import dash  
import dash\_core\_components as dcc  
import dash\_html\_components as html  
from dash.dependencies import Input, Output  
import pandas as pd  
import plotly.graph\_objs as go  
  
# Load CSV file from Datasets folder  
df1 = pd.read\_csv('../Datasets/CoronavirusTotal.csv')  
df2 = pd.read\_csv('../Datasets/CoronaTimeSeries.csv')  
df3 = pd.read\_csv('../Datasets/Olympic2016Rio.csv')  
df4 = pd.read\_csv('../Datasets/Weather2014-15.csv')  
  
app = dash.Dash()  
  
# Bar chart data  
barchart\_df = df1[df1['Country'] == 'US']  
barchart\_df = barchart\_df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
barchart\_df = barchart\_df.groupby(['State'])['Confirmed'].sum().reset\_index()  
barchart\_df = barchart\_df.sort\_values(by=['Confirmed'], ascending=[False]).head(20)  
data\_barchart = [go.Bar(x=barchart\_df['State'], y=barchart\_df['Confirmed'])]  
  
# Stack bar chart data  
stackbarchart\_df = df1.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
stackbarchart\_df['Unrecovered'] = stackbarchart\_df['Confirmed'] - stackbarchart\_df['Deaths'] - stackbarchart\_df[  
 'Recovered']  
stackbarchart\_df = stackbarchart\_df[(stackbarchart\_df['Country'] != 'China')]  
stackbarchart\_df = stackbarchart\_df.groupby(['Country']).agg(  
 {'Confirmed': 'sum', 'Deaths': 'sum', 'Recovered': 'sum', 'Unrecovered': 'sum'}).reset\_index()  
stackbarchart\_df = stackbarchart\_df.sort\_values(by=['Confirmed'], ascending=[False]).head(20).reset\_index()  
trace1\_stackbarchart = go.Bar(x=stackbarchart\_df['Country'], y=stackbarchart\_df['Unrecovered'], name='Under Treatment',  
 marker={'color': '#CD7F32'})  
trace2\_stackbarchart = go.Bar(x=stackbarchart\_df['Country'], y=stackbarchart\_df['Recovered'], name='Recovered',  
 marker={'color': '#9EA0A1'})  
trace3\_stackbarchart = go.Bar(x=stackbarchart\_df['Country'], y=stackbarchart\_df['Deaths'], name='Deaths',  
 marker={'color': '#FFD700'})  
data\_stackbarchart = [trace1\_stackbarchart, trace2\_stackbarchart, trace3\_stackbarchart]  
  
# Line Chart  
line\_df = df2  
line\_df['Date'] = pd.to\_datetime(line\_df['Date'])  
data\_linechart = [go.Scatter(x=line\_df['Date'], y=line\_df['Confirmed'], mode='lines', name='Death')]  
  
# Multi Line Chart  
multiline\_df = df2  
multiline\_df['Date'] = pd.to\_datetime(multiline\_df['Date'])  
trace1\_multiline = go.Scatter(x=multiline\_df['Date'], y=multiline\_df['Death'], mode='lines', name='Death')  
trace2\_multiline = go.Scatter(x=multiline\_df['Date'], y=multiline\_df['Recovered'], mode='lines', name='Recovered')  
trace3\_multiline = go.Scatter(x=multiline\_df['Date'], y=multiline\_df['Unrecovered'], mode='lines', name='Under Treatment')  
data\_multiline = [trace1\_multiline, trace2\_multiline, trace3\_multiline]  
  
# Bubble chart  
bubble\_df = df1.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
bubble\_df['Unrecovered'] = bubble\_df['Confirmed'] - bubble\_df['Deaths'] - bubble\_df['Recovered']  
bubble\_df = bubble\_df[(bubble\_df['Country'] != 'China')]  
bubble\_df = bubble\_df.groupby(['Country']).agg(  
 {'Confirmed': 'sum', 'Recovered': 'sum', 'Unrecovered': 'sum'}).reset\_index()  
data\_bubblechart = [  
 go.Scatter(x=bubble\_df['Recovered'],  
 y=bubble\_df['Unrecovered'],  
 text=bubble\_df['Country'],  
 mode='markers',  
 marker=dict(size=bubble\_df['Confirmed'] / 200, color=bubble\_df['Confirmed'] / 200, showscale=True))  
]  
  
# Heatmap  
data\_heatmap = [go.Heatmap(x=df2['Day'],  
 y=df2['WeekofMonth'],  
 z=df2['Recovered'].values.tolist(),  
 colorscale='Jet')]  
  
# Olympic Barchart  
olympic\_barchart\_df = df3.sort\_values(by=['Total'], ascending=[False]).head(20)  
olympic\_barchart\_data = [go.Bar(x=olympic\_barchart\_df['NOC'], y=olympic\_barchart\_df['Total'])]  
  
# Olympic Stack Bar Chart  
trace1 = go.Bar(x=olympic\_barchart\_df['NOC'], y=olympic\_barchart\_df['Gold'], name='Gold', marker={'color': '#FFD700'})  
trace2 = go.Bar(x=olympic\_barchart\_df['NOC'], y=olympic\_barchart\_df['Silver'], name='Silver', marker={'color': '#9EA0A1'})  
trace3 = go.Bar(x=olympic\_barchart\_df['NOC'], y=olympic\_barchart\_df['Bronze'], name='Bronze', marker={'color': '#CD7F32'})  
olympic\_stackbarchart\_data = [trace1, trace2, trace3]  
  
# Weather Line Chart & Multi Line Chart  
months = ["July", "August", "September", "October", "November", "December",  
 "January", "February", "March", "April", "May", "June"]  
weather\_max\_line\_df = pd.DataFrame()  
weather\_min\_line\_df = pd.DataFrame()  
weather\_mean\_line\_df = pd.DataFrame()  
  
for x in months:  
 weather\_filtered\_df = df4[df4['month'] == x]  
 new\_weather\_df = weather\_filtered\_df.sort\_values(by=['actual\_max\_temp'], ascending=[False]).head(31)  
 maxTemp = new\_weather\_df.iloc[0]['actual\_max\_temp']  
 weather\_max\_line\_df = weather\_max\_line\_df.append(new\_weather\_df.iloc[[0]])  
  
 new\_weather\_df = weather\_filtered\_df.sort\_values(by=['actual\_min\_temp'], ascending=[True]).head(31)  
 minTemp = new\_weather\_df.iloc[0]['actual\_min\_temp']  
 weather\_min\_line\_df = weather\_min\_line\_df.append(new\_weather\_df.iloc[[0]])  
  
 meanTemp = {'month': [x], 'actual\_mean\_temp': [(maxTemp + minTemp) / 2]}  
 weather\_mean\_line\_df = weather\_mean\_line\_df.append(pd.DataFrame(meanTemp, columns=['month', 'actual\_mean\_temp']))  
  
weather\_linechart\_data = [go.Scatter(x=weather\_max\_line\_df['month'], y=weather\_max\_line\_df['actual\_max\_temp'], mode='lines', name='Temp')]  
  
weather\_trace1 = go.Scatter(x=weather\_max\_line\_df['month'], y=weather\_max\_line\_df['actual\_max\_temp'], mode='lines', name='Max')  
weather\_trace2 = go.Scatter(x=weather\_min\_line\_df['month'], y=weather\_min\_line\_df['actual\_min\_temp'], mode='lines', name='Min')  
weather\_trace3 = go.Scatter(x=weather\_mean\_line\_df['month'], y=weather\_mean\_line\_df['actual\_mean\_temp'], mode='lines', name='Mean')  
weather\_multiline\_data = [weather\_trace1, weather\_trace2, weather\_trace3]  
  
# Weather Bubble Chart  
weather\_max\_df = pd.DataFrame()  
weather\_min\_df = pd.DataFrame()  
for x in months:  
 # Filtering month  
 weather\_filtered\_df = df4[df4['month'] == x]  
  
 # Sorting values and select max average temp  
 new\_weather\_df = weather\_filtered\_df.sort\_values(by=['average\_max\_temp'], ascending=[False]).head(31)  
  
 weather\_max\_df = weather\_max\_df.append(new\_weather\_df.iloc[[0]])  
  
 # Sorting values and select min average temp  
 new\_weather\_df = weather\_filtered\_df.sort\_values(by=['average\_min\_temp'], ascending=[True]).head(31)  
  
 weather\_min\_df = weather\_min\_df.append(new\_weather\_df.iloc[[0]])  
  
weather\_bubble\_data = [  
 go.Scatter(x=weather\_min\_df["average\_min\_temp"],  
 y=weather\_max\_df["average\_max\_temp"],  
 text=weather\_min\_df["month"],  
 mode="markers",  
 marker=dict(size=weather\_max\_df["average\_max\_temp"],  
 color=weather\_max\_df["average\_max\_temp"], showscale=True))  
]  
  
# Weather Heatmap  
weather\_heatmap\_data = [go.Heatmap(x=df4['day'], y=df4['month'], z=df4['record\_max\_temp'].values.tolist(),  
 colorscale='Jet')]  
  
# Layout  
app.layout = html.Div(children=[  
 html.H1(children='Python Dash',  
 style={  
 'textAlign': 'center',  
 'color': '#ef3e18'  
 }  
 ),  
 html.Div('Web dashboard for Data Visualization using Python', style={'textAlign': 'center'}),  
 html.Div('Coronavirus COVID-19 Global Cases - 1/22/2020 to 3/17/2020', style={'textAlign': 'center'}),  
 html.Br(),  
 html.Br(),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Interactive Bar chart', style={'color': '#df1e56'}),  
 html.Div('This bar chart represent the number of confirmed cases in the first 20 countries of selected continent.'),  
 dcc.Graph(id='graph1'),  
 html.Div('Please select a continent', style={'color': '#ef3e18', 'margin':'10px'}),  
 dcc.Dropdown(  
 id='select-continent',  
 options=[  
 {'label': 'Asia', 'value': 'Asia'},  
 {'label': 'Africa', 'value': 'Africa'},  
 {'label': 'Europe', 'value': 'Europe'},  
 {'label': 'North America', 'value': 'North America'},  
 {'label': 'Oceania', 'value': 'Oceania'},  
 {'label': 'South America', 'value': 'South America'}  
 ],  
 value='Europe'  
 ),  
 html.Br(),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Bar chart', style={'color': '#df1e56'}),  
 html.Div('This bar chart represent the number of confirmed cases in the first 20 states of the US.'),  
 dcc.Graph(id='graph2',  
 figure={  
 'data': data\_barchart,  
 'layout': go.Layout(title='Corona Virus Confirmed Cases in The US',  
 xaxis={'title': 'States'}, yaxis={'title': 'Number of confirmed cases'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Stack bar chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This stack bar chart represent the CoronaVirus deaths, recovered and under treatment of all reported first 20 countries except China.'),  
 dcc.Graph(id='graph3',  
 figure={  
 'data': data\_stackbarchart,  
 'layout': go.Layout(title='Corona Virus Cases in the first 20 country expect China',  
 xaxis={'title': 'Country'}, yaxis={'title': 'Number of cases'},  
 barmode='stack')  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Line chart', style={'color': '#df1e56'}),  
 html.Div('This line chart represent the Corona Virus confirmed cases of all reported cases in the given period.'),  
 dcc.Graph(id='graph4',  
 figure={  
 'data': data\_linechart,  
 'layout': go.Layout(title='Corona Virus Confirmed Cases From 2020-01-22 to 2020-03-17',  
 xaxis={'title': 'Date'}, yaxis={'title': 'Number of cases'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Multi Line chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This line chart represent the CoronaVirus death, recovered and under treatment cases of all reported cases in the given period.'),  
 dcc.Graph(id='graph5',  
 figure={  
 'data': data\_multiline,  
 'layout': go.Layout(  
 title='Corona Virus Death, Recovered and under treatment Cases From 2020-01-22 to 2020-03-17',  
 xaxis={'title': 'Date'}, yaxis={'title': 'Number of cases'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Bubble chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This bubble chart represent the Corona Virus recovered and under treatment of all reported countries except China.'),  
 dcc.Graph(id='graph6',  
 figure={  
 'data': data\_bubblechart,  
 'layout': go.Layout(title='Corona Virus Confirmed Cases',  
 xaxis={'title': 'Recovered Cases'}, yaxis={'title': 'under Treatment Cases'},  
 hovermode='closest')  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Heat map', style={'color': '#df1e56'}),  
 html.Div(  
 'This heat map represent the Corona Virus recovered cases of all reported cases per day of week and week of month.'),  
 dcc.Graph(id='graph7',  
 figure={  
 'data': data\_heatmap,  
 'layout': go.Layout(title='Corona Virus Recovered Cases',  
 xaxis={'title': 'Day of Week'}, yaxis={'title': 'Week of Month'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Olympic bar chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This bar chart represents the total medals won by the top 20 countries in the 2016 Olympics.'),  
 dcc.Graph(id='graph8',  
 figure={  
 'data': olympic\_barchart\_data,  
 'layout': go.Layout(title='2016 Olympics Top 20 Countries by Total Medals',  
 xaxis={'title': 'Country'}, yaxis={'title': 'Medals'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Olympic stack bar chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This stack bar chart represents the total gold, silver and bronze medals won by the top 20 countries in the 2016 Olympics.'),  
 dcc.Graph(id='graph9',  
 figure={  
 'data': olympic\_stackbarchart\_data,  
 'layout': go.Layout(title='2016 Olympics Top 20 Countries by Total Medals',  
 xaxis={'title': 'Country'}, yaxis={'title': 'Medals'},  
 barmode='stack')  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Line chart', style={'color': '#df1e56'}),  
 html.Div('This line chart represent the max temp in each month from July 2014 to June 2105'),  
 dcc.Graph(id='graph10',  
 figure={  
 'data': weather\_linechart\_data,  
 'layout': go.Layout(title='Weather for July 2014 - June 2015',  
 xaxis={'title': 'Month'}, yaxis={'title': 'Max temp (F)'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Multi Line chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This multi line chart represent the max, min and mean temp in each month from July 2014 to June 2105.'),  
 dcc.Graph(id='graph11',  
 figure={  
 'data': weather\_multiline\_data,  
 'layout': go.Layout(  
 title='Weather for July 2014 - June 2015',  
 xaxis={'title': 'Month'}, yaxis={'title': 'Temperature(F)'})  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Bubble chart', style={'color': '#df1e56'}),  
 html.Div(  
 'This bubble chart represents the relation of min temp to max temp each month from July 2014 to June 2015.'),  
 dcc.Graph(id='graph12',  
 figure={  
 'data': weather\_bubble\_data,  
 'layout': go.Layout(title='Weather for July 2014 - June 2015',  
 xaxis={'title': 'Min Tempterature (F)'}, yaxis={'title': 'Max Tempterature (F)'},  
 hovermode='closest')  
 }  
 ),  
 html.Hr(style={'color': '#7FDBFF'}),  
 html.H3('Heat map', style={'color': '#df1e56'}),  
 html.Div(  
 'This heatmap represents the record max temperature each day of the week for months July 2014 to June 2015.'),  
 dcc.Graph(id='graph13',  
 figure={  
 'data': weather\_heatmap\_data,  
 'layout': go.Layout(title='Weather for July 2014 - June 2015',  
 xaxis={'title': 'Day of Week'}, yaxis={'title': 'Month'})  
 }  
 )  
])  
  
  
@app.callback(Output('graph1', 'figure'),  
 [Input('select-continent', 'value')])  
def update\_figure(selected\_continent):  
 filtered\_df = df1[df1['Continent'] == selected\_continent]  
  
 filtered\_df = filtered\_df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
 new\_df = filtered\_df.groupby(['Country'])['Confirmed'].sum().reset\_index()  
 new\_df = new\_df.sort\_values(by=['Confirmed'], ascending=[False]).head(20)  
 data\_interactive\_barchart = [go.Bar(x=new\_df['Country'], y=new\_df['Confirmed'])]  
 return {'data': data\_interactive\_barchart, 'layout': go.Layout(title='Corona Virus Confirmed Cases in '+selected\_continent,  
 xaxis={'title': 'Country'},  
 yaxis={'title': 'Number of confirmed cases'})}  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server()









