A

Pick any publicly-traded company that trades on the Nasdaq or the NYSE.

a. What company did you select, and what is its ticker symbol?

Apple Inc., which trades on the Nasdaq under the ticker symbol "AAPL".

B

Loaded the data

In [ ]:
```python
#c
import pandas as pd

apple=pd.read_csv("AAPL.csv",index_col="Date",parse_dates=True)

apple.head(10)
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2022-04-25	161.119995	163.169998	158.460007	162.880005	161.899261	96046400
2022-04-26	162.250000	162.339996	156.720001	156.800003	155.855881	95623200
2022-04-27	155.910004	159.789993	155.380005	156.570007	155.627258	88063200
2022-04-28	159.250000	164.520004	158.929993	163.639999	162.654694	130216800
2022-04-29	161.839996	166.199997	157.250000	157.649994	156.700729	131747600
2022-05-02	156.710007	158.229996	153.270004	157.960007	157.008896	123055300
2022-05-03	158.149994	160.710007	156.320007	159.479996	158.519730	88966500
2022-05-04	159.669998	166.479996	159.259995	166.020004	165.020370	108256500
2022-05-05	163.850006	164.080002	154.949997	156.770004	155.826050	130525300
2022-05-06	156.009995	159.440002	154.179993	157.279999	156.562668	116124600

In [ ]:
```python
#c

apple.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 250 entries, 2022-04-25 to 2023-04-21
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       250 non-null    float64
 1   High       250 non-null    float64
 2   Low        250 non-null    float64
 3   Close      250 non-null    float64
 4   Adj Close  250 non-null    float64
 5   Volume     250 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 13.7 KB
```

In [ ]:
```python
#D

# check the data type of the index column
print(apple.index.dtype)
```

```
datetime64[ns]
```

D

The DataFrame provided is indexed by time values, as evidenced by the "Date" column and confirmed by checking the data type of the index column.

E (A)

Now, view the max and min value of your index attribute.

In [ ]:
```python
#E (a)

apple.index.max()
```

Out[ ]: `Timestamp('2023-04-21 00:00:00')`

In [ ]:
```python
#E (a)

apple.index.min()
```

Out[ ]: `Timestamp('2022-04-25 00:00:00')`

E(B)

Now, view the argmax and argmin values of your index attribute.

In [ ]:
```python
#E(B)
apple.index.argmax()
```

Out[ ]: 249

In [ ]:
```python
#E(B)
apple.index.argmin()
```

Out[ ]: 0

What do the results of max, min, argmax, and argmin represent?

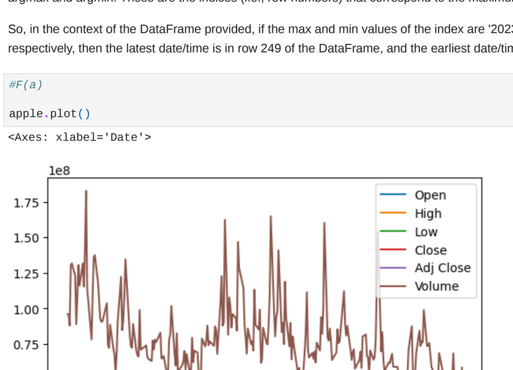
max and min: These are the maximum and minimum values in the index, respectively. In a time-based index, these values represent the latest and earliest dates/times in the DataFrame.

argmax and argmin: These are the indices (i.e., row numbers) that correspond to the maximum and minimum values in the index, respectively. In a time-based index, these values represent the row numbers that correspond to the latest and earliest dates/times in the DataFrame.

So, in the context of the DataFrame provided, if the max and min values of the index are '2023-04-21' and '2022-04-25', respectively, then the DataFrame covers a period of time from April 25, 2022 to April 21, 2023. And if the argmax and argmin values of the index are 249 and 0, respectively, then the latest date/time is in row 249 of the DataFrame, and the earliest date/time is in row 0 of the DataFrame.

In [ ]:
```python
#F(a)

apple.plot()
```

Out[ ]: `<Axes: xlabel='Date'>`



F(a)

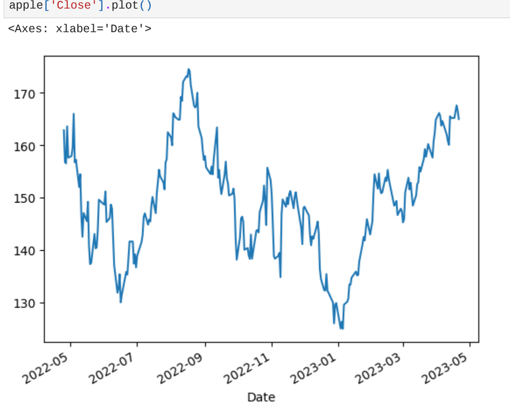
the plot shows multiple lines, it may be challenging to interpret because it can be difficult to distinguish between the different lines and to determine which line corresponds to which variable. Additionally, the plot may be difficult to read if the x-axis labels are not formatted in a clear and consistent manner, or if there are too many data points or too much information displayed at once.

To make the plot easier to understand, it can be helpful to add labels to the axes and to the individual lines, to use different colors or line styles for different lines, and to adjust the scale and formatting of the axes to better display the data. Additionally, it may be helpful to plot subsets of the data at a time or to aggregate the data in a meaningful way before plotting it.

As seen ,its trying to plot for all the axes and its really important to choose the ones that makes sense and helpful to the model as said above.

In [ ]:
```python
#F(B)
apple['Close'].plot()
```

Out[ ]: `<Axes: xlabel='Date'>`

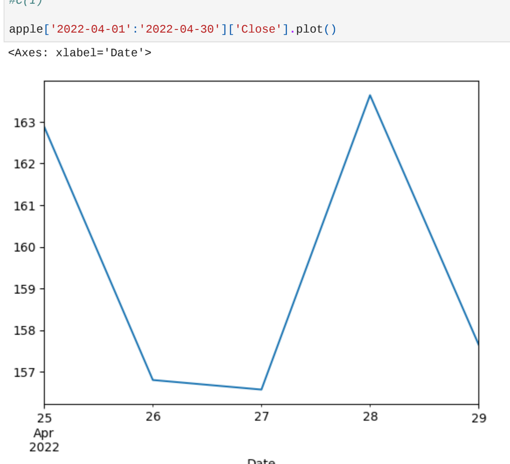


F(B)

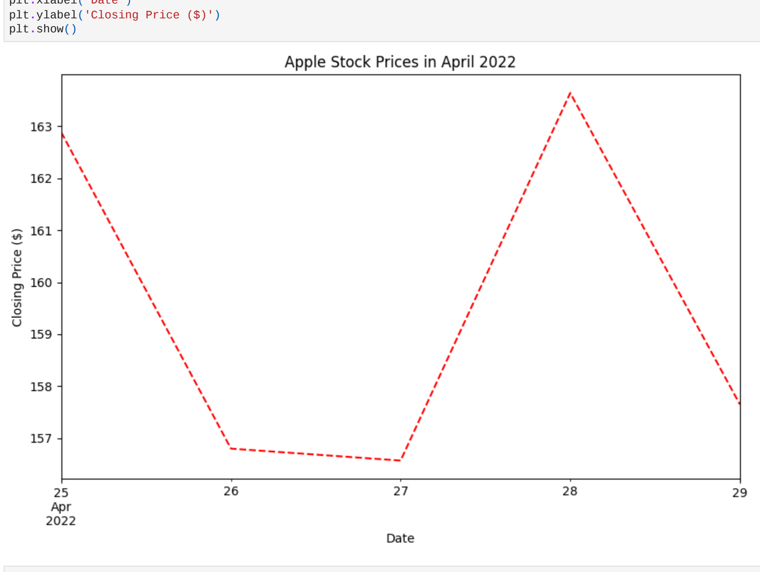
Plotting the 'Close' variable only makes the graph more easily interpretable because it simplifies the visual representation to only show the closing prices for the stock. This allows for a clearer understanding of the overall trend of the stock price over time, without the potential confusion of having multiple lines for different variables.

In [ ]:
```python
#C(I)

apple['2022-04-01':'2022-04-30']['Close'].plot()
```

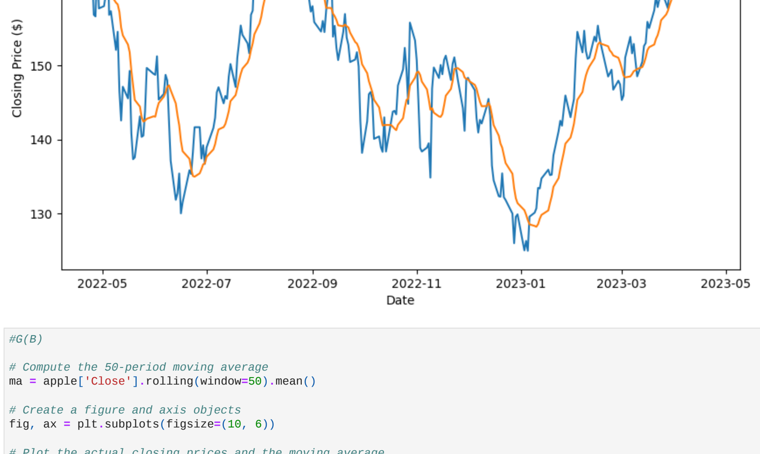
Out[ ]: `<Axes: xlabel='Date'>`



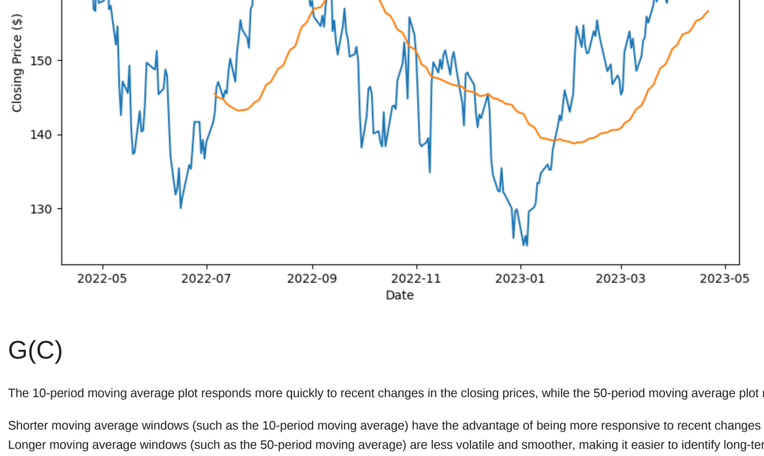
In [ ]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
apple['2022-04-01':'2022-04-30']['Close'].plot(color='red', linestyle='--')
plt.title('Apple Stock Prices in April 2022')
plt.xlabel('Date')
plt.ylabel('Closing Price ($)')
plt.show()
```



In [ ]:
```python
#G(A)

# Compute the 10-period moving average
ma = apple['Close'].rolling(window=10).mean()

# Create a figure and axis objects
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the actual closing prices and the moving average
ax.plot(apple['Close'], label='Actual Closing Prices')
ax.plot(ma, label='10-Period Moving Average')

# Set the axis labels and legend
ax.set_xlabel('Date')
ax.set_ylabel('Closing Price ($)')
ax.legend()

# Show the plot
plt.show()
```



In [ ]:
```python
#G(B)

# Compute the 50-period moving average
ma = apple['Close'].rolling(window=50).mean()

# Create a figure and axis objects
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the actual closing prices and the moving average
ax.plot(apple['Close'], label='Actual Closing Prices')
ax.plot(ma, label='50-Period Moving Average')

# Set the axis labels and legend
ax.set_xlabel('Date')
ax.set_ylabel('Closing Price ($)')
ax.legend()

# Show the plot
plt.show()
```



G(C)

The 10-period moving average plot responds more quickly to recent changes in the closing prices, while the 50-period moving average plot responds more slowly and is smoother.

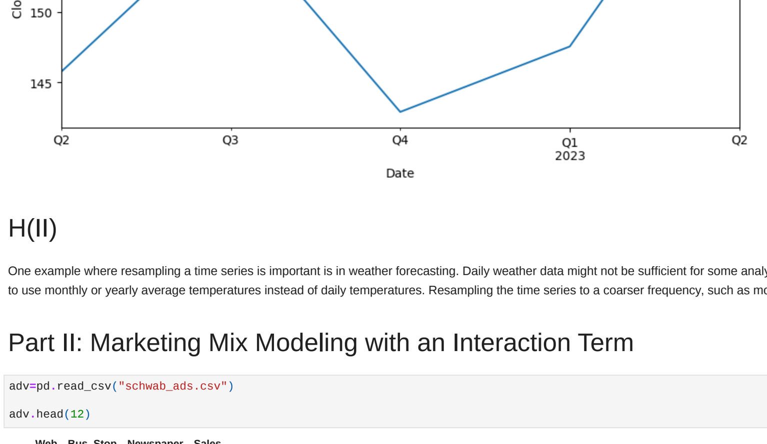
Shorter moving average windows (such as the 10-period moving average) have the advantage of being more responsive to recent changes in prices, allowing traders to identify trends earlier. However, they are also more volatile and susceptible to false signals, which can lead to losses. Longer moving average windows (such as the 50-period moving average) are less volatile and smoother, making it easier to identify long-term trends. However, they are less responsive to recent changes in prices, which can result in missed opportunities to enter or exit trades. Ultimately, the choice of moving average window length will depend on the trading strategy and time horizon of the trader.

Resampling

In [ ]:
```python
#H (I)
apple_quarterly = apple['Close'].resample('Q').mean()
print(apple_quarterly)
```

```
Date
2022-06-30    145.752129
2022-09-30    156.950468
2022-12-31    142.911110
2023-03-31    147.551129
2023-06-30    164.637144
Freq: Q-DEC, Name: Close, dtype: float64
```

In [ ]:
```python
#H (I)
fig, ax = plt.subplots(figsize=(10, 5))
apple_quarterly.plot(ax=ax)

# set the x-axis label
ax.set_xlabel('Date')

# set the y-axis label
ax.set_ylabel('Closing Price (USD)')

# set the plot title
ax.set_title('Apple Quarterly Closing Prices')

# show the plot
plt.show()
```



H(II)

One example where resampling a time series is important is in weather forecasting. Daily weather data might not be sufficient for some analyses or applications. For example, if you wanted to assess the trend in average temperature over a year or a decade, it would be more appropriate to use monthly or yearly average temperatures instead of daily temperatures. Resampling the time series to a coarser frequency, such as monthly or yearly, can help smooth out short-term fluctuations and reveal longer-term patterns in the data.

Part II: Marketing Mix Modeling with an Interaction Term

In [ ]:
```python
adv=pd.read_csv("schwab_ads.csv")

adv.head(12)
```

Out[ ]:

	Web	Bus_Stop	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6
10	66.1	5.8	24.2	12.6
11	214.7	24.0	4.0	17.4

In [ ]:
```python
# create a new variable 'Total_Spending' that shows the total spending
adv['Total_Spending'] = adv['Web'] + adv['Bus_Stop'] + adv['Newspaper']

adv.head(10)
```

Out[ ]:

	Web	Bus_Stop	Newspaper	Sales	Total_spend	Total_Spending
0	230.1	37.8	69.2	22.1	337.1	337.1
1	44.5	39.3	45.1	10.4	128.9	128.9
2	17.2	45.9	69.3	12.0	132.4	132.4
3	151.5	41.3	58.5	16.5	251.3	251.3
4	180.8	10.8	58.4	17.9	250.0	250.0
5	8.7	48.9	75.0	7.2	132.6	132.6
6	57.5	32.8	23.5	11.8	113.8	113.8
7	120.2	19.6	11.6	13.2	151.4	151.4
8	8.6	2.1	1.0	4.8	11.7	11.7
9	199.8	2.6	21.2	15.6	223.6	223.6

In [28]:
```python
adv['Total_spend'].corr(adv['Sales'])
```

Out[28]: 0.924917006249931

The high correlation value of 0.92 suggests a strong positive linear relationship between total marketing spending and sales. However, correlation does not imply causation. There may be other factors that influence sales, such as market demand, competition, product quality, pricing, etc. Also, the relationship may not be linear or may have a time lag. Therefore, we cannot conclude from this correlation alone that more ad spending leads to more sales. Further analysis, such as regression modeling, controlled experiments, and domain expertise, may be needed to establish causal relationships and make actionable recommendations.

In [29]:
```python
#C

adv[['Web','Bus_Stop','Newspaper']].corr()
```

Out[29]:

	Web	Bus_Stop	Newspaper
Web	1.000000	0.054809	0.056648
Bus_Stop	0.054809	1.000000	0.354104
Newspaper	0.056648	0.354104	1.000000

Are any of these correlations so high that we might not be able to use them together in a linear model?

No, none of the correlations are so high that we might not be able to use them together in a linear model. A correlation of 1 would indicate a perfect linear relationship, but the highest correlation value here is only 0.354104, which suggests a relatively weak linear relationship between bus stop ad spending and newspaper ad spending. However, all three variables can still be used together in a linear model, as long as we are aware of their respective correlations and interpret the results accordingly.

In [37]:
```python
import statsmodels.api as sm

X = sm.add_constant(adv[['Web', 'Bus_Stop', 'Newspaper']])
y = adv['Sales']

model1 = sm.OLS(y, X).fit()
print(model1.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.903
Model:                            OLS   Adj. R-squared:                  0.901
Method:                 Least Squares   F-statistic:                     605.4
Date:                Sun, 23 Apr 2023   Prob (F-statistic):           8.13e-99
Time:                        23:01:07   Log-Likelihood:                -383.34
No. Observations:                 200   AIC:                             774.7
Df Residuals:                     196   BIC:                             787.9
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6251      0.308     15.041      0.000       4.019       5.232
Web            0.0544      0.001     39.592      0.000       0.052       0.057
Bus_Stop       0.1070      0.008     12.604      0.000       0.090       0.124
Newspaper      0.0003      0.006      0.058      0.954      -0.011       0.012
==============================================================================
Omnibus:                       16.081   Durbin-Watson:                   2.251
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               27.655
Skew:                          -0.431   Prob(JB):                     9.88e-07
Kurtosis:                       4.605   Cond. No.                         454.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

D(I)

What is the p-value of the F-Statistic for this model? What does this suggest about the model?

The F-statistic for the model is 605.4 and the corresponding p-value is 8.13e-99. This suggests that the model is statistically significant and that at least one of the predictors (web spending, bus stop ad spending, newspaper spending) is significantly related to the outcome variable (sales).

D(II)

What are the p-values for each of the individual predictors used in this model? What does this suggest about these predictors?

The p-values for each of the individual predictors used in this model are:

Web: 0.000 Bus_Stop: 0.000 Newspaper: 0.954 These p-values suggest that Web and Bus_Stop have a statistically significant relationship with Sales, while Newspaper does not. Specifically, the low p-values for Web and Bus_Stop indicate that there is strong evidence to reject the null hypothesis that the coefficients for these variables are equal to zero. In contrast, the high p-value for Newspaper suggests that we do not have enough evidence to reject the null hypothesis for this variable, and that it is not a significant predictor of Sales in this model.

In [38]:
```python
#E
import statsmodels.api as sm

adv['Interaction'] = adv['Web'] * adv['Bus_Stop']

X = adv[['Web', 'Bus_Stop', 'Interaction']]
y = adv['Sales']

X = sm.add_constant(X)

model2 = sm.OLS(y, X).fit()

print(model2.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.914
Model:                            OLS   Adj. R-squared:                  0.913
Method:                 Least Squares   F-statistic:                     694.7
Date:                Sun, 23 Apr 2023   Prob (F-statistic):          3.90e-104
Time:                        23:01:15   Log-Likelihood:                -370.83
No. Observations:                 200   AIC:                             749.7
Df Residuals:                     196   BIC:                             762.9
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           6.1926      0.410     15.101      0.000       5.384       7.001
Web             0.0436      0.002     17.512      0.000       0.039       0.048
Bus_Stop        0.0423      0.015      2.869      0.005       0.013       0.071
Interaction     0.0004   8.67e-05      5.110      0.000       0.000       0.001
==============================================================================
Omnibus:                        8.006   Durbin-Watson:                   2.295
Prob(Omnibus):                  0.018   Jarque-Bera (JB):               10.847
Skew:                          -0.258   Prob(JB):                      0.00441
Kurtosis:                       4.017   Cond. No.                     1.80e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.8e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

E(I)

What do you notice about the p-values for each of these predictors?

The p-values for all three predictors are statistically significant at the 0.05 level, indicating that they are all likely to have a significant impact on the outcome variable (Sales) in this model. The p-value for the interaction term is also statistically significant, suggesting that there is a significant interaction effect between web ad spending and bus stop ad spending on Sales.

E(II)

How does the r-squared of this model compare to the r-squared of a model built to predict sales, but with only bus bus stop spending and web ad spending, but without the interaction term? What does this difference suggest about the inclusion of the interaction? (you will need to generate another model to answer this, but it won't take long)

In [39]:
```python
import statsmodels.api as sm

X = adv[['Web', 'Bus_Stop']]
y = adv['Sales']

X = sm.add_constant(X)

model3 = sm.OLS(y, X).fit()
predictions = model3.predict(X)

print(model3.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.903
Model:                            OLS   Adj. R-squared:                  0.902
Method:                 Least Squares   F-statistic:                     912.7
Date:                Sun, 23 Apr 2023   Prob (F-statistic):          2.39e-100
Time:                        23:01:32   Log-Likelihood:                -383.34
No. Observations:                 200   AIC:                             772.7
Df Residuals:                     197   BIC:                             782.6
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6309      0.290     15.952      0.000       4.058       5.203
Web            0.0544      0.001     39.726      0.000       0.052       0.057
Bus_Stop       0.1072      0.008     13.522      0.000       0.092       0.123
==============================================================================
Omnibus:                       16.227   Durbin-Watson:                   2.252
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               27.973
Skew:                          -0.434   Prob(JB):                     8.43e-07
Kurtosis:                       4.613   Cond. No.                         425.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

E(II)

The R-squared of this model is 0.903.

Comparing this to the R-squared of the model with the interaction term (0.914), we can see that the model with the interaction term has a higher R-squared value, indicating that it explains more of the variance in the data.

This suggests that the interaction term is a significant predictor and adds value to the model. Including the interaction term allows us to capture the combined effect of bus stop and web ad spending on sales, which we would miss if we only included these variables separately.

In [45]:
```python
import numpy as np

# input values
web = 220
bus_stop = 30


# coefficients from model
coef = np.array([4.6309, 0.0544, 0.1072])

# calculate predicted sales
sales_pred = np.dot(coef, np.array([1, web, bus_stop]))

print("Predicted Sales: ", sales_pred)
```

```
Predicted Sales:  19.8149
```

E(IV)

The interaction effect between bus stop and web ad spending suggests that the combination of these two advertising methods has a greater impact on sales than the sum of their individual effects. In other words, the effect of web ad spending on sales depends on the level of bus stop ad spending and vice versa. The predicted sales for a marketer using 220 units of web ad spending and 30 units of bus stop ad spending is 19.8149.

In [49]:
```python
#F
import statsmodels.api as sm

X = adv[['Web', 'Bus_Stop', 'Total_spend']]
X['Web_Total'] = X['Web'] * X['Total_spend']
X['Bus_Stop_Total'] = X['Bus_Stop'] * X['Total_spend']
y = adv['Sales']

X = sm.add_constant(X)
model = sm.OLS(y, X).fit()

print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.933
Model:                            OLS   Adj. R-squared:                  0.931
Method:                 Least Squares   F-statistic:                     536.8
Date:                Mon, 24 Apr 2023   Prob (F-statistic):          1.61e-111
Time:                        00:05:23   Log-Likelihood:                -346.52
No. Observations:                 200   AIC:                             705.0
Df Residuals:                     194   BIC:                             724.8
Df Model:                           5                                         
Covariance Type:            nonrobust                                         
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              5.0375      0.481     10.482      0.000       4.090       5.985
Web                0.0738      0.006     13.033      0.000       0.063       0.085
Bus_Stop          -0.0087      0.018     -0.487      0.627      -0.044       0.026
Total_spend        0.0008      0.006      0.152      0.879      -0.010       0.012
Web_Total         -0.0001   1.44e-05     -6.951      0.000      -0.000   -7.18e-05
Bus_Stop_Total     0.0006   7.52e-05      7.831      0.000       0.000       0.001
==============================================================================
Omnibus:                        2.193   Durbin-Watson:                   2.252
Prob(Omnibus):                  0.334   Jarque-Bera (JB):                1.848
Skew:                          -0.147   Prob(JB):                        0.397
Kurtosis:                       3.369   Cond. No.                     2.38e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.38e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
<ipython-input-49-efbdc6e75555>:5: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Web_Total'] = X['Web'] * X['Total_spend']
<ipython-input-49-efbdc6e75555>:6: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Bus_Stop_Total'] = X['Bus_Stop'] * X['Total_spend']
```

In this model, there is an interaction term between the variables Web and Total_spend, as well as an interaction term between Bus_Stop and Total_spend. The effect of the interaction term between Web and Total_spend is negative, meaning that the effect of Total_spend on Sales decreases as Web spending increases. The effect of the interaction term between Bus_Stop and Total_spend is positive, meaning that the effect of Total_spend on Sales increases as Bus_Stop spending increases. This interaction makes sense as it suggests that the impact of Total_spend on Sales is not the same across different levels of spending on Web and Bus_Stop advertising.

Part III: Wildcard: A Real-World Consulting Opportunity

One possible recommendation is to use an online platform for grading assignments, such as Blackboard or Turnitin. This would allow the lecturer to easily delegate grading to TAs or even peer-reviewers, while maintaining consistency in grading standards. Another option is to implement self-grading or peer grading assignments, where students are given rubrics to grade each other's work, with the lecturer serving as a final arbiter for any discrepancies. This can also provide students with valuable feedback and improve their understanding of course material. Additionally, implementing automated grading tools, such as Grammarly or Turnitin's Feedback Studio, can help reduce the time spent grading assignments, although it may not be appropriate for all types of assignments. Finally, the lecturer can consider reducing the number of assignments or adjusting the workload to make grading more manageable.

Another additional point would be - using a combination of TA grading and peer grading could also be considered, where students grade each other's assignments under the supervision of the TA. This can be a good way to reduce the workload on the lecturer while also encouraging student engagement and feedback.