In [1]:

```
%pip install seaborn
%pip install folium
```

```
Requirement already satisfied: seaborn in c:\users\lenovo\appdata\local\programs\python\py
thon310\lib\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\lenovo\appdata\local\progr
ams\python\python310\lib\site-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=0.25 in c:\users\lenovo\appdata\local\programs\pyth
on\python310\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\lenovo\appdata\local\pr
ograms\python\python310\lib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\lenovo\appdata\local\programs
\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\lenovo\appdata\local\programs\pyth
on\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lenovo\appdata\local\programs
\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\lenovo\appdata\local\programs
\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\lenovo\appdata\local\programs\p
ython\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\lenovo\appdata\local\programs\pyt
hon\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\lenovo\appdata\local\programs
\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\appdata\local\progr
ams\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\appdata\local\programs\pyth
on\python310\lib\site-packages (from pandas>=0.25->seaborn) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\appdata\local\programs\python\p
ython310\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn)
(1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: folium in c:\users\lenovo\appdata\local\programs\python\pyt
hon310\lib\site-packages (0.14.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\lenovo\appdata\local\programs\pyt
hon\python310\lib\site-packages (from folium) (0.6.0)
Requirement already satisfied: jinja2>=2.9 in c:\users\lenovo\appdata\local\programs\pytho
n\python310\lib\site-packages (from folium) (3.0.3)
Requirement already satisfied: numpy in c:\users\lenovo\appdata\local\programs\python\pyth
on310\lib\site-packages (from folium) (1.23.5)
Requirement already satisfied: requests in c:\users\lenovo\appdata\local\programs\python\p
ython310\lib\site-packages (from folium) (2.26.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\lenovo\appdata\local\programs\p
ython\python310\lib\site-packages (from jinja2>=2.9->folium) (2.1.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\lenovo\appdata\local\prog
rams\python\python310\lib\site-packages (from requests->folium) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lenovo\appdata\local\program
s\python\python310\lib\site-packages (from requests->folium) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\lenovo\appdata\local
\programs\python\python310\lib\site-packages (from requests->folium) (2.0.7)
Requirement already satisfied: idna<4,>=2.5 in c:\users\lenovo\appdata\local\programs\pyth
on\python310\lib\site-packages (from requests->folium) (3.3)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:

```
import numpy as np
import pandas as pd
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import folium
```

In [3]:

```
import pandas as pd
df = pd.read_csv('D:\data visualisation in coursera\historical_automobile_sales.csv')
print('Data downloaded and read into a dataframe!')
```

```
Data downloaded and read into a dataframe!
```

In [4]:

```
df.head(10)
```

Out[4]:

	Date	Year	Month	Recession	Consumer_Confidence	Seasonality_Weight	Price	Advertising_Expendi
0	1/31/1980	1980	Jan	1	108.24	0.50	27483.571	1
1	2/29/1980	1980	Feb	1	98.75	0.75	24308.678	3
2	3/31/1980	1980	Mar	1	107.48	0.20	28238.443	3
3	4/30/1980	1980	Apr	1	115.01	1.00	32615.149	1
4	5/31/1980	1980	May	1	98.72	0.20	23829.233	1
5	6/30/1980	1980	Jun	1	105.55	0.75	23829.315	2
6	7/31/1980	1980	Jul	1	82.45	0.50	32896.064	2
7	8/31/1980	1980	Aug	1	98.76	0.25	28837.174	2
8	9/30/1980	1980	Sep	1	87.68	0.07	22652.628	1
9	10/31/1980	1980	Oct	1	101.45	0.00	27712.800	4

In [5]:

```
df.describe()
```

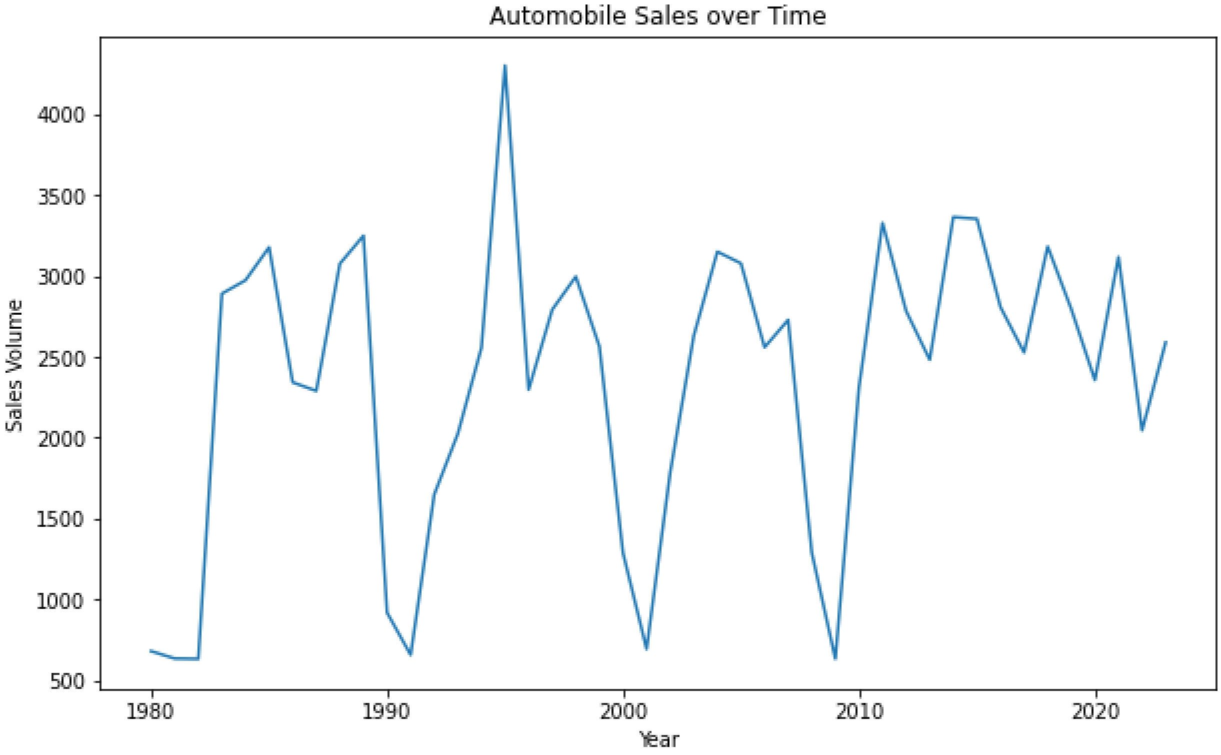
Out[5]:

	Year	Recession	Consumer_Confidence	Seasonality_Weight	Price	Advertising_Expenditure
count	528.000000	528.000000	528.000000	528.000000	528.000000	528.000000
mean	2001.500000	0.214015	101.140170	0.575795	24964.991956	3067.456439
std	12.710467	0.410526	10.601154	0.454477	4888.073433	1139.564637
min	1980.000000	0.000000	73.900000	0.000000	8793.663000	1009.000000
25%	1990.750000	0.000000	94.035000	0.250000	21453.300500	2083.500000
50%	2001.500000	0.000000	100.740000	0.500000	25038.691500	3072.000000
75%	2012.250000	0.000000	108.240000	0.750000	28131.684750	4067.250000
max	2023.000000	1.000000	131.670000	1.500000	44263.657000	4983.000000

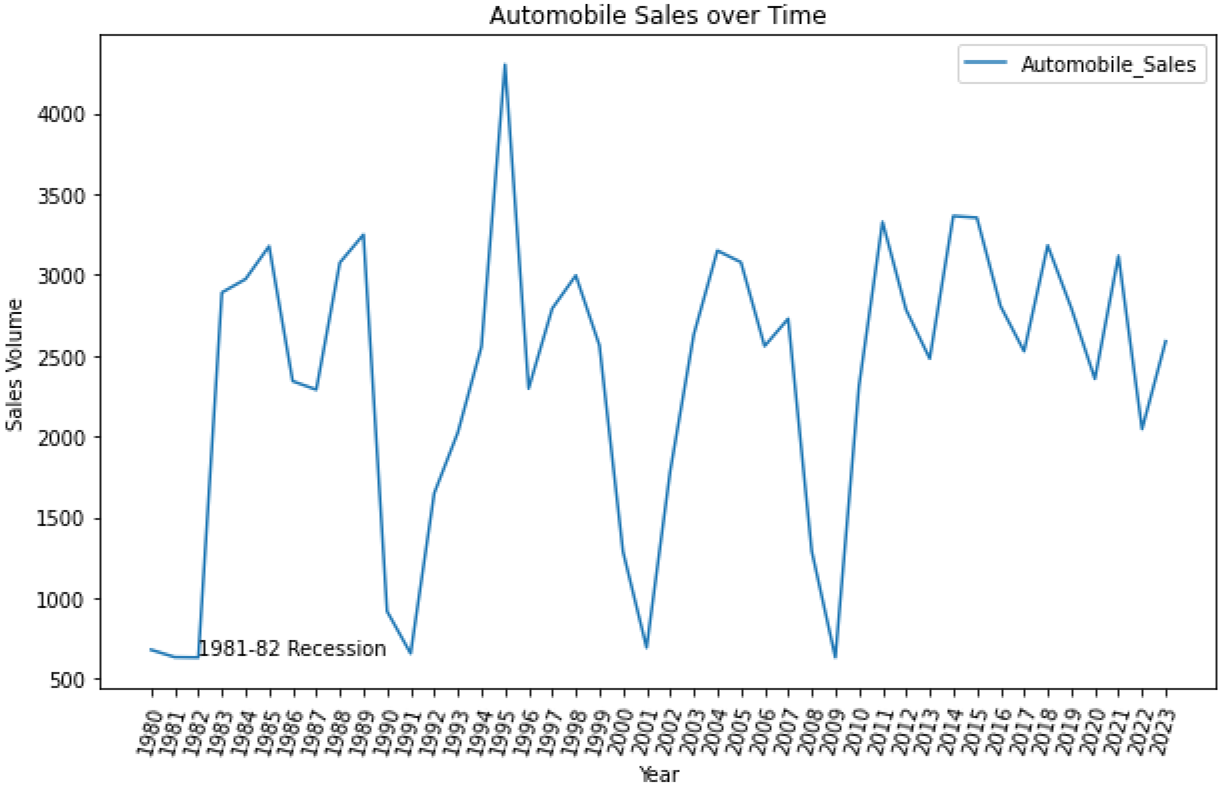
In [13]:

```
df_line = df.groupby(df['Year'])['Automobile_Sales'].mean()
#create figure
plt.figure(figsize=(10, 6))
df_line.plot(kind = 'line')
plt.xlabel('Year')
plt.ylabel('Sales Volume')
plt.title('Automobile Sales over Time')
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



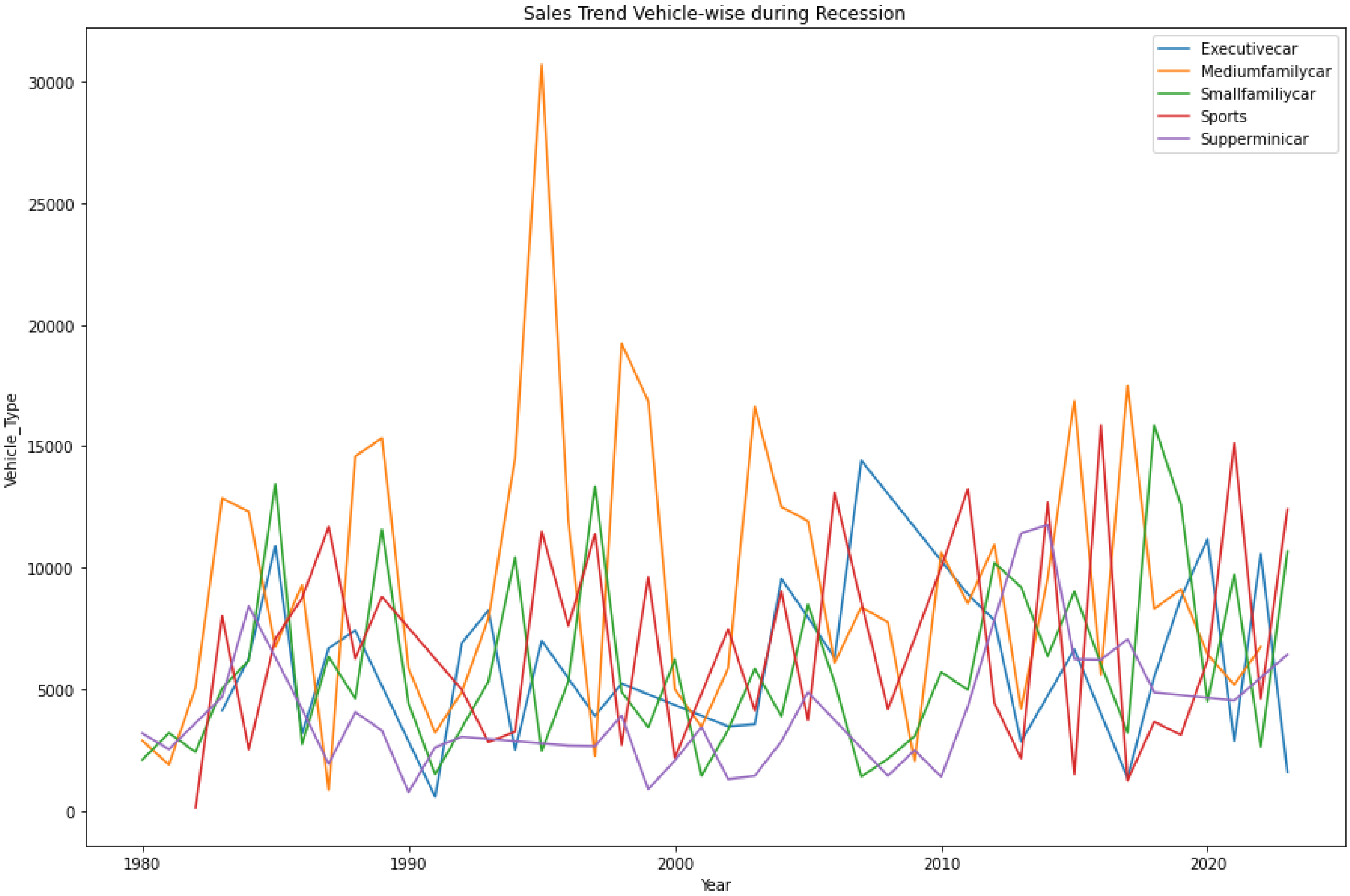
```
In [19]:
plt.figure(figsize=(10, 6))
df_line =df.groupby(df['Year'])['Automobile_Sales'].mean()
df_line.plot(kind = 'line')
plt.xticks(list(range(1980,2024)), rotation = 75)
plt.xlabel('Year')
plt.ylabel('Sales Volume')
plt.title('Automobile Sales over Time')
plt.text(1982, 650, '1981-82 Recession')
plt.legend()
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

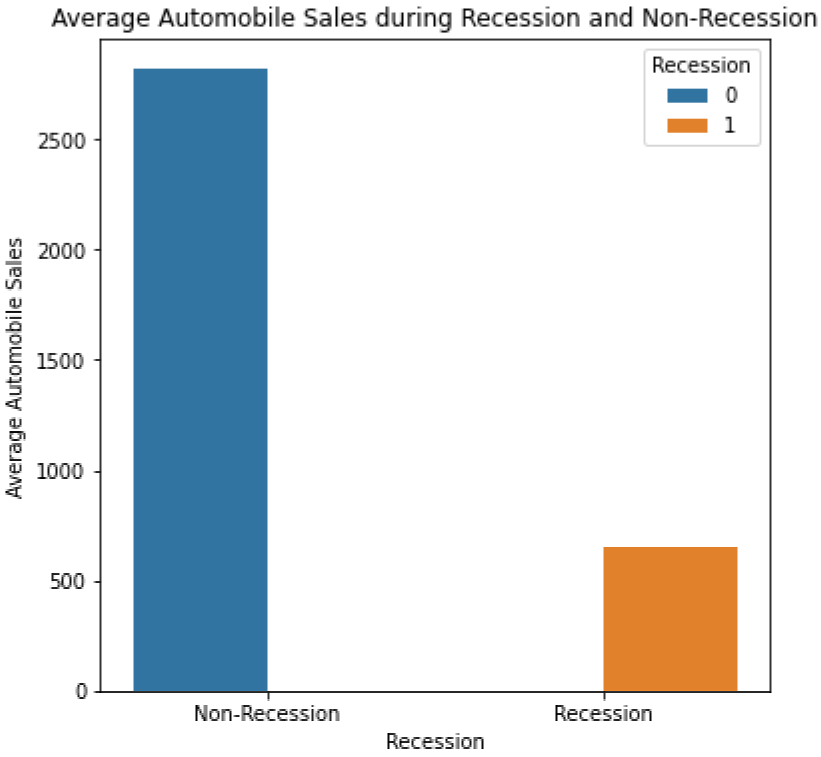
In [70]:

```
df_Mline = df.groupby(['Year','Vehicle_Type'], as_index=False)['Automobile_Sales'].sum().r
df_Mline.set_index('Year', inplace=True)
df_Mline = df_Mline.groupby(['Vehicle_Type'])['Automobile_Sales']
plt.figure(figsize=(15,10))

df_Mline.plot(kind='line')

plt.xlabel('Year')
plt.ylabel('Vehicle_Type')
plt.title('Sales Trend Vehicle-wise during Recession')
plt.legend()
plt.show()
```



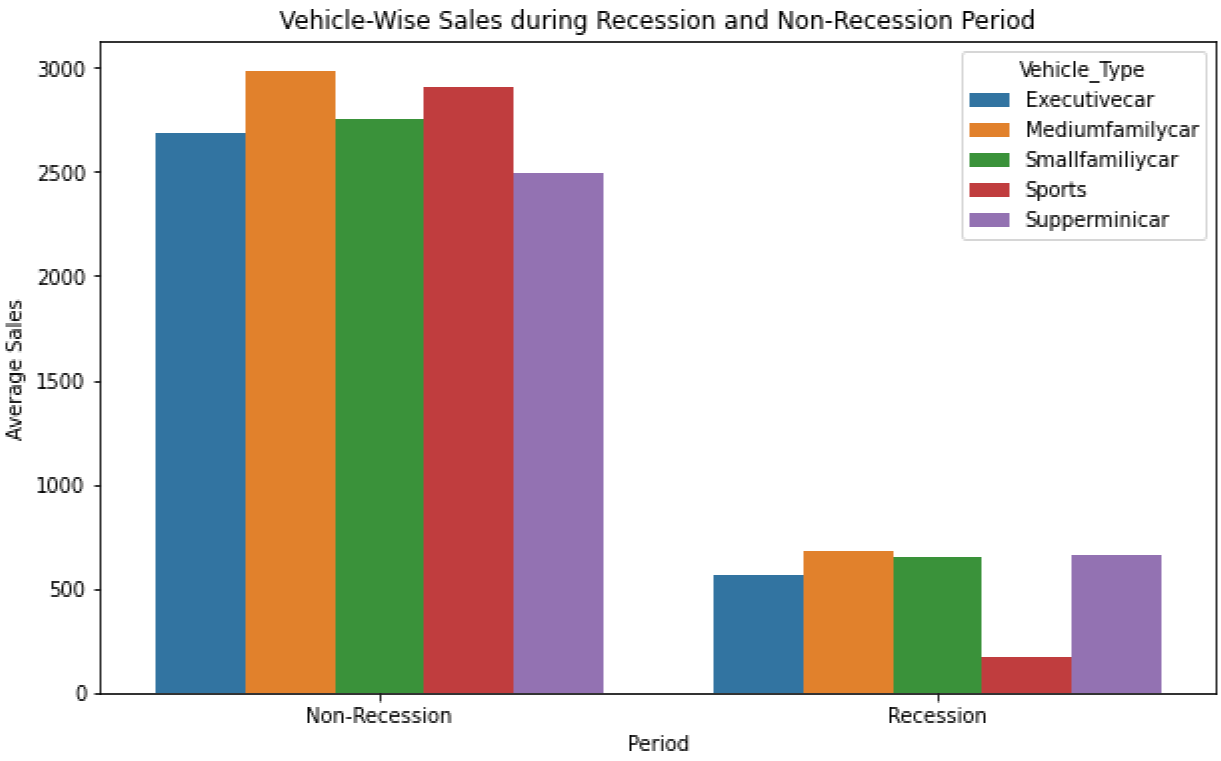
In [36]:

```
new_df = df.groupby('Recession')['Automobile_Sales'].mean().reset_index()

# Create the bar chart using seaborn
plt.figure(figsize=(6,6))
sns.barplot(x='Recession', y='Automobile_Sales', hue='Recession',  data=new_df)
plt.ylabel('Average Automobile Sales')
plt.title('Average Automobile Sales during Recession and Non-Recession'
plt.xticks(ticks=[0, 1], labels=['Non-Recession', 'Recession'])
plt.show()
```



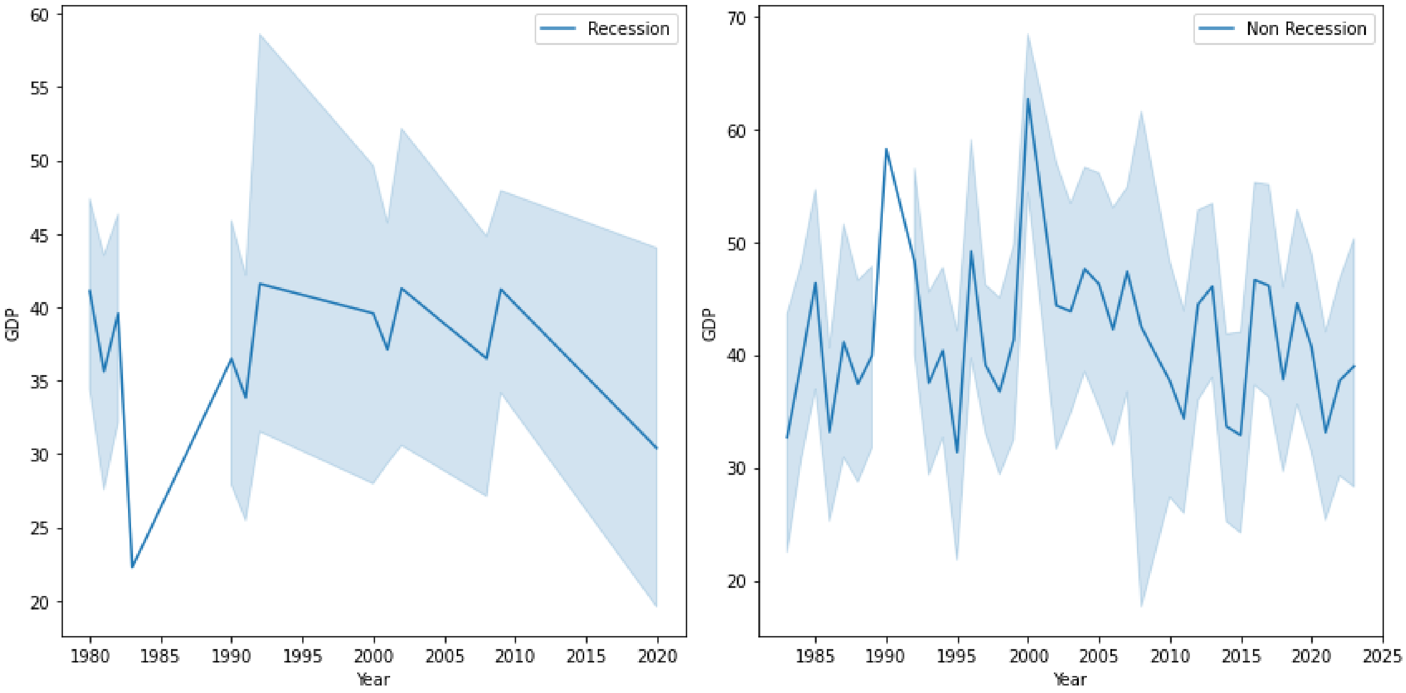
In [44]:

```
 # Filter the data for recessionary periods
recession_data = df[df['Recession'] == 1]

dd=df.groupby(['Recession','Vehicle_Type'])['Automobile_Sales'].mean().reset_index()

    # Calculate the total sales volume by vehicle type during recessions
    #sales_by_vehicle_type = recession_data.groupby('Vehicle_Type')['Automobile_Sales'].su

    # Create the grouped bar chart using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='Recession', y='Automobile_Sales', hue='Vehicle_Type', data=dd)
plt.xticks(ticks=[0, 1], labels=['Non-Recession', 'Recession'])
plt.xlabel('Period')
plt.ylabel('Average Sales')
plt.title('Vehicle-Wise Sales during Recession and Non-Recession Period')

plt.show()
```



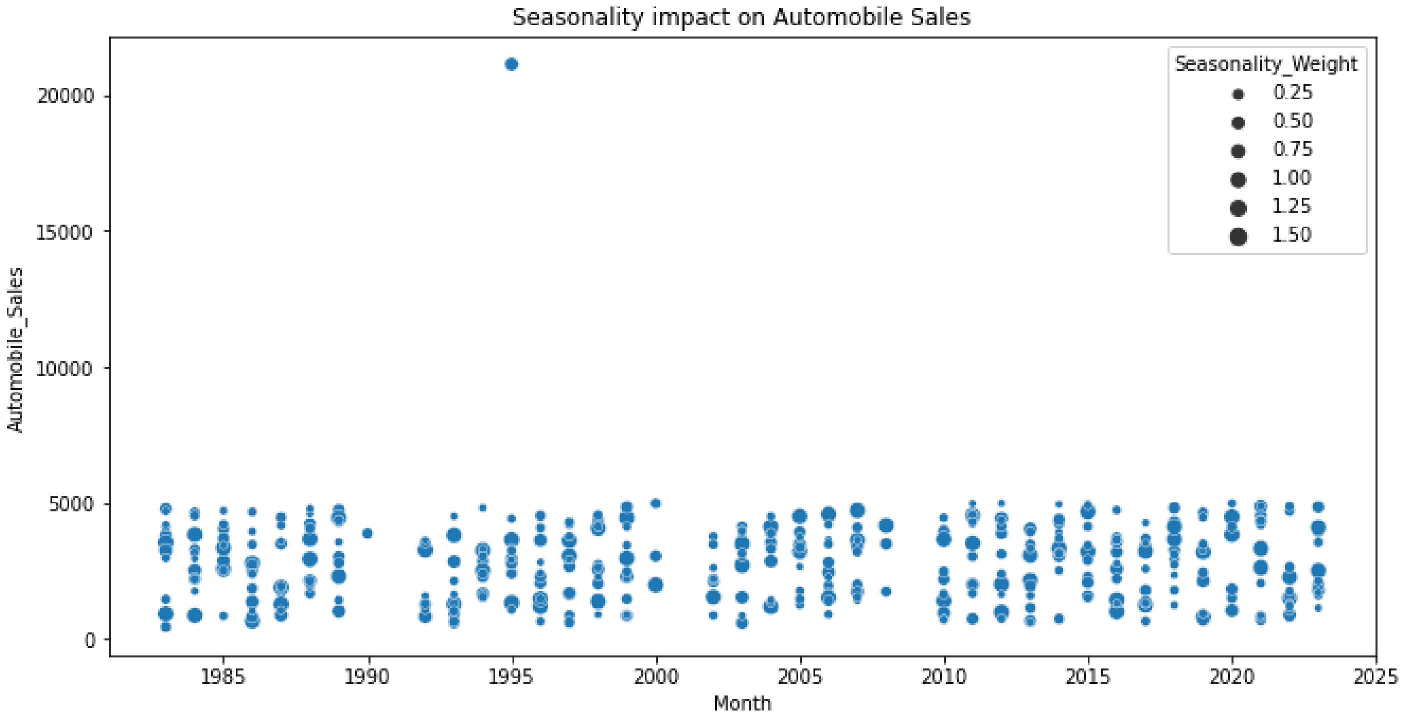
In [63]:

```
  #Create dataframes for recession and non-recession period
rec_data = df[df['Recession'] == 1]
non_rec_data = df[df['Recession'] == 0]

#Figure
fig=plt.figure(figsize=(12, 6))

plt.subplot(1,2,1)
sns.lineplot(x='Year', y='GDP', data=rec_data, label='Recession')
ax0.set_xlabel('Year')
ax0.set_ylabel('GDP')
ax0.set_title('GDP Variation during Recession Period')

plt.subplot(1,2, 2)
sns.lineplot(x='Year', y='GDP', data=non_rec_data, label='Non Recession')
ax1.set_xlabel('Year')
ax1.set_ylabel('GDP')
ax1.set_title('GDP Variation during non_Recession Period')

plt.tight_layout()
plt.show()
```



In [79]:

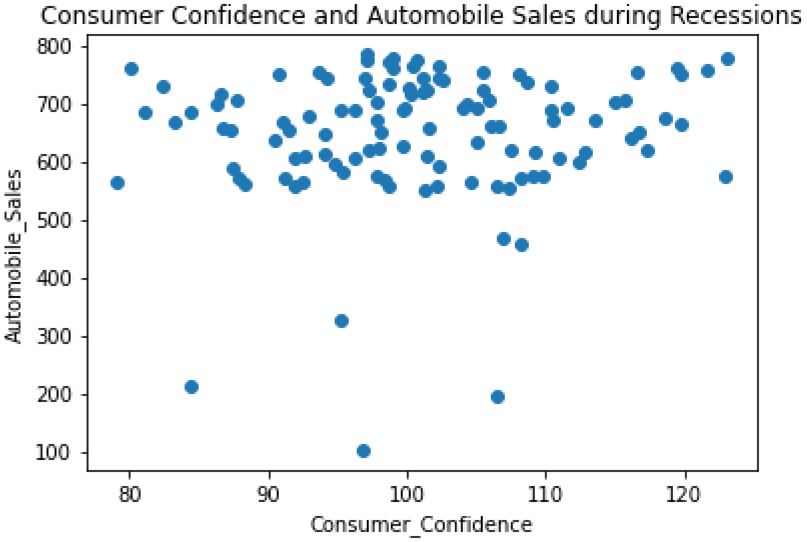
```
non_rec_data = df[df['Recession'] == 0]

size=non_rec_data['Seasonality_Weight'] #for bubble effect

plt.figure(figsize=(12,6))

sns.scatterplot(data=non_rec_data, x='Year', y='Automobile_Sales', size=size)

    #you can further include hue='Seasonality_Weight', legend=False)

plt.xlabel('Month')
plt.ylabel('Automobile_Sales')
plt.title('Seasonality impact on Automobile Sales')

plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

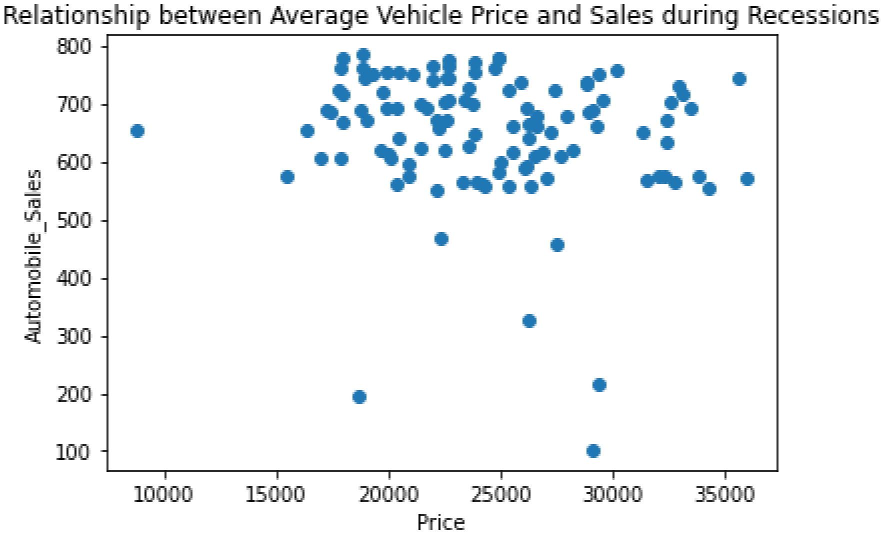
In [88]:

```
#Create dataframes for recession and non-recession period
rec_data = df[df['Recession'] == 1]
plt.scatter(x=recession_data['Consumer_Confidence'], y=recession_data['Automobile_Sales']

plt.xlabel('Consumer_Confidence')
plt.ylabel('Automobile_Sales')
plt.title('Consumer Confidence and Automobile Sales during Recessions')
plt.show()
```



In [94]:

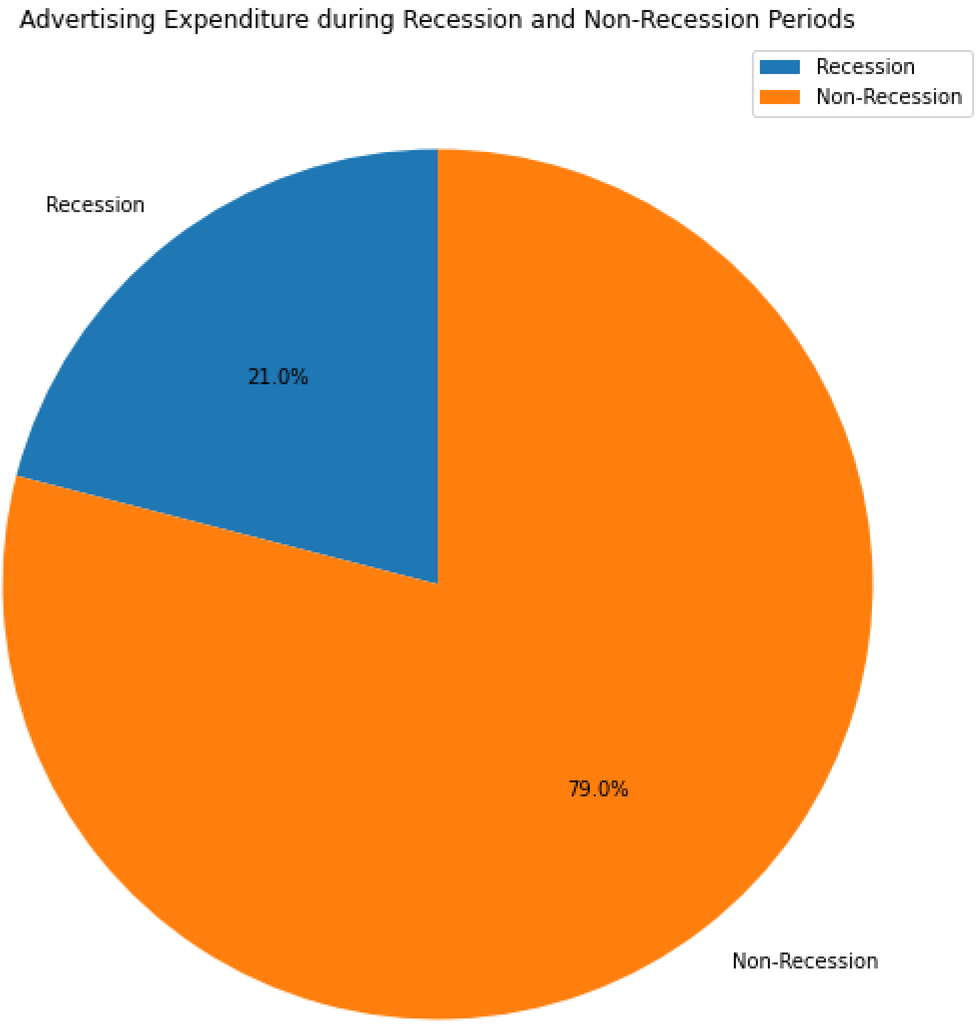
```
plt.scatter(x=recession_data['Price'], y=rec_data['Automobile_Sales'])

plt.xlabel('Price')
plt.ylabel('Automobile_Sales')
plt.title('Relationship between Average Vehicle Price and Sales during Recessions')
plt.show()
```



In [114…

```
# Filter the data

# Calculate the total advertising expenditure for both periods
RAtotal = rec_data['Advertising_Expenditure'].sum()
NRAtotal = non_rec_data['Advertising_Expenditure'].sum()

# Create a pie chart for the advertising expenditure
plt.figure(figsize=(10,10))
```

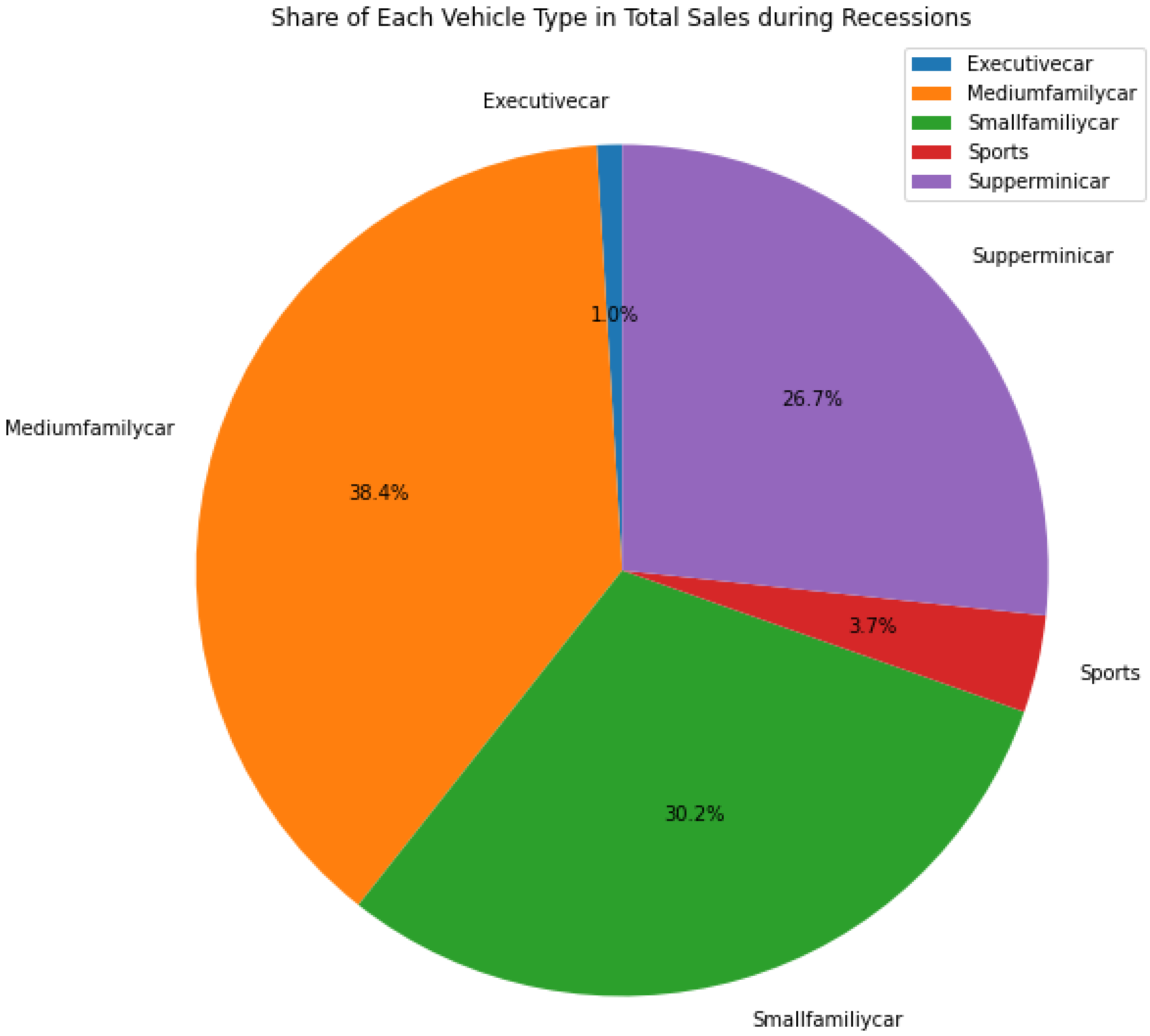
```
labels = ['Recession', 'Non-Recession']
sizes = [RAtotal, NRAtotal]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.legend()

plt.title('Advertising Expenditure during Recession and Non-Recession Periods')

plt.show()
```



In [116…

```
VTsales = rec_data.groupby('Vehicle_Type')['Advertising_Expenditure'].sum()

 # Create a pie chart for the share of each vehicle type in total sales during recessions
plt.figure(figsize=(10,10))

labels = VTsales.index
sizes = (VTsales.values)
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.legend()

plt.title('Share of Each Vehicle Type in Total Sales during Recessions')

plt.show()
```

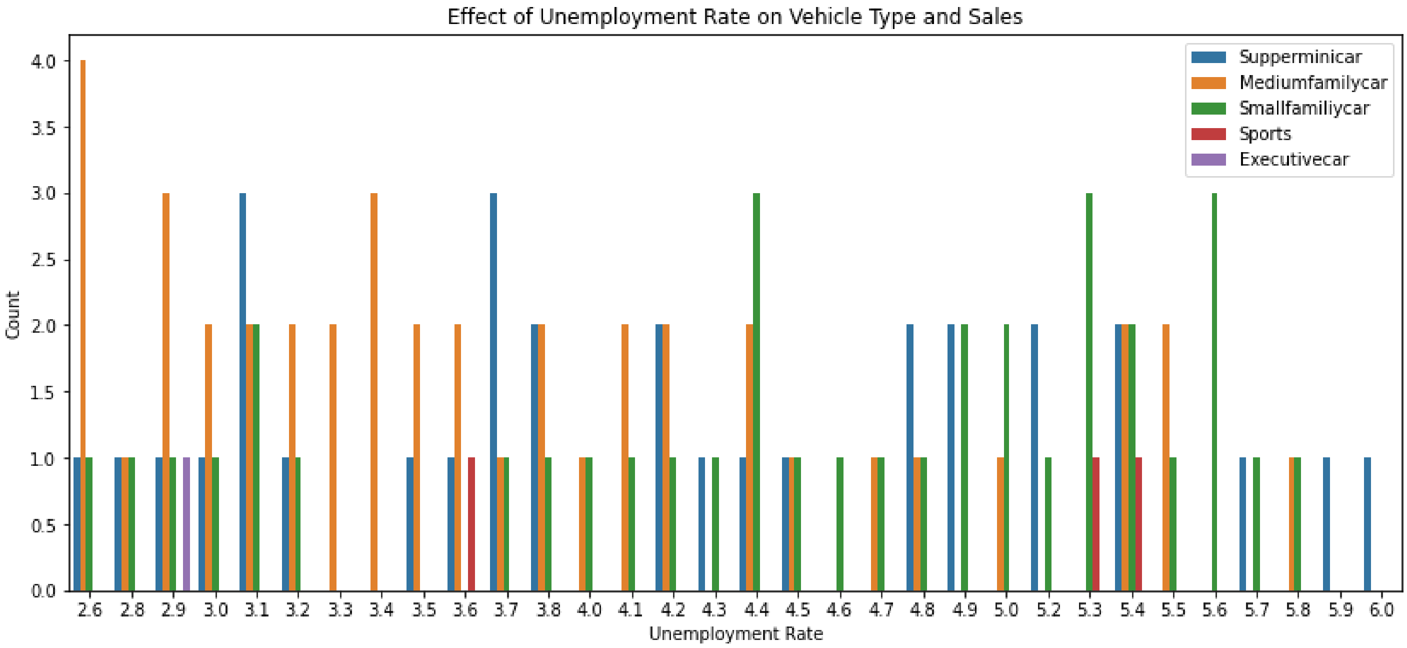
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [123…

```
plt.figure(figsize=(14, 6))

sns.countplot(data=rec_data, x='unemployment_rate', hue='Vehicle_Type')

plt.xlabel('Unemployment Rate')
plt.ylabel('Count')
plt.title('Effect of Unemployment Rate on Vehicle Type and Sales')
plt.legend(loc='upper right')
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [126…

```
 import requests

def download(url, filename):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            with open(filename, "wb") as f:
                f.write(response.content)
                print(f"Downloaded {filename} successfully.")
        else:
            print(f"Failed to download {filename}. Status code: {response.status_code}")
    except Exception as e:
        print(f"An error occurred: {e}")

path = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSk
download(path, "us-states.json")

filename = "us-states.json"
```

```
Downloaded us-states.json successfully.
```

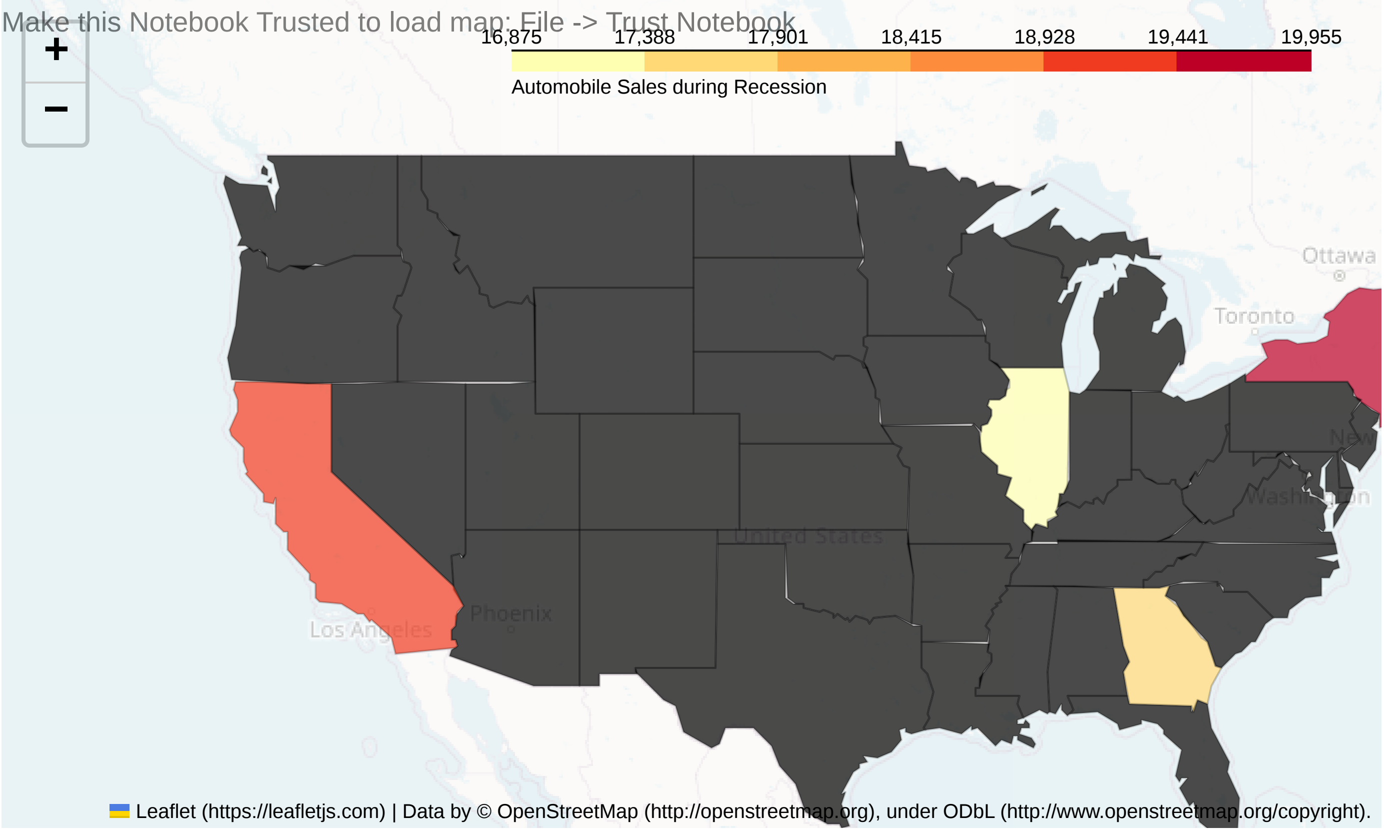
In [147…

```
# Filter the data for the recession period and specific cities
recession_data = df[df['Recession'] == 1]

    # Calculate the total sales by city
sales_by_city = recession_data.groupby('City')['Automobile_Sales'].sum().reset_index()

    # Create a base map centered on the United States
map1 = folium.Map(location=[37.0902, -95.7129], zoom_start=4)

    # Create a choropleth layer using Folium
choropleth = folium.Choropleth(
        geo_data= 'us-states.json',  # GeoJSON file with state boundaries
        data=sales_by_city,
        columns=['City', 'Automobile_Sales'],
        key_on='feature.properties.name',
        fill_color='YlOrRd',
        fill_opacity=0.7,
        line_opacity=0.2,
        legend_name='Automobile Sales during Recession').add_to(map1)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
    # Add tooltips to the choropleth layer
choropleth.geojson.add_child(folium.features.GeoJsonTooltip(['name'], labels=True))

    # Display the map
map1
```

Out[147…


Make this Notebook Trusted to load map: File -> Trust Notebook
+
−
16,875
17,388
17,901
18,415
18,928
19,441
19,955
Automobile Sales during Recession
Ottawa
Toronto
New
Washington
United States
Phoenix
Los Angeles
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [ ]:

```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js