Splunk4Ninjas - Dashboard Studio

Minimum versions required:
Splunk Enterprise 9.3
Splunk Cloud 9.2.2403

splunk>
a CISCO company

Forward-looking statements

This presentation may contain forward-looking statements that are subject to the safe harbors created under the Securities Act of 1933, as amended, and the Securities Exchange Act of 1934, as amended. All statements other than statements of historical facts are statements that could be deemed forward-looking statements. These statements are based on current expectations, estimates, forecasts, and projections about the industries in which we operate and the beliefs and assumptions of our management based on the information currently available to us. Words such as “expects,” “anticipates,” “targets,” “goals,” “projects,” “intends,” “plans,” “believes,” “momentum,” “seeks,” “estimates,” “continues,” “endeavors,” “strives,” “may,” variations of such words, and similar expressions are intended to identify such forward-looking statements. In addition, any statements that refer to (1) our goals, commitments, and programs; (2) our business plans, initiatives, and objectives; and (3) our assumptions and expectations, including our expectations regarding our financial performance, products, technology, strategy, customers, markets, acquisitions and investments are forward-looking statements. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation. Readers are cautioned that these forward-looking statements are only predictions and are subject to risks, uncertainties, and assumptions that are difficult to predict, including those identified in the “Risk Factors” section of Cisco’s most recent report on Form 10-Q filed on February 20, 2024 and its most recent report on Form 10-K filed on September 7, 2023, as well as the “Risk Factors” section of Splunk’s most recent report on Form 10-Q filed with the SEC on November 28, 2023. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by Cisco or Splunk, on Cisco or Splunk’s website or otherwise, it may not contain current or accurate information. Cisco and Splunk undertake no obligation to revise or update any forward-looking statements for any reason, except as required by law.

In addition, any information about new products, features, functionality or our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment or be relied upon in making a purchasing decision. We undertake no commitment, promise or obligation either to develop the features or functionalities described, in beta or in preview (used interchangeably), or to include any such feature or functionality in a future release. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

splunk>
a CISCO company

Prerequisites

Prior to beginning this workshop, you should have either:

- Taken Splunk4Rookies - Dashboard Studio, or
- Have a basic understanding/have built a dashboard in Dashboard Studio

You should already understand:

- How to navigate the editing UI and have a high level understanding of the source code
- How to add visualizations, searches, and inputs

Learning objectives

- **Add interactivity**
  - Configure conditional show/hide
- **Edit source code**
  - Apply options with Dynamic Options Syntax
- **Advanced visualizations**
  - Add multiple data sources to a visualization
  - Add custom dynamic SVGs
- **Apply best practices**
  - Base and chain searches
  - Reusing searches for multiple visualizations
  - Design best practices

Enroll in Today's Workshop

Tasks

1. Get a splunk.com account if you don't have one yet:
   https://splk.it/SignUp
2. Enroll in the Splunk Show workshop event:
   https://show.splunk.com/event/<eventID>
3. Download the hands-on lab guide:
   https://splk.it/S4NDS-Lab-Guide

   Contains step-by-step instructions for all of today's exercises!
4. Download a copy of today's slide deck:
   https://splk.it/S4NDS-Attendee

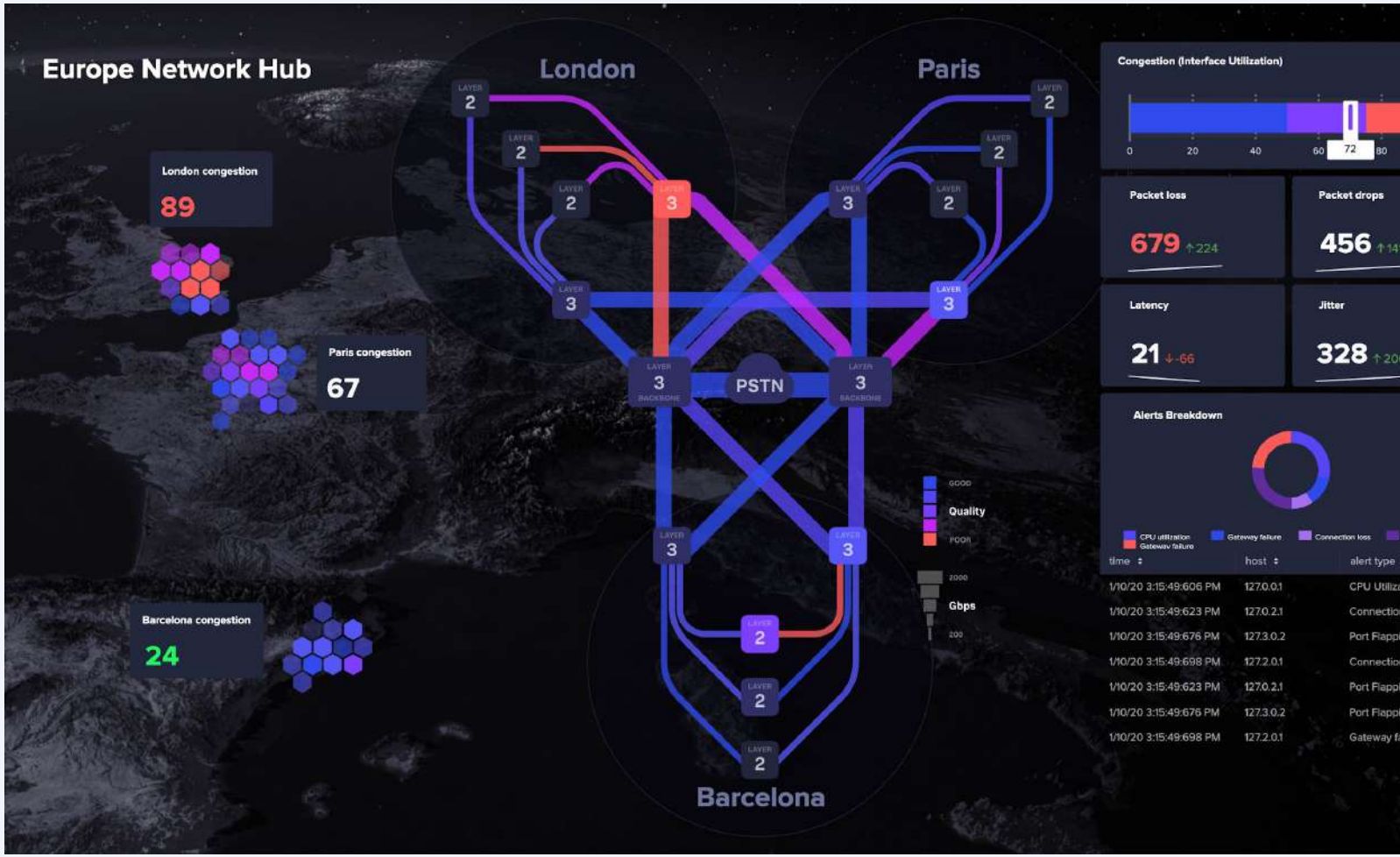
Goal

Enroll in today's event

About Dashboard Studio

Dashboard Studio is the future of Splunk dashboards

Designed for intuitive point-and-click building, while maintaining flexibility for advanced use cases



The evolution of Dashboard Studio

Splunk Enterprise 8.2

Beautiful, but static dashboards

- Pixel perfect layouts
- Drag & drop editing
- Native support for images, text, custom SVGs

Splunk Enterprise 9.0 and 9.1

Interactivity & token support

- Set & pass tokens via drilldown interactions
- Search-based tokens
- Show/hide panels
- Inputs in canvas

Splunk Enterprise 9.2

Usability & viz improvements

- Trellis for single value visualizations
- Events viewer
- Bigger code editor
- Classic to Studio conversion

Splunk Enterprise 9.3

Sharing & more interactivity

- Scheduled email & API export
- Shared tokens
- Set multiple interactions

Most features are available in Dashboard Studio

Key features	Classic Dashboards	Dashboard Studio
Data sources ad-hoc, base and post-process, saved searches	✅	✅ UI to easily manage and reuse
Standard charts axes charts, maps, single values	✅	✅
3rd party visualizations	✅	❌ custom choro SVGs
Flexible layouts	❌ rigid, requires dev skills to customize	✅ native support for customization
Inputs	✅	✅
Interactivity link to other pages, set and pass tokens	✅	✅
Tokenization eval, set, condition; search-based tokens	✅	🟡 workarounds available for most use cases
Sharing scheduled email export, export to CSV, on-demand pdf/png export	🟡 limited csv export, pdf breaks layout	✅
Extensions custom CSS, HTML, JS	✅	❌ some styling natively supported

When to use Dashboard Studio vs Classic

Dashboard Studio

- Executive or customer facing dashboards
- Visualizing physical objects or space
- New dashboards
- New or less technical users
- Any use case that doesn't require heavy customization or advanced interactivity

Classic

- Dashboards requiring advanced interactivity or advanced token manipulation
  - Token eval
  - Buttons
- You need specific custom visualizations
- You need custom interactions using custom JS
- You need a custom styling

High visibility dashboards

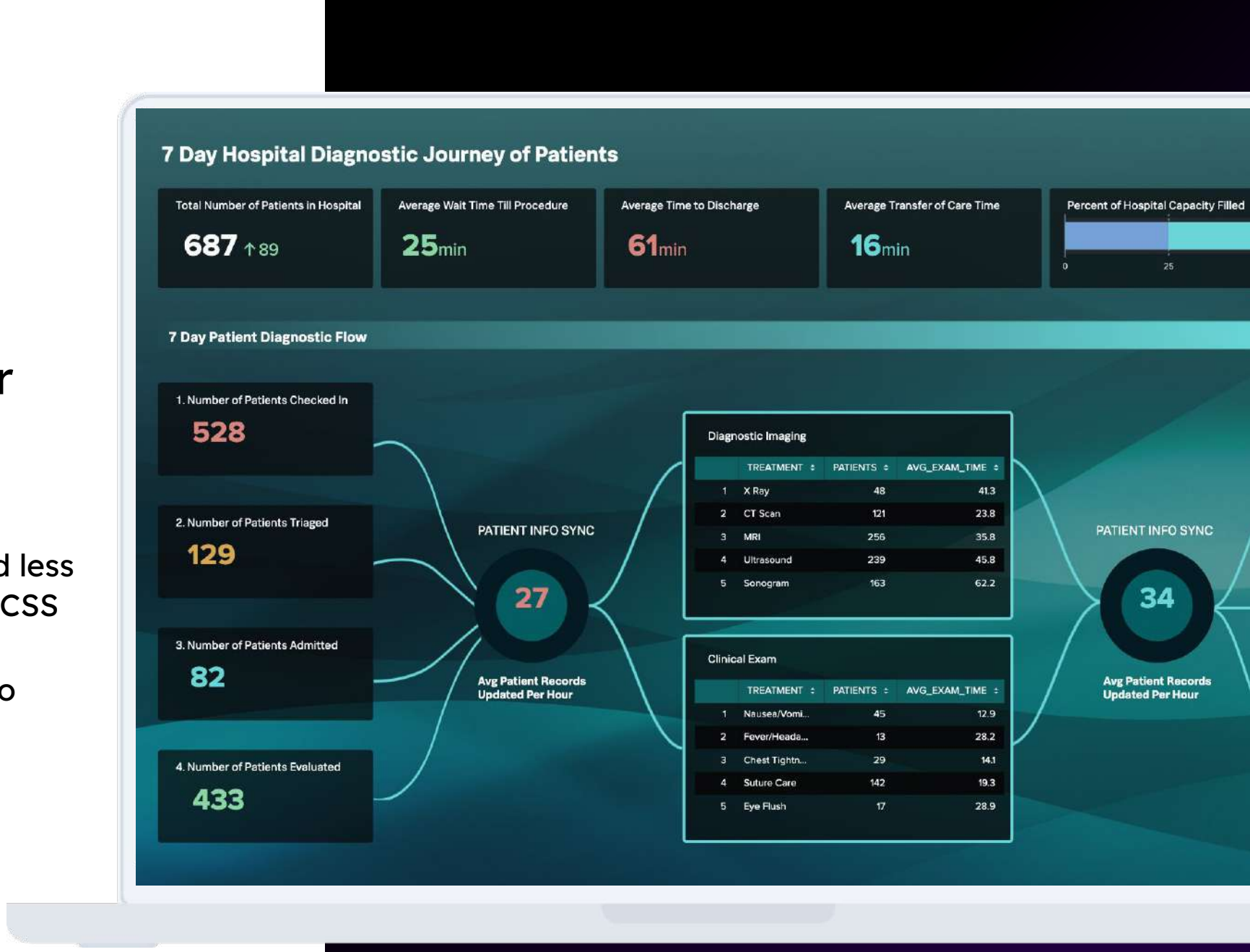
Create intuitive, visually appealing dashboards that **executives and business partners** can understand

- **Enable teams** outside of IT and security to leverage Splunk
- Simultaneously **increase data density and readability,** especially for display walls

Empower more users

Free up your developer resources with Dashboard Studio

- **No custom development** and less app management for customCSS or customJS
- Enable **less technical users** to self-serve in customizing dashboards

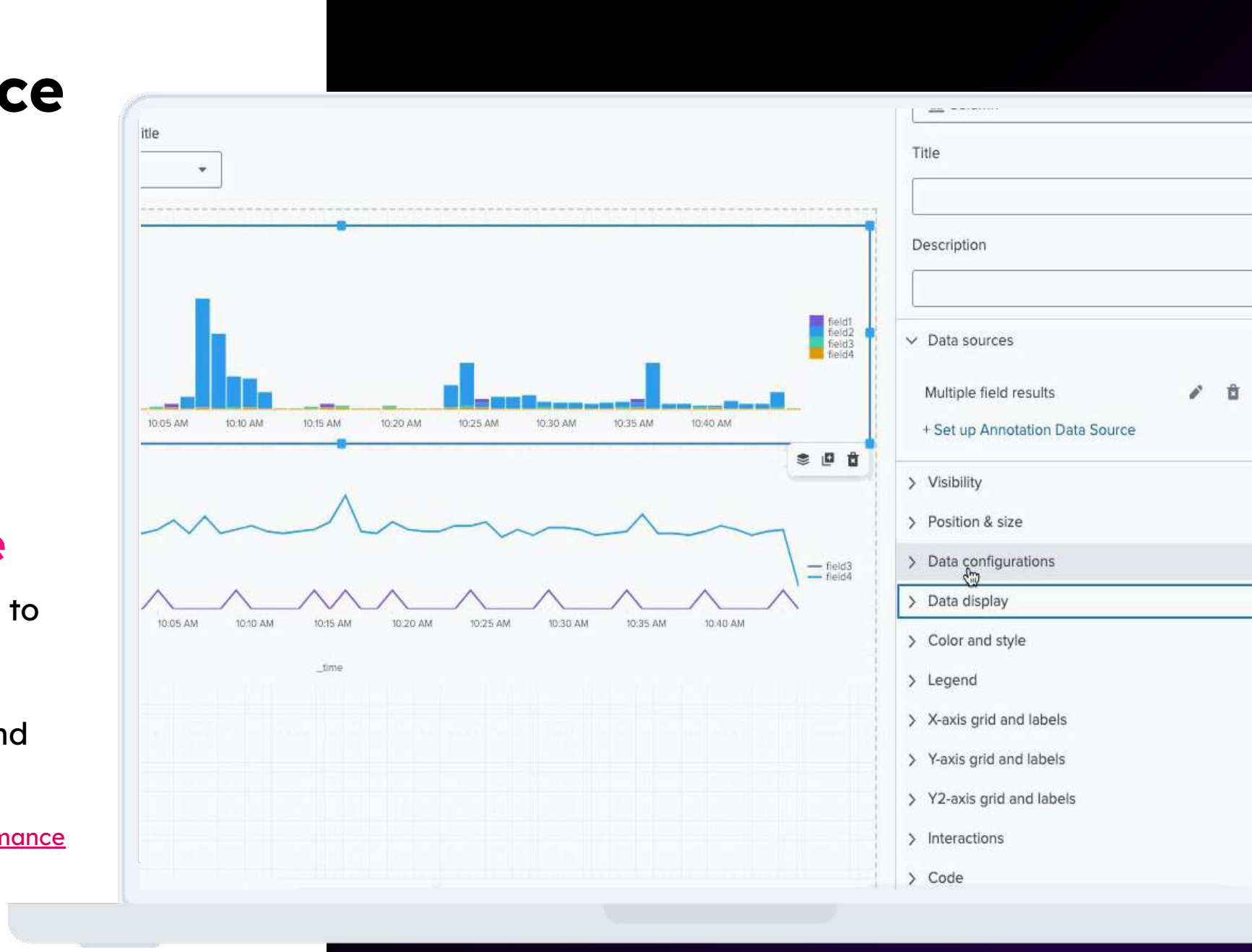


Minimize resource consumption

Reuse searches to reduce resource consumption and **improve performance**

- Point-and-click configuration to use one search for multiple visualizations
- Intuitive UI to create base and chain searches

Check out the Improving Dashboard Performance and Resource Usage tech talk to learn more



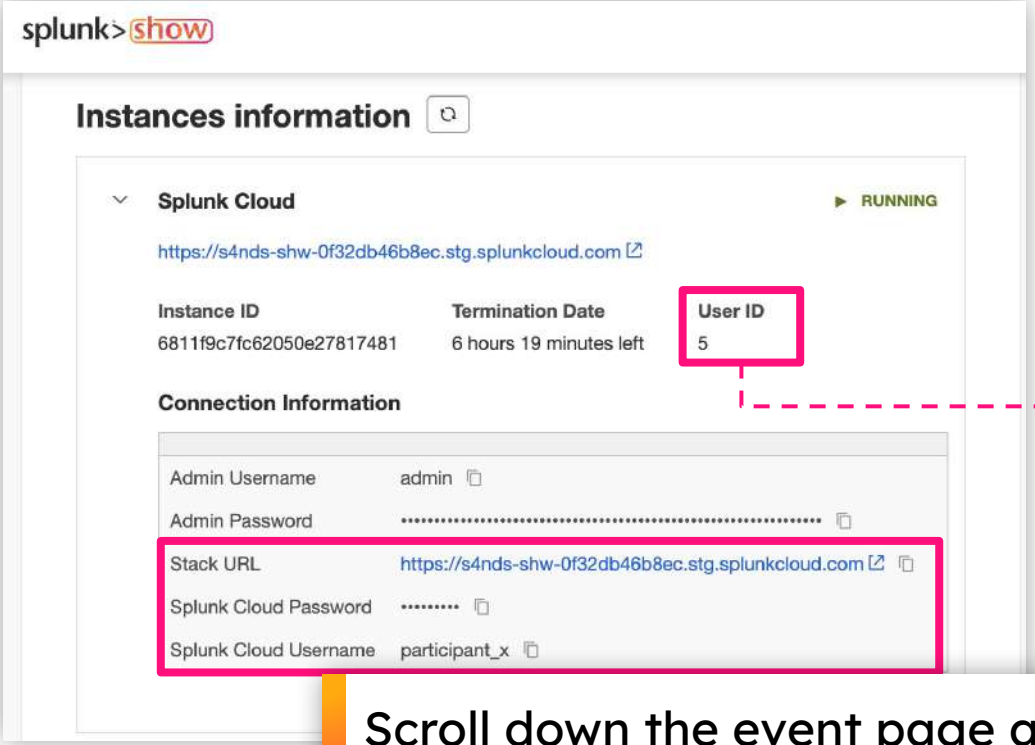
Dashboard Studio unlocks new possibilities

Key use cases	Classic Dashboards	Dashboard Studio
High visibility dashboards for executives and business partners	❌ Requires custom development	✅ Native support, point-and-click UI
Easy dashboard customization enable end users to apply advanced configurations	❌ Requires custom development	✅ Native support, point-and-click UI
Optimized for performance reduce resource consumption	❌ Limited; requires source code editing	✅ Default behavior, point-and-click UI
Effective visual communication flexible layouts and graphical elements	❌ Rigid layouts, charts only	✅ Flexible layouts, graphical elements
New features and enhancements with each major Splunk release	❌	✅

Workshop Time!

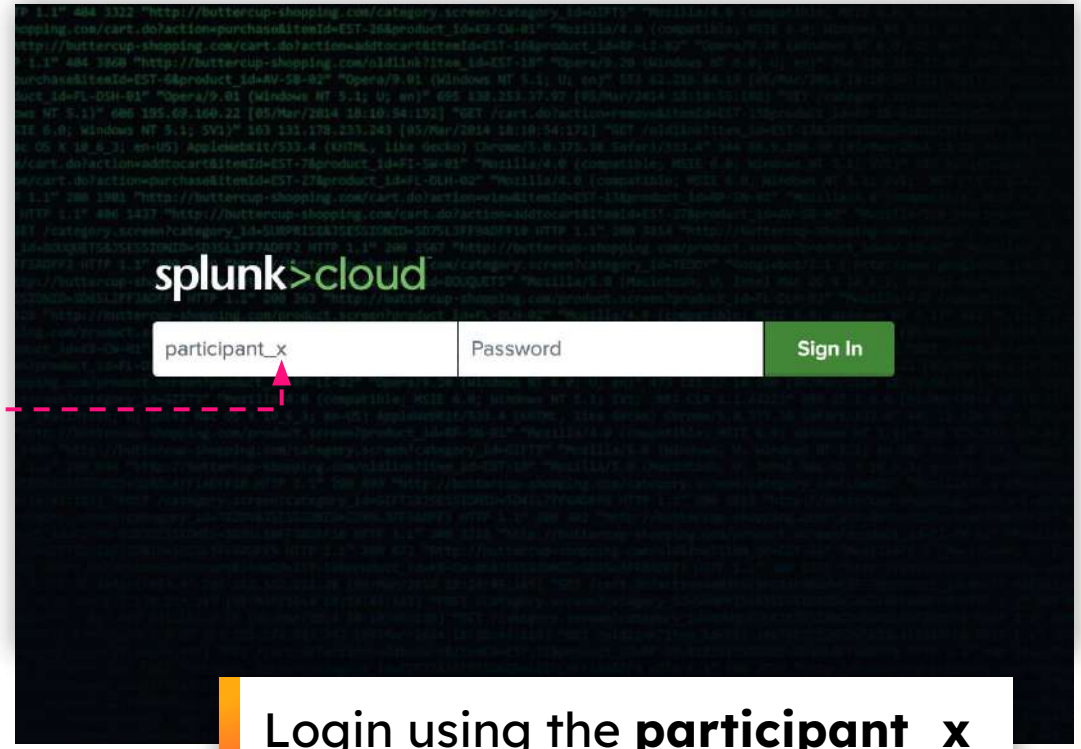
Login to Splunk

Locate your instance URL and credentials in the Splunk Show event
https://show.splunk.com



Scroll down the event page and expand the **Splunk Cloud** section to view your login details

Log in to your Splunk instance

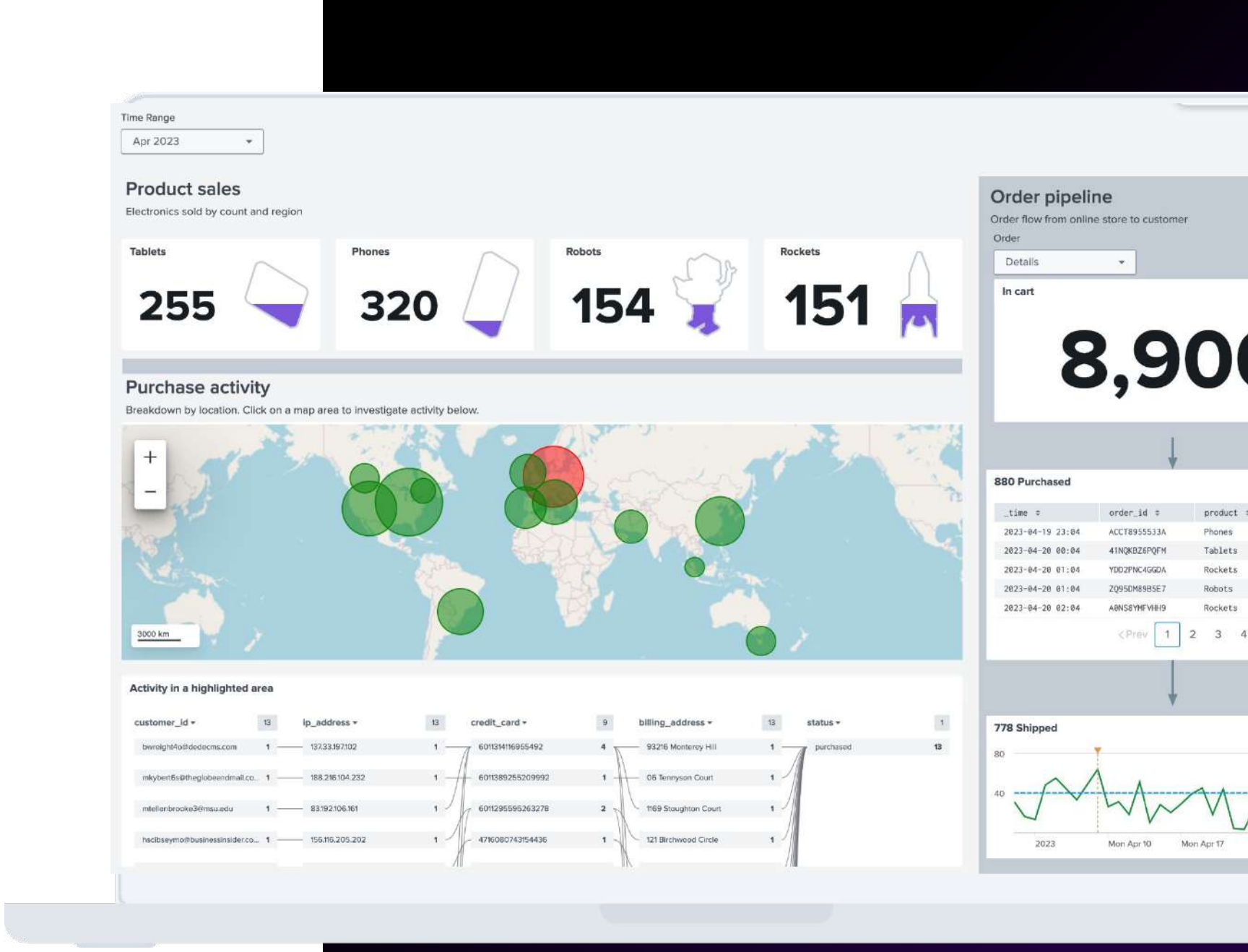


Login using the **participant_x** credentials, replacing ‘x’ with your displayed User ID

Scenario

You are an analyst at **Buttercup Games**

You have been tasked with building a **dashboard** for analyzing customer purchase activity



Dashboard use case and requirements

Use case

Your dashboard will provide insights into product purchase trends such as popular products and order pipeline details for both Buttercup games managers and analysts.

The managers only need a high level overview, but analysts will need to toggle between the overview and a detailed view to drill into more specific order information and investigate suspicious credit card activity.

Requirements

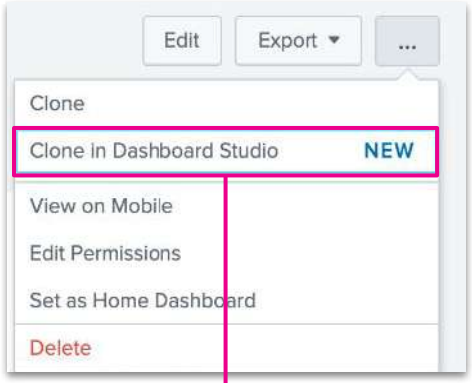
- Interactive inputs that allow users to toggle between an overview or detailed view for order pipeline status
- Insights into product purchasing trends and metrics
- Insights into abnormal credit card behavior

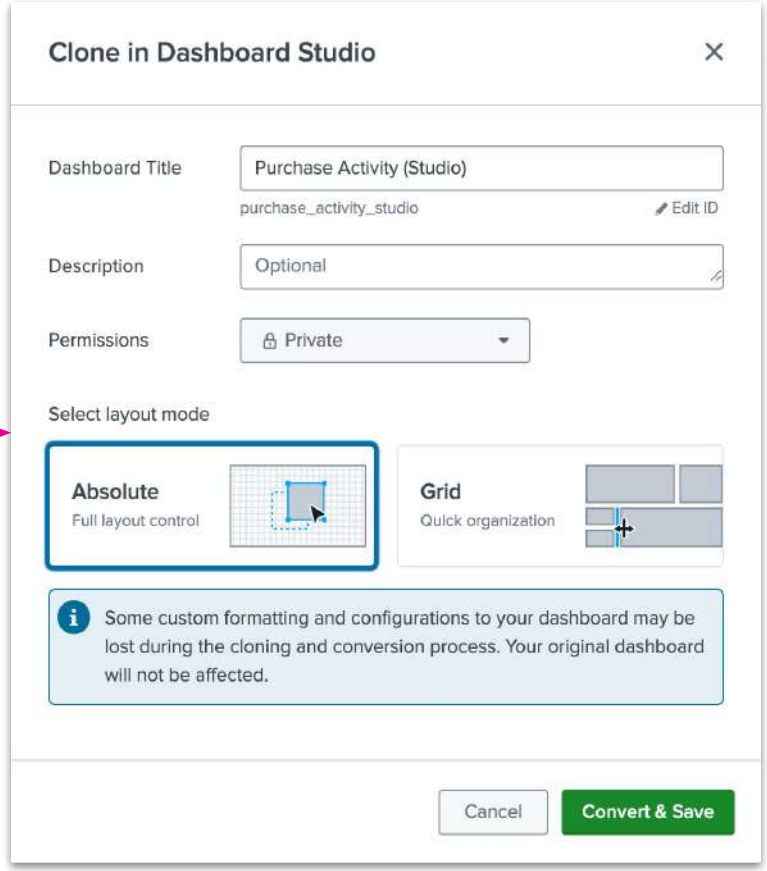
Exercise 1: Set up your environment

Convert a Classic dashboard

Instead of building a net-new dashboard, let's **clone** an existing Classic dashboard!

1. Open the **Purchase Activity (Classic)** dashboard
2. Expand the **...** menu and select **Clone in Dashboard Studio**
3. Update the Dashboard Title to **Purchase Activity (Studio)**
4. Select **Absolute** layout
5. Select **Convert & Save**



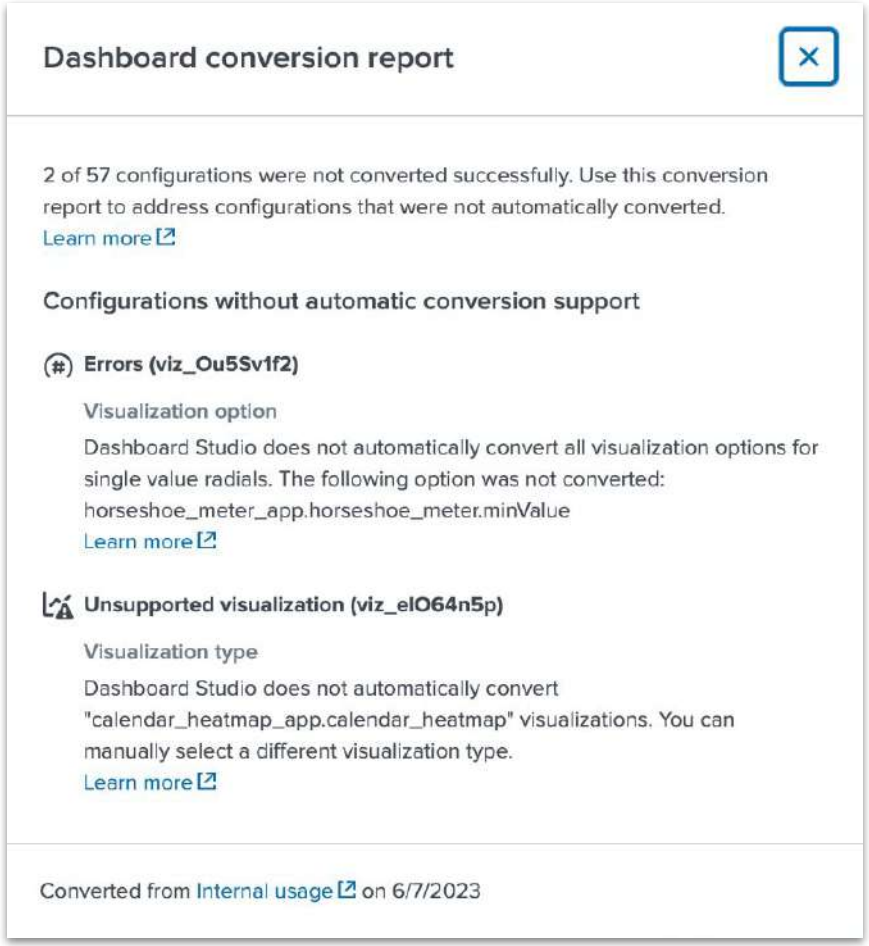


Note: The converted canvas size should be sufficient, but feel free to make the canvas larger as needed to more easily fit your visualizations

Conversion report

Upon conversion you probably saw a similar popup:

In this case, all conversions were converted, but if there were configurations that did not, you would be prompted to view a detailed report that provides more information about what exactly did not convert.

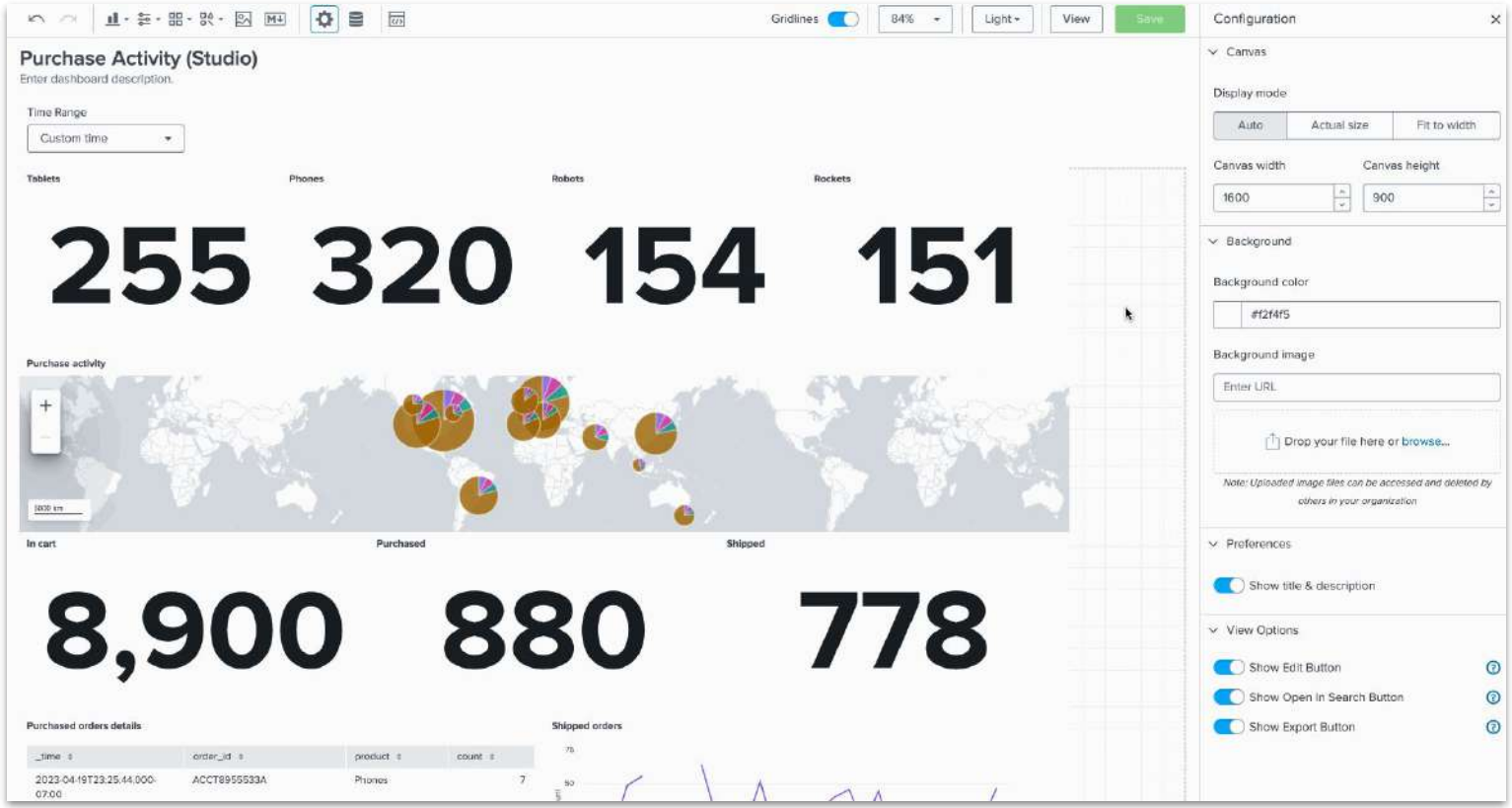


Example conversion report

Exercise 2: Configure Product Purchases Single Values

Resize product single values

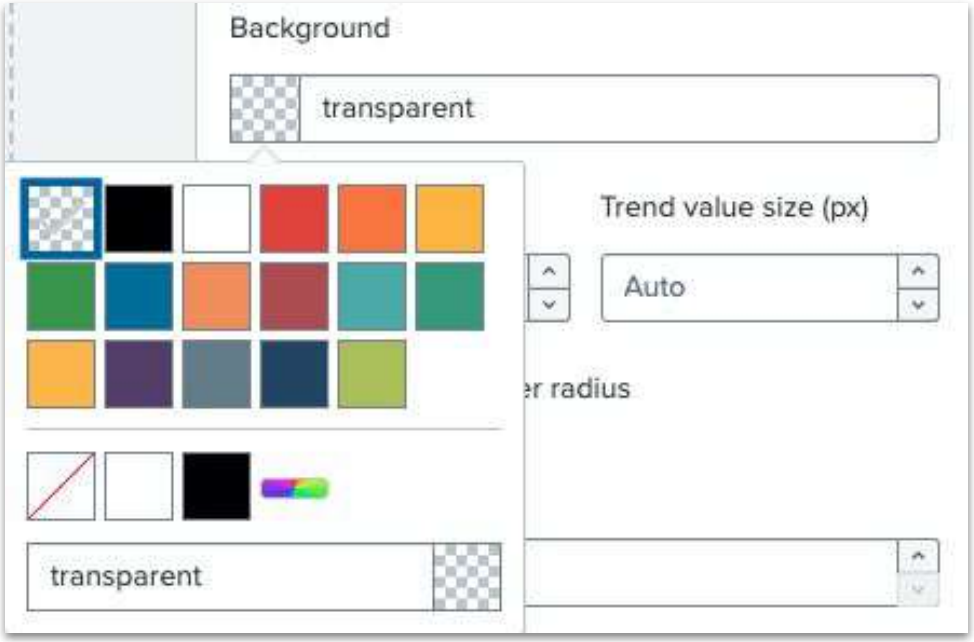
1. Multi-select the single values
   a. If using Mac:
      CMD + click
   b. If using Mac or Windows:
      click and drag to highlight
2. Bulk update size to be
   140 x 140



3. Set background color to transparent

Two ways to set the same setting for multiple objects

Use the UI under Color & Style to set Background color to transparent:

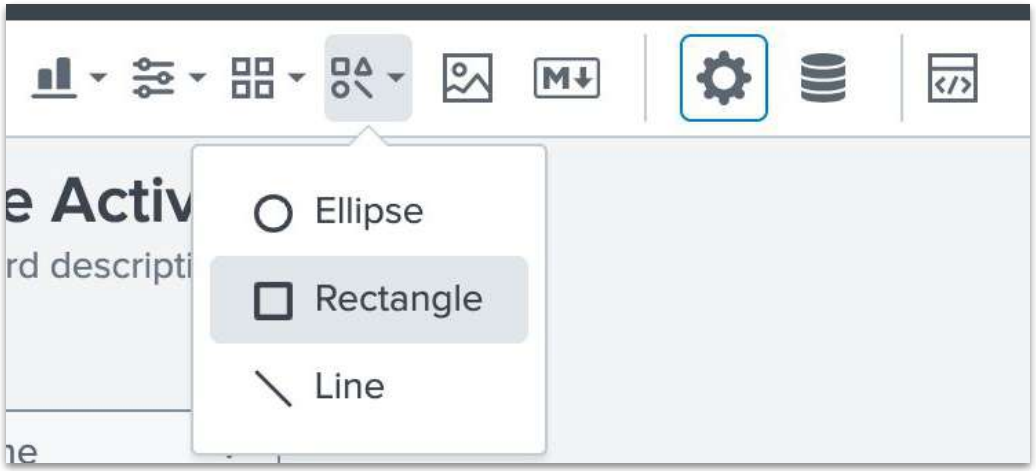


Copy/paste source code between objects:

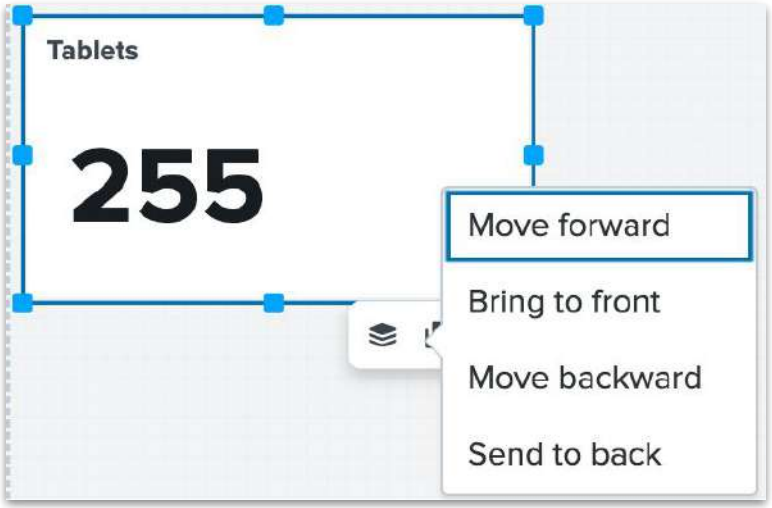
Rearrange the single values

4. Add a rectangle:
   - Width: **250**
   - Height: **140**
   - Fill color: **white**
   - Stroke color: **transparent**
5. Position behind a single value, using the layering options
6. Position the single value to be on the left, saving room for an SVG on the right later
7. Clone the rectangle to repeat steps 2 and 3 for the other KPIs

1

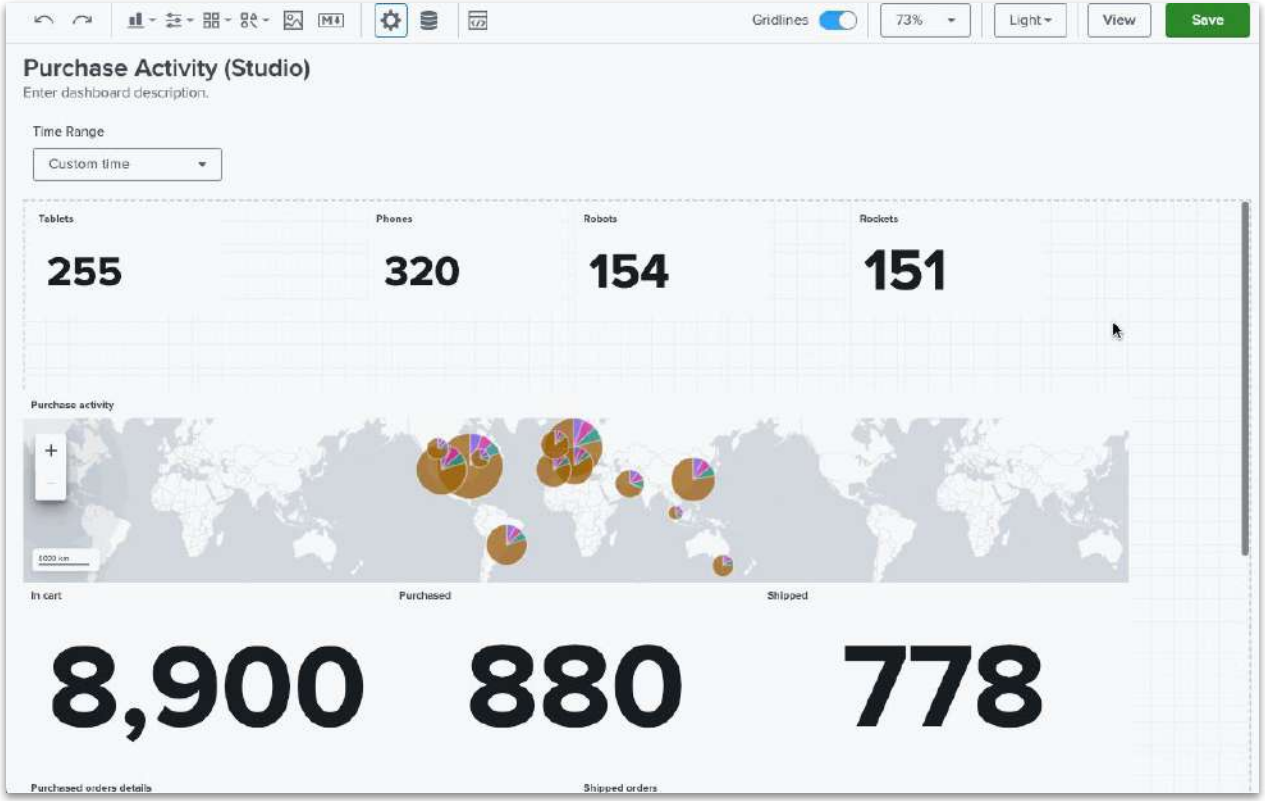


2



Reposition the single values

8. Reposition the single values so that:
   a. There is room above them for a custom label and description
   b. The single values take up the left two thirds width of the canvas
   c. There is a little bit of separation above the Purchase activity map



Add custom label and divider

9. Add a markdown text component and position it above the single values:

   # Product sales
   Electronics sold by count and region

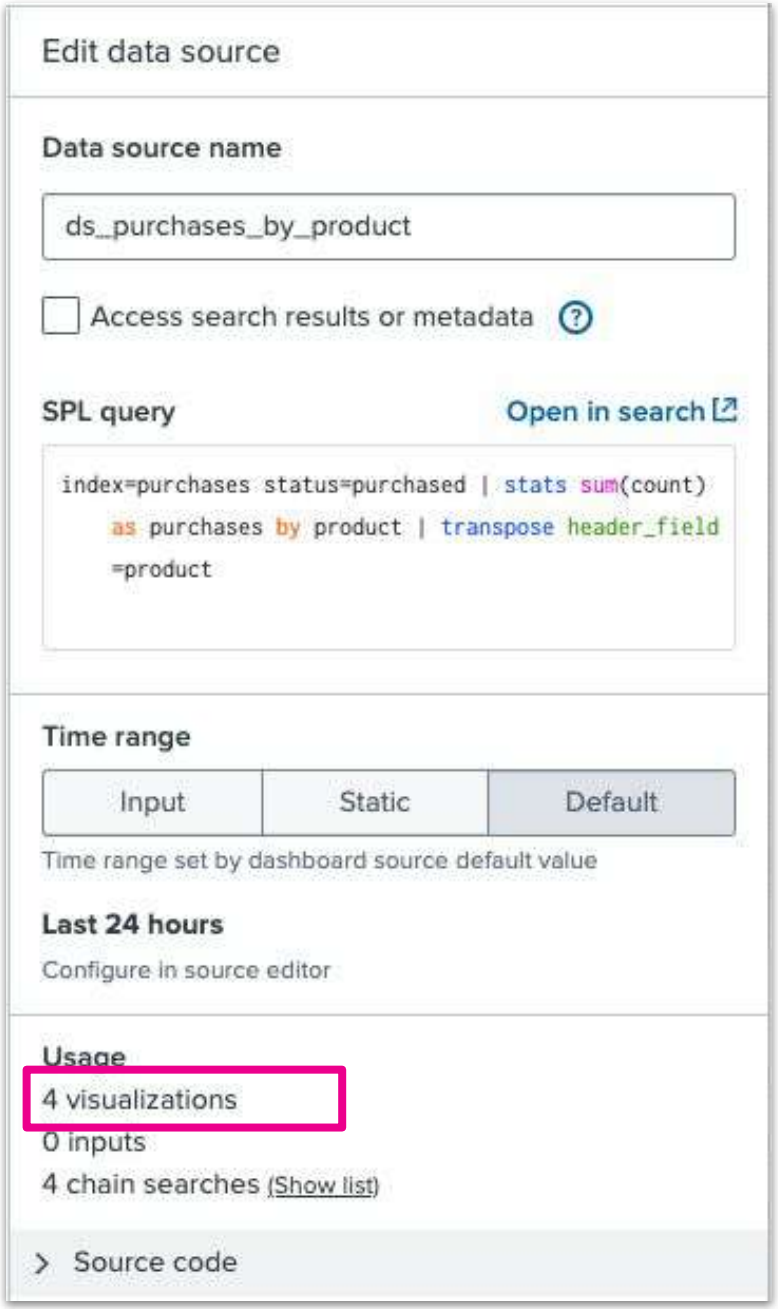
10. Add a rectangle to serve as a divider between the single values and map
    a. Set the stroke color to **transparent**
    b. Set radius to **10**

11. Position the rectangle between the single values and the map. The width should match the width of the 4 single value KPIs

Update the single values to use a single search

Dashboard Studio allows you to select the field you want to display from a given search. This allows us to consolidate the number of searches needed in the dashboard. Now we'll update each product search to **use ds_purchases_by_product**.

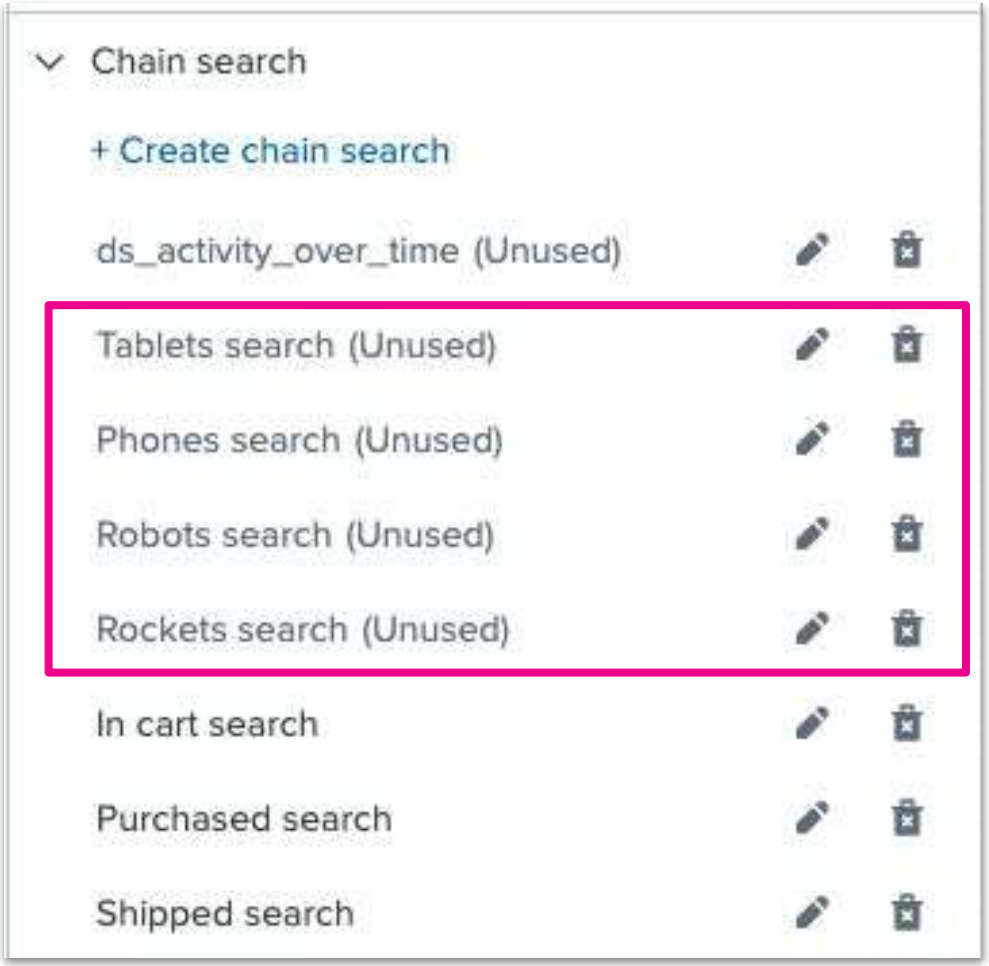
12. Select each product KPI, and remove its data source by clicking the trash can icon ( )
13. Select **+ Setup primary data source** and select the d**s_purchases_by_product** data source
14. Under Selected Data Field select the appropriate field for the single value



Delete the unused chain searches

Now that we don't need the individual chain searches for each product type, we can delete them.

15. Navigate to the **Data overview** panel by selecting the database icon in the toolbar
16. Under **Chain search**, delete the unused searches

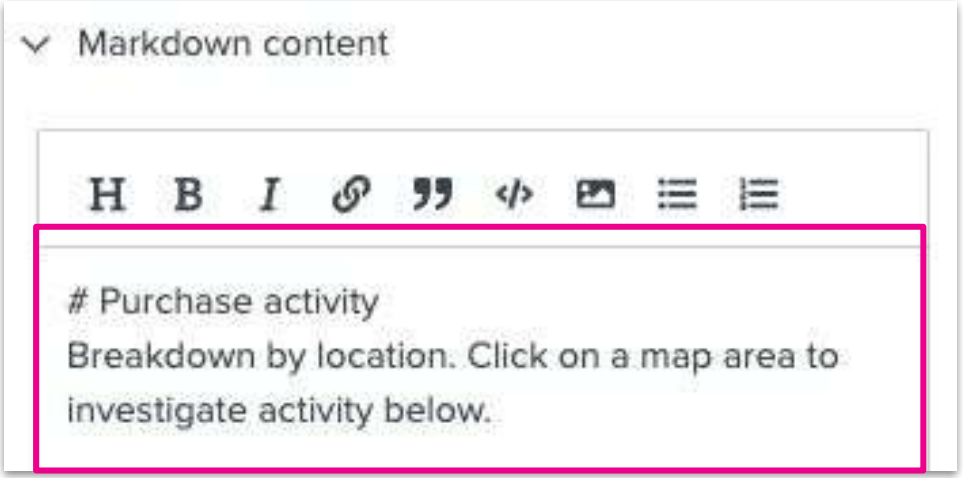


Checkpoint! Save your dashboard periodically so you don't lose your changes.

Exercise 3: Customize the map and add a link graph

Resize and reposition the map

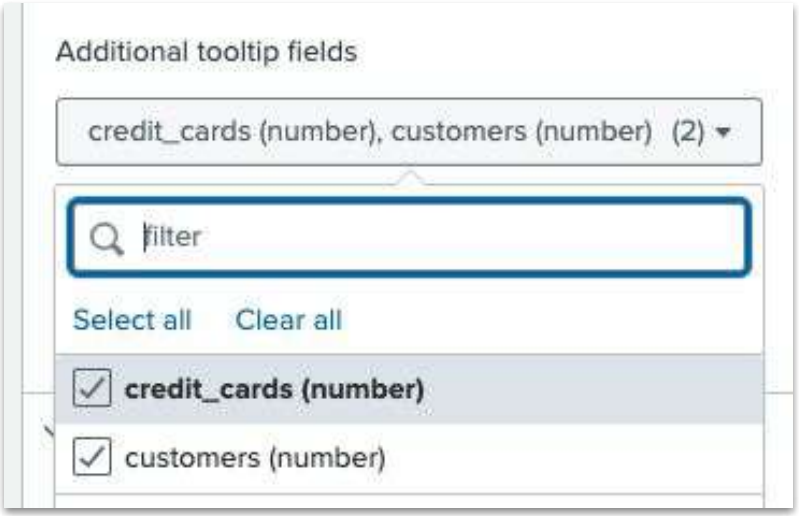
1. Resize the map width to match the divider, and resize the height to show more map (~300 px)
2. Move it down to make room for a custom label
3. Use Markdown to add a label:
   # Purchase activity
   Breakdown by location. Click on a map area to investigate activity below.
   a. You may need to move the other charts off the canvas to make room
4. Remove the map visualization title
5. Under **Data configurations**, under **Bubble size**, deselect all fields except for **products_purchased (number)**

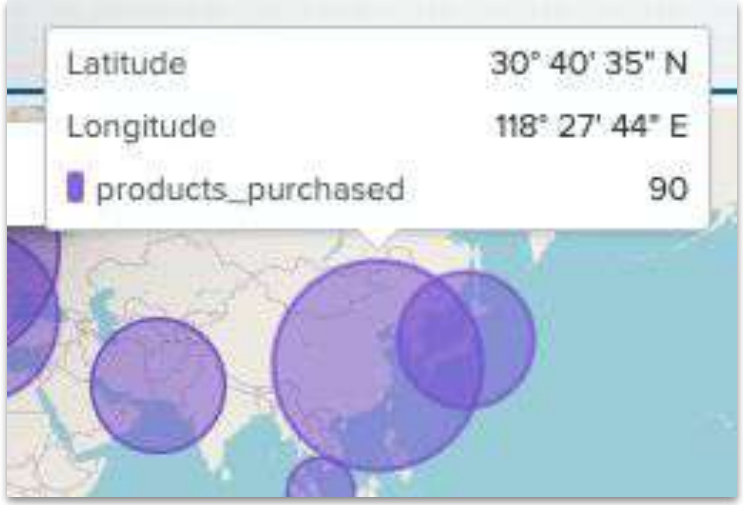


Add additional tooltip fields

You can specify additional field values to display in your map tooltips.

6. Under **Data configurations,** under **Additional tooltip fields,** select:
   a. credit_cards (number)
   b. customers (number)



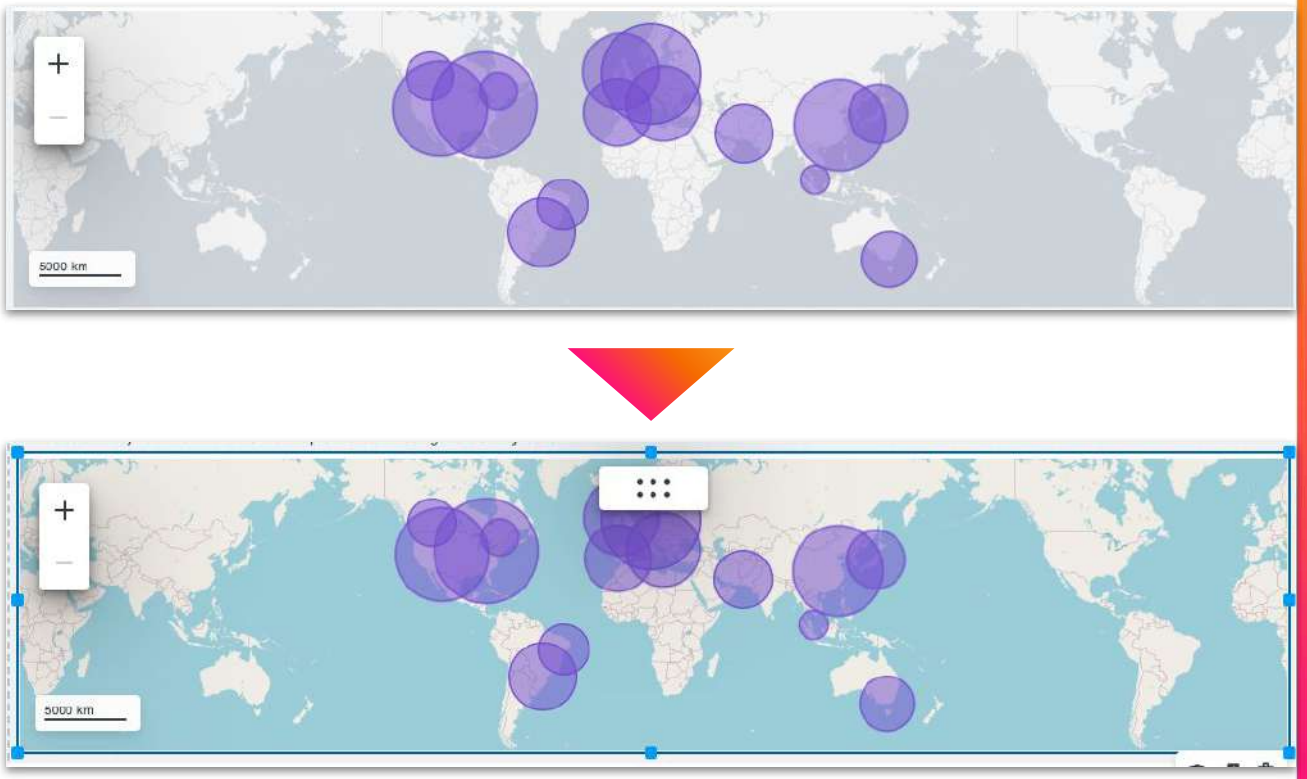


Add a custom tile server

About custom tile servers

- You must specify the tile server type (e.g. vector or raster)
- You can find map tile servers online
  - Examples: OpenStreetMap, OpenMapTiles, MapTiler, etc.

7. Under **Color and style**
   a. Set **Base layer tile server:** https://tile.openstreetmap.org/{z}/{x}/{y}.png
   b. Set **Base layer tile server type** to **Raster**



Set tokens on click

8. Under **Interactions**, click on **+ Add Interaction** and on the dropdown select **Set Tokens**
9. Specify the following, no default values:
   a. **Token name:** mapBoundsEast
      Token value: row._geo_bounds_east.value
   b. **Token name:** mapBoundsWest
      Token value: row._geo_bounds_west.value
   c. **Token name:** mapBoundsSouth
      Token value: row._geo_bounds_south.value
   d. **Token name:** mapBoundsNorth
      Token value: row._geo_bounds_north.value

Note: You can configure these drilldowns in the UI or copy the source code and paste it into the map's source code at the same level as "dataSources", "options", etc.

```
"eventHandlers": [
    {
        "type": "drilldown.setToken",
        "options": {
            "tokens": [
                {
                    "token": "mapBoundsNorth",
                    "key": "row._geo_bounds_north.value"
                },
                {
                    "token": "mapBoundsEast",
                    "key": "row._geo_bounds_east.value"
                },
                {
                    "token": "mapBoundsSouth",
                    "key": "row._geo_bounds_south.value"
                },
                {
                    "token": "mapBoundsWest",
                    "key": "row._geo_bounds_west.value"
                }
            ]
        }
    }
],
```

Add a link graph

10. Add a Link Graph under your map
11. Create a new search titled **"Activity in a highlighted area"**:

```
index=purchases status=purchased
| iplocation ip_address
| where (lat>=$mapBoundsSouth$ and lat<=$mapBoundsNorth$)
  and (lon>=$mapBoundsWest$ and lon<=$mapBoundsEast$)
| table customer_id ip_address credit_card
  billing_address status
```

12. Under **Visibility**, check the **"When data is unavailable, hide element"** box
13. Save, and try it out in View mode!

Hint: Select a map bubble to set tokens and observe the link graph's appearance

Configuring visibility

As of Splunk Enterprise 9.3, you can configure visibility based on whether the search returns results or not. However, you may want to apply other logic.

There are two ways to achieve this today:

1. Create a token which gets set to something valid when you want to display results, and something invalid when you don't want to display results (we'll do this in Exercise 4)
2. Add a pipe to the end of your SPL like:

   to only return results when the conditions inside the parentheses are met

```
| head limit=100 ($metric|s$ = "Occupancy" OR $metric|s$ = "*")
```

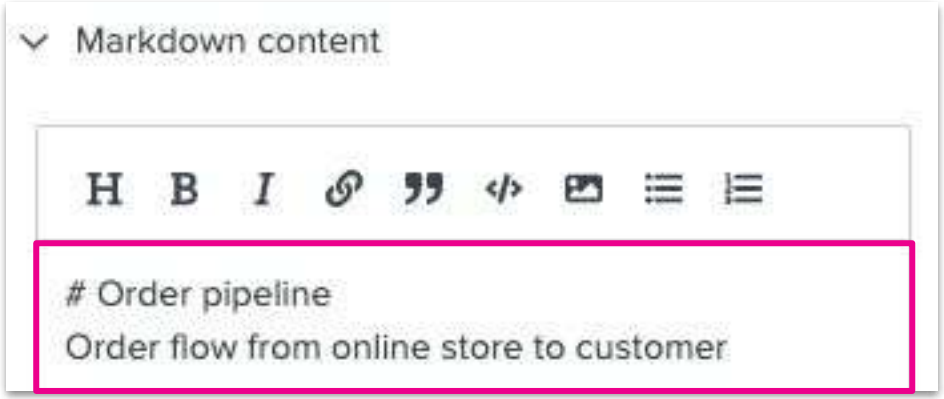
Check out this Splunk blog post or the Examples Hub for a full example.

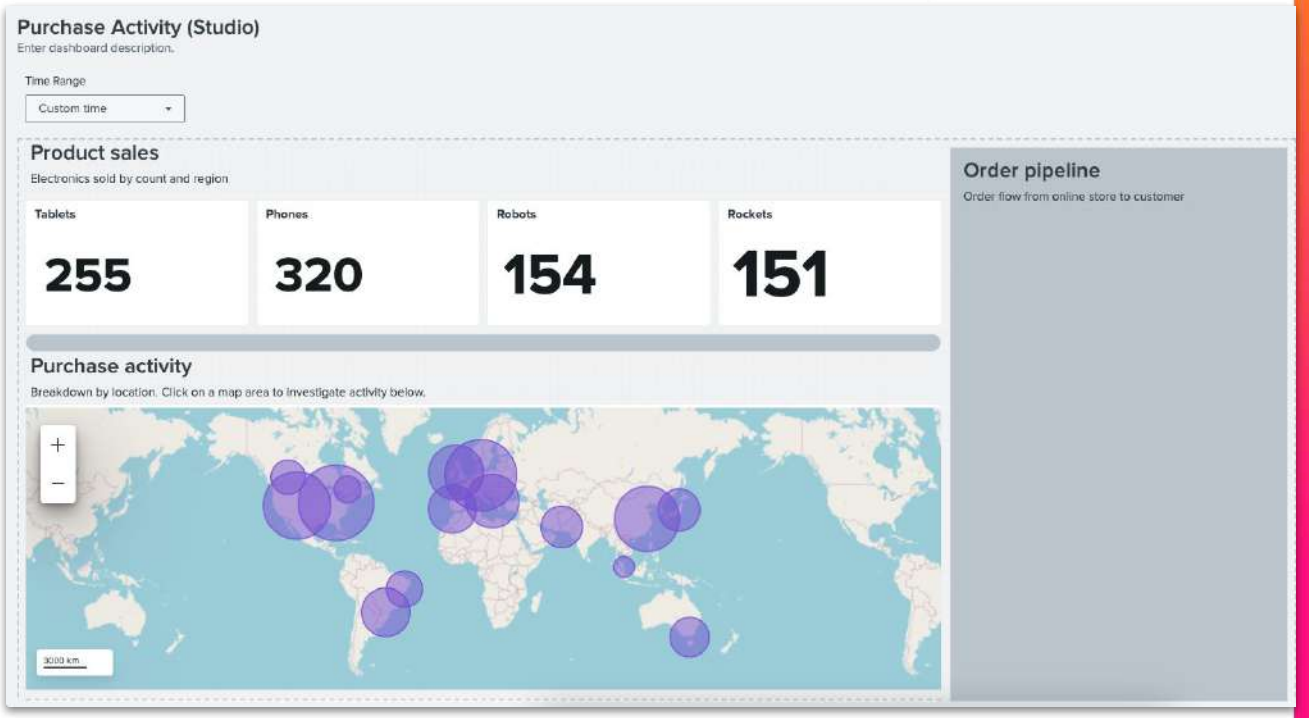
Exercise 4: Set up the conversion funnel

Create a section for the funnel

1. Add a rectangle to fill the remaining third of the dashboard
2. Set stroke color to **transparent**
3. Add a custom label with Markdown:

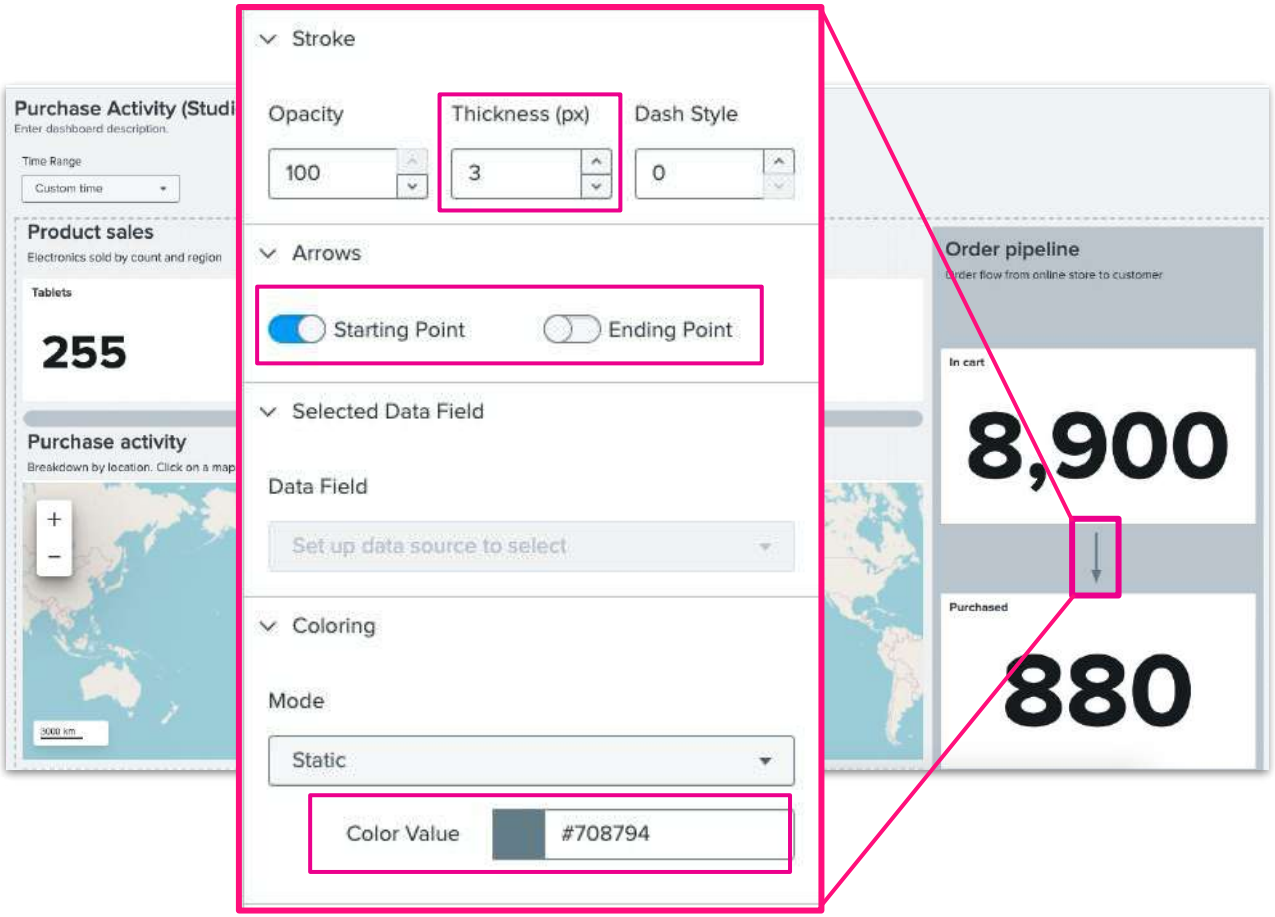
   # Order pipeline
   Order flow from online store to customer





Place order status KPIs

4. Assemble the order status KPIs (In cart, Purchased, Shipped) to the conversion funnel
5. Add arrows (line shape) in between them to indicate the flow
   a. Set **Thickness** to **3**
   b. Under **Arrows** select **Starting/Ending Point**
      i. Select whichever one will have the arrow pointing down
   c. Set **Color Value** to **gray**
6. Copy/paste the arrow and place it between the "**Purchased**" and "**Shipped**" KPIs



Set up an alternative "details" view

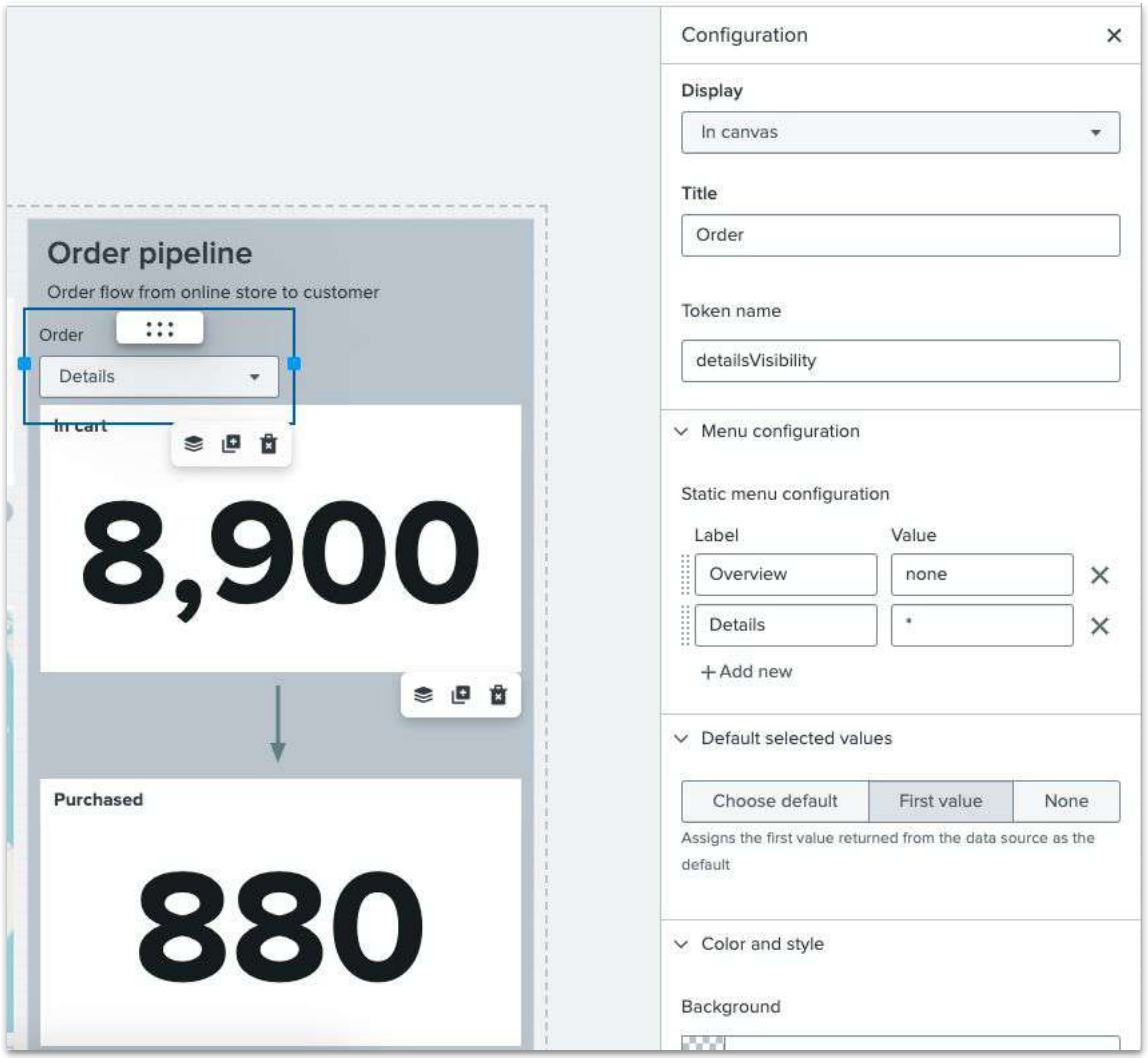
By default, users will see the 3 single value conversion funnel overview. We want to provide users a way to see more detailed information about each of those steps.

We will add a dropdown for users to choose between the "Overview" and "Details" views.

We will overlay detailed charts on the conversion funnel that will be hidden when users select "Overview" and visible when users select "Details"

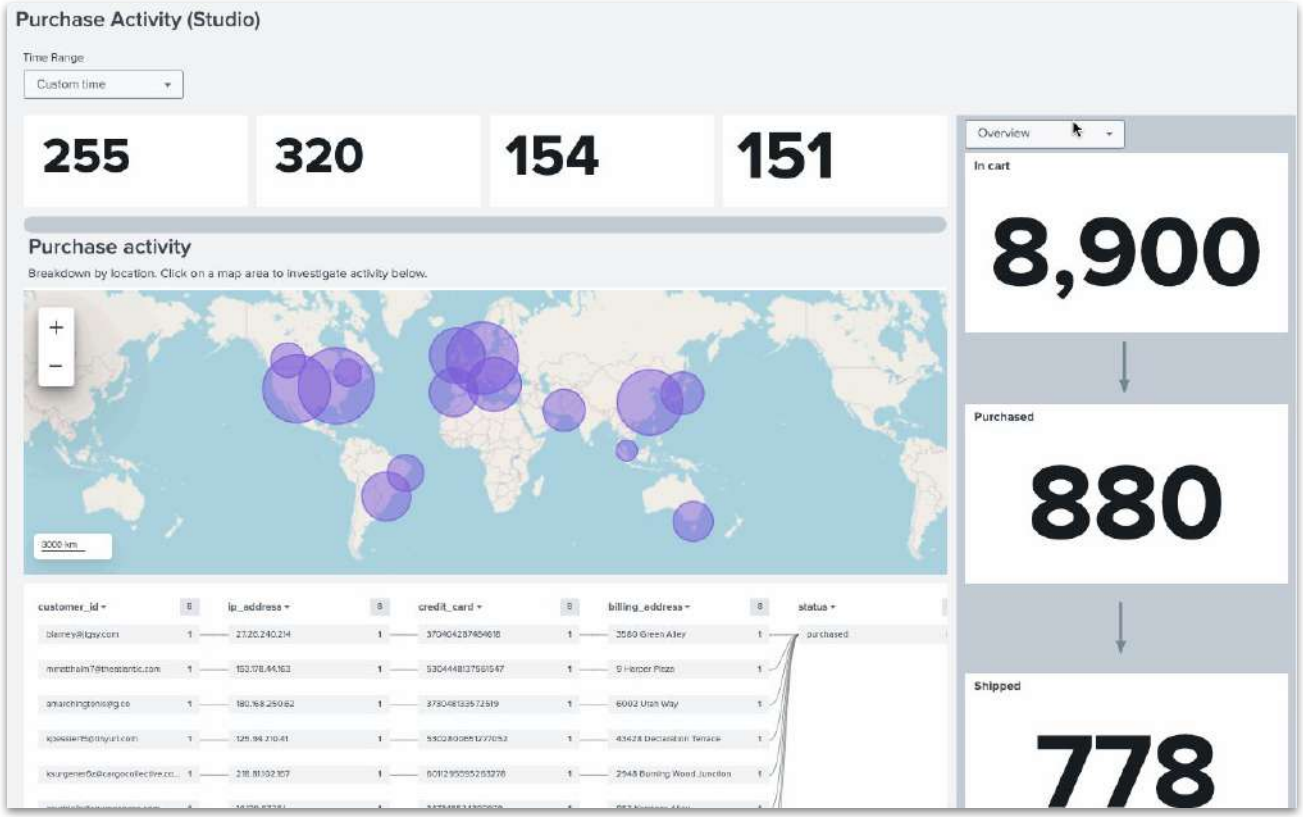
Set up an alternative "details" view

7. Add a dropdown input
8. Under **Display**, select "In canvas"
9. Position above the "In cart" single value
   a. Title: Order
   b. Token name: detailsVisibility
   c. Menu item 1:
      - Label: Overview
      - Value: none
   d. Menu item 2
      - Label: Details
      - Value: *
   e. Default selected values: First value



Set up an alternative "details" view

10. Place the "Purchased orders details" table over the "Purchased" single value
11. Place "Shipped orders" chart over the "Shipped" single value
12. Add **$detailsVisibility$** to the first pipe of each search
    a. The result should be "No search results returned". This is because the Orders dropdown is set to "Overview" which means $detailsVisibility$ is set to “none”.
    b. Update the Orders dropdown to "Details" to see the charts appear.
13. Under **Visibility**, check "**When no data is available, hide element**"

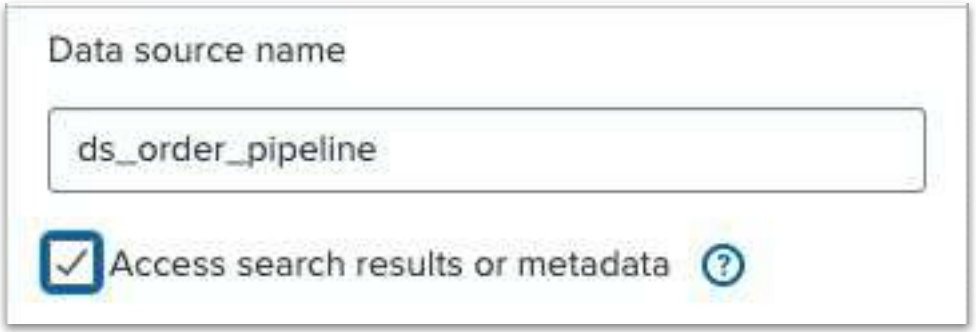


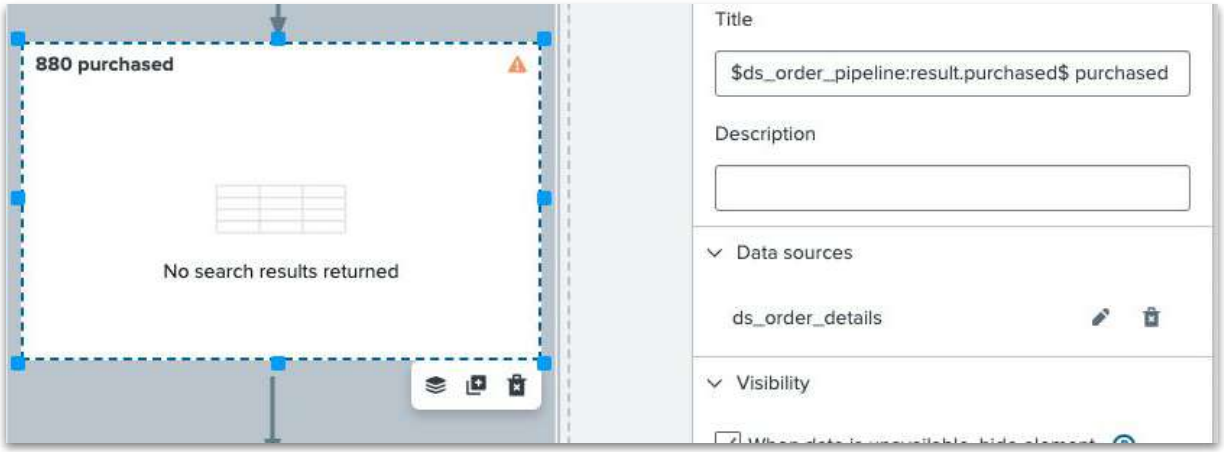
Go to View mode and switch between Overview and Details to test visibility

Set up an alternative "details" view

When viewing "details", we can't see the total number of purchased and shipped orders. Let's add them to the viz titles.

14. Navigate to the **ds_order_pipeline** search and check "**Access search results or metadata**". Make sure to select "Apply & close".
15. Update the table's title to: **$ds_order_pipeline:result.purchased$ purchased**
16. Update the line chart's title to: **$ds_order_pipeline:result.shipped$ shipped**





Checkpoint! Save your dashboard periodically so you don't lose your changes.

About search-based tokens

Search-based tokens allow you to reference a search's job metadata or the first row of results directly as tokens using the following syntax:

- $Data source name:job.<metadata>$
- $Data source name:result.<fieldname>$

For example, if we had a data source called **Order pipeline**, we could reference:

- Job ID using **$Order pipeline:job.sid$**
- Shipped orders status using **$Order pipeline:result.shipped$**

To learn more, check out the Dashboard Studio docs

About Dynamic Options Syntax

What is Dynamic Options Syntax (DOS)?

Also known as Splunk Visualizations' domain-specific language (DSL)

DOS enables precise customization during configuration of components in Studio

```
"someOption": "> someDataSource | someSelectors(...) | someFormatter(...)"
```

majorColor

primary

seriesByName('count')
lastPoint()

rangeValue(someColorContext)

Always starts with ">"

Sections separated using "|"

For example, instead of using a static hex code such as "#FFFFFF" to color the major value in a Single Value component, you can use dynamically-powered range value thresholds: 0-40=red, 40-70=yellow, 70-100=green that depend on a specified field hidden from view.

What is Dynamic Options Syntax (DOS)?

Also known as Splunk Visualizations' domain-specific language (DSL)

DOS enables precise customization during configuration of components in Studio

```
"someOption": "> someDataSource | someSelectors(...) | someFormatter(...)"
```

majorColor	primary	seriesByName('count') lastPoint()	rangeValue(someColorContext)

The configuration option is the aspect of the visualization component you'd like to change, for example:

- The major value's color in a Single Value
- The y-axis' field selections in a Line chart

What is Dynamic Options Syntax (DOS)?

Also known as Splunk Visualizations' domain-specific language (DSL)

DOS enables precise customization during configuration of components in Studio

```
"someOption": "> someDataSource | someSelectors(...) | someFormatter(...)"
```

majorColor	primary	seriesByName('count') lastPoint()	rangeValue(someColorContext)

The first part of the DOS string is the source of data or values you'd like to to dynamically power the configuration, for example:

- The primary, or annotation data source
- Another configuration option to use the exact same value

What is Dynamic Options Syntax (DOS)?

Also known as Splunk Visualizations' domain-specific language (DSL)

DOS enables precise customization during configuration of components in Studio

```
"someOption": "> someDataSource | someSelectors(...) | someFormatter(...)"
```

majorColor	primary	seriesByName('count') lastPoint()	rangeValue(someColorContext)

The second part of the DOS string is often a filter for the data source (and is sometimes optional), for example:

- Selecting the field named "count", and then filtering further to select the "count" column's last point
- Selecting only the numeric columns

Note: these require single quotation marks `'` or escaped double quotation marks `\"` around the selection parameters

What is Dynamic Options Syntax (DOS)?

Also known as Splunk Visualizations' domain-specific language (DSL)

DOS enables precise customization during configuration of components in Studio

```
"someOption": "> someDataSource | someSelectors(...) | someFormatter(...)"
```

majorColor	primary	seriesByName('count') lastPoint()	rangeValue(someColorContext)

The third part of the DOS string is often a formatter for the data source (and is sometimes optional), for example:

- Using range value thresholds to color, which are referenced using a variable named defined in the context section of the stanza
- Adding a unit prefix

Dynamic Options Syntax (DOS) Example

Here's what the example we just walked through would look like in source:

```
"viz_fWZQq3mP": {
    "type": "splunk.singlevalue",
    "options": {
        "majorColor": "> primary | seriesByName('count') | lastPoint() | rangeValue(majorColorEditorConfig)"
    },
    "dataSources": {
        "primary": "ds_AZuF9nvk"
    },
    "context": {
        "majorColorEditorConfig": [
            {
                "to": 0,
                "value": "#D41F1F"
            },
            {
                "from": 0,
                "value": "#118832"
            }
        ]
```

Exercise 5: Color table columns by hidden values with DOS

Color table columns by hidden values

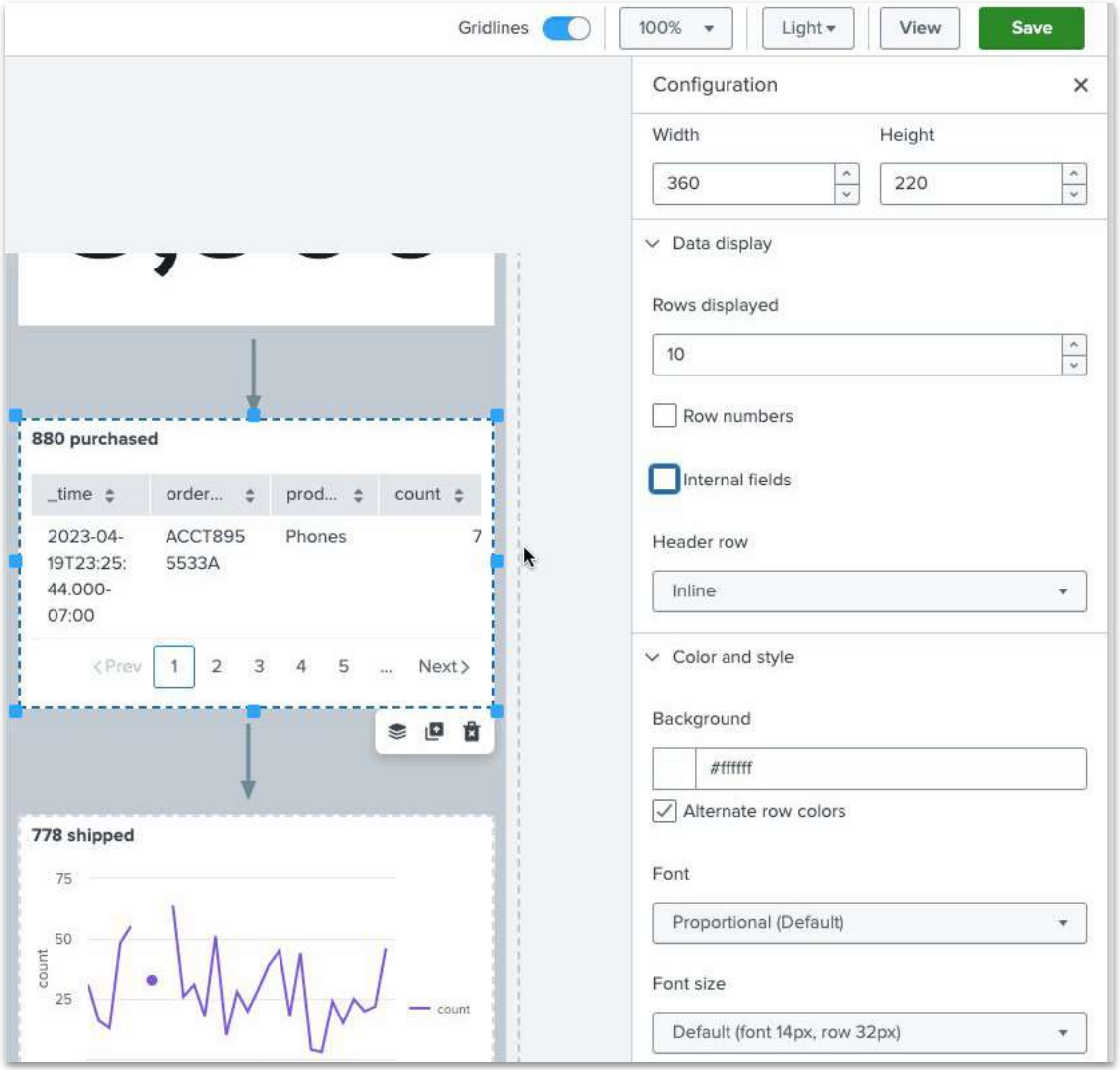
We want to color the **count** column red or green depending on whether there are more of that product in stock, which is driven by the field **_has_stock.**

_has_stock is hidden by default because the underscore (_) indicates that it should be treated as an internal field, and under Global Formatting, "Internal Fields" is unchecked by default.

You can check the box to see **_has_stock** in the table.

Even though the **_has_stock** field is hidden, we can still use it to determine the color of the values in the **count** column

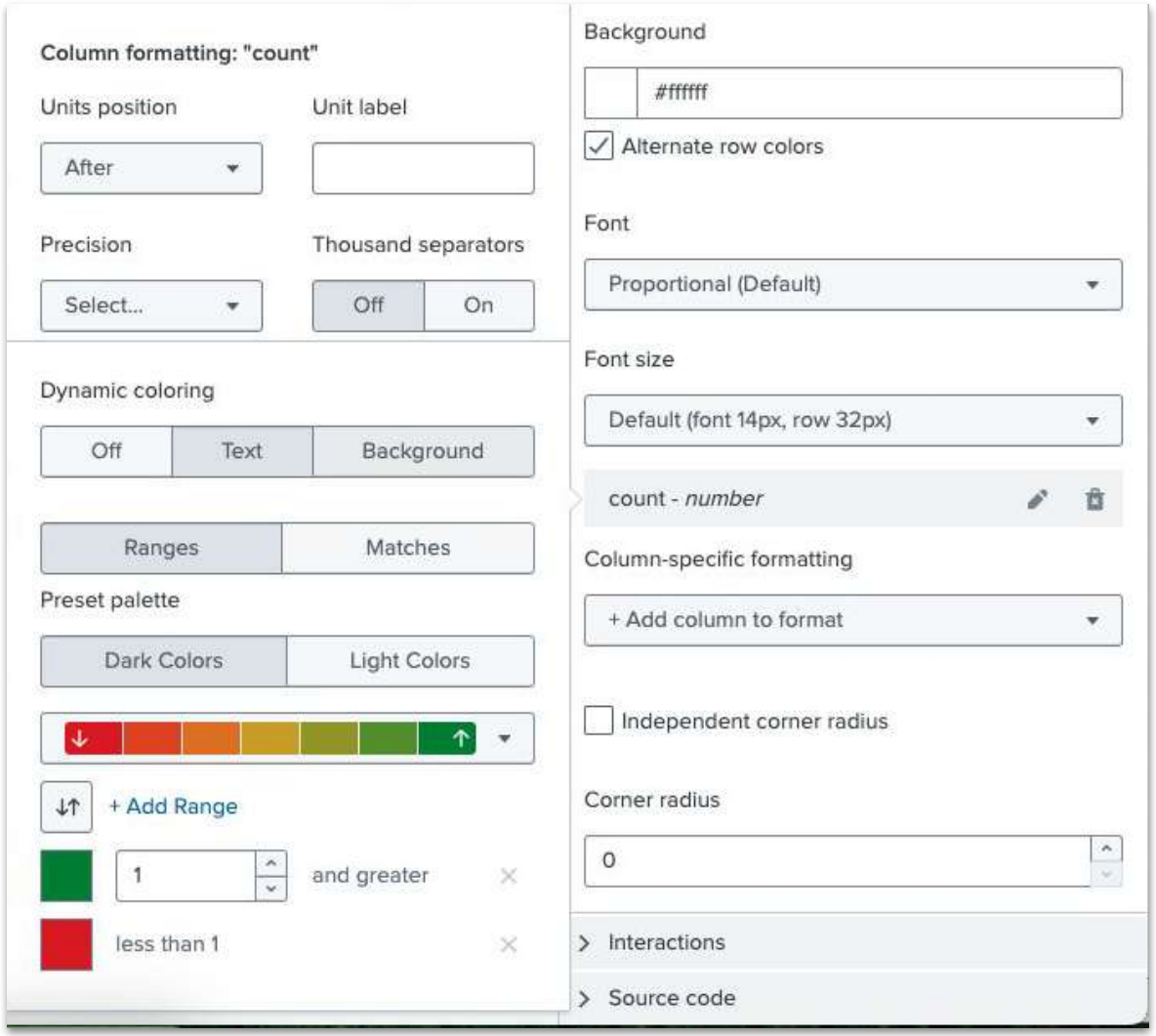
Note: You may need to adjust the table columns or scroll horizontally to view all columns. Ensure "Details" is selected in the dropdown



Applying color to "count"

As a best practice, configure as much as possible in the UI **first** before modifying the source code to minimize the chances of syntax errors.

1. Under **Column-specific formatting**, add the **count** column to format
2. Under **Dynamic coloring** select **Text**
3. Apply ranges so that **1 and greater** is **green**, and **less than 1** is **red**



Update the DOS to use _has_stock

So far we've configured the **count** column to be green when **count** is greater than 0 and red when 0.

What we want is to configure **count** to be green when **_has_stock** is greater than 0 and red when 0.

4. In source, under **"options"** > **"columnFormat"** > **"count"** > **"rowColors",** change **count** to **_has_stock**

Your source code should look like this:

```
2    "type": "splunk.table",
3    "options": {
4        "columnFormat": {
5            "count": {
6                "data": "> table | seriesByName(\"count\") | formatByType(countColumnFormatEditorConfig)",
7                "rowColors": "> table | seriesByName(\"_has_stock\") | rangeValue(countRowColorsEditorConfig)"
8            }
```

count powers the data configuration option

_has_stock powers the rowColors configuration option

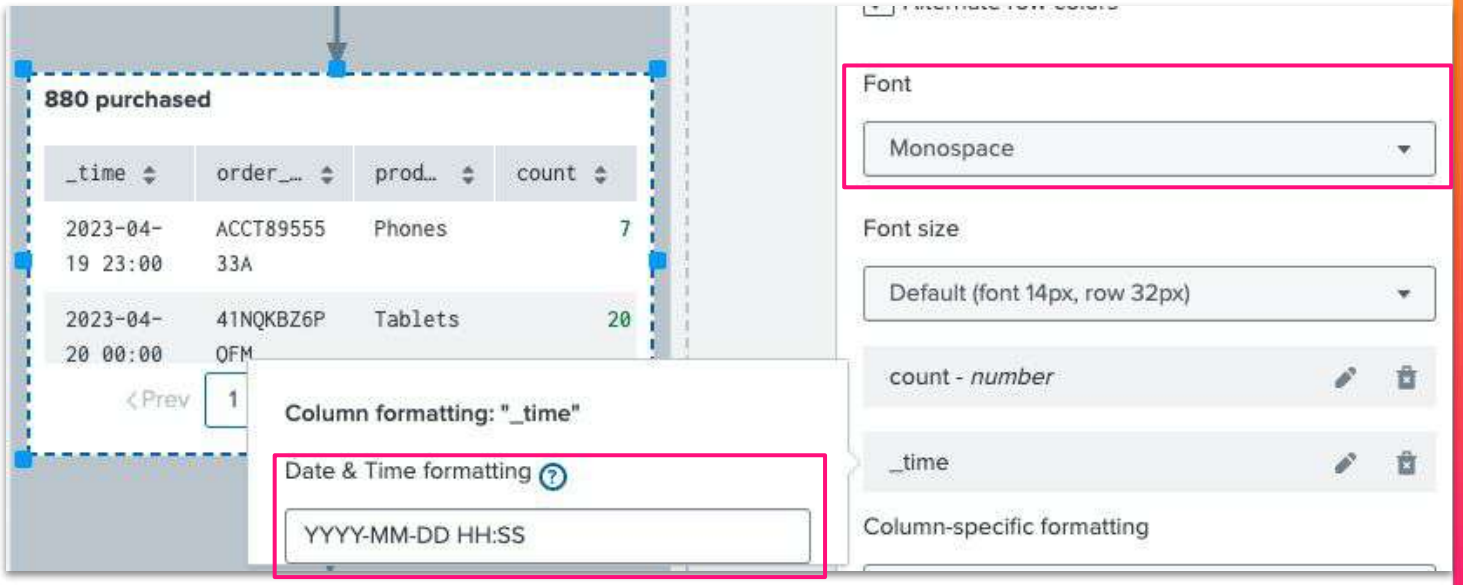
Update the DOS to use _has_stock

Your table should now have values in **count** that are red or green, depending on the value of the **_has_stock** field

Finish formatting the table

5. Under **Color and Style** change Font to "Monospace"
6. Under **Column-specific formatting**, select _time and apply the format:

   YYYY-MM-DD HH:SS



Exercise 6: Add secondary data sources to line charts with DOS

Beautify the "Shipped orders" line chart

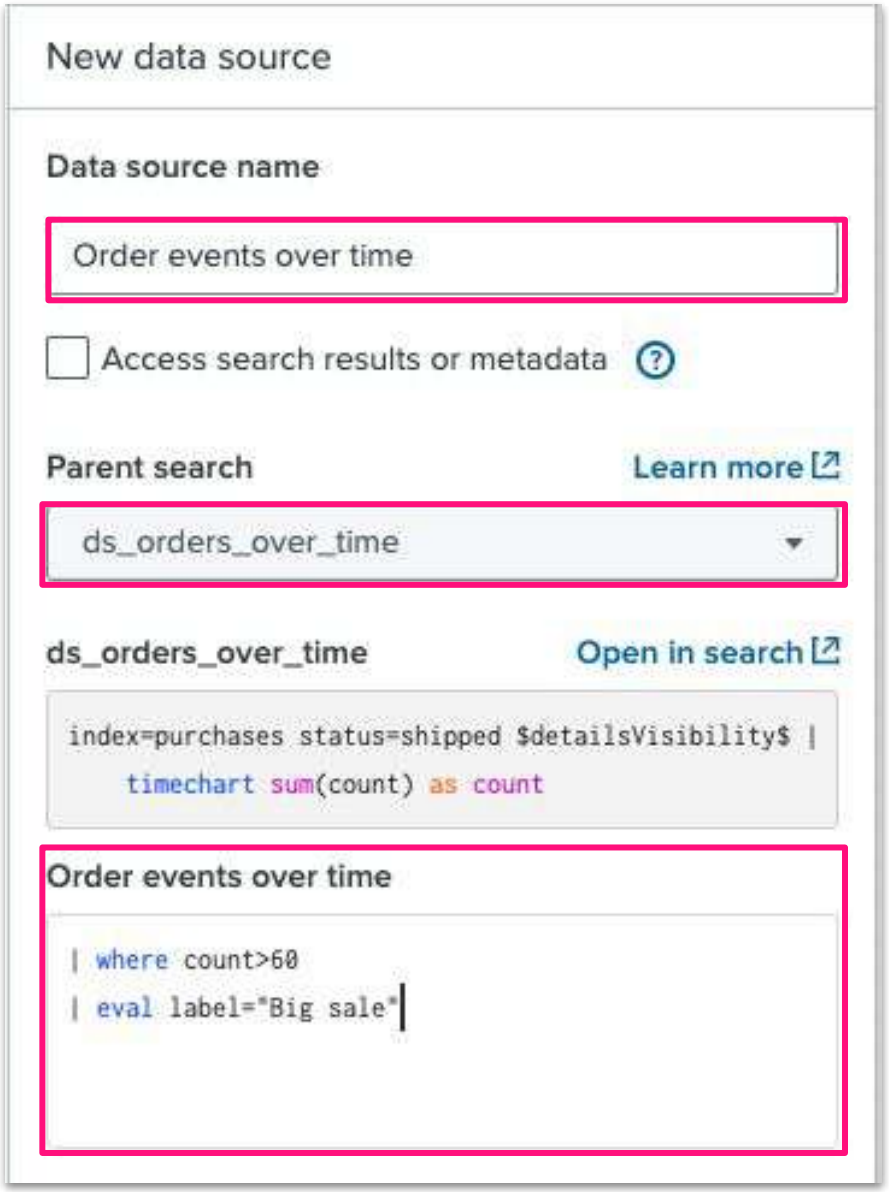
1. Under **Data display**, change **Null value display** to "**Connect**"
2. Under **Color and style**, add a series to **Series color by field name**. Select the field name **count** and apply the color **#118832**
3. Under **Legend** change **Legend display** to "**Off**"
4. Under **X-axis grid and labels**, deselect **Axis title**, and change **Number of time label parts** to 1
5. Under **Y-axis grid and labels,** specify **Axis label text** as "**Shipped**"

Add an annotation data source

Use annotations to add context to your charts. Ensure your annotation data source returns fields that corresponds to the chart's x-axis (usually _time), the label you want to display, and, optionally, a color field.

6. Select the line chart. Under **Data sources**, click on **+ Set up Annotation Data Source**
7. Create a new chain search and click on **Apply & close**
   a. Name: **Order events over time**
   b. Parent search: **ds_orders_over_time**
   c. Chain SPL:

```
| where count>60
| eval label="Big sale"
```



Configure the chart annotation

Under **Data configurations:**

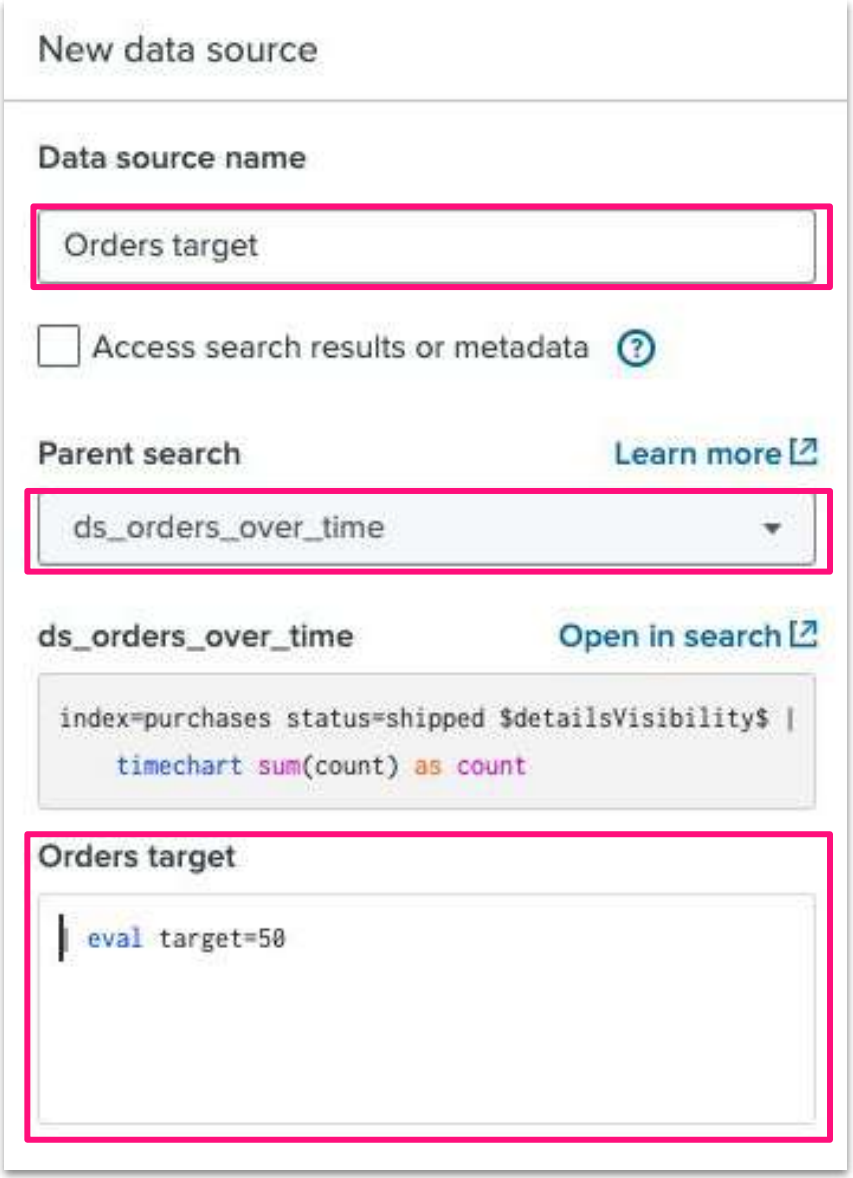
8. Select **Annotation x** to be "**_time (time)**"
9. Select **Annotation labels** to be "**label (string)**"

Now in **View** mode, you can hover on the annotation to see the label

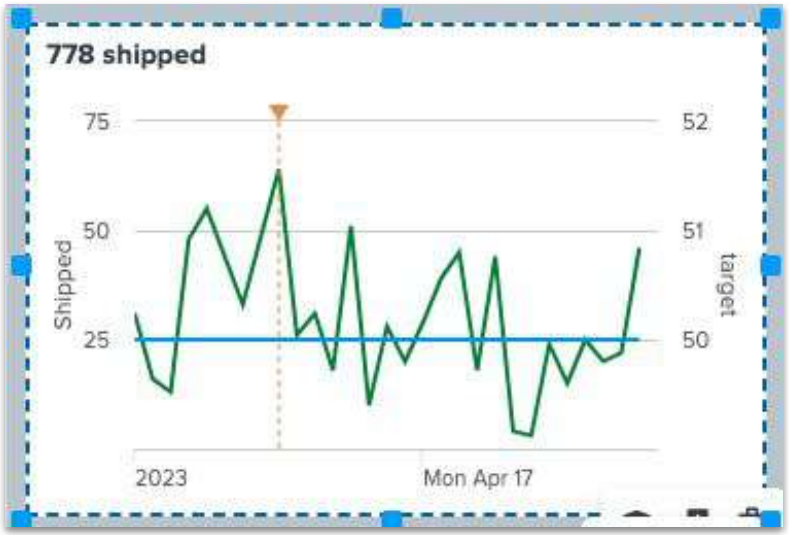
Create a search for "target" orders

We want to see whether our shipped orders are meeting our targets, so we will overlay a horizontal line on the line chart as a secondary data source.

10. Navigate to the Data Overview
11. Create a new chain search:
    a. Name: **Orders target**
    b. Parent search: **ds_orders_over_time**
    c. Chain SPL: | eval target=50
12. Open the code editor and copy the data source ID
13. Click on **Apply & Close**



Use DOS to add target data source



Let's add the target as a secondary data source and display it using the y2 axis.

14. Select the line chart and open the code editor
15. Under **"dataSources"** add the ID you just copied from the last step:

    "secondary": "ds_<your ID>"

16. Under **"options"** add:

    "y2": "> secondary | frameBySeriesNames('target')"

```
14        "xAxisTitleVisibility": "hide",
15        "xAxisMaxLabelParts": 1,
16        "yAxisTitleText": "Shipped",
17        "annotationX": "> annotation | seriesByName('_time')",
18        "annotationLabel": "> annotation | seriesByName('label')",
19        "y2": "> secondary | frameBySeriesNames('target')"
20    },
21    "dataSources": {
22        "primary": "ds_orders_over_time",
23        "annotation": "ds_OPHAbljY",
24        "secondary": "ds_TEn8e0aK"
25    },
```

Checkpoint! Save your dashboard periodically so you don't lose your changes.

Exercise 7: Color map bubbles with DOS

Search for abnormal activity

We want to color our map bubbles green for normal credit card activity, and red for abnormal credit card activity (more customers than credit cards in use)

1. Update the map data source to create a new field to designate abnormal activity by appending:

```
| fillnull
| eval abnormal_activity=if(customers > credit_cards, 1, 0)
```

2. Click on **Apply & close**

Define coloring rules in source

3. Under **"context"** create a section called **thresholdConfig:**

```
"context": {
    "thresholdConfig": [
        {
            "to": 1,
            "value": "green"
        },
        {
            "from": 1,
            "value": "red"
        }
    ]
}
```

Apply colors using DOS

4. Under **"options"** under **"layers"** add the following:

```
"dataColors": "> primary | seriesByName('abnormal_activity') | rangeValue(thresholdConfig)",
```

Option name to color map bubbles

Reference the field that will determine the color

Format using rangeValue, as defined in thresholdConfig

Your map should now be colored like this

Exercise 8: Add custom SVGs

What is an SVG?

More documentation about SVGs can be found here

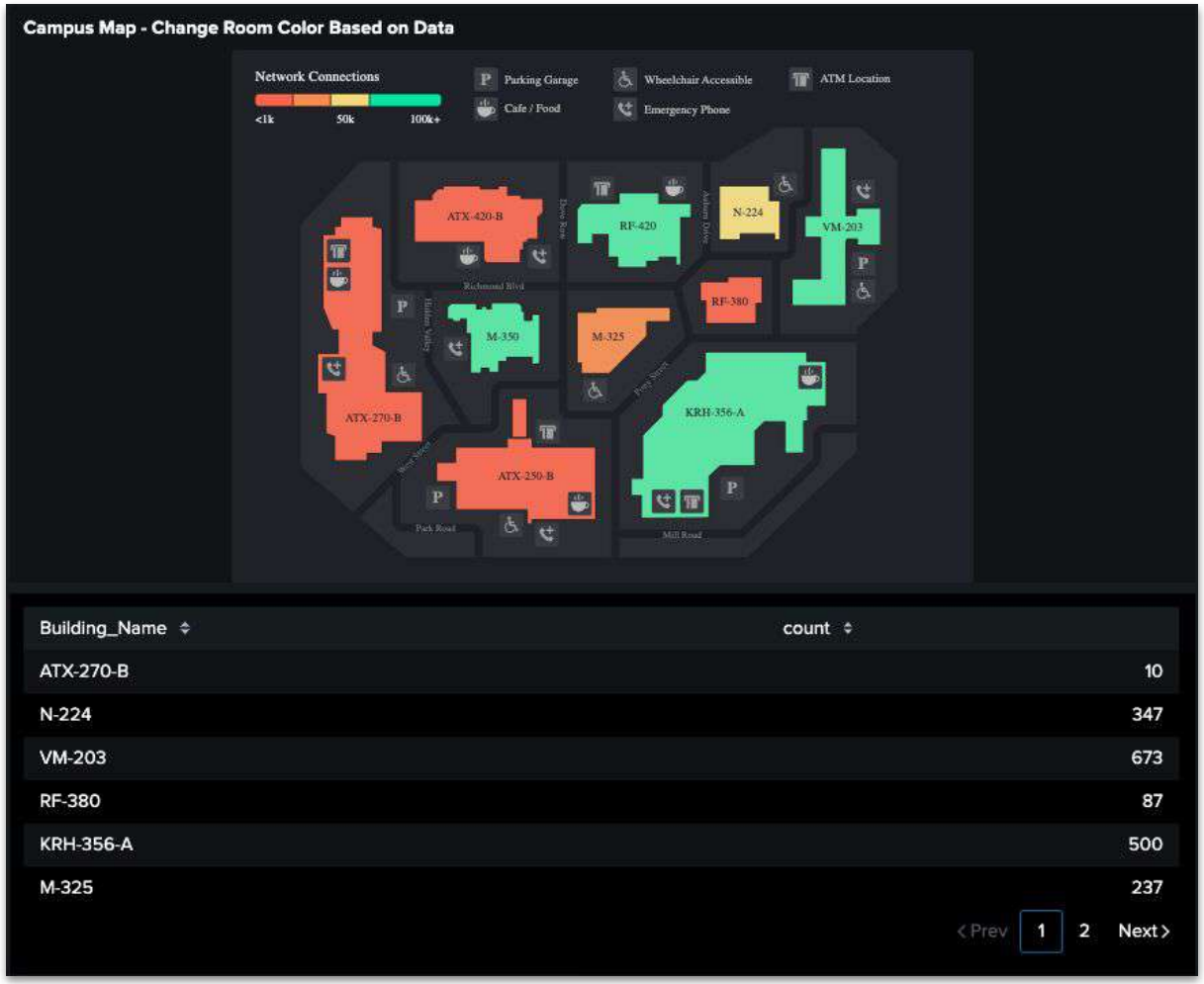
- Scalable Vector Graphics (SVG) is an XML-based markup language for describing two-dimensional based vector graphics
- SVGs can be used for dynamic visualizations in Dashboard Studio, for example:
  - Dynamically coloring a floor plan based on occupancy
  - Interactions like hovering over an SVG to display more information
  - If you can dream it, you can probably build it
- There are many online resources and tools to help you create your own SVGs, which you can then add to Dashboard Studio

SVGs in the Examples Hub

You can find multiple SVG examples in the Examples Hub, such as this campus map.

The campus map is an SVG, where each building can be dynamically colored based on search results.

In this example, the buildings are colored based on how many active connections there are per building.



Anatomy of an SVG

```
<svg xmlns="http://www.w3.org/2000/svg" width="326" height="415" viewBox="0 0 326 415" fill="none">
  <path id="tablet" stroke-width="8.2029" stroke="#c3cbd4" d="M93.7386 64.891C93.5076 64.891 81.8406
  73.557 6882 78.6381L28.961 212.709C27.4598 217.79 27.6907 223.333 29.5384 228.414C31.2706 233.
  628C222 351.284 226.861 351.315 231.6 350.268C231.6 350.268 241.211 348.313 245.484 346.581C24
  872 203.156C300.336 192.647 295.717 179.714 286.825 174.286L107.371 66.1976C102.752 63.3107 97.901
</svg>
```

1. The `<path>` element is where we will define the shape (see docs for more details)
2. The `id` attribute assigns a unique name to an element, which should map to a search result field name to be able to color dynamically (see previous slide example)
3. The `d` attribute is what actually defines the path to be drawn

Options for adding SVGs to dashboards

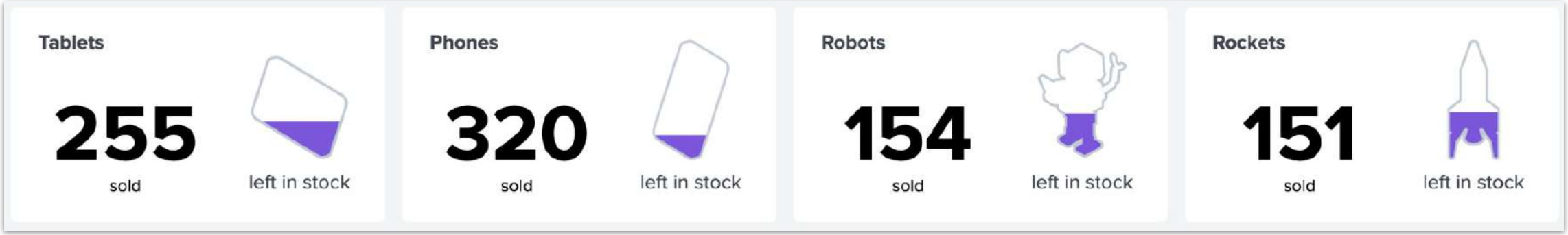
Upload your SVG	Reference via URL	Embed inline SVG
Best when: ● Your search will return fields that map to the SVG path ids ● You don't need to move your dashboard to another environment	**Best when:** ● You want to reuse an SVG in different environments ● For Splunk Cloud 9.2.2403 or Splunk Enterprise 9.3 or higher, you can upload to your app's static folder	**Best when:** ● You want to reuse an SVG in different environments ● You want to embed token values in your SVG definition
Gotchas to look out for: ● Uploaded SVGs are placed in the kv-store, so will not transfer if you move your dashboard	**Gotchas to look out for:** ● For custom SVGs before Splunk Cloud 9.2.2403 or Splunk Enterprise 9.3, you'll need a webserver to host the SVG	**Gotchas to look out for:** ● Most difficult set up ● Difficult to read in the source editor

Using inline SVGs

- To embed an in-line SVG, you will put the entire SVG definition into the **"svg"** option of your visualization
- These SVGs must be **escaped**, meaning they cannot have new lines characters or double quotes.
  - You can use find and replace in a text editor to replace `\n` with **nothing** and replace `"` with `\"`
- Note that if you have an escaped SVG definition, it will technically not be valid per validation rules, but it will work in Dashboard Studio
- In the next exercise, we will add custom, escaped SVGs to our dashboard

Add inline SVGs to your dashboard

We are going to add product SVGs that will dynamically fill with a purple color as more product is sold.



Create the inventory data source

1. Create a new search that will return what % of product is left in inventory
   a. Name: **Inventory totals**
   b. SPL:

```
| makeresults count=1
| eval inventory_tablets=400
| eval inventory_phones=400
| eval inventory_robots=250
| eval inventory_rockets=250
| eval current_tablets=
  (inventory_tablets-$ds_purchases_by_product:result.Tablets$)/inventory_tablets*100
| eval current_phones=(inventory_phones-$ds_purchases_by_product:result.Phones$)/inventory_phones*100
| eval current_robots=(inventory_robots-$ds_purchases_by_product:result.Robots$)/inventory_robots*100
| eval current_rockets=
  (inventory_rockets-$ds_purchases_by_product:result.Rockets$)/inventory_rockets*100
| table _time current_tablets current_phones current_robots current_rockets
```

   c. Check the box that says "Access search results or metadata"

Click on **Apply & Close**

Create the inventory data source

You may have noticed that the search we just created uses tokens from the search **ds_purchases_by_product** (example: $ds_purchases_by_product:result.Tablets$)

In order to use those search results, we need to enable "**Access search results or metadata**" on **ds_purchases_by_product**

2. Edit the **ds_purchases_by_product** data source and check the "**Access search results or metadata**" **box**

   Click on **Apply & Close**

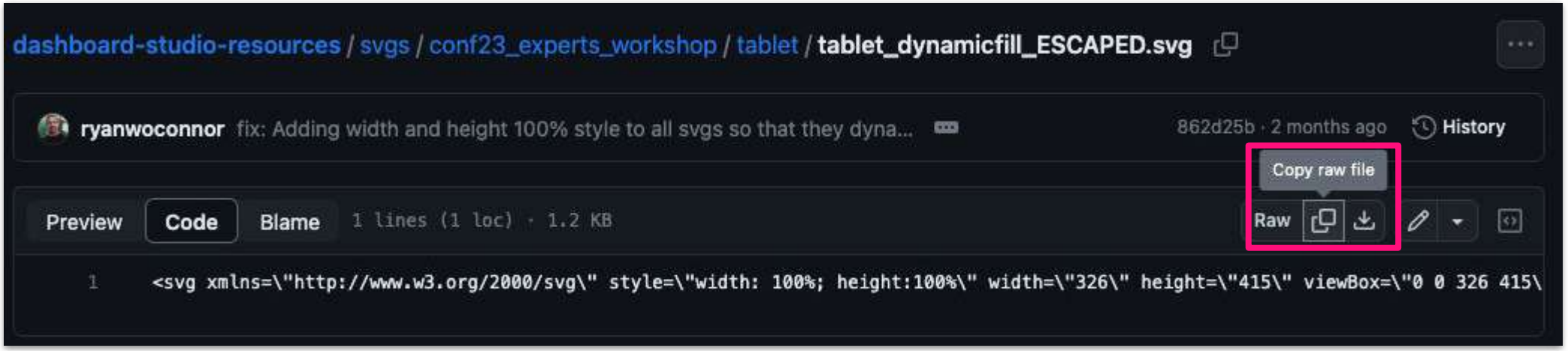
Note: You may need to save and refresh the dashboard to see the SVGs fill.

Copy the SVG definition

You will find the definitions for each of the SVGs in our public GitHub repo

For the sake of today's exercise, here are the links for Tablet, Phone, Robot and Rocket

3. Copy the raw SVG code for the Tablet SVG using the "Copy raw file" button:



Add an SVG to your dashboard

4. Add a Choropleth SVG and place it next to a single value KPI
5. When prompted to select a data source, select "**Cancel**"
6. Open the Code editor and under **"options"**, add:

   `"svg": "<copied file>"`

7. Repeat these steps for the other products: **Phone**, **Robot** and **Rocket**

Checkpoint! Save your dashboard periodically so you don't lose your changes.

Let's breakdown the dynamic fill

```
<svg xmlns="http://www.w3.org/2000/svg" width="326" height="415" viewBox="0 0 326 415" fill="none">
  <defs>
    <linearGradient x1="0" x2="0" y1="1" y2="0" id="fill_tablet" fill="#FFF">
      <stop offset="0%" stop-opacity="1" stop-color="#7B56DB"/>
      <stop offset="$Inventory totals:result.current_tablets$%" stop-opacity="1" stop-color="#7B56DB"/>
      <stop offset="$Inventory totals:result.current_tablets$%" stop-opacity="1" stop-color="#FFF"/>
      <stop offset="100%" stop-opacity="1" stop-color="#FFF"/>
    </linearGradient>
  </defs>
  <path id="tablet" stroke-width="8. fill="url(#fill_tablet)" stroke="#c3cbd4" d="M93.7386 64.891C93.5076 64.891 81.8406 67.9618 80.6954 68.
  2763C76.7167 69.6896 73.3049 73.5571 71.6882 78.6381L28.961 212.709C27.4598 217.79 27.6907 223.333 29.5384 228.414C31.2706 233.496 34.6195 237.
  884 39.0077 240.54L218.462 348.628C222.735 351.284 226.861 351.315 231.6 350.268C231.6 350.268 241.211 348.313 245.484 346.581C249.295 344.849
  252.528 341.269 254.145 336.188L296.872 203.156C300.336 192.647 295.717 179.714 286.825 174.286L107.371 66.1976C102.752 63.3107 97.9018 63.932
  93.7386 64.891Z"/>
</svg>
```

1. Under <defs> we're defining a linearGradient with the id="fill_tablet", which is referenced later in <path>
2. This is where we are defining the linear gradient that will fill the SVG based on inventory levels

Let's breakdown the dynamic fill

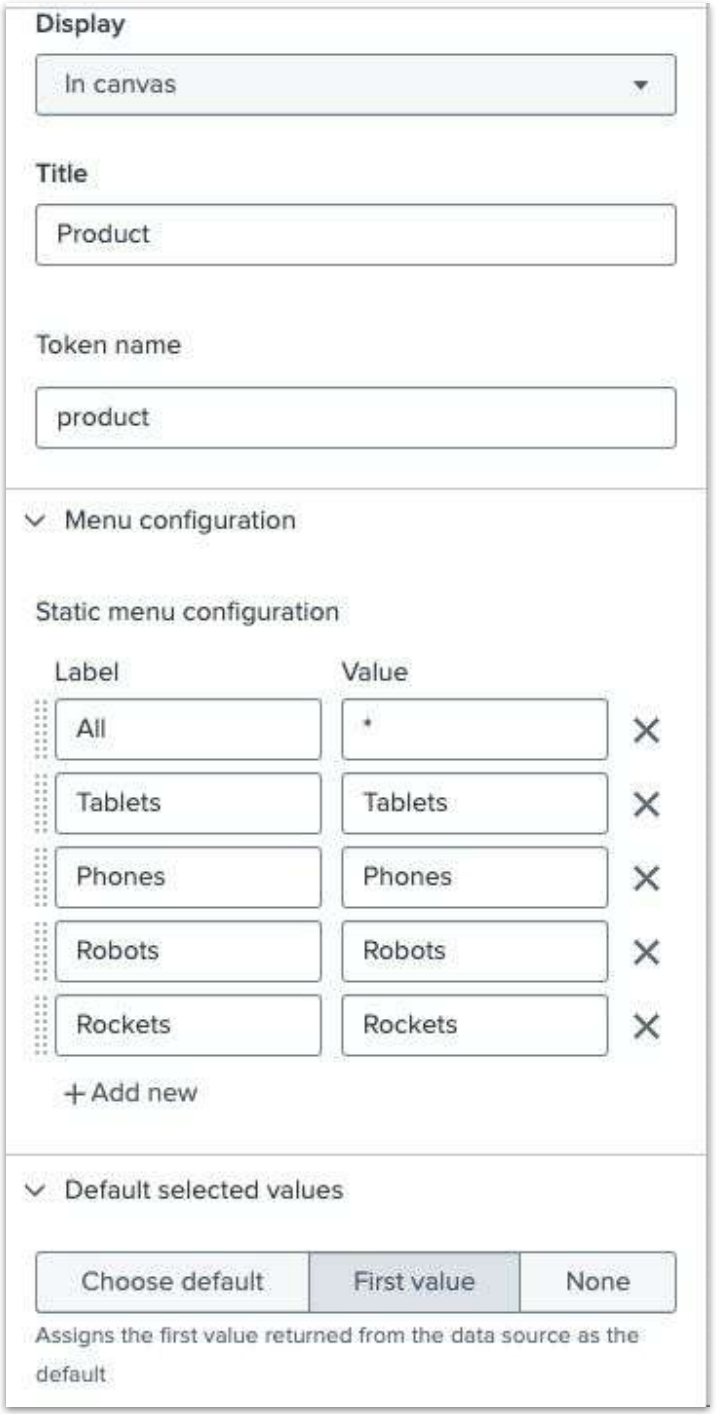
```
<svg xmlns="http://www.w3.org/2000/svg" width="326" height="415" viewBox="0 0 326 415" fill="none">
  <defs>
    <linearGradient x1="0" x2="0" y1="1" y2="0" id="fill_tablet" fill="#FFF">
      <stop offset="0%" stop-opacity="1" stop-color="#7B56DB"/>
      <stop offset="$Inventory totals:result.current_tablets$%" stop-opacity="1" stop-color="#7B56DB"/>
      <stop offset="$Inventory totals:result.current_tablets$%" stop-opacity="1" stop-color="#FFF"/>
      <stop offset="100%" stop-opacity="1" stop-color="#FFF"/>
    </linearGradient>
  </defs>
  <path id="tablet" stroke-width="8.2029" fill="url(#fill_tablet)" stroke="#c3cbd4" d="M93.7386 64.891C93.5076 64.891 81.8406 67.9618 80.6954 68.
  2763C76.7167 69.6896 73.3049 73.5571 71.6882 78.6381L28.961 212.709C27.4598 217.79 27.6907 223.333 29.5384 228.414C31.2706 233.496 34.6195 237.
  884 39.0077 240.54L218.462 348.628C222.735 351.284 226.861 351.315 231.6 350.268C231.6 350.268 241.211 348.313 245.484 346.581C249.295 344.849
  252.528 341.269 254.145 336.188L296.872 203.156C300.336 192.647 295.717 179.714 286.825 174.286L107.371 66.1976C102.752 63.3107 97.9018 63.932
  93.7386 64.891Z"/>
</svg>
```

3. Each <stop> element is where we define positions and colors
   a. For example starting at 0% up to inventory levels, we want purple (#7B56DB"). Above the inventory level, we want white (#FFF)
4. This is where we are embedding the search results from our inventory search to determine how full the SVG should be

Exercise 9: Configure additional interactions

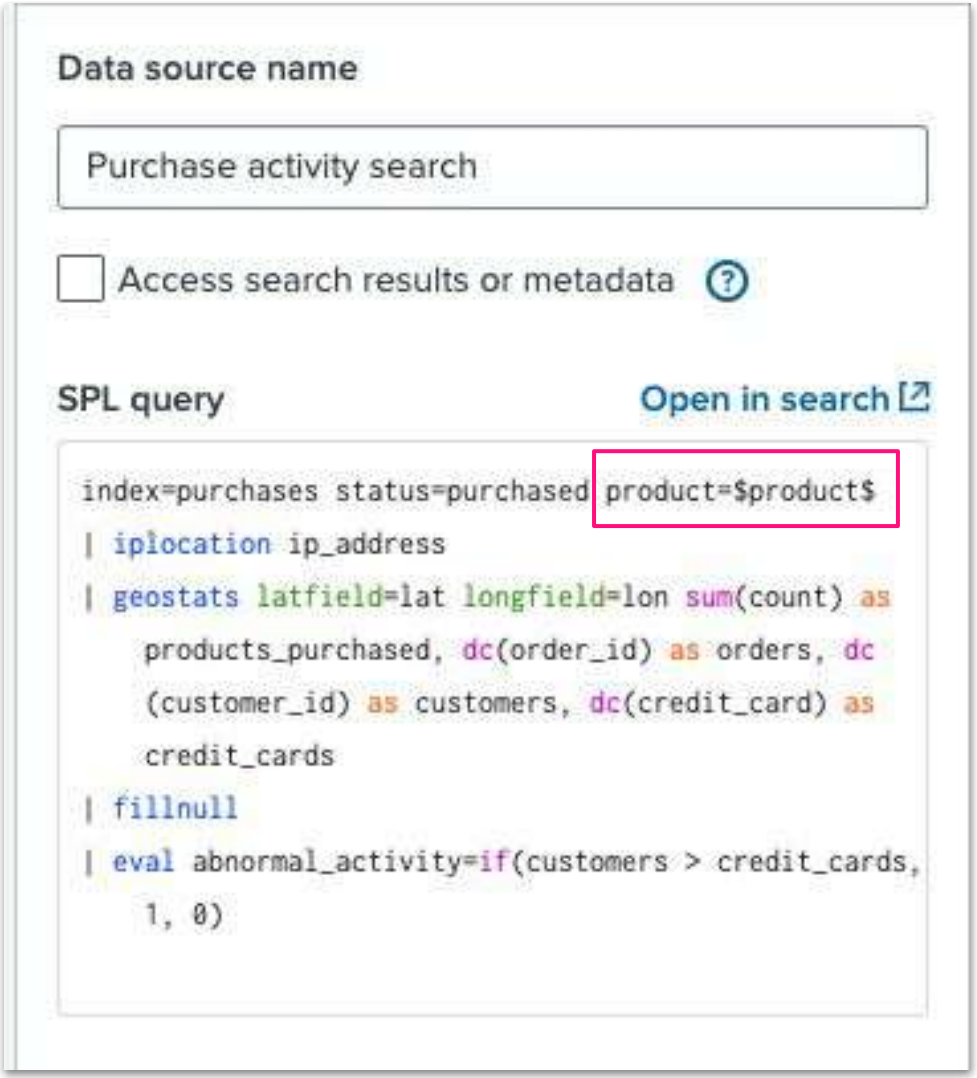
Filter the map by product

1. Add a dropdown input
2. Under **Display,** select **In canvas**
3. Reposition and resize the dropdown to be next to the **Purchase activity** custom map label
   a. **Title:** Product
   b. **Token name:** product
   c. Menu items - Label (value)
      i. All (*)
      ii. Tablets
      iii. Phones
      iv. Robots
      v. Rockets
   d. **Default selected values:** First value



Filter the map by product

4. Select the map
5. Edit the **Purchase activity search**
6. Add product=$product$ to the first pipe
7. Select **Apply & close**
8. Select different products from the dropdown to see the map update



Configure interactions to set the product filter

Multiple interactions can set a token value

1. Select the **Tablets** single value
2. Under **Interactions**, select **+ Add interaction**
3. Under **On click**, select **Set tokens**
   a. **Token name:** product
   b. **Token value:** name
4. Select **Apply**
5. Repeat for Phones, Robots, and Rockets
6. Save your dashboard and go to View mode

Select one of the product KPIs, and notice the Product input and map update accordingly

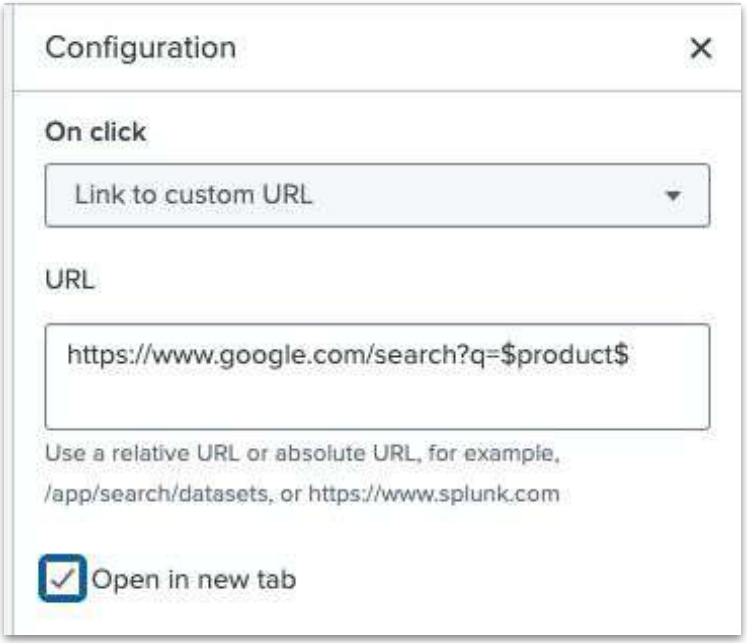
Note: There is a known bug where some charts without drilldowns in Classic are converting with "Link to Search" interactions. You can delete these interactions.

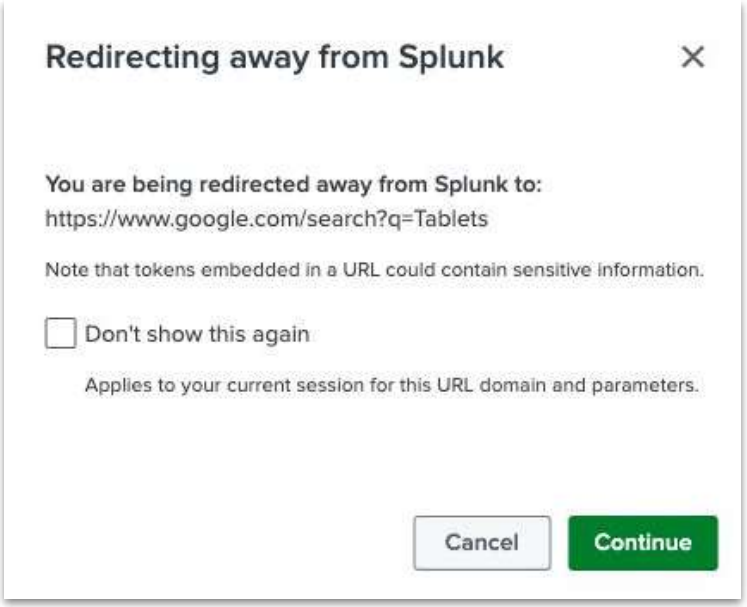
Set up additional interactions

You can configure multiple interactions on a given visualization. The order in which the interactions are listed are the order in which they will execute.

1. Back in Edit mode, select the **Tablets** KPIs
2. Under **Interactions**, select **+ Add interaction**
3. Under **On click**, select **Link to custom URL**
   a. https://www.google.com/search?q=$product$
   b. Check the box to **Open in a new tab**
4. Select **Apply**

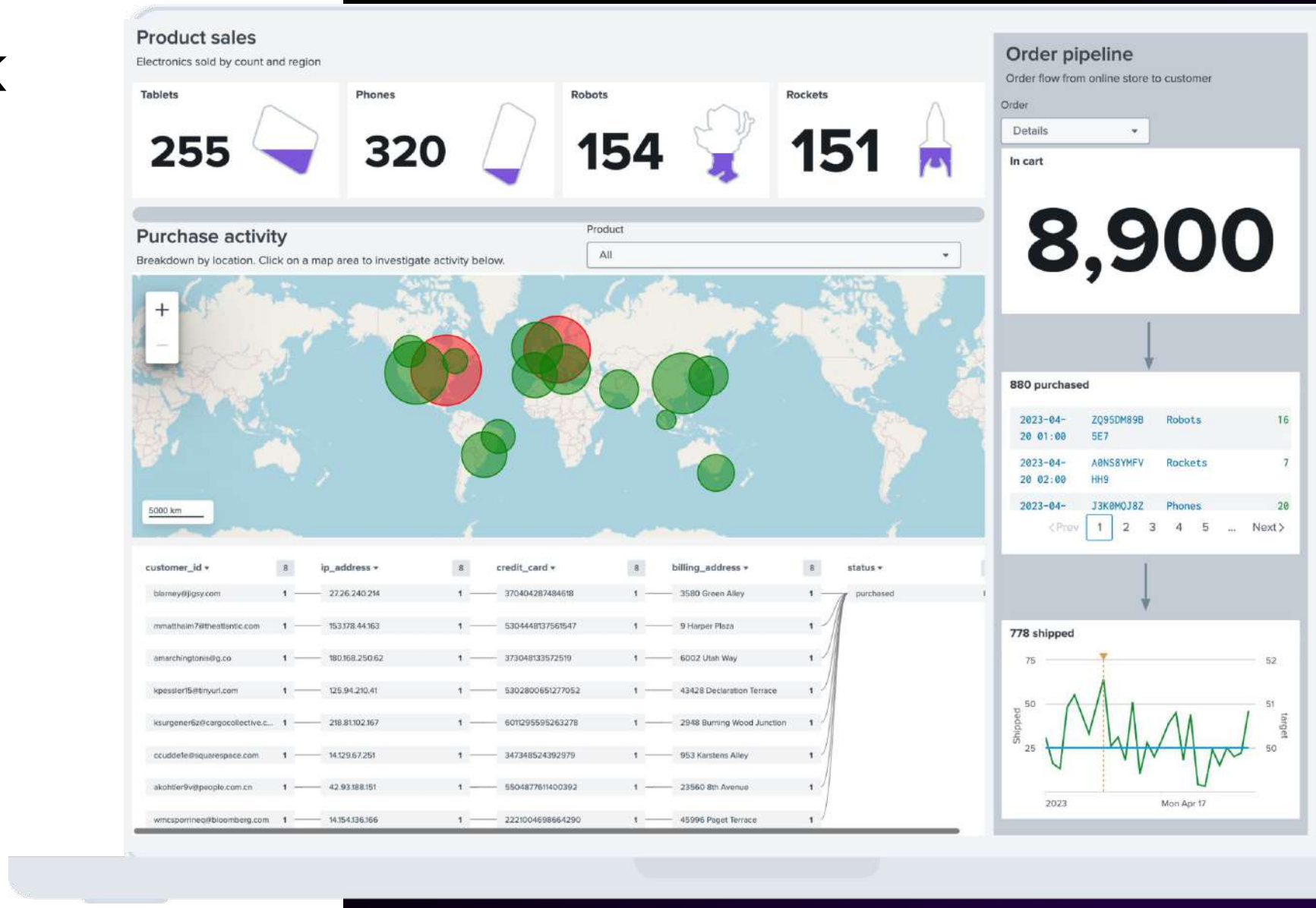
Save and go to View mode. Select the Tablet and see that the Product input is set to Tablet **and** you are prompted to navigate to Google.





Your dashboard should now look like this

🎉🎉🎉



Design best practices for your next Studio dashboard

Design tip 1:
Use viewer-centered dashboard design

1. **What decisions will your dashboard likely be informing?**
   - Will it be important for your viewers to compare one value to another, see a change over time, or drill down on a certain data point?
   - Design your dashboard so that viewers can easily make these key observations
   - For example, use a bar chart to compare values, and a line chart to see changes over time
2. **Leverage consistency and familiarity when possible**
   - Familiarity can help viewers orient themselves, and quickly understand the dashboard
   - For example, if your organization uses a certain color pallet to communicate certain warning levels, try to stay consistent with the existing pallet

Check out more design best practices here:
https://www.splunk.com/en_us/form/dashboard-design-best-practices.html

Design tip 2:
Communication best practices

1. **Consider information hierarchy**
   - Place high-level information at the top of the dashboard
   - Add charts that provide details about the high level information below
2. **When necessary, use text to instruct users on the functionality of your dashboard**
   - For example, using Markdown to instruct users to select a customer from the table in step 9
3. **Use short, descriptive titles for charts**
4. **Be intentional about chart type**
   - Dashboard Design: Visualization Choices and Configurations

Revisiting our learning objectives

Today you learned how to:

- ✓ Configure interactivity
- ✓ Configure conditional show/hide
- ✓ Apply options with Dynamic Options Syntax
- ✓ Add multiple data sources to a visualization
- ✓ Add custom dynamic SVGs
- ✓ Apply best practices
  - ✓ Base and chain searches
  - ✓ Reusing searches for multiple visualizations
  - ✓ Design best practices

Additional resources

- Visit the in-product **Examples Hub**
- Check out the Dashboard Studio **docs**
- Read our Dashboard Studio **blog posts** for new features and tips and tricks
- Join the **#dashboard_studio** Splunk user groups Slack channel
- Take an **EDU course** about dashboards
  - Intro to Dashboards
  - Dynamic Dashboards
- Watch the dashboards performance best practices tech talk
  - Improving Dashboard Performance and Resource Usage

Thank you