# 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.graph\_objects as go  
import plotly.express as px  
from dash import no\_update  
  
  
# Create a dash application  
app = dash.Dash(\_\_name\_\_)  
  
# REVIEW1: Clear the layout and do not display exception till callback gets executed  
app.config.suppress\_callback\_exceptions = True  
  
# Read the airline data into pandas dataframe  
airline\_data =  pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',   
                            encoding = "ISO-8859-1",  
                            dtype={'Div1Airport': str, 'Div1TailNum': str,   
                                   'Div2Airport': str, 'Div2TailNum': str})  
  
  
# List of years   
year\_list = [i for i in range(2005, 2021, 1)]  
  
"""Compute graph data for creating yearly airline performance report   
  
Function that takes airline data as input and create 5 dataframes based on the grouping condition to be used for plottling charts and grphs.  
  
Argument:  
       
    df: Filtered dataframe  
      
Returns:  
   Dataframes to create graph.   
"""  
def compute\_data\_choice\_1(df):  
    # Cancellation Category Count  
    bar\_data = df.groupby(['Month','CancellationCode'])['Flights'].sum().reset\_index()  
    # Average flight time by reporting airline  
    line\_data = df.groupby(['Month','Reporting\_Airline'])['AirTime'].mean().reset\_index()  
    # Diverted Airport Landings  
    div\_data = df[df['DivAirportLandings'] != 0.0]  
    # Source state count  
    map\_data = df.groupby(['OriginState'])['Flights'].sum().reset\_index()  
    # Destination state count  
    tree\_data = df.groupby(['DestState', 'Reporting\_Airline'])['Flights'].sum().reset\_index()  
    return bar\_data, line\_data, div\_data, map\_data, tree\_data  
  
  
"""Compute graph data for creating yearly airline delay report  
  
This function takes in airline data and selected year as an input and performs computation for creating charts and plots.  
  
Arguments:  
    df: Input airline data.  
      
Returns:  
    Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.  
"""  
def compute\_data\_choice\_2(df):  
    # Compute delay averages  
    avg\_car = df.groupby(['Month','Reporting\_Airline'])['CarrierDelay'].mean().reset\_index()  
    avg\_weather = df.groupby(['Month','Reporting\_Airline'])['WeatherDelay'].mean().reset\_index()  
    avg\_NAS = df.groupby(['Month','Reporting\_Airline'])['NASDelay'].mean().reset\_index()  
    avg\_sec = df.groupby(['Month','Reporting\_Airline'])['SecurityDelay'].mean().reset\_index()  
    avg\_late = df.groupby(['Month','Reporting\_Airline'])['LateAircraftDelay'].mean().reset\_index()  
    return avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late  
  
  
# Application layout  
app.layout = html.Div(children=[   
                                # TASK1: Add title to the dashboard  
                                # Enter your code below. Make sure you have correct formatting.  
      
                                # REVIEW2: Dropdown creation  
                                # Create an outer division   
                                html.Div([  
                                    # Add an division  
                                    html.Div([  
                                        # Create an division for adding dropdown helper text for report type  
                                        html.Div(  
                                            [  
                                            html.H2('Report Type:', style={'margin-right': '2em'}),  
                                            ]  
                                        ),  
                                        # TASK2: Add a dropdown  
                                        # Enter your code below. Make sure you have correct formatting.  
                                          
                                    # Place them next to each other using the division style  
                                    ], style={'display':'flex'}),  
                                      
                                   # Add next division   
                                   html.Div([  
                                       # Create an division for adding dropdown helper text for choosing year  
                                        html.Div(  
                                            [  
                                            html.H2('Choose Year:', style={'margin-right': '2em'})  
                                            ]  
                                        ),  
                                        dcc.Dropdown(id='input-year',   
                                                     # Update dropdown values using list comphrehension  
                                                     options=[{'label': i, 'value': i} for i in year\_list],  
                                                     placeholder="Select a year",  
                                                     style={'width':'80%', 'padding':'3px', 'font-size': '20px', 'text-align-last' : 'center'}),  
                                            # Place them next to each other using the division style  
                                            ], style={'display': 'flex'}),    
                                          ]),  
                                  
                                # Add Computed graphs  
                                # REVIEW3: Observe how we add an empty division and providing an id that will be updated during callback  
                                html.Div([ ], id='plot1'),  
      
                                html.Div([  
                                        html.Div([ ], id='plot2'),  
                                        html.Div([ ], id='plot3')  
                                ], style={'display': 'flex'}),  
                                  
                                # TASK3: Add a division with two empty divisions inside. See above disvision for example.  
                                # Enter your code below. Make sure you have correct formatting.  
                                 
                                ])  
  
# Callback function definition  
# TASK4: Add 5 ouput components  
# Enter your code below. Make sure you have correct formatting.  
@app.callback( [....],  
               [Input(component\_id='input-type', component\_property='value'),  
                Input(component\_id='input-year', component\_property='value')],  
               # REVIEW4: Holding output state till user enters all the form information. In this case, it will be chart type and year  
               [State("plot1", 'children'), State("plot2", "children"),  
                State("plot3", "children"), State("plot4", "children"),  
                State("plot5", "children")  
               ])  
# Add computation to callback function and return graph  
def get\_graph(chart, year, children1, children2, c3, c4, c5):  
        
        # Select data  
        df =  airline\_data[airline\_data['Year']==int(year)]  
         
        if chart == 'OPT1':  
            # Compute required information for creating graph from the data  
            bar\_data, line\_data, div\_data, map\_data, tree\_data = compute\_data\_choice\_1(df)  
              
            # Number of flights under different cancellation categories  
            bar\_fig = px.bar(bar\_data, x='Month', y='Flights', color='CancellationCode', title='Monthly Flight Cancellation')  
              
            # TASK5: Average flight time by reporting airline  
            # Enter your code below. Make sure you have correct formatting.  
              
              
            # Percentage of diverted airport landings per reporting airline  
            pie\_fig = px.pie(div\_data, values='Flights', names='Reporting\_Airline', title='% of flights by reporting airline')  
              
            # REVIEW5: Number of flights flying from each state using choropleth  
            map\_fig = px.choropleth(map\_data,  # Input data  
                    locations='OriginState',   
                    color='Flights',    
                    hover\_data=['OriginState', 'Flights'],   
                    locationmode = 'USA-states', # Set to plot as US States  
                    color\_continuous\_scale='GnBu',  
                    range\_color=[0, map\_data['Flights'].max()])   
            map\_fig.update\_layout(  
                    title\_text = 'Number of flights from origin state',   
                    geo\_scope='usa') # Plot only the USA instead of globe  
              
            # TASK6: Number of flights flying to each state from each reporting airline  
            # Enter your code below. Make sure you have correct formatting.  
              
              
              
            # REVIEW6: Return dcc.Graph component to the empty division  
            return [dcc.Graph(figure=tree\_fig),   
                    dcc.Graph(figure=pie\_fig),  
                    dcc.Graph(figure=map\_fig),  
                    dcc.Graph(figure=bar\_fig),  
                    dcc.Graph(figure=line\_fig)  
                   ]  
        else:  
            # REVIEW7: This covers chart type 2 and we have completed this exercise under Flight Delay Time Statistics Dashboard section  
            # Compute required information for creating graph from the data  
            avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_data\_choice\_2(df)  
              
            # Create graph  
            carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrrier delay time (minutes) by airline')  
            weather\_fig = px.line(avg\_weather, x='Month', y='WeatherDelay', color='Reporting\_Airline', title='Average weather delay time (minutes) by airline')  
            nas\_fig = px.line(avg\_NAS, x='Month', y='NASDelay', color='Reporting\_Airline', title='Average NAS delay time (minutes) by airline')  
            sec\_fig = px.line(avg\_sec, x='Month', y='SecurityDelay', color='Reporting\_Airline', title='Average security delay time (minutes) by airline')  
            late\_fig = px.line(avg\_late, x='Month', y='LateAircraftDelay', color='Reporting\_Airline', title='Average late aircraft delay time (minutes) by airline')  
              
            return[dcc.Graph(figure=carrier\_fig),   
                   dcc.Graph(figure=weather\_fig),   
                   dcc.Graph(figure=nas\_fig),   
                   dcc.Graph(figure=sec\_fig),   
                   dcc.Graph(figure=late\_fig)]  
  
  
# Run the app  
if \_\_name\_\_ == '\_\_main\_\_':  
    app.run\_server()

--

latoya dupree  
apple bank for savings