splunk>

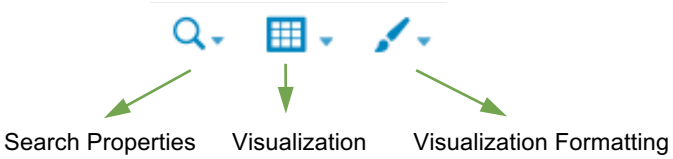
Advanced Dashboards & Visualizations Lab Exercises

Overview

Welcome to the Splunk Education lab environment. These exercises will guide you through the process of creating a set of dashboards and forms. You will create prototypes, refine them, and make changes requested by stakeholders. If you get stuck, consult the *lab solutions* document or the *example dashboard* in the course app.

Important: Save all knowledge objects in the Advanced Dashboards & Visualizations app with permissions set to private.

Visualization Editor Icons



- **Search Properties** icon: indicates the panel's document type –a magnifying glass for a panel based on a search, pivoting arrows for a pivot, or a sheet of paper for a search- or pivot-based report. Select this to modify the panel's search or report.
- **Visualization** icon: indicates the type of visualization displayed on the panel. Select this to specify the type of visualization displayed.
- **Visualization Formatting** icon: Select this to specify options for the selected visualization. The options available depend on the visualization selected.

Typographical Conventions

- **Blue** text highlighting indicates **add** text.
- **Red** text highlighting indicates **remove** text.
- **Grey** text provides **placement** information.

Lab Connection Info

You may want to write lab connection information (provided by your instructor) in the spaces below.

Class Number:	
Splunk Web URL:	
Splunk Web username:	
Splunk Web password:	

Source Types

There are two source types used in these exercises. They are referred to by the type of data they represent.

Type	Source type	Interesting Fields
Online transactions	access_combined	action, bytes, categoryId, clientip, itemId, JSESSIONID, price, productId, product_name, referer, referer_domain, sale_price, status, user, useragent
Retail sales	vendor_sales	AcctID, categoryId, product_name, productId, sale_price, Vendor, VendorCity, VendorCountry, VendorID, VendorStateProvince

Lab Exercise 1 – Get to Know Your Data

Description

Unlike traditional approaches of normalizing data into a common schema when it is collected, Splunk normalizes data at search-time to a common set of field names and tags that can be defined any time after data collection.

In this exercise, you will normalize your data to the Splunk Common Information Model (CIM) using the CIM add-on. This should be a refresher for event types, field aliases, and tags, covered in courses taken prior to this, such as Searching & Reporting, and Creating Knowledge Objects.

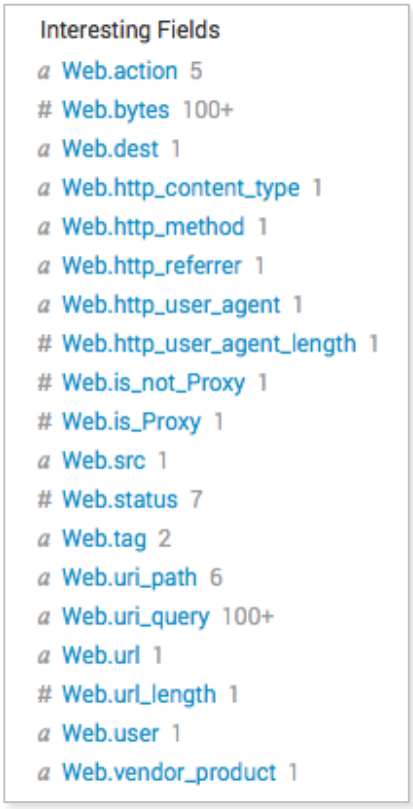
Important: Normalizing to a CIM data model requires setting permissions to *Global* for event types, field aliases, and tags. Doing that requires administrator-level access. Since this is a power user course, you will leave permissions for these knowledge objects set to *Private*.

Perform all searches and save all objects in the Advanced Dashboards & Visualizations app.

Scenario: The Buttercup Games Sales Team wants a dashboard that that displays web store server error information. It should have panels that show purchases compared to lost sales, server errors customers have received, and common errors by server. The sales team has already indexed their data in Splunk.

Your first step is to examine their data and normalize it.

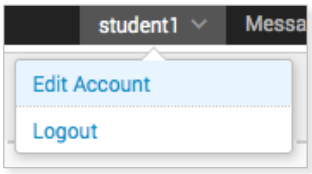
Example: At the end of this lab exercise, when validating data against the CIM datamodel, your Interesting Fields list should look similar to this:



Steps

Task 1: Change the account name and time zone.

1. Login to the class lab server.
   For example, http://class.server-name.splunk.com
2. Change your account settings:
   - Full name: <your last name>
   - Time zone: <your local time zone>
   - Default app: advdash



Task 2: Examine your data.

3. Search online transactions over the last 4 hours.
4. Examine the values of five fields required by your dashboard:
   - host
   - action
   - clientip
   - status
   - useragent

   Note: For a list of status codes and a description for each, search for: `| inputlookup http_status.csv`
5. Examine the Web data model in the CIM Reference Tables.
6. What tag is required for the parent object?

   _____

7. Examine the **Fields for Web** event objects table. Which fields in the data model match the fields needed for your dashboard?

Required Field	Field in Data Model
host	
action	
clientip	
status	
useragent	

Task 3: Create an event type and tag.

8. Search for all action types related to online transactions in the last 4 hours.
9. Save the search as an event type named: `access_combined` with a tag named: `web`

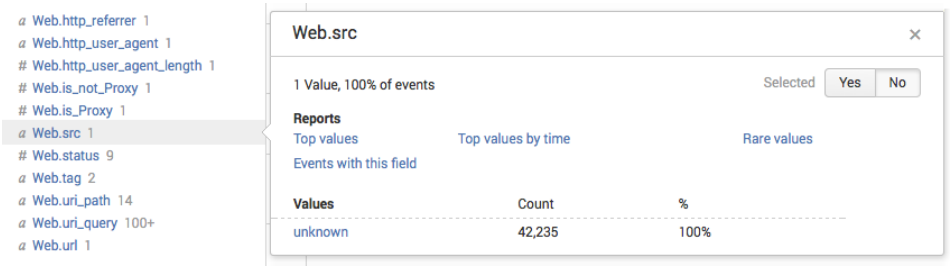
   Note: In a production environment, a Splunk administrator would later set the permissions of this event type to Global.

Task 4: Test your tag and event type.

10. Search using the event type.
    What sourcetype is returned? _____
11. Search using the tag.
    What sourcetype is returned? _____

12. Search for: `| datamodel Web Web search | fields Web*`
    Reminder: When using the datamodel command, all arguments are case-sensitive (including the "search" argument).
13. Notice the addition of a set of fields with the prefix Web**.** These are attributes from the CIM's Web data model. Some are populated based on the tag *web* in the event type you created; some are not. Instead they show: *unknown*

Example:



Field in Your Data	Matching Attribute	Data Model Field Populated?
host	Web.dest	
action	Web.action	
clientip	Web.src	
status	Web.status	
useragent	Web.http_user_agent	

Task 5: Create field aliases.

14. Create field aliases for the needed attributes that didn't populate.

Task 6: Validate your data against a CIM data model.

15. Search again for: `| datamodel Web Web search | fields Web*`
    Reminder: When using the datamodel command, all arguments are case-sensitive (including the "search" argument).
16. All five fields you require should now be populating the targeted Web data model attributes:

Field in Your Data	Matching Attribute	Data Model Field Populated?
host	Web.dest	
action	Web.action	
clientip	Web.src	
status	Web.status	
useragent	Web.http_user_agent	

Note: If your data model fields are not populating, delete the field alias and create it again. Be careful to avoid typos.

Lab Exercise 2 – Create a Prototype

Description

Having examined the data and normalized it, the next step is to create a dashboard prototype based on the use case and wireframe. The prototype will use basic searches and visualizations.

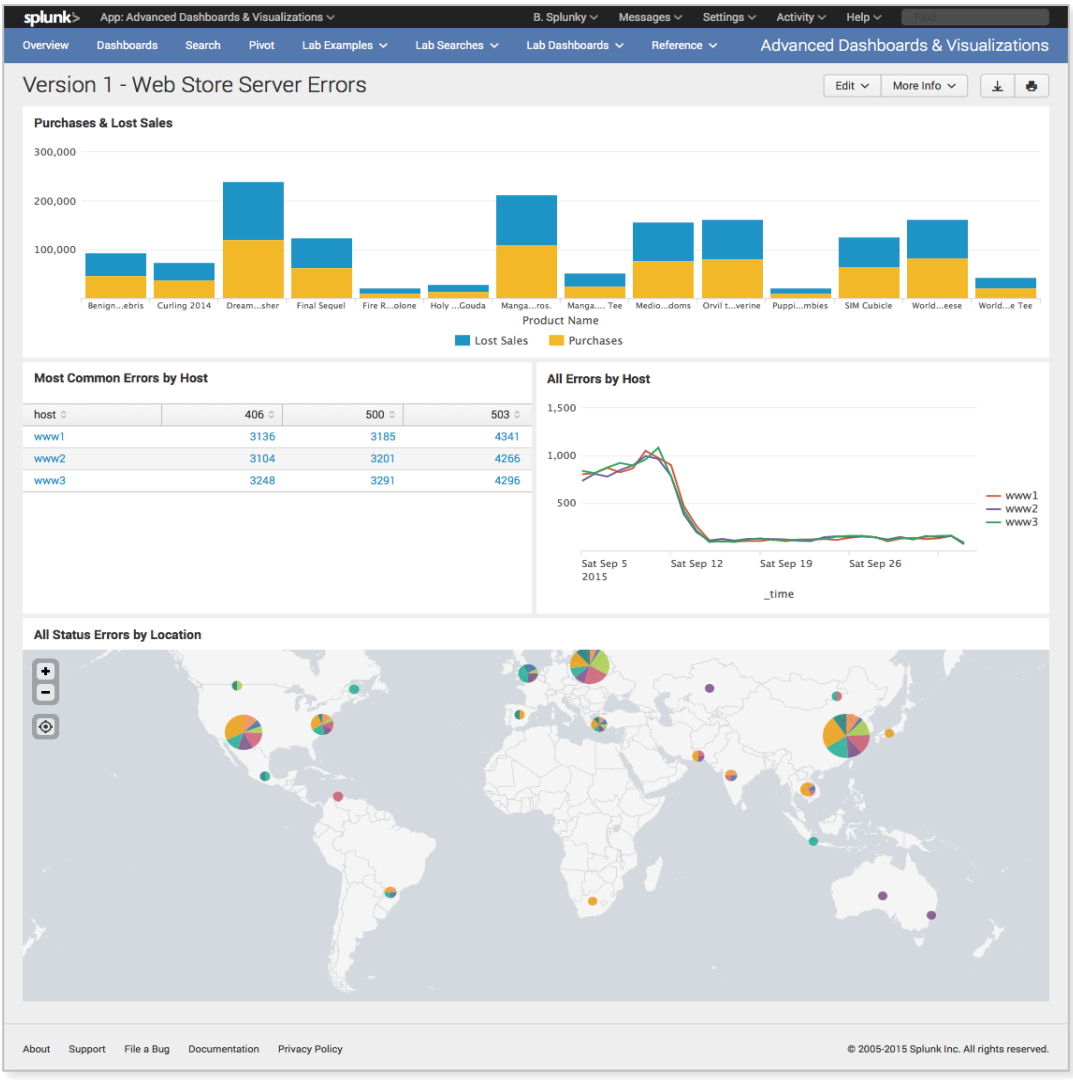
Important: Perform all searches and save all knowledge objects in the Advanced Dashboards & Visualizations app.

Scenario: The Buttercup Games Sales Team wants a dashboard that displays information about web store sales related to server health. It should have panels that display the following:

- Chart of purchases and lost sales
- Table of most common web server errors by host
- Chart of all server errors by host
- Map of web server errors by location

Working Example: Advanced Dashboards & Visualizations > Lab Examples > Lab 2 - Create a Prototype

Example:



Naming: Define naming conventions for your knowledge objects early in the development process. Doing so will avoid confusion later. This becomes especially important when creating multiple iterations of dashboards and reports.

In the following tasks, consult the table below when naming objects.

Scope	Definition	Convention
Working Group	Corresponds to the working group(s) of the user saving the search	bcg
Category	Corresponds to the area of concern	webstore
Object Type	Indicates the type of knowledge object	report, dash, panel
Description	Short, meaningful	server_errors
Example	bcg_webstore_report_server_errors	

Questions

Question 1: Examine these searches. Which of these searches would return a **single series**?

a) … | table Vendor

b) … | stats avg(price) by Vendor

c) … | stats count AS views

Question 2: Examine these searches. Which of these searches would return a **multi-series**?

a) … | chart sum(price) as hourlySales by _time | where hourlySales > 50

b) … | chart count over VendorCountry by product_name limit=5 useother=f

c) … | chart sum(price) as hourlySales by _time | where hourlySales < 200

d) … | chart count(user) as totalAccess | where totalAccess < 100

Question 3: Examine these searches. Which of these searches would return a **time series**?

a) … | timechart per_hour(price) by product_name usenull=f useother=f

b) … | chart count(user) as totalAccess by _time span=1h | where totalAccess > 100

c) … | chart count(user) as totalAccess by _time span=1h | where totalAccess < 100

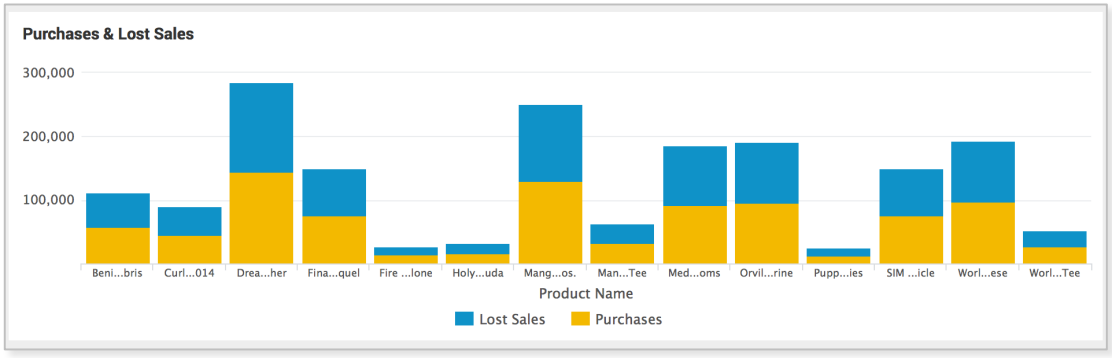
Steps

Task 1: Create a dashboard with a panel that shows online purchases and lost sales for orders over $1000.

1. Search online transactions for purchases or lost sales events over the last 30 days.
   Hint: action=remove, action=purchase
2. Calculate the total value of sales as totalSales by action and product_name.
3. Filter for total sales greater than $1000.
4. Ensure product_name is on the x-axis, action identifies the series, and totalSales is on the y-axis.
   Hint: xyseries
5. Rename product_name as Product Name, remove as Lost Sales, purchase as Purchases, and display the results as a column chart.

6. Save the search as a report and add it to a new dashboard:
   - Report Title: `bcg_webstore_report_purchases_vs_lost_sales`
   - Dashboard Title: `Version 1 - Web Store Server Errors`
   - Dashboard ID: `bcg_webstore_dash_v1_server_errors`
   - Dashboard Permissions: Private
   - Panel Title: `Purchases & Lost Sales`
   - Panel Powered By: Report
7. Open the dashboard in edit mode.
8. Format the panel visualization to display a stacked column chart with the legend on the bottom.
9. Exit dashboard edit mode.

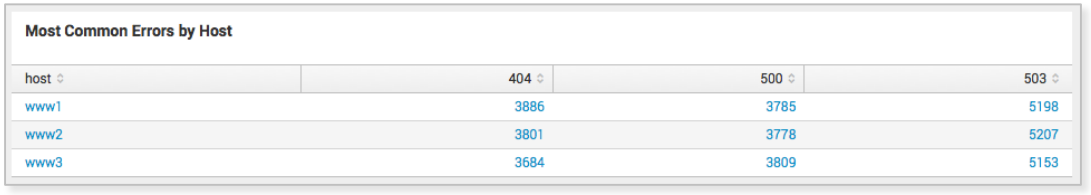
Example:



Task 2: Add a table of most common web server errors by host.

10. Search online transactions for HTTP status errors over the last 30 days.
    Hint: status!=200
11. Count errors by host and status.
12. Limit results to only the top three server errors.
13. Remove the grouping called OTHER.
14. Save the search as a report and add it to your Version 1 - Web Store Server Errors dashboard:
    - Report Title: `bcg_webstore_report_common_errors_by_host`
    - Dashboard Title: Version 1 - Web Store Server Errors
    - Panel Title: `Most Common Errors by Host`
    - Panel Powered By: Report
    - Panel Content: Statistics Table

Example:

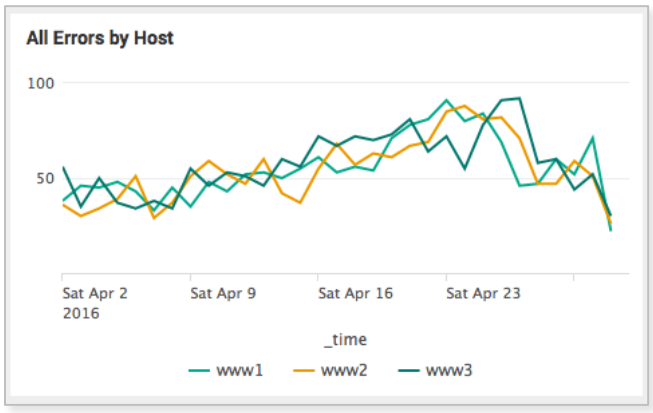


Task 3: Add a time series chart of all web server errors.

15. Search online transactions for HTTP status errors over the last 30 days.
16. Count action values by host.

17. Save the search as a report and add it to your Version 1 - Web Store Server Errors dashboard:
    - Report Title: `bcg_webstore_report_all_errors_by_host`
    - Dashboard Title: Version 1 - Web Store Server Errors
    - Panel Title: `All Errors by Host`
    - Panel Powered By: Report
    - Legend Position: bottom
18. Format the panel visualization to display a line chart with the legend on the bottom.
19. Position panel on the right of the Most Common Errors by Host panel.
20. Exit dashboard edit mode.

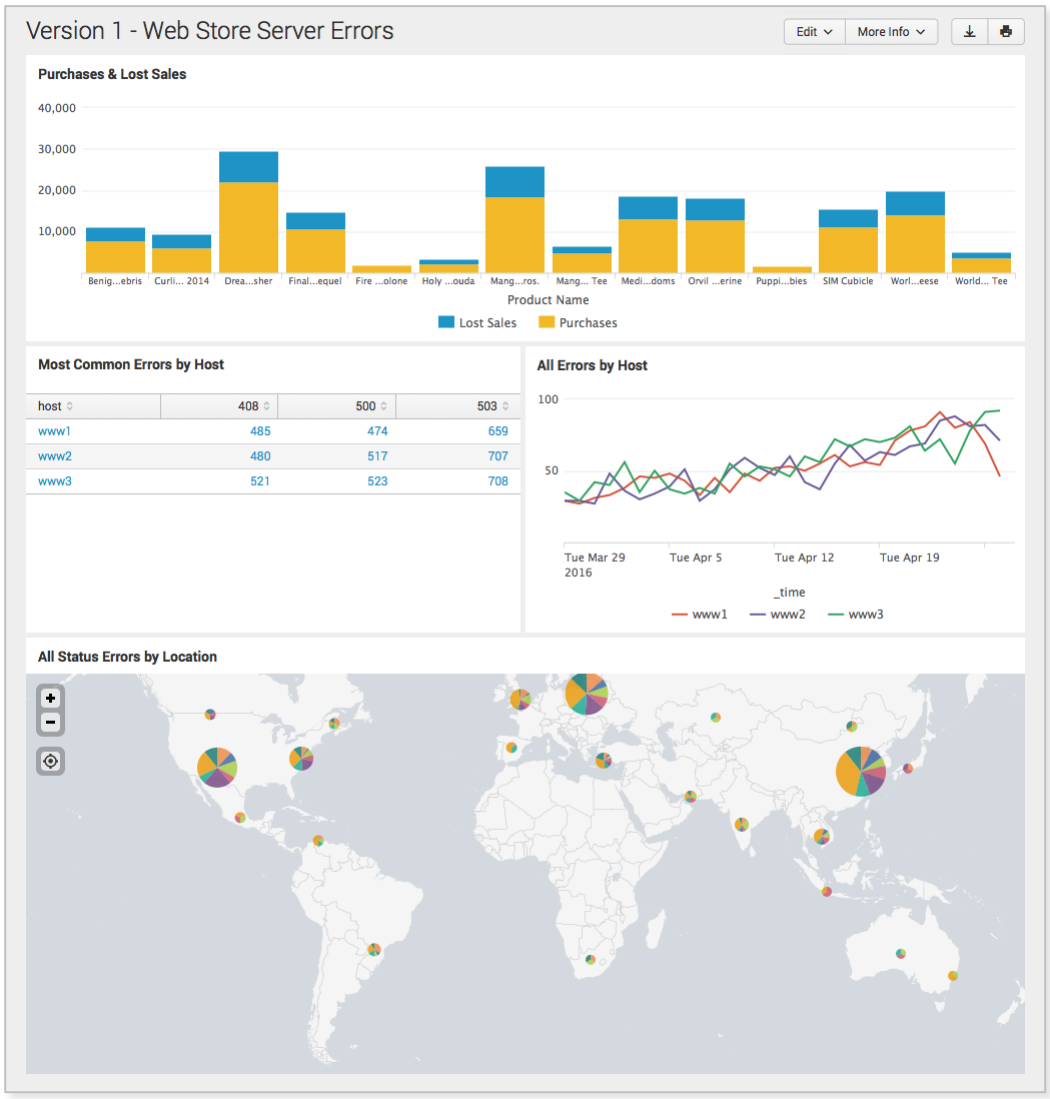
Example:



Task 4: Add a map of web store server error locations.

21. Search online transactions for all client ip's and HTTP error codes for the last 30 days.
    Hint: `status!=200`
22. Eliminate duplicate client ip's and hosts.
23. Extract location information from IP addresses and add the prefix cip_ to the clientip field.
    Hint: `iplocation`
24. Generate statistics based on ip location and count events by status.
25. Save the search as a report and add it to an existing dashboard:
    - Report Title: `bcg_webstore_report_status_errors_by_location`
    - Dashboard Title: Version 1 - Web Store Server Errors
    - Panel Title: `All Status Errors by Location`
    - Panel Powered By: Report
    - Panel Content: Cluster Map
26. View the dashboard and make sure it matches the example.
    Example on next page.

Example:



Refine

Once you have a prototype, make refinements such as:

- **Write better searches**
  Narrow down your search as much as possible from the start and limit the data that has to be retrieved from disk to an absolute minimum. Add formatting to the search results.
- **Schedule reports**
  You can schedule reports to conserve resources, use report acceleration for faster panel loading, trigger alerts, email, run a script, etc.
- **Accelerate reports**
  Use automatically-created summaries to speed up completion times for certain kinds of reports.

Task 5: Accelerate and schedule your reports.

27. Navigate to: **Settings** > **Searches, reports, and alerts**
28. Make sure the App context is set to: Advanced Dashboards and Visualizations
29. In the Owner dropdown, select your name.
30. Open each of your reports:
    - bcg_webstore_report_all_errors_by_host
    - bcg_webstore_report_common_errors_by_host
    - bcg_webstore_report_purchases_vs_lost_sales
    - bcg_webstore_report_status_errors_by_location
31. Select **Accelerate this search**.
32. Set summary range to: 1 month
33. Set schedule for:
    - Schedule type: Basic
    - Run every: day at midnight
    - Schedule Window: `15`

    Note: If you receive an error while trying to accelerate one of the reports, de-select the acceleration option, but still schedule the search.

Questions

Question 1: Why is acceleration not possible for the *bcg_webstore_report_status_errors_by_location* report?

_____

Question 2: What, if any, change(s) would be required to accelerate it?

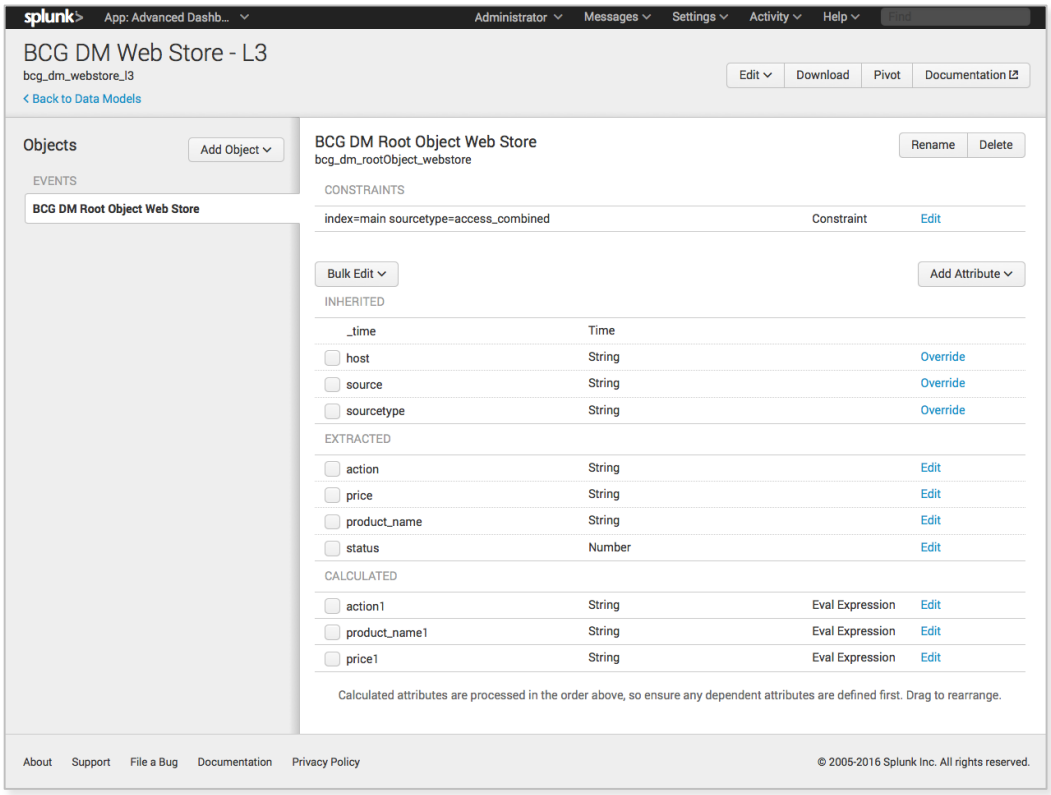
_____

Challenge Lab (optional)

Create a Data Model

This task walks you through the process of creating a data model that can be accelerated – a data model similar to one used in later labs.

Example:



Task 1: Create a data model and add a root event.
This data model will be used for the web store dashboard. Its root event will use the access_combined sourcetype as a constraint.

Note: Since this is a power-user classroom environment, you will create this data model in the Search & Reporting app.

1. Create a new data model in the Search & Reporting app.
   - Title: `BCG DM Web Store`
   - ID: `bcg_dm_webstore`
   - App: Search & Reporting
2. Add a Root Event object.
   - Object Name: `BCG DM Root Object Web Store`
   - Object ID: `bcg_dm_rootObject_webstore`
   - Constraints: `index=main sourcetype=access_combined`
3. Preview events to verify the attribute is working properly, then save it.

Task 2: Add attributes.

4. Add four auto-extracted fields:
   - action
   - price
   - product_name
   - status
5. Add an eval expression for NULL values to the action field.
   Note: Searches against this data model will require all fields identified in the data cube to have a value.
   - Eval Expression: `if(isnull(action),"NULL",action)`
   - Field Name: `action1`
   - Display Name: `action1`
6. Preview events to verify the attribute is working properly; then save it.
7. Create two more eval expression attributes for NULL values, similar to action1, for the price and product_name fields. Use the same naming convention as the action field.

Task 3: Test your data model.

8. Navigate to the Search & Reporting app.
9. Using the datamodel command, search the bcg_dm_webstore data model's bcg_dm_rootObject_webstore object over the last 24 hours.
   Tip: Since the datamodel command is a generating command, it should be preceded with a pipe.
10. Use the fields command to display all fields in the data model.
    Hint: the datamodel fields are prefixed with: *bcg_dm_rootObject*.
11. Examine the field values in the sidebar of the action*, price*, and product_name* fields.

Questions

Question 1: What is the difference between the values of bcg_dm_rootObject_webstore.action and bcg_dm_rootObject_webstore.action1?

_____

Question 2: What is the difference between the values of bcg_dm_rootObject_webstore.price and bcg_dm_rootObject_webstore.price1?

_____

Question 3: What is the difference between the values of bcg_dm_rootObject_webstore.product_name and bcg_dm_rootObject_webstore.product_name1?

_____

Lab Exercise 3 – Improve Performance

Description

In this exercise you will improve dashboard performance by using a global search and post-process searches. After creating prebuilt panels, a dashboard, adding and customizing new visualizations, you will accelerate the global search.

Important: When editing the simple XML, type the text manually or copy it fro0m the dashboard editor not this document. Character formatting and artifacts created by the PDF generation process can cause errors in the XML.

Perform all searches and save all knowledge objects in the Advanced Dashboards & Visualizations app.

Scenario: The stakeholders have approved the prototype dashboard, with some changes:

- Make the dashboard load faster.
- Since only a single indexer is being used, reduce the search load from this dashboard.
- Use the company's brand colors for the chart of purchases and lost sales.
- Add two new panels to show the dollar amount of purchases and lost sales in the past month.
- Change the cluster map to a choropleth map.

Your next step is to improve the performance of the dashboard, and incorporate the other changes requested by the stakeholders.

Working Example: Advanced Dashboards & Visualizations > Lab Examples > Lab 3 - Improve Performance

Example:

Steps

Task 1: Create prebuilt panels.

1. Select **Lab Dashboards** > **Version 1 - Web Store Server Errors**.
2. Open the dashboard in edit mode
3. Select **Convert to Prebuilt Panel** from the Most Common Errors by Host options menu. 
   - ID: `bcg_webstore_panel_<panel_title>`
     For example: bcg_webstore_panel_most_common_errors
   - Permissions: Private
4. Click **Save**.
5. Click **OK** to confirm.
6. Repeat the above steps to convert two other panels:
   - All Errors by Host
   - All Status Errors by Location
7. Exit dashboard edit mode.

Task 2: Create a new dashboard.

8. In the Lab Searches menu, click the **search icon**  for Lab 3 - Search 1.
   Note: This search returns data from a data model that is not accelerated. Later, you will revise it. At that point, you will see a performance difference when opening or refreshing the dashboard that uses an accelerated data model.
9. Save the search as a panel on a new dashboard:
   - Dashboard: New
   - Dashboard Title: `Version 2 - Web Store Server Errors`
   - Dashboard ID: `bcg_webstore_dash_v2_server_errors`
   - Panel Title: `Base Search`
   - Panel Powered By: Inline Search
10. Save, then view the dashboard.

Task 3: Add a base search.

11. Open the Splunk XML Editor.
12. Locate the Base Search panel's search query.
13. Delete the opening and closing <row>, <panel>, <table> and <title> tags.

```
<dashboard>
  <label>Version 2 - Web Store Server Errors</label>
  <row>
    <panel>
       <table>
         <title>Base Search</title>
         <search>
         ...
         </search>
       </table>
    </panel>
  </row>
</dashboard>
```

14. Rename the opening <search> tag to: `<search id="baseSearch">`

```
<dashboard>
 <label>Version 2 - Web Store Server Errors</label>
    <search id="baseSearch">
...
```

15. Save the XML changes.
    Note: The dashboard will not display anything yet. This is normal and expected.

Task 4: Add a chart that uses a post-process search.

16. In the Lab Searches menu, click the **search icon**  for Lab 3 - Search 2.

    Note: This search is dependent on the base search. It will not display results at this point. This is normal and expected.

17. Save the search as a panel to an existing dashboard:
    - Dashboard: Version 2 - Web Store Server Errors
    - Panel Title: `Purchases & Lost Sales`
    - Panel Powered By: Inline Search
18. Save, then view the dashboard.
19. Open the Splunk XML Editor.
20. Locate the XML for the Purchases & Lost Sales panel.
21. Revise the opening <search> tag to be: `<search base="baseSearch">`

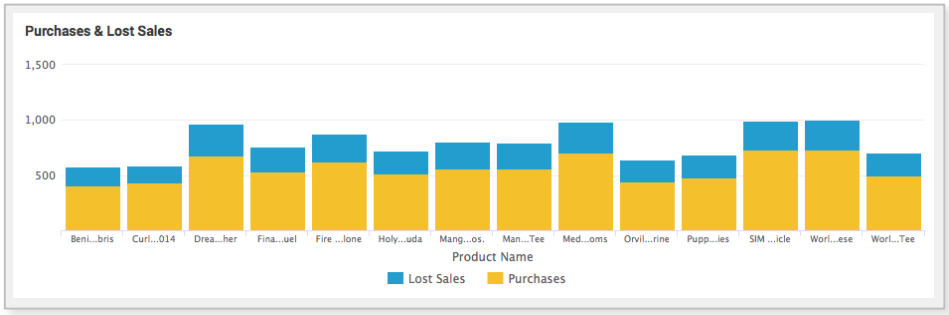
```
...
<table>
 <title>Purchases &amp;amp; Lost Sales</title>
 <search base="baseSearch">
 <query>
...
```

22. Remove the Purchases & Lost Sales panel's earliest and latest tags.

```
...
</query>
  <earliest>-30d@d</earliest>
  <latest>now</latest>
 </search>
...
```

23. Save the XML changes.
24. Format the panel to display a column chart in stacked mode with the legend on the bottom.
25. Exit dashboard edit mode.

Example:



Task 5: Add a single value panel that uses a post-process search.

26. In the Lab Searches menu, click the **search icon** for Lab 3 - Search 3.

    Note: The search will not display anything yet. This is normal and expected.

27. Save the search as a dashboard panel and add it to an existing dashboard:
    - Dashboard: Version 2 - Web Store Server Errors
    - Panel Title: `Purchases`
    - Panel Powered By: Inline Search

28. Save, then view the dashboard.

    Note: The Purchases panel will not display anything yet. This is normal and expected.

29. Open the Splunk XML Editor.

30. Locate the XML for the Purchases panel.

31. Revise the opening <search> tag to: `<search base="baseSearch">`

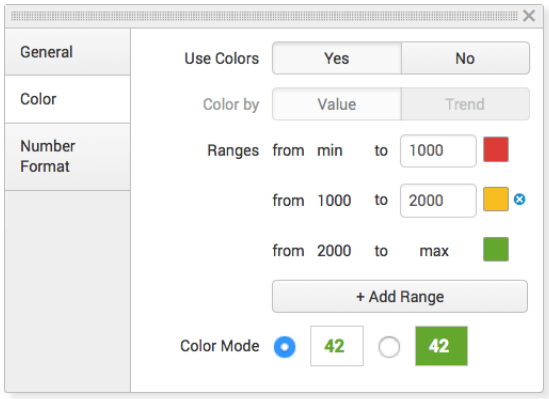
```
...
<panel>
 <table>
  <title>Purchases</title>
   <search base="baseSearch">
    <query>search product_name!=NULL action=purchase | stats sum(price)
    as Purchases</query>
...
```

32. Remove the Purchases panel's earliest and latest tags.

```
...
 <query>search product_name!=NULL action=purchase | stats sum(price)
  as Purchases</query>
   <earliest>-30d@d</earliest>
   <latest>now</latest>
 </search>
...
```

33. Save the XML changes.

34. Change the Purchases panel's visualization to single value with the following format:
    - Caption: Last 30 Days
    - Use Colors: Yes
    - Color Mode: No background
    - Color Ranges:
      - from min - 1000, red (d93f3c)
      - from 1000 - 2000, yellow (f7bc38)
      - from 2000 - max, green (65a637)
    - Number Format:
      - Unit: $
      - Unit Position: Before



Task 6: Add a second single value panel that uses a post-process search.

35. In the Lab Searches menu, click the **search icon** for Lab 3 - Search 4.

    Note: The search will not display anything yet. This is normal and expected.

36. Save the search as a dashboard panel and add it to an existing dashboard:
    - Dashboard: Version 2 - Web Store Server Errors
    - Panel Title: `Lost Sales`
    - Panel Powered By: Inline Search
37. Save, then view the dashboard.

    Note: The Lost Sales panel will not display anything yet. This is normal and expected.

38. Open the Splunk XML Editor.
39. Locate the XML for the Lost Sales panel.
40. Revise the opening <search> tag to: `<search base="baseSearch">`

```
...
  <title>Lost Sales</title>
  <search base="baseSearch">
   <query>search product_name!=NULL action=remove | stats sum(price) as
   LostSales</query>
...
```

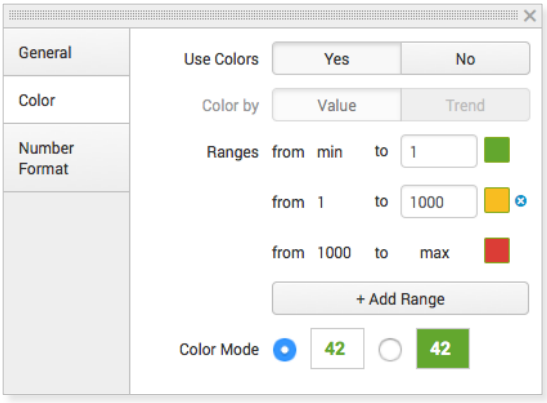
41. Remove the Lost Sales panel's earliest and latest tags.

```
...
  </query>
    <earliest>-30d@d</earliest>
    <latest>now</latest>
 </search>
...
```

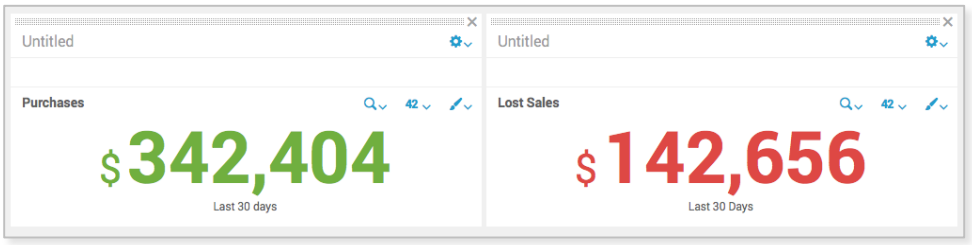
42. Delete the closing row tag and opening row tags between the single value panels.

```
...
 </single>
     </panel>
   </row>
   <row>
     <panel>
 <table>
...
```

43. Save the XML changes.
44. Change the Lost Sales panel's visualization to single value with the following format:
    - Caption: Last 30 Days
    - Use Colors: **Yes**
    - Color Ranges:
      - from min - 1, green (#65a637)
      - from 1 - 1000, yellow (#f7bc38)
      - from 1000 - max, red (#d93f3c)
    - Number Format:
      - Unit: $
      - Unit Position Before



Example:



Task 7: Add prebuilt panels.

45. Click **+Add Panel** > **Add Prebuilt Panel**.
46. Click **bcg_webstore_panel_most_common_errors by_host**.
47. Click **Add to Dashboard**.
48. Repeat the above steps for two other prebuilt panels:
    - bcg_webstore_panel_all_errors_by_host
    - bcg_webstore_panel_all_status_errors_by_location
49. Close the Add Panel sidebar.
50. Position the All Errors by Host panel to the right of Most Common Errors by Host.

Task 8: Convert prebuilt panels to inline panels driven by inline searches.

51. On the Most Common Errors by Host panel, click the **Options** icon. 
52. Select **Convert to Inline Panel**.
53. Click **Convert**.
54. Click the **Search Report** icon. 
55. Select the panel's report name, and click **Clone to an Inline Search**.
56. Click **Clone to Inline Search**.
57. Repeat the above steps for the remaining prebuilt panels:
    - All Errors by Host
    - All Status Errors by Location

Task 9: Add two post-process searches.

58. Open the Splunk XML Editor.
59. Locate the XML for the Most Common Errors by Host panel.
60. Revise the opening <search> tag to be: `<search base="baseSearch">`
61. At the beginning of the query, add the word: `search`

```
...
 <table>
  <title>Most Common Errors by Host</title>
   <search base="baseSearch">
    <query>search sourcetype=access_combined status!=200
      | chart count by host, status limit=3 useother=f</query>
...
```

62. Remove the earliest and latest tags.

```
...
</query>
   <earliest>-30d@d</earliest>
   <latest>now</latest>
</search>
...
```

63. Locate the XML for the All Errors by Host panel, and repeat the above steps.
64. Save the XML changes and make sure all panels populate with data.

Task 10: Customize chart colors.

65. Open the Splunk XML Editor.
66. Locate the option tags for the Purchases & Lost Sales panel.
67. Add a **charting.fieldColors** option to set the Purchases to orange (0xEFC94C) and the Lost Sales field to red (0xED553B).

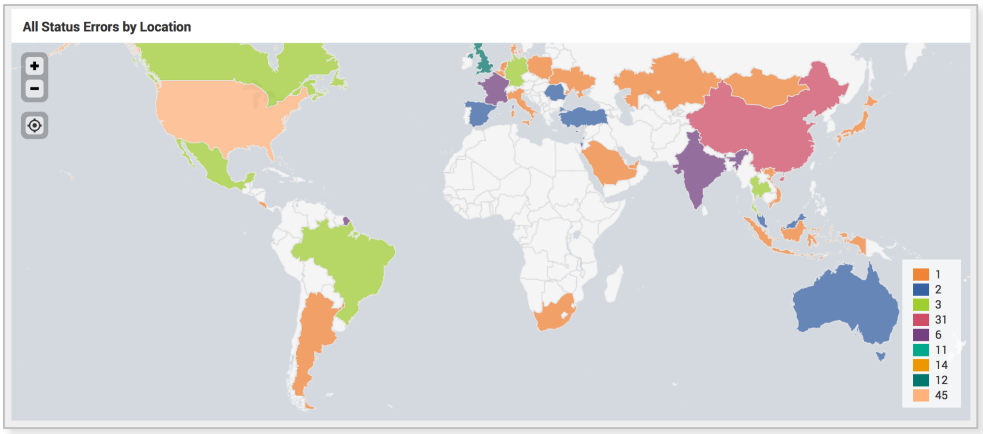
```
...
<option name="charting.legend.placement">bottom</option>
<option name="charting.fieldColors">{"Purchases":0xEFC94C,"Lost
Sales":0xED553B}</option>
</chart>
...
```

68. Save the XML changes and make sure the Purchases & Lost Sales panel's colors have changed to orange and red.

Task 11: Change the cluster map to display a choropleth.

69. Edit the search string for the All Status Errors by Location panel:
    - Delete the geostats command and its arguments.
    - Use a stats command to count by cip_Country.
    - Using the geom command, specify the built-in geo_countries lookup, and map colors to the cip_Country field. Hint: `featureIdField`.
70. Save the search changes.
    Note: The map will not display yet. This is expected.
71. Change the All Status Errors by Location panel's visualization to a **Choropleth Map**.
72. Change the map's color mode to categorical.

Example:



Task 12: Schedule the choropleth map search.

73. Convert the panel's search to a report.
    - Report Title: bcg_webstore_report_choro_status_errors_by_location
74. Exit dashboard edit mode.
    Note: Notice the map panel title has changed to the report name.
75. Open the dashboard in edit mode.
76. Change the map panel's title to: `All Status Errors by Location`
77. Exit dashboard edit mode.
78. Navigate to: **Settings** > **Searches, reports, and alerts**.
79. Set the App context is set to: Advanced Dashboards and Visualizations
80. In the Owner dropdown, select your name.
81. Open bcg_webstore_report_choro_status_errors_by_location.
82. Select **Schedule this search**.
83. Set schedule for:
    - Schedule type: Basic
    - Run every: day at midnight
    - Schedule Window: `15`
84. Save changes.

Task 13: Accelerate the global search.

85. Open the search jobs page.
86. Make sure the App context is set to: Advanced Dashboards & Visualizations
87. Make sure the Owner is set to your name.
88. Select all searches, and click **Delete**.
    Tip: Use link at bottom of window.
89. Click **OK** to confirm.
90. Open the **Version 2 - Web Store Server Errors** dashboard and wait for all the panels to populate.
91. Return to the search jobs page and make a note of the run time for the base search.

    _____

92. Return to the **Version 2 - Web Store Server Errors** dashboard.
93. Open the Splunk XML editor.

94. Update the global search's query to use the tstats argument summariesonly=t and the bcg_dm_webstore_xl accelerated data model.

    Note: The summariesonly argument only applies when using tstats against an accelerated data model. When set to true, 'tstats' only generates results from the TSIDX data that has been automatically generated by the acceleration.

```
...
 <label>Version 2 - Web Store Server Errors</label>
 <search id="baseSearch">
 <query>| tstats summariesonly=t count from datamodel="bcg_dm_webstore_xl"
 by _time, host, sourcetype,
...
```

95. Save the XML changes.
96. Wait for all the panels to populate.
97. Return to the search jobs page and make a note of the run time for the base search.

---

98. Compare the two sets of search times. Consider percentage of difference in the times.

Lab Exercise 4 – Add Interactivity

Description

In this exercise you will create several forms, some cascading inputs, a dynamic drilldown, a pan & zoom, and a visualization event handler.

Important: Perform all searches and save all knowledge objects in the Advanced Dashboards & Visualizations app.

Scenario: The Buttercup Games Sales Team is impressed with the new web store server errors dashboard. Now, they would like to add a form to their app.

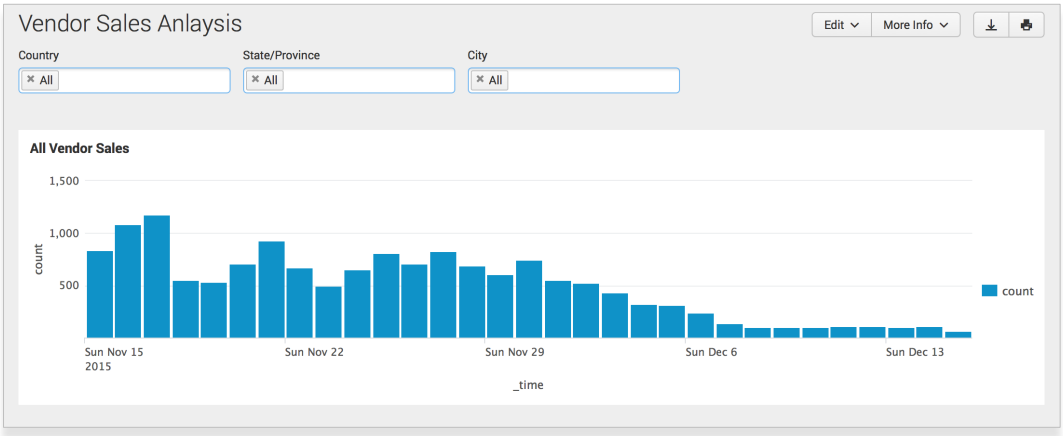
The form should display two charts of vendor data based on user input for country, state or province, and city. The first chart should display a timeline of all vendor sales with a pan & zoom for the time range. Selecting a portion of the timeline should set the second chart's time range.

The second chart should display sales by category. When a user clicks on a category, they should be redirected to a form that shows all category sales in the time range. Since these charts are related and dynamically connected, the team wants them to appear on one panel.

Working Example: Lab Examples > Lab 4a - Cascading Inputs - Example

Lab Exercise 4a: Cascading Inputs

Example:



Steps

Task 1: Create a form.

1. In the Lab Searches menu, click the **search icon** for Lab 4 - Search 1.
   Note: This search string is not intended to display results at this point. This is expected.
2. Save the search as a panel on a new dashboard:
   - Dashboard: New
   - Dashboard Title: `Vendor Sales Analysis`
   - Dashboard ID: `bcg_vendor_form_sales_analysis`
   - Panel Title: `All Vendor Sales`
   - Panel Powered By: Inline Search
3. Save, then view the dashboard.
   Note: The panel will not populate until after you've added inputs by completing the next series of tasks.

Task 2: Add a menu input for country.

4. Enter dashboard edit mode.
5. Click **+ Add Input** and select **Multiselect**.
6. Click the **Edit Input** icon (pencil) and add the following settings:

   General

   - Label: `Country`
   - Select: Search on Change

   Token Options

   - Token: v_country_tok
   - Token Prefix: `(`
   - Token Suffix: `)`
   - Token Value Prefix: `VendorCountry="`
   - Token Value Suffix: `"`
   - Delimiter: `OR`
     Important: Add a space before, and after the delimiter.
7. Examine the Preview output of the token value for any errors. It should match the example exactly as shown. Make changes if necessary. Then, scroll down to add further settings to the input.

   Example:

   Static Options

   - Name: `All`
   - Value: `*`

   Token Options

   - Default: `All`

   Dynamic Options

   - Search String: `sourcetype="vendor_sales" | stats count by "VendorCountry"`
   - Time Input: Last 7 Days
   - Field For Label: `VendorCountry`
   - Field For Value: `VendorCountry`
8. Click **Apply**, then click in the Country input and make sure a list of countries displays.
   Note: If the input is not working, check its settings for typos.

Task 3: Add a menu input for state/province.

9. Click **+ Add Input** and select **Multiselect**.
10. Click the **Edit Input** icon (pencil) and add the following settings:

    General

    - Label: `State/Province`
    - Select: Search on Change

Token Options

- Token: `v_state_tok`
- Token Prefix: `(`
- Token Suffix:)
- Token Value Prefix: `VendorStateProvince="`
- Token Value Suffix: `"`
- Delimiter: OR
  Important: Add a space before, and after the delimiter.

11. Examine the Preview output for the token value for any errors. It should match the example exactly.
    Example:

    Preview (VendorStateProvince="value1" OR V
    endorStateProvince="value2" OR ...)

    Static Options
    - Name: `All`
    - Value: `*`

    Token Options
    - Default: `All`

    Dynamic Options
    - Search String: `sourcetype="vendor_sales" $v_country_tok$ | stats count by "VendorStateProvince"`
    - Time Input: Last 7 Days
    - Field for Label: `VendorStateProvince`
    - Field for Value: `VendorStateProvince`
12. Click **Apply**, then click in the State/Province input and make sure a list of states/provinces displays.
    Note: If the input is not working, check its settings for typos.

Task 4: Add a menu input for City.

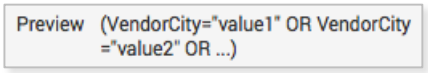
13. Click **+ Add Input** and select **Multiselect**.
14. Click the **Edit Input** icon (pencil) and add the following settings:

    General
    - Label: `City`
    - Select Search on Change

    Token Options
    - Token box: `v_city_tok`
    - Token Prefix: `(`
    - Token Suffix: `)`
    - Token Value Prefix: `VendorCity="`
    - Token Value Suffix: `"`
    - Delimiter: `OR`
      Important: Add a space before, and after the delimiter.

15. Examine the Preview output of the token value for any errors. It should match the example exactly as shown. Make changes if necessary.

    Example:

    Static Options

    - Name: `All`
    - Value: `*`

    Token Options

    - Default: All

    Dynamic Options

    - Search String: `sourcetype="vendor_sales" $v_country_tok$ $v_state_tok$ | stats count by "VendorCity"`
    - Time Input: Last 7 Days
    - Field for Label: `VendorCity`
    - Field for Value: `VendorCity`

16. Click **Apply**, then click in the City input and make sure a list of cities displays.
    Note: If the input is not working, check its settings for typos.
17. Change the All Vendor Sales panel visualization to a **Column** chart.
18. Exit dashboard edit mode.

Task 5: Test the form inputs:

19. Select a **Country, State/Province,** and **City**.
    Note: Remove the **All** option when selecting other values. Otherwise, the choices in the next menu are not limited to your selection(s).

    - Form input(s) not working?
      - Look at the XML, where errors can be more obvious.
      - Compare the XML for each input. Make sure the OR delimiter has a space on both sides of it. For example, `<delimiter> OR </delimiter>`.
    - Menus not filtering properly?
      - Reload/refresh your web page and try again.

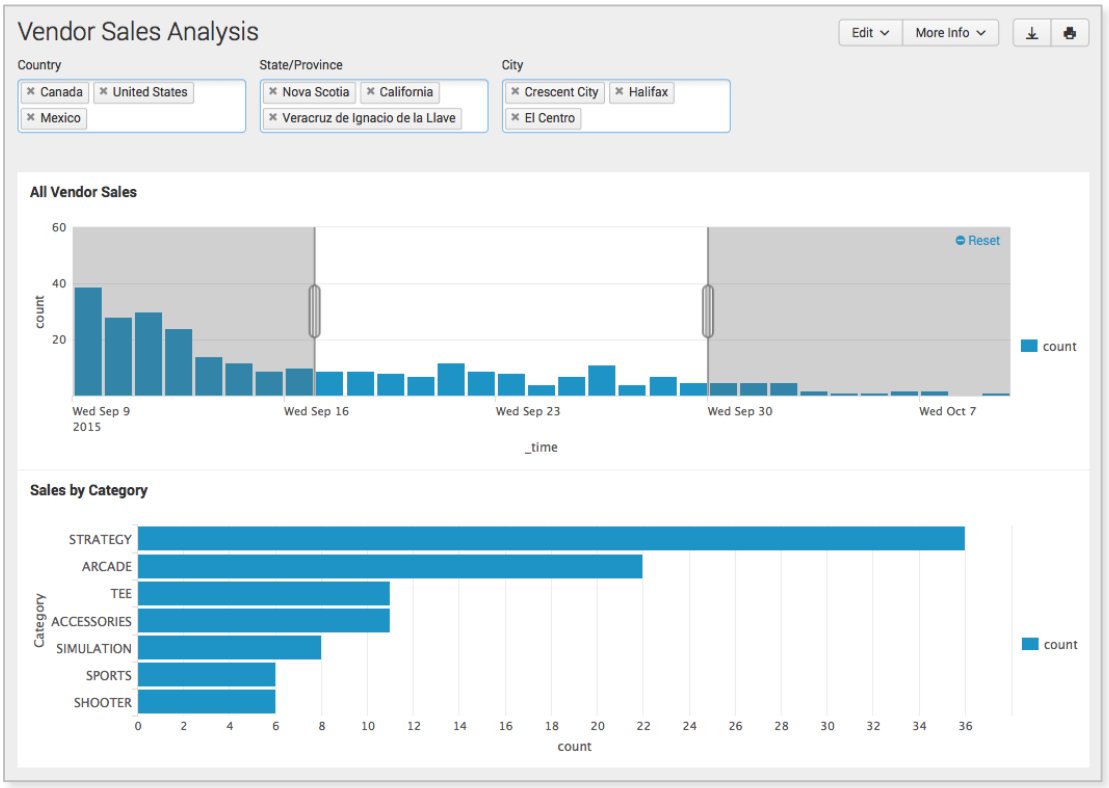
Lab Exercise 4b: Pan & Zoom with Dynamic Drilldown

Description

In this exercise you will use the <selection> and <set> elements to capture a time range from one chart and apply it to another chart on the same dashboard.

Working Example: Lab Examples > Lab 4b - Pan & Zoom with Drilldown - Example

Example:



Task 6: Add the Sales by Category chart.

20. In the Lab Searches menu, click the **search icon** for Lab 4 - Search 2.

    Note: This search string is not intended to display results at this point.

21. Save the search as a dashboard panel and add it to an existing dashboard:
    - Dashboard Title: Vendor Sales Analysis
    - Panel Title: `Sales by Category`
    - Panel Powered By: Inline Search
22. Save, then view the dashboard.
23. Open the dashboard in edit mode.
24. Change the Sales by Category panel visualization to a Bar chart.

Task 7: Merge the visualizations to one panel.

25. Open the Splunk XML Editor.

26. Delete four tags between the two panels: the closing panel tag, closing row tag, opening row, and opening panel tags.

```
...
  </chart>
 </panel>
</row>
<row>
 <panel>
  <chart>
...
```

Task 8: Add a visualization event handler.

27. Locate the XML code for the All Vendor Sales panel.
28. Add the following `<selection>` XML before the closing `</chart>` tag:

```
...
 <option name="charting.legend.placement">right</option>
  <selection>
   <set token="selection.earliest">$start$</set>
   <set token="selection.latest">$end$</set>
  </selection>
 </chart>
...
```

29. Locate the XML for the Sales by Category panel.
30. Replace the existing earliest and latest settings with predefined time selection tokens.

```
...
 </query>
  <earliest>-30d@d</earliest>
  <latest>now</latest>
  <earliest>$selection.earliest$</earliest>
  <latest>$selection.latest$</latest>
 </search>
...
```

31. Save the XML changes.
32. Exit dashboard edit mode.

Task 9: Test the cascading menus and event handler.

33. Select a **Country, State/Province,** and **City**.
    Note: Remove the All option when selecting other values; otherwise, the choices in the next menu are not limited to your selection(s).
34. Reduce the time range by clicking and dragging on the All Vendor Sales visualization.
    Note: The Sales by Category panel should refresh accordingly.

Task 10: Add a dynamic drilldown.

35. Open the Splunk XML Editor.
36. Locate the XML for the Sales by Category panel.
37. Add the following <drilldown> and <link> XML below the option tags:

```
...
<option name="charting.legend.placement">right</option>
  <drilldown>
    <link>
    <![CDATA[/app/advdash/bcg_vendor_sales_drilldown_dest?form.categoryId
    =$click.value$&earliest=$earliest$&latest=$latest$]]>
    </link>
  </drilldown>
</chart>
...
```

Note: If you copy and paste the above text, watch for extra spaces. Especially after categoryId.

38. Save the XML changes.

Task 11: Create a drilldown destination form.

39. In the Lab Searches menu, click the **search icon**  for Lab 4 - Search 3.

Note: This search is not intended to display results at this point. This is normal and expected.

40. Save the search as a dashboard panel and add it to a new dashboard.
    - Dashboard Title: `Sales Drilldown Destination`
    - Dashboard ID: `bcg_vendor_sales_drilldown_dest`
      Note: Use this exact dashboard ID name. As you can see in the previous task, it is hard coded in the drilldown link.
    - Panel Title: `Sales Trend: $categoryId$`
    - Panel Power by: Inline Search
41. Save, then view the dashboard.
42. Enter dashboard edit mode.
43. Click **+ Add Input** > **Text**.
44. Click the **Edit Input** icon (pencil) add the following settings:

    General
    - Label: `Enter a category name:`
    - Token: `categoryId`
    - Default: `*`
45. Click **Apply**.

Task 12: Add a time input.

46. Use the following settings:

    Token Options

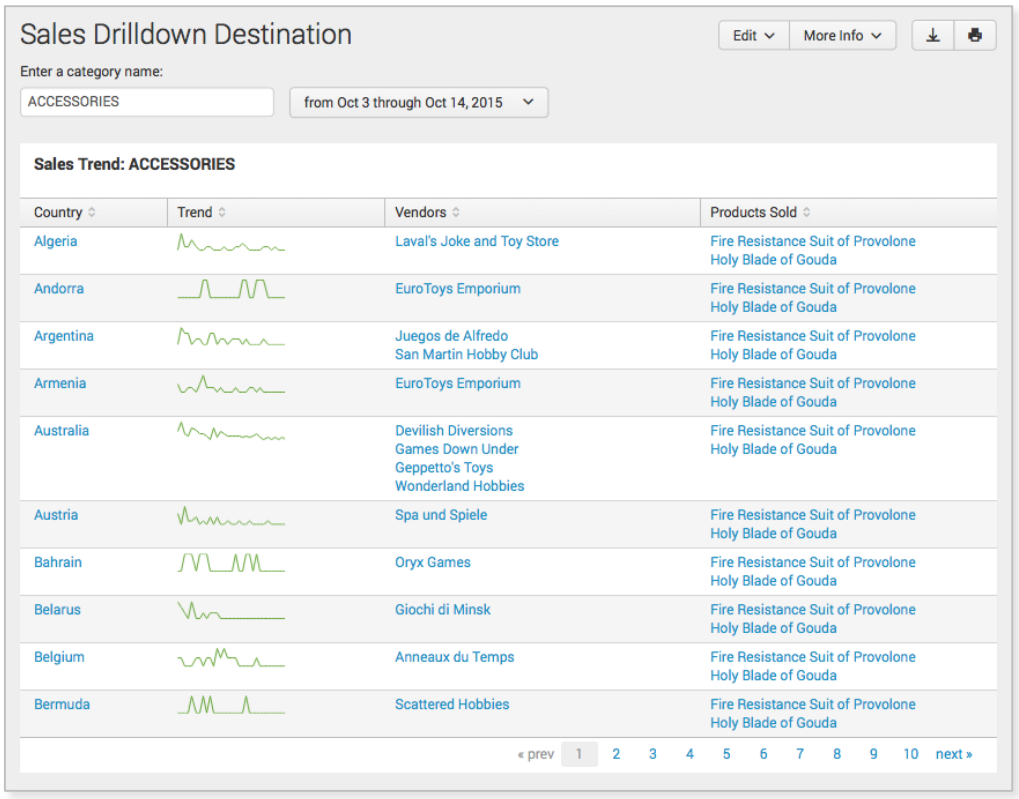
    - Token: Delete the default token. Leave empty.
    - Default: Last 30 Days

47. Click the **search properties icon**  on the Sales Trend table.
48. Select **Edit Search String**.
49. Select **Time Range Scope** > **Shared Time Picker (global)**.
50. Click **Save**.
51. Exit dashboard edit mode.

Task 13: Test the dynamic drilldown.

52. Select **Lab Dashboards** > **Vendor Sales Analysis**.
53. Select a Country, State/Province, and City.
54. Reduce the time range by clicking and dragging on the All Vendor Sales chart.
55. On the Sales by Category panel, click one of the categories on the bar chart.
    The drilldown destination form should open and populate with the category you clicked on, and the time range should display the custom time range value passed from the Vendor Sales Analysis form.

Example:

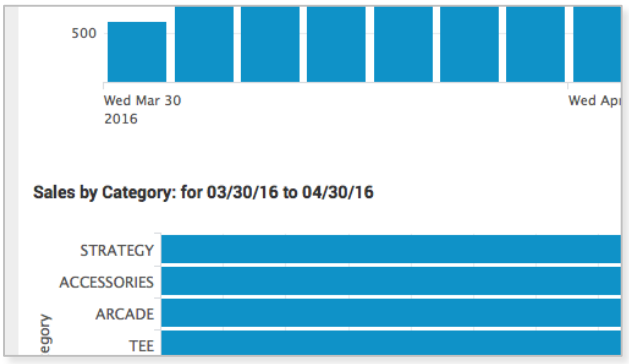


Challenge Lab (optional)

Add a selected time range to the panel title.

In your dashboard you have a pan & zoom that shows a selection in the top panel visually and on the x-axis. Yet the bar chart doesn't show the time range. Adding the selected time range to the sales by category panel title is an easy way to show this.

Example:



Task 1: Add a time range to a panel title.

1. Select **Lab Dashboards** > **Vendor Sales Analysis**.
2. Open the Splunk XML editor.
3. Locate the All Vendor Sales <selection> tag.
4. Set two additional tokens in the <selection> group, one for the earliest selected time and one for the latest selected time.

```
<selection>
<set token="selection.earliest">$start$</set>
<set token="selection.latest">$end$</set>
<eval token="fromDate">strftime($selection.earliest$,"%m/%d/%y")</eval>
<eval token="toDate">strftime($selection.latest$, "%m/%d/%y")</eval>
</selection>
```

   Note: In Splunk 6.3 there is a known issue that requires adding single quotes around the token when used in the strftime function. For example: strftime('$selection.latest$',"%m/%d/%y"). This is no longer an issue in Splunk 6.4.

5. Locate the Sales by Category <title> tag.
6. Add the two tokens to the title.

```
<title>Sales by Category for $fromDate$ to $toDate$</title>
```

7. Save the XML changes.

Task 2: Test the new panel title.

8. Select a Country, State/Province, and City.
9. Reduce the time range by clicking and dragging on the All Vendor Sales chart.
   The Sales by Category panel title should reflect the dates of the selection. If not, check for XML typos.

Lab Exercise 5 – Add Advanced Visualizations

Description

In this exercise you will use simple XML extensions to add advanced visualizations and behaviors to a dashboard. The dashboard will include a horizon chart. A horizon chart is similar to an area chart that is restricted to a smaller vertical space. Portions of the chart that fall above or below the defined area are shown in shades of the main color or different colors.

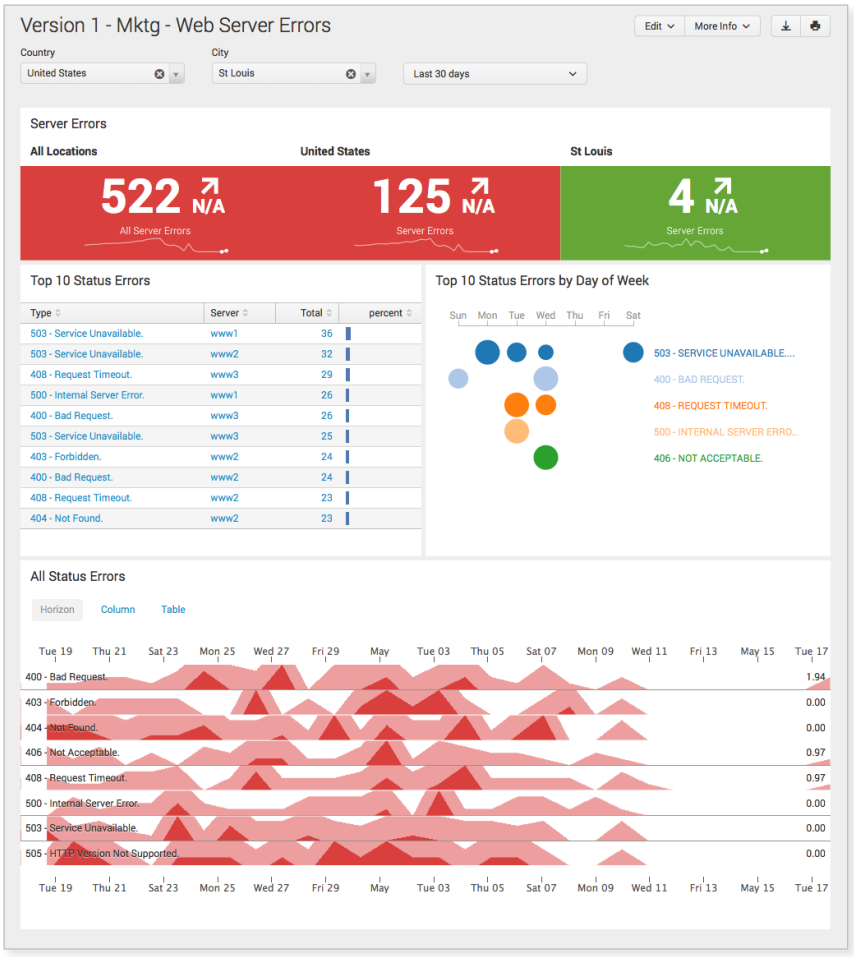
Scenario: The web marketing team has seen the sales team's web store server errors dashboard and would like something similar but with a greater emphasis on status errors. The dashboard should allow users to select a time range, and locations by country or city.

It should have panels that display metrics for the following:

- Global server errors
- Global status errors over time
- Server errors by country
- Server errors by city
- Top 10 status errors
- Top 10 status errors by day of week

Working Example: Lab Examples > Lab 5 - Add Advanced Visualizations - Example

Example:



Steps

Task 1: Create a new dashboard.

1. Use the following title and ID:
   - Title: `Version 1 - Mktg - Web Server Errors`
   - ID: `mktg_web_dash_server_errors_v1`
   - Permissions: Private

Task 2: Add a dropdown input for Country.

2. Use the following settings:

   General
   - Label: `Country`
   - Select: Search on Change

   Token Options
   - Token: `country_tok`

   Static Options
   - Name: `All`
   - Value: `*`

   Token Options
   - Default: `All`

   Dynamic Options
   - Search String: `sourcetype=access_combined | iplocation clientip | stats count by "Country"`
   - Time Input: Last 7 Days
   - Field For Label: `Country`
   - Field For Value: `Country`
3. Click **Done.**
4. Click the **Country** dropdown menu and make sure a list of countries displays.

Task 3: Add a dropdown input for City.

5. Use the following settings:

   General
   - Label: `City`
   - Select: Search on Change

   Token Options
   - Token box: `city_tok`

   Static Options
   - Name: `All`
   - Value: `*`

   Token Options
   - Default: `All`

Dynamic Options

- Search String: `sourcetype=access_combined | iplocation clientip | eval City = if(match(City,"^\w"),City,"unknown") | search Country="$country_tok$" City!="unknown" | stats count by "City"`
- Time Input: Last 7 Days
- Field For Label: `City`
- Field for Value: `City`

6. Click **Done.**
7. Click the **City** dropdown menu and make sure a list of cities displays.

Task 4: Add a time input.

8. Use the following settings:

General

- Select: Search on Change

Token Options

- Token: `time_tok`
- Default: Last 7 Days

Task 5: Add a panel that shows status errors for all locations.

9. Search for HTTP status error events from online transactions during the last 7 days.
10. Use the `iplocation` command to extract the location of the errors based on clientip.
11. Add a `timechart` command to count the events.
12. Display the results as a Single Value visualization.
13. Save the search as a panel on an existing dashboard:
    - Dashboard: `Version 1 - Mktg - Web Server Errors`
    - Panel Title: `All Locations`
    - Panel Powered By: Inline Search
    - Panel Content: Single Value
14. Save, then view the dashboard.

Task 6: Add a panel that shows status errors for a selected country.

15. Open the Splunk XML Editor.
16. Copy the XML for the All Locations panel.
17. Paste the XML on the same row, after the existing All Locations panel.
18. Revise the visualization title to use the country token: `$country_tok$`
19. Add a search command to the query that filters results to the current value of the token: `| search Country="$country_tok$"`

```
...
<option name="useThousandSeparators">1</option>
   </single>
  </panel>
  <panel>
   <single>
     <title>$country_tok$</title>
```

```
      <search>
        <query>sourcetype=access_combined  status!=200 clientip=*
          | iplocation clientip | search Country="$country_tok$"
          | timechart count</query>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </search>
      <option name="colorBy">value</option>
      <option name="colorMode">none</option>
      <option name="drilldown">none</option>
      <option name="numberPrecision">0</option>
      ...
    </single>
  </panel>
 </row>
</form>
```

Task 7: Add a panel that shows status errors for a selected city.

20. Copy the XML for the Country panel, and paste it on the same row, after the existing Country panel.
21. Revise the visualization title to use the city token: `$city_tok$`
22. Revise the search to use the current value of the **city** token: `City="$city_tok$"`

```
    ...
    </single>
  </panel>
  <panel>
   <single>
     <title>$city_tok$</title>
     <search>
       <query>sourcetype=access_combined  status!=200 clientip=*
         | iplocation clientip | search Country="$country_tok$"
             City="$city_tok$" | timechart count</query>
       <earliest>-7d@h</earliest>
       <latest>now</latest>
     </search>
     <option name="colorBy">value</option>
     <option name="colorMode">none</option>
     <option name="drilldown">none</option>
     <option name="numberPrecision">0</option>
     <option name="showSparkline">1</option>
     <option name="showTrendIndicator">1</option>
     <option name="trendColorInterpretation">standard</option>
     <option name="trendDisplayMode">absolute</option>
     <option name="unitPosition">after</option>
     <option name="useColors">0</option>
     <option name="useThousandSeparators">1</option>
   </single>
  </panel>
 </row>
</form>
```

Task 8: Add a panel title and merge the single-value visualizations to one panel.

23. On the panel with the All Locations visualization, add opening and closing <title> tags with the text: `Server Errors`
24. Delete the closing and opening <panel> tags between the three visualizations.

```
...
 <row>
   <panel>
    <title>Server Errors</title>
    <single>
     <title>All Locations</title>
      <search>
...
    </single>
  </panel>
  <panel>
    <single>
...
    </single>
  </panel>
  <panel>
    <single>
...
```

25. Save the XML changes.
    The three single-value visualizations should now display in one panel.

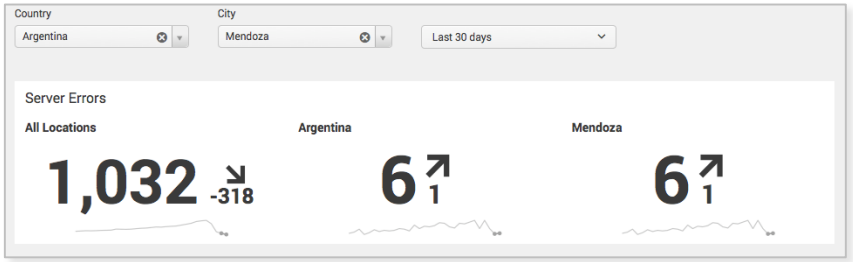
Task 9: Revise each visualization's search to use the Shared Time Picker.

26. Enter dashboard edit mode.
27. Click the search properties icon on the All Locations visualization. 
28. Select **Edit Search String**.
29. Select **Time Range Scope** > **Shared Time Picker (time_tok)**.
30. Click **Save**.
31. Repeat the above steps for the two other panels:
    - Country
    - City
32. Exit dashboard edit mode.

Task 10: Test the tokens.

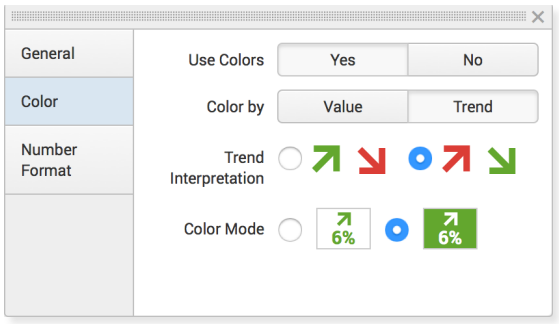
33. Select a Country, City, and time from the dropdown menus.
34. Verify the Country and City panel titles and values update to reflect your input choices.

Example:



Task 11: Format the single-value panels.

35. Format the All Locations visualization with the following settings:



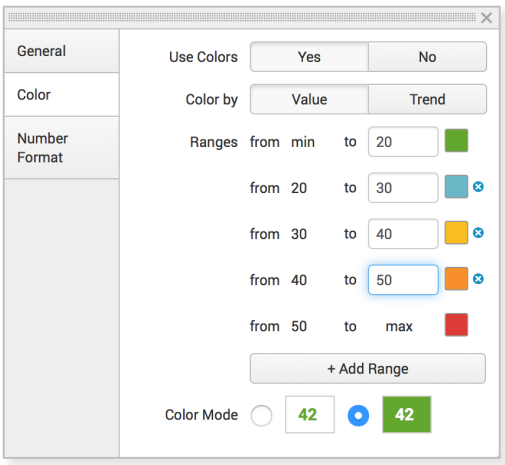
General

- Drilldown: No
- Show Trend Indicator: Yes
- Show Trend in: Percent
- Compared to: Custom > Auto
- Caption: `All Server Errors`
- Show Sparkline: Yes

Color

- Use Colors: Yes
- Color by: Trend
- Trend Interpretation: Negative values in green
- Color Mode: Block background

36. Format the two remaining single value visualizations, Country and City, with the following settings:



General

- Drilldown: No
- Show Trend Indicator: Yes
- Show Trend in: Percent
- Compared to: Custom > Auto
- Caption: `Server Errors`
- Show Sparkline: Yes

Color

- Use Colors: Yes
- Color by: Value
- Ranges:
  - from min - 20, green (65a637)
  - from 20 - 30, light blue (6db7c6)
  - from 30 - 40, yellow (f7bc38)
  - from 40 - 50, orange (f7bc38)
  - from 50 - max, red (d93f3c)
- Color Mode: Block background

Task 12: Add a prebuilt panel that displays a table of the top 10 status errors.

37. Add a prebuilt panel: Top 10 Status Errors

Task 13: Convert the prebuilt panel to inline.

38. On the Top 10 Status Errors panel, click the **Options** icon.

Task 14: Add simple XML extensions for JavaScript and CSS to display a bar in a table cell.

39. Open the Splunk XML Editor.

40. Locate the opening <form> tag and add a script reference for table_data_bar.js and table_data_bar.css:

```
<form script="table_data_bar.js" stylesheet="table_data_bar.css">
  <label>Version 1 - Mktg - Web Server Errors</label>
...
```

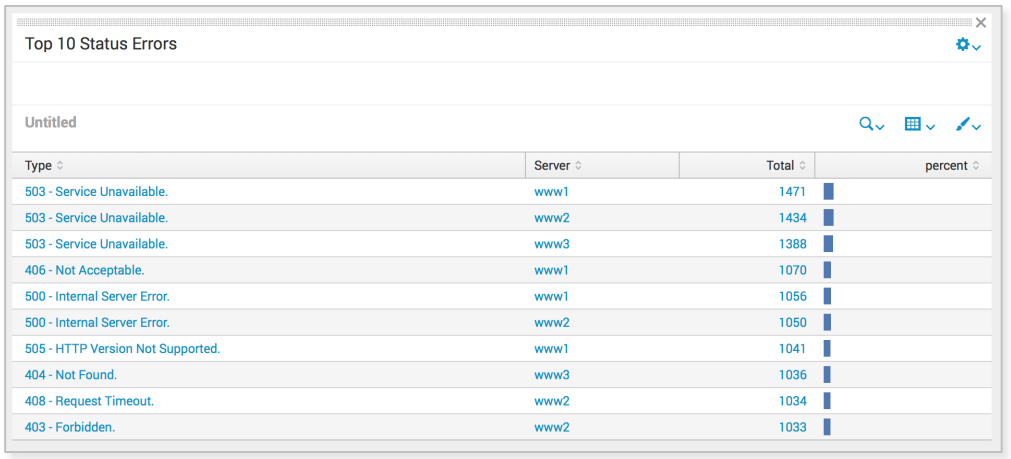
Note: These two files have already been uploaded to /appserver/static directory of the course app. You can find them after class in the Splunk Dashboard Examples app on Splunkbase.

41. Locate the XML for the Top 10 Status Errors table.
42. Add an id to the table opening tag: `id="table1"`

```
...
<row>
 <panel>
   <title>Top 10 Status Errors</title>
  <table id="table1">
    <search>
...
```

43. Save the XML changes.

Example:



Task 15: Use simple XML extensions to display a punchcard visualization.

44. Add a prebuilt panel: Top 10 Server Errors by Day of Week
45. Position panel on the right of the Top 10 Status Errors panel.

Task 16: Convert the prebuilt panel to inline.

46. On the Top 10 Server Errors by Day of Week panel, click the **Options** icon.
47. Open the Splunk XML Editor.
48. Locate the <form> tag and add a script reference for autodiscover.js followed by a comma.

```
<form script="autodiscover.js, table_data_bar.js" stylesheet="table_data_bar.css">
<label>Version 1 - Mktg - Web Server Errors</label>
...
```

49. Locate the XML for the Top 10 Server Errors by Day of Week panel.
50. Remove the opening and closing <table> </table> tags.

51. Revise the <search> tag to have an id: `<search id="punchcard_search">`

```
...
    </table>
  </panel>
  <panel>
   <title>Top 10 Server Errors by Day of Week</title>
    <table>
    <search id="punchcard_search">
     <query>sourcetype=access_combined  status!=200 clientip=*
     | iplocation clientip
     | search Country="$country_tok$" City="$city_tok$"
     | eval wday=strftime(_time, "%a")
     | top wday, description</query>
     <earliest>$time_tok.earliest$</earliest>
     <latest>$time_tok.latest$</latest>
    </search>
    </table>
  </panel>
 </row>
...
</form>
```

52. Add opening and closing <html> </html> tags after the closing </search> tag.

```
...
   <latest>$time_tok.latest$</latest>
    </search>
    <html>
    </html>
  </panel>
 </row>
</form>
```

53. Add tag references and data options:

```
  ...
   <html>
     <!--- Add code from step 53 Lab 5 Supporting File.txt here --->
   </html>
   ...
```

Important: When adding single quotes, as in this step, use **escaped** single quotes for the settings nested under the data-options element, as shown in the supporting file code. For comparison, these are the settings:

- id reference: `id="punchcard"`
- class reference: `class="splunk-view"`
- data-require element with the path to the visualization files: `data-require="app/advdash/components/punchcard/punchcard"`

data-options

- data-options element: `data-options='{`
- 'managerid': 'punchcard_search',
- 'range_values': ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'],
- 'height': 300

54. Save the XML changes.
    Note: The punchcard panel should now populate. If not, check for typos in the XML, a missed step, such as the autodiscover.js extension, or copy the code from the lab 5 supporting file.

Task 17: Add prebuilt panels.

55. Add three prebuilt panels:
    - Lab 5 - Chart 1
    - Lab 5 - Chart 2
    - Lab 5 - Chart 3

Task 18: Convert prebuilt panels to inline panels.

56. Convert all three prebuilt panels to inline.

Task 19: Change panel visualizations.

57. Change Lab 5 - Chart 1 table to a Horizon chart.
58. Change Lab 5 - Chart 2 table to a Column chart.

Task 20: Add a link switcher.

59. On the dashboard, click **+ Add Input** > **Link List**.
60. Drag the input onto the horizon chart.
61. Click the **Edit Input** icon (pencil) and add the following settings:

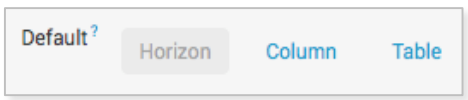
    General
    - Label: delete field1 (leave empty)
    - Select: Search on Change

    Token Options
    - Token: `unused`
      Note: Type the text: `unused`

    Static Options

Name	Value
Horizon	horizon
Column	column
Table	table

    Token Options
    - Default: Horizon



62. Click **Apply**.

Task 21: Add a form input event handler.

63. Open the Splunk XML Editor.
64. Locate the XML for the link input.
    Tip: Scroll down to the XML for Lab 5 - Chart 1.
65. Add opening and closing <change></change> tags after the closing </choice> tag for Table.
66. Add opening and closing <condition></condition> tags between the <change></change> tags.

```
...
<choice value="table">Table</choice>
  <change>
   <condition>
   </condition>
  </change>
<default>horizon</default>
</input>
...
```

67. Add the attribute `value="horizon"` to the opening <condition> tag.

```
...
<choice value="table">Table</choice>
  <change>
   <condition value="horizon">
   </condition>
 </change>
 <default>horizon</default>
</input>
...
```

68. Add an <unset token> open and closing tag between the <condition> tags with the token: `showTable`
69. Add another <unset token> open and closing tag between the <condition> tags with the token: `showCol`
70. Add a <set token> open and closing tag between the <condition> tags with the token: `showHorizon` and the value: `true`

```
...
<choice value="table">Table</choice>
  <change>
   <condition value="horizon">
      <unset token="showTable"></unset>
      <unset token="showCol"></unset>
      <set token="showHorizon">true</set>
   </condition>
</change>
...
```


71. Repeat the above steps, adding a <condition> with a <set token> and two <unset token> tags for the two remaining visualizations: column and table

```
...
  <change>
   <condition value="horizon">
     <unset token="showTable"></unset>
     <unset token="showCol"></unset>
     <set token="showHorizon">true</set>
   </condition>
   <condition value="column">
     <unset token="showTable"></unset>
     <set token="showCol">true</set>
     <unset token="showHorizon"></unset>
   </condition>
   <condition value="table">
     <set token="showTable">true</set>
     <unset token="showCol"></unset>
     <unset token="showHorizon"></unset>
   </condition>
  </change>
...
```

72. Revise the panel title to be: `All Status Errors`

```
...
<row>
 <panel>
  <title>All Status Errors</title>
  <input type="link" token="unused" searchWhenChanged="true">
...
```

73. Locate the XML for the horizon chart visualization.

74. Add an id to the <viz> tag: `id="horizon"`

75. Add a depends attribute to the <viz> tag: `depends="$showHorizon$"`

```
...
<default>horizon</default>
</input>
<viz id="horizon" depends="$showHorizon$" type="horizon_chart_app.horizon_chart">
<search>
...
```

76. Locate the XML for Lab 5 - Chart 2.

77. Delete the title.

```
<row>
 <panel>
  <title>Lab 5 - Chart 2</title>
  <chart>
   <search>
...
```

78. Add an id to the <chart> tag: `id="column"`

79. Add a depends attribute to the <chart> tag: `depends="$showCol$"`

```
...
<panel>
<chart id="column" depends="$showCol$">
...
```


80. Locate the XML for Lab 5 - Chart 3.
81. Delete the title tag and its contents.

```
<row>
 <panel>
  <title>Lab 5 - Chart 3</title>
  <table>
   <search>
...
```

82. Add an id to the <table> tag: `id="table"`
83. Add a depends attribute to the <table> tag: `depends="$showTable$"`

```
...
<panel>
<table id="table" depends="$showTable$">
...
```

84. Save the XML changes.

Task 22: Format the Horizon Chart.

85. Format the horizon chart visualization with the following settings:

General

- Number of bands: 2
- Calculate relative change: No
- Show change in: Absolute value
- Smooth: Yes

Colors

- Negative color: Blue (6db7c6)
- Positive color: Red (d93f3c)

86. Exit dashboard edit mode.
87. Test the links.

Example: