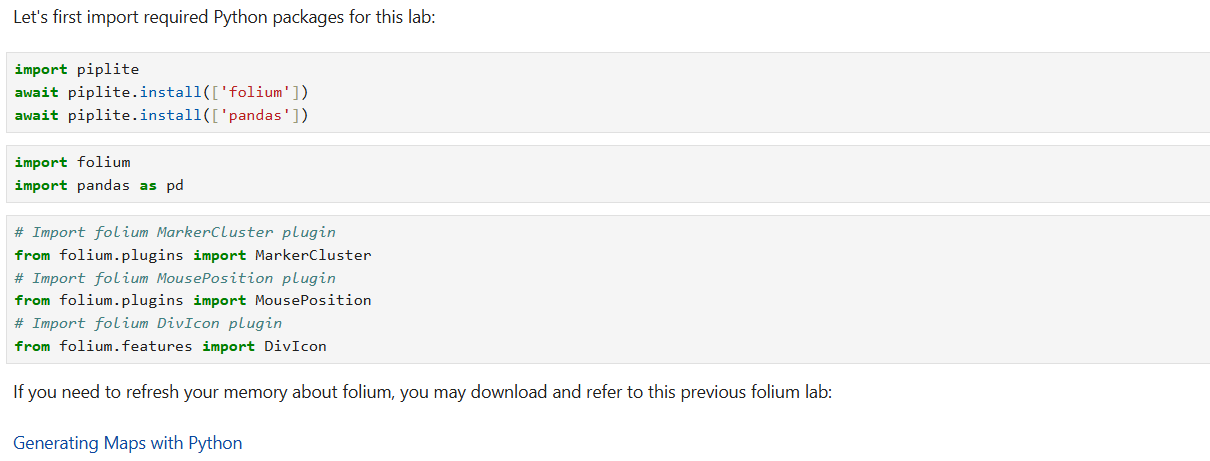
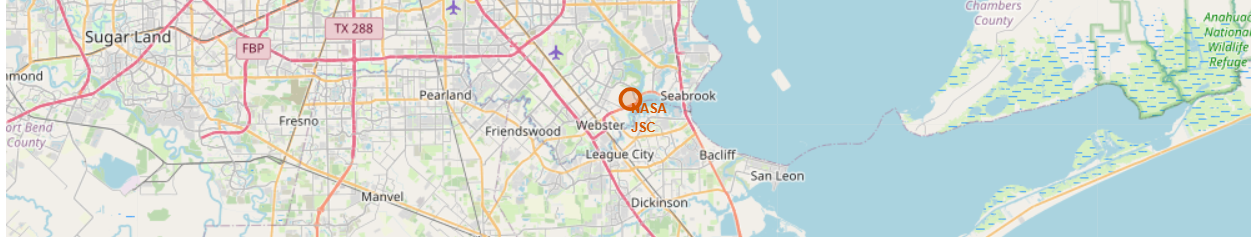
M3 LABS

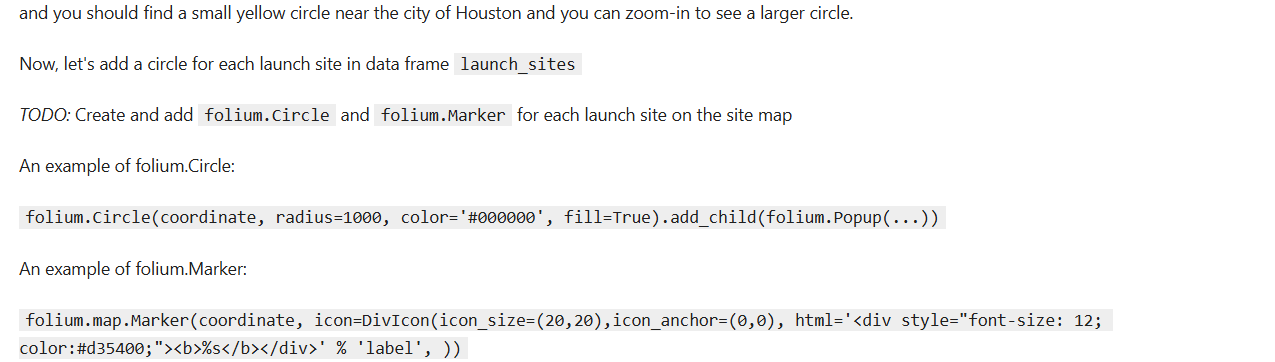
**Hands-on Lab: Interactive Visual Analytics with Folium lab**





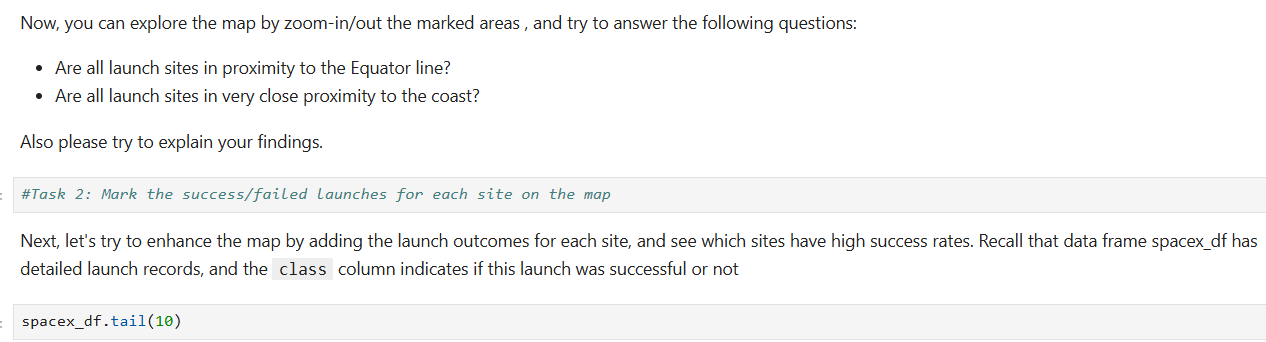


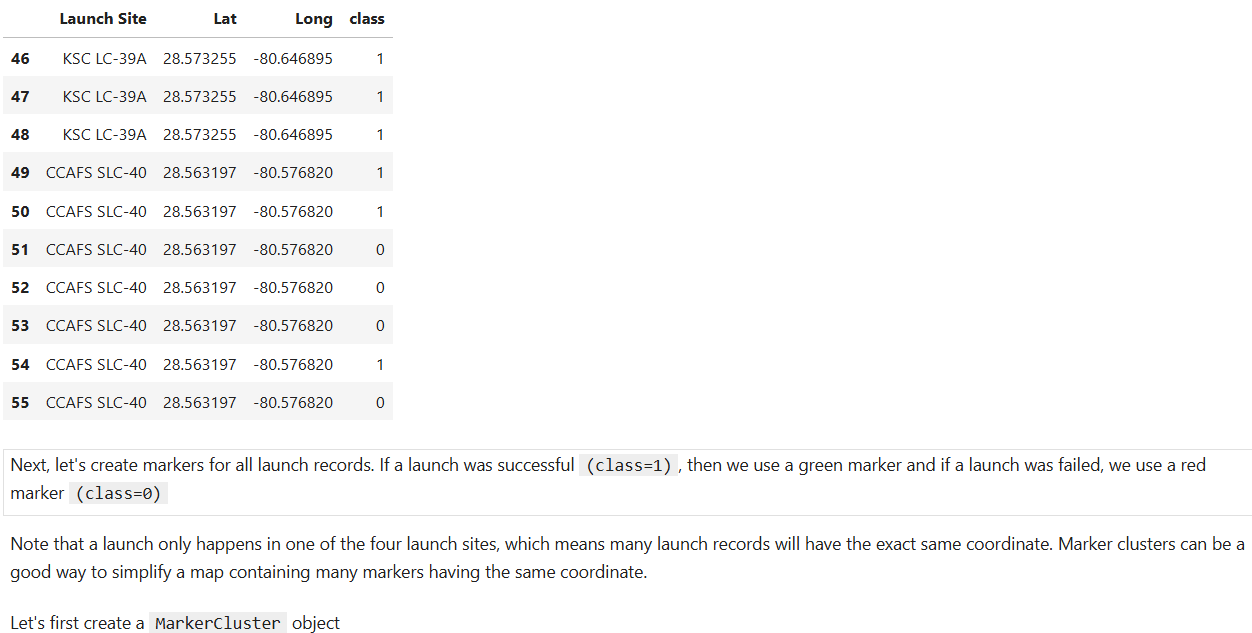




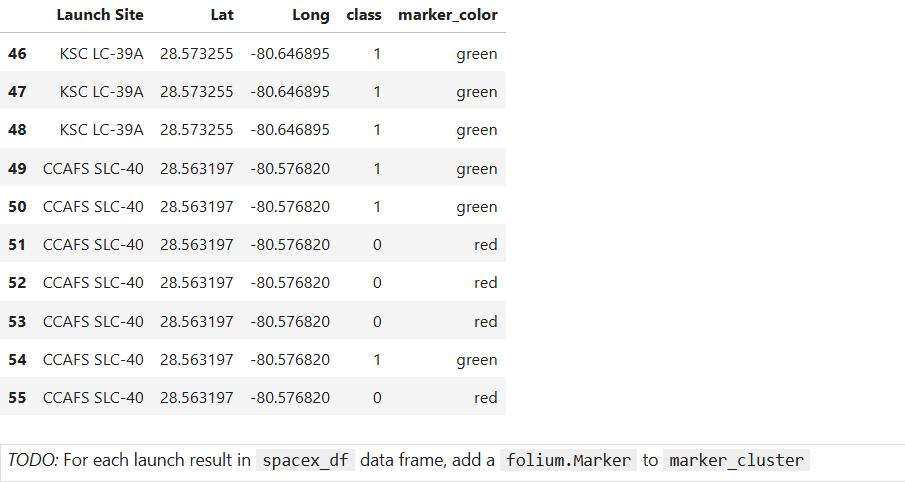


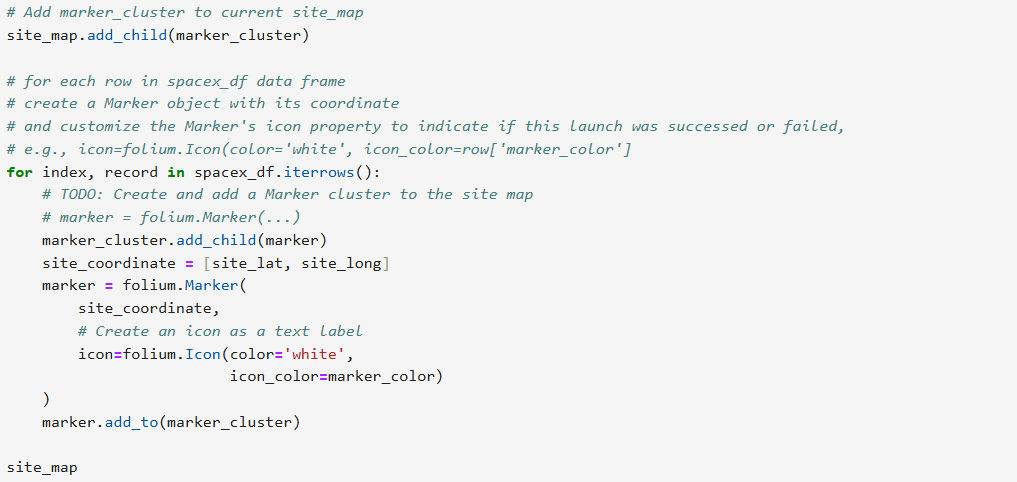




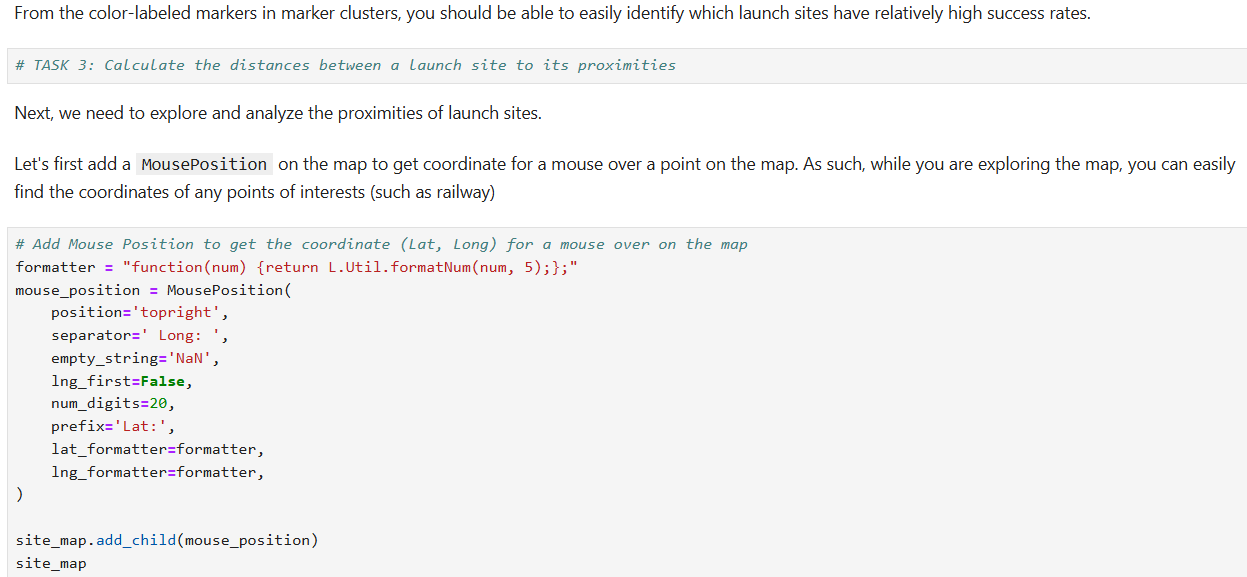


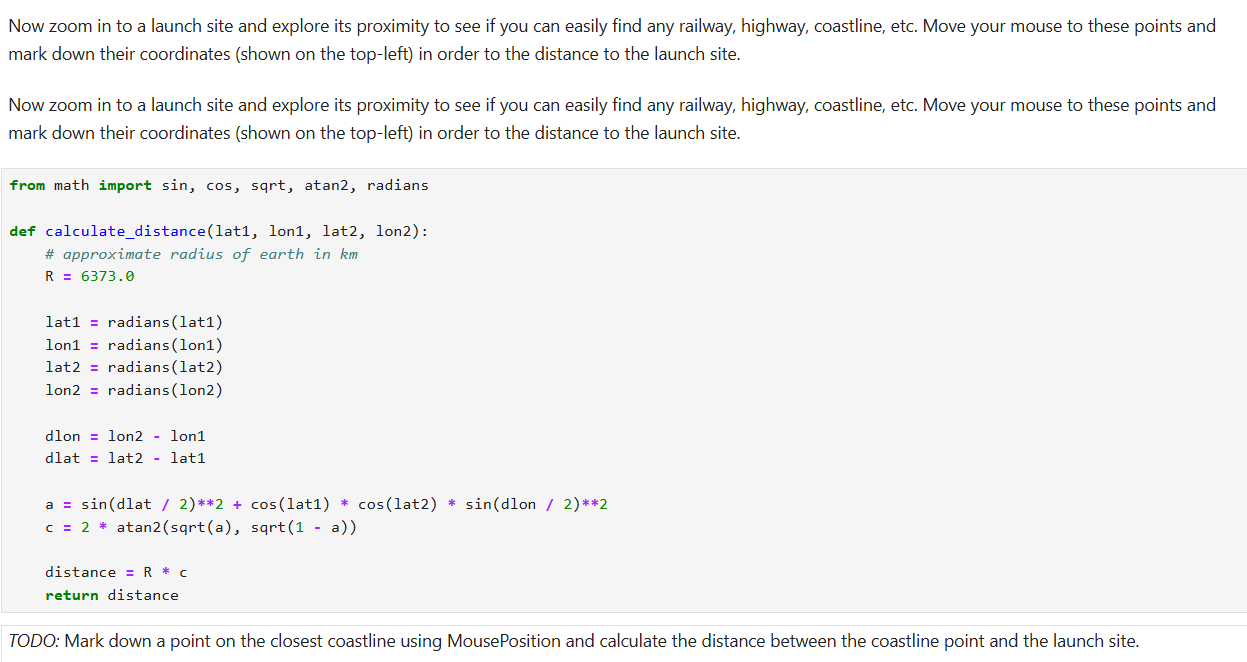




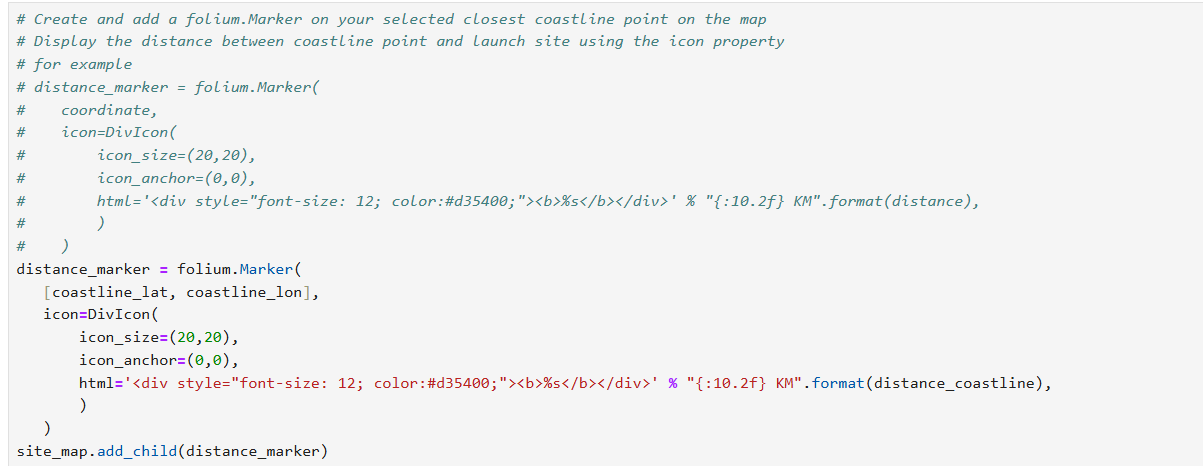


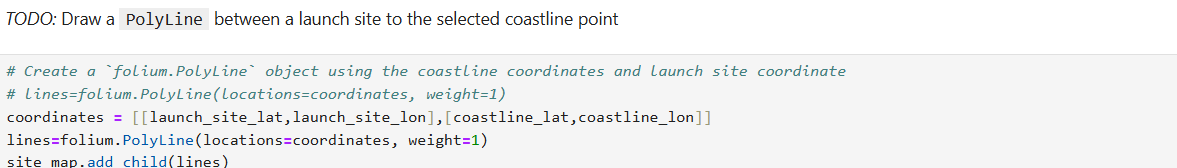




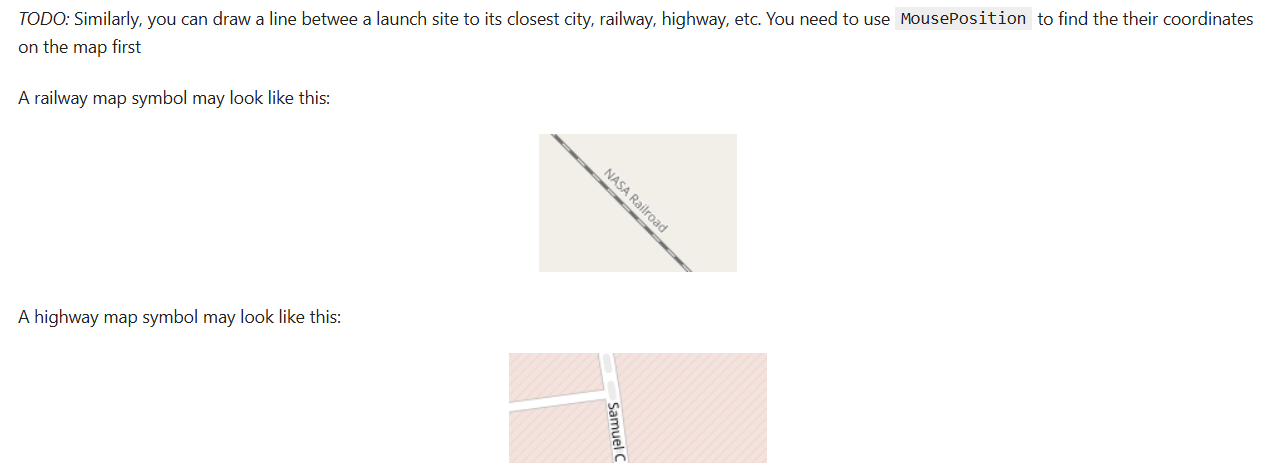


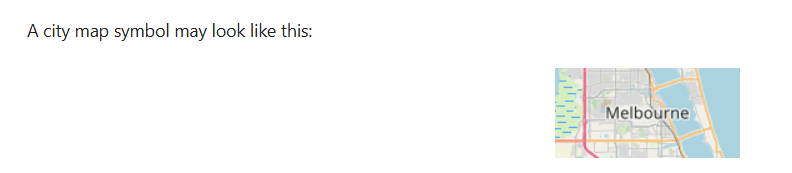








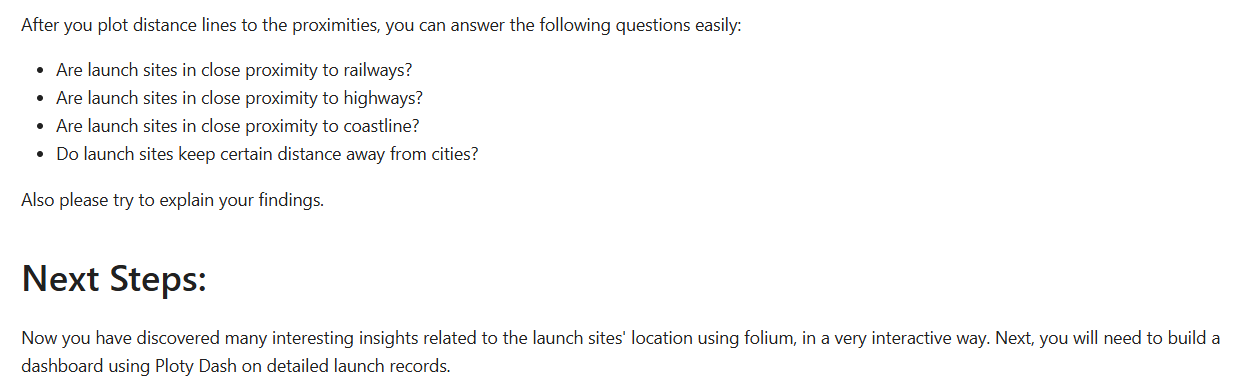






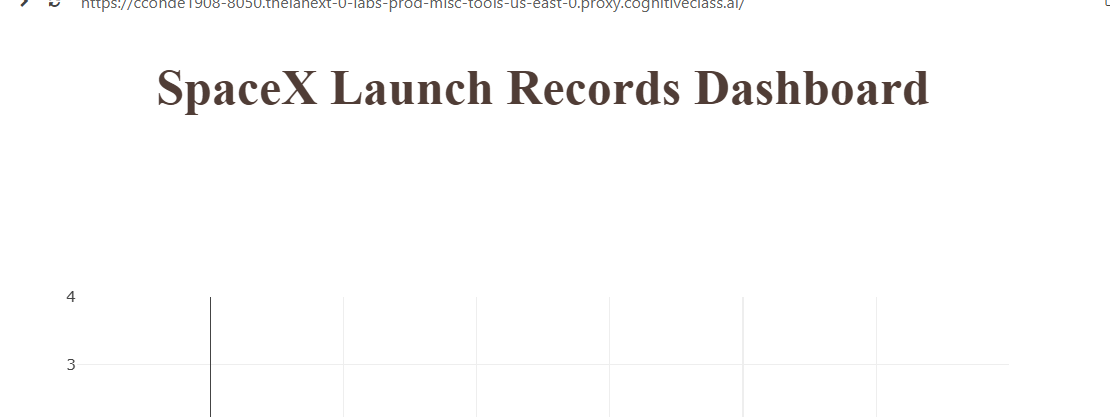






**Hands-on Lab: Build an Interactive Dashboard with Ploty Dash**

**8050**

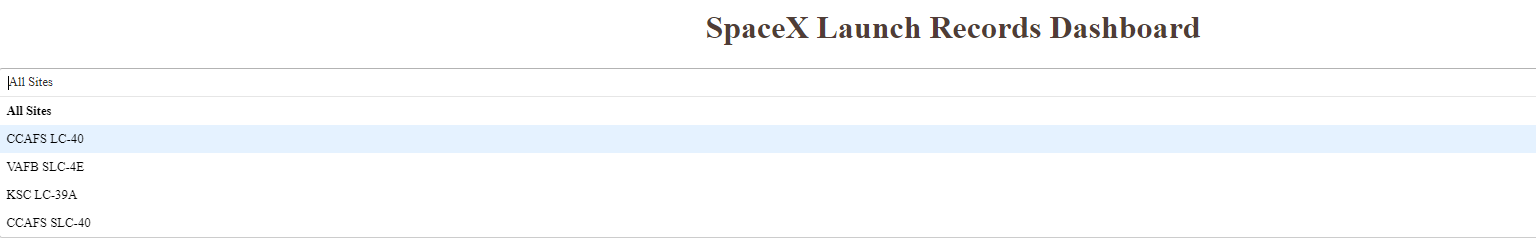
****

**TASK 1: Add a Launch Site Drop-down Input Component**

We have four different launch sites and we would like to first see which one has the largest success count. Then,  
we would like to select one specific site and check its detailed success rate (class=0 vs. class=1).

As such, we will need a dropdown menu to let us select different launch sites.

* Find and complete a commented dcc.Dropdown(id='site-dropdown',...) input with following attributes:
  + id attribute with value site-dropdown
  + options attribute is a list of dict-like option objects (with label and value attributes). You can set  
    the label and value all to be the launch site names in the spacex\_df  
    and you need to include the default All option. e.g.,



**TASK 2: Add a callback function to render success-pie-chart based on selected site dropdown**

The general idea of this callback function is to get the selected launch site from site-dropdown and render  
a pie chart visualizing launch success counts.

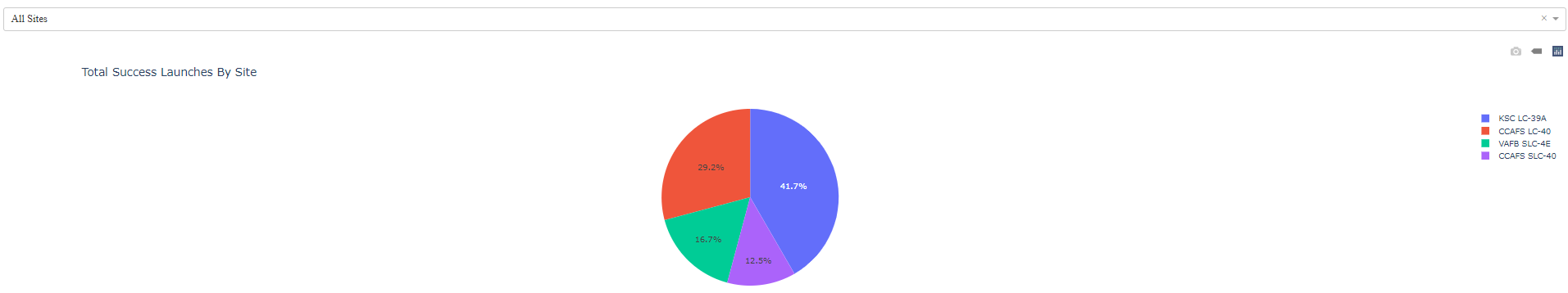
Dash callback function is a type of Python function which will be automatically called by  
Dash whenever receiving an input component updates, such as a click or dropdown selecting event.

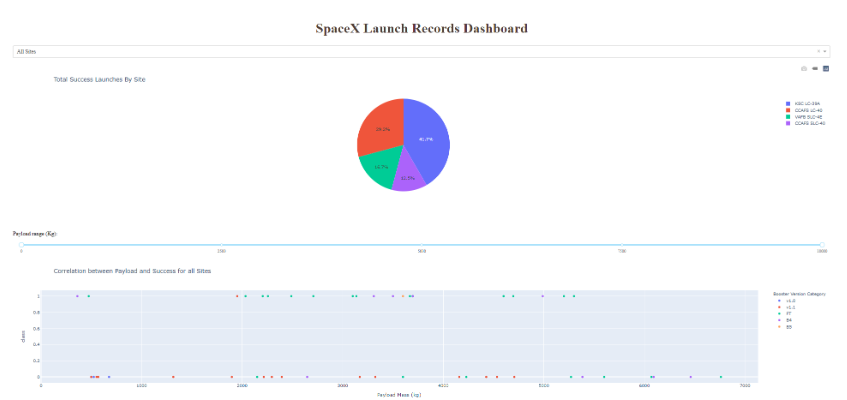
If you need to refresh your memory about Plotly Dash callback functions,  
you may refer to the lab you have learned before:

[Plotly Dash Lab](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/labs/Module%204/4.7_Dash_Interactivity.py)

Let’s add a callback function in spacex\_dash\_app.py including the following application logic:

* Input is set to be the site-dropdown dropdown, i.e., Input(component\_id='site-dropdown', component\_property='value')
* Output to be the graph with id success-pie-chart, i.e., Output(component\_id='success-pie-chart', component\_property='figure')
* A If-Else statement to check if ALL sites were selected or just a specific launch site was selected
  + If ALL sites are selected, we will use all rows in the dataframe spacex\_df to render and return a pie chart graph to show the total success launches (i.e., the total count of class column)
  + If a specific launch site is selected, you need to filter the dataframe spacex\_df first in order  
    to include the only data for the selected site.  
    Then, render and return a pie chart graph to show the success (class=1) count and failed (class=0) count for the selected site.





**TASK 3: Add a Range Slider to Select Payload**

Next, we want to find if variable payload is correlated to mission outcome. From a dashboard point of view, we  
want to be able to easily select different payload range and see if we can identify some visual patterns.

Find and complete a commented dcc.RangeSlider(id='payload-slider',...) input with the following attribute:

* id to be payload-slider
* min indicating the slider starting point, we set its value to be 0 (Kg)
* max indicating the slider ending point to, we set its value to be 10000 (Kg)
* step indicating the slider interval on the slider, we set its value to be 1000 (Kg)
* value indicating the current selected range, we could set it to be min\_payload and max\_payload



**TASK 4: Add a callback function to render the success-payload-scatter-chart scatter plot**

Next, we want to plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome (i.e., class column).  
As such, we can visually observe how payload may be correlated with mission outcomes for selected site(s).

In addition, we want to color-label the Booster version on each scatter point so that we may  
observe mission outcomes with different boosters.

Now, let’s add a call function including the following application logic:

* Input to be [Input(component\_id='site-dropdown', component\_property='value'), Input(component\_id="payload-slider", component\_property="value")]  
  Note that we have two input components, one to receive selected launch site and another to receive selected payload range
* Output to be Output(component\_id='success-payload-scatter-chart', component\_property='figure')
* A If-Else statement to check if ALL sites were selected or just a specific launch site was selected
  + If ALL sites are selected, render a scatter plot to display all values for variable Payload Mass (kg) and variable class.  
    In addition, the point color needs to be set to the booster version i.e., color="Booster Version Category"
  + If a specific launch site is selected, you need to filter the spacex\_df first, and render a scatter chart to show  
    values Payload Mass (kg) and class for the selected site, and color-label the point using Boosster Version Category likewise.

