# # 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, State

import plotly.express as px

import numpy as np

# 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',...)

                                html.Br(),

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

                                    options=[

                                        {'label': 'All Sites', 'value': 'ALL'},

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

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

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

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

                                        ],

                                        value='ALL',

                                        placeholder="Select a Launch Site here",

                                        searchable=True

                                        ),

                                # 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',...)

                                html.Div([

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

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

                                    marks={0: '0',

                                            2500: '2500',

                                            5000: '5000',

                                            7500: '7500',

                                            10000: '10000'},

                                    value=[0, 10000])

                                ]),

                                # 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

# Function decorator to specify function input and output

@app.callback([Output(component\_id='success-pie-chart', component\_property='figure'),

            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')])

#@app.callback([Output(component\_id='success-pie-chart', component\_property='figure'),

#            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')],

#            [State('success-pie-chart', 'figure'),

#            State('success-payload-scatter-chart', 'figure')]

#              )

#def get\_pie\_chart(entered\_site, slider\_value, o1, o2):

def get\_pie\_chart(entered\_site, payload):

    filtered\_df = spacex\_df

    if entered\_site == 'ALL':

        fig = px.pie(spacex\_df, values='class',

        names='Launch Site',

        title='Total Success Launces by Site')

        #fig2 = px.pie(spacex\_df, values='class',

        #names='Launch Site',

        #title='Total Success Launces by Site')

        fig2 = px.scatter(spacex\_df, x=spacex\_df['Payload Mass (kg)'], y=spacex\_df['class'],

        color ='Booster Version Category',

        title='Correlation between Payload and Success for all sites')

        return (fig, fig2)

    else:

        filtered\_df ['size'] = np.zeros(56)

        filtered\_df = spacex\_df[spacex\_df['Launch Site']==entered\_site].groupby(['Launch Site', 'class'], as\_index=False).count().reset\_index()

        fig = px.pie(filtered\_df, values='size',

        names='class',

        title='Total Success Launces for site '+ entered\_site)

        filtered\_df2 = spacex\_df[spacex\_df['Launch Site']==entered\_site]

        fig2 = px.scatter(filtered\_df2, x=filtered\_df2['Payload Mass (kg)'], y=filtered\_df2['class'],

        color ='Booster Version Category',

        title='Correlation between Payload and Success for '+ entered\_site)

        return (fig, fig2)

        # return the outcomes piechart for a selected site

# TASK 4:

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

#def get\_scatter\_chart(entered\_site2):

#    filtered\_df = spacex\_df

#    if entered\_site2 == 'ALL':

#        #px.scatter(df2, x=df2.index, y=df2.columns)

#        fig = px.scatter(spacex\_df, x='Payload Mass (kg)', y='class',

#        color ="Booster Version Category",

#        title='Correlation between Payload and Success for all sites')

#        return fig

#    else:

        #filtered\_df ['size'] = np.zeros(56)

        #filtered\_df = spacex\_df[spacex\_df['Launch Site']==entered\_site].groupby(['Launch Site', 'class'], as\_index=False).count().reset\_index()

        #fig = px.pie(filtered\_df, values='size',

        #names='class',

        #title='Total Success Launces for site '+ entered\_site)

 #       return fig

# Run the app

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

    app.run\_server()

Text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated with medium confidence

Application

Description automatically generated with medium confidence

Application

Description automatically generated with medium confidence

Only needs to connect the slider to the graph (everything works):

# 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, State

import plotly.express as px

import numpy as np

# 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',...)

                                html.Br(),

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

                                    options=[

                                        {'label': 'All Sites', 'value': 'ALL'},

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

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

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

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

                                        ],

                                        value='ALL',

                                        placeholder="Select a Launch Site here",

                                        searchable=True

                                        ),

                                # 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',...)

                                html.Div([

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

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

                                    marks={0: '0',

                                            2500: '2500',

                                            5000: '5000',

                                            7500: '7500',

                                            10000: '10000'},

                                    value=[0, 10000])

                                ]),

                                # 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

# Function decorator to specify function input and output

@app.callback([Output(component\_id='success-pie-chart', component\_property='figure'),

            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')])

#@app.callback([Output(component\_id='success-pie-chart', component\_property='figure'),

#            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')],

#            [State('success-pie-chart', 'figure'),

#            State('success-payload-scatter-chart', 'figure')]

#              )

#def get\_pie\_chart(entered\_site, slider\_value, o1, o2):

def get\_pie\_chart(entered\_site, payload\_range):

    filtered\_df = spacex\_df

    if entered\_site == 'ALL':

        fig = px.pie(spacex\_df, values='class',

        names='Launch Site',

        title='Total Success Launces by Site')

        #fig2 = px.pie(spacex\_df, values='class',

        #names='Launch Site',

        #title='Total Success Launces by Site')

        fig2 = px.scatter(spacex\_df, x=spacex\_df['Payload Mass (kg)'], y=spacex\_df['class'],

        color ='Booster Version Category',

        title='Correlation between Payload and Success for all sites')

        return (fig, fig2)

    else:

        filtered\_df ['size'] = np.zeros(56)

        filtered\_df = spacex\_df[spacex\_df['Launch Site']==entered\_site].groupby(['Launch Site', 'class'], as\_index=False).count().reset\_index()

        fig = px.pie(filtered\_df, values='size',

        names='class',

        title='Total Success Launces for site '+ entered\_site)

        filtered\_df2 = spacex\_df[spacex\_df['Launch Site']==entered\_site]

        fig2 = px.scatter(filtered\_df2, x=filtered\_df2['Payload Mass (kg)'], y=filtered\_df2['class'],

        color ='Booster Version Category',

        title='Correlation between Payload and Success for '+ entered\_site)

        fig2.update\_xaxes(range = payload\_range)

        return (fig, fig2)

        # return the outcomes piechart for a selected site

# TASK 4:

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

#def get\_scatter\_chart(entered\_site2):

#    filtered\_df = spacex\_df

#    if entered\_site2 == 'ALL':

#        #px.scatter(df2, x=df2.index, y=df2.columns)

#        fig = px.scatter(spacex\_df, x='Payload Mass (kg)', y='class',

#        color ="Booster Version Category",

#        title='Correlation between Payload and Success for all sites')

#        return fig

#    else:

        #filtered\_df ['size'] = np.zeros(56)

        #filtered\_df = spacex\_df[spacex\_df['Launch Site']==entered\_site].groupby(['Launch Site', 'class'], as\_index=False).count().reset\_index()

        #fig = px.pie(filtered\_df, values='size',

        #names='class',

        #title='Total Success Launces for site '+ entered\_site)

 #       return fig

# Run the app

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

    app.run\_server()

Text

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated