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

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.

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 application via the following tasks:

* TASK 1: Add a Launch Site Drop-down Input Component
* TASK 2: Add a callback function to render success-pie-chart based on selected site dropdown
* TASK 3: Add a Range Slider to Select Payload
* TASK 4: Add a callback function to render the success-payload-scatter-chart scatter plot

After visual analysis using the dashboard, you should be able to obtain some insights to answer the following five 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?

# Import required libraries

import pandas as pd

import dash

import dash\_html\_components as html

import dash\_core\_components as dcc

from dash.dependencies import Input, Output

import plotly.express as px

# Read the airline data into pandas dataframe

spacex\_df = pd.read\_csv("spacex\_launch\_dash.csv")

max\_payload = spacex\_df['Payload Mass (kg)'].max()

min\_payload = spacex\_df['Payload Mass (kg)'].min()

# Create a dash application

app = dash.Dash(\_\_name\_\_)

# Create an app layout

app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',

                                        style={'textAlign': 'center', 'color': '#503D36',

                                               'font-size': 40}),

                                # TASK 1: Add a dropdown list to enable Launch Site selection

                                # The default select value is for ALL sites

                                dcc.Dropdown(id='site-dropdown',

                                              options=[{'label': 'All Sites', 'value': 'ALL'},

                                                       {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},

                                                       {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'},

                                                       {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},

                                                       {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'}],

                                              value='ALL',

                                              placeholder="Select a Launch Site here",

                                              searchable=True),

                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites

                                # If a specific launch site was selected, show the Success vs. Failed counts for the site

                                html.Div(dcc.Graph(id='success-pie-chart')),

                                html.Br(),

                                html.P("Payload range (Kg):"),

                                # TASK 3: Add a slider to select payload range

                                dcc.RangeSlider(id='payload-slider',

                                                min=0, max=10000, step=1000,

                                                marks={0: '0',

                                                       1000: '1000',

                                                       2000: '2000',

                                                       3000: '3000',

                                                       4000: '4000',

                                                       5000: '5000',

                                                       6000: '6000',

                                                       7000: '7000',

                                                       8000: '8000',

                                                       9000: '9000',

                                                       10000: '10000'},

                                                value=[min\_payload, max\_payload]),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success

                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),

                                ], style = {'height': '100vh'})

# TASK 2:

# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output

# 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(filtered\_df, values = 'class',

                     names = 'Launch Site',

                     title = 'SpaceX Launch Site Success Distribution (All Sites)')

        return fig

    else:

        # return the outcomes piechart for a selected site

        filtered\_df = spacex\_df[spacex\_df['Launch Site'] == entered\_site]

        filtered\_df = filtered\_df.groupby(['Launch Site', 'class']).size().reset\_index(name = 'class\_count')

        fig = px.pie(filtered\_df, values = 'class\_count', names = filtered\_df['class'].map({1: "Success", 0: "Failure"}),

                     title = f"SpaceX Success Rate of {entered\_site} Launch Site")

        fig.update\_traces(marker = dict(colors=['red', 'green']))

        return fig

# TASK 4:

# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output

@app.callback(

    Output(component\_id='success-payload-scatter-chart', component\_property='figure'),

    [

        Input(component\_id='site-dropdown', component\_property='value'),

        Input(component\_id='payload-slider', component\_property='value')

    ]

)

def update\_scatter\_chart(selected\_site, payload\_range):

    # Filter the dataframe based on the payload range

    filtered\_df = spacex\_df[

        (spacex\_df['Payload Mass (kg)'] >= payload\_range[0]) &

        (spacex\_df['Payload Mass (kg)'] <= payload\_range[1])

    ]

    if selected\_site == 'ALL':

        # Render scatter plot for all sites

        fig = px.scatter(

            filtered\_df,

            x='Payload Mass (kg)',

            y='class',

            color='Booster Version Category',

            title='Payload vs. Outcome for All Sites',

            labels={'class': 'Launch Outcome', 'Payload Mass (kg)': 'Payload Mass (kg)'}

        )

    else:

        # Filter for the specific site

        site\_filtered\_df = filtered\_df[filtered\_df['Launch Site'] == selected\_site]

        fig = px.scatter(

            site\_filtered\_df,

            x='Payload Mass (kg)',

            y='class',

            color='Booster Version Category',

            title=f'Payload vs. Outcome for {selected\_site}',

            labels={'class': 'Launch Outcome', 'Payload Mass (kg)': 'Payload Mass (kg)'}

        )

    return fig

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

    app.run\_server(debug=True, port=8080)