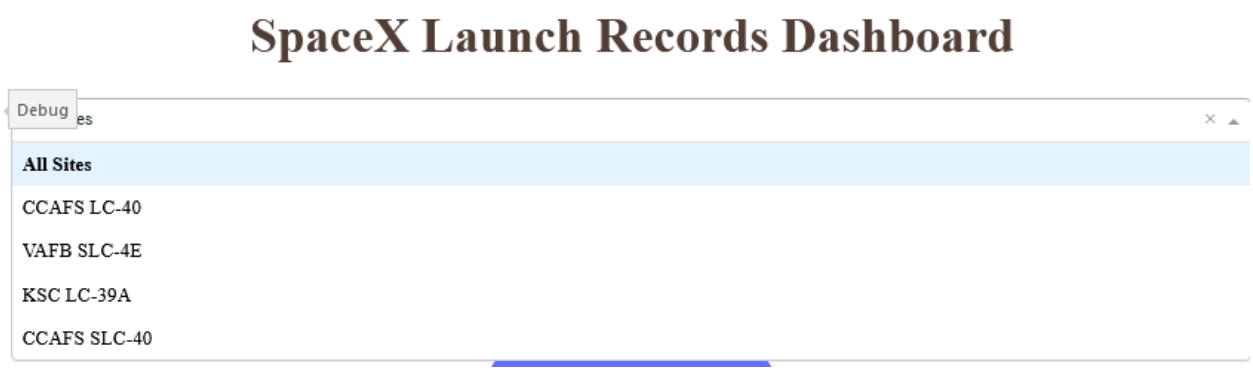
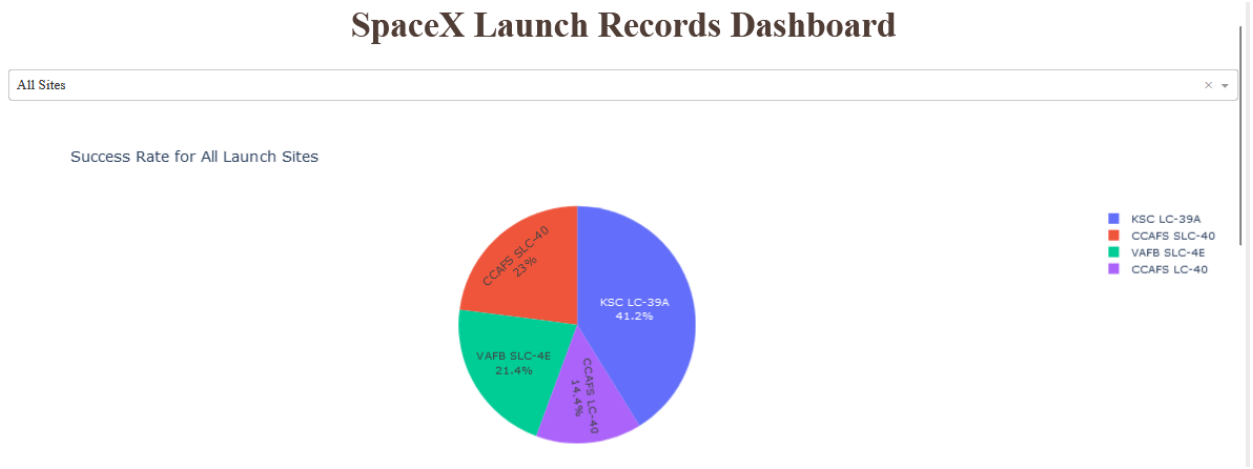
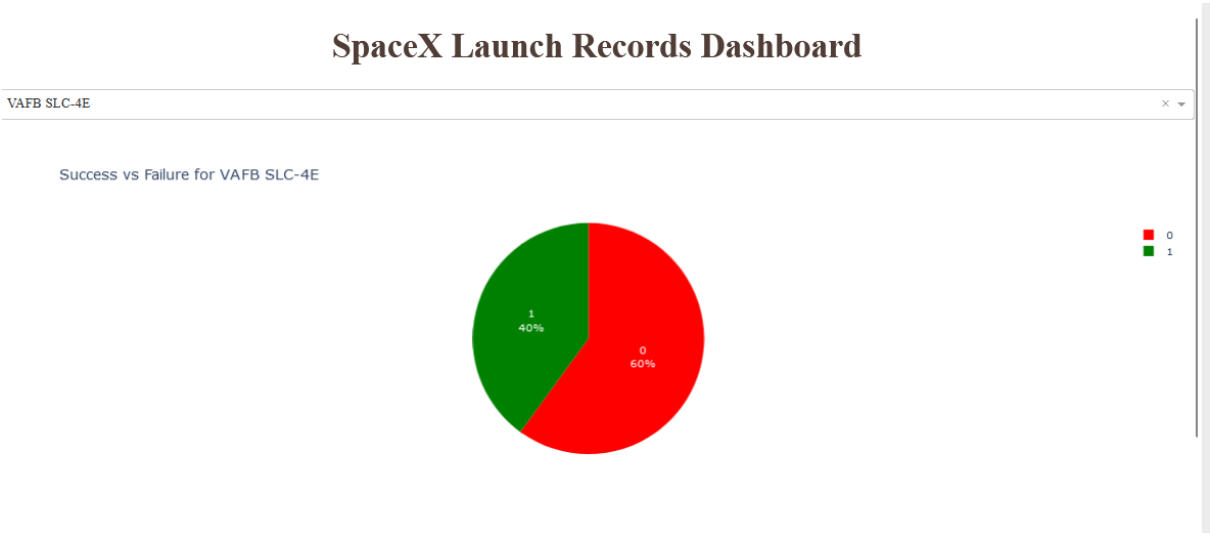
Task 1:


SpaceX Launch Records Dashboard
Debug
All Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Task 2:

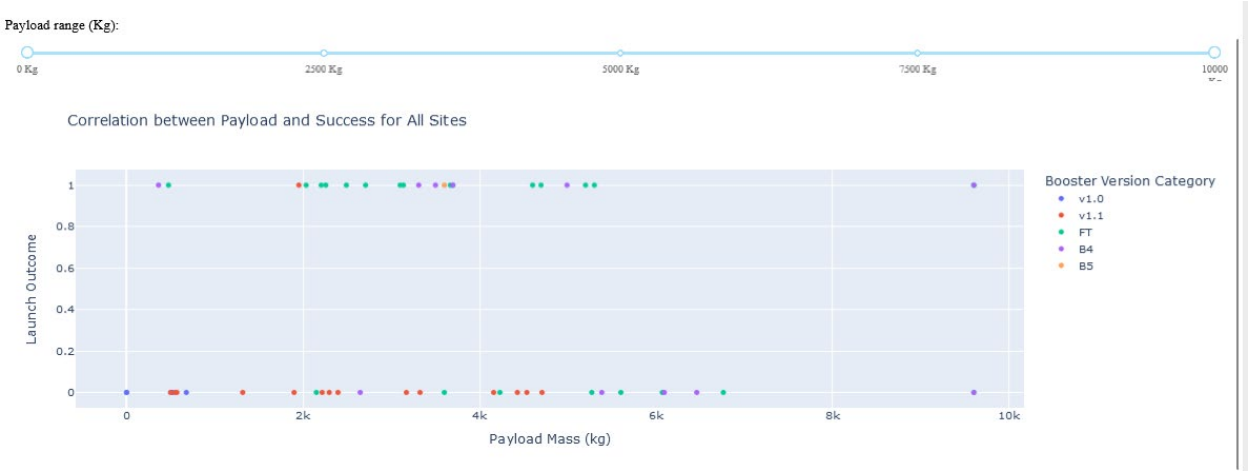

SpaceX Launch Records Dashboard
All Sites
Success Rate for All Launch Sites
KSC LC-39A
41.2%
CCAFS SLC-40
23%
VAFB SLC-4E
21.4%
CCAFS LC-40
14.4%
KSC LC-39A
CCAFS SLC-40
VAFB SLC-4E
CCAFS LC-40


SpaceX Launch Records Dashboard
VAFB SLC-4E
Success vs Failure for VAFB SLC-4E
1
40%
0
60%
0
1

Task 3:


Payload range (Kg):
0 Kg
2500 Kg
5000 Kg
7500 Kg
10000

Task 4:


Payload range (Kg):
0 Kg
2500 Kg
5000 Kg
7500 Kg
10000
Correlation between Payload and Success for All Sites
Launch Outcome
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT
B4
B5


Payload range (Kg):
0 Kg
2500 Kg
5000 Kg
7500 Kg
10000
Correlation between Payload and Success for All Sites
Launch Outcome
Payload Mass (kg)
Booster Version Category
v1.1
FT
B4
B5

Code:

```
# 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__)

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
  dcc.Dropdown(
    id='site-dropdown',
    options=[{'label': 'All Sites', 'value': 'ALL'}] +
          [{'label': site, 'value': site} for site in spacex_df['Launch Site'].unique()],
    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
    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,
        value=[min_payload, max_payload],
        marks={0: '0 Kg', 2500: '2500 Kg', 5000: '5000 Kg', 7500: '7500 Kg', 10000: '10000 Kg'}
    ),

    # TASK 4: Add a scatter chart to show the correlation between payload and launch success
    html.Div(dcc.Graph(id='success-payload-scatter-chart')),
])

# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
def update_pie_chart(selected_site):
    if selected_site == 'ALL':
        # Group by Launch Site and calculate success rate
```

```
        success_rates = spacex_df.groupby('Launch Site')['class'].mean().reset_index()
        success_rates['Success Rate'] = success_rates['class'] * 100

        fig = px.pie(success_rates,
                     values='Success Rate',
                     names='Launch Site',
                     title='Success Rate for All Launch Sites',
                     hover_data=['Success Rate'],
                     labels={'Success Rate': 'Success Rate (%)'})

        # Add percentage labels inside the pie chart
        fig.update_traces(textposition='inside', textinfo='percent+label')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == selected_site]
        fig = px.pie(filtered_df,
                     names='class',
                     title=f'Success vs Failure for {selected_site}',
                     labels={'class':'Outcome'},
                     color='class',
                     color_discrete_map={0: 'red', 1: 'green'})

        # Rename labels and add percentage labels inside the pie chart
        fig.update_traces(textposition='inside',
                          textinfo='percent+label')

    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_plot(selected_site, payload_range):
    low, high = payload_range
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= low) & (spacex_df['Payload Mass (kg)'] <= high)]

    if selected_site == 'ALL':
        fig = px.scatter(
            filtered_df,
            x='Payload Mass (kg)',
            y='class',
            color='Booster Version Category',
            title='Correlation between Payload and Success for All Sites',
            labels={'class': 'Launch Outcome'}
        )
    else:
        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'Correlation between Payload and Success for {selected_site}',
            labels={'class': 'Launch Outcome'}
        )
    return fig
```

```
# Run the app
if __name__ == '__main__':
    app.run_server()
```