splunk>

Creating Dashboards Lab Exercises

Overview

Welcome to the Splunk Education lab environment. These exercises will guide you through the process of creating a set of dashboards, forms, and visualizations. If you get stuck, consult the example dashboard in the course app.

IMPORTANT: Save all knowledge objects in the Creating Dashboards course app with permissions set to private.

If you copy text from this document, please note that character formatting and artifacts created by the PDF generation process can cause errors in the XML. Consider using a text editor as an interim step.

Panel Visualization Editor

Search	Visualization Picker	Format Menu	More Actions
Select this to modify the search. The icon indicates the panel's search type: • magnifying glass: inline search • sheet of paper: a report • pivoting arrows: a pivot	Select this to choose the visualization type.	Select this to format the visualization.	Select this to split a visualization into a trellis layout or to access the drilldown editor.

Typographical Conventions

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

Lab Connection Info

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

Splunk Web URL:	
Splunk Web username:	
Splunk Web password:	

Source Types

The two source types used in these exercises 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
Server access data	linux_secure	action, app, dest, process, src_ip, src_port, user, vendor_action
Event Annotations	bcg_sale_date	Sale_Category, Sale_Date, message

Lab Exercise 1 – Create a Prototype

Description

Create a dashboard prototype based on the scenario and wireframe. The prototype will use basic searches and visualizations.

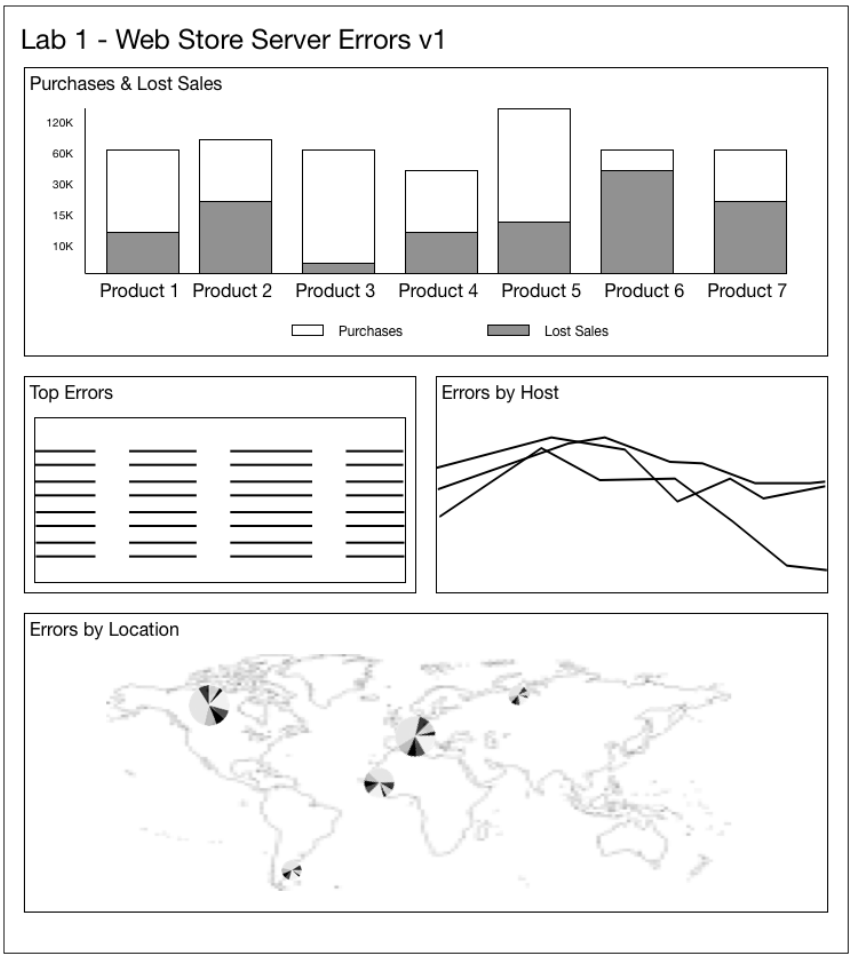
IMPORTANT: Perform all searches and save all knowledge objects in the Creating Dashboards course app.

Scenario: **The sales team wants a dashboard that displays information about web store 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 web server errors by host
- Map of web server errors by client location

Working Example: Creating Dashboards > Lab Examples > Lab 1 Example - Create a Prototype

Wireframe:



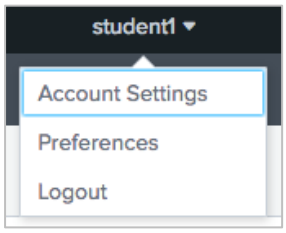
Steps

Task 1: Change the account name and time zone.

☐ 1. Navigate to **User Menu** > **Account Settings**.

☐ 2. In the Full name box, enter your name: `<your firstname lastname>`
For example: bob munson

☐ 3. Click **Save**.



☐ 4. Navigate to **User Menu > Preferences.**

☐ 5. Enter the following settings:

- Time zone: `<your local time zone>`
- Default application: Creating Dashboards

☐ 6. Click **Apply**.

Task 2: Create a dashboard with a panel that shows online purchases and lost sales.

☐ 7. Navigate to **Apps** > **Creating Dashboards**.

TIP: Since your default application is now Creating Dashboards, click the Splunk logo in the upper left to go to this app.

☐ 8. Search online transactions for purchases or lost sales events **over the last 7 days**.

```
index=web sourcetype=access_combined (action=remove OR action=purchase)
```

☐ 9. Chart the sum of sales as totalSales by product_name and action.

```
index=web sourcetype=access_combined (action=remove OR action=purchase)
| chart sum(price) as totalSales by product_name, action
```

☐ 10. Rename product_name as Product Name, remove as Lost Sales, and purchase as Purchases.

```
index=web sourcetype=access_combined (action=remove OR action=purchase)
| chart sum(price) as totalSales by product_name, action
| rename product_name as "Product Name", remove as "Lost Sales", purchase as Purchases
```

☐ 11. Display the results as a **column chart**.

a. Click the **Visualization** tab.
b. Click the **Select visualization** menu.
c. Select **Column Chart**.

☐ 12. Format the visualization to display a stacked column chart with the legend on the bottom.

a. Click the **Format visualization** button.
b. Click the **General** tab and set Stack Mode to **stacked**.
c. Click the **Legend** tab and position the legend on the **Bottom**.

☐ 13. Save the search as a **report,** add it to a **new dashboard,** and **view the dashboard**:

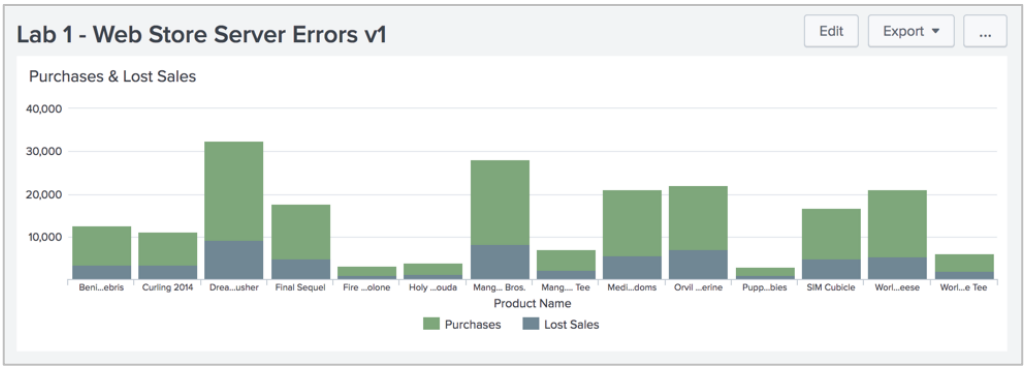
- Report Title: `L1_ws_purch_vs_lost_sales`
- Content: Column Chart
- Time Range Picker: No

- Dashboard Title: `Lab 1 - Web Store Server Errors v1`
- Dashboard ID: `lab_1__web_store_server_errors_v1`
- Dashboard Permissions: Private
- Panel Title: `Purchases & Lost Sales`
- Panel Powered By: Report
- Drilldown No action
- Panel Content: Column Chart

---

a. Click **Save As** > **Report**.
b. In the Report Title box, type: `L1_ws_purch_vs_lost_sales`
c. Select Time Range Picker: No
d. Click **Save**.
e. Click **Add to Dashboard**.
f. Select Dashboard: New
g. In the Dashboard Title box, type: `Lab 1 - Web Store Server Errors v1`
h. In the Dashboard ID box, accept the default: lab_1__web_store_server_errors_v1
i. Dashboard Permissions: Private
j. In the Panel Title box, type: `Purchases & Lost Sales`
k. Select Panel Powered By: Report
l. Click **Save**.
m. Click **View Dashboard.**

Example:



Task 3: Add a table of most common web server errors by host.

☐ 14. Search online transactions for all HTTP status errors **over the last 7 days**.

```
index=web sourcetype=access_combined status>399
```

☐ 15. Count errors by host and status.

```
index=web sourcetype=access_combined status>399
 | chart count by host, status
```

☐ 16. Limit results to only the top three errors.

```
index=web sourcetype=access_combined status>399
 | chart count by host, status limit=3
```

☐ 17. Remove the grouping called OTHER.

```
index=web sourcetype=access_combined status>399
 | chart count by host, status limit=3 useother=f
```

☐ 18. Save the search as a **report,** add it to an **existing dashboard,** and **view the dashboard:**

- Report Title: `L1_ws_common_errors_by_host`
- Content: Statistics Table
- Time Range Picker: No

- Dashboard: Lab 1 - Web Store Server Errors v1
- Panel Title: `Most Common Errors by Host`
- Panel Powered By: Report
- Drilldown No action
- Panel Content: Statistics Table

Example:

Most Common Errors by Host

host	400	406	503
www1	551	574	796
www2	556	516	751
www3	540	556	740

Task 4: Add a time series chart of all web server errors.

☐ 19. Search online transactions for HTTP status errors **over the last 7 days**.

```
index=web sourcetype=access_combined status>399
```

☐ 20. Plot a count of action values by host over time.

```
index=web sourcetype=access_combined status>399
 | timechart count(action) by host
```

☐ 21. Display the results as a **line chart**.

☐ 22. Format the visualization to display the **legend on the bottom**.

☐ 23. Save the search as a **report**, add it to an **existing dashboard,** and **view the dashboard**:

- Report Title: `L1_ws_all_errors_by_host`
- Content: Line Chart
- Time Range Picker: No

- Dashboard: Lab 1 - Web Store Server Errors v1
- Panel Title: `All Errors by Host`
- Panel Powered By: Report
- Drilldown No action
- Panel Content: Line Chart

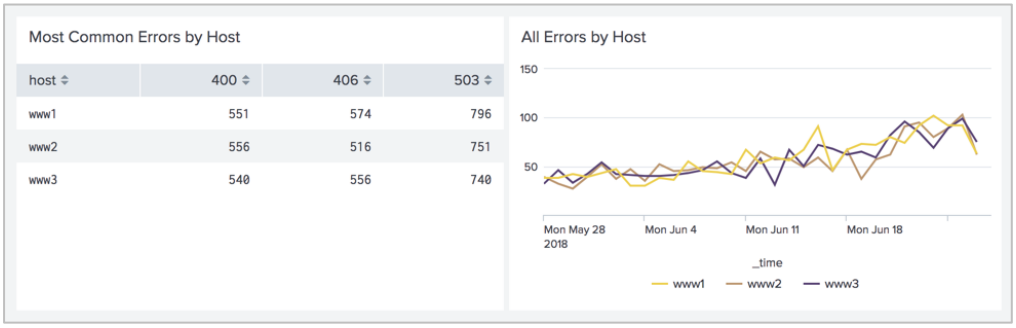
☐ 24. Open the dashboard editor.

a. Click **Edit**.

☐ 25. Position the All Errors by Host panel on the right of the Most Common Errors by Host panel.

☐ 26. Save the dashboard changes.

Example:



Task 5: Add a map of web store server error locations.

☐ 27. Search online transactions for client IP addresses and HTTP error codes **over the last 7 days**.

```
index=web sourcetype=access_combined clientip=* status>399
```

☐ 28. Eliminate duplicate client IP addresses and hosts.

```
index=web sourcetype=access_combined clientip=* status>399
| dedup clientip, host
```

☐ 29. Use the iplocation command to extract location information from IP addresses and add a prefix to the fields generated by it.

```
index=web sourcetype=access_combined clientip=* status>399
| dedup clientip, host
| iplocation prefix=cip_ clientip
```

☐ 30. Create a cluster map based on ip location and count events by status.

```
index=web sourcetype=access_combined clientip=* status>399
| dedup clientip, host
| iplocation prefix=cip_ clientip
| geostats latfield=cip_lat longfield=cip_lon count by status
```

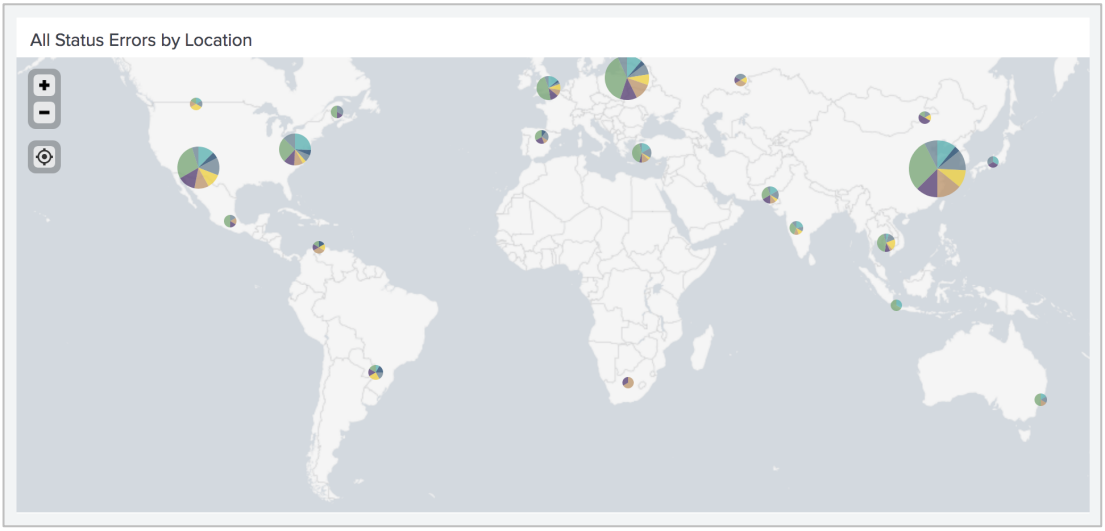
NOTE: There is a space between prefix=cip_ and clientip.

☐ 31. Save the search as a **report**, add it to an **existing dashboard,** and **view the dashboard**:

- Report Title: `L1_ws_status_errors_by_location`
- Content: Cluster Map
- Time Range Picker: No

- Dashboard Title: Lab 1 - Web Store Server Errors v1
- Panel Title: `All Status Errors by Location`
- Panel Powered By: Report
- Drilldown: No action
- Panel Content: Cluster Map

Example:

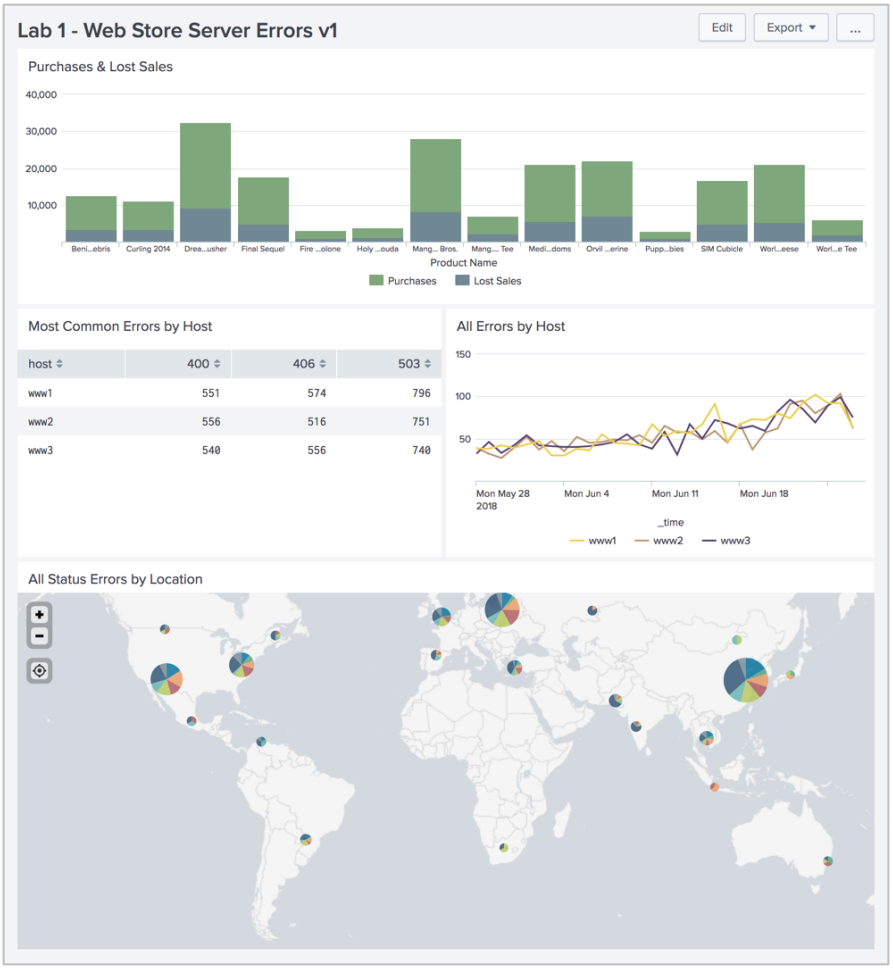


Task 6: Create prebuilt panels.

- [ ] 32. Open the dashboard editor.
- [ ] 33. On the Most Common Errors by Host panel, click the **Options Menu** icon .
- [ ] 34. Select **Convert to Prebuilt Panel**.
- [ ] 35. Add the prefix `L1_ws_` to the panel title.
    - ID: `L1_ws_<panel_id>`
      Example: L1_ws_most_common_errors_by_host
    - Permissions: Private
- [ ] 36. Click **Apply**.
- [ ] 37. Click **OK** to confirm.
- [ ] 38. Repeat the above steps to convert two other panels and use the same prefix:
    - All Errors by Host: `L1_ws_all_errors_by_host`
    - All Status Errors by Location: `L1_ws_all_status_errors_by_location`
- [ ] 39. Save the dashboard changes.

Example on next page.

Example:



☐ 40. Navigate to: **Settings** > **User Interface** > **Prebuilt Panels**

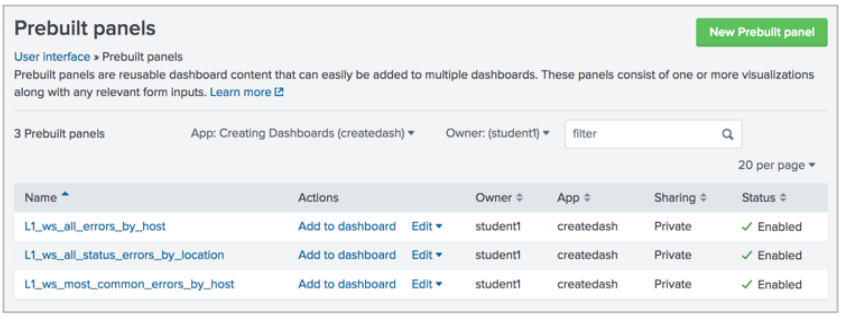
☐ 41. In the Filter by owner dropdown, select your name.

☐ 42. In the App dropdown, select **Creating Dashboards (createdash)**.

☐ 43. Make sure all three of your prebuilt panels are listed:

- L1_ws_all_errors_by_host
- L1_ws_all_status_errors_by_location
- L1_ws_most_common_errors_by_host

Example:

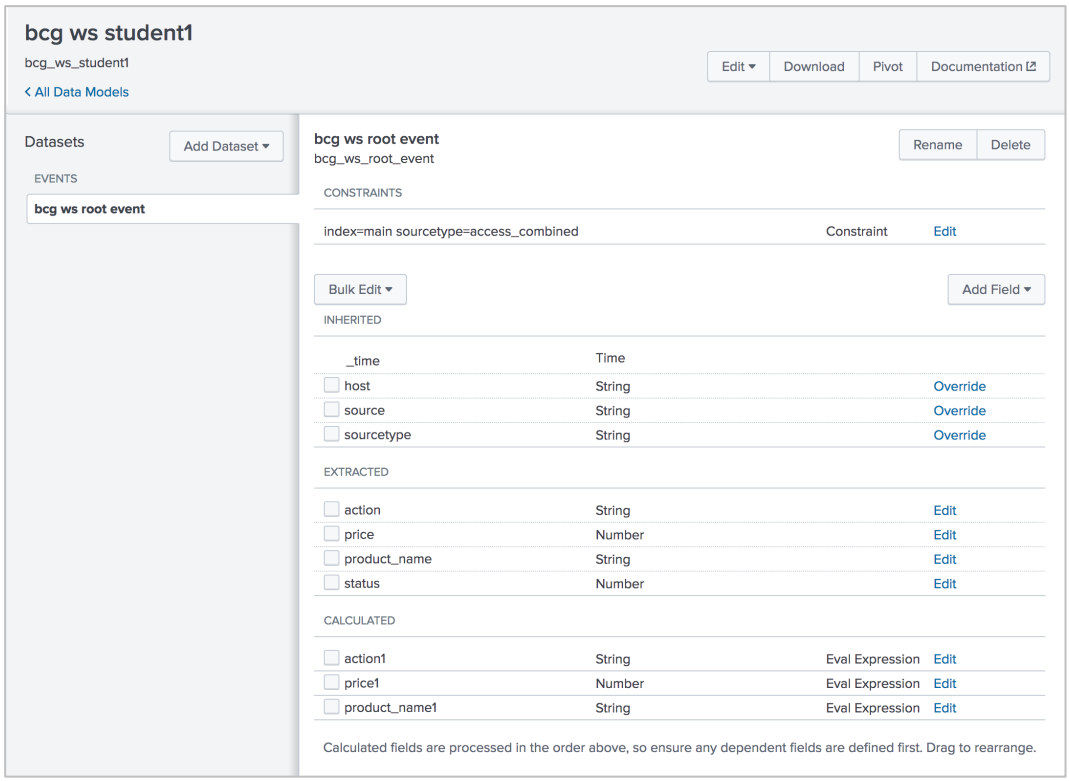


Challenge Lab Exercise (optional)

Create a Data Model

This exercise walks you through the process of creating a data model that can be accelerated. You must first complete this challenge exercise if you intend to do the lab 3 challenge exercise.

Example:



Task 1: Create a data model.

IMPORTANT: Append your student number to the data model ID to distinguish it from other student's data models. For example, bcg_ws_student7.

☐ 1. Select **Settings** > **Data models**.

☐ 2. In the App dropdown, select **Creating Dashboards (createdash)**.

☐ 3. Click **New Data Model** and enter the following settings replacing <X> with your student number:

- Title: bcg ws `student<X>`
- ID: bcg_ws_`student<X>`
- App: Creating Dashboards

☐ 4. Click **Create**.

☐ 5. Add a root event dataset.

- Dataset Name: `bcg ws root event`
- Dataset ID: `bcg_ws_root_event`
- Constraints: `index=web sourcetype=access_combined`

☐ 6. Preview events to verify the constraint is working properly, then **save** it.

Task 2: Add auto-extracted fields.

☐ 7. Select **Add Field** > **Auto-Extracted**.

☐ 8. Select five auto-extracted fields:

- action
- index
- price
- product_name
- status

☐ 9. Click **Save.**

Task 3: Add eval expression fields for NULL values.

NOTE: Searches against this data model require all fields identified in the data cube to have a value. When using a data model as part of a global search this is a best practice. In this dataset, the status field does not contain null values. So, you will create eval expressions to eliminate null values from three other fields: action, price, product_name.

☐ 10. Select **Add Field** > **Eval Expression**.

☐ 11. Add an eval expression for the action field:

- Eval Expression: `if(isnull(action),"NULL",action)`
- Field Name: `action1`
- Display Name: `action1`

☐ 12. Preview events to verify the field is working properly.

☐ 13. Click **Save**.

☐ 14. Create two more eval expression attributes for NULL values, using the same naming conventions, for these fields:

- price
- product_name

TIP: The price field type is Number.

Task 4: Test your data model.

IMPORTANT: Append your student number to the data model ID in the search. For example, bcg_ws_student7.

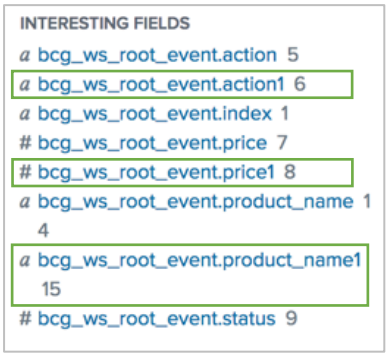
☐ 15. Using the datamodel command, search the data model's root event **over the last 24 hours**.
Tip: The datamodel command is a generating command. It should be preceded with a pipe.

```
| datamodel bcg_ws_student<x> bcg_ws_root_event search
```

☐ 16. Examine the values of the action*, price*, and product_name* fields.

Example on next page.

Example:



Questions

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

_____

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

_____

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

_____

Lab Exercise 2 – Add Interactivity

Description

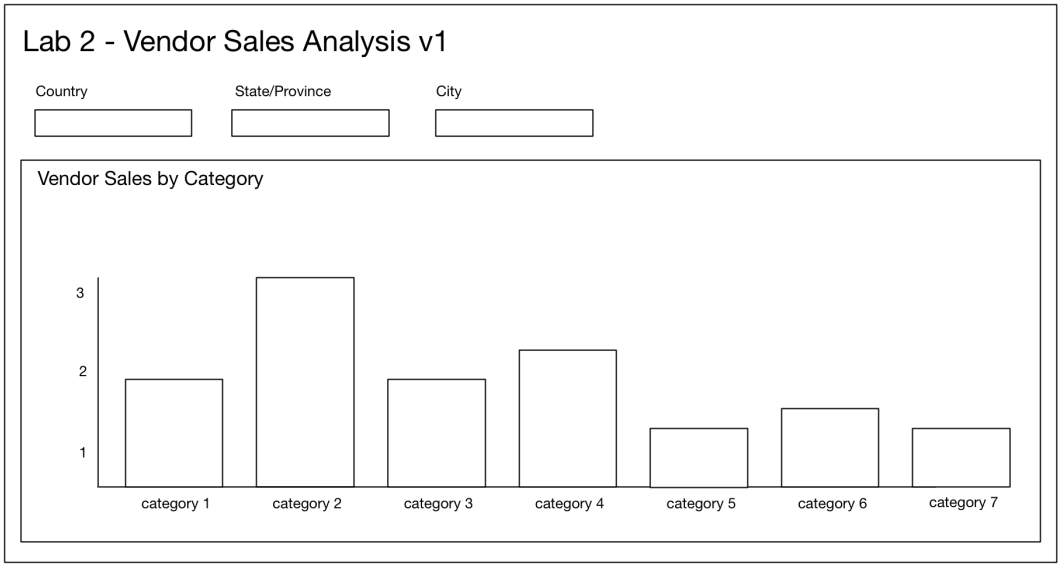
In this exercise you will use tokens with token filters to create cascading inputs that define a chart's search. The optional challenge exercise uses an event handler to unset input tokens forcing the menus to reset to their default. Event handlers are discussed in more detail in a later lab exercise.

IMPORTANT: Perform all searches and save all knowledge objects in the Creating Dashboards course app.

Scenario: **The sales team is impressed with the new server errors dashboard. They would like to add a form to their app. The form should display a column chart of vendor data based on user input for country, state or province, and city.**

Working Example: Creating Dashboards > Lab Examples > Lab 2 Example - Add Interactivity

Wireframe:



Steps

Task 1: Create a dashboard.

☐ 1. Create a new dashboard:

- Dashboard Title: Lab 2 - Vendor Sales Analysis v1
- Dashboard ID: lab_2__vendor_sales_analysis_v1
- Permissions: Private

☐ 2. Click **+Add Panel** > **New from Report** > **Show More**.

☐ 3. In the sidebar, click **Lab 2 - Search 1**.

☐ 4. Click **Add to Dashboard** and close the Add Panel sidebar.

NOTE: This panel's search includes tokens and will not display results until all three inputs are added. This is expected.

☐ 5. Remove the visualization title Lab 2 - Search 1

☐ 6. Add the panel title: `Vendor Sales by Category`

Task 2: Change the panel's search to be an inline search.

☐ 7. On the Vendor Sales by Category panel, click the **View report** icon.

☐ 8. Click **Clone to Inline**.

☐ 9. Click **Clone to Inline Search**.

Task 3: Add a token filter for quotes.

☐ 10. Click the panel's **Edit search** icon.

☐ 11. Notice each token has four delimiters (two on either side of the token).

NOTE: When saving a search, Splunk sees dollar signs ($) in the query as static text and automatically escapes those characters by adding a delimiter ($).

☐ 12. Revise each of the tokens to have **one delimiter and a filter** to wrap the value in quotes.

```
index=sales sourcetype=vendor_sales VendorCountry=$$v_country_tok|s$$
VendorStateProvince=$$v_state_tok|s$$ VendorCity=$$v_city_tok|s$$
| stats count by categoryId
```

IMPORTANT: Make sure the search string is as shown above: include space between the trailing token delimiter and the next field name and leave no spaces around the filter.

☐ 13. Click **Apply**.

Task 4: Add a menu input for country.

NOTE: The inputs will not populate **until after you've added all three** inputs. Remember to set the default to All.

☐ 14. Click **+Add Input** > **Dropdown**.

☐ 15. Click the **Edit Input** icon (pencil) and add the following settings:

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

Token Options
- Token: `v_country_tok`

Static Options
- Name: All
- Value: *

Token Options
- Default: All

Dynamic Options

- Search String:
```
| inputlookup bcg_vendors
| search VendorStateProvince=$v_state_tok|s$
   VendorCity=$v_city_tok|s$
| dedup VendorCountry
| fields VendorCountry
| sort VendorCountry
```
- Time Input: Last 24 hours
- Field For Label: `VendorCountry`
- Field For Value: `VendorCountry`

☐ 16. Click **Apply**.

Task 5: Add a menu input for state/province.

NOTE: The input will not populate **until after you've added all three** inputs. Remember to set the default to All.

☐ 17. Click **+Add Input** > **Dropdown**.

☐ 18. Click the **Edit Input** icon (pencil) and add the following settings:

General

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

Token Options

- Token: `v_state_tok`

Static Options

- Name: All
- Value: *

Token Options

- Default: All

Dynamic Options

- Search String:
```
| inputlookup bcg_vendors
| search VendorCountry=$v_country_tok|s$
   VendorCity=$v_city_tok|s$
| dedup VendorStateProvince
| fields VendorStateProvince
| sort VendorStateProvince
```
- Time Input: Last 24 hours
- Field for Label: `VendorStateProvince`
- Field for Value: `VendorStateProvince`

☐ 19. Click **Apply**.

Task 6: Add a menu input for City.

NOTE: Remember to set the default to All.

☐ 20. Click **+Add Input** > **Dropdown**.

☐ 21. Click the **Edit Input** icon (pencil) and add the following settings:

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

Token Options
- Token box: `v_city_tok`

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

Token Options
- Default: All

Dynamic Options
- Search String:
```
| inputlookup bcg_vendors
| search VendorCountry=$v_country_tok|s$
  VendorStateProvince=$v_state_tok|s$
| dedup VendorCity
| fields VendorCity | sort VendorCity
```
- Time Input: Last 24 hours
- Field for Label: `VendorCity`
- Field for Value: `VendorCity`

☐ 22. Click **Apply**.

☐ 23. Save the dashboard and **reload your browser**.

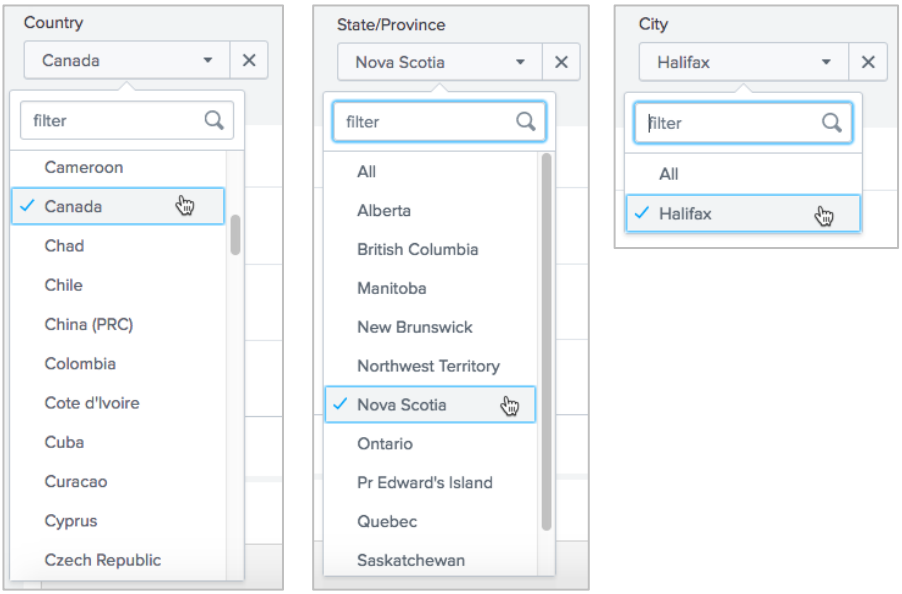
Task 7: Test the form inputs.

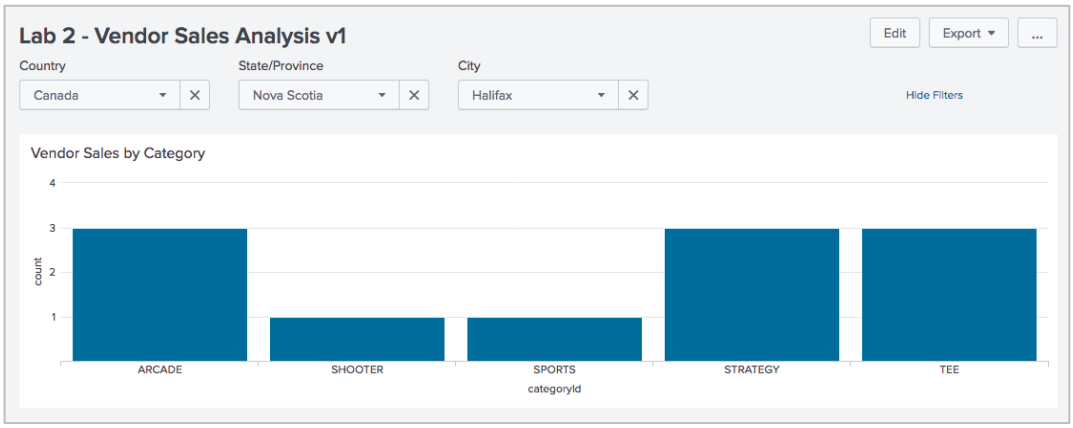
☐ 24. Select a Country, State/Province, and City.

- Form input(s) not filtering properly? Look at the settings for typos. Check the XML, where errors may be more obvious.
- No results are found in the chart? Try choosing a different country. Not every Country, State/Province, City combination will provide data to populate the chart.

Example on next page.

Example:


Country
Canada
filter
Cameroon
Canada
Chad
Chile
China (PRC)
Colombia
Cote d'Ivoire
Cuba
Curacao
Cyprus
Czech Republic
State/Province
Nova Scotia
filter
All
Alberta
British Columbia
Manitoba
New Brunswick
Northwest Territory
Nova Scotia
Ontario
Pr Edward's Island
Quebec
Saskatchewan
City
Halifax
filter
All
Halifax


Lab 2 - Vendor Sales Analysis v1
Edit
Export
...
Country
Canada
State/Province
Nova Scotia
City
Halifax
Hide Filters
Vendor Sales by Category
4
3
2
1
count
ARCADE
SHOOTER
SPORTS
STRATEGY
TEE
categoryId

splunk>

Challenge Lab Exercise (optional)

Use tokens to reset cascading inputs

In this exercise you will use an event handler to unset input tokens making the cascading menus reset. Event handlers are discussed in detail in module 6 of this course.

Task 1: Add a radio input with an event handler to reset the menus.

☐ 1. Open the dashboard editor.

☐ 2. Click **+Add Input** > **Radio**.

☐ 3. Click the **Edit Input icon** (pencil) and add the following settings:

General

- Label: *Delete field1 and leave blank*
- Search on Change: ☑

Token Options

- Token: `reset_menus_tok`

Static Options

- Name: Reset Menus
- Value: Boom

Token Options

- Default: Reset Menus

☐ 4. Click **Apply**.

☐ 5. Open the XML editor.

a. Click **Edit** > **Source**.

☐ 6. Locate the XML for the radio input.

☐ 7. Add an event handler to unset the state and city input tokens between the closing </default> and closing </input> tags.

```
...
 <input type="radio" token="reset_menus_tok" searchWhenChanged="false">
   <choice value="Boom">Reset Menus</choice>
   <default>Boom</default>
      <change>
        <unset token="form.v_country_tok"></unset>
        <unset token="form.v_state_tok"></unset>
        <unset token="form.v_city_tok"></unset>
        <set token="form.reset_menus_tok">No</set>
      </change>
 </input>
...
```

☐ 8. Save the XML changes and reload your browser.

☐ 9. Select a country, state/province and city.

☐ 10. Click the Reset Menus input and make sure the menus and panel reset.

Lab Exercise 3 – Improve Performance

Description

In this exercise you will improve dashboard performance by first creating and scheduling reports. Then, by creating a global search and finally by using an accelerated data model.

IMPORTANT: If you copy text from this document, please note that character formatting and artifacts created by the PDF generation process can cause errors in the XML. Consider using a text editor as an interim step.

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

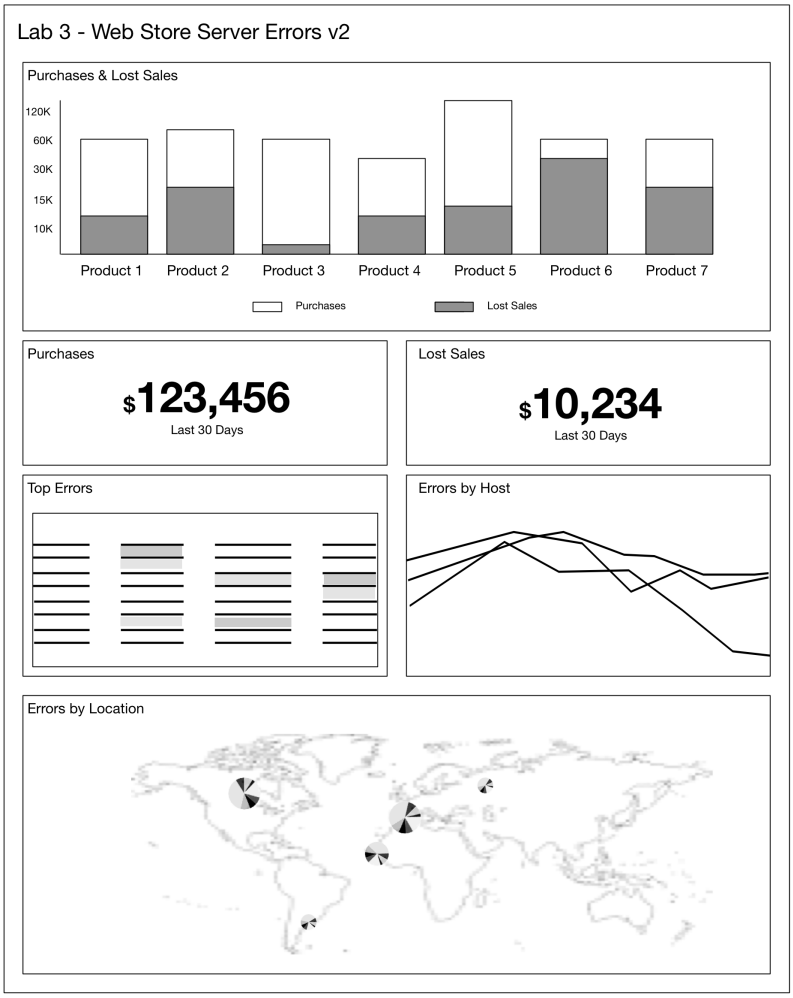
- Make the dashboard load faster.

- Reduce the search load from this dashboard.

- Add panels to show the dollar amount of purchases and lost sales.

Working Example: Creating Dashboards > Lab Examples > Lab 3 Example

Wireframe:



Steps

Task 1: Schedule your reports.

- [ ] 1. Navigate to **Settings** > **Searches, reports, and alerts.**
- [ ] 2. Make sure the App context is set to: **Creating Dashboards (createdash)**
- [ ] 3. In the Owner dropdown, select your name.
- [ ] 4. Locate **L1_ws_all_errors_by_host.**
- [ ] 5. In the Actions column select: **Edit** > **Edit Schedule**
- [ ] 6. Select: **Schedule Report**
- [ ] 7. Set schedule for:
    - Schedule: Run every week
    - On: Sunday at: 0:00
    - Time Range: Presets > Last 30 Days
    - Schedule Window: 30 minutes
- [ ] 8. Click **Save**.
- [ ] 9. Repeat the steps above for the remaining reports:
    - L1_ws_common_errors_by_host
    - L1_ws_purch_vs_lost_sales
    - L1_ws_status_errors_by_location

Task 2: Accelerate your reports.

- [ ] 10. Locate **L1_ws_all_errors_by_host.**
- [ ] 11. In the Actions column select **Edit** > **Edit Acceleration**
- [ ] 12. Select: **Accelerate Report**
- [ ] 13. Set summary range to: **1 Month**
- [ ] 14. Click **Save.**
- [ ] 15. Repeat the steps above for the following reports:
    - L1_ws_common_errors_by_host
    - L1_ws_purch_vs_lost_sales
- [ ] 16. Why is acceleration not possible for *L1_ws_status_errors_by_location*?

    _____

- [ ] 17. What, if any, change(s) would be required to accelerate it?

    _____

Task 3: Create a dashboard.

☐ 18. Navigate to the Creating Dashboards app and create a new dashboard:

- Dashboard Title: `Lab 3 - Web Store Server Errors v2`
- Dashboard ID: `lab_3__web_store_server_errors_v2`
- Permissions: Private

☐ 19. Add a new panel from a report: Lab 3 - Search 1.

☐ 20. Clone the new panel's search to inline.

Task 4: Name the base search.

☐ 21. Open the XML editor.

☐ 22. Locate the panel's search query.

☐ 23. Delete the opening and closing <row>, <panel>, <table> and <title> elements and all the options.

IMPORTANT: Do not delete the search query.

```
<dashboard>
  <label>Lab 3 - Web Store Server Errors v2</label>
  <row>
    <panel>
     <table>
      <title>Lab 3 - Search 1</title>
        <search>
         <query>| tstats summariesonly=f count from
          datamodel="bcg_ws_noxl" by _time, host, sourcetype,
          bcg_ws_root_event.status, bcg_ws_root_event.action1,
          bcg_ws_root_event.product_name1,
          bcg_ws_root_event.price1 span=1s | rename
          bcg_ws_root_event.status as status,
          bcg_ws_root_event.action1 as action,
          bcg_ws_root_event.product_name1 as product_name,
          bcg_ws_root_event.price1 as price</query>
         <earliest>-30d@d</earliest>
         <latest>now</latest>
         <sampleRatio>1</sampleRatio>
        </search>
        <option name="count">10</option>
        <option name="dataOverlayMode">none</option>
        <option name="drilldown">none</option>
        <option name="percentagesRow">false</option>
        <option name="rowNumbers">false</option>
        <option name="totalsRow">false</option>
        <option name="wrap">true</option>
      </table>
    </panel>
  </row>
</dashboard>
```

☐ 24. Add an ID to the opening <search> element: `<search id="baseSearch">`

```
<dashboard>
 <label>Lab 3 - Web Store Server Errors v2</label>
    <search id="baseSearch">
     <query>| tstats summariesonly=f count from
       datamodel="bcg_ws_noxl" by _time, host, sourcetype,
       bcg_ws_root_event.status, bcg_ws_root_event.action1,
       bcg_ws_root_event.product_name1, bcg_ws_root_event.price1
     ...
```

☐ 25. Save the XML changes.

Task 5: Add a chart that uses a post-process search.

☐ 26. Open the dashboard editor.

☐ 27. Add a new panel from a report: Lab 3 - Search 2

☐ 28. Clone the new panel's search to inline.

☐ 29. Remove the visualization title Lab 3 - Search 2

☐ 30. Add the panel title: `Purchases & Lost Sales`

☐ 31. Switch to the XML editor.

☐ 32. Locate the XML for the Purchases & Lost Sales panel.

☐ 33. Add an ID to the opening <search> element: `<search base="baseSearch">`

☐ 34. At the beginning of the query add the **`search`** command.

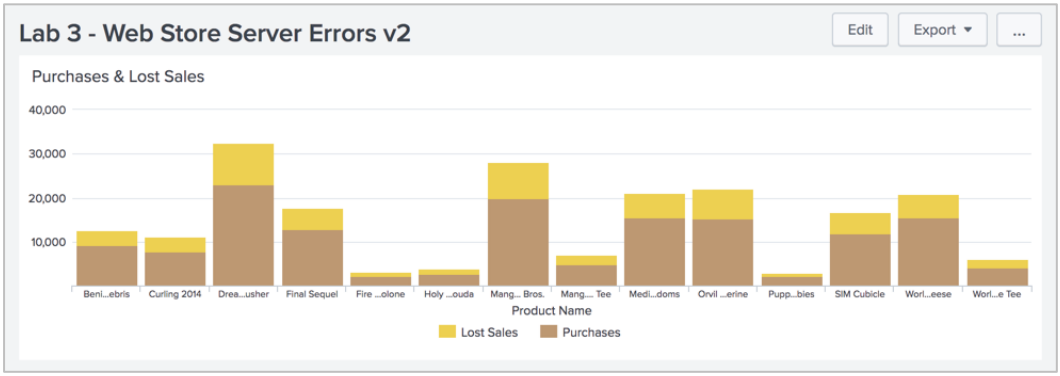
```
...
<panel>
 <title>Purchases &amp;amp; Lost Sales</title>
 <chart>
  <search base="baseSearch">
  <query>search (action=purchase OR action=remove) | stats sum(price)
as Sales by product_name, action | search Sales &amp;gt; 50 | xyseries
product_name, action, Sales | rename product_name as "Product Name",
remove as "Lost Sales", purchase as Purchases</query>
...
```

☐ 35. Delete the Purchases & Lost Sales panel's earliest, latest, and sampleRatio elements.

```
...
</query>
  <earliest>-30d@d</earliest>
  <latest>now</latest>
  <sampleRatio>1</sampleRatio>
 </search>
...
```

☐ 36. Save the XML changes.

Example:



Task 6: Add a single value panel and a post-process search.

- ☐ 37. Open the dashboard editor.
- ☐ 38. Add a new panel from a report: Lab 3 - Search 3
- ☐ 39. Clone the new panel's search to inline.
- ☐ 40. Remove the visualization title: Lab 3 - Search 3
- ☐ 41. Add the panel title: `Purchases`
- ☐ 42. Open the XML editor.
- ☐ 43. Locate the XML for the Purchases panel.
- ☐ 44. Add an ID to the opening <search> element: `<search base="baseSearch">`
- ☐ 45. At the beginning of the query add the **`search`** command.

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

- ☐ 46. Delete the Purchases panel's earliest, latest, and sampleRatio elements.

```
...
</query>
 <earliest>-30d@d</earliest>
 <latest>now</latest>
 <sampleRatio>1</sampleRatio>
</search>
...
```

- ☐ 47. Save the XML changes.

Task 7: Add a second single value panel and a post-process search.

- ☐ 48. Open the dashboard editor.
- ☐ 49. Add a new panel from a report: Lab 3 - Search 4
- ☐ 50. Clone the new panel's search to inline.
- ☐ 51. Remove the visualization title: Lab 3 - Search 4
- ☐ 52. Add the panel title: `Lost Sales`
- ☐ 53. Position the Lost Sales panel on the right of the Purchases panel.
- ☐ 54. Open the XML editor.
- ☐ 55. Locate the XML for the Lost Sales panel.
- ☐ 56. Add an ID to the opening <search> element: `<search base="baseSearch">`
- ☐ 57. Add the **`search`** command to the beginning of the query.

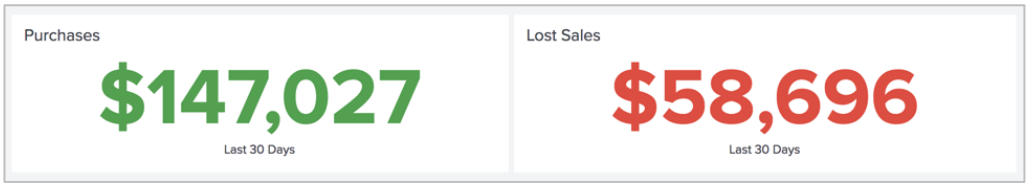
```
...
  <title>Lost Sales</title>
   <single>
   <search base="baseSearch">
      <query>search action=remove
     | stats sum(price) as LostSales</query>
...
```

- ☐ 58. Delete the Lost Sales panel's earliest, latest, and sampleRatio elements.

```
...
</query>
 <earliest>-30d@d</earliest>
 <latest>now</latest>
 <sampleRatio>1</sampleRatio>
</search>
...
```

- ☐ 59. Save the XML changes.

Example:



Task 8: Add prebuilt panels.

- ☐ 60. Open the dashboard editor.
- ☐ 61. Add three prebuilt panels:
  - Most Common Errors by Host
  - All Errors by Host
  - All Status Errors by Location
- ☐ 62. Position the All Errors by Host panel to the right of Most Common Errors by Host on a row above the All Status Errors by Location panel.

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

- [ ] 63. On the Most Common Errors by Host panel, click the **Options** icon. 
- [ ] 64. Click **Convert to Inline Panel**.
- [ ] 65. Click **Convert**.
- [ ] 66. Click the panel's **View report** icon. 
- [ ] 67. Click **Clone to Inline**.
- [ ] 68. Click **Clone to Inline Search**.
- [ ] 69. Repeat the above steps for the remaining prebuilt panels:
    - All Errors by Host
    - All Status Errors by Location
- [ ] 70. Convert the three new panels to *inline panels* driven by *inline searches*.

Task 9: Revise two searches to be post-process.

- [ ] 71. Open the XML editor.
- [ ] 72. Locate the XML for the Most Common Errors by Host panel.
- [ ] 73. Add an ID to the opening <search> element: `<search base="baseSearch">`
- [ ] 74. Add the **`search`** command to the beginning of the query
- [ ] 75. Delete the index and sourcetype references from the query.

NOTE: The index and sourcetype are contained in the datamodel which is referenced in the base search. Thus, they are not needed in the post-process.

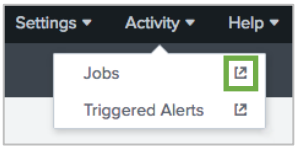
```
...
<panel>
 <title>Most Common Errors by Host</title>
  <table>
    <search base="baseSearch">
      <query>search index=web sourcetype=access_combined
      status>399 | chart count by host, status limit=3
      useother=f</query>
...
```

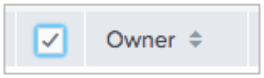
- [ ] 76. Remove the earliest, latest, and sample ratio elements.

```
...
</query>
  <earliest>-30d@h</earliest>
  <latest>now</latest>
  <sampleRatio>1</sampleRatio>
</search>
...
```

- [ ] 77. Locate the XML for the **All Errors by Host** panel and repeat the above steps.
- [ ] 78. Save the XML changes and make sure all panels populate with data.

Task 10: Clear search jobs.

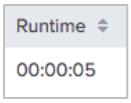




- [ ] 79. Select **Activity** > **Jobs**.
- [ ] 80. Make sure the App context is set to: Creating Dashboards
- [ ] 81. Make sure the Owner is set to your name.
- [ ] 82. Delete your search history.
    a. Select the box beside **Owner**.
    b. Click **Edit Selected** > **Delete**.
    c. Click **Delete** to confirm.
    d. Repeat the above steps until all jobs have been deleted.

Task 11: Accelerate the global search.

- [ ] 83. Return to the Lab 3 - Web Store Server Errors v2 dashboard and reload your browser.
- [ ] 84. Wait for all the panels to populate.
- [ ] 85. Return to the **Jobs** page and reload your browser.
- [ ] 86. Make sure the App context is set to: **Creating Dashboards**
- [ ] 87. Make sure the Owner is set to your name.
- [ ] 88. Note the runtime for the base search.



_____

NOTE: The base search begins with: | tstats summariesonly=f count from datamodel...
As noted earlier, this search uses the non-accelerated data model with tstats summariesonly=f. In an upcoming step you will revise this search to use an accelerated data model with tstats summaries=t and then compare the runtimes.

- [ ] 89. Return to the Lab 3 - Web Store Server Errors v2 dashboard.
- [ ] 90. Open the XML editor.
- [ ] 91. Update the base search's query to use the tstats argument **summariesonly=t** and the **bcg_ws_xl** accelerated data model.

```
...
<label>Lab 3 Web Store Server Errors v2</label>
<search id="baseSearch">
<query>| tstats summariesonly=t count from
 datamodel="bcg_ws_xl" by _time, host, sourcetype,
...
```

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.

- [ ] 92. Save the XML changes.
- [ ] 93. Wait for all the panels to populate.
- [ ] 94. Return to the **Jobs** page and reload your browser.

☐ 95. Make sure the App context is set to: Creating Dashboards

☐ 96. Make sure the Owner is set to your name.

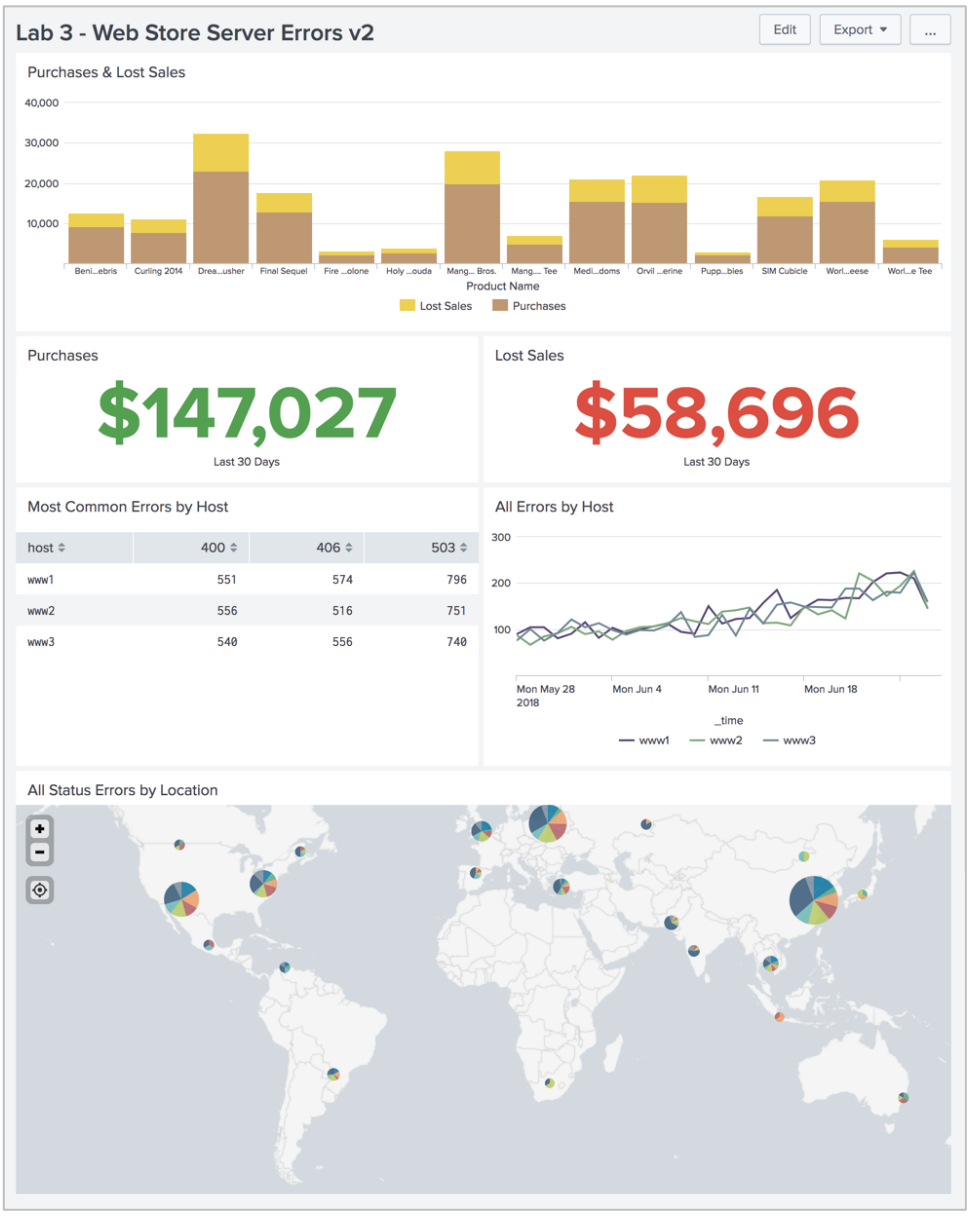
☐ 97. Note the run time for the revised base search.

_____

NOTE: The revised base search begins with: | tstats summariesonly=t count from datamodel="bcg_ws_xl"

☐ 98. Compare the two search runtimes. Consider *percentage* of difference in the times.

Example:



Challenge Lab Exercise (optional)

Use Your Accelerated Data Model

This exercise walks you through the process of accelerating the data model you created in the lab 1 challenge exercise. If you have not already completed it, go back and finish. Then, return here.

IMPORTANT: Accelerating a data model requires administrator-level access. For the purposes of this exercise you have been given this particular privilege.

Data model acceleration should be reserved for data models that are heavily used. Summaries take up space, and sometimes a significant amount of it, so it's important to avoid overuse of data model acceleration.

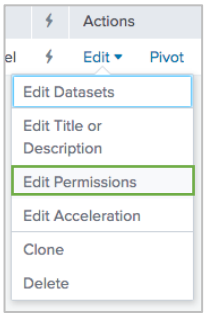
After you enable acceleration for a data model, Splunk begins building summaries that span the range you've specified. It builds them in indexes with events that contain the fields specified in the data model.

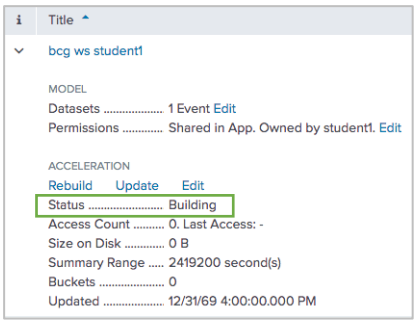
Splunk runs a search every 5 minutes to update existing summaries. It runs a maintenance process every 30 minutes to remove outdated summaries. These settings can be adjusted by a Splunk Administrator.

Task 1: Accelerate your data model.

NOTE: The acceleration requires ~15-20 minutes to complete. You may want to wait until the next lab exercise before testing the accelerated dashboard.

- ☐ 1. Navigate to: **Settings** > **Data Models**.
- ☐ 2. Make sure the App context is set to: Creating Dashboards
- ☐ 3. Make sure the Owner is set to your name.
- ☐ 4. Locate the data model you created in the lab 1 challenge exercise, i.e. bcg ws student<X>
- ☐ 5. In the Actions column, click **Edit** > **Edit Permissions**.
- ☐ 6. Select **Display for App**.
- ☐ 7. Select **Read** for Everyone.
- ☐ 8. Select **Write** for power.
- ☐ 9. Click **Save**.
- ☐ 10. In the Actions column, click **Edit** > **Edit Acceleration**.
- ☐ 11. In the Edit Acceleration dialog, click **Accelerate**.
- ☐ 12. In the Summary Range dropdown, select **7 Days**.
- ☐ 13. Click **Save**.
- ☐ 14. Click the **arrow** beside the data model's name to see status, size, and other statistics.



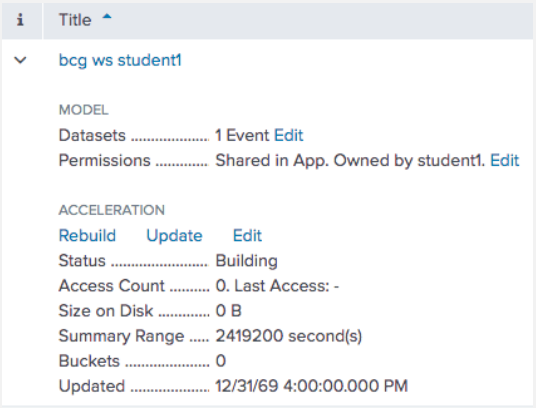


IMPORTANT: The acceleration requires ~15-20 minutes to complete. **You may want to wait until the next lab exercise before testing the accelerated dashboard**.

After saving the XML, if your panels show the message: “No results found,” verify Acceleration Status displays: *100% Completed*.

If your panels show the message "Error in "TsidxStats': Could not find datamodel," look for typos in the XML.

Compare the spelling of the data model ID in the XML to the actual data model ID spelling.



Task 2: Clear search jobs.

- ☐ 15. Open the **Jobs** page.
- ☐ 16. Make sure the App context is set to: **Creating Dashboards**
- ☐ 17. Make sure the Owner is set to your name.
- ☐ 18. Select all searches and click **Delete**.
- ☐ 19. Click **Delete** to confirm.
- ☐ 20. Repeat the above steps until all jobs have been deleted.

Task 3: Revise the base search.

- ☐ 21. Return to the Lab 3 - Web Store Server Errors v2 dashboard.
- ☐ 22. Open the XML editor.
- ☐ 23. Update the global search's query to use your data model by changing the id referenced to: bcg_ws_student<X>. For example, bcg_ws**_student7**

```
...
 <label>Lab 3 - Web Store Server Errors v2</label>
 <search id="baseSearch">
    <query>| tstats summariesonly=t count from
      datamodel="bcg_ws_student<x>" by _time, host, sourcetype,
...
```

- ☐ 24. Save the XML changes.
- ☐ 25. Wait for all the panels to populate.
- ☐ 26. Return to the search jobs page and make a note of the runtime for the base search.

_____

- ☐ 27. Compare the search time to the base search runtime that used un-accelerated data model. Consider *percentage* of difference in the times.

Lab Exercise 4 – Customizing Dashboards

Description

Once you have a high performance dashboard that displays the data your stakeholders require, make user interface customizations.

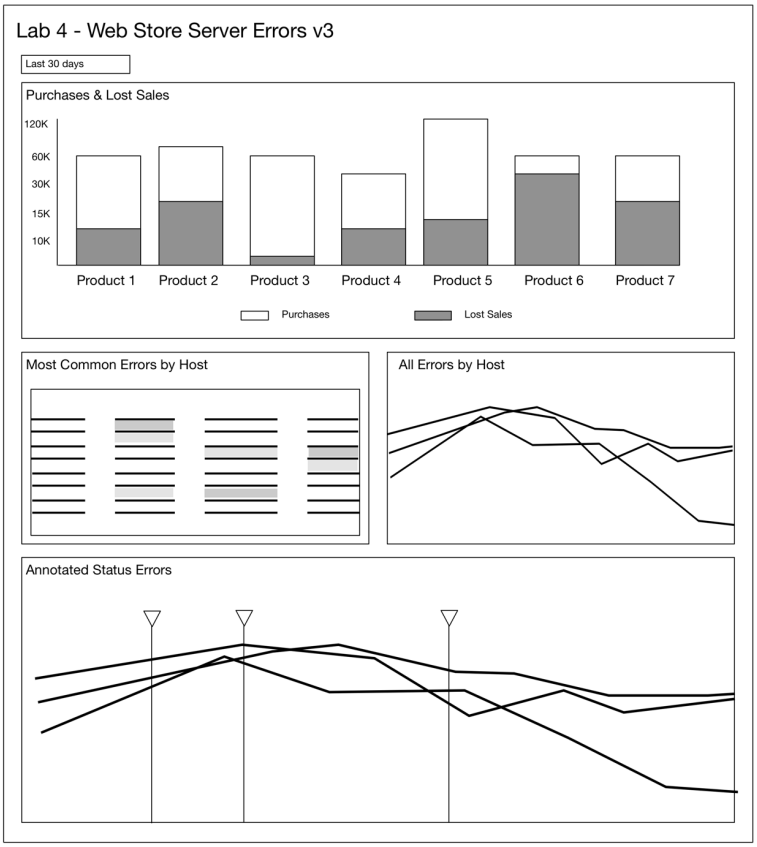
IMPORTANT: If you copy text from this document, please note that character formatting and artifacts created by the PDF generation process can cause errors in the XML. Consider using a text editor as an interim step.

Scenario: **The stakeholders have approved the higher-performance dashboard. Now they'd like a few customizations, including:**

- Remove the map and single value panels
- Add an input for time range
- Add a status errors chart annotated with new release and sale dates
- Hide the search controls for all panels
- Set the refresh time for all the panels
- Use the company's brand colors for the chart of purchases and lost sales
- Change the server table's cell colors to match severity of the value shown

Working Example: Creating Dashboards > Lab Examples > Lab 4 Example

Wireframe:



Steps

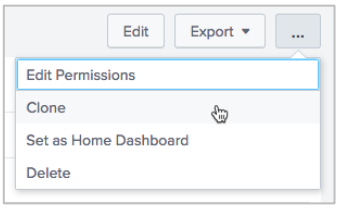
Task 1: Clone a dashboard.

☐ 1. Navigate to the Lab 3 - Web Store Server Errors v2 dashboard.

☐ 2. Clone the dashboard and rename it:

- Title: `Lab 4 - Web Store Server Errors v3`
  Note: Remove the word Clone from the end of the title.
- ID: lab_4__web_store_server_errors_v3
- Permissions: Private

a. Select: **...** > **Clone**.
b. Enter title and ID.
c. Click **Clone Dashboard**.
d. Click **View**.



☐ 3. Open the dashboard editor.

☐ 4. Delete three panels:

- Purchases
- Lost Sales
- All Status Errors by Location

a. Click the ☒ in the panel's upper right corner.

Task 2: Add a time input.

☐ 5. Add a time input and use the following settings:

General
- Label: *leave blank*
- Search on Change: ☑

Token Options
- Token: `time_tok`
- Default: Last 30 Days

☐ 6. Open the XML editor.

☐ 7. Locate the XML for the base search.

☐ 8. Revise the base search to use the time input.

```
...
bcg_ws_root_event.price1 span=1s | rename bcg_ws_root_event.status as
status, bcg_ws_root_event.action1 as action,
bcg_ws_root_event.product_name1 as product_name, bcg_ws_root_event.price1
as price</query>
    <earliest>$time_tok.earliest$</earliest>
    <latest>$time_tok.latest$</latest>
    <sampleRatio>1</sampleRatio>
  </search>
...
```

Task 3: Set the panel refresh time.

☐ 9. Set the base search to refresh every **5 minutes**.

```
...
 <earliest>$time_tok.earliest$</earliest>
 <latest>$time_tok.latest$</latest>
 <sampleRatio>1</sampleRatio>
 <refresh>5m</refresh>
</search>
<row>
...
```

NOTE: Panels using post process searches will now automatically refresh when the base search is refreshed.

Task 4: Customize chart colors.

☐ 10. Locate the option elements for the Purchases & Lost Sales panel.

☐ 11. Add a charting.fieldColors property to set the Purchases to orange (0xEFC94C) and the Lost Sales field to red (0xED553B).

```
...
  <option name="trellis.scales.shared">1</option>
  <option name="trellis.size">medium</option>
  <option name="charting.fieldColors">{"Purchases":0xEFC94C,"Lost
  Sales":0xED553B}</option>
  </chart>
 </panel>
</row>
...
```

Task 5: Hide search controls.

☐ 12. Add chart properties to hide the Inspect, Export and Open Search panel buttons:

- Inspect button link.inspectSearch.visible
- Export Results button link.exportResults.visible
- Open Search button link.openSearch.visible

```
...
 <option name="charting.fieldColors">{"Purchases":0xEFC94C,"Lost
  Sales":0xED553B}</option>
 <option name="link.inspectSearch.visible">0</option>
 <option name="link.exportResults.visible">0</option>
 <option name="link.openSearch.visible">0</option>
 </chart>
...
```

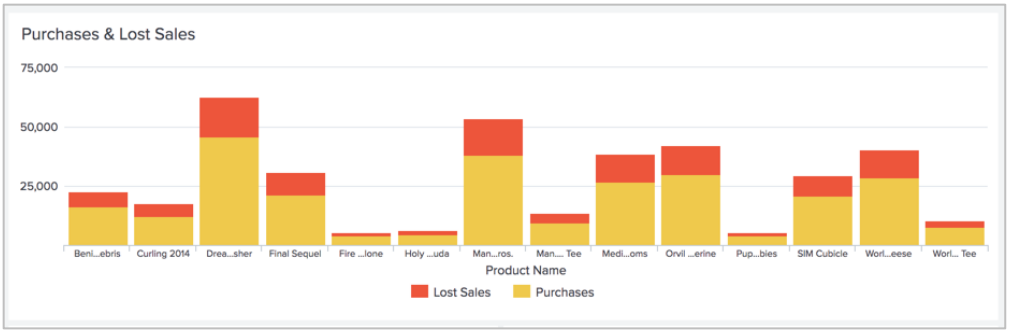
☐ 13. Add these chart properties to the remaining panels option elements:

- Most Common Errors by Host
- All Errors by Host

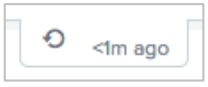
☐ 14. Save the XML changes.

☐ 15. Make sure the Purchases & Lost Sales panel's colors have changed to orange and red.
If not, go back to the XML editor and look for typos in the charting.fieldColors property setting.

Example:



☐ 16. Make sure each panel displays only the refresh button and refresh time.



Task 6: Add a prebuilt panel.

☐ 17. Add a prebuilt panel: Lab 4 - Annotated Status Errors

☐ 18. Convert the panel to inline.

☐ 19. Remove **Lab 4 –** from the panel's title.

Task 7: Revise search to be post-process.

☐ 20. Open the XML editor.

☐ 21. Locate the XML for the Annotated Status Errors panel.

☐ 22. Add an ID to the opening <search> element: `<search base="baseSearch">`

☐ 23. Add the **search** command to the beginning of the query.

```
...
<panel>
 <title>Annotated Status Errors</title>
  <chart>
  <search base="baseSearch">
   <query>search sourcetype=access_combined status!=200
    | timechart count by action
   </query>
  </search>
  <option name="charting.chart">area</option>
  <option name="charting.drilldown">all</option>
  <option name="charting.legend.placement">bottom</option>
  <option name="refresh.display">none</option>
  </chart>
</panel>
...
```

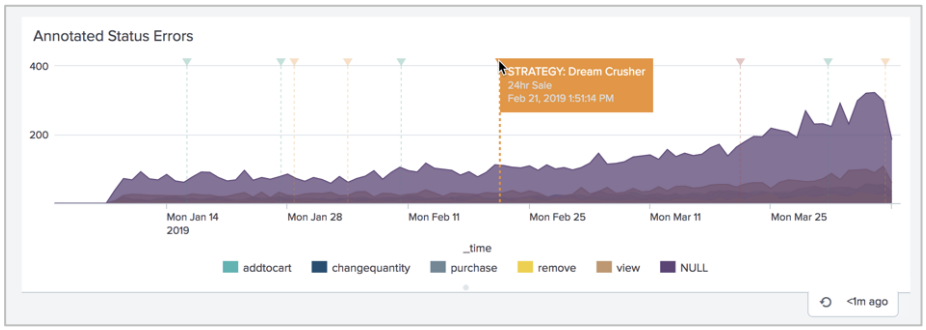
☐ 24. Add an annotation search.

```
...
<panel>
 <title>Annotated Status Errors</title>
  <chart>
  <search type="annotation">
   <query>index="sales" sourcetype="bcg_sale_dates"
   | eval annotation_category=message
   | eval annotation_label=Sale_Category
   </query>
  <earliest>$time_tok.earliest$</earliest>
  <latest>$time_tok.latest$</latest>
 </search>
  <search base="baseSearch">
   <query>search sourcetype=access_combined status!=200
    | timechart count by action
   </query>
  </search>
  <option name="charting.chart">area</option>
  <option name="charting.drilldown">all</option>
  <option name="charting.legend.placement">bottom</option>
  <option name="refresh.display">none</option>
  </chart>
 </panel>
...
```

☐ 25. Save the XML changes.

☐ 26. Test the annotated panel by moving your mouse over one of the markers to see the message.

Example:



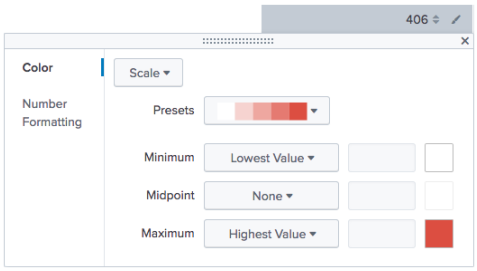
Task 8: Add column summaries and cell colors.

☐ 27. Open to the dashboard editor.

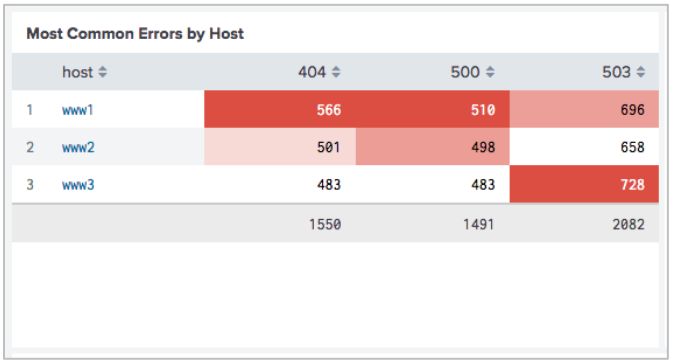
☐ 28. On the Most Common Errors by Host visualization, click the **Format visualization** icon.

☐ 29. Click the **Summary tab**.

- [ ] 30. Select **Totals: Yes** and close the window.
- [ ] 31. Click the **paintbrush** icon for the first status column.
- [ ] 32. In the colors dropdown, select **Scale**.
- [ ] 33. In the Presets dropdown, select the **Sequential**, white to red.
- [ ] 34. Repeat the above steps for the two remaining status columns.
- [ ] 35. Save the dashboard changes.



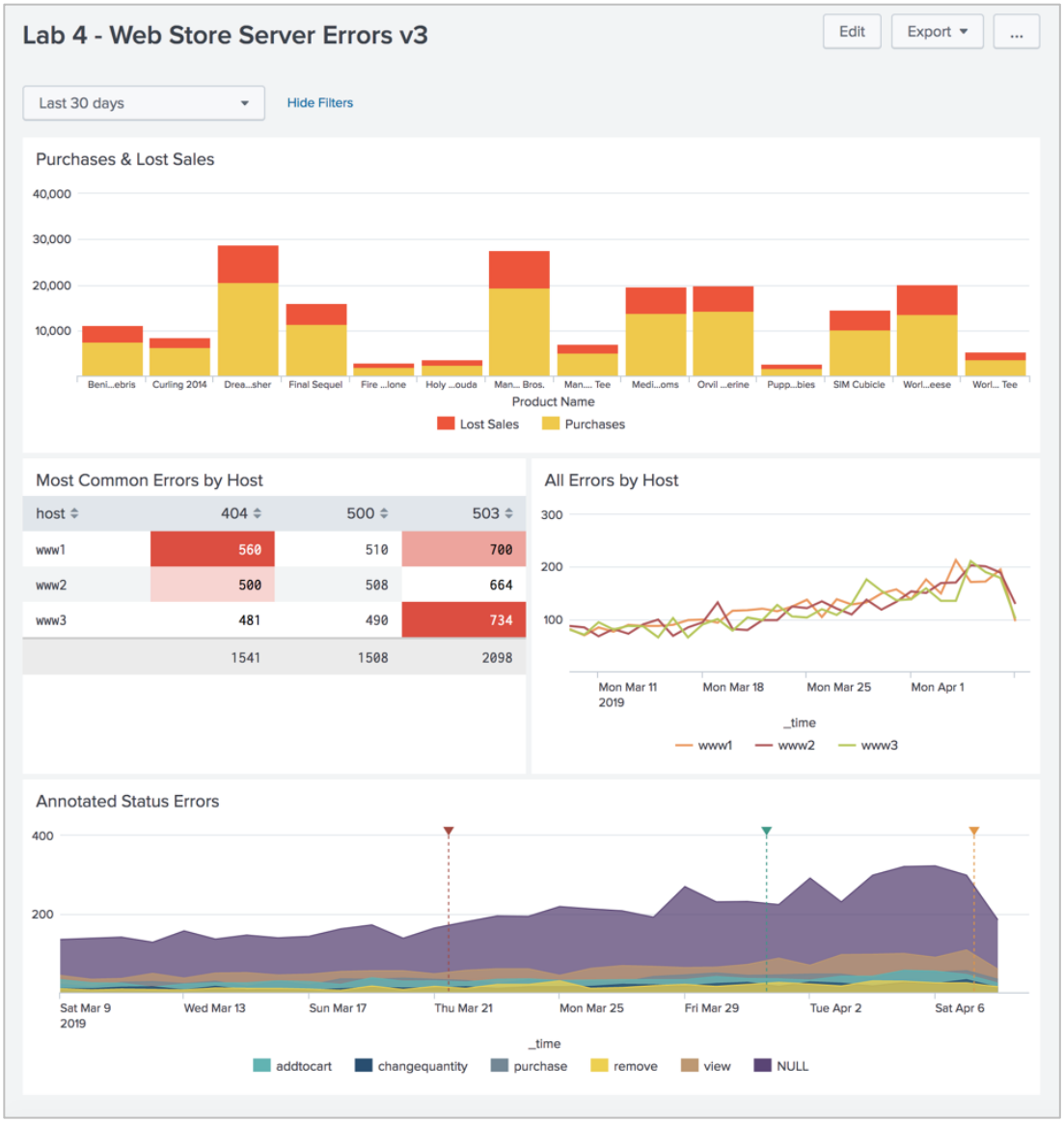
Example:



Most Common Errors by Host

	host	404	500	503
1	www1	566	510	696
2	www2	501	498	658
3	www3	483	483	728
		1550	1491	2082

Example:


Lab 4 - Web Store Server Errors v3
Edit
Export
...
Last 30 days
Hide Filters
Purchases & Lost Sales
40,000
30,000
20,000
10,000
Beni...ebris
Curling 2014
Drea...sher
Final Sequel
Fire ...lone
Holy ...ouda
Man... Bros.
Man.... Tee
Medi...oms
Orvil ...erine
Pupp...bies
SIM Cubicle
Worl...eese
Worl... Tee
Product Name
Lost Sales
Purchases
Most Common Errors by Host
host
404
500
503
www1
560
510
700
www2
500
508
664
www3
481
490
734
1541
1508
2098
All Errors by Host
300
200
100
Mon Mar 11
2019
Mon Mar 18
Mon Mar 25
Mon Apr 1
_time
www1
www2
www3
Annotated Status Errors
400
200
Sat Mar 9
2019
Wed Mar 13
Sun Mar 17
Thu Mar 21
Mon Mar 25
Fri Mar 29
Tue Apr 2
Sat Apr 6
_time
addtocart
changequantity
purchase
remove
view
NULL

Lab Exercise 5: Add a Dynamic Drilldown

Description

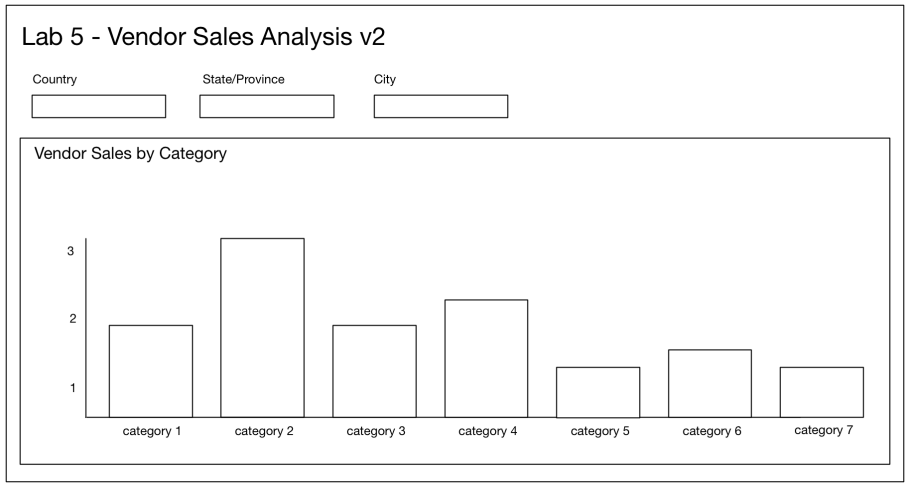
In this exercise you will add a dynamic drilldown from the Vendor Sales by Category chart to a new form.

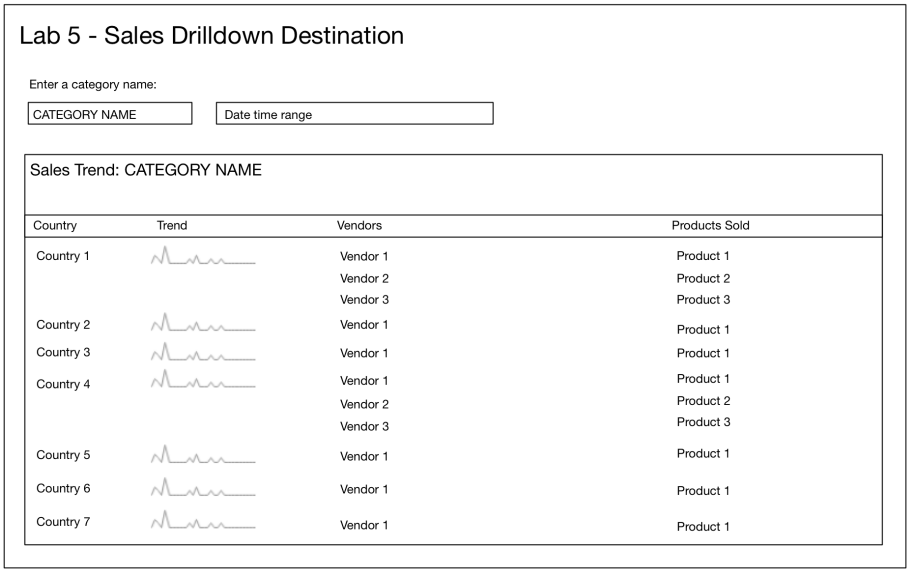
Scenario: **The sales team wants to add a dynamic drilldown to the sales by category bar chart that takes users to a different form displaying information based on which category in the chart was clicked.**

IMPORTANT: If you copy text from this document please note that character formatting and artifacts created by the PDF generation process can cause errors in the XML.

Working Example: Creating Dashboards > Lab Examples > Lab 5 Example – Add a Dynamic Drilldown

Wireframe:





Task 1: Create a drilldown destination form.

☐ 1. Create a new dashboard:

- Dashboard Title: `Lab 5 - Sales Drilldown Destination`
- Dashboard ID: lab_5__sales_drilldown_destination
  Important: Use this exact dashboard ID.
- Permissions: Private

☐ 2. Add a new panel from a report: L5_bcg_category_trends

NOTE: The dashboard will not display results until inputs are added in the next series of tasks.

☐ 3. Clone the new panel's search to inline.

☐ 4. Add the panel title: `Sales Trend: $categoryId_tok$`

☐ 5. Delete the visualization title: ~~`L5_bcg_category_trends`~~

Task 2: Add a text input.

☐ 6. Add a **text input** and use the following settings:

General

- Label: `Enter a category name:`
- Search on Change: ☑
- Token: `categoryId_tok`
  Note: There is a capital "I" in categoryId
- Default: `*`

☐ 7. Click **Apply**.

Task 3: Add a time input.

☐ 8. Add a **time input** and use the following settings:

Token Options

- Search on Change: ☑
- Token: *Delete the default token. Leave the box empty.*
- Default: `Last 7 Days`

☐ 9. Click **Apply**.

☐ 10. Click the **Edit search** icon on the Sales Trend panel.

☐ 11. Revise the categoryId token to have **one delimiter and a filter** to wrap the value in quotes.

```
index=sales sourcetype=vendor_sales categoryId=$$categoryId_tok|s$$
product_name="*" Vendor="*" | stats sparkline(count) as Trend values(Vendor)
as "Vendors" values(product_name) as "Products Sold" by VendorCountry |
rename VendorCountry as "Country" | sort Country
```

☐ 12. In the Time Range dropdown, select: **Shared Time Picker (global)**.

☐ 13. Click **Apply**.

☐ 14. Save the dashboard changes.

Task 4: Create the source form by cloning the Vendor Sales Analysis form.

☐ 15. Navigate to the Lab 2 - Vendor Sales Analysis v1 form.

☐ 16. Clone the form using this naming:

- Title: `Lab 5 - Vendor Sales Analysis v2`
  Note: Remove the word Clone at end of title.
- ID: lab_5__vendor_form_sales_analysis_v2

☐ 17. In the *Dashboard has been cloned* window, click **Edit Panels**.

Task 5: Add a dynamic drilldown.

☐ 18. Click the **More actions** icon on the Vendor Sales by Category panel.

☐ 19. Select **Edit Drilldown** and use these settings:

- On Click: Link to dashboard
- App: Creating Dashboards
- Dashboard: Lab 5 – Sales Drilldown Destination
- Open in new tab: *Select this option*

☐ 20. Click **Advanced** and add these parameters:

- `form.categoryId_tok` = $click.value$
- `earliest` = $earliest$
- `latest` = $latest$

☐ 21. Click **Apply**.

☐ 22. Save the form changes and reload your browser.

Task 6: Test the dynamic drilldown.

☐ 23. Select a Country, State/Province, and City.

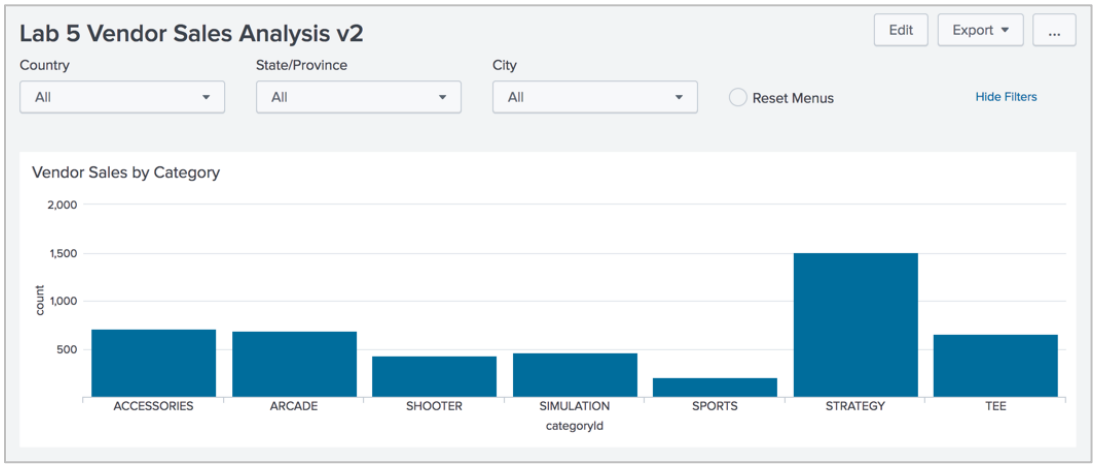
☐ 24. Click one of the categories on the Vendor Sales by Category chart.

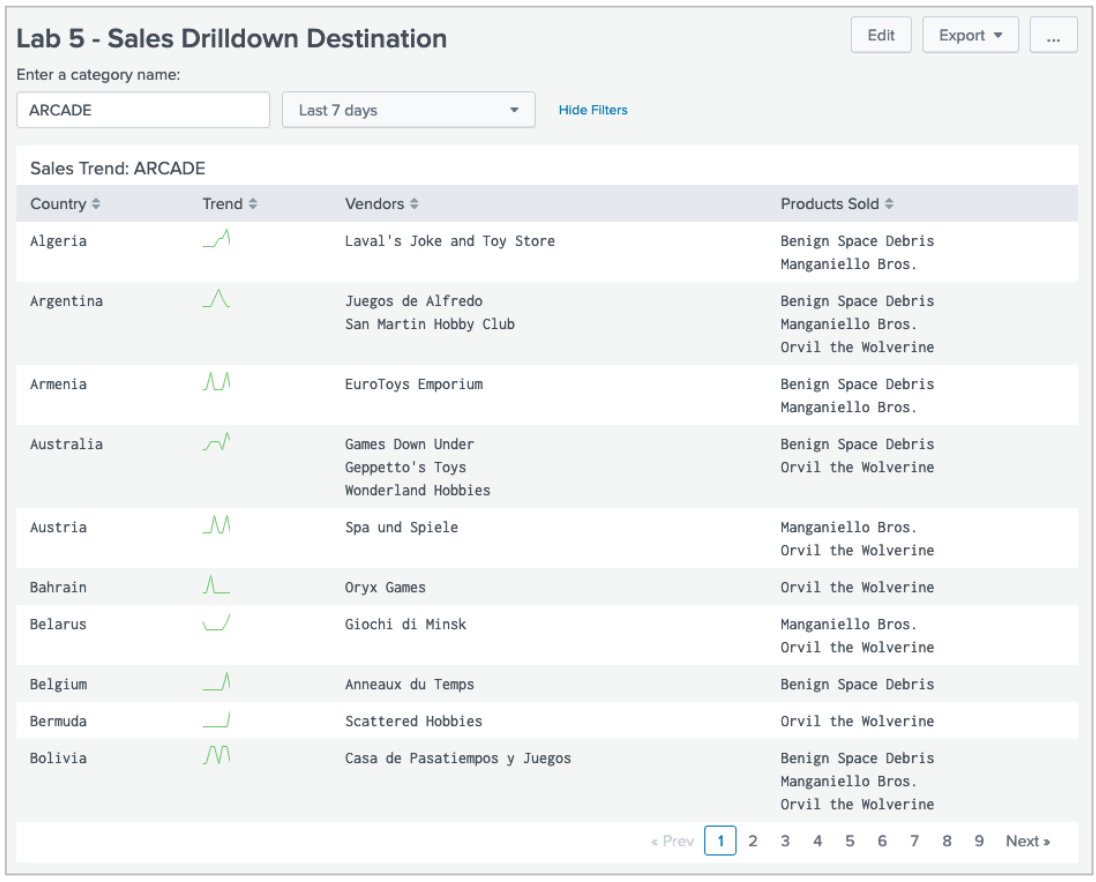
The Lab 5 – Sales Drilldown Destination opens and populates with data for the category.

Troubleshooting

- Reload your browser.
- Check for typos and trailing word spaces in the drilldown parameters.
- Make sure the destination dashboard is the target.
- Check for typos in the search used by the destination dashboard.

Example:


Lab 5 Vendor Sales Analysis v2
Edit
Export
...
Country
All
State/Province
All
City
All
Reset Menus
Hide Filters
Vendor Sales by Category
2,000
1,500
1,000
500
count
ACCESSORIES
ARCADE
SHOOTER
SIMULATION
SPORTS
STRATEGY
TEE
categoryId


Lab 5 - Sales Drilldown Destination
Edit
Export
...
Enter a category name:
ARCADE
Last 7 days
Hide Filters
Sales Trend: ARCADE
Country
Trend
Vendors
Products Sold
Algeria
Laval's Joke and Toy Store
Benign Space Debris
Manganiello Bros.
Argentina
Juegos de Alfredo
San Martin Hobby Club
Benign Space Debris
Manganiello Bros.
Orvil the Wolverine
Armenia
EuroToys Emporium
Benign Space Debris
Manganiello Bros.
Australia
Games Down Under
Geppetto's Toys
Wonderland Hobbies
Benign Space Debris
Orvil the Wolverine
Austria
Spa und Spiele
Manganiello Bros.
Orvil the Wolverine
Bahrain
Oryx Games
Orvil the Wolverine
Belarus
Giochi di Minsk
Manganiello Bros.
Orvil the Wolverine
Belgium
Anneaux du Temps
Benign Space Debris
Bermuda
Scattered Hobbies
Orvil the Wolverine
Bolivia
Casa de Pasatiempos y Juegos
Benign Space Debris
Manganiello Bros.
Orvil the Wolverine
« Prev
1
2
3
4
5
6
7
8
9
Next »

Lab Exercise 6 – Add Advanced Visualizations & Behaviors

Description

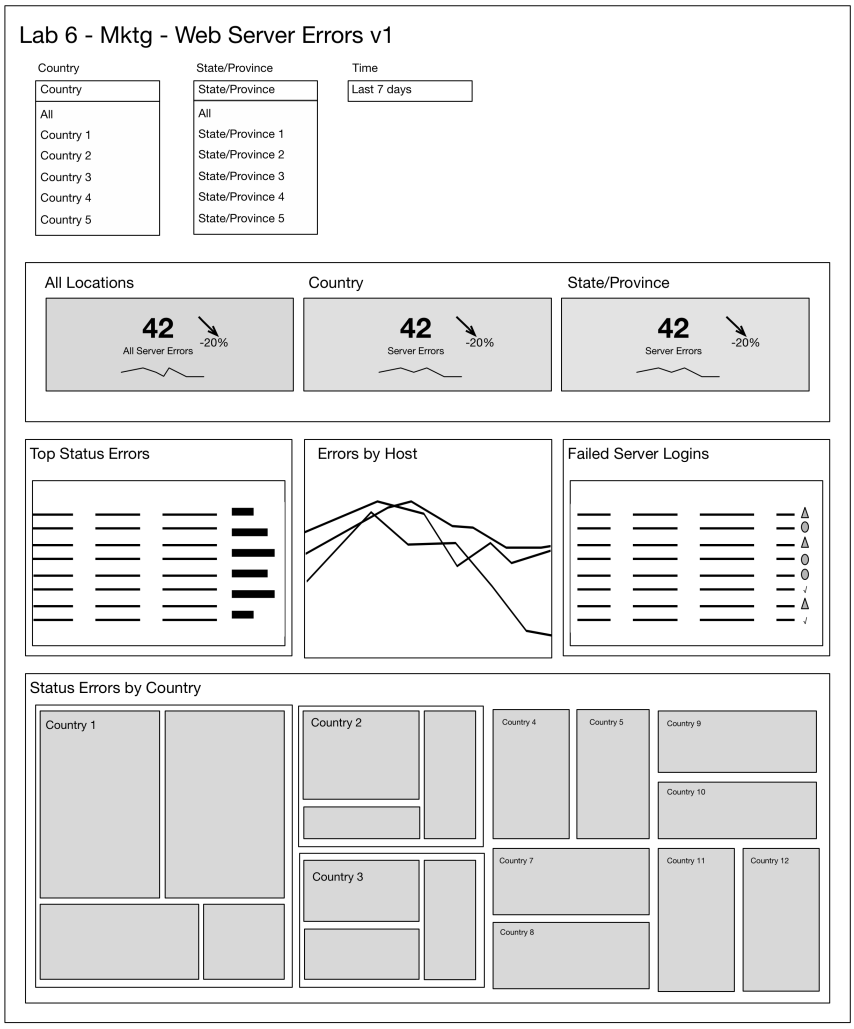
In this exercise, you will use simple XML extensions to add advanced visualizations and behaviors to a dashboard. The dashboard will include a treemap chart. A treemap chart displays hierarchical data using nested rectangles.

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.**

IMPORTANT: If you copy text from this document please note that character formatting and artifacts created by the PDF generation process can cause errors in the XML.

Working Example: Creating Dashboards > Lab Examples > Lab 6 Example - Mktg - Web Server Errors

Wireframe:



Steps

Task 1: Create a new dashboard.

☐ 1. Create a new dashboard and use the following naming:

- Dashboard Title: `Lab 6 - Mktg - Web Server Errors v1`
- Dashboard ID: lab_6__mktg__web_server_errors_v1
- Permissions: Private

Task 2: Add a multiselect input for country.

NOTE: The input will not populate **until after you've added the state/province input**.

☐ 2. Add a multiselect input and use the following settings:

General

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

Token Options

- Token: `country_tok`
- Token Prefix: `(`
- Token Suffix: `)`
- Token Value Prefix: `Country="`
- Token Value Suffix: `"`
- Delimiter: OR

IMPORTANT: Add a space before and after the delimiter.

Preview (Country="value1" OR Country="value2" OR ...)

Static Options

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

Token Options

- Default: `All`

Dynamic Options

- Search String:

```
| inputlookup bcg_online
| dedup Country
| fields Country
| sort Country
```

- Time Input: Last 24 Hours
- Field For Label: `Country`
- Field For Value: `Country`

☐ 3. Click **Apply**.

☐ 4. Click the **Country** input and make sure a list of countries appears.

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

☐ 5. Add a multiselect input and use the following settings:

General

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

Token Options

- Token: `state_tok`
- Token Prefix: (
- Token Suffix:)
- Token Value Prefix: `Region="`
- Token Value Suffix: "
- Delimiter: OR

IMPORTANT: Add a space before and after the delimiter.

Preview **(Region="value1" OR Region="value2" OR ...)**

Static Options

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

Token Options

- Default: `All`

Dynamic Options

- Search String:

```
| inputlookup bcg_online
| search $country_tok$
| dedup Region
| fields Region
| sort Region
```

- Time Input: Last 24 Hours
- Field For Label: `Region`
- Field For Value: `Region`

☐ 6. Click **Apply**.

☐ 7. Click the **State/Province** input and make sure a list of states appears.

NOTE: If an input is not working, check its settings for typos and reload your browser.

Task 4: Add a time input.

☐ 8. Add a time input and use the following settings:

General

- Label: `Time`
- Search on Change: ☑

Token Options

- Token: `time_tok`
- Default: Last 30 Days

☐ 9. Click **Apply**.

NOTE: When adding a selection to multiselect input the default selection is not removed. This behavior can be added using JavaScript and a simple xml extension. A file with the necessary JavaScript has already been loaded to your Splunk instance. The code is shown at the end of this document.

Task 5: Add custom multiselect input behavior.

☐ 10. Open the XML Editor.

☐ 11. Add a script reference for `clear_ms_all.js` to the opening <form> tag.

```
<form script="clear_ms_all.js">
  <label>Lab 6 - Mktg - Web Server Errors v1</label>
...
```

☐ 12. Locate the XML for the **Country** input and add an id: `id="input1"`

```
...
<fieldset submitButton="false">
 <input id="input1" type="multiselect" token="country_tok" searchWhenChanged="true">
  <label>Country</label>
...
```

☐ 13. Locate the XML for the **State/Province** input and add an id: `id="input2"`

```
...
</input>
 <input id="input2" type="multiselect" token="country_tok" searchWhenChanged="true">
  <label>State/Province</label>
...
```

☐ 14. Save the XML changes and reload your browser.

Task 6: Test the multiselect inputs.

☐ 15. Select one or more values from each of the multiselect input fields.

☐ 16. Make sure the All selection is automatically removed.

Troubleshooting

- Reload your browser.
- Check for typos in the XML.

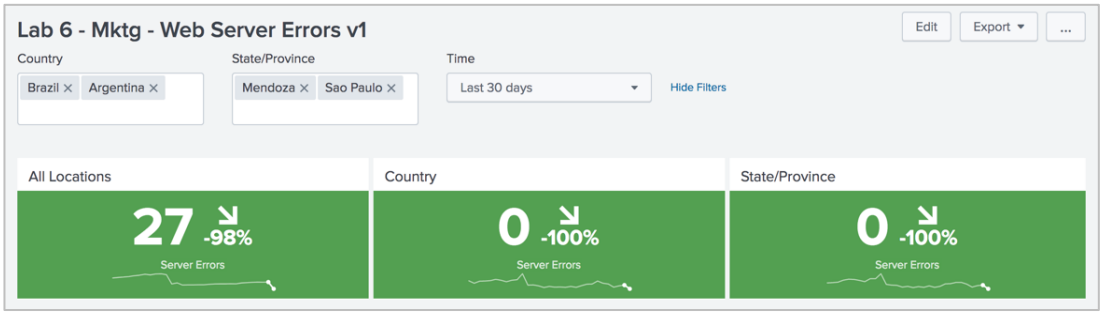
Task 7: Add three single value panels.

- [ ] 17. Open the dashboard editor.
- [ ] 18. Add three prebuilt panels:
    - Lab 6 - All Locations
    - Lab 6 - Country
    - Lab 6 - State/Province
- [ ] 19. Convert the panels to inline panels.
- [ ] 20. Remove **Lab 6 –** from each panel's title.
- [ ] 21. Position the panels on the same row: All Locations, Country and State/Province.
- [ ] 22. Save the dashboard changes and reload your browser.

Task 8: Test the tokens.

- [ ] 23. Select a Country, State/Province, and time from the inputs.
- [ ] 24. Verify the Country and State/Province panel values update to reflect the new input choices.

Example:



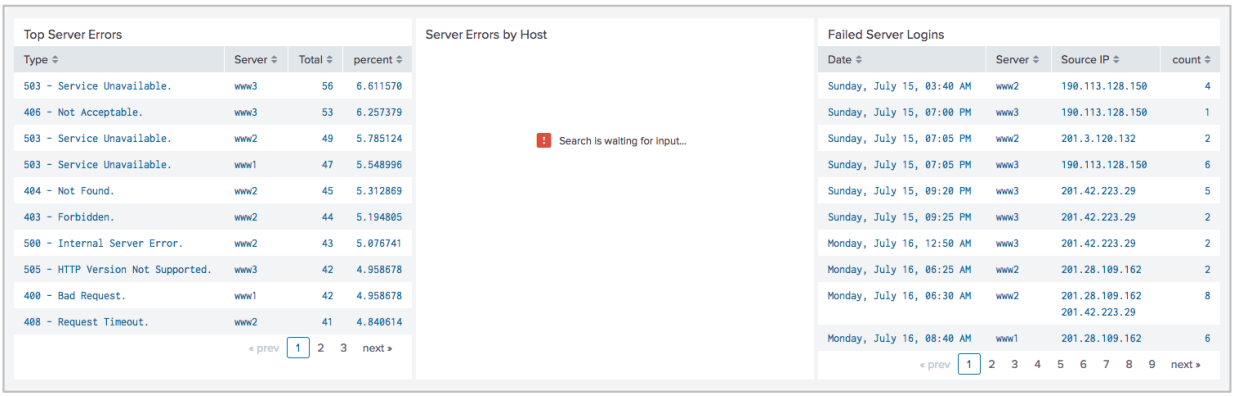
Task 9: Add a chart and two table panels.

- [ ] 25. Add three prebuilt panels:
    - Lab 6 - Top Server Errors
    - Lab 6 - Server Errors by Host
    - Lab 6 - Failed Server Logins

NOTE: The Lab 6 – Server Errors by Host panel will not populate until a later task is completed. This is normal and expected.

- [ ] 26. Convert the panels to inline panels.
- [ ] 27. Remove **Lab 6 –** from each panel's title.
- [ ] 28. Position the panels on the same row: Top Server Errors, Server Errors by Host, Failed Server Logins.
- [ ] 29. Save the dashboard changes and reload your browser.

Example:



Task 10: Display a data bar in the Top Server Errors table.

NOTE: By referencing a JavaScript and CSS file, the Top Server Errors table will display a data bar in place of the percent field's value.

Since these two files are in the /appserver/static directory of the Splunk Dashboards Examples app (already installed on your lab server), **include the app reference** for each file.

☐ 30. Open the XML editor.

☐ 31. Locate the opening <form> element and add a script reference for `table_data_bar.js` and `table_data_bar.css`.

```
<form script="clear_ms_all.js, simple_xml_examples:table_data_bar.js"
stylesheet="simple_xml_examples:table_data_bar.css">
  <label>Lab 6 - Mktg - Web Server Errors v1</label>
...
```

☐ 32. Locate the XML for the Top Server Errors table.

☐ 33. Add an id to the opening table element: `id="table1"`

```
...
<row>
 <panel>
   <title>Top Server Errors</title>
    <table id="table1">
...
```

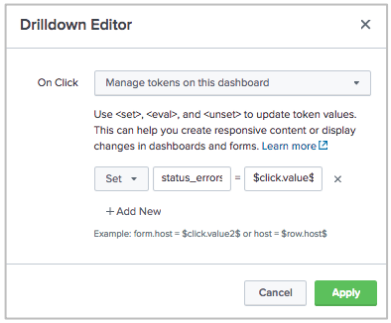
NOTE: Since the table id is referenced in the JavaScript file, adding it here links this table to the appropriate JavaScript.

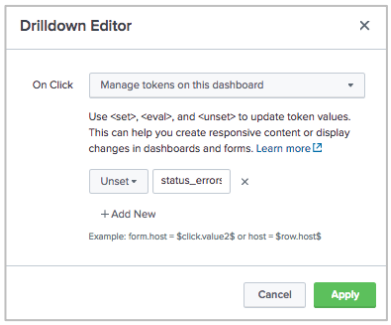
☐ 34. Save the XML changes and reload your browser.

Example:

Task 11: Configure dynamic display for the Server Errors by Host panel.

- ☐ 35. Open the dashboard editor.
- ☐ 36. On the Top Server Errors panel, click the **More actions** button.
- ☐ 37. Select **Edit Drilldown**.
- ☐ 38. Select **Manage tokens on this dashboard** from the On Click dropdown.
- ☐ 39. Select **Set** from the dropdown menu.
- ☐ 40. Type `status_errors_tok` in the left box.
- ☐ 41. Click in the box on the right and select `$click.value$`
- ☐ 42. Click **Apply**.
- ☐ 43. On the Server Errors by Host panel, click the **More actions** button.
- ☐ 44. Select **Edit Drilldown**.
- ☐ 45. Select **Manage tokens on this dashboard** from the On Click dropdown.
- ☐ 46. Select **Unset** from the dropdown menu.
- ☐ 47. Type `status_errors_tok` in the box.
- ☐ 49. Click **Apply**.
- ☐ 49. Open the XML editor.
- ☐ 50. Locate the XML for Server Errors by Host panel.
- ☐ 51. Add a depends attribute for the status_errors_tok token to the <chart> tag.
- ☐ 52. Add the token to the panel and chart titles.






```
...
<panel>
 <title>Server Errors by Host - $status_errors_tok$</title>
  <chart depends="$status_errors_tok$">
   <search>
...
```

- ☐ 53. Save the dashboard changes and reload your browser.

Task 12: Test the dynamic display.

☐ 54. Click on one of the errors listed in the Top Server Errors panel.

☐ 55. Make sure the Errors by Host panel appears and its title includes the value from the Type column of the table row you clicked on.

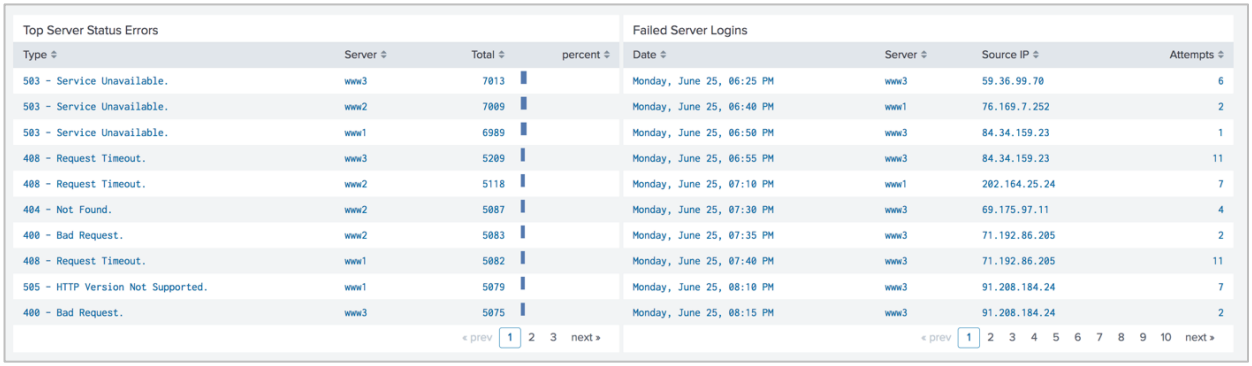
☐ 56. Click one of the lines listed in the Server Errors by Host panel's line chart.

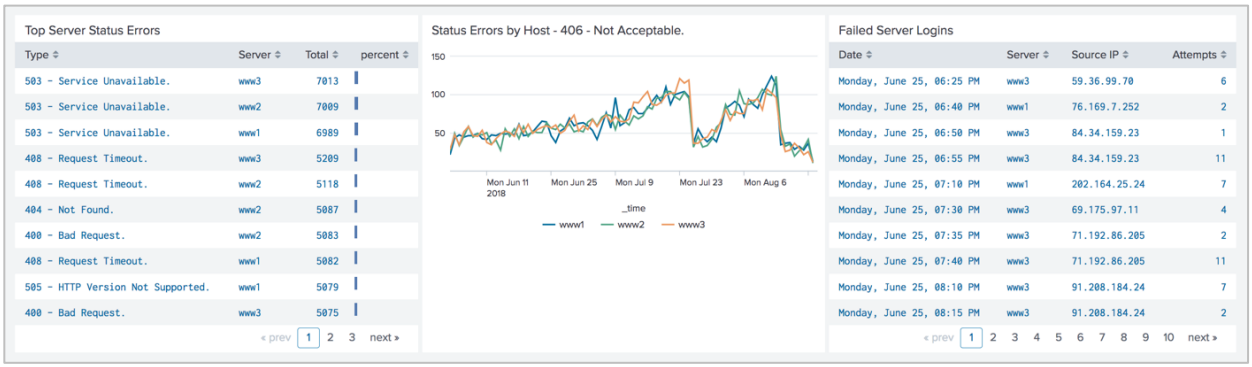
☐ 57. Make sure the Errors by Host panel disappears.

Troubleshooting

- Reload your browser.
- Check for typos and trailing word spaces in the drilldown parameters and XML.

Example:





Task 13: Display icons in the Failed Server Logins table.

NOTE: By referencing a JavaScript and CSS file, the Failed Server Logins table will display an icon beside each value in the Attempts column based on custom conditions.

☐ 58. Open the XML Editor.

☐ 59. Locate the opening <form> element and add a script reference for `table_icons_inline.js` and `table_decorations.css`.

```
<form script="clear_ms_all.js, simple_xml_examples:table_data_bar.js,
table_icons_inline.js" stylesheet="simple_xml_examples:table_data_bar.css,
table_decorations.css">

  <label>Lab 6 - Mktg - Web Server Errors v1</label>

...
```

☐ 60. Locate the XML for the Failed Server Logins table.

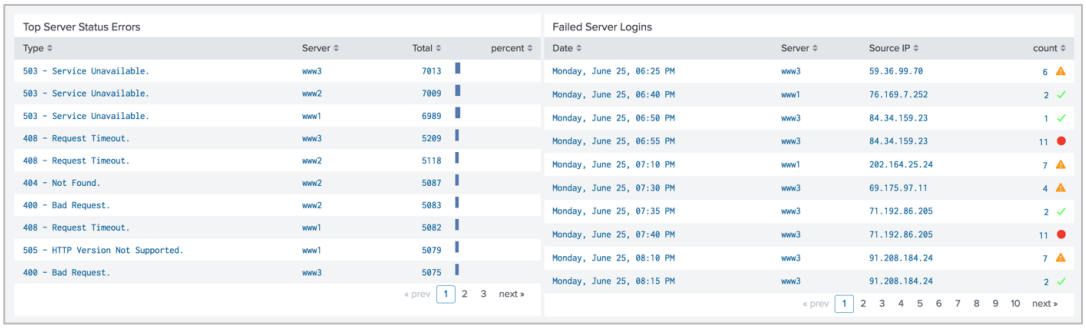
☐ 61. Add an id to the opening table element: `id="table2"`

```
...
 <panel>
   <title>Failed Server Logins</title>
    <table id="table2">
...
```

NOTE: Since the table id is referenced in the JavaScript file, adding it here links this table to the appropriate JavaScript.

☐ 62. Save the XML changes and reload your browser.

Example:



Task 14: Add a Splunk Custom Visualization.

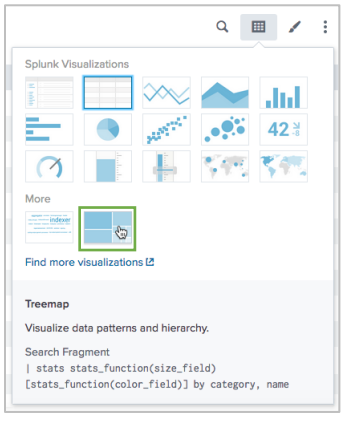
☐ 63. Add a prebuilt panel: Lab 6 – Status Errors by Country

☐ 64. Convert the panel to an inline panel.

☐ 65. Remove **Lab 6 –** from the panel's title.

☐ 66. On the Status Errors by Country panel, click the **Select visualization** icon.

☐ 67. Under More, select **Treemap**.



☐ 68. Format the visualization with the following settings:

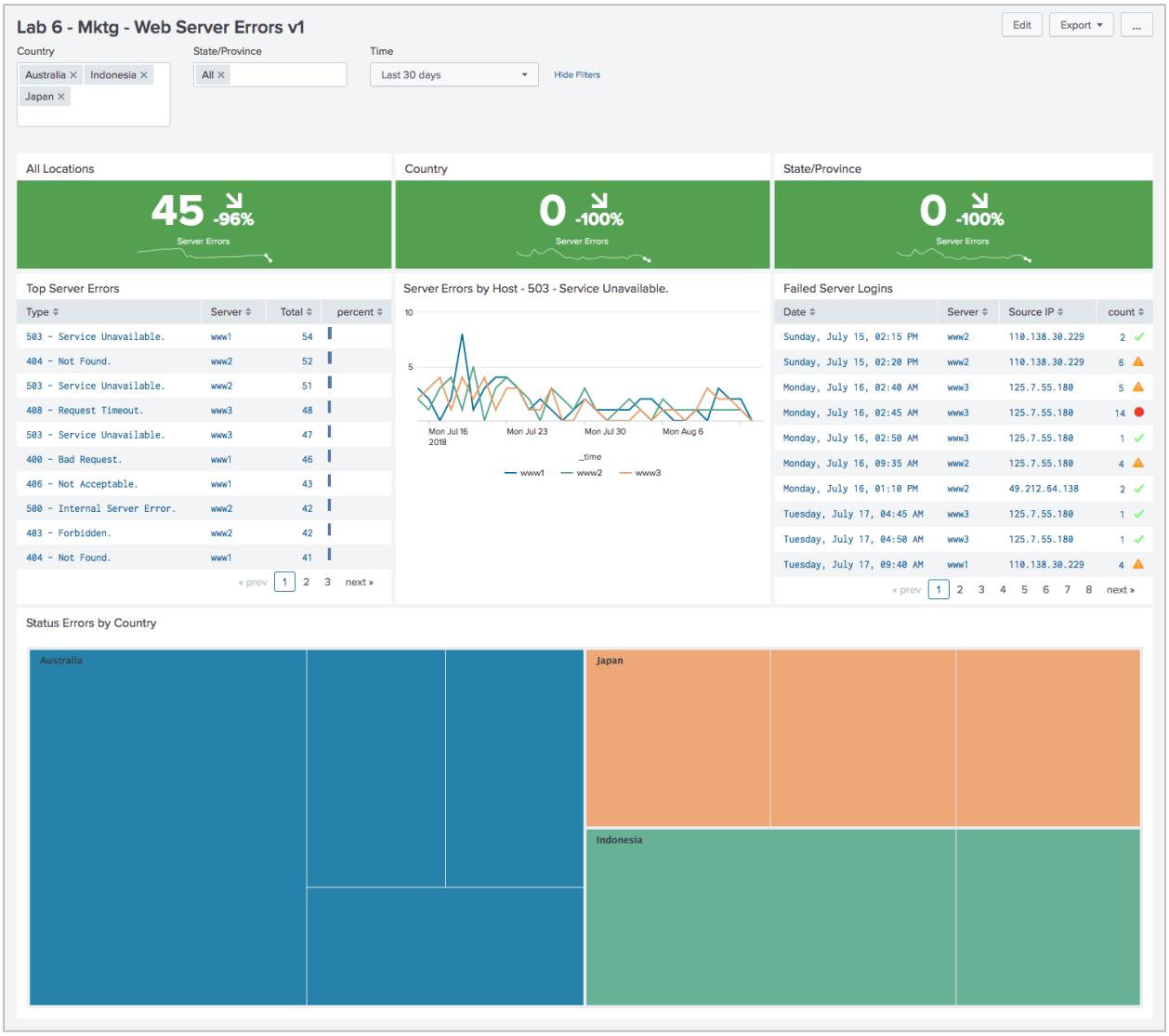
General
- Use zoom: Yes
- Show labels: Yes
- Show legend: No
- Show tooltips: Yes
- Max categories: 50

Color
- Use colors: Yes
- Color mode: Categorical

☐ 69. Save the dashboard changes and reload your browser.

☐ 70. Move your mouse over the different tiles in the visualization to see the pop-ups for the different error values.

Example:


Lab 6 - Mktg - Web Server Errors v1
Edit
Export
Country
Australia
Indonesia
Japan
State/Province
All
Time
Last 30 days
Hide Filters
All Locations
45 -96%
Server Errors
Country
0 -100%
Server Errors
State/Province
0 -100%
Server Errors
Top Server Errors
Type
Server
Total
percent
503 - Service Unavailable. www1 54
404 - Not Found. www2 52
503 - Service Unavailable. www2 51
408 - Request Timeout. www3 48
503 - Service Unavailable. www3 47
400 - Bad Request. www1 46
406 - Not Acceptable. www1 43
500 - Internal Server Error. www2 42
403 - Forbidden. www2 42
404 - Not Found. www1 41
« prev 1 2 3 next »
Server Errors by Host - 503 - Service Unavailable.
10
5
Mon Jul 16 2018
Mon Jul 23
Mon Jul 30
Mon Aug 6
_time
www1
www2
www3
Failed Server Logins
Date
Server
Source IP
count
Sunday, July 15, 02:15 PM www2 110.138.30.229 2
Sunday, July 15, 02:20 PM www2 110.138.30.229 6
Monday, July 16, 02:40 AM www3 125.7.55.180 5
Monday, July 16, 02:45 AM www3 125.7.55.180 14
Monday, July 16, 02:50 AM www3 125.7.55.180 1
Monday, July 16, 09:35 AM www2 125.7.55.180 4
Monday, July 16, 01:10 PM www2 49.212.64.138 2
Tuesday, July 17, 04:45 AM www3 125.7.55.180 1
Tuesday, July 17, 04:50 AM www3 125.7.55.180 1
Tuesday, July 17, 09:40 AM www1 110.138.30.229 4
« prev 1 2 3 4 5 6 7 8 next »
Status Errors by Country
Australia
Japan
Indonesia

Congratulations

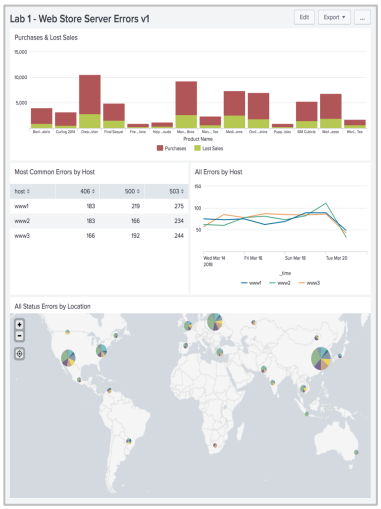
You've Completed the Lab Exercises!

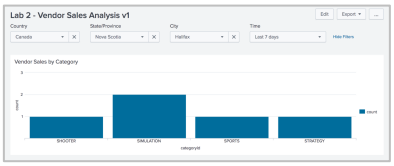
You created dashboards and forms:

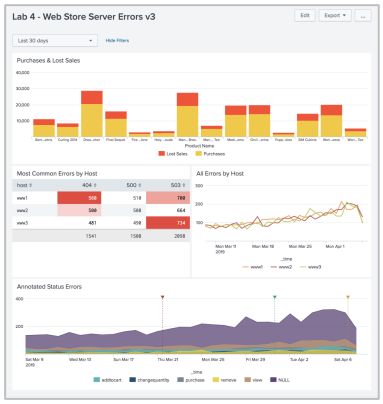
- from the dashboards page
- when saving searches
- when saving reports
- by cloning existing dashboards

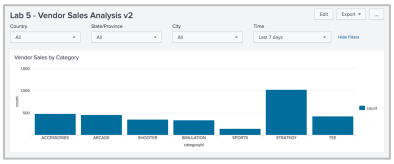
Those dashboards and forms include:

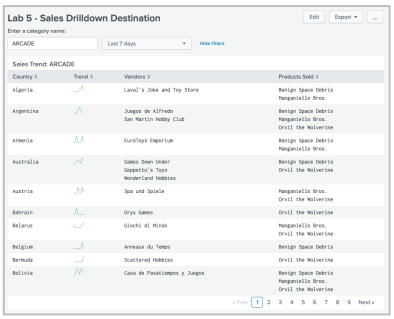
- cascading inputs
- tokens
- prebuilt panels
- accelerated and scheduled reports
- the tstats command against an accelerated data model
- an annotated search
- custom chart colors
- modified dashboard properties
- a dynamic drilldown
- a hidden panel
- a splunk custom visualization
- simple xml extensions

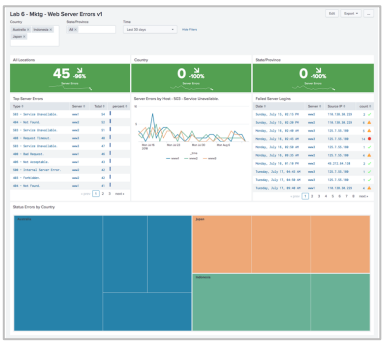












Simple XML Extension

Code Examples

Clear Multiselect Input

Below is a snippet of the JavaScript used to clear the ALL selection when other option(s) are chosen from a multiselect input (clear_ms_all.js).

```
var selection = [];
/* The code below is for the first multiselect input. For additional, copy and paste it below.
Then change multi1 to multi2, multi3, etc. and change input1 to input2, input3, etc. */
var multi1 = splunkjs.mvc.Components.getInstance("input1");

 multi1.on("change", function(){
      // get current selection
      selection = multi1.val();

      // check if there is more than one entry and one of them is "*"
      if (selection.length > 1 && ~(selection.indexOf("*"))) {
          if (selection.indexOf("*") == 0) {
              // "*" was first, remove it and leave rest
              selection.splice(selection.indexOf("*"), 1);
              multi1.val(selection);
              multi1.render();
          } else {
              // if "*" was added later, remove rest and leave it
              multi1.val("*");
              multi1.render();

          }
      }
  });
```

Table Icons & Table Data Bars

The files used for the data bars and icons in exercise 6 are available in the Splunk Dashboards Examples app (https://splunkbase.splunk.com/app/1603/).

Table Icons

`table_icons_inline.js`

`table_decorations.css`

Table Data Bars

`table_data_bar.js`

`table_data_bar.css`