```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import pandas as pd
import plotly.graph_objs as go
import plotly.express as px

# Load the data using pandas
data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-
SkillsNetwork/Data%20Files/historical_automobile_sales.csv')

# Initialize the Dash app
app = dash.Dash(__name__)

# Set the title of the dashboard
#app.title = "Automobile Statistics Dashboard"

#---------------------------------------------------------------------------
----
# Create the dropdown menu options
dropdown_options = [
    {'label': 'Yearly Statistics', 'value': 'Yearly Statistics'},
    {'label': 'Recession Period Statistics', 'value': 'Recession Period
Statistics'}
]
# List of years
year_list = [i for i in range(1980, 2024, 1)]
#---------------------------------------------------------------------------
----------
# Create the layout of the app
app.layout = html.Div([
    #TASK 2.1 Add title to the dashboard
    html.H1("Automobile Sales Statistics Dashboard",
            style = {'textAlign': 'center', 'color': '#503D36', 'font-size':
'24px'}),

    html.Div([#TASK 2.2: Add two dropdown menus
        html.Label("Select Statistics:"),
        dcc.Dropdown(
            id='dropdown-statistics',
            options=[
                {'label':'Yearly Statistics', 'value':'Yearly Statistcis'},
                {'label':'Recession Period Statistics', 'value':'Recession
Period Statistics'}
            ],
            placeholder='Select a report type',
            value='Select Statistics',
```

```
            style={'width':'80%', 'padding':'3px', 'font-size':20,
'textAlignLast':'center'}
        )
    ]),
    html.Div(dcc.Dropdown(
            id='select-year',
            options=[{'label': i, 'value': i} for i in year_list],
            value='Select a year'
        )),
    html.Div([#TASK 2.3: Add a division for output display
    html.Div(id='output-container', className='chart-grid',
style={'display':'flex'}),])
])
#TASK 2.4: Creating Callbacks
# Define the callback function to update the input container based on the
selected statistics
@app.callback(
    Output(component_id='select-year', component_property='disabled'),
    Input(component_id='report-type-dropdown',component_property='value'))
def update_input_container(selected_statistics):
    if selected_statistics =='Yearly Statistics':
        return False
    else:
        return True

#Callback for plotting
# Define the callback function to update the input container based on the
selected statistics
@app.callback(
    Output(component_id='output-container', component_property='children'),
    [Input(component_id='dropdown-statistics', component_property='value'),
Input(component_id='select-year', component_property='value')])
def update_output_container(selected_statistics, input_year):
    if selected_statistics == 'Recession Period Statistics':
        # Filter the data for recession periods
        recession_data = data[data['Recession'] == 1]

#TASK 2.5: Create and display graphs for Recession Report Statistics

#Plot 1 Automobile sales fluctuate over Recession Period (year wise)
        # use groupby to create relevant data for plotting
        yearly_rec=recession_data.groupby('Year')['Automobile_Sales'].mean().r
eset_index()
        R_chart1 = dcc.Graph(
            figure=px.line(yearly_rec,
                x='Year',
                y='Automobile_Sales',
                title="Average Automobile Sales fluctuation over Recession
Period"))

#Plot 2 Calculate the average number of vehicles sold by vehicle type
        # use groupby to create relevant data for plotting
        average_sales =
recession_data.groupby('Vehicle_Type')['Automobile_Sales'].mean().reset_index(
)
        R_chart2  = dcc.Graph(
```

```
            figure=px.bar(average_sales,
            x='Vehicle_Type',
            y='Automobile_Sales',
            title="Average vehicles sold by vehicle type")

# Plot 3 Pie chart for total expenditure share by vehicle type during
recessions
        # use groupby to create relevant data for plotting
        exp_rec =
recession_data.groupby('Vehicle_Type')['Advertising_Expenditure'].sum().reset_
index()
        R_chart3 = dcc.Graph(
            figure=px.pie(exp_rec,
                values='Advertising_Expenditure',
                names='Vehicle_Type',
                title="Expenditure share by vehicle type during recession"
            )
        )

# Plot 4 bar chart for the effect of unemployment rate on vehicle type and
sales
        unemployment_data =
reccession_data.groupby('Vehicle_Type')['Unemployment_Rate'].mean().reset_inde
x()
        R_chart4 = dcc.Graph(
            figure=px.bar(unemployment_data,
            x='Vehicle_Type',
            y='Unemployment_Rate',
            title="Effect of unemployment rate on vehicle type and sales")
        )


        return [
            html.Div(className='chart-item',
children=[html.Div(children=R_chart1),html.Div(children=R_chart2)]),
            html.Div(className='chart-item',
children=[html.Div(children=R_chart3),html.Div(children=R_chart4)])
            ]

# TASK 2.6: Create and display graphs for Yearly Report Statistics
 # Yearly Statistic Report Plots
    elif (input_year and selected_statistics=='Yearly Statistics') :
        yearly_data = data[data['Year'] == input_year]

#TASK 2.5: Creating Graphs Yearly data

#plot 1 Yearly Automobile sales using line chart for the whole period.
        yas= data.groupby('Year')['Automobile_Sales'].mean().reset_index()
        Y_chart1 = dcc.Graph(
            figure=px.line(yas,
            x='Year',
            y='Automobile_Sales',
            title="Yearly Automobile sales over time"))

# Plot 2 Total Monthly Automobile sales using line chart.
```

```
        monthly_sales =
yearly_data.groupby('Month')['Automobile_Sales'].sum().reset_index()
        Y_chart2 = dcc.Graph(
            figure=px.line(monthly_sales,
            x='Month',
            y='Automobile_Sales',
            title="Total Monthly Automobile sales for the Year
{}".format(input_year))
        )

            # Plot bar chart for average number of vehicles sold during the
given year
        avr_vdata=yearly_data.groupby('Vehicle_Type')['Automobile_Sales'].mean
().reset_index()
        Y_chart3 = dcc.Graph(
            figure=px.bar(avr_vdata,
            x='Vehicle_Type',
            y='Automobile_Sales',
            title='Average Vehicles Sold by Vehicle Type in the year
{}'.format(input_year)))

            # Total Advertisement Expenditure for each vehicle using pie chart
        exp_data=yearly_data.groupby('Vehicle_Type')['Advertising_Expenditure'
].sum().reset_index()
        Y_chart4 = dcc.Graph(
            figure=px.pie(exp_data,
            values='Advertising_Expenditure',
            names='Vehicle_Type',
            title="Advertisement Expenditure for each vehicle for the year
{}".format(input_year))
        )

#TASK 2.6: Returning the graphs for displaying Yearly data
        return [
                html.Div(className='chart-item',
children=[html.Div(children=Y_chart1,html.Div(children=Y_chart2)],style={'disp
lay' : 'flex'}),
                html.Div(className='chart-item',
children=[html.Div(children=Y_chart3,html.Div(children=Y_chart4)],style={'disp
lay' : 'flex'})
                ]

    else:
        return None

# Run the Dash app
if __name__ == '__main__':
    app.run_server(debug=True)
```