# 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',...)

                                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),

                                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,

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

                                                marks={0: '0', 2000:'2000',4000:'4000', 6000:'6000', 8000:'8000', 10000: '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

@app.callback(

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

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

def build\_graph(site\_dropdown):

    if site\_dropdown == 'ALL':

        piechart = px.pie(data\_frame = spacex\_df, names='Launch Site', values='class' ,title='Total Launches for All Sites')

        return piechart

    else:

        #specific\_df = spacex\_df['Launch Site']

        specific\_df=spacex\_df.loc[spacex\_df['Launch Site'] == site\_dropdown]

        piechart = px.pie(data\_frame = specific\_df, names='class',title='Total Launch for a Specific Site')

        return piechart

# 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\_graph(site\_dropdown, payload\_slider):

    if site\_dropdown == 'ALL':

        filtered\_data = spacex\_df[(spacex\_df['Payload Mass (kg)']>=payload\_slider[0])

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

        scatterplot = px.scatter(data\_frame=filtered\_data, x="Payload Mass (kg)", y="class",

        color="Booster Version Category")

        return scatterplot

    else:

        specific\_df=spacex\_df.loc[spacex\_df['Launch Site'] == site\_dropdown]

        filtered\_data = specific\_df[(specific\_df['Payload Mass (kg)']>=payload\_slider[0])

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

        scatterplot = px.scatter(data\_frame=filtered\_data, x="Payload Mass (kg)", y="class",

        color="Booster Version Category")

        return scatterplot

# Run the app

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

    app.run\_server()