Build a Dashboard Application with Plotly 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 Payloa#d TASK 4: Add a callback function to render the success-payload-scatter-chart scatter pl#ot Note:Please take screenshots of the Dashboard and save them. Further upload your notebook to githu#T.

The github url and the screenshots are later required in the presentation sl

In [ ]:
```
!pip install jupyter-dash dash dash-bootstrap-components pandas

!pip install jupyter-dash
```

In [3]:
```
# Import necessary libraries
from jupyter_dash import JupyterDash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import pandas as pd
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_3660\1786318461.py:3: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  import dash_core_components as dcc
C:\Users\dell\AppData\Local\Temp\ipykernel_3660\1786318461.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
```

In [4]:
```
# Load your SpaceX data
spacex_df = pd.read_csv(r'D:\06_CONTINUOUS LEARNING\01-DATA SCIENTIST\IBM Data Scie
```

In [6]:
```
spacex_df.head()
```

Out[6]:

	Unnamed: 0	Flight Number	Date	Time (UTC)	Booster Version	Launch Site	Payload	Payload Mass (kg)	Orbit	Cus
0	0	1	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	
1	1	2	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel o...	0.0	LEO (ISS)	
2	2	3	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2+	525.0	LEO (ISS)	
3	3	4	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	
4	4	5	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	

In [ ]:
```
from dash import Dash, dcc, html  # Updated imports
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd

# Assuming  data is already loaded
# spacex_df = pd.read_csv('path_to_spacex_data.csv')

app1 = Dash(__name__, use_pages=False)  # Updated to use standard Dash with pages d
app1.layout = html.Div([
    html.H1("SpaceX Launch Records Dashboard"),
    dcc.Dropdown(
        id='site-dropdown',
        options=[{'label': 'All Sites', 'value': 'ALL'}] +
                [{'label': site, 'value': site} for site in spacex_df['Launch Site'
        value='ALL',
```

```
        searchable=True,
        style={'width': '50%', 'padding': '3px', 'fontSize': '20px'}
    ),
    dcc.Graph(id='success-pie-chart')
])

@app1.callback(
    Output('success-pie-chart', 'figure'),
    Input('site-dropdown', 'value')
)
def update_pie_chart(entered_site):
    filtered_df = spacex_df if entered_site == 'ALL' else spacex_df[spacex_df['Laun
    pie_data = filtered_df['class'].value_counts().reset_index()
    pie_data.columns = ['class', 'count']
    fig = px.pie(pie_data, values='count', names='class', title='Launch Success and
    return fig

# Run this app within the notebook
app1.run_server(mode='inline')
```

In [15]:

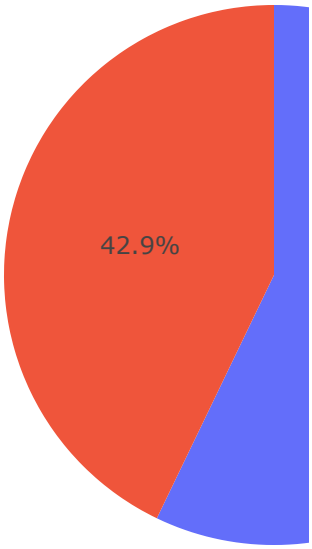
```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px
import pandas as pd

# Assuming  data is loaded correctly
# spacex_df = pd.read_csv('path_to_your_data.csv')

# Setup for Dash app (note the import change if needed)
app = Dash(__name__, use_pages=False)

app.layout = html.Div([
    html.H1("SpaceX Launch Records Dashboard"),

    dcc.Dropdown(
        id='site-dropdown',
        options=[{'label': 'All Sites', 'value': 'ALL'}] +
                [{'label': site, 'value': site} for site in spacex_df['Launch Site'
        value='ALL',
        searchable=True,
        style={'width': '50%', 'padding': '3px', 'fontSize': '20px'}
    ),

    dcc.RangeSlider(
        id='payload-slider',
        min=0,
        max=int(spacex_df['Payload Mass (kg)'].max()),
        step=1000,
        marks={i: f'{i} Kg' for i in range(0, int(spacex_df['Payload Mass (kg)'].ma
        value=[0, int(spacex_df['Payload Mass (kg)'].max())]
    ),

    dcc.Graph(id='success-payload-scatter-chart')
])

@app.callback(
    Output('success-payload-scatter-chart', 'figure'),
```

```
    [Input('site-dropdown', 'value'), Input('payload-slider', 'value')]
)
def update_scatter_plot(selected_site, payload_range):
    min_payload, max_payload = payload_range
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= min_payload) &
                            (spacex_df['Payload Mass (kg)'] <= max_payload)]
    if selected_site != 'ALL':
        filtered_df = filtered_df[filtered_df['Launch Site'] == selected_site]

    fig = px.scatter(
        filtered_df,
        x='Payload Mass (kg)',
        y='class',
        color='Booster Version',
        title=f"Payload vs. Outcome for {selected_site if selected_site != 'ALL' el
    )
    return fig

# To run the app inline in a Jupyter Notebook
app.run_server(mode='inline')
```

SpaceX Launch Records Dashboard

All Sites

Launch Success and Failures



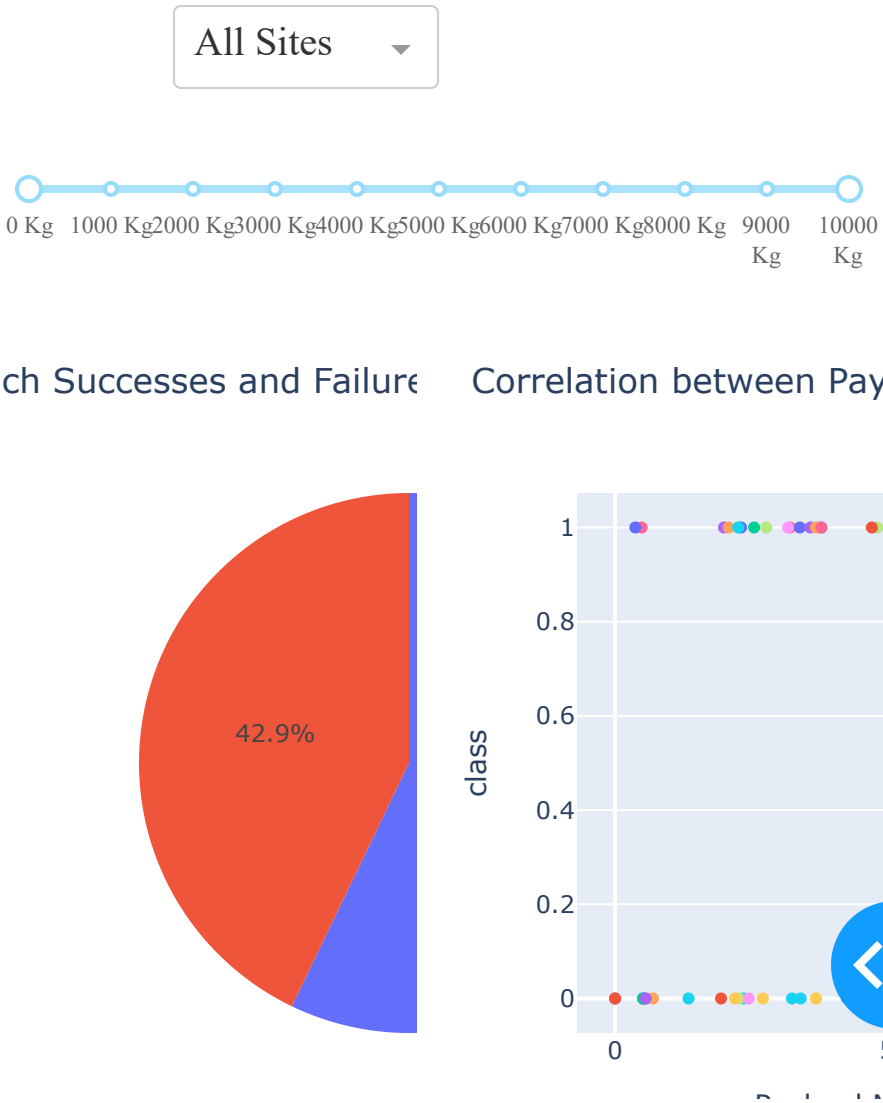
In [ ]:
```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px
import pandas as pd


# Initialize the Dash app
app = Dash(__name__)

# Define the layout of the app
app.layout = html.Div([
    html.H1("SpaceX Launch Records Dashboard"),

    # Dropdown to select the launch site
    html.Div([
        dcc.Dropdown(
            id='site-dropdown',
```

```
            options=[{'label': 'All Sites', 'value': 'ALL'}] +
                    [{'label': site, 'value': site} for site in spacex_df['Launch S
            value='ALL',
            placeholder="Select a Launch Site here",
            searchable=True,
            style={'width': '50%', 'padding': '3px', 'fontSize': '20px'}
        )
    ], style={'display': 'flex', 'justifyContent': 'center', 'padding': '20px'}),

    # RangeSlider to select the payload range
    html.Div([
        dcc.RangeSlider(
            id='payload-slider',
            min=0,
            max=10000,  # Assume 10000 is the max payload mass, or use spacex_df['P
            step=1000,
            marks={i: f'{i} Kg' for i in range(0, 10001, 1000)},
            value=[0, 10000]
        )
    ], style={'width': '80%', 'padding': '20px', 'margin': 'auto'}),

    # Pie chart for launch success counts
    html.Div(dcc.Graph(id='success-pie-chart'), style={'width': '48%', 'display': '

    # Scatter plot for payload vs. outcome
    html.Div(dcc.Graph(id='success-payload-scatter-chart'), style={'width': '48%',
])

# Callback to update the pie chart based on the selected site
@app.callback(
    Output('success-pie-chart', 'figure'),
    Input('site-dropdown', 'value')
)
def update_pie_chart(entered_site):
    filtered_df = spacex_df if entered_site == 'ALL' else spacex_df[spacex_df['Laun

    pie_data = filtered_df['class'].value_counts().reset_index()
    pie_data.columns = ['class', 'count']
    fig = px.pie(
        pie_data,
        values='count',
        names='class',
        title=f'Launch Successes and Failures for {entered_site if entered_site !=
        color_discrete_map={0: 'red', 1: 'green'}
    )
    return fig

# Callback to update the scatter plot based on the selected site and payload range
@app.callback(
    Output('success-payload-scatter-chart', 'figure'),
    [Input('site-dropdown', 'value'), Input('payload-slider', 'value')]
)
def update_scatter_plot(selected_site, payload_range):
    min_payload, max_payload = payload_range
    if selected_site == 'ALL':
        filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= min_payload) &
```

```
                                (spacex_df['Payload Mass (kg)'] <= max_payload)]
        title = 'Correlation between Payload and Success for All Sites'
    else:
        filtered_df = spacex_df[(spacex_df['Launch Site'] == selected_site) &
                                (spacex_df['Payload Mass (kg)'] >= min_payload) &
                                (spacex_df['Payload Mass (kg)'] <= max_payload)]
        title = f'Correlation between Payload and Success for {selected_site}'

    fig = px.scatter(
        filtered_df,
        x='Payload Mass (kg)',
        y='class',
        color='Booster Version',
        title=title
    )
    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, use_reloader=False)  # use_reloader=False only neede
```

In [19]:
```
if __name__ == '__main__':
    app.run_server(debug=True, use_reloader=False, port=8051)  # Use a different po
```

SpaceX Launch Records Dashboard



In [ ]:
```
Which site has the largest successful launches?
CCAFS SLC-40 appears to have the largest portion of successful launches.

Which site has the highest launch success rate?
KSC LC-39A seems to have the highest success rate based on the pie chart's blue-to-

Which payload range(s) has the highest launch success rate?
that middle payload ranges have a higher density of successful launches (blue point

Which payload range(s) has the lowest launch success rate?
4000 range 6000  range has the lowest launch success rate

Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch succ
F9 v1.1 has the highest launch success rate
```