Name: Vedant Devidas Patil

IST 652 – Scripting for Data Analysis.

MINI PROJECT 2 – NETFLIX STOCK ANALYSIS

Data and Source:

The Netflix stock market semi-structured data can be pulled from online websites and one of the sources that I have used is as mentioned below.

Link:
https://markets.financialcontent.com/stocks/quote/historical?Month=11&Range=12&Year=2022&Symbol=537%3A1413346

The dataset contains the Netflix stock market data from past 1 year on daily basis. The dataset consists of the following columns:

Date: The daily date in string format

Open: The opening price of the stock at that particular day

High: The highest price that the stock went in that entire day

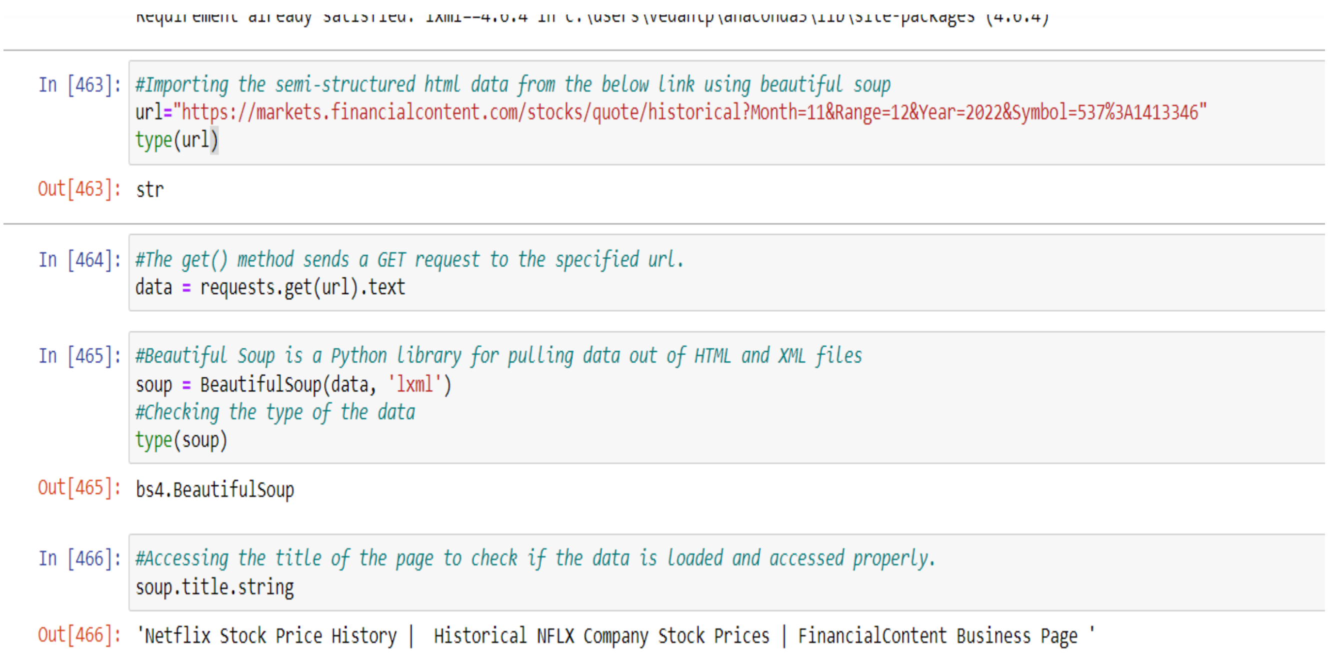
Low: The lowest price that the stock went in that entire day.

Close: The price of the stock when the market closed.

Volume: The number of transactions that are initiated into stock buying and selling.

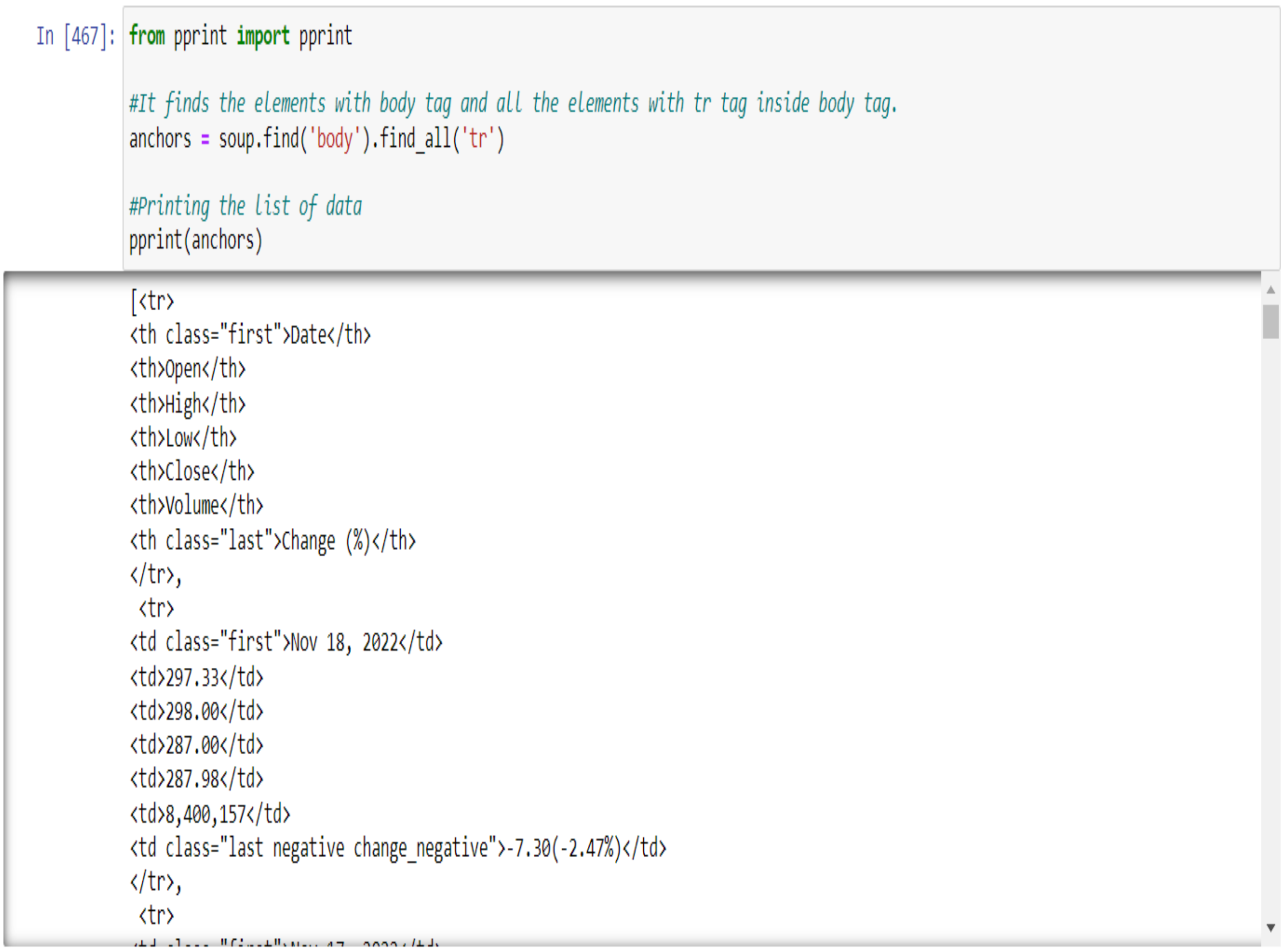
Description of Data Exploration and Data Cleaning:

1) Now, since the data is semi-structured, we'll have to perform web scrapping to pull the data from the online website.



```
Requirement already satisfied: lxml==4.6.4 in c:\users\vedantp\anaconda3\lib\site-packages (4.6.4)

In [463]: #Importing the semi-structured html data from the below link using beautiful soup
          url="https://markets.financialcontent.com/stocks/quote/historical?Month=11&Range=12&Year=2022&Symbol=537%3A1413346"
          type(url)

Out[463]: str

In [464]: #The get() method sends a GET request to the specified url.
          data = requests.get(url).text

In [465]: #Beautiful Soup is a Python library for pulling data out of HTML and XML files
          soup = BeautifulSoup(data, 'lxml')
          #Checking the type of the data
          type(soup)

Out[465]: bs4.BeautifulSoup

In [466]: #Accessing the title of the page to check if the data is loaded and accessed properly.
          soup.title.string

Out[466]: 'Netflix Stock Price History |  Historical NFLX Company Stock Prices | FinancialContent Business Page '
```

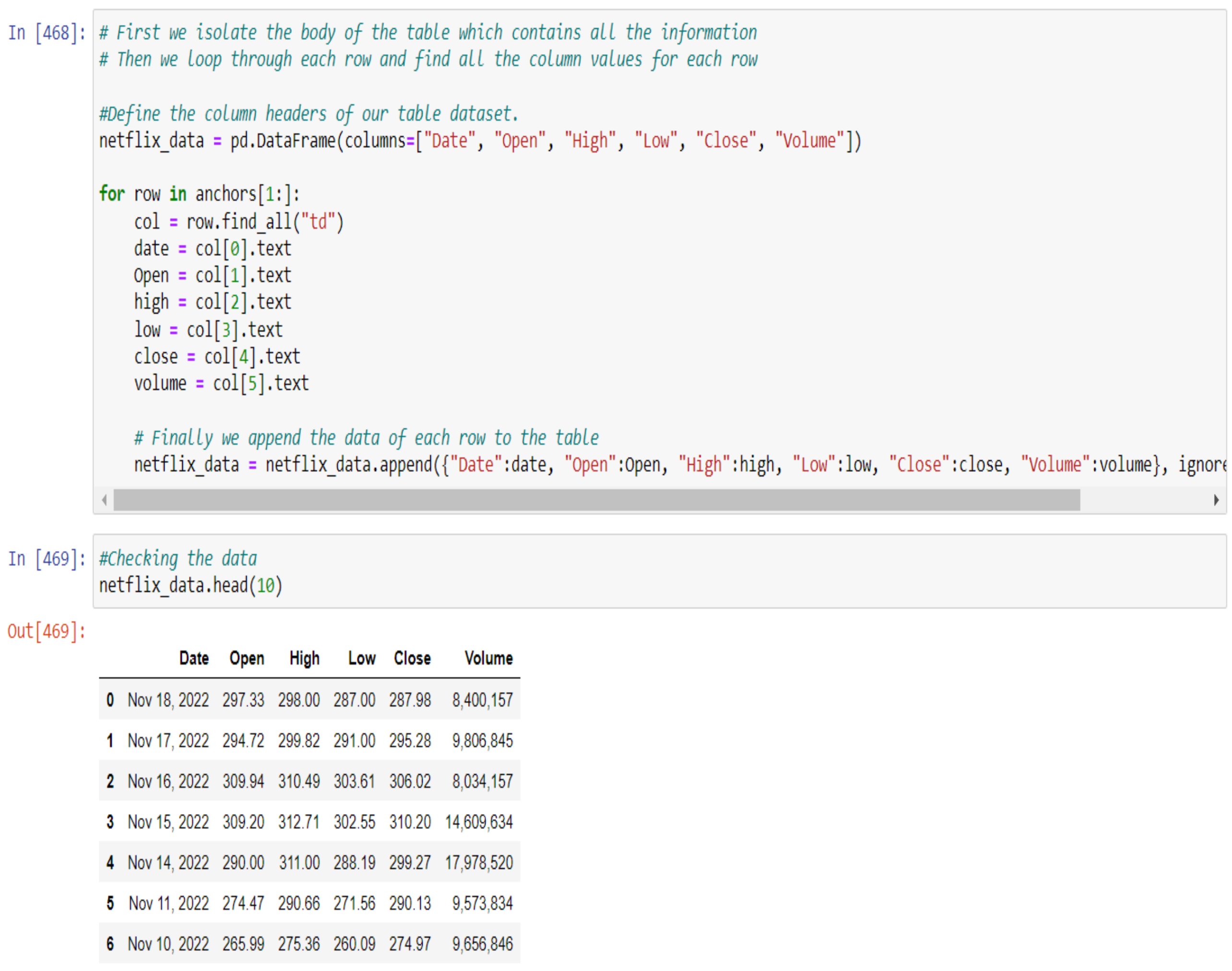
2) The above screenshot shows the steps for importing data using web scrapping technique using Beautiful soup package. The URL is passed into the requests.get() method that sends a GET request to the specified url. Next, the BeautifulSoup() method creates a parse tree for parsed pages that can be used to extract data from HTML.



```
In [467]: from pprint import pprint

          #It finds the elements with body tag and all the elements with tr tag inside body tag.
          anchors = soup.find('body').find_all('tr')

          #Printing the list of data
          pprint(anchors)
```

```
[<tr>
<th class="first">Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th class="last">Change (%)</th>
</tr>,
 <tr>
<td class="first">Nov 18, 2022</td>
<td>297.33</td>
<td>298.00</td>
<td>287.00</td>
<td>287.98</td>
<td>8,400,157</td>
<td class="last negative change_negative">-7.30(-2.47%)</td>
</tr>,
 <tr>
```

3) Now, since we are accessing the table on the page, it is very important to understand the HTML structure of the webpage so that we can use appropriate tags to extract the required information. In the above image, I'm accessing the "body" tag and all the "tr" elements on the page and created a list out of it. The output shows the HTML structure as a list.

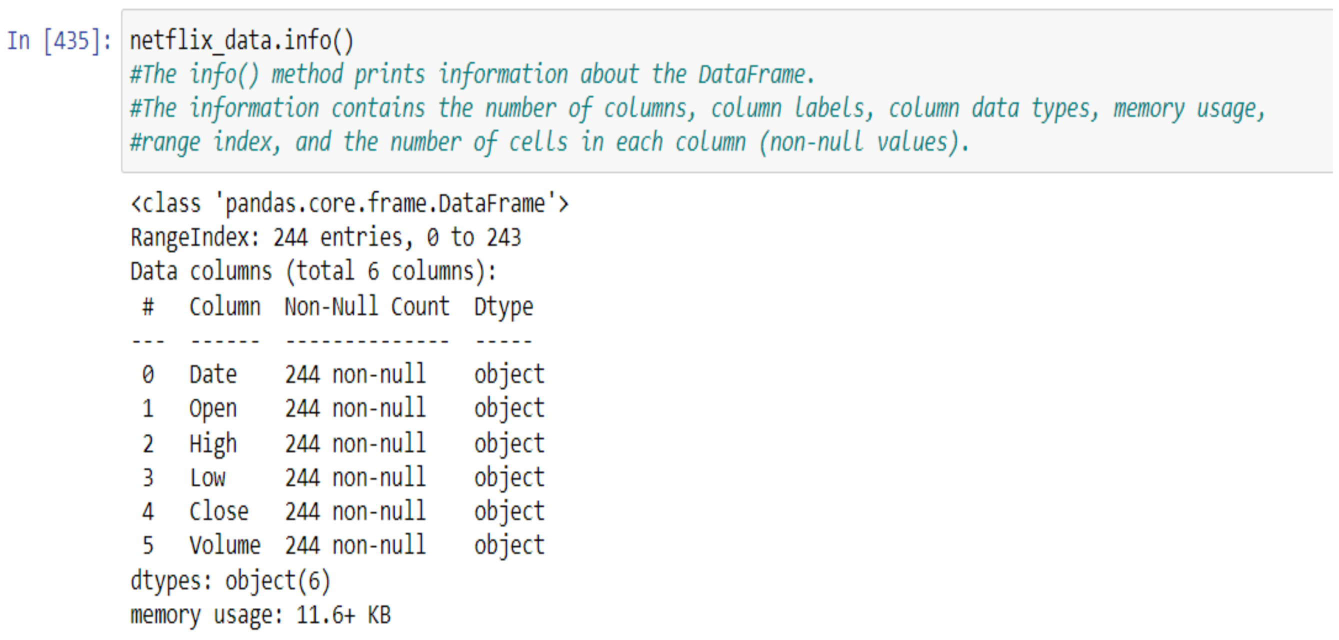


```
In [468]: # First we isolate the body of the table which contains all the information
          # Then we loop through each row and find all the column values for each row

          #Define the column headers of our table dataset.
          netflix_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Volume"])

          for row in anchors[1:]:
              col = row.find_all("td")
              date = col[0].text
              Open = col[1].text
              high = col[2].text
              low = col[3].text
              close = col[4].text
              volume = col[5].text

              # Finally we append the data of each row to the table
              netflix_data = netflix_data.append({"Date":date, "Open":Open, "High":high, "Low":low, "Close":close, "Volume":volume}, ignore
```

```
In [469]: #Checking the data
          netflix_data.head(10)
```

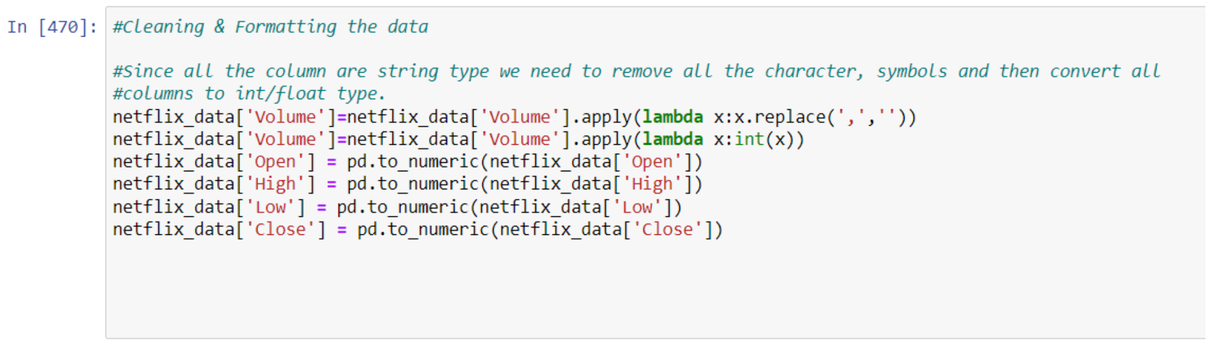
```
Out[469]:
           Date     Open    High     Low   Close      Volume
  0  Nov 18, 2022  297.33  298.00  287.00  287.98   8,400,157
  1  Nov 17, 2022  294.72  299.82  291.00  295.28   9,806,845
  2  Nov 16, 2022  309.94  310.49  303.61  306.02   8,034,157
  3  Nov 15, 2022  309.20  312.71  302.55  310.20  14,609,634
  4  Nov 14, 2022  290.00  311.00  288.19  299.27  17,978,520
  5  Nov 11, 2022  274.47  290.66  271.56  290.13   9,573,834
  6  Nov 10, 2022  265.99  275.36  260.09  274.97   9,656,846
```

4) Further, to create a dataframe from the data present in the HTML structure, a new dataframe netflix_data has been defined by defining column names. The list is traversed using for loop and all the elements are added to the Netflix_data dataframe. The output shows the data from the HTML structure to Python dataframe.

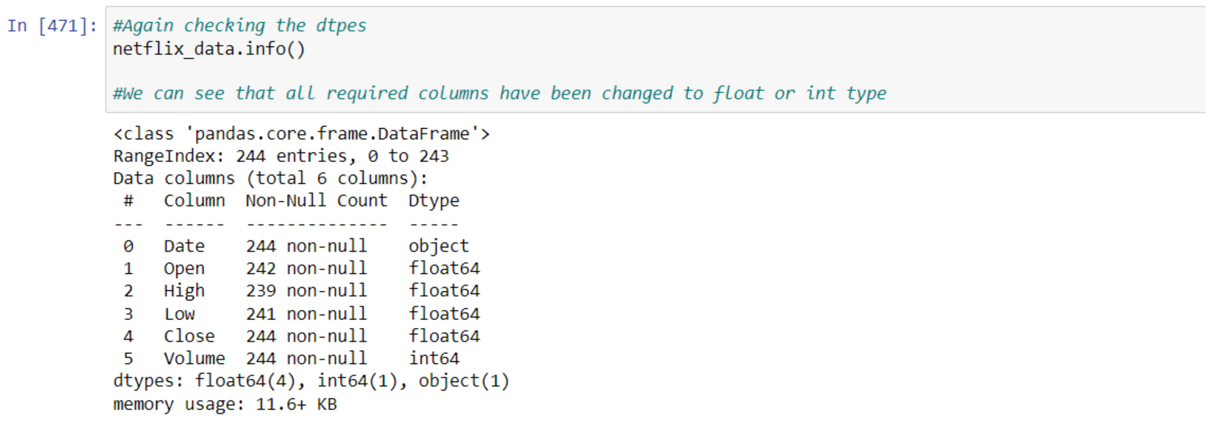


```
In [435]: netflix_data.info()
          #The info() method prints information about the DataFrame.
          #The information contains the number of columns, column labels, column data types, memory usage,
          #range index, and the number of cells in each column (non-null values).

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Date    244 non-null    object
 1   Open    244 non-null    object
 2   High    244 non-null    object
 3   Low     244 non-null    object
 4   Close   244 non-null    object
 5   Volume  244 non-null    object
dtypes: object(6)
memory usage: 11.6+ KB
```

5) The above screenshot shows the info() method that gives an information about all the Columns and datatypes, from which we can infer which column datatypes should be changed according to our requirements. Also, it shows us about how many non-null values are present in each column.



```
In [470]: #Cleaning & Formatting the data

          #Since all the column are string type we need to remove all the character, symbols and then convert all
          #columns to int/float type.
          netflix_data['Volume']=netflix_data['Volume'].apply(lambda x:x.replace(',',''))
          netflix_data['Volume']=netflix_data['Volume'].apply(lambda x:int(x))
          netflix_data['Open'] = pd.to_numeric(netflix_data['Open'])
          netflix_data['High'] = pd.to_numeric(netflix_data['High'])
          netflix_data['Low'] = pd.to_numeric(netflix_data['Low'])
          netflix_data['Close'] = pd.to_numeric(netflix_data['Close'])
```

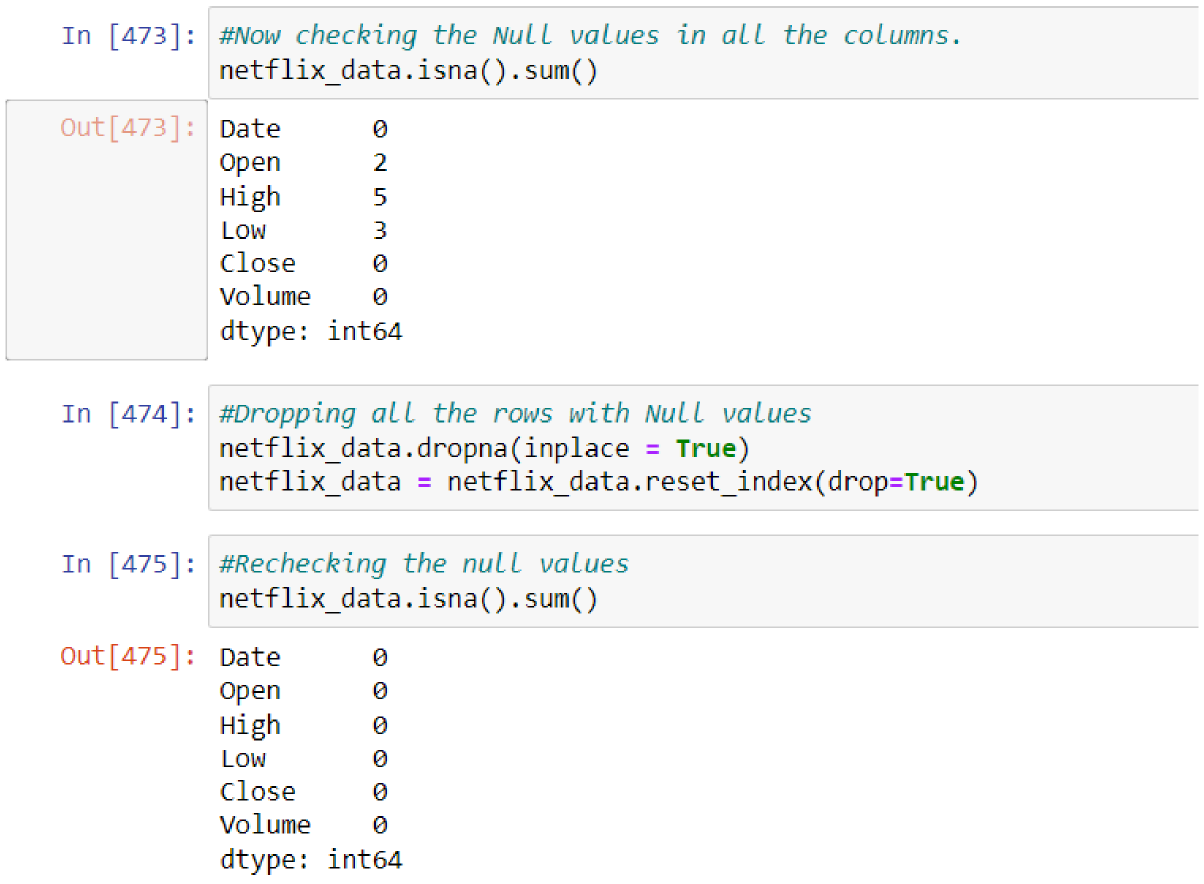


```
In [471]: #Again checking the dtpes
          netflix_data.info()

          #We can see that all required columns have been changed to float or int type

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    244 non-null    object 
 1   Open    242 non-null    float64
 2   High    239 non-null    float64
 3   Low     241 non-null    float64
 4   Close   244 non-null    float64
 5   Volume  244 non-null    int64  
dtypes: float64(4), int64(1), object(1)
memory usage: 11.6+ KB
```

6) In order to perform mathematical operations on the columns and their data, it becomes vital to convert required data from string to int or float datatypes. Also, removing unnecessary information from the data such as symbols, commas, etc becomes important too. The replace()

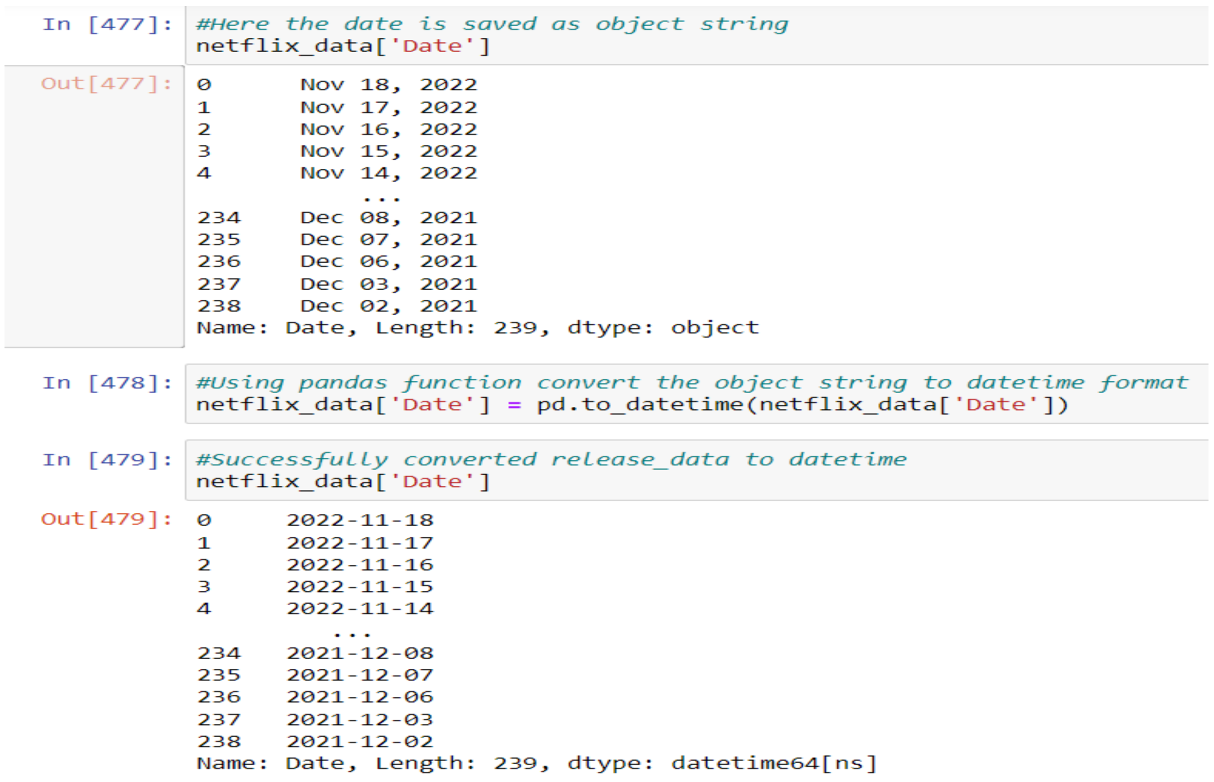
method above is replacing the comma with blank and to_numeric() method converts the datatype of the variables from string to float64 type. The output displays the required expected outcome.

Removing NULL Values:



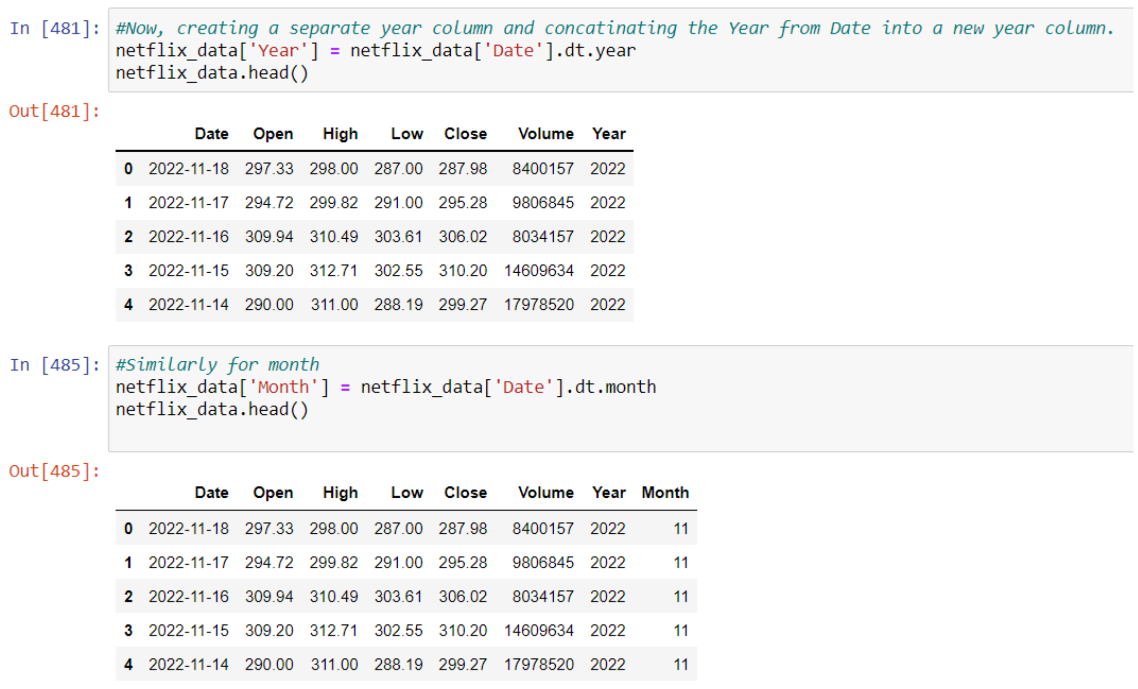
```
In [473]: #Now checking the Null values in all the columns.
          netflix_data.isna().sum()

Out[473]: Date      0
          Open      2
          High      5
          Low       3
          Close     0
          Volume    0
          dtype: int64

In [474]: #Dropping all the rows with Null values
          netflix_data.dropna(inplace = True)
          netflix_data = netflix_data.reset_index(drop=True)

In [475]: #Rechecking the null values
          netflix_data.isna().sum()

Out[475]: Date      0
          Open      0
          High      0
          Low       0
          Close     0
          Volume    0
          dtype: int64
```

7) Data Cleaning also involves cleaning NULL values which include any blank spaces, NA or NAN values that can introduce biased data output. Thus, in order to remove the null values, we first checked how many null values are present in each column. The output shows we have 5 Null values in the ‘High’ column, 3 in the ‘Low’ column, and 2 in the ‘Open’ column. To remove those we used dropna() method to drop values and reser_index() method to reset the index values after dropping the values.



```
In [477]: #Here the date is saved as object string
          netflix_data['Date']

Out[477]: 0       Nov 18, 2022
          1       Nov 17, 2022
          2       Nov 16, 2022
          3       Nov 15, 2022
          4       Nov 14, 2022
                      ...
          234     Dec 08, 2021
          235     Dec 07, 2021
          236     Dec 06, 2021
          237     Dec 03, 2021
          238     Dec 02, 2021
          Name: Date, Length: 239, dtype: object

In [478]: #Using pandas function convert the object string to datetime format
          netflix_data['Date'] = pd.to_datetime(netflix_data['Date'])

In [479]: #Successfully converted release_data to datetime
          netflix_data['Date']

Out[479]: 0      2022-11-18
          1      2022-11-17
          2      2022-11-16
          3      2022-11-15
          4      2022-11-14
                    ...
          234    2021-12-08
          235    2021-12-07
          236    2021-12-06
          237    2021-12-03
          238    2021-12-02
          Name: Date, Length: 239, dtype: datetime64[ns]
```

8) The above code converts the Date (String type) to datetime format so that we can separate the year and month and put that data into separate columns resulting in simplicity to perform mathematical operations.



```
In [481]: #Now, creating a separate year column and concatinating the Year from Date into a new year column.
          netflix_data['Year'] = netflix_data['Date'].dt.year
          netflix_data.head()

Out[481]:
```

	Date	Open	High	Low	Close	Volume	Year
0	2022-11-18	297.33	298.00	287.00	287.98	8400157	2022
1	2022-11-17	294.72	299.82	291.00	295.28	9806845	2022
2	2022-11-16	309.94	310.49	303.61	306.02	8034157	2022
3	2022-11-15	309.20	312.71	302.55	310.20	14609634	2022
4	2022-11-14	290.00	311.00	288.19	299.27	17978520	2022

```
In [485]: #Similarly for month
          netflix_data['Month'] = netflix_data['Date'].dt.month
          netflix_data.head()

Out[485]:
```

	Date	Open	High	Low	Close	Volume	Year	Month
0	2022-11-18	297.33	298.00	287.00	287.98	8400157	2022	11
1	2022-11-17	294.72	299.82	291.00	295.28	9806845	2022	11
2	2022-11-16	309.94	310.49	303.61	306.02	8034157	2022	11
3	2022-11-15	309.20	312.71	302.55	310.20	14609634	2022	11
4	2022-11-14	290.00	311.00	288.19	299.27	17978520	2022	11

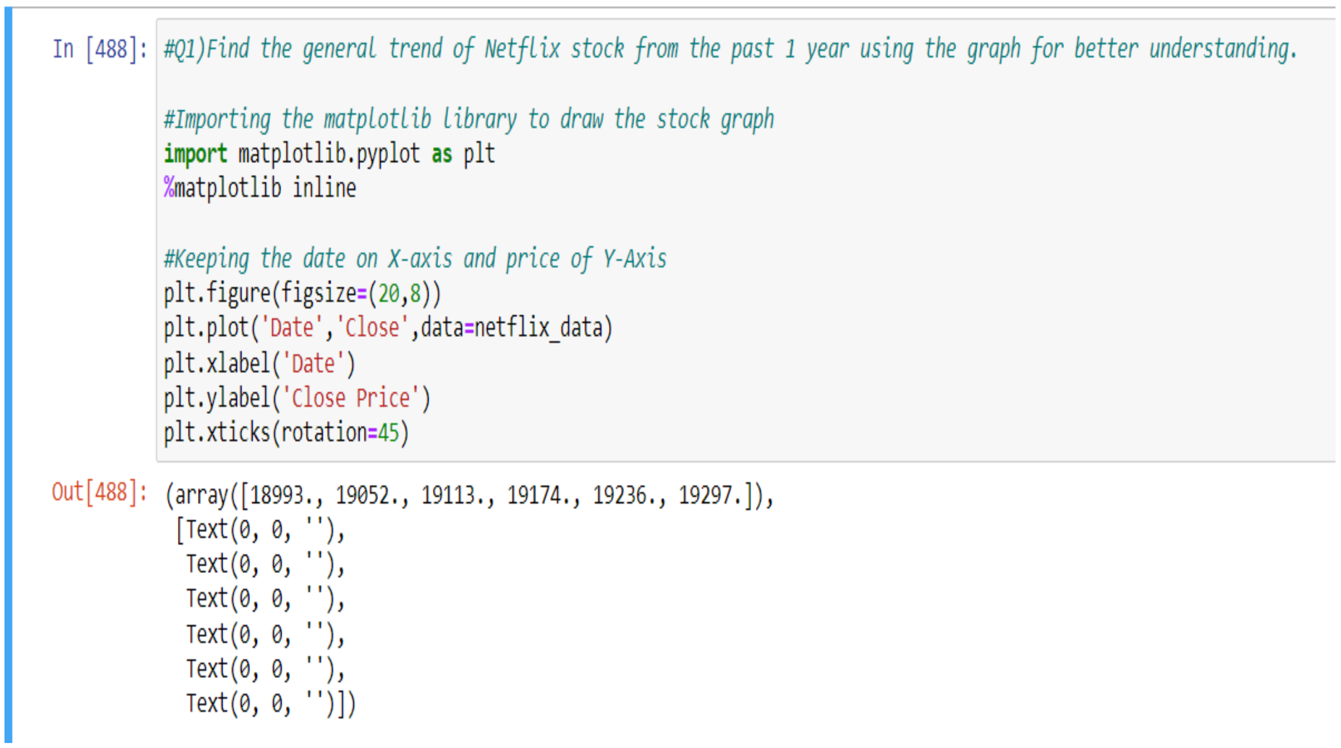
9) The year and month has been concatenated from the Date Column and appended in separate Year and Month Columns.

Q1) Find the general trend of Netflix stock from the past 1 year using the graph for better understanding.

_Here we are finding out the visualization of the past one-year daily stock price trend in order to better understand how Netflix stock performed last year. This will help us to analyze which time period was a bad time for the organization and which was a good time in terms of revenue and profits too.

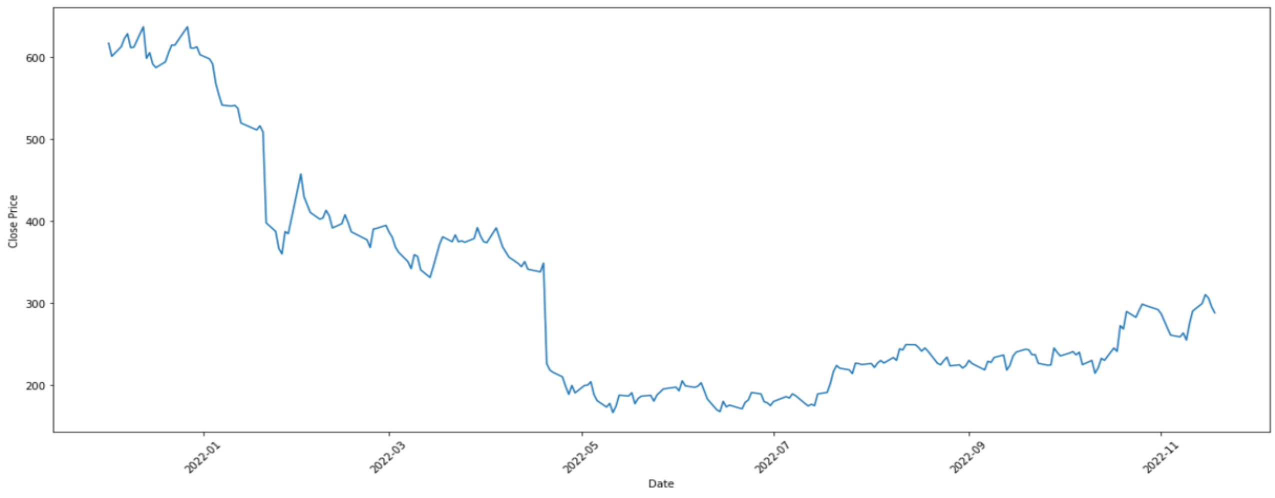
Here, the unit of analysis is Stock price and the comparison factor is the Date from the present date to the last year.

Program & Description & Output:



```
In [488]: #Q1)Find the general trend of Netflix stock from the past 1 year using the graph for better understanding.

          #Importing the matplotlib library to draw the stock graph
          import matplotlib.pyplot as plt
          %matplotlib inline

          #Keeping the date on X-axis and price of Y-Axis
          plt.figure(figsize=(20,8))
          plt.plot('Date','Close',data=netflix_data)
          plt.xlabel('Date')
          plt.ylabel('Close Price')
          plt.xticks(rotation=45)

Out[488]: (array([18993., 19052., 19113., 19174., 19236., 19297.]),
           [Text(0, 0, ''),
            Text(0, 0, ''),
            Text(0, 0, ''),
            Text(0, 0, ''),
            Text(0, 0, ''),
            Text(0, 0, '')])
```

I have used matplotlib package to present the visualization about stock price. Used Date as X-axis and Closing price of the stock as Y-axis, and rotated the label of date on Y-axis by 45 degrees.



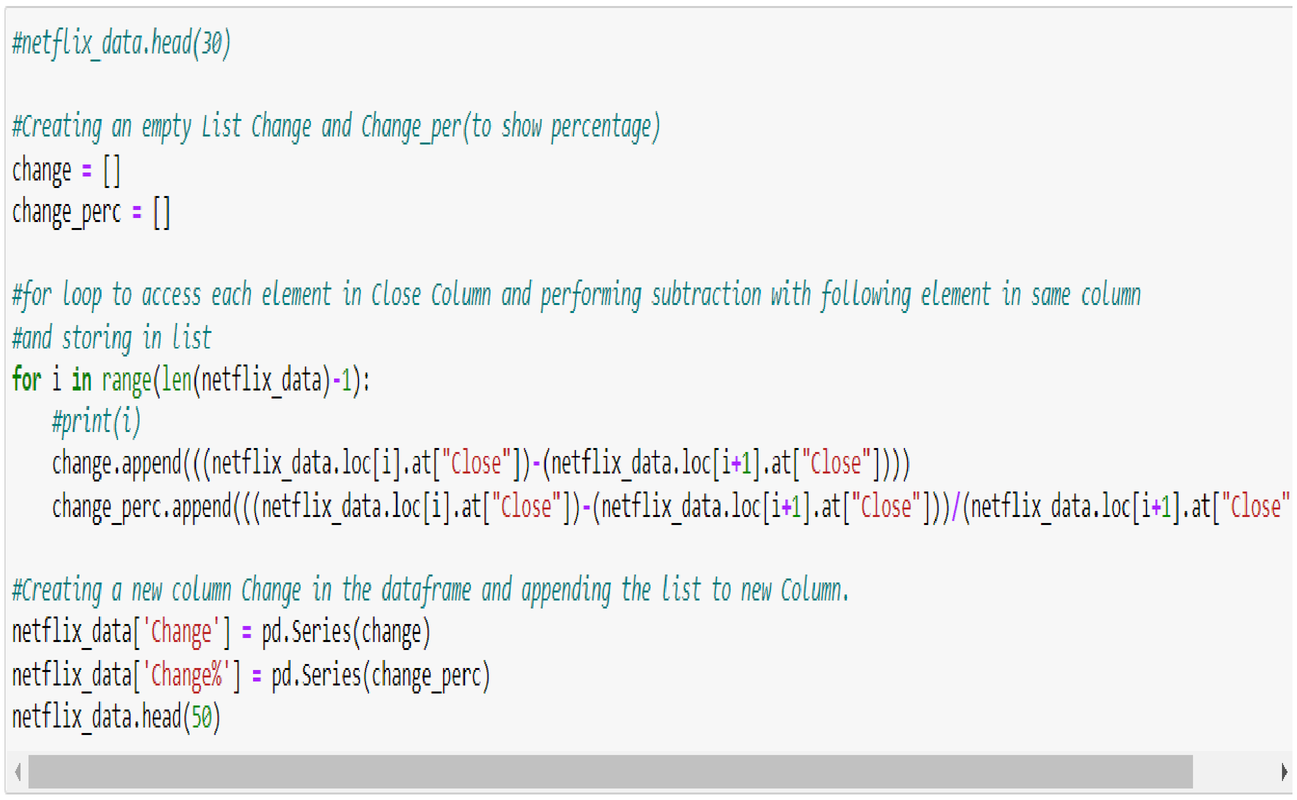
From the Graph, we can see that Netflix stock is performing poorly for the past 1 year. There was a sharp from January 2022 to May 2022, however, it managed to sustain a certain level but did not show an exponential increase and is still below what it was last year.

Q2) Calculate whether the Netflix stock ended on a positive note or negative note each day for the entire year.

Here we are determining whether the Netflix stock ended on the higher side or lower side by comparing the closing price of the current day with the closing price of the past day. We will find the difference and if it is negative the stock ended on a lower note than yesterday and if Change is positive, that means the stock performed well on that day.

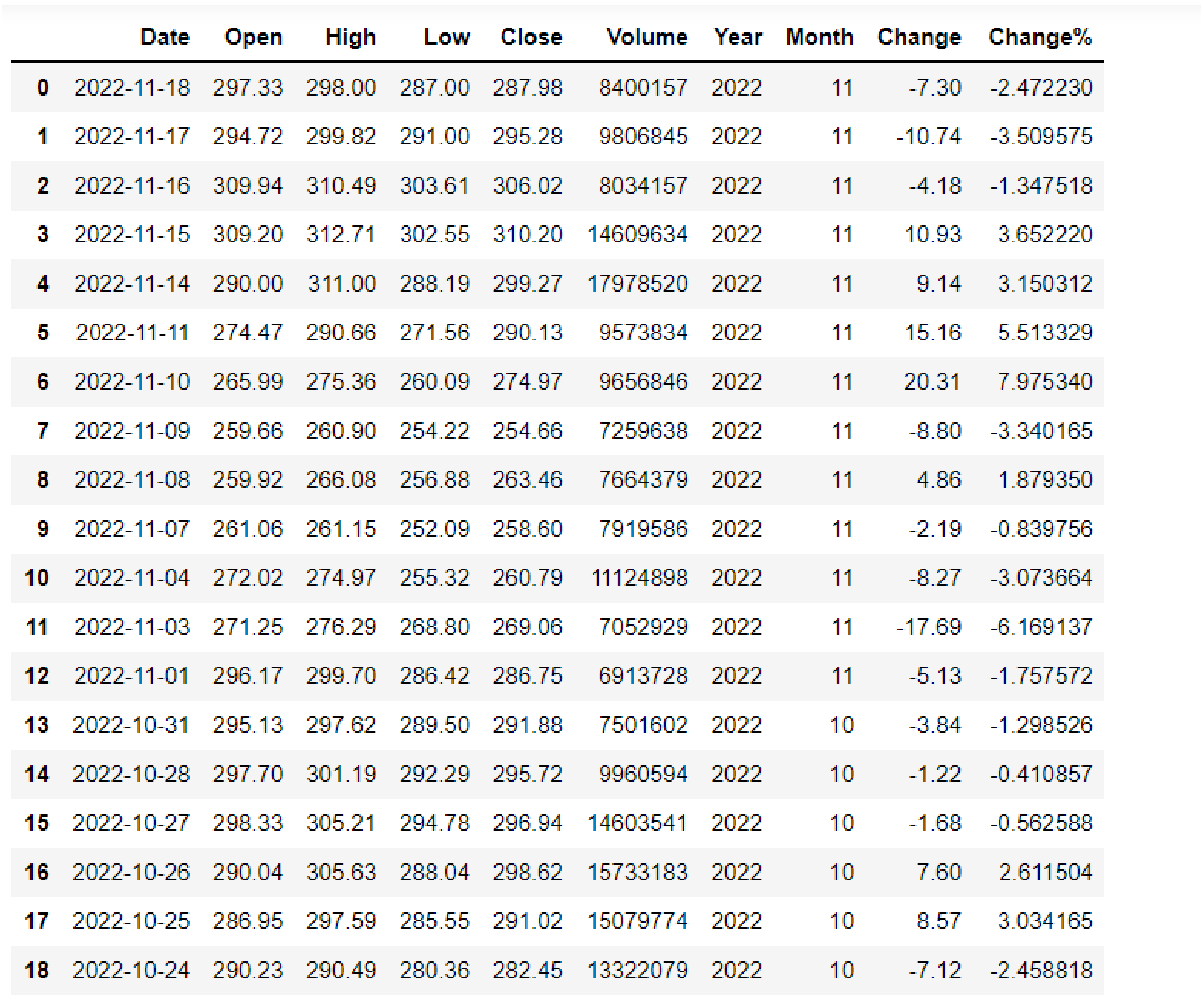
Here, the unit of analysis is the Closing price of the stock on each day and the comparison factor is the Date.

Program & Description & Output:



```
#netflix_data.head(30)

#Creating an empty List Change and Change_per(to show percentage)
change = []
change_perc = []

#for loop to access each element in Close Column and performing subtraction with following element in same column
#and storing in list
for i in range(len(netflix_data)-1):
    #print(i)
    change.append(((netflix_data.loc[i].at["Close"])-(netflix_data.loc[i+1].at["Close"])))
    change_perc.append((((netflix_data.loc[i].at["Close"])-(netflix_data.loc[i+1].at["Close"]))/(netflix_data.loc[i+1].at["Close"

#Creating a new column Change in the dataframe and appending the list to new Column.
netflix_data['Change'] = pd.Series(change)
netflix_data['Change%'] = pd.Series(change_perc)
netflix_data.head(50)
```

Output:



	Date	Open	High	Low	Close	Volume	Year	Month	Change	Change%
0	2022-11-18	297.33	298.00	287.00	287.98	8400157	2022	11	-7.30	-2.472230
1	2022-11-17	294.72	299.82	291.00	295.28	9806845	2022	11	-10.74	-3.509575
2	2022-11-16	309.94	310.49	303.61	306.02	8034157	2022	11	-4.18	-1.347518
3	2022-11-15	309.20	312.71	302.55	310.20	14609634	2022	11	10.93	3.652220
4	2022-11-14	290.00	311.00	288.19	299.27	17978520	2022	11	9.14	3.150312
5	2022-11-11	274.47	290.66	271.56	290.13	9573834	2022	11	15.16	5.513329
6	2022-11-10	265.99	275.36	260.09	274.97	9656846	2022	11	20.31	7.975340
7	2022-11-09	259.66	260.90	254.22	254.66	7259638	2022	11	-8.80	-3.340165
8	2022-11-08	259.92	266.08	256.88	263.46	7664379	2022	11	4.86	1.879350
9	2022-11-07	261.06	261.15	252.09	258.60	7919586	2022	11	-2.19	-0.839756
10	2022-11-04	272.02	274.97	255.32	260.79	11124898	2022	11	-8.27	-3.073664
11	2022-11-03	271.25	276.29	268.80	269.06	7052929	2022	11	-17.69	-6.169137
12	2022-11-01	296.17	299.70	286.42	286.75	6913728	2022	11	-5.13	-1.757572
13	2022-10-31	295.13	297.62	289.50	291.88	7501602	2022	10	-3.84	-1.298526
14	2022-10-28	297.70	301.19	292.29	295.72	9960594	2022	10	-1.22	-0.410857
15	2022-10-27	298.33	305.21	294.78	296.94	14603541	2022	10	-1.68	-0.562588
16	2022-10-26	290.04	305.63	288.04	298.62	15733183	2022	10	7.60	2.611504
17	2022-10-25	286.95	297.59	285.55	291.02	15079774	2022	10	8.57	3.034165
18	2022-10-24	290.23	290.49	280.36	282.45	13322079	2022	10	-7.12	-2.458818

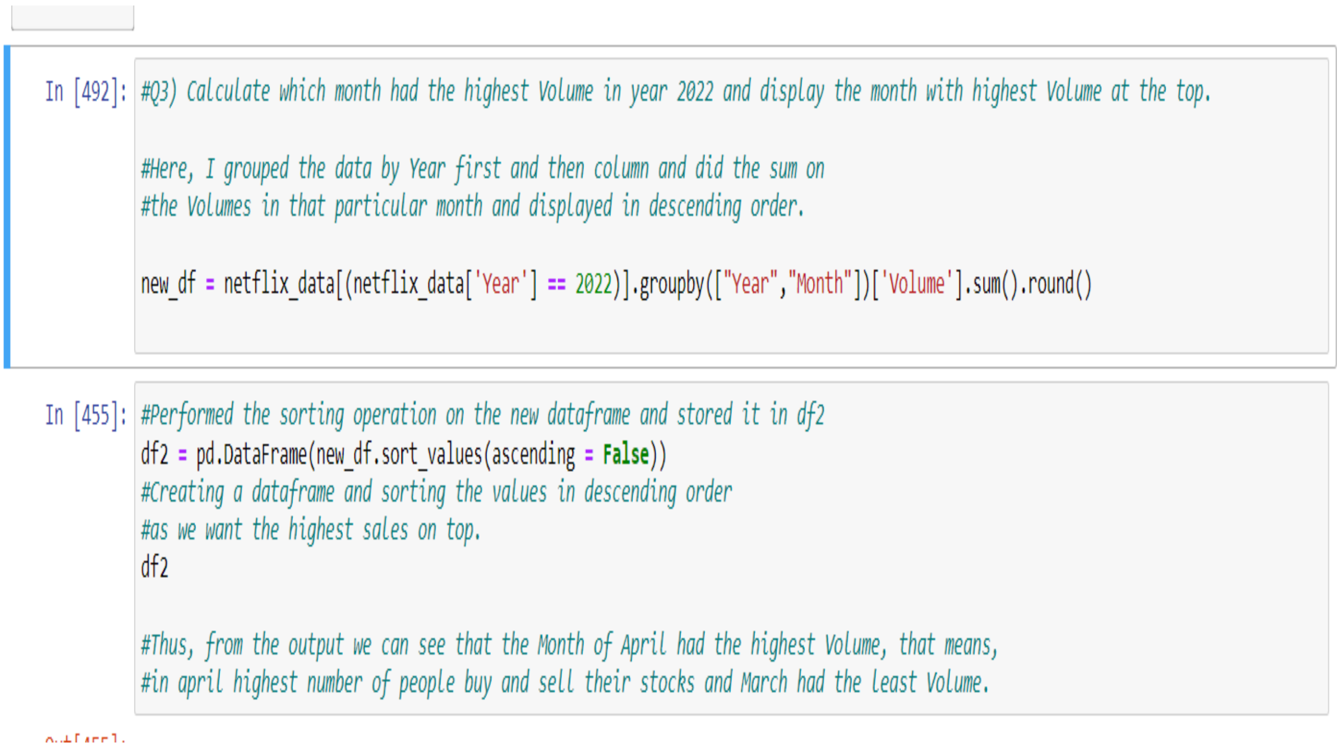
The Change column suggests how much the stock price has fallen on that particular day when compared to the previous day. This helps us to find which days can be profitable and are good for trading and which days can result in losses. Further, we can also find the comparison between the Volume and Change% on that day to get an insight into how Volume could play an important role in stock price.

Q3) Calculate which month had the highest Volume in the year 2022 and display the month with the highest Volume at the top.

Here, I grouped the data by Year first and then by column and did the sum on the Volumes in that particular month and displayed it in descending order.

Here, the unit of analysis is Volume and the comparison factor are Year and Month.

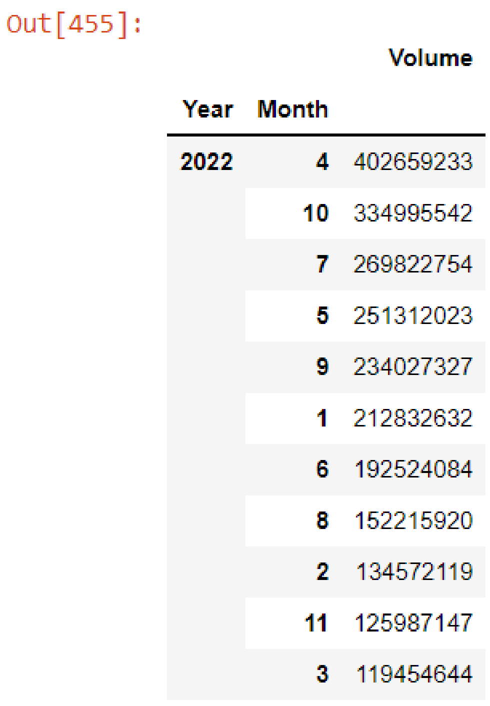
Code and Description:



```
In [492]: #Q3) Calculate which month had the highest Volume in year 2022 and display the month with highest Volume at the top.

          #Here, I grouped the data by Year first and then column and did the sum on
          #the Volumes in that particular month and displayed in descending order.

          new_df = netflix_data[(netflix_data['Year'] == 2022)].groupby(["Year","Month"])['Volume'].sum().round()
```

```
In [455]: #Performed the sorting operation on the new dataframe and stored it in df2
          df2 = pd.DataFrame(new_df.sort_values(ascending = False))
          #Creating a dataframe and sorting the values in descending order
          #as we want the highest sales on top.
          df2

          #Thus, from the output we can see that the Month of April had the highest Volume, that means,
          #in april highest number of people buy and sell their stocks and March had the least Volume.
```

Output:



```
Out[455]:
                    Volume
Year  Month
2022      4      402659233
         10      334995542
          7      269822754
          5      251312023
          9      234027327
          1      212832632
          6      192524084
          8      152215920
          2      134572119
         11      125987147
          3      119454644
```

Therefore, from the output, we can infer that the Volume in the Month of April was the maximum whereas in the Month of March it was the least. Further, we can use this information to analyze what factor in April was influencing the high Volumes and get an idea if these high

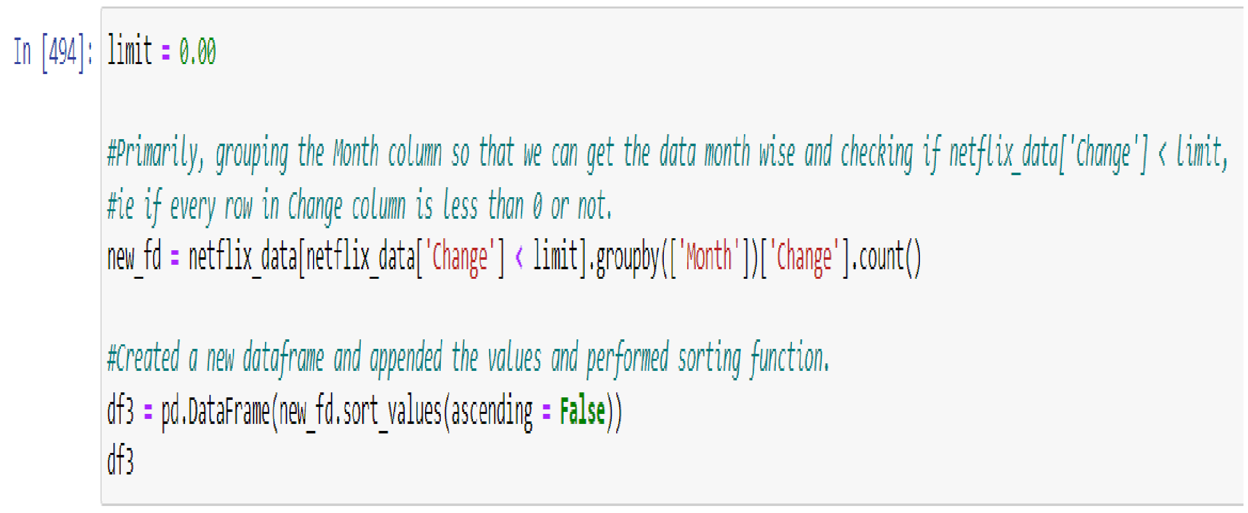
Volume could land a good profit to the investors. If so, the investors could use those external factors and get an idea if that month will have a high volume or not and can initiate a trade accordingly.

Q4) Find out which month performed the worst in the last 12 months(ie Dec 2021 - Present),

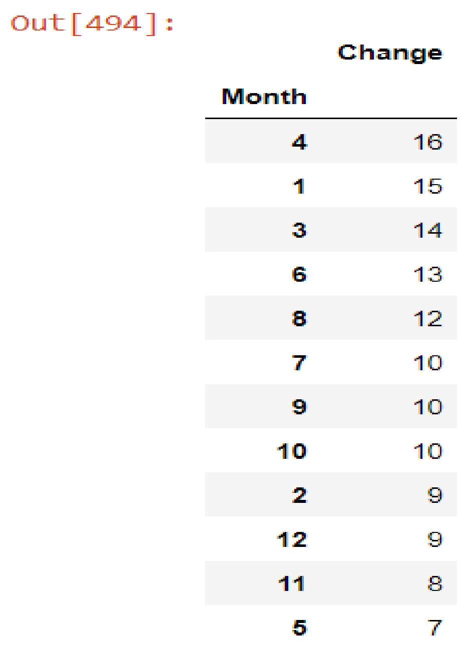
Here, we are analyzing how many days in a particular month the stock performed badly and ended in Red. To achieve this, I utilized the data frame created in Q2 and used the ‘change’ column. We are finding the count of all the days in each month where the stock ended in Red or had a 'change' value as negative this will show us which month was bad for Netflix stock.

Here, the unit of analysis is Change factor and the comparison factor is Month.

Program & Description



```
In [494]: limit = 0.00

          #Primarily, grouping the Month column so that we can get the data month wise and checking if netflix_data['Change'] < limit,
          #ie if every row in Change column is less than 0 or not.
          new_fd = netflix_data[netflix_data['Change'] < limit].groupby(['Month'])['Change'].count()

          #Created a new dataframe and appended the values and performed sorting function.
          df3 = pd.DataFrame(new_fd.sort_values(ascending = False))
          df3
```



Out[494]:

	Change
Month	
4	16
1	15
3	14
6	13
8	12
7	10
9	10
10	10
2	9
12	9
11	8
5	7

The output clearly states that April month had 16 days wherein the stock closed in Red or negative. Also, this was the same month where we had the maximum Volume, as shown in Q3 output. Therefore, we can conclude that April month was the month when maximum people were bearish on Netflix stock.