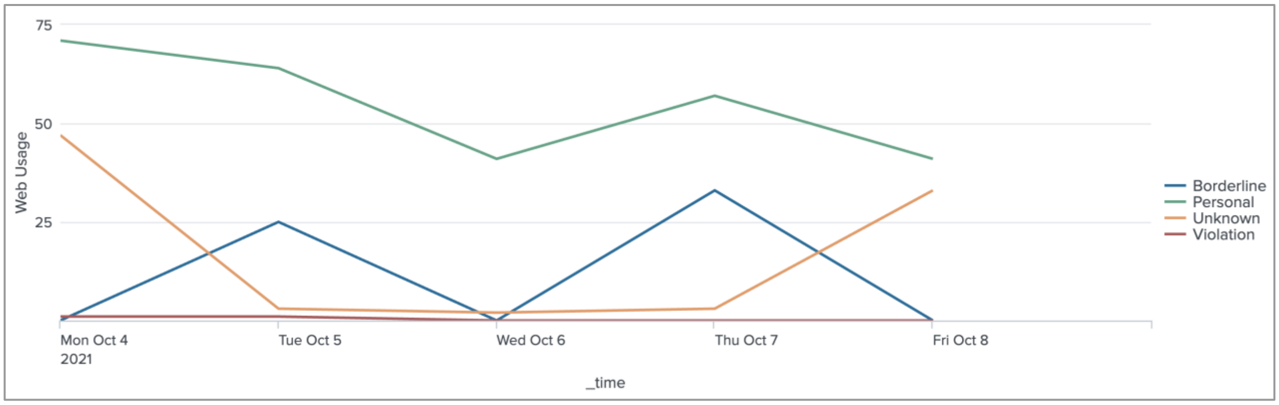
_time	Borderline	Personal	Unknown	Violation
2021-10-04	0	71	47	1
2021-10-05	25	64	3	1
2021-10-06	0	41	2	0
2021-10-07	33	57	3	0
2021-10-08	0	41	33	0

3. Rename **`_time`** as "Day" and use the **`eval`** command to format **`_time`** values as Day 00 where 00 is the numerical day of the month.



Day	Borderline	Personal	Unknown	Violation
Mon 04	0	71	47	1
Tue 05	25	64	3	1
Wed 06	0	41	2	0
Thu 07	33	57	3	0
Fri 08	0	41	33	0

4. Display results as a line chart and set the label for the Y-axis to “Web Usage.”



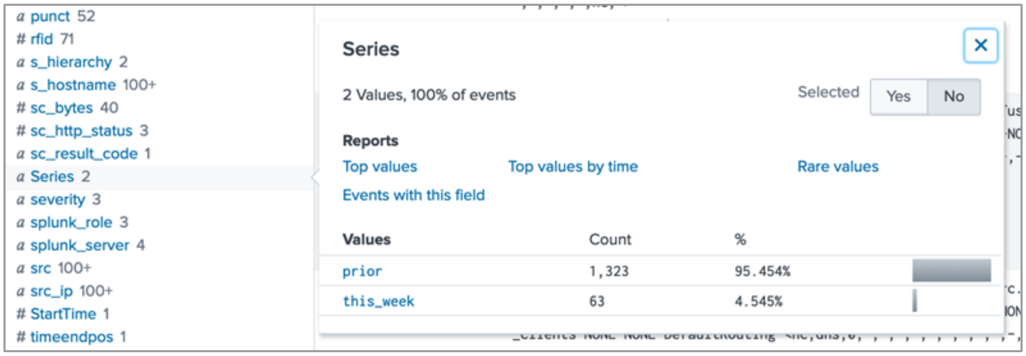
5. Save your search as a report with the name **L2S1**.

Scenario: The Network team would like to see the pattern of server errors over the last week compared to the daily average from 1 month ago through yesterday.

Task 2: Use the timechart command and time functions to fulfill the scenario request.

6. Search the web security appliance data (`index=network sourcetype=cisco_wsa_squid`) for server errors (`sc_http_status>=500`) from 1 month ago through the end of yesterday, i.e., today at 12AM.
7. Create a new field called "StartTime" and set the value to seven days ago from today, snapped to the beginning of the day.
8. In the **Interesting Fields** list, click on `StartTime`. You should only see one value.
9. Next, pipe results to the following `eval` command. This `eval` command creates a new field called "Series" and uses the `if` function to assign a value of "this_week" to events that occurred after the value of `StartTime` or "prior" if the event occurred before the value of `StartTime`.

```
| eval Series = if(_time>=StartTime,"this_week","prior")
```



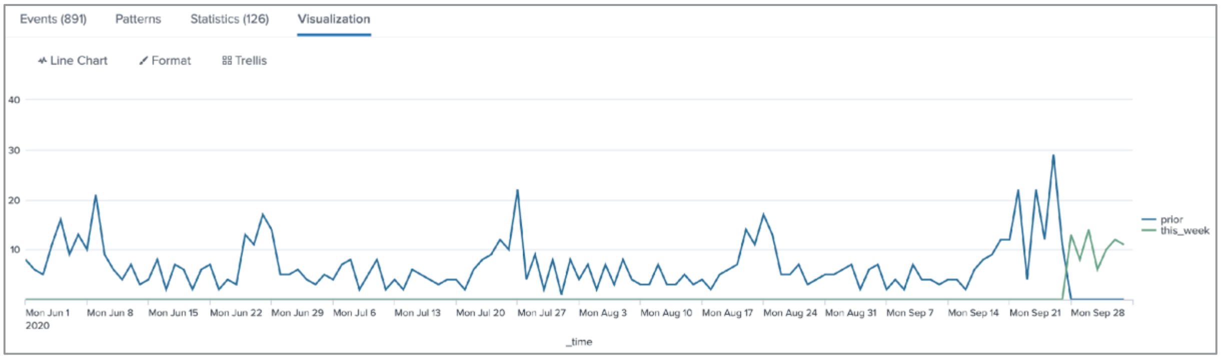
a punct 52
# rfid 71
a s_hierarchy 2
a s_hostname 100+
# sc_bytes 40
# sc_http_status 3
a sc_result_code 1
a Series 2
a severity 3
a splunk_role 3
a splunk_server 4
a src 100+
a src_ip 100+
StartTime 1
# timeendpos 1

Series

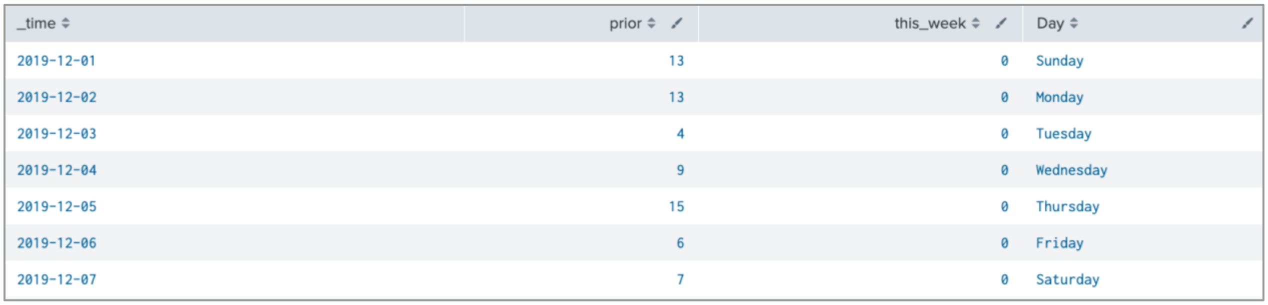
2 Values, 100% of events

Values	Count	%
prior	1,323	95.454%
this_week	63	4.545%

10. Use the `timechart` command to count events by `Series` into 1-day groupings. Then, visualize the results as a **Line Chart**.



11. Create a field called "Day" that formats **`_time`** values as the unabbreviated full name of the day, e.g., Monday, Tuesday, Wednesday, etc. When done, click the **Statistics** tab to verify that a **`Day`** column was added to your results table.



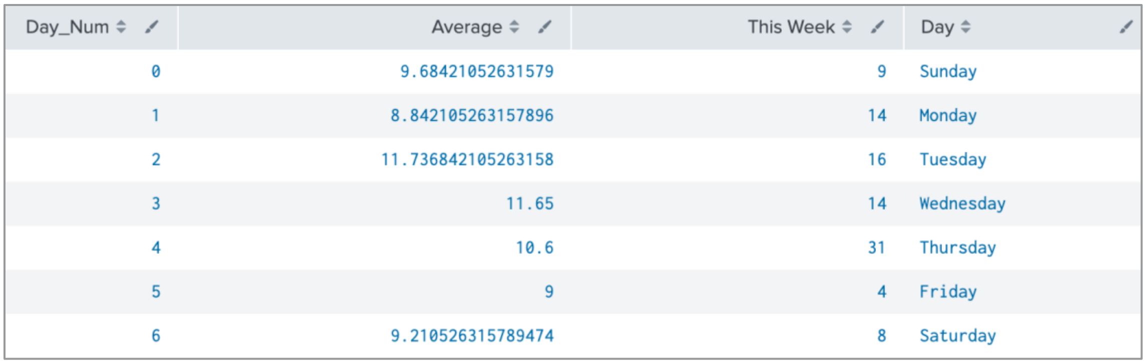
_time	prior	this_week	Day
2019-12-01	13	0	Sunday
2019-12-02	13	0	Monday
2019-12-03	4	0	Tuesday
2019-12-04	9	0	Wednesday
2019-12-05	15	0	Thursday
2019-12-06	6	0	Friday
2019-12-07	7	0	Saturday

12. Create another field called "Day_Num" that formats **`_time`** values as the ordinal day of the week, e.g., 0 for Sunday, 1 for Monday, etc.
13. Pipe results to the following **`stats`** command.

```
| stats avg(prior) as Average, sum(this_week) as "This Week", values(Day) as Day
  by Day_Num
```

This **`stats`** command groups the results of the following functions by **`Day_Num`** using a **`by`** clause.

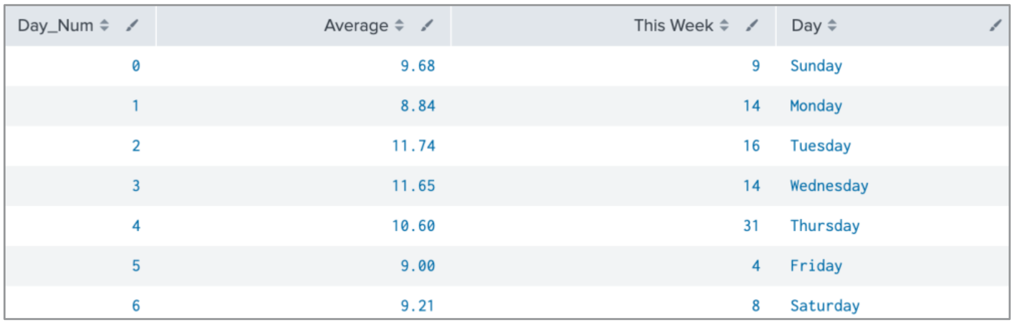
- The **`average`** function to calculate the average value of **`prior`**
- The **`sum`** function to calculate the sum of **`this_week`** values
- The **`values`** function to list the unique values of **`Day`**
- An **`as`** clause is included after each function so that the resulting fields are called "Average", "This Week", and "Day", respectively.



Day_Num	Average	This Week	Day
0	9.68421052631579	9	Sunday
1	8.842105263157896	14	Monday
2	11.736842105263158	16	Tuesday
3	11.65	14	Wednesday
4	10.6	31	Thursday
5	9	4	Friday
6	9.210526315789474	8	Saturday

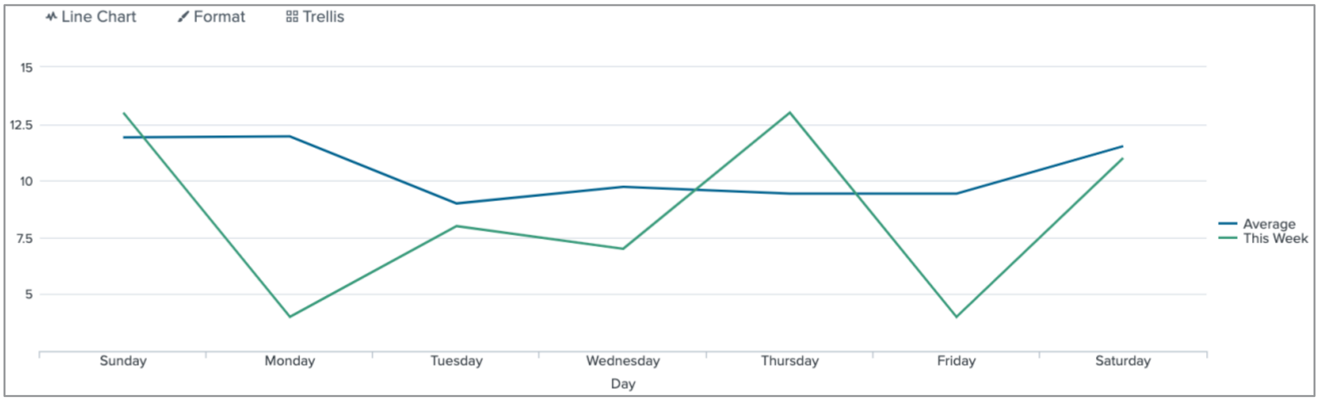
14. Round the values of **`Average`** to two decimal places by piping results to the following **`eval`** command:

```
| eval Average = round(Average,2)
```



Day_Num	Average	This Week	Day
0	9.68	9	Sunday
1	8.84	14	Monday
2	11.74	16	Tuesday
3	11.65	14	Wednesday
4	10.60	31	Thursday
5	9.00	4	Friday
6	9.21	8	Saturday

15. Finally, display the values of **`Day`, `Average`,** and **`This_Week`** in a table and then display results in a **Line Chart**. (Hint: Review the Common Commands & Functions table at the beginning of this document to find the command that allows you to specify which fields to return as a table.)

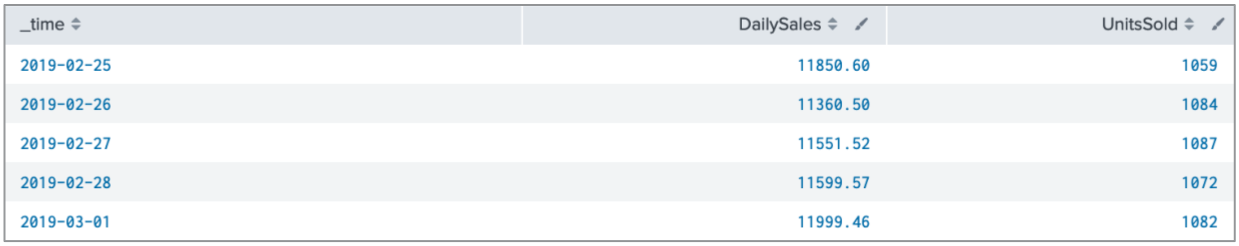


16. Save your search as a report with the name **L2S2.**

Scenario: Sales wants a detailed report of successful online sales from the previous business week with information about daily sales totals, number of units sold, and the average sale amount.

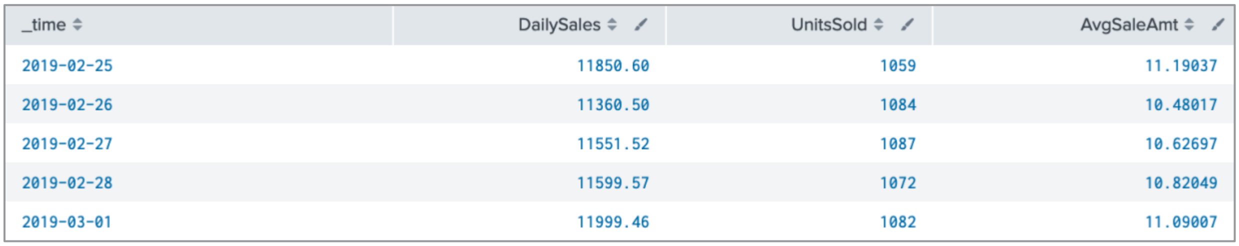
Task 3: Use the `timechart` command to calculate statistics grouped by time and use the `eval` command to format your results.

17. Search for successful purchase events from the online sales data that contain a value for **`productId`** (**`index=web sourcetype=access_combined status=200 productId=*`**) during the **Previous business week**. Use the **`date_hour`** field to limit the results of your basic search to those which occurred between 9AM and 5PM.
18. Use **`timechart`** to calculate the sum of **`price`** as "DailySales" and all count all events as "UnitsSold".



_time	DailySales	UnitsSold
2019-02-25	11850.60	1059
2019-02-26	11360.50	1084
2019-02-27	11551.52	1087
2019-02-28	11599.57	1072
2019-03-01	11999.46	1082

19. Use the **eval** command to create a new field called **AvgSaleAmt** which divides **DailySales** by **UnitsSold**.



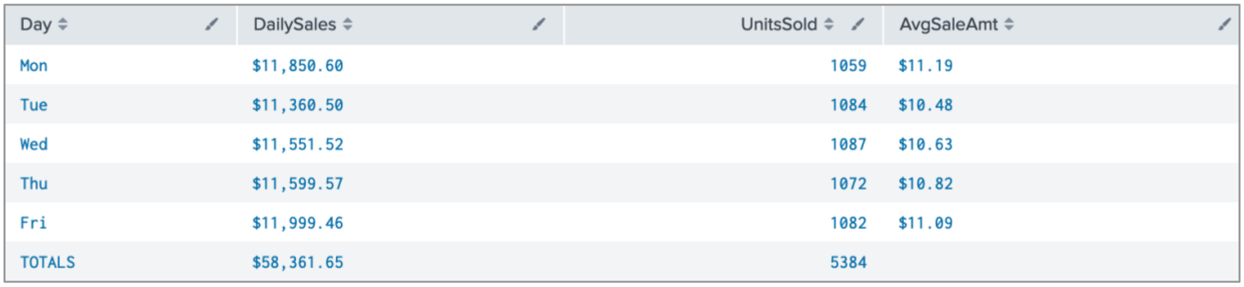
_time	DailySales	UnitsSold	AvgSaleAmt
2019-02-25	11850.60	1059	11.19037
2019-02-26	11360.50	1084	10.48017
2019-02-27	11551.52	1087	10.62697
2019-02-28	11599.57	1072	10.82049
2019-03-01	11999.46	1082	11.09007

20. Rename **_time** as **Day**. Then, use the **eval** command to format the values of **Day** so that they are the abbreviated weekday names like “Sun”, “Mon”, “Tue”, etc.

21. Finally, pipe the results of your search to the following **addtotals** and **foreach** commands.

```
| addtotals col=t row=f label=TOTALS labelfield=Day DailySales, UnitsSold
| foreach Daily*, Avg*
  [eval <<FIELD>> = "$".tostring(<<FIELD>>,"commas")]
```

The **addtotals** command calculates totals for the **DailySales** and **UnitsSold** columns. The **foreach** command applies formatting to the **DailySales** and **AvgSaleAmt** values.



Day	DailySales	UnitsSold	AvgSaleAmt
Mon	$11,850.60	1059	$11.19
Tue	$11,360.50	1084	$10.48
Wed	$11,551.52	1087	$10.63
Thu	$11,599.57	1072	$10.82
Fri	$11,999.46	1082	$11.09
TOTALS	$58,361.65	5384	

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