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

app=dash.Dash(\_\_name\_\_)

app.config.suppress\_callback\_exceptions = True

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})

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):

       bar\_data = df.groupby(['Month','CancellationCode'])['Flights'].sum().reset\_index()

    line\_data = df.groupby(['Month','Reporting\_Airline'])['AirTime'].mean().reset\_index()

       div\_data = df[df['DivAirportLandings'] != 0.0]

    map\_data = df.groupby(['OriginState'])['Flights'].sum().reset\_index()

    tree\_data = df.groupby(['DestState', 'Reporting\_Airline'])['Flights'].sum().reset\_index()

    return bar\_data, line\_data, div\_data, map\_data, tree\_data

Arguments:

    df: Input airline data.

def compute\_data\_choice\_2(df):

    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

app.layout=html.Div(children=[html.H1('US Domestic Airline Flights Performance',

                                style={'textAlign': 'center', 'color':

'#503D36','font-size': 24})],

                               html.Div([

                                    html.Div([

html.Div(

                                            [

                                            html.H2('Report Type:',

style={'margin-right': '2em'}),

                                            ]

                                        ),

                                                                                  dcc.Dropdown(id='input-type',

options=[{'label': 'Yearly Airline Performance Report',

'value': 'OPT1'},

                           {'label': 'Yearly Airline Delay Report', 'value':

'OPT2'}

                          ],

                  placeholder='Select a report type',

                  style={'width':'80%', 'padding':'3px','font-

size':20,'textAlign': 'center'})

                          ], style={'display':'flex'}),

                                   html.Div([

                                             html.Div(

                                            [

                                            html.H2('Choose Year:',

style={'margin-right': '2em'})

                                            ]

                                        ),

dcc.Dropdown(id='input-year',

             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'}),

                                                                                       ], style={'display': 'flex'}),

        ]),

                                html.Div([ ], id='plot1'),

                                html.Div([

                                        html.Div([ ], id='plot2'),

                                        html.Div([ ], id='plot3')

                                ], style={'display': 'flex'}),

                                html.Div([

                                        html.Div([ ], id='plot4'),

                                        html.Div([ ], id='plot5')

                                ], style={'display': 'flex'}),

@app.callback( [Output(component\_id='plot1', component\_property='children'),

                Output(component\_id='plot2', component\_property='children'),

               Output(component\_id='plot3', component\_property='children'),

               Output(component\_id='plot4', component\_property='children'),

               Output(component\_id='plot5', component\_property='children')],

               [Input(component\_id='input-type', component\_property='value'),

                Input(component\_id='input-year', component\_property='value')],

               [State("plot1", 'children'), State("plot2", "children"),

                State("plot3", "children"), State("plot4", "children"),

                State("plot5", "children")

               ])

def get\_graph(chart, year, children1, children2, c3, c4, c5):

               df =  airline\_data[airline\_data['Year']==int(year)]

        if chart == 'OPT1':

                       bar\_data, line\_data, div\_data, map\_data, tree\_data = compute\_data\_choice\_1(df)

            bar\_fig = px.bar(bar\_data, x='Month', y='Flights',

color='CancellationCode', title='Monthly Flight

Cancellation')

            line\_fig = px.line(line\_data, x='Month', y='AirTime',

color='Reporting\_Airline', title='Average monthly

flight time (minutes) by airline')

            pie\_fig = px.pie(div\_data, values='Flights',

names='Reporting\_Airline', title='% of flights by

reporting airline')

            map\_fig = px.choropleth(map\_data,

                    locations='OriginState',

                    color='Flights',

                    hover\_data=['OriginState', 'Flights'],

                    locationmode = 'USA-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')

            tree\_fig = px.treemap(tree\_data, path=['DestState',

'Reporting\_Airline'],

                      values='Flights',

                      color='Flights',

                      color\_continuous\_scale='RdBu',

                      title='Flight count by airline to destination state'

                )

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

    app.run\_server()

            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:

            avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late =

compute\_data\_choice\_2(df)

            carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay',

color='Reporting\_Airline', title='Average

carrier 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)]

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

    app.run\_server()