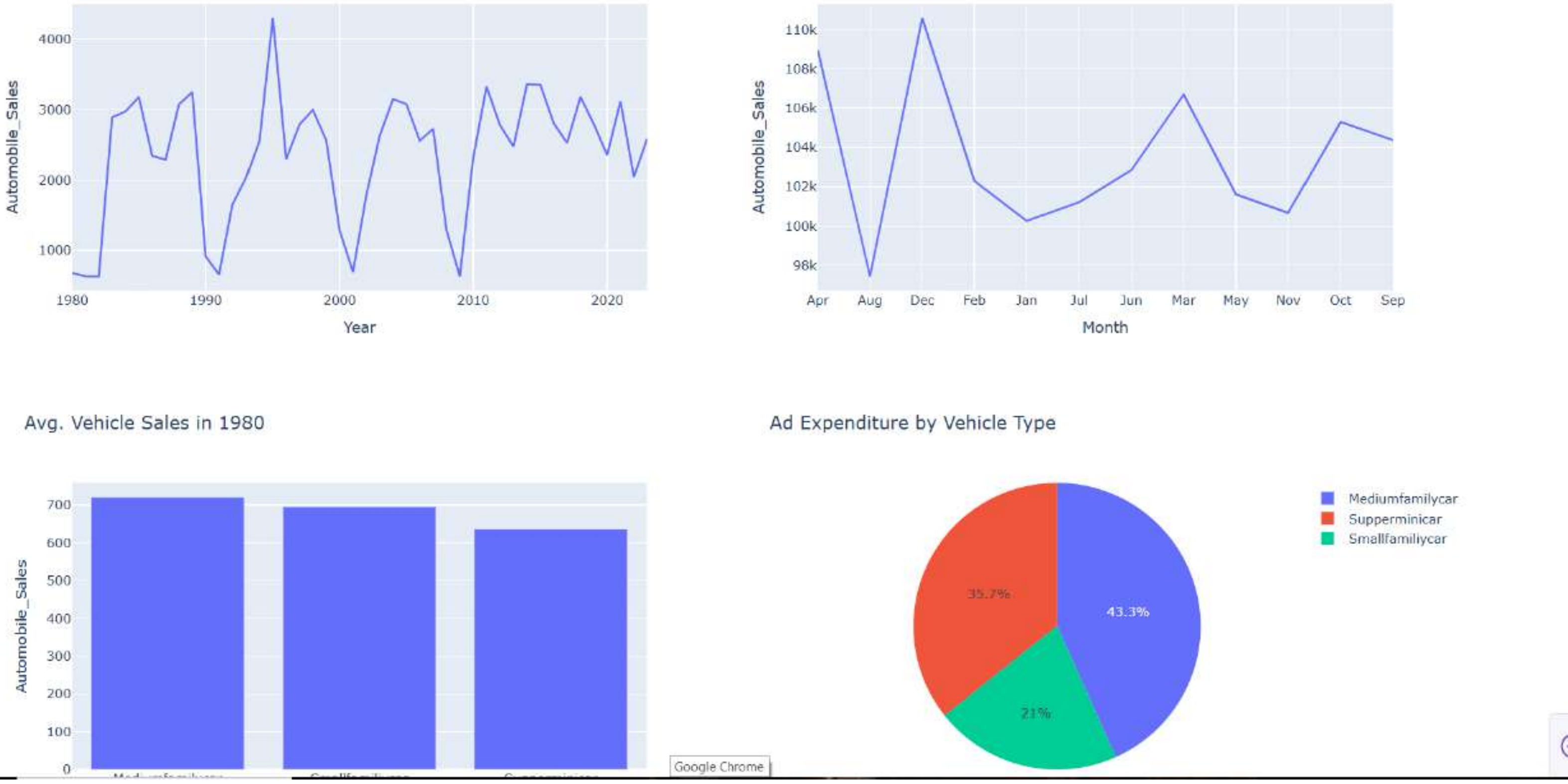
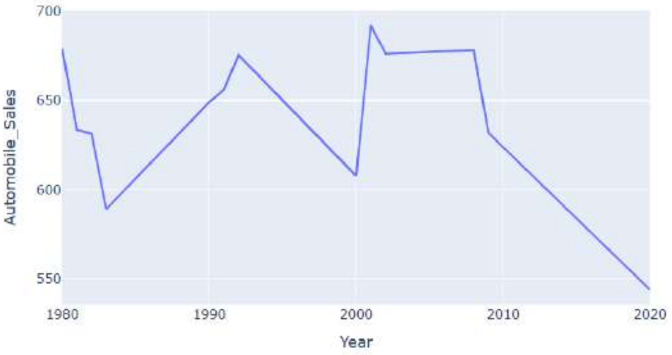
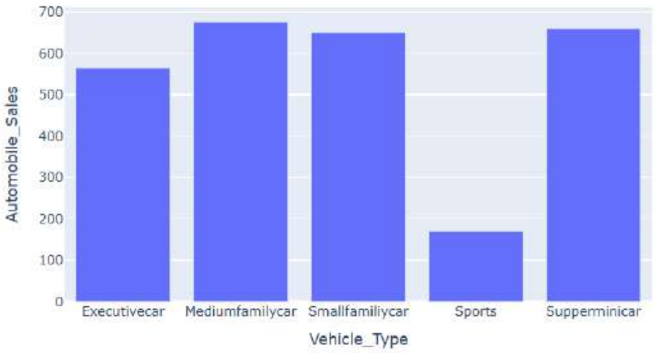


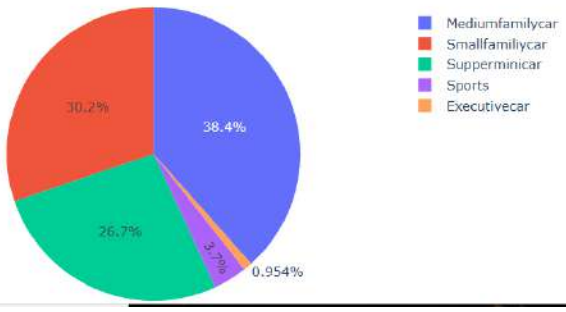
Avg. Vehicle Sales in 1980

Ad Expenditure by Vehicle Type

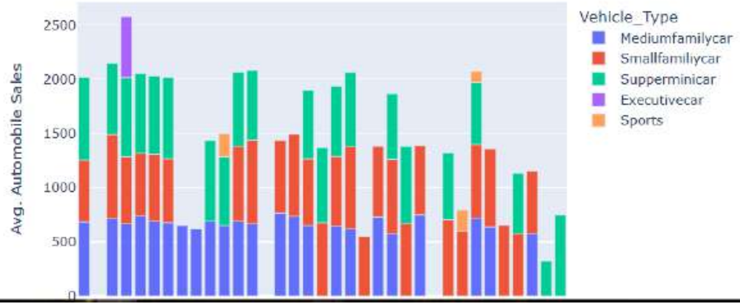


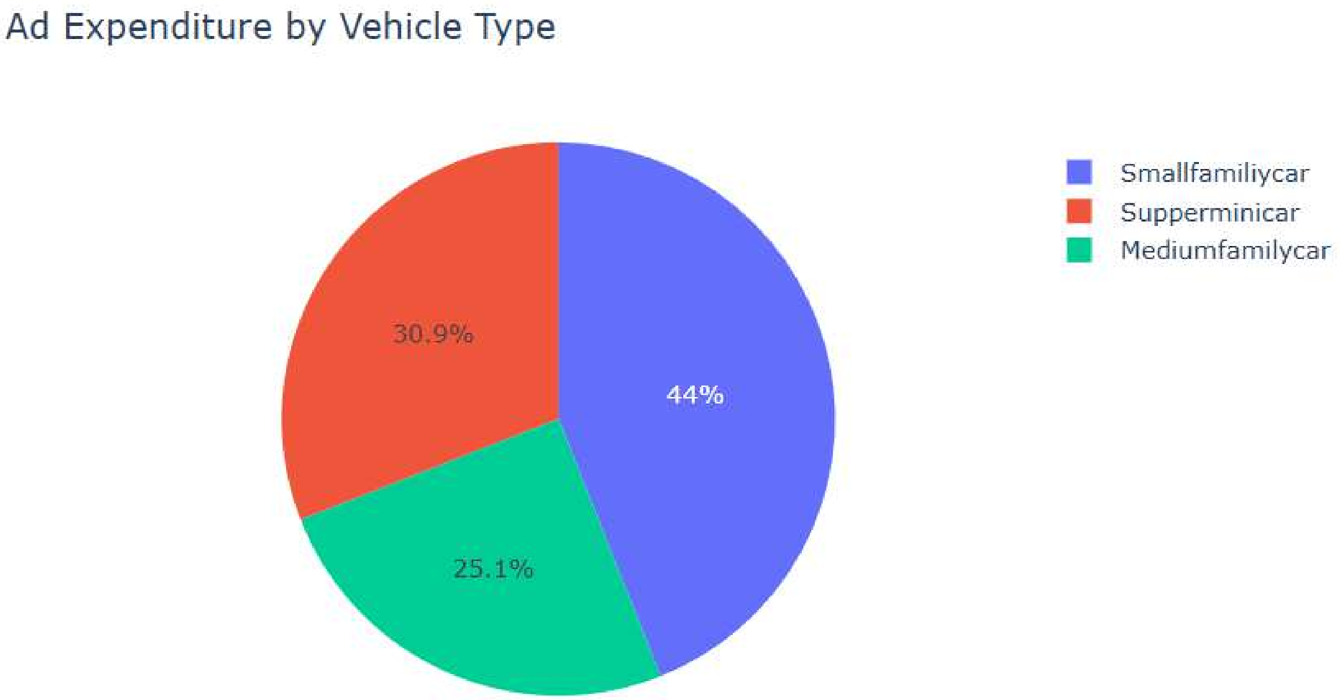


Ad Expenditure by Vehicle Type

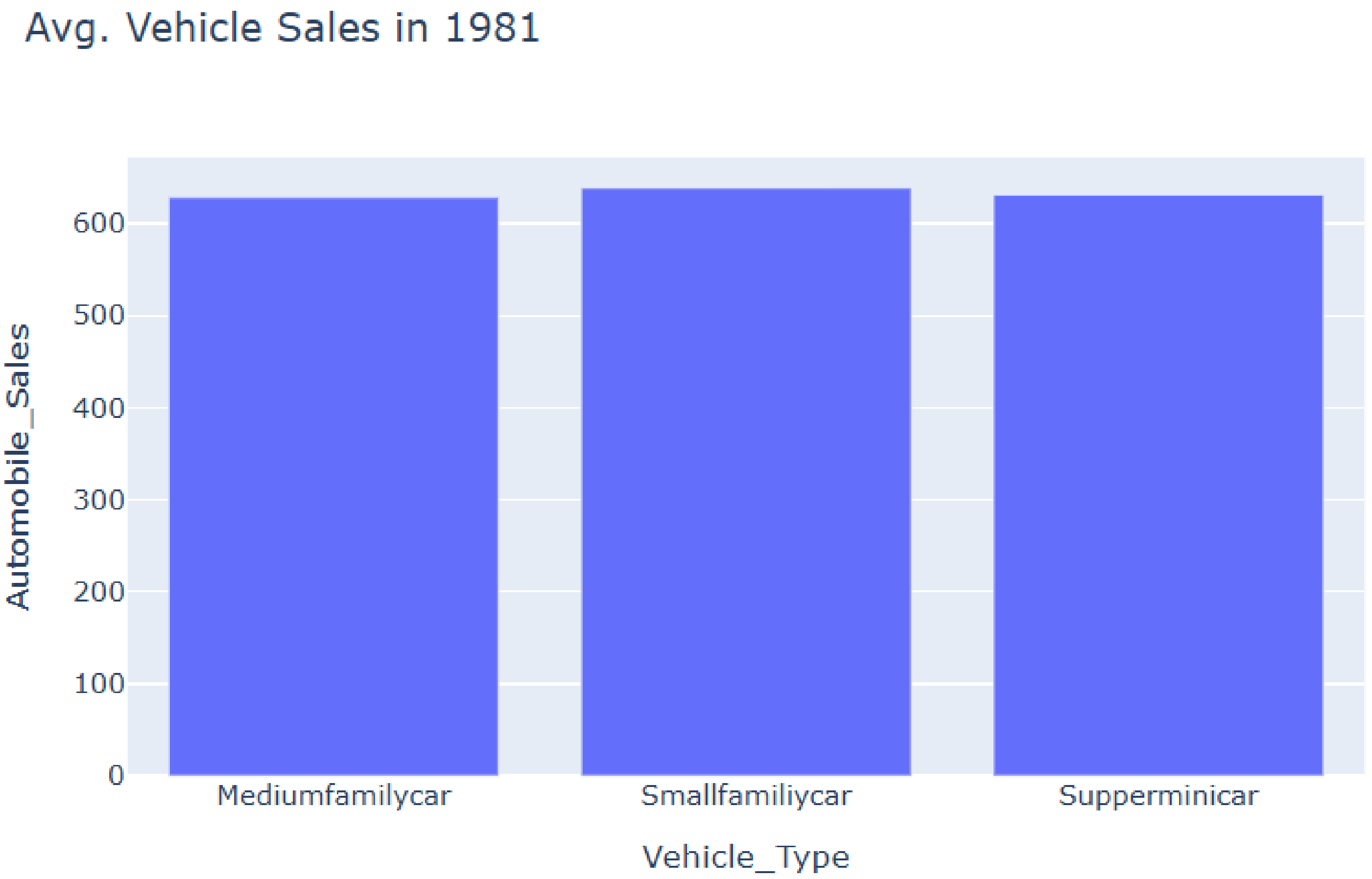


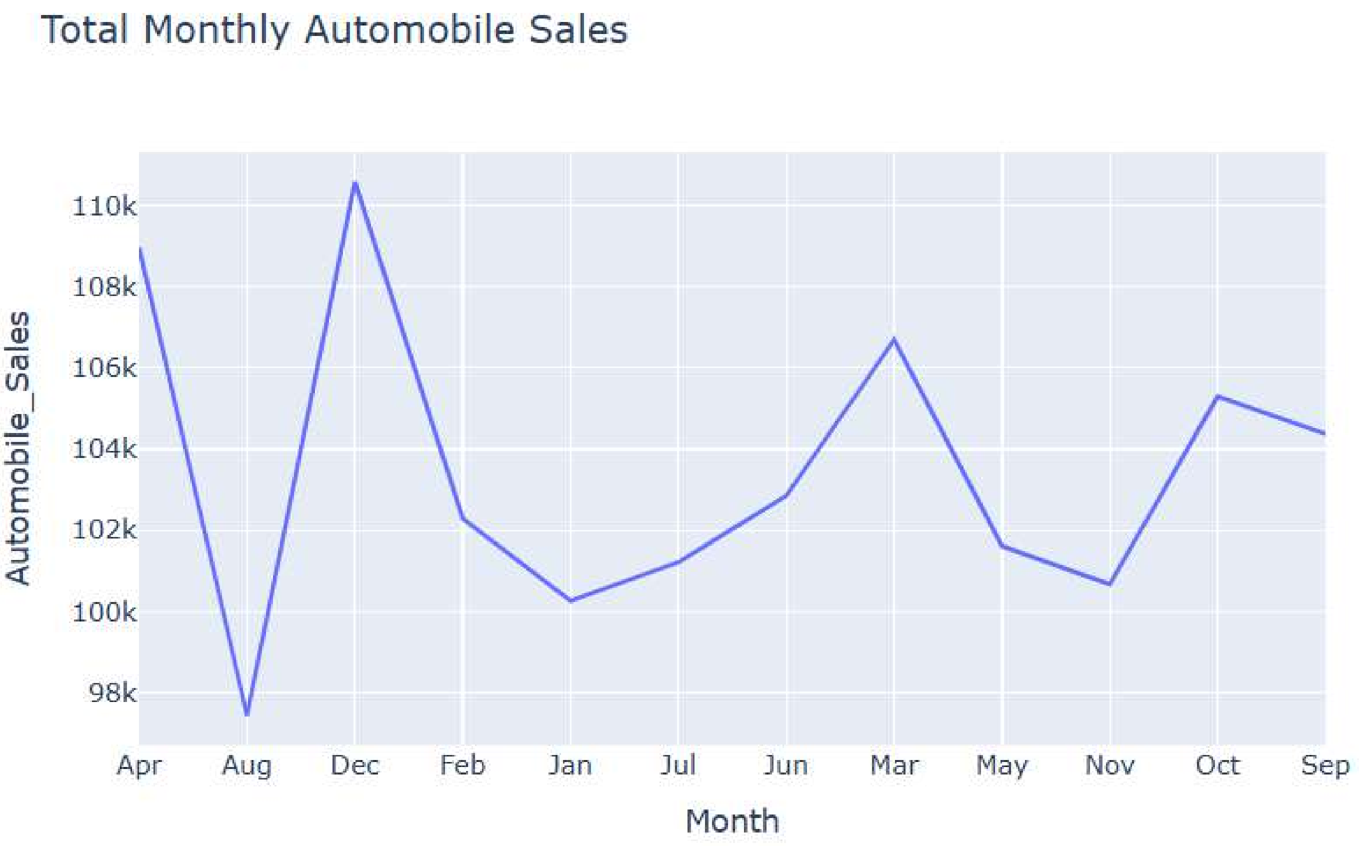
Effect of Unemployment Rate on Vehicle Type and Sales

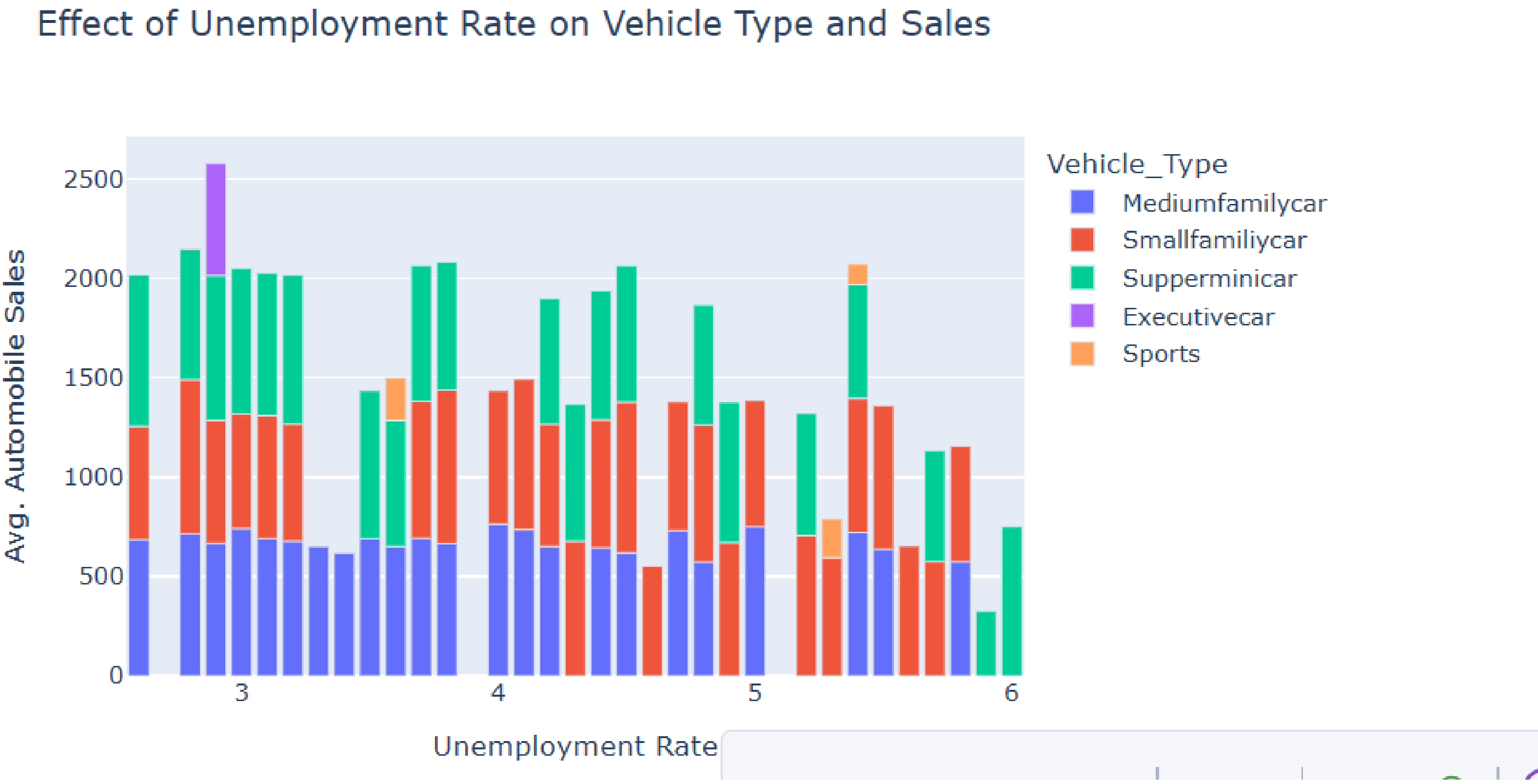




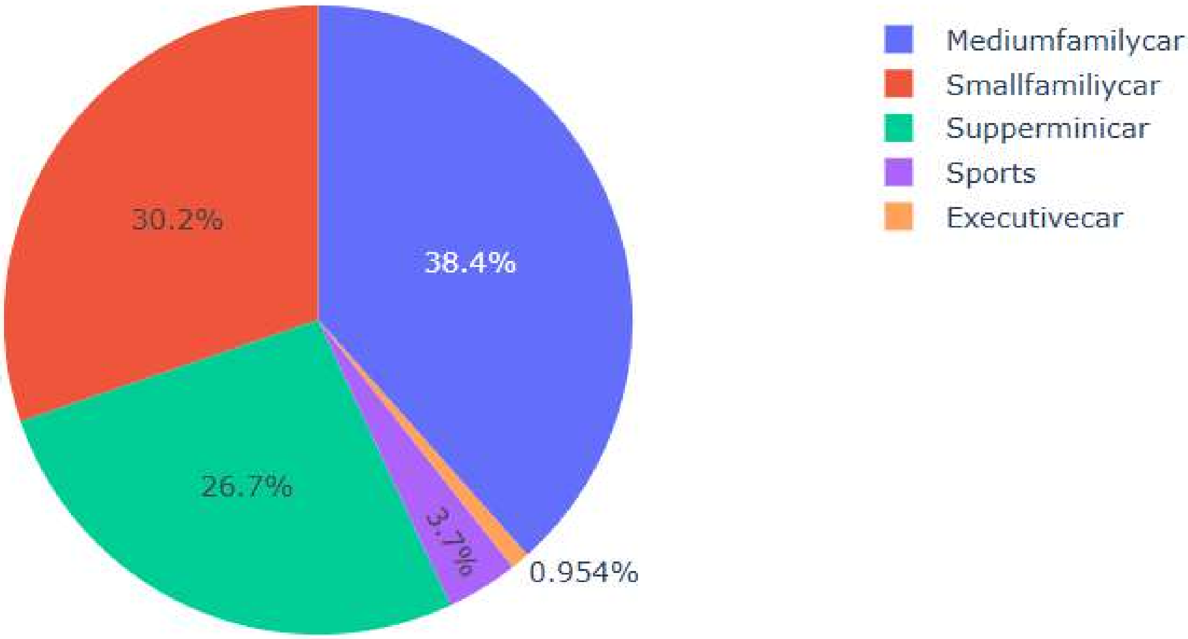
Ad Expenditure by Vehicle Type

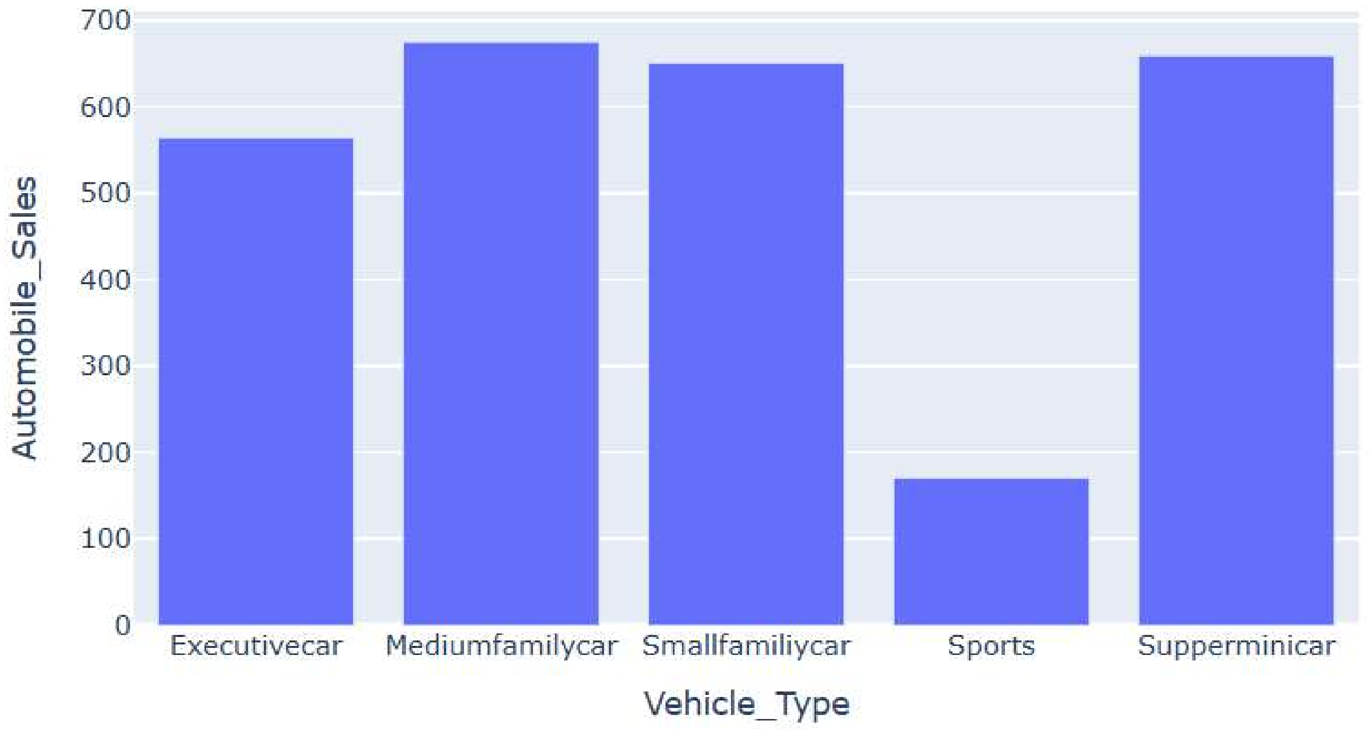

Avg. Vehicle Sales in 1981
Automobile_Sales
600
500
400
300
200
100
0
Mediumfamilycar
Smallfamiliycar
Supperminicar
Vehicle_Type


Total Monthly Automobile Sales
Automobile_Sales
110k
108k
106k
104k
102k
100k
98k
Apr
Aug
Dec
Feb
Jan
Jul
Jun
Mar
May
Nov
Oct
Sep
Month

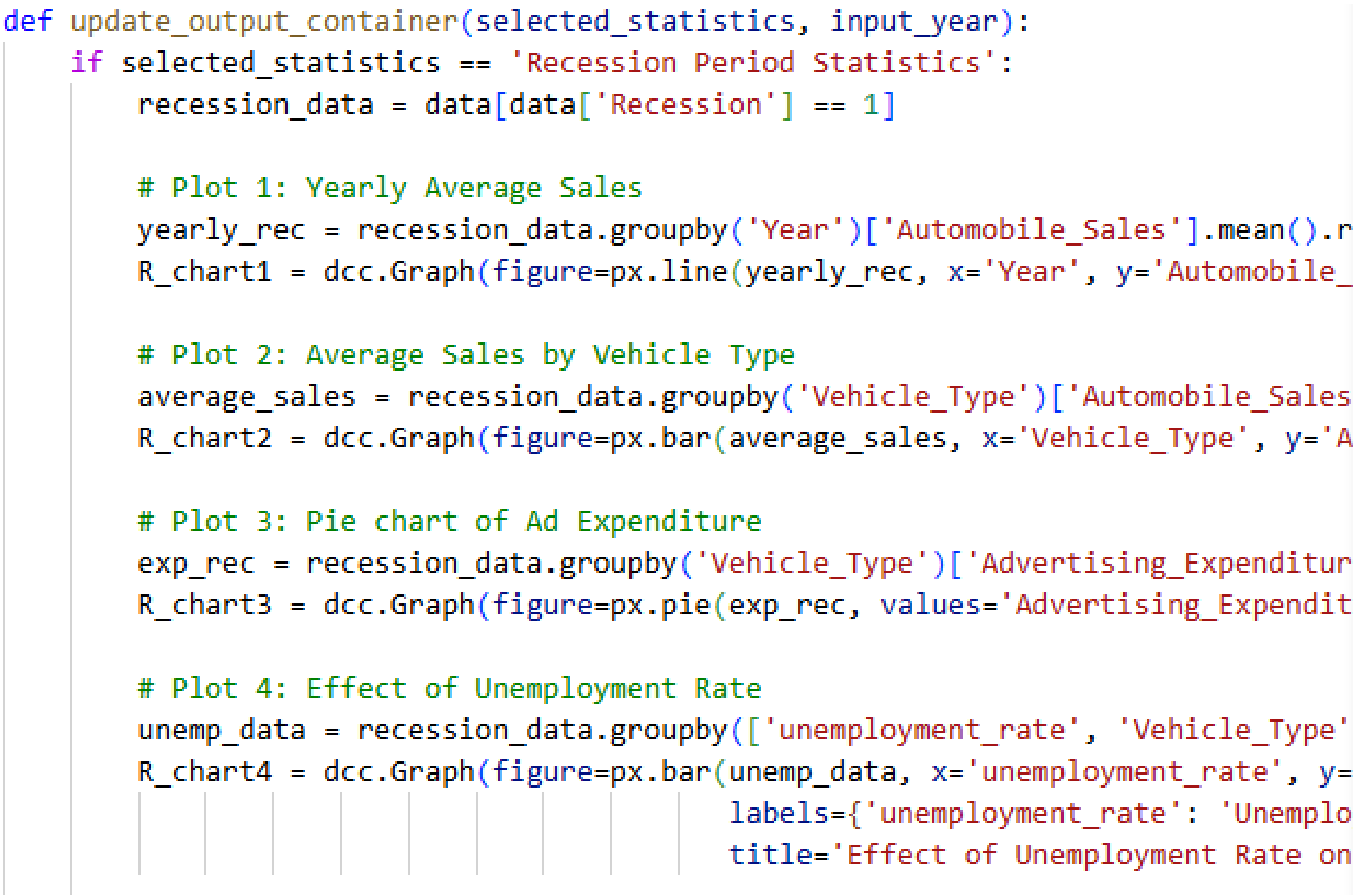

Effect of Unemployment Rate on Vehicle Type and Sales
Avg. Automobile Sales
0
500
1000
1500
2000
2500
3
4
5
6
Unemployment Rate
Vehicle_Type
Mediumfamilycar
Smallfamiliycar
Supperminicar
Executivecar
Sports

Ad Expenditure by Vehicle Type

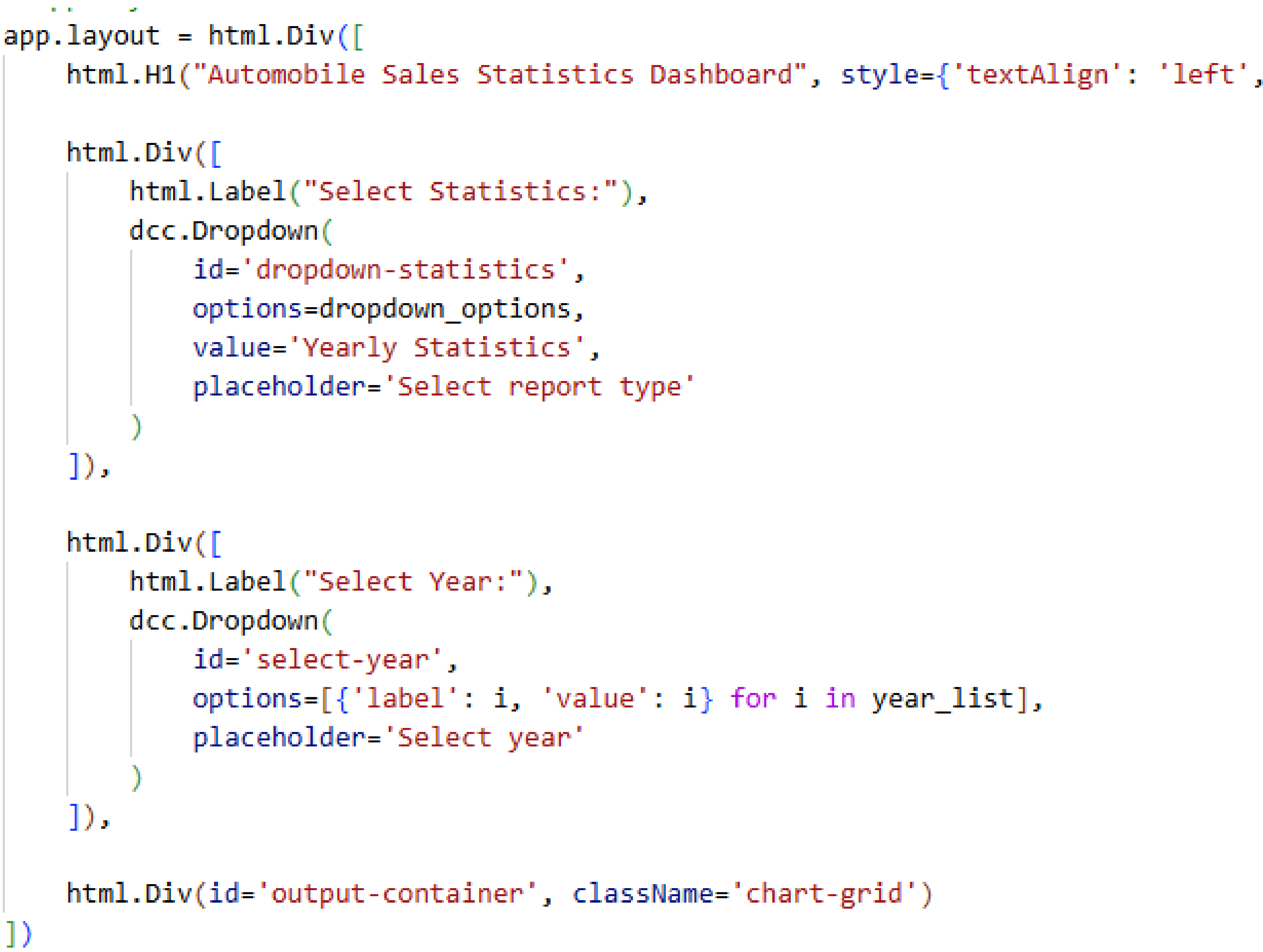




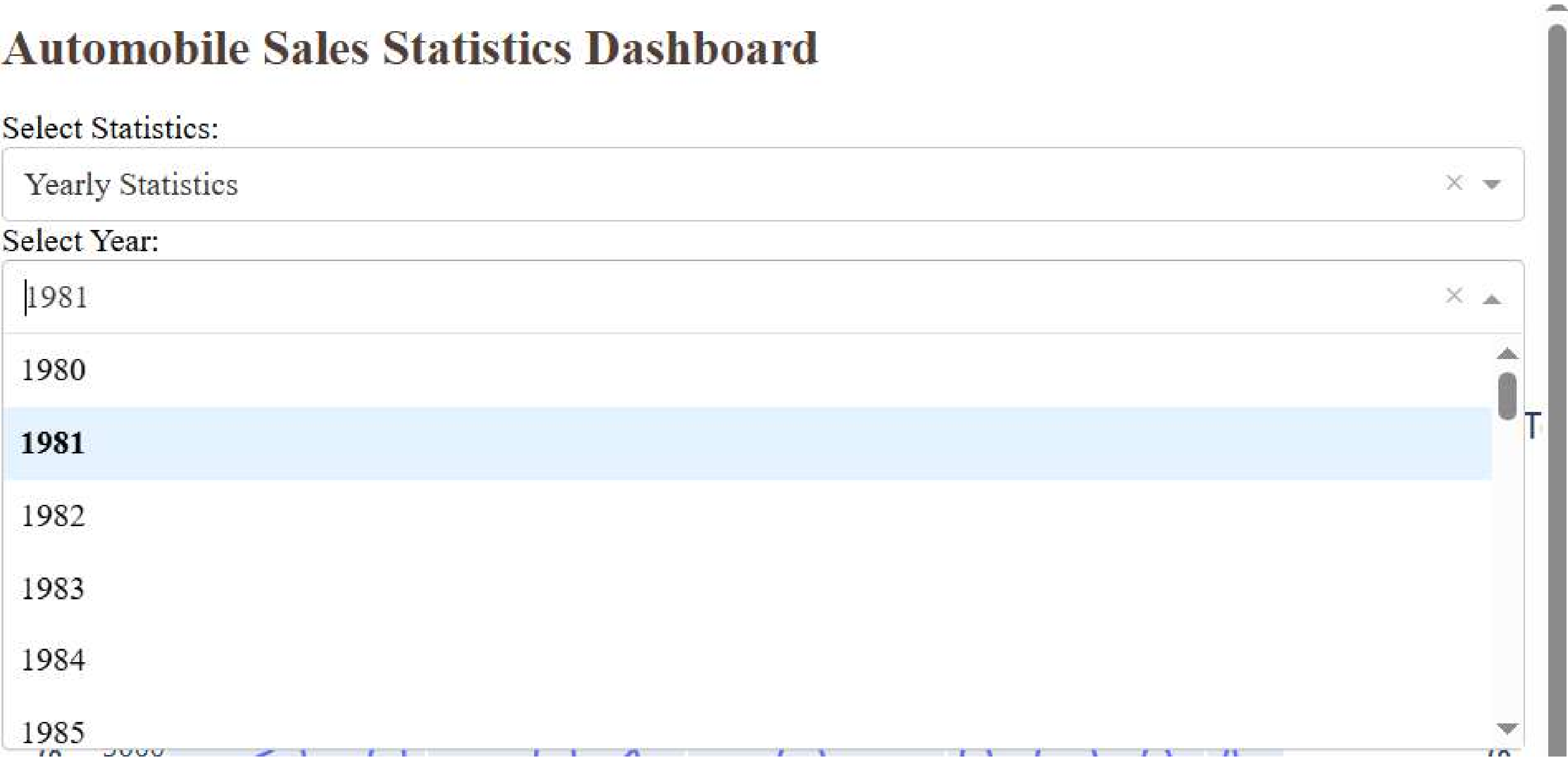
Avg. Vehicle Sales by Type During Recession



```
def update_output_container(selected_statistics, input_year):
    if selected_statistics == 'Recession Period Statistics':
        recession_data = data[data['Recession'] == 1]

        # Plot 1: Yearly Average Sales
        yearly_rec = recession_data.groupby('Year')['Automobile_Sales'].mean().r
        R_chart1 = dcc.Graph(figure=px.line(yearly_rec, x='Year', y='Automobile_

        # Plot 2: Average Sales by Vehicle Type
        average_sales = recession_data.groupby('Vehicle_Type')['Automobile_Sales
        R_chart2 = dcc.Graph(figure=px.bar(average_sales, x='Vehicle_Type', y='A

        # Plot 3: Pie chart of Ad Expenditure
        exp_rec = recession_data.groupby('Vehicle_Type')['Advertising_Expenditur
        R_chart3 = dcc.Graph(figure=px.pie(exp_rec, values='Advertising_Expendit

        # Plot 4: Effect of Unemployment Rate
        unemp_data = recession_data.groupby(['unemployment_rate', 'Vehicle_Type'
        R_chart4 = dcc.Graph(figure=px.bar(unemp_data, x='unemployment_rate', y=
                                           labels={'unemployment_rate': 'Unemplo
                                           title='Effect of Unemployment Rate on
```



```
app.layout = html.Div([
    html.H1("Automobile Sales Statistics Dashboard", style={'textAlign': 'left',

    html.Div([
        html.Label("Select Statistics:"),
        dcc.Dropdown(
            id='dropdown-statistics',
            options=dropdown_options,
            value='Yearly Statistics',
            placeholder='Select report type'
        )
    ]),

    html.Div([
        html.Label("Select Year:"),
        dcc.Dropdown(
            id='select-year',
            options=[{'label': i, 'value': i} for i in year_list],
            placeholder='Select year'
        )
    ]),

    html.Div(id='output-container', className='chart-grid')
])
```



Automobile Sales Statistics Dashboard

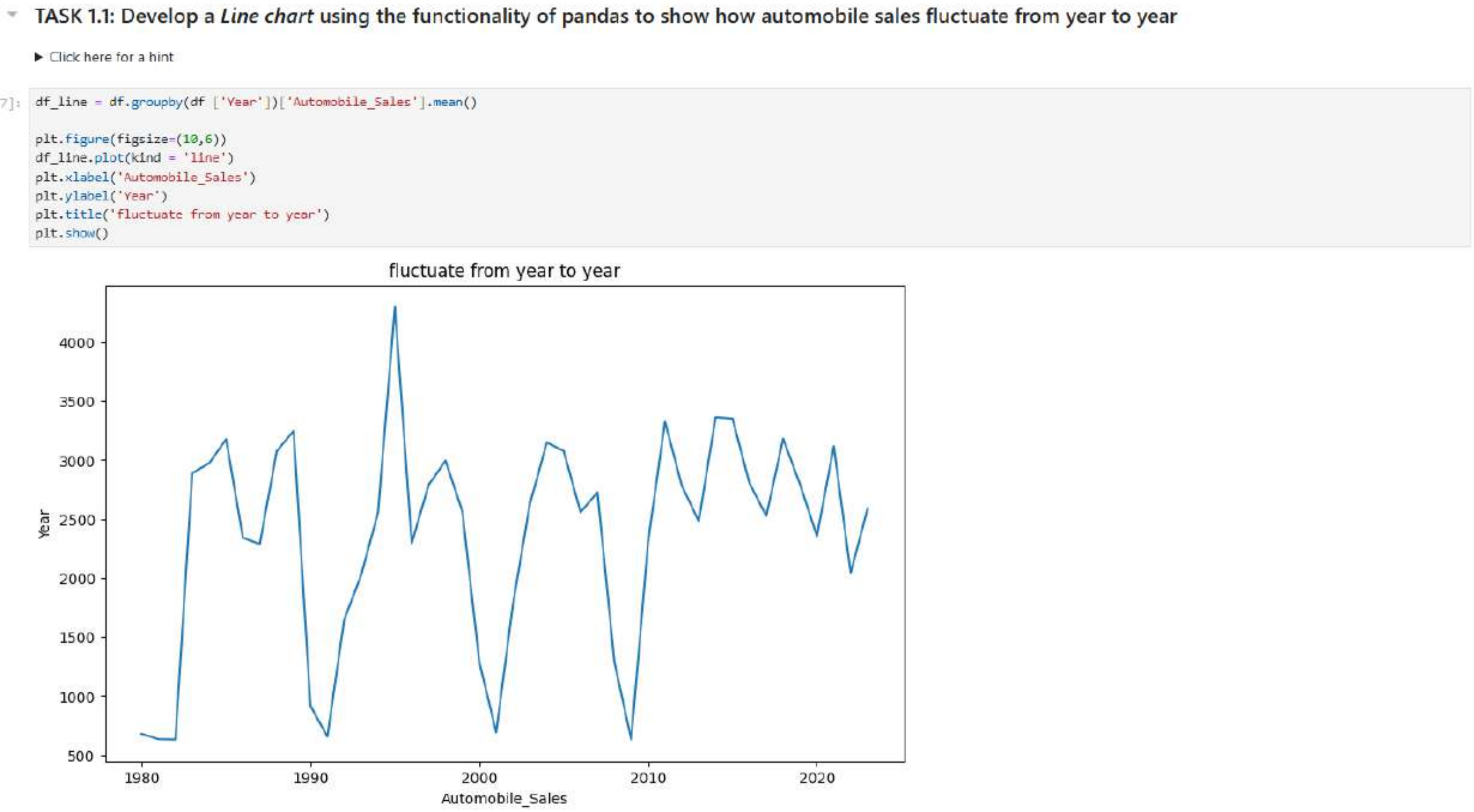
Select Statistics:

Yearly Statistics

Select Year:

1981

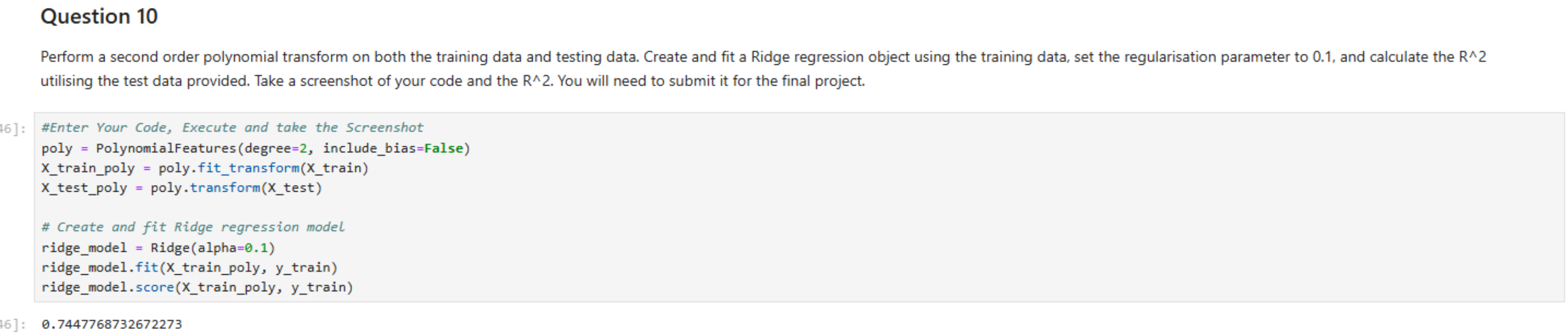
Automobile Sales Statistics Dashboard



TASK 1.1: Develop a *Line chart* using the functionality of pandas to show how automobile sales fluctuate from year to year

▶ Click here for a hint

```
7]: df_line = df.groupby(df ['Year'])['Automobile_Sales'].mean()

    plt.figure(figsize=(10,6))
    df_line.plot(kind = 'line')
    plt.xlabel('Automobile_Sales')
    plt.ylabel('Year')
    plt.title('fluctuate from year to year')
    plt.show()
```

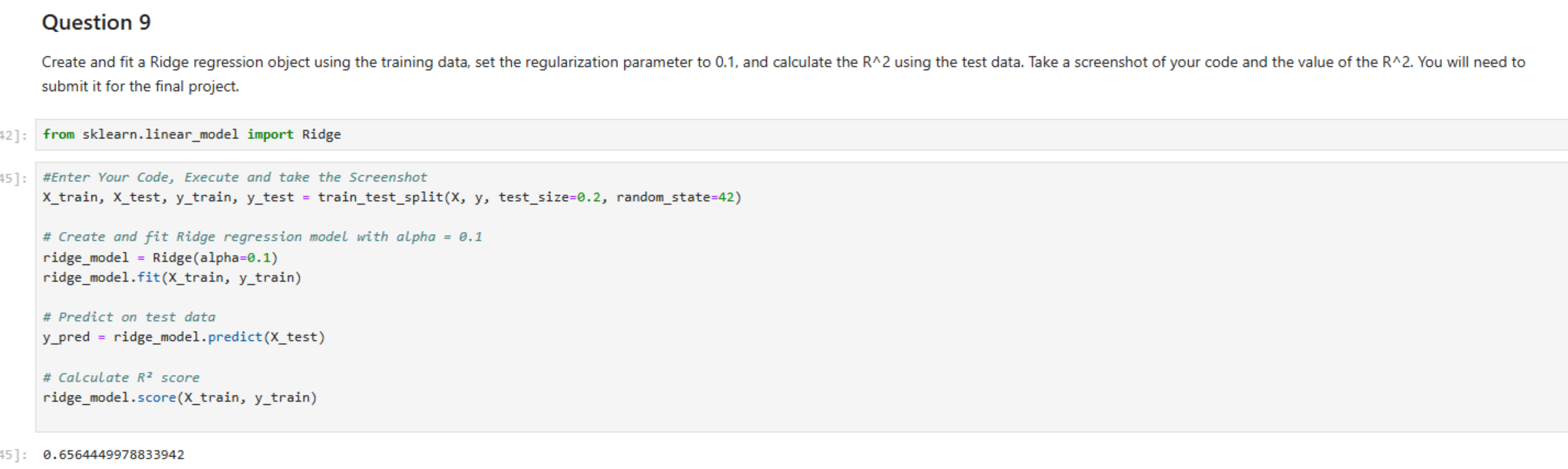


Question 10

Perform a second order polynomial transform on both the training data and testing data. Create and fit a Ridge regression object using the training data, set the regularisation parameter to 0.1, and calculate the R^2 utilising the test data provided. Take a screenshot of your code and the R^2. You will need to submit it for the final project.

```
#Enter Your Code, Execute and take the Screenshot
poly = PolynomialFeatures(degree=2, include_bias=False)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Create and fit Ridge regression model
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(X_train_poly, y_train)
ridge_model.score(X_train_poly, y_train)
```

```
0.7447768732672273
```



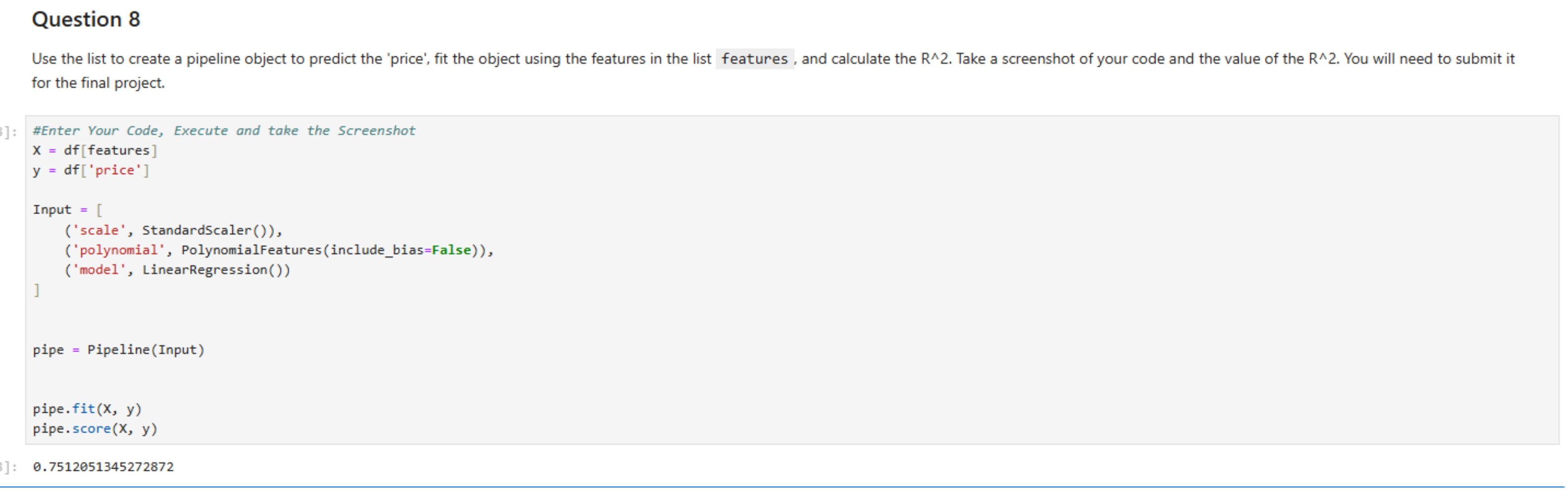
Question 9

Create and fit a Ridge regression object using the training data, set the regularization parameter to 0.1, and calculate the R^2 using the test data. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
42]: from sklearn.linear_model import Ridge
```

```
45]: #Enter Your Code, Execute and take the Screenshot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit Ridge regression model with alpha = 0.1
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(X_train, y_train)

# Predict on test data
y_pred = ridge_model.predict(X_test)

# Calculate R² score
ridge_model.score(X_train, y_train)
```

```
45]: 0.6564449978833942
```

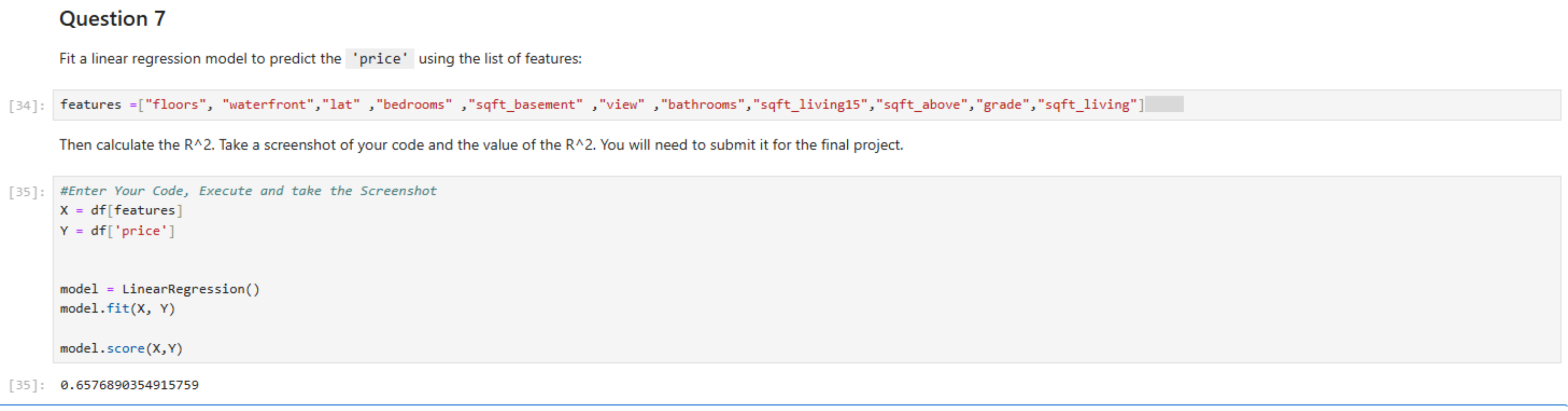


Question 8

Use the list to create a pipeline object to predict the 'price', fit the object using the features in the list `features`, and calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
#Enter Your Code, Execute and take the Screenshot
X = df[features]
y = df['price']

Input = [
    ('scale', StandardScaler()),
    ('polynomial', PolynomialFeatures(include_bias=False)),
    ('model', LinearRegression())
]


pipe = Pipeline(Input)


pipe.fit(X, y)
pipe.score(X, y)
```

```
0.7512051345272872
```



Question 7

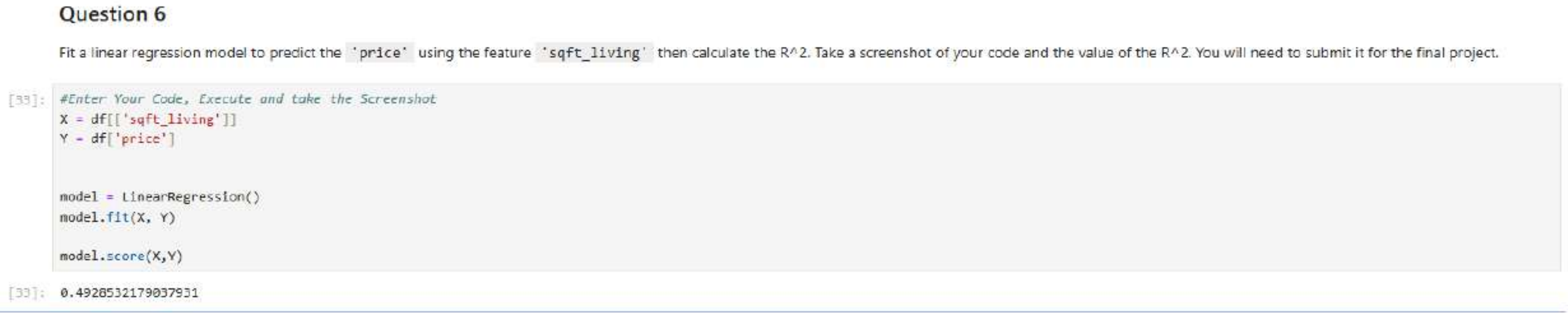
Fit a linear regression model to predict the `'price'` using the list of features:

[34]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
```

Then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

[35]:
```
#Enter Your Code, Execute and take the Screenshot
X = df[features]
Y = df['price']


model = LinearRegression()
model.fit(X, Y)

model.score(X,Y)
```

[35]:
```
0.6576890354915759
```

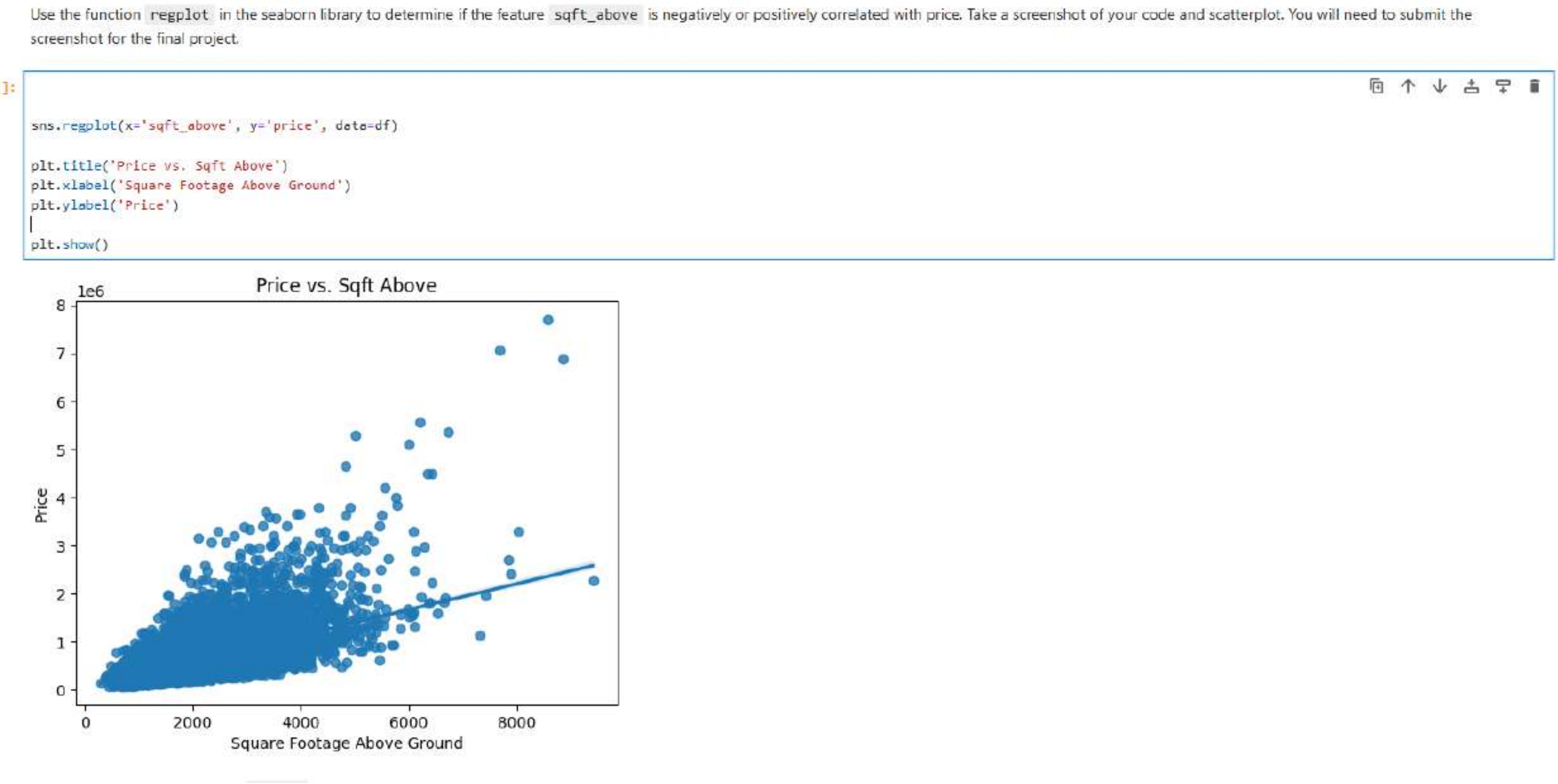


Question 6

Fit a linear regression model to predict the `'price'` using the feature `'sqft_living'` then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

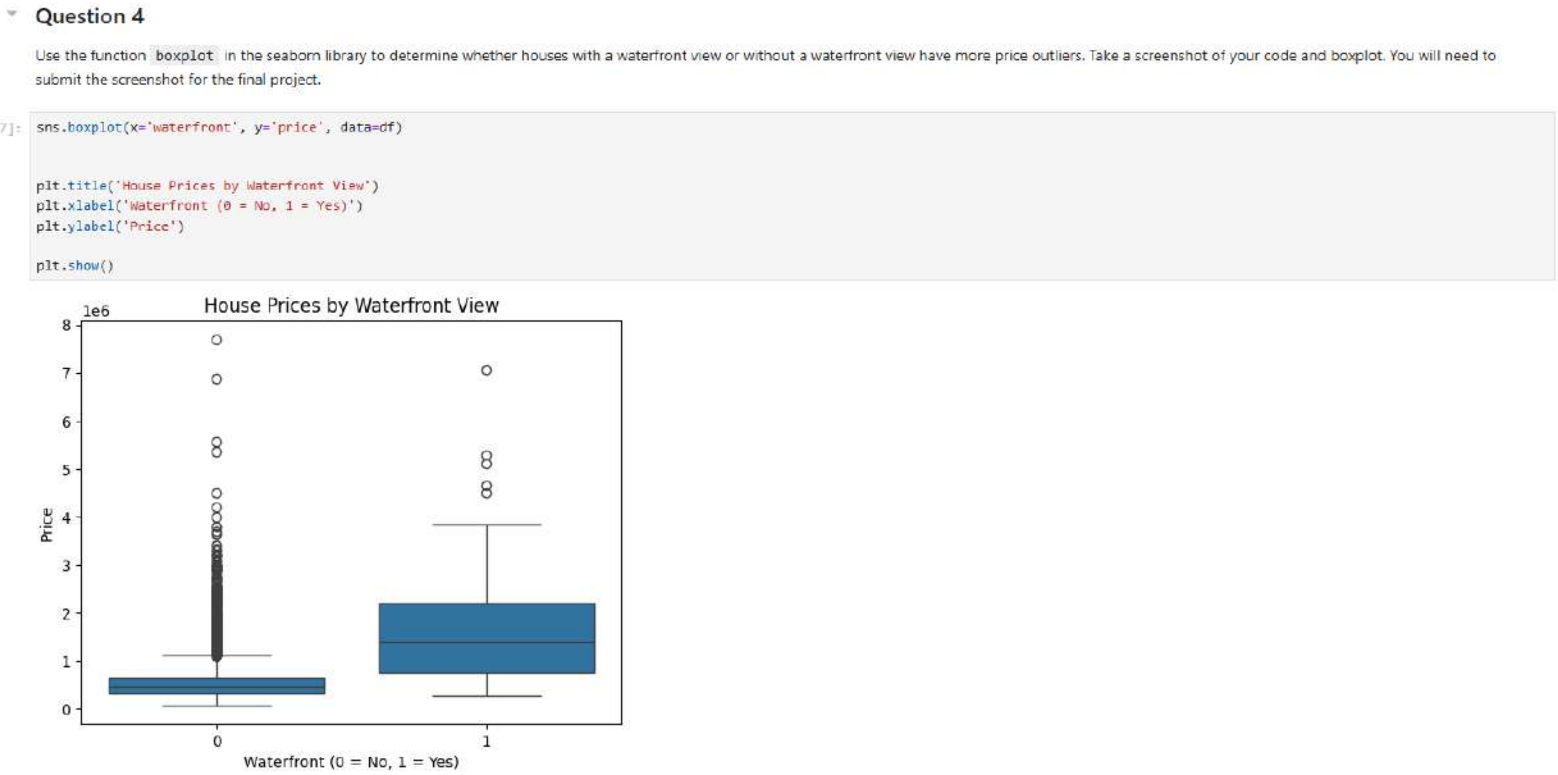
```
[33]: #Enter Your Code, Execute and take the Screenshot
      X = df[['sqft_living']]
      Y = df['price']


      model = LinearRegression()
      model.fit(X, Y)

      model.score(X,Y)
```

```
[33]: 0.4928532179037931
```



Use the function `regplot` in the seaborn library to determine if the feature `sqft_above` is negatively or positively correlated with price. Take a screenshot of your code and scatterplot. You will need to submit the screenshot for the final project.

```
]:
sns.regplot(x='sqft_above', y='price', data=df)

plt.title('Price vs. Sqft Above')
plt.xlabel('Square Footage Above Ground')
plt.ylabel('Price')

plt.show()
```



Question 4

Use the function `boxplot` in the seaborn library to determine whether houses with a waterfront view or without a waterfront view have more price outliers. Take a screenshot of your code and boxplot. You will need to submit the screenshot for the final project.

[27]:

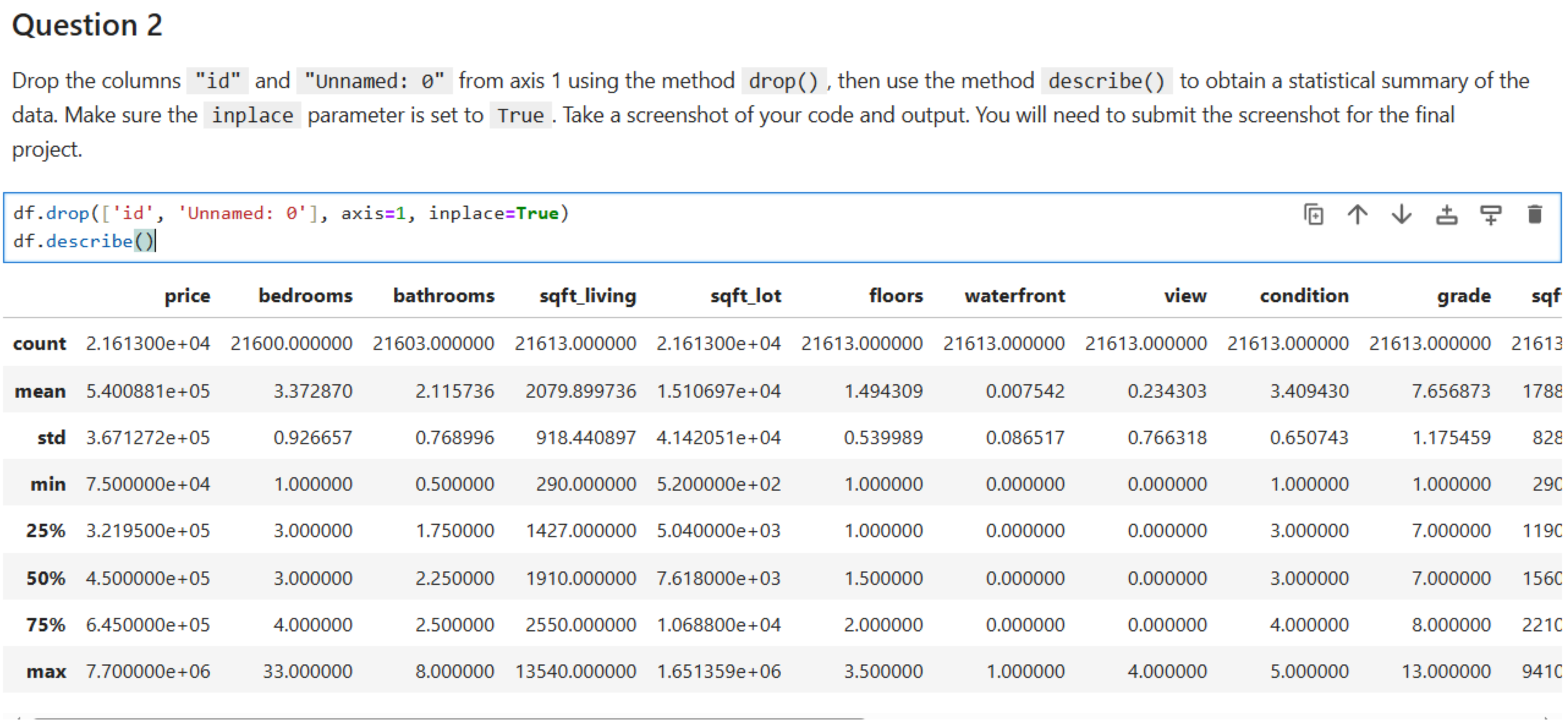
```
sns.boxplot(x='waterfront', y='price', data=df)


plt.title('House Prices by Waterfront View')
plt.xlabel('Waterfront (0 = No, 1 = Yes)')
plt.ylabel('Price')

plt.show()
```

Question 3

Use the method `value_counts` to count the number of houses with unique floor values, use the method `.to_frame()` to convert it to a data frame. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
#Enter Your Code, Execute and take the Screenshot
floor_counts = df['floors'].value_counts().to_frame()
floor_counts
```

```
        count
floors
   1.0  10680
   2.0   8241
   1.5   1910
   3.0    613
   2.5    161
   3.5      8
```

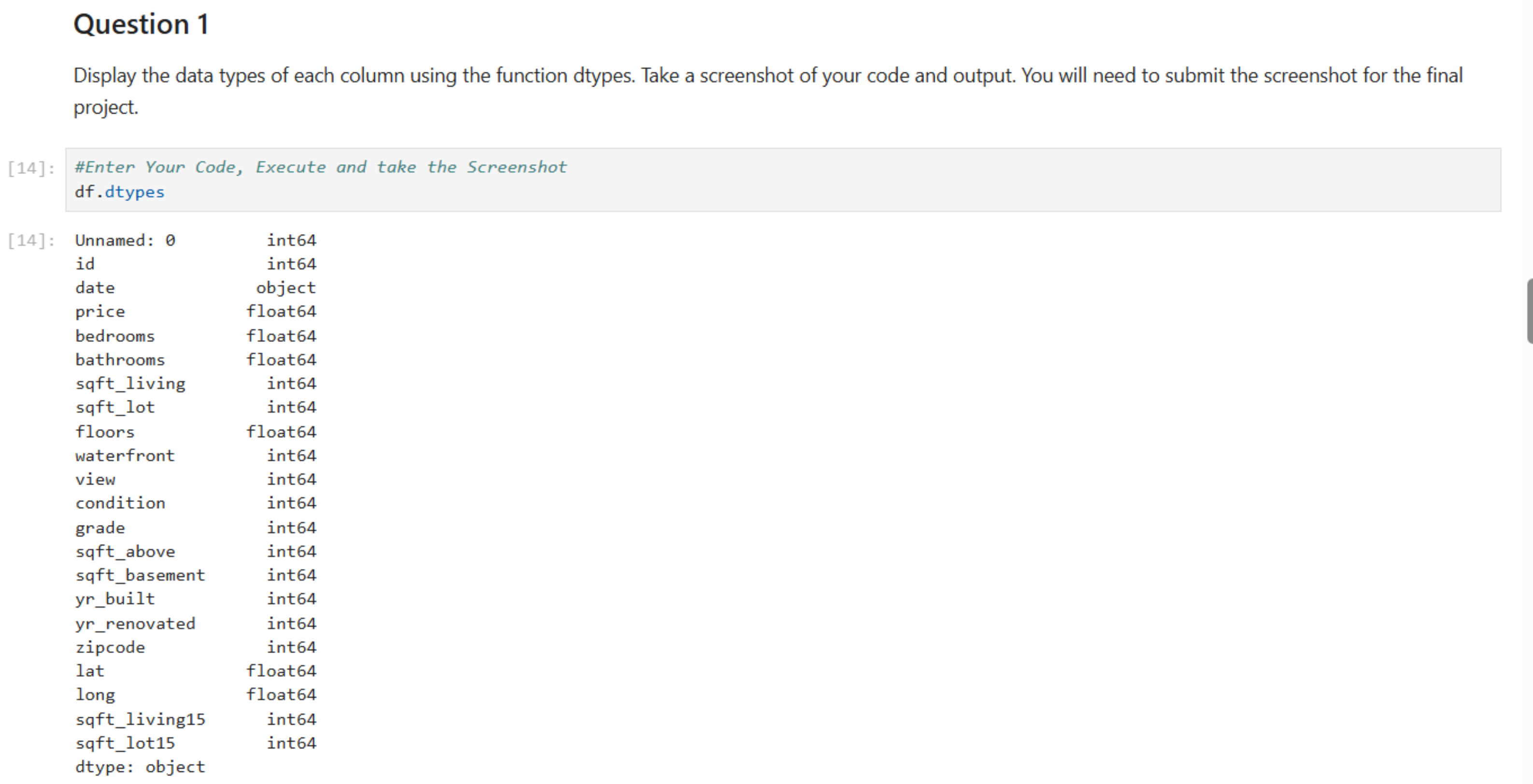


Question 2

Drop the columns `"id"` and `"Unnamed: 0"` from axis 1 using the method `drop()`, then use the method `describe()` to obtain a statistical summary of the data. Make sure the `inplace` parameter is set to `True`. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
df.drop(['id', 'Unnamed: 0'], axis=1, inplace=True)
df.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqf
count	2.161300e+04	21600.000000	21603.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613
mean	5.400881e+05	3.372870	2.115736	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873	1788
std	3.671272e+05	0.926657	0.768996	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459	828
min	7.500000e+04	1.000000	0.500000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000	290
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000	1190
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000	1560
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000	2210
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000	9410

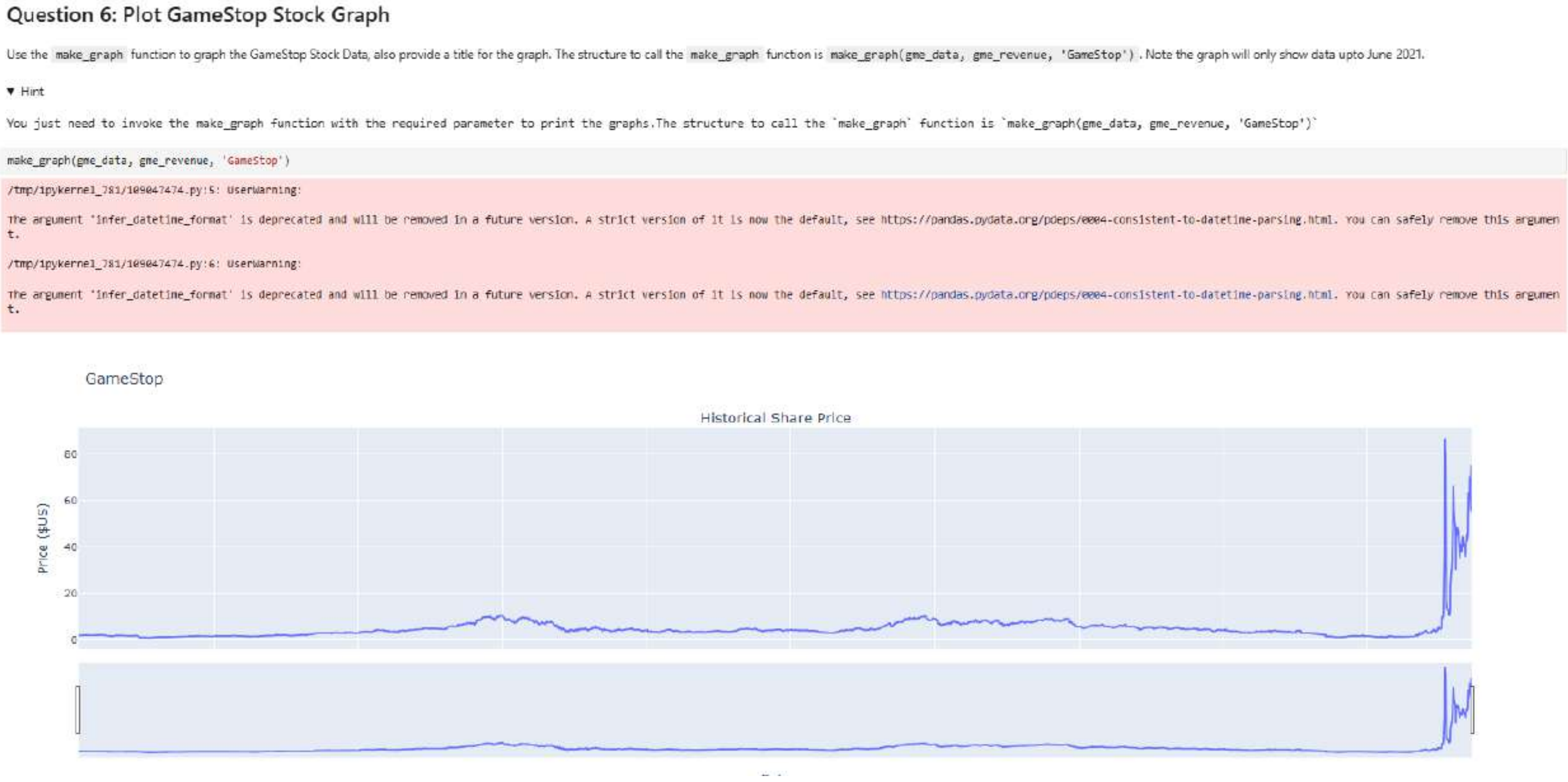


Question 1

Display the data types of each column using the function dtypes. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[14]: #Enter Your Code, Execute and take the Screenshot
      df.dtypes
```

```
[14]: Unnamed: 0          int64
      id                  int64
      date               object
      price             float64
      bedrooms          float64
      bathrooms         float64
      sqft_living         int64
      sqft_lot            int64
      floors            float64
      waterfront          int64
      view                int64
      condition           int64
      grade               int64
      sqft_above          int64
      sqft_basement       int64
      yr_built            int64
      yr_renovated        int64
      zipcode             int64
      lat               float64
      long              float64
      sqft_living15       int64
      sqft_lot15          int64
      dtype: object
```



Question 6: Plot GameStop Stock Graph

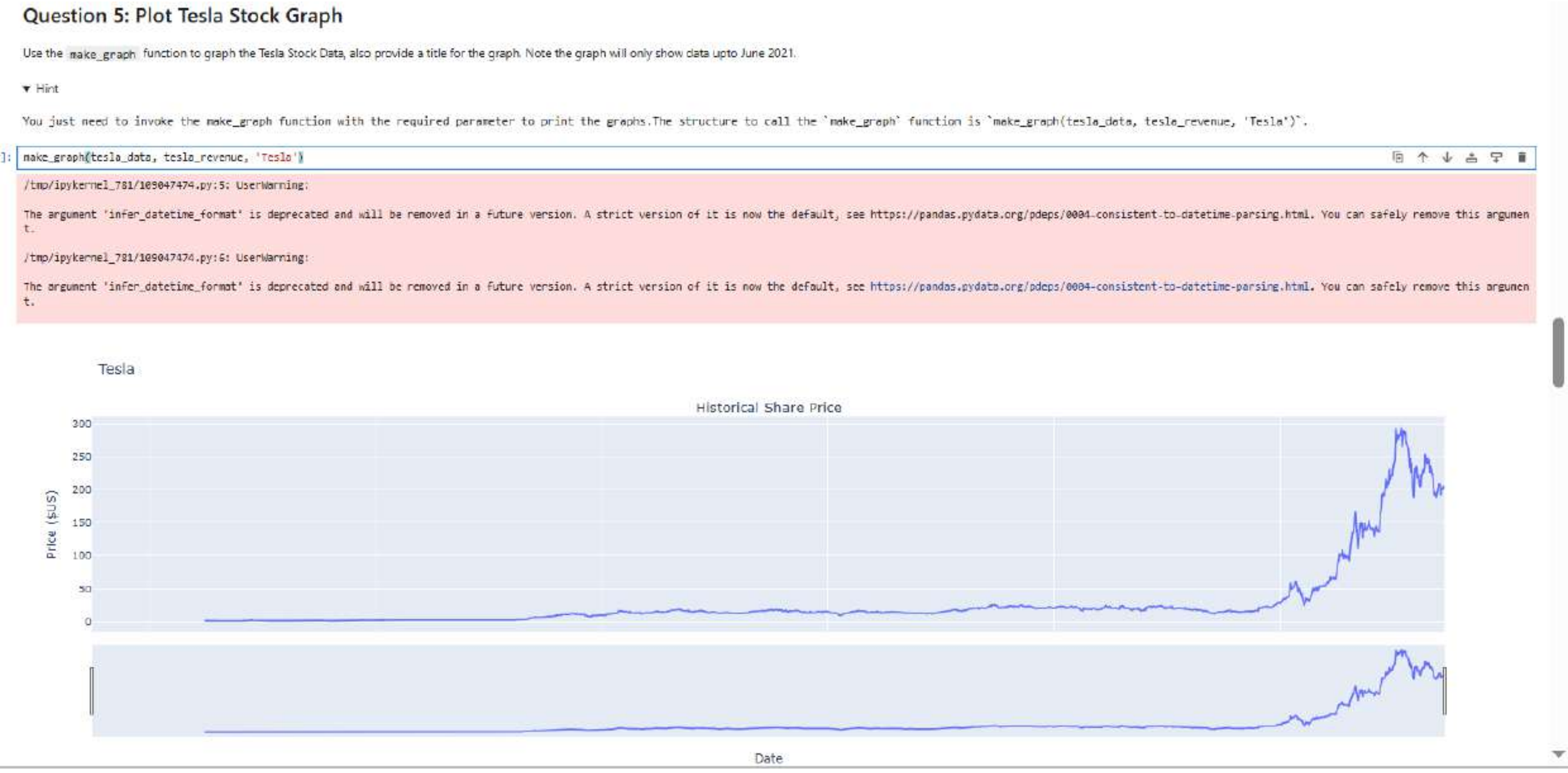
Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

▼ Hint

```
You just need to invoke the make_graph function with the required parameter to print the graphs.The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`
```

```
make_graph(gme_data, gme_revenue, 'GameStop')
```

```
/tmp/ipykernel_781/109047474.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argumen
t.

/tmp/ipykernel_781/109047474.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argumen
t.
```



Question 5: Plot Tesla Stock Graph

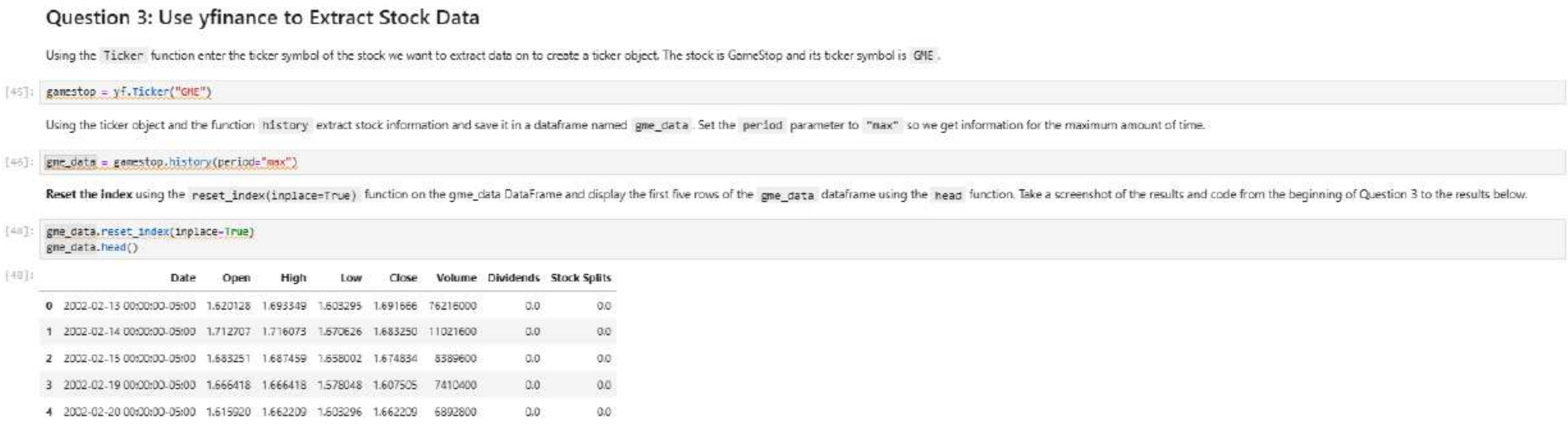
Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

▼ Hint

You just need to invoke the make_graph function with the required parameter to print the graphs.The structure to call the `make_graph` function is `make_graph(tesla_data, tesla_revenue, 'Tesla')`.

```
]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```

```
/tmp/ipykernel_781/109047474.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argumen
t.

/tmp/ipykernel_781/109047474.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You can safely remove this argumen
t.
```



Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

```
[45]: gamestop = yf.Ticker("GME")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

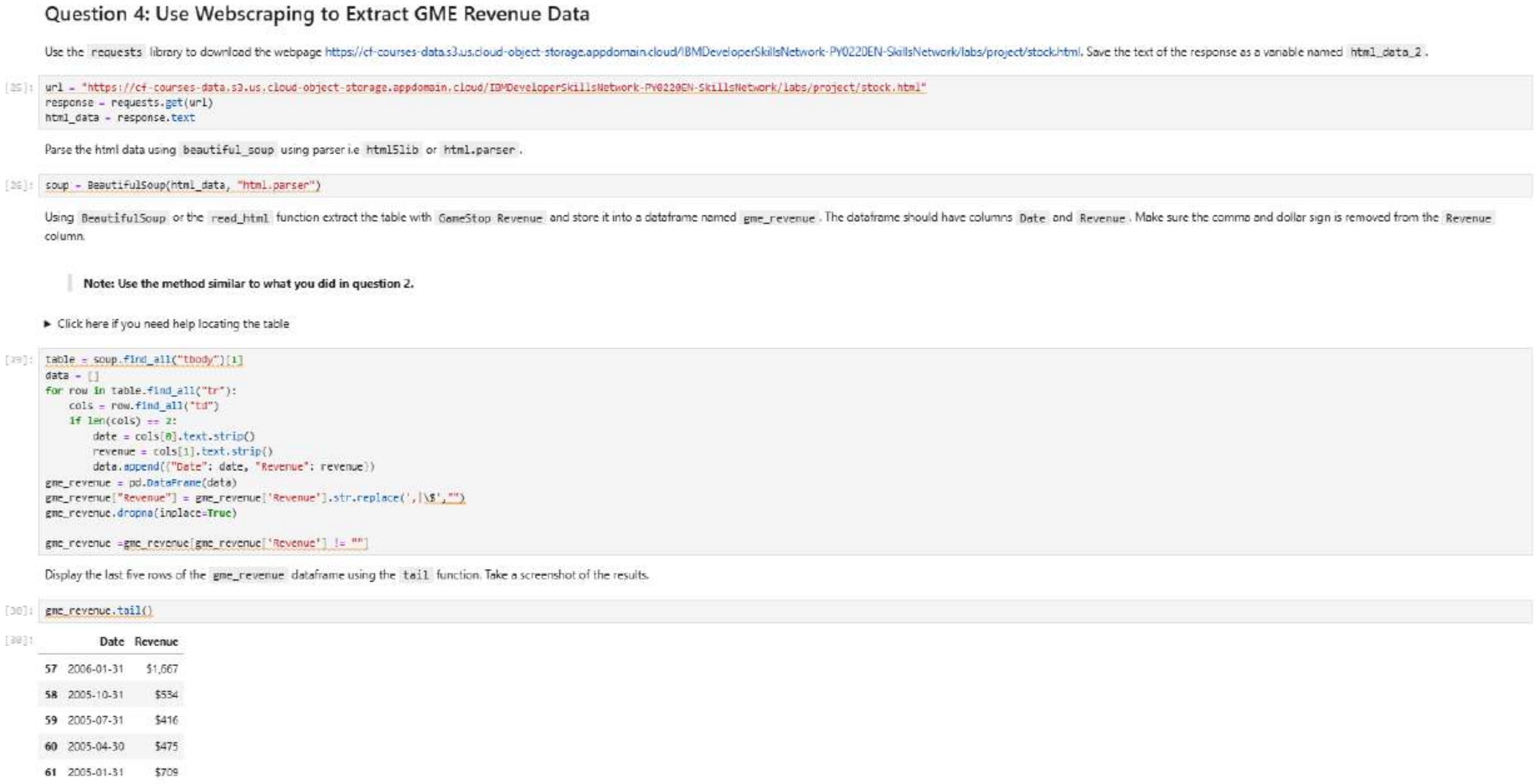
```
[46]: gme_data = gamestop.history(period="max")
```

Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

```
[48]: gme_data.reset_index(inplace=True)
      gme_data.head()
```

[48]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620128	1.693349	1.603295	1.691666	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716073	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683251	1.687459	1.658002	1.674834	8389600	0.0	0.0
3	2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578048	1.607505	7410400	0.0	0.0
4	2002-02-20 00:00:00-05:00	1.615920	1.662209	1.603296	1.662209	6892800	0.0	0.0



Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2`.

[25]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
response = requests.get(url)
html_data = response.text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

[26]:
```
soup = BeautifulSoup(html_data, "html.parser")
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

> **Note: Use the method similar to what you did in question 2.**

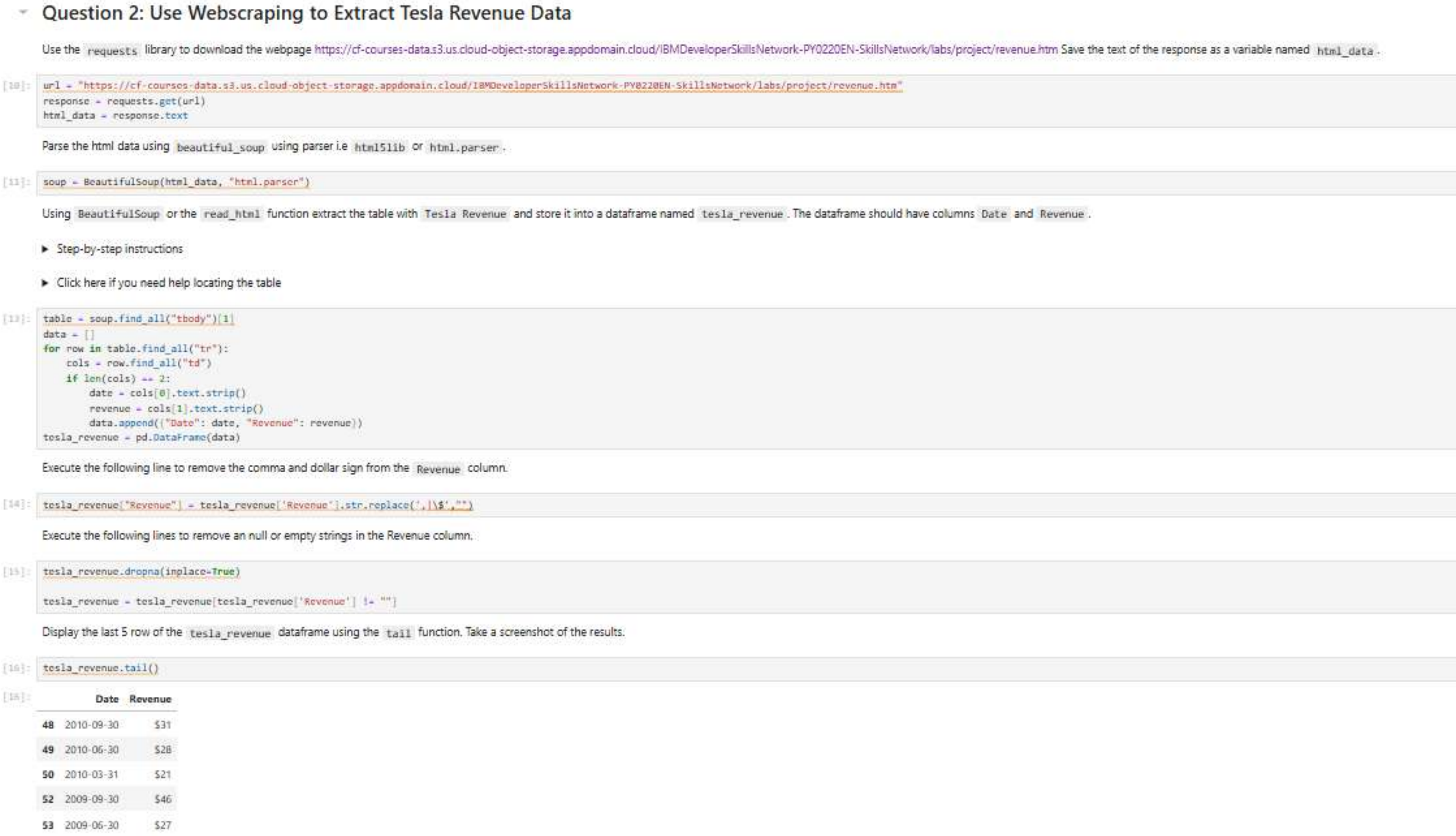
▶ Click here if you need help locating the table

[29]:
```
table = soup.find_all("tbody")[1]
data = []
for row in table.find_all("tr"):
    cols = row.find_all("td")
    if len(cols) == 2:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        data.append({"Date": date, "Revenue": revenue})
gme_revenue = pd.DataFrame(data)
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
gme_revenue.dropna(inplace=True)

gme_revenue =gme_revenue[gme_revenue['Revenue'] != ""]
```

Display the last five rows of the `gme_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[30]:
```
gme_revenue.tail()
```

[30]:
```
          Date  Revenue
57  2006-01-31   $1,667
58  2005-10-31     $534
59  2005-07-31     $416
60  2005-04-30     $475
61  2005-01-31     $709
```



Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

[10]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
response = requests.get(url)
html_data = response.text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

[11]:
```
soup = BeautifulSoup(html_data, "html.parser")
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

▶ Step-by-step instructions

▶ Click here if you need help locating the table

[13]:
```
table = soup.find_all("tbody")[1]
data = []
for row in table.find_all("tr"):
    cols = row.find_all("td")
    if len(cols) == 2:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        data.append({"Date": date, "Revenue": revenue})
tesla_revenue = pd.DataFrame(data)
```

Execute the following line to remove the comma and dollar sign from the `Revenue` column.

[14]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

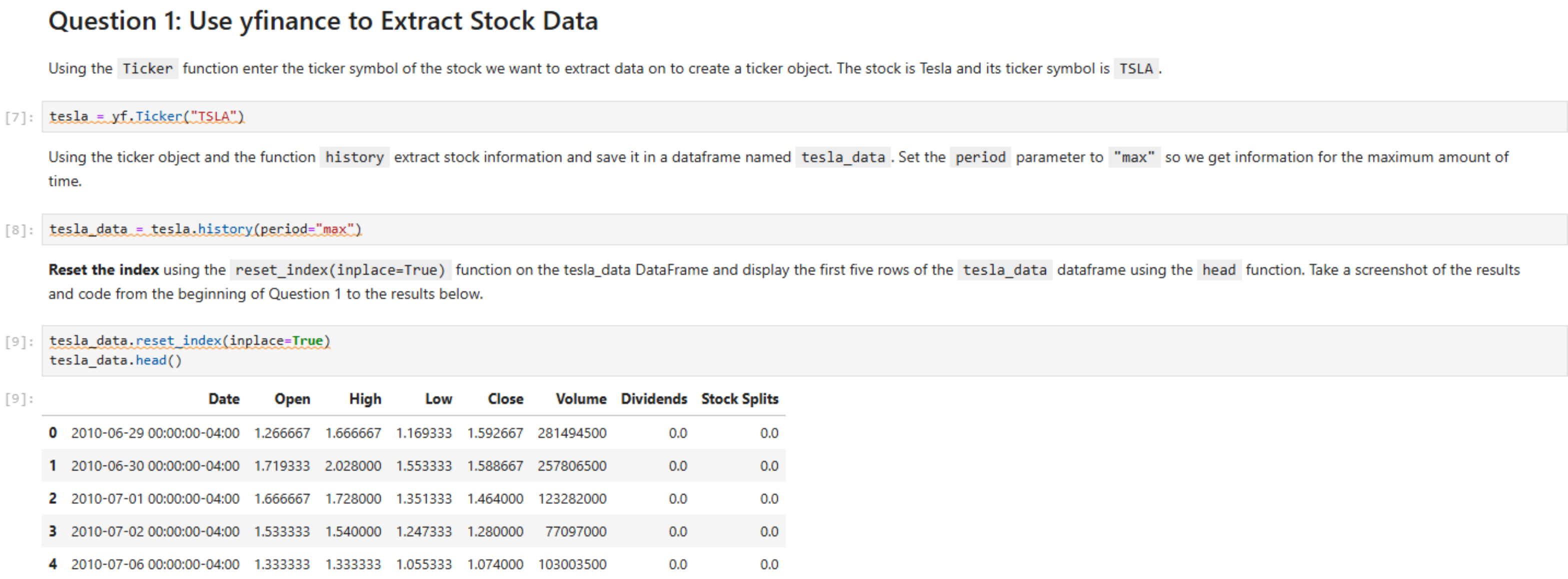
Execute the following lines to remove an null or empty strings in the Revenue column.

[15]:
```
tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[16]:
```
tesla_revenue.tail()
```

[16]:
```
          Date  Revenue
48  2010-09-30      $31
49  2010-06-30      $28
50  2010-03-31      $21
52  2009-09-30      $46
53  2009-06-30      $27
```



Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

```
[7]: tesla = yf.Ticker("TSLA")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

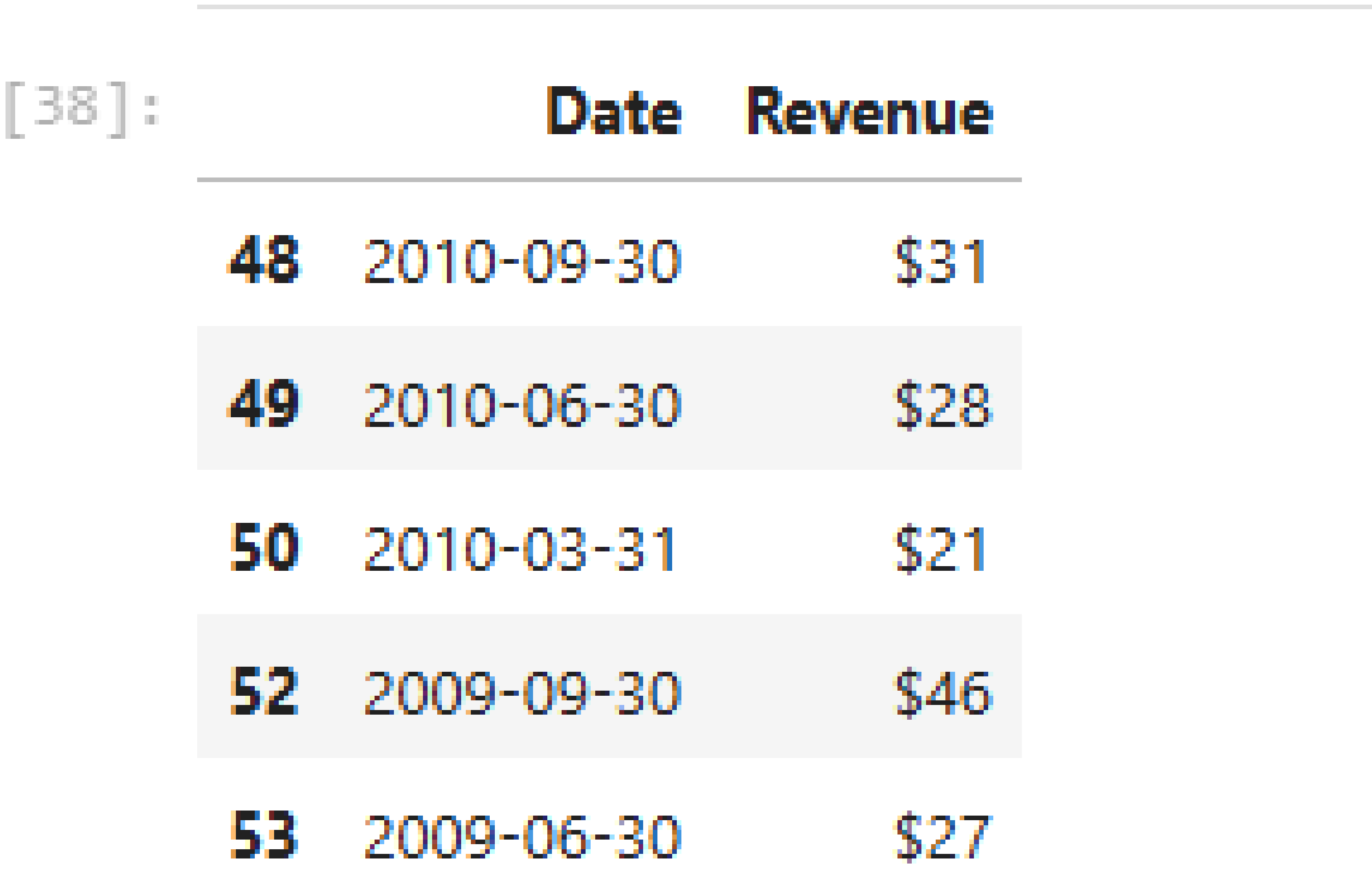
```
[8]: tesla_data = tesla.history(period="max")
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

```
[9]: tesla_data.reset_index(inplace=True)
     tesla_data.head()
```

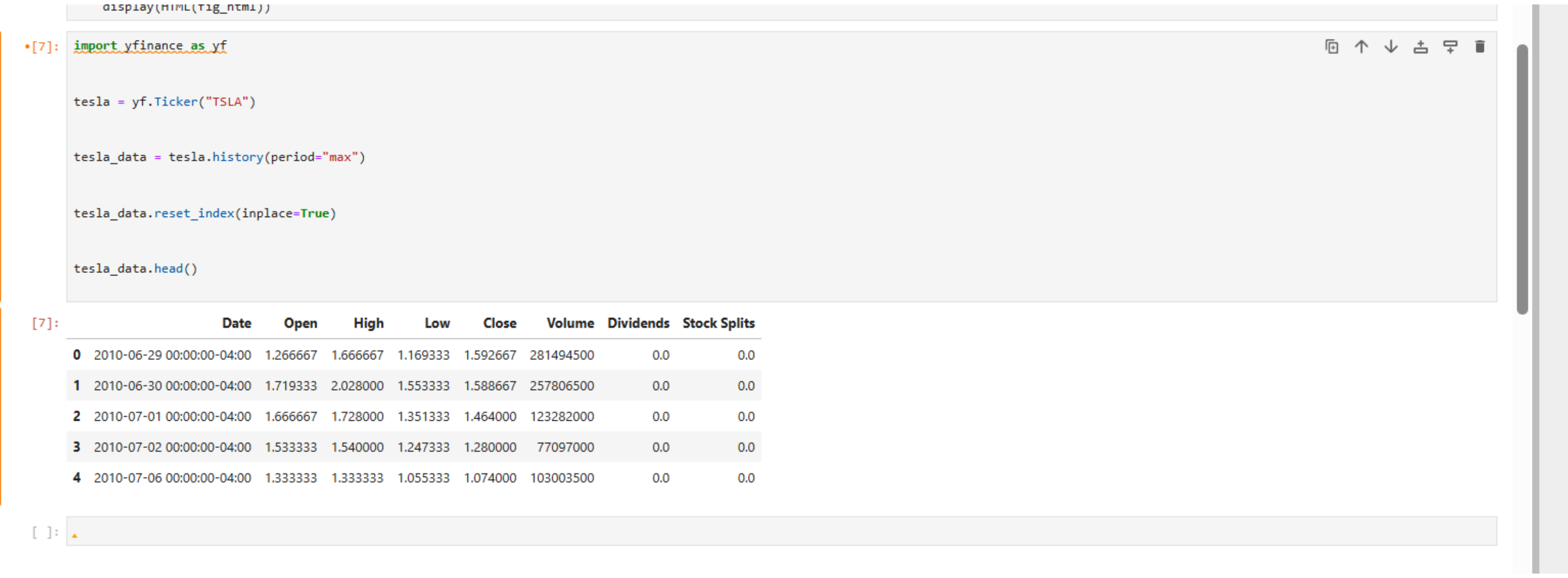
[9]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
1	2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2	2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
3	2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
4	2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	0.0	0.0



[38]:

	Date	Revenue
48	2010-09-30	$31
49	2010-06-30	$28
50	2010-03-31	$21
52	2009-09-30	$46
53	2009-06-30	$27



```
    display(HTML(fig_html))
```

•[7]:

```
import yfinance as yf


tesla = yf.Ticker("TSLA")


tesla_data = tesla.history(period="max")


tesla_data.reset_index(inplace=True)


tesla_data.head()
```

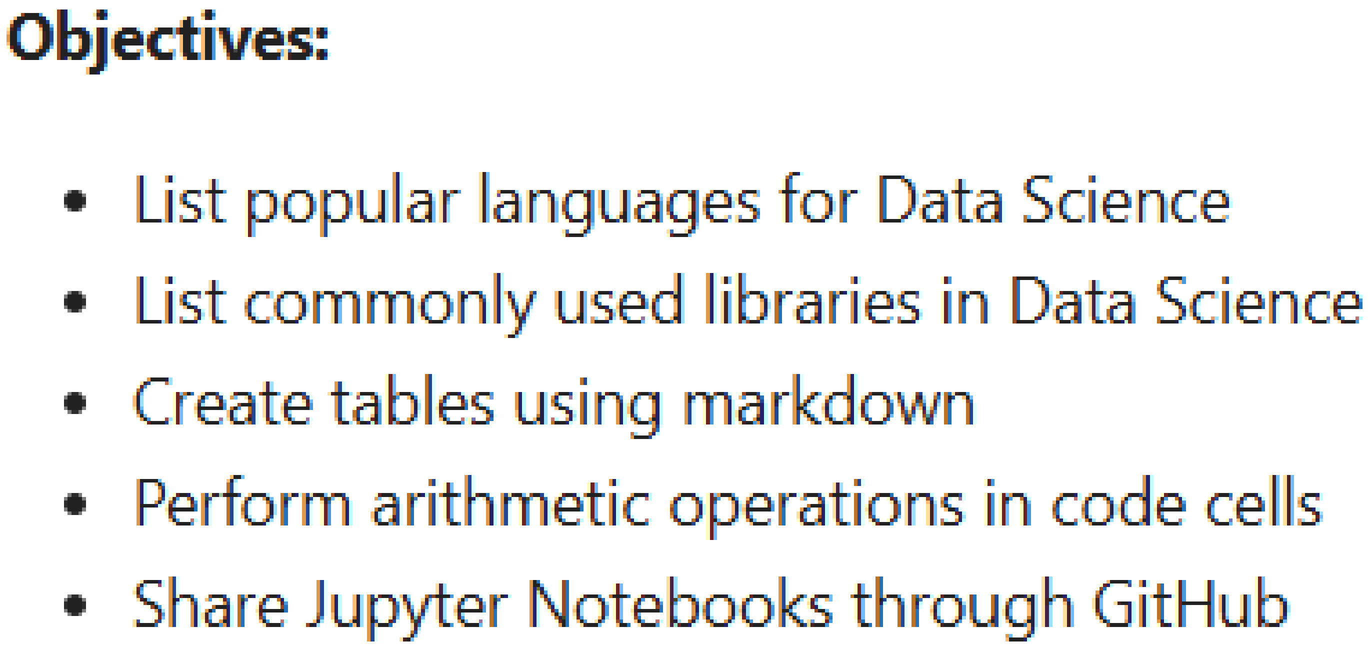
[7]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
1	2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2	2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
3	2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
4	2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	0.0	0.0

[ ]:

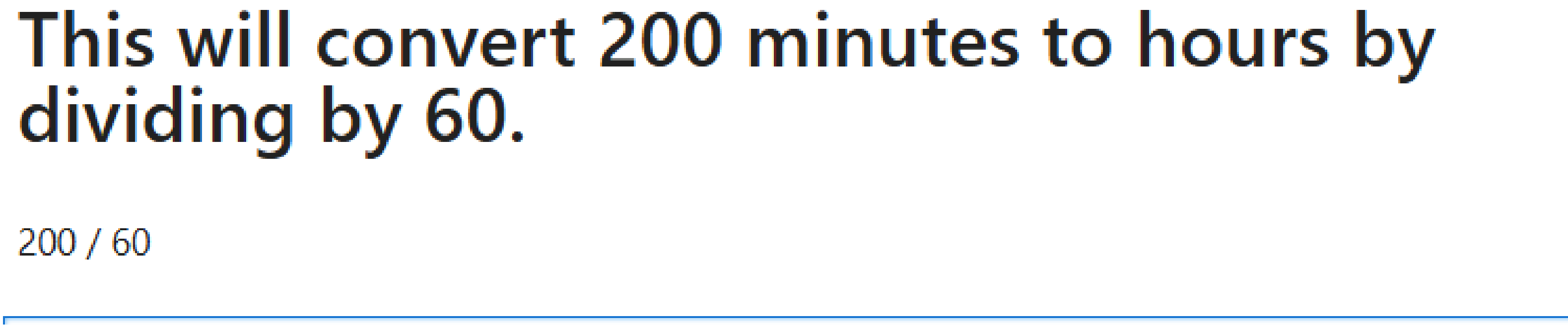
Author

B sai vinay



Objectives:

- List popular languages for Data Science
- List commonly used libraries in Data Science
- Create tables using markdown
- Perform arithmetic operations in code cells
- Share Jupyter Notebooks through GitHub



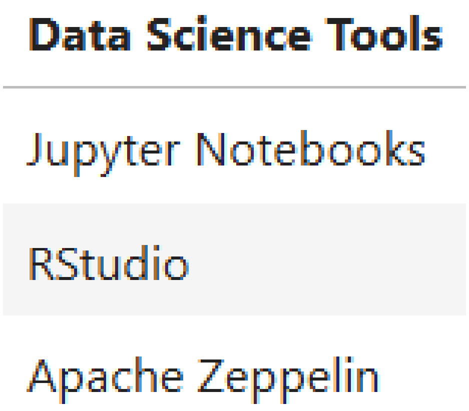
This will convert 200 minutes to hours by dividing by 60.

```
200 / 60
```

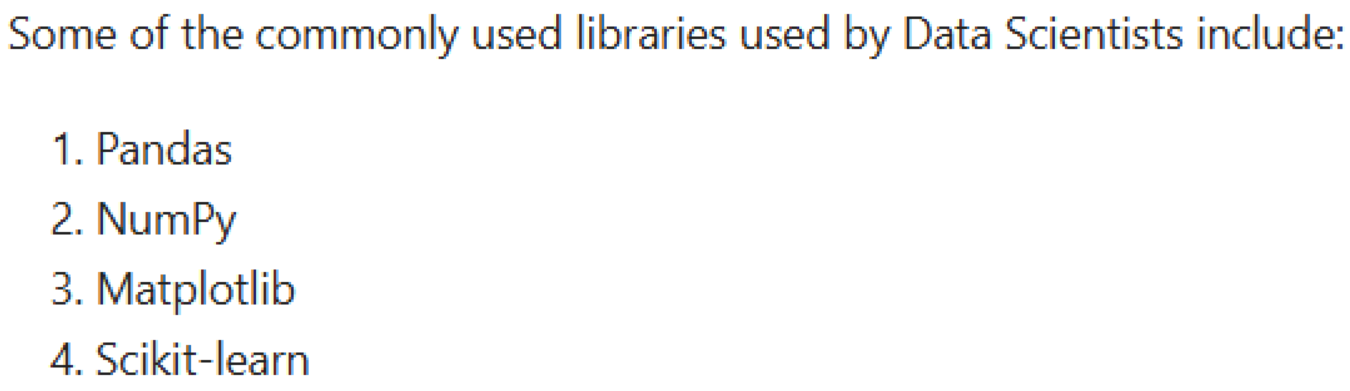
This is a simple arithmetic expression to multiply then add integers.

(3 * 4) + 5

Below are a few examples of evaluating arithmetic expressions in Python.

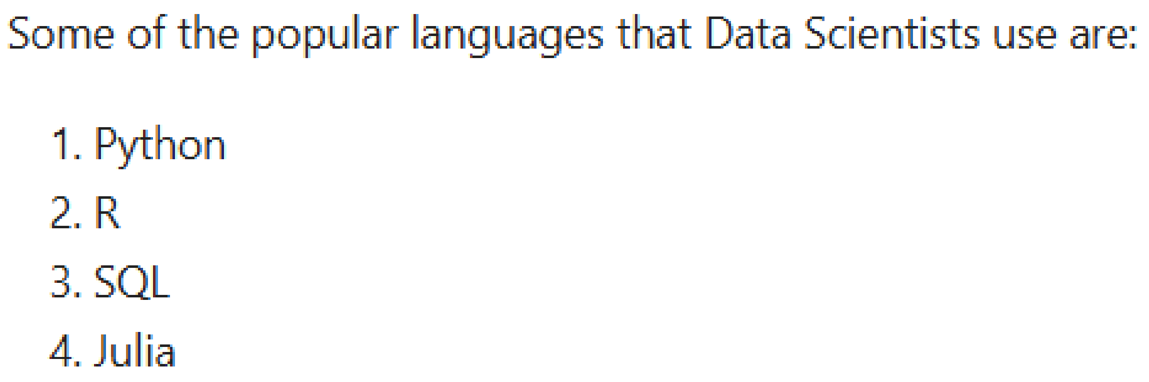


Data Science Tools
Jupyter Notebooks
RStudio
Apache Zeppelin



Some of the commonly used libraries used by Data Scientists include:

1. Pandas
2. NumPy
3. Matplotlib
4. Scikit-learn



Some of the popular languages that Data Scientists use are:

1. Python
2. R
3. SQL
4. Julia

In this notebook, Data Science Tools and Ecosystem are summarized.

Data Science Tools and Ecosystem