dash\_basics.py

# Import required packages

import pandas as pd

import plotly.express as px

import dash

from dash import html

from dash import dcc

# 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})

# Randomly sample 500 data points. Setting the random state to be 42 so that we get same result.

data = airline\_data.sample(n=500, random\_state=42)

# Pie Chart Creation

fig = px.pie(data, values='Flights', names='DistanceGroup', title='Distance group proportion by flights')

# Create a dash application

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

# Get the layout of the application and adjust it.

# Create an outer division using html.Div and add title to the dashboard using html.H1 component

# Add description about the graph using HTML P (paragraph) component

# Finally, add graph component.

app.layout = html.Div(children=[html.H1('Airline Dashboard',

style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),

                                html.P('Proportion of distance group (250 mile distance interval group) by flights.', style={'textAlign':'center', 'color': '#F57241'}),

                                dcc.Graph(figure=fig),

                    ])

# Run the application

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

    app.run\_server()

data\_interactivity.py

# Import required libraries

import pandas as pd

import plotly.graph\_objects as go

import dash

from dash import html

from dash import dcc

from dash.dependencies import Input, Output

# 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})

# Create a dash application

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

# Get the layout of the application and adjust it.

# Create an outer division using html.Div and add title to the dashboard using html.H1 component

# Add a html.Div and core input text component

# Finally, add graph component.

app.layout = html.Div(children=[html.H1('Airline Performance Dashboard',

style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),

    html.Div(["Input Year", dcc.Input(id='input-year', value=2010, type='number', style={'height' : '50px', 'font-size': 35})],

    style={'font-size': 40}),

    html.Br(),

    html.Br(),

    html.Div(dcc.Graph(id='line-plot')),])

# add callback decorator

@app.callback(Output(component\_id='line-plot', component\_property='figure'),

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

# Add computation to callback function and return graph

def get\_graph(entered\_year):

    # Select data based on the entered year

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

    # Group the data by Month and compute average over arrival delay time.

    line\_data = df.groupby('Month')['ArrDelay'].mean().reset\_index()

    #

    fig = go.Figure(data=go.Scatter(x=line\_data['Month'], y=line\_data['ArrDelay'], mode='lines', marker=dict(color='green')))

    fig.update\_layout(title='Month vs Average Flight Delay Time',xaxis\_title='Month', yaxis\_title='ArrDelay')

    return fig

# Run the app

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

    app.run\_server()

flight\_details.py

# Import required libraries

import pandas as pd

import plotly.graph\_objects as go

import plotly.express as px

import dash

from dash import html

from dash import dcc

from dash.dependencies import Input, Output

# 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})

# Create a dash application

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

# Build dash app layout

app.layout = html.Div(children=[ html.H1('Flight Delay Time Statistics',

style={'textAlign': 'center', 'color': '#503D36', 'font-size': 30}),

html.Div(["Input Year: ", dcc.Input(id='input-year', value=2010, type='number', style={'height' : '35px', 'font-size': 30})],

        style={'font-size': 30}),

        html.Br(),

        html.Br(),

        html.Div([

                 html.Div(dcc.Graph(id='carrier-plot')),

                 html.Div(dcc.Graph(id='weather-plot'))

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

        html.Div([

                 html.Div(dcc.Graph(id='nas-plot')),

                 html.Div(dcc.Graph(id='security-plot'))

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

        html.Div(dcc.Graph(id='late-plot'), style={'width':'65%'})

    ])

""" Compute\_info function description

This function takes in airline data and selected year as an input and performs computation for creating charts and plots.

Arguments:

    airline\_data: Input airline data.

    entered\_year: Input year for which computation needs to be performed.

Returns:

    Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.

"""

def compute\_info(airline\_data, entered\_year):

    # Select data

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

    # 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

# Callback decorator

@app.callback( [

               Output(component\_id='carrier-plot', component\_property='figure'),

               Output(component\_id='weather-plot', component\_property='figure'),

               Output(component\_id='nas-plot', component\_property='figure'),

               Output(component\_id='security-plot', component\_property='figure'),

               Output(component\_id='late-plot', component\_property='figure')

               ],

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

# Computation to callback function and return graph

def get\_graph(entered\_year):

    # Compute required information for creating graph from the data

    avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_info(airline\_data, entered\_year)

    # Line plot for carrier delay

    carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrier delay time (minutes) by airline')

    # Line plot for weather delay

    weather\_fig = px.line(avg\_weather, x='Month', y='WeatherDelay', color='Reporting\_Airline', title='Average weather delay time (minutes) by airline')

    # Line plot for nas delay

    nas\_fig = px.line(avg\_NAS, x='Month', y='NASDelay', color='Reporting\_Airline', title='Average NAS delay time (minutes) by airline')

    # Line plot for security delay

    sec\_fig = px.line(avg\_sec, x='Month', y='SecurityDelay', color='Reporting\_Airline', title='Average security delay time (minutes) by airline')

    # Line plot for late aircraft delay

    late\_fig = px.line(avg\_late, x='Month', y='LateAircraftDelay', color='Reporting\_Airline', title='Average late delay time (minutes) by airline')

    return[carrier\_fig, weather\_fig, nas\_fig, sec\_fig, late\_fig]

# Run the app

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

    app.run\_server()

dash\_layout.py

# Import required packages

import pandas as pd

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.express as px

from dash.dependencies import Input, Output

# Add Dataframe

df = pd.DataFrame({

    "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],

    "Amount": [4, 1, 2, 2, 4, 5],

    "City": ["SF", "SF", "SF", "NYC", "MTL", "NYC"]

})

# Add a bar graph figure

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

app = dash.Dash()

app.layout = html.Div(children=[

    html.H1(

        children='Dashboard',

        style={

            'textAlign': 'center'

        }

    )

    ,# Create dropdown

    dcc.Dropdown(options=[

            {'label': 'New York City', 'value': 'NYC'},

            {'label': 'Montréal', 'value': 'MTL'},

            {'label': 'San Francisco', 'value': 'SF'}

        ],

        value='NYC' # Providing a vallue to dropdown

    )

    ,# Bar graph

    dcc.Graph(id='example-graph-2',figure=fig)

])

# Run Application

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

    app.run\_server()

Dash\_Auto.py

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\_\_)

# REVIEW1: Clear the layout and do not display exception till callback gets executed

app.config.suppress\_callback\_exceptions = True

# Read the automobiles data into pandas dataframe

auto\_data =  pd.read\_csv('automobileEDA.csv',

                            encoding = "ISO-8859-1",

                            )

#Layout Section of Dash

app.layout = html.Div(children=[#TASK 3A

    html.H1('Car Automobile Components',

                                style={'textAlign': 'center', 'color': '#503D36',

                                'font-size': 24}),

     #outer division starts

    html.Div([

                   # First inner divsion for  adding dropdown helper text for Selected Drive wheels

                    html.Div(

                            #TASK 3B

                            [html.H2('Drive Wheels Type:', style={'margin-right': '2em'}),]

                     ),

                    #TASK 3C

                    dcc.Dropdown(

                            id='demo-dropdown',

                        options=[

                             {'label': 'Rear Wheel Drive', 'value': 'rwd'},

                            {'label': 'Front Wheel Drive', 'value': 'fwd'},

                             {'label': 'Four Wheel Drive', 'value': '4wd'}

                        ],

                        value='rwd'

                    ),

                    #Second Inner division for adding 2 inner divisions for 2 output graphs

                    html.Div([

                        #TASK 3D

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

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

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

    ])

    #outer division ends

])

#layout ends

#Place to add @app.callback Decorator

#TASK 3E

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

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

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

#Place to define the callback function .

#TASK 3F

def display\_selected\_drive\_charts(value):

   filtered\_df = auto\_data[auto\_data['drive-wheels']==value].groupby(['drive-wheels','body-style'],as\_index=False). \

            mean()

   filtered\_df = filtered\_df

   fig1 = px.pie(filtered\_df, values='price', names='body-style', title="Pie Chart")

   fig2 = px.bar(filtered\_df, x='body-style', y='price', title='Bar Chart')

   return [dcc.Graph(figure=fig1),

            dcc.Graph(figure=fig2) ]

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

    app.run\_server()