

**IBM Developer** 

SKILLS NETWORK

This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. You will be guided to build this dashboard

SpaceX Launch Records Dashboard

Build a Dashboard Application with Plotly Dash

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

Note:Please take screenshots of the Dashboard and save them. Further upload your notebook to github.

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

The github url and the screenshots are later required in the presentation slides.

Your completed dashboard application should look like the following screenshot:

application via the following tasks:

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

TASK 3: Add a Range Slider to Select Payload

In this lab, you will be building a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.

Setup development environment Install required Python packages • Open a new terminal, by clicking on the menu bar and selecting **Terminal**->**New Terminal**, as in the image below. File Edit Selection **Terminal** View Run Help **New Terminal** ^ 仚` dash\_interactivity.py × Run Task... Run Build Task... Run Test Task... Run Last Task ① ℋ K Show Running Tasks... Restart Running Task... Terminate Task... Attach Task... Configure Tasks... **Run Selected Text** Now, you have script and terminal ready to start the lab.

Run Selection View **Terminal** Help File Edit Go dash\_interactivity.py × 1 theia@theiadocker-saishruthitn: /home/project × theia@theiadocker-saishruthitn:/home/project\$ □ Install python packages required to run the application. Copy and paste the below command to the terminal. pip3 **install** pandas dash theia@theiadocker-saishruthitn: /home/project × theia@theiadocker-saishruthitn:/home/project: pip install pandas dash Collecting pandas Downloading https://files.pythonhosted.org/packages/db/83/7d4008ffc2988066ff37f6a0bb6d7b60822367dcb36ba5e39aa7801fda54/pandas-0.24.2-cp27-cp27mu-manylinux1\_x86\_64.whl (10.1MB) 100% | | 10.1MB 116kB/s Collecting dash Downloading https://files.pythonhosted.org/packages/d4/50/e7c2830168db186f84b7de2988543e974433a6cdb0a0b23d51c781e2b2ab/dash-1. 20.0.tar.gz (77kB) | 81kB 11.0MB/s 100% | Collecting numpy>=1.12.0 (from pandas) Downloading https://files.pythonhosted.org/packages/3a/5f/47e578b3ae79e2624e205445ab77a1848acdaa2929a00eeef6b16eaaeb20/numpy-1 .16.6-cp27-cp27mu-manylinux1\_x86\_64.whl (17.0MB) 100% | ■| 17.0MB 68kB/s Collecting pytz>=2011k (from pandas)

Downloading https://files.pythonhosted.org/packages/70/94/784178ca5dd892a98f113cdd923372024dc04b8d40abe77ca76b5fb90ca6/pytz-20 21.1-py2.py3-none-any.whl (510kB) || 512kB 2.6MB/s 100% Python 3.6.9 64-bit 8 5 A 0 Ln 6, Col 35 LF UTF-8 Spaces: 4 Python 🐥 Download a skeleton dashboard application and dataset First, let's get the SpaceX Launch dataset for this lab: Run the following wget command line in the terminal to download dataset as spacex launch dash.csv wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex\_launch\_dash.csv" Ð Download a skeleton Dash app to be completed in this lab: wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module\_3/spacex\_dash\_app.py" Ð Test the skeleton app by running the following command in the terminal: python3 spacex\_dash\_app.py Ð Observe the port number (8050) shown in the terminal. Problems 🍮 Python × \* Serving Flask app "dash\_basics" (lazy loading) \* Environment: production Use a production WSGI server stead. ⋆ Debug mode: off \* Running or http://127.0.0.1:8050/ (Press CTRL+C to quit) Click on the Launch Application option from the menu bar. **IBMCloud** Selection View Go Run Terminal He File Edit dash\_basics.py × fig = px.pie(data, values='Month', names='DistanceGroup', title='Distance group proportion by month') 18 # Create a dash application 21 app = dash.Dash(\_\_name\_\_)

theia@theiadocker-saishruthitn:/home/project\$ python dash\_basics.py Dash is running on http://127.0.0.1:8050/ # Get the layout of the application and adjust it. # Create an outer division using html.Div and add title to the dashboard using html.H1 component # Add description about the graph using HTML P (paragraph) component Provide the port number 8050 and click ok labs.cognitiveclass.ai says 8050 Help Cancel names='DistanceGroup', title='Distance group proportion # Create a dash application app = dash.Dash(\_\_name\_\_) • You should see a nearly blank web page indicating a successfully running dash app. Next, let's fill the skeleton app with required input/output components and callback functions. If you need to refresh your memory about Plotly Dash components and callback functions, you may refer to the lab you have learned before: Plotly Dash Lab 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., options=[{'label': 'All Sites', 'value': 'ALL'},{'label': 'site1', 'value': 'site1'}, ...] value attribute with default dropdown value to be ALL meaning all sites are selected placeholder attribute to show a text description about this input area, such as Select a Launch Site here searchable attribute to be True so we can enter keywords to search launch sites Here is an example of dcc. Dropdown: dcc.Dropdown(id='id', options=[ {'label': 'All Sites', 'value': 'ALL'}, {'label': 'site1', 'value': 'site1'}, ], value='ALL',

placeholder="place holder here", searchable=True ), If you need more help about Dropdown(), refer to the Plotly Dash Reference section towards the end of this lab. Your completed dropdown menu should look like the following screenshot: SpaceX Launch Records Dashboard All Sites All Sites CCAFS LC-40 VAFB SLC-4E KSC LC-39A CCAFS SLC-40 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 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. Here is an example of a callback function: # Function decorator to specify function input and output @app.callback(Output(component\_id='success-pie-chart', component\_property='figure'), Input(component\_id='site-dropdown', component\_property='value')) def get\_pie\_chart(entered\_site): filtered\_df = spacex\_df if entered\_site == 'ALL': fig = px.pie(data, values='class', names='pie chart names', title='title') return fig else: # return the outcomes piechart for a selected site The rendered pie chart should look like the following screenshots: Pie chart for all sites are selected All Sites Total Success Launches By Site CCAFS LC-40 VAFB SLC-4E

· Pie chart for is selected CCAFS LC-40 Total Success Launches for site CCAFS LC-40 If you need more reference about dash callbacks and plotly pie charts, refer to the Plotly Dash Reference section towards the end of this lab. TASK 3: Add a Range Slider to Select Payload 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 Here is an example of RangeSlider: dcc.RangeSlider(id='id', min=0, max=10000, step=1000, marks={0: '0', 100: '100'}, value=[min\_value, max\_value]) You completed payload range slider should be similar the following screenshot: Payload range (Kg):

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 If you need more reference about range slider, refer to the Plotly Dash Reference towards the end of this lab. TASK 4: Add a callback function to render the success-payload-scatterchart 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.

You rendered scatter point should look like the following screenshot: Payload range (Kg): Correlation between Payload and Success for all Sites Payload Mass (kg) If you need more reference about dash callbacks and plotly scatter plots, refer to the Plotly Dash Reference towards the end of this lab. Finding Insights Visually Now with the dashboard completed, you should be able to use it to analyze SpaceX launch data, and answer the following questions: 1. Which site has the largest successful launches? 2. Which site has the highest launch success rate? 3. Which payload range(s) has the highest launch success rate? 4. Which payload range(s) has the lowest launch success rate? 5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?

Plotly Dash Reference Dropdown (input) component Refer here for more details about dcc.Dropdown() Range slider (input) component Refer here for more details about dcc.RangeSlider() Pie chart (output) component Refer here for more details about plotly pie charts Scatter chart (output) component Refer here for more details about plotly scatter charts **Author** Yan Luo Other contributor(s) Joseph Santarcangelo Changelog **Date** Changed by **Change Description** Version 03-09-2021 1.1 Lakshmi Holla Added a note.

Initial version created 06-01-2021 1.0 Yan © IBM Corporation 2021. All rights reserved.