

## QUERY 1-

1.1 Import the csv file of the stock you have been allotted using 'pd.read\_csv()' function into a dataframe. Shares of a company can be offered in more than one category. The category of a stock is indicated in the 'Series' column. If the csv file has data on more than one category, the 'Date' column will have repeating values. To avoid repetitions in the date, remove all the rows where 'Series' column is NOT 'EQ'. Analyze and understand each column properly. You'd find the head(), tail() and describe() functions to be immensely useful for exploration. You're free to carry out any other exploration of your own.

```
In [1]: import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: data=pd.read_csv('BAJFINANCE.csv')
```

```
In [3]: data.head()
```

Out[3]:

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty	T
0	BAJFINANCE	EQ	15-May-2017	1314.75	1323.0	1348.5	1317.65	1332.0	1332.95	1335.96	634195	8.472579e+08	16701	188623	
1	BAJFINANCE	EQ	16-May-2017	1332.95	1337.4	1358.8	1327.20	1346.0	1347.75	1344.96	788530	1.060542e+09	22552	274106	
2	BAJFINANCE	EQ	17-May-2017	1347.75	1350.0	1374.0	1297.50	1320.0	1324.80	1343.51	2408302	3.235570e+09	67508	468034	
3	BAJFINANCE	EQ	18-May-2017	1324.80	1299.0	1343.0	1296.05	1310.7	1314.55	1325.97	1210985	1.605728e+09	34422	305579	
4	BAJFINANCE	EQ	19-May-2017	1314.55	1324.0	1333.9	1265.55	1288.0	1289.15	1295.81	1167010	1.512226e+09	32394	346261	

In [4]: data.tail()

Out[4]:

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty
490	BAJFINANCE	EQ	07- May- 2019	3034.30	3052.9	3069.80	3007.6	3023.0	3017.05	3041.32	970264	2.950886e+09	56586	233523
491	BAJFINANCE	EQ	08- May- 2019	3017.05	3012.0	3017.00	2900.0	2910.0	2921.30	2969.30	1155023	3.429605e+09	70959	375292
492	BAJFINANCE	EQ	09- May- 2019	2921.30	2900.0	2991.80	2885.0	2969.0	2971.35	2951.93	1745234	5.151803e+09	92225	369765
493	BAJFINANCE	EQ	10- May- 2019	2971.35	2970.1	2996.00	2900.0	2922.0	2922.85	2929.29	1630089	4.775006e+09	84565	296922
494	BAJFINANCE	EQ	13- May- 2019	2922.85	2929.9	2957.95	2906.0	2935.0	2931.85	2932.66	1356229	3.977358e+09	61078	371085

```
In [5]: data.Date.value_counts()
```

```
Out[5]: 25-Mar-2019    2
        20-Nov-2017    1
        08-Mar-2018    1
        31-Jan-2019    1
        21-Dec-2017    1
        05-Sep-2017    1
        31-Jan-2018    1
        13-Jun-2018    1
        22-Feb-2019    1
        02-Feb-2018    1
        31-Jul-2017    1
        06-Jul-2017    1
        17-May-2018    1
        22-Dec-2017    1
        29-Oct-2018    1
        24-Sep-2018    1
        22-Mar-2019    1
        16-Apr-2019    1
        21-May-2018    1
        24-Oct-2018    1
        15-Apr-2019    1
        14-Sep-2017    1
        13-Jun-2017    1
        28-May-2018    1
        30-Jan-2019    1
        29-Jan-2018    1
        08-Mar-2019    1
        05-Feb-2019    1
        08-May-2019    1
        07-May-2019    1
        ..
        28-Feb-2019    1
        12-Sep-2018    1
        20-Jun-2017    1
        28-Jan-2019    1
        31-May-2017    1
        26-Oct-2018    1
        07-Sep-2017    1
        07-Dec-2017    1
        12-Jul-2018    1
        29-May-2017    1
```

```

22-Nov-2018    1
15-Mar-2018    1
23-Apr-2019    1
08-Jun-2018    1
29-Nov-2017    1
17-Jul-2018    1
11-Jan-2019    1
11-Aug-2017    1
04-May-2018    1
13-Oct-2017    1
05-Dec-2018    1
20-Feb-2019    1
01-Jun-2017    1
30-Oct-2018    1
01-Oct-2018    1
06-Nov-2018    1
04-Sep-2017    1
20-Aug-2018    1
19-Mar-2019    1
05-Jun-2018    1

```

Name: Date, Length: 494, dtype: int64

In [6]: data[data.Date=='25-Mar-2019']

Out[6]:

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Delivera (
462	BAJFINANCE	BL	25- Mar- 2019	329.30	2836.65	2836.65	2836.65	2836.65	2836.65	2836.65	50175	1.423289e+08	1	501
463	BAJFINANCE	EQ	25- Mar- 2019	2836.65	2804.00	2862.00	2801.00	2851.00	2849.80	2838.58	1066757	3.028074e+09	53129	2805

```
In [7]: data.Series.value_counts()
```

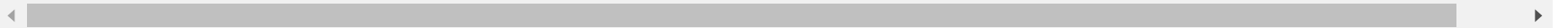
```
Out[7]: EQ      494  
        BL        1  
        Name: Series, dtype: int64
```

```
In [8]: data=data[data.Series=='EQ']  
        data.reset_index(inplace=True, drop=True)
```

```
In [9]: data.head(5)
```

```
Out[9]:
```

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty	T
0	BAJFINANCE	EQ	15-May-2017	1314.75	1323.0	1348.5	1317.65	1332.0	1332.95	1335.96	634195	8.472579e+08	16701	188623	
1	BAJFINANCE	EQ	16-May-2017	1332.95	1337.4	1358.8	1327.20	1346.0	1347.75	1344.96	788530	1.060542e+09	22552	274106	
2	BAJFINANCE	EQ	17-May-2017	1347.75	1350.0	1374.0	1297.50	1320.0	1324.80	1343.51	2408302	3.235570e+09	67508	468034	
3	BAJFINANCE	EQ	18-May-2017	1324.80	1299.0	1343.0	1296.05	1310.7	1314.55	1325.97	1210985	1.605728e+09	34422	305579	
4	BAJFINANCE	EQ	19-May-2017	1314.55	1324.0	1333.9	1265.55	1288.0	1289.15	1295.81	1167010	1.512226e+09	32394	346261	



```
In [10]: data.tail(5)
```

```
Out[10]:
```

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty
489	BAJFINANCE	EQ	07-May-2019	3034.30	3052.9	3069.80	3007.6	3023.0	3017.05	3041.32	970264	2.950886e+09	56586	233523
490	BAJFINANCE	EQ	08-May-2019	3017.05	3012.0	3017.00	2900.0	2910.0	2921.30	2969.30	1155023	3.429605e+09	70959	375292
491	BAJFINANCE	EQ	09-May-2019	2921.30	2900.0	2991.80	2885.0	2969.0	2971.35	2951.93	1745234	5.151803e+09	92225	369765
492	BAJFINANCE	EQ	10-May-2019	2971.35	2970.1	2996.00	2900.0	2922.0	2922.85	2929.29	1630089	4.775006e+09	84565	296922
493	BAJFINANCE	EQ	13-May-2019	2922.85	2929.9	2957.95	2906.0	2935.0	2931.85	2932.66	1356229	3.977358e+09	61078	371085

```
In [11]: data.columns
```

```
Out[11]: Index(['Symbol', 'Series', 'Date', 'Prev Close', 'Open Price', 'High Price',  
               'Low Price', 'Last Price', 'Close Price', 'Average Price',  
               'Total Traded Quantity', 'Turnover', 'No. of Trades', 'Deliverable Qty',  
               '% Dly Qt to Traded Qty'],  
              dtype='object')
```

```
In [12]: data.shape
```

```
Out[12]: (494, 15)
```

```
In [13]: data.describe()
```

Out[13]:

	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of T
<b>count</b>	494.000000	494.000000	494.000000	494.000000	494.000000	494.000000	494.000000	4.940000e+02	4.940000e+02	494.0
<b>mean</b>	2120.230466	2124.776113	2153.414879	2092.138462	2123.693117	2123.503947	2123.734453	1.406506e+06	3.103248e+09	57442.9
<b>std</b>	492.172335	492.939290	497.783073	486.822543	492.355111	492.181884	492.179360	1.223566e+06	2.868243e+09	44617.8
<b>min</b>	1224.350000	1230.000000	1254.900000	1190.550000	1225.000000	1224.350000	1231.010000	1.500180e+05	2.115608e+08	3910.0
<b>25%</b>	1724.900000	1726.250000	1747.137500	1705.075000	1725.000000	1725.800000	1724.342500	7.325970e+05	1.349683e+09	31587.0
<b>50%</b>	1943.775000	1952.550000	1967.375000	1917.000000	1949.375000	1947.525000	1938.615000	1.053290e+06	2.340650e+09	46966.5
<b>75%</b>	2560.775000	2560.000000	2606.712500	2525.775000	2565.037500	2563.950000	2568.597500	1.570271e+06	3.638027e+09	65324.5
<b>max</b>	3131.750000	3130.100000	3165.000000	3102.300000	3123.000000	3131.750000	3135.440000	1.035730e+07	2.310416e+10	420208.0

QUERY 2-

1.2 Calculate the maximum, minimum and mean price for the last 90 days. (price=Closing Price unless stated otherwise)

```
In [14]: data.iloc[-90:,3:10].describe().astype(int)
```

Out[14]:

	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price
<b>count</b>	90	90	90	90	90	90	90
<b>mean</b>	2776	2780	2813	2745	2779	2779	2779
<b>std</b>	201	200	200	203	201	201	202
<b>min</b>	2458	2468	2530	2355	2458	2458	2466
<b>25%</b>	2603	2610	2636	2572	2601	2603	2606
<b>50%</b>	2701	2719	2734	2677	2709	2709	2709
<b>75%</b>	2997	3009	3029	2972	2994	2997	3004
<b>max</b>	3131	3130	3165	3102	3123	3131	3135

QUERY 3-

1.3 Analyse the data types for each column of the dataframe. Pandas knows how to deal with dates in an intelligent manner. But to make use of Pandas functionality for dates, you need to ensure that the column is of type 'datetime64(ns)'. Change the date column from 'object' type to 'datetime64(ns)' for future convenience. See what happens if you subtract the minimum value of the date column from the maximum value.

```
In [15]: data.dtypes
```

```
Out[15]: Symbol          object
Series          object
Date            object
Prev Close      float64
Open Price      float64
High Price      float64
Low Price       float64
Last Price      float64
Close Price     float64
Average Price   float64
Total Traded Quantity  int64
Turnover        float64
No. of Trades   int64
Deliverable Qty int64
% Dly Qt to Traded Qty float64
dtype: object
```

```
In [16]: data['Date_new'] = pd.to_datetime(data['Date'], format='%d-%b-%Y')
```



```
In [17]: data.dtypes
```

```
Out[17]: Symbol                object  
Series                object  
Date                  object  
Prev Close            float64  
Open Price             float64  
High Price            float64  
Low Price             float64  
Last Price            float64  
Close Price           float64  
Average Price         float64  
Total Traded Quantity  int64  
Turnover              float64  
No. of Trades         int64  
Deliverable Qty       int64  
% Dly Qt to Traded Qty float64  
Date_new              datetime64[ns]  
dtype: object
```

In [18]: data.head()

Out[18]:

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty	T
0	BAJFINANCE	EQ	15-May-2017	1314.75	1323.0	1348.5	1317.65	1332.0	1332.95	1335.96	634195	8.472579e+08	16701	188623	
1	BAJFINANCE	EQ	16-May-2017	1332.95	1337.4	1358.8	1327.20	1346.0	1347.75	1344.96	788530	1.060542e+09	22552	274106	
2	BAJFINANCE	EQ	17-May-2017	1347.75	1350.0	1374.0	1297.50	1320.0	1324.80	1343.51	2408302	3.235570e+09	67508	468034	
3	BAJFINANCE	EQ	18-May-2017	1324.80	1299.0	1343.0	1296.05	1310.7	1314.55	1325.97	1210985	1.605728e+09	34422	305579	
4	BAJFINANCE	EQ	19-May-2017	1314.55	1324.0	1333.9	1265.55	1288.0	1289.15	1295.81	1167010	1.512226e+09	32394	346261	



In [ ]:


#### QUERY 4-

1.4 In a separate array , calculate the monthwise VWAP (Volume Weighted Average Price ) of the stock. (  $VWAP = \frac{\text{sum}(\text{price} \times \text{volume})}{\text{sum}(\text{volume})}$  ) To know more about VWAP , visit - VWAP definition {Hint : Create a new dataframe column 'Month'. The values for this column can be derived from the 'Date' column by using appropriate pandas functions. Similarly, create a column 'Year' and initialize it. Then use the 'groupby()' function by month and year. Finally, calculate the vwap value for each month (i.e. for each group created).

In [19]: data.head()

Out[19]:

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price	Close Price	Average Price	Total Traded Quantity	Turnover	No. of Trades	Deliverable Qty	T
0	BAJFINANCE	EQ	15-May-2017	1314.75	1323.0	1348.5	1317.65	1332.0	1332.95	1335.96	634195	8.472579e+08	16701	188623	
1	BAJFINANCE	EQ	16-May-2017	1332.95	1337.4	1358.8	1327.20	1346.0	1347.75	1344.96	788530	1.060542e+09	22552	274106	
2	BAJFINANCE	EQ	17-May-2017	1347.75	1350.0	1374.0	1297.50	1320.0	1324.80	1343.51	2408302	3.235570e+09	67508	468034	
3	BAJFINANCE	EQ	18-May-2017	1324.80	1299.0	1343.0	1296.05	1310.7	1314.55	1325.97	1210985	1.605728e+09	34422	305579	
4	BAJFINANCE	EQ	19-May-2017	1314.55	1324.0	1333.9	1265.55	1288.0	1289.15	1295.81	1167010	1.512226e+09	32394	346261	



In [20]: data['month']=data.Date.str.slice(3)

```
In [21]: vwap=np.zeros(data.month.unique().size)
         for i,j in enumerate (data.month.unique()):
             temp=data[data.month==j]
             vwap[i]=((temp['Total Traded Quantity']*temp['Average Price']).sum())/((temp['Total Traded Quantity']).sum())

VWAP=pd.DataFrame(vwap,data.month.unique())
VWAP.columns=['VWAP']
VWAP
```

Out[21]:

	VWAP
May-2017	1301.059425
Jun-2017	1375.446644
Jul-2017	1563.516279
Aug-2017	1736.693028
Sep-2017	1872.924034
Oct-2017	1853.718973
Nov-2017	1772.439729
Dec-2017	1719.514755
Jan-2018	1727.635061
Feb-2018	1644.963197
Mar-2018	1688.380205
Apr-2018	1901.559415
May-2018	2045.916102
Jun-2018	2256.790981
Jul-2018	2570.553425
Aug-2018	2844.204587
Sep-2018	2450.313201
Oct-2018	2196.878791
Nov-2018	2394.139795
Dec-2018	2529.970008

	VWAP
<b>Jan-2019</b>	2555.242081
<b>Feb-2019</b>	2632.320490
<b>Mar-2019</b>	2856.880035
<b>Apr-2019</b>	3042.798394
<b>May-2019</b>	2998.428990

#### QUERY 5-

1.5 Write a function to calculate the average price over the last N days of the stock price data where N is a user defined parameter. Write a second function to calculate the profit/loss percentage over the last N days. Calculate the average price AND the profit/loss percentages over the course of last - 1 week, 2 weeks, 1 month, 3 months, 6 months and 1 year. {Note : Profit/Loss percentage between N days is the percentage change between the closing prices of the 2 days }

```
In [22]: def meanprice(df,x):
          return df.iloc[-x,:]['Close Price'].mean()
          print(meanprice(data,5))
          np.round(meanprice(data,5),2)
```

2952.88

Out[22]: 2952.88

```
In [23]: def meanprice(df,x):
          return df.iloc[-x,:]['Close Price'].mean()
          print(meanprice(data,10))
          np.round(meanprice(data,10),2)
```

3023.5099999999993

Out[23]: 3023.51

```
In [24]: def meanprice(df,x):  
         return df.iloc[-x:, :][ 'Close Price' ].mean()  
         print(meanprice(data,21))  
         np.round(meanprice(data,21),2)
```

3027.0642857142857

Out[24]: 3027.06

```
In [25]: def meanprice(df,x):  
         return df.iloc[-x:, :][ 'Close Price' ].mean()  
         print(meanprice(data,42))  
         np.round(meanprice(data,42),2)
```

2975.3357142857153

Out[25]: 2975.34

```
In [26]: def meanprice(df,x):  
         return df.iloc[-x:, :][ 'Close Price' ].mean()  
         print(meanprice(data,126))  
         np.round(meanprice(data,126),2)
```

2687.0646825396802

Out[26]: 2687.06

```
In [27]: def meanprice(df,x):  
         return df.iloc[-x:, :][ 'Close Price' ].mean()  
         print(meanprice(data,252))  
         np.round(meanprice(data,252),2)
```

2537.6093253968247

Out[27]: 2537.61

```
In [28]: def profitloss(df,x):  
         print (np.round(100*(df.iloc[-1, :][ 'Close Price' ]-df.iloc[-(x+1), :][ 'Close Price' ])/df.iloc[-(x+1), :][ 'Close Price' ],2))  
         return  
         profitloss(data,5)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> -3.38 %

```
In [29]: def profitloss(df,x):
        print (np,round(100*(df.iloc[-1,:]['Close Price']-df.iloc[-(x+1),:]['Close Price'])/df.iloc[-(x+1),:]['Close Price'],2))
        return
profitloss(data,10)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> -4.84 %

```
In [30]: def profitloss(df,x):
        print (np,round(100*(df.iloc[-1,:]['Close Price']-df.iloc[-(x+1),:]['Close Price'])/df.iloc[-(x+1),:]['Close Price'],2))
        return
profitloss(data,21)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> -3.05 %

```
In [31]: def profitloss(df,x):
        print (np,round(100*(df.iloc[-1,:]['Close Price']-df.iloc[-(x+1),:]['Close Price'])/df.iloc[-(x+1),:]['Close Price'],2))
        return
profitloss(data,42)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> 7.21 %

```
In [32]: def profitloss(df,x):
        print (np,round(100*(df.iloc[-1,:]['Close Price']-df.iloc[-(x+1),:]['Close Price'])/df.iloc[-(x+1),:]['Close Price'],2))
        return
profitloss(data,126)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> 24.26 %

```
In [33]: def profitloss(df,x):
        print (np,round(100*(df.iloc[-1,:]['Close Price']-df.iloc[-(x+1),:]['Close Price'])/df.iloc[-(x+1),:]['Close Price'],2))
        return
profitloss(data,252)
```

<module 'numpy' from 'F:\\\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\\_\_init\_\_.py'> 56.42 %

#### QUERY 6-

1.6 Add a column 'Day\_Perc\_Change' where the values are the daily change in percentages i.e. the percentage change between 2 consecutive day's closing prices. Instead of using the basic mathematical formula for computing the same, use 'pct\_change()' function provided by Pandas for dataframes. You will note that the first entry of the column will have a 'Nan' value. Why does this happen? Either remove the first row, or set the entry to 0 before proceeding.

```
In [34]: data['Day_Perc_Change']=100*data['Close Price'].pct_change()
data.iloc[0,-1]=0
```

```
In [35]: data
```

0	BAJFINANCE	EQ	May-2017	1314.75	1323.00	1348.50	1317.65	1332.00	1332.95	1335.96	634195	8.472579e+08	16701	1
1	BAJFINANCE	EQ	16-May-2017	1332.95	1337.40	1358.80	1327.20	1346.00	1347.75	1344.96	788530	1.060542e+09	22552	2
2	BAJFINANCE	EQ	17-May-2017	1347.75	1350.00	1374.00	1297.50	1320.00	1324.80	1343.51	2408302	3.235570e+09	67508	4
3	BAJFINANCE	EQ	18-May-2017	1324.80	1299.00	1343.00	1296.05	1310.70	1314.55	1325.97	1210985	1.605728e+09	34422	3
4	BAJFINANCE	EQ	19-May-2017	1314.55	1324.00	1333.90	1265.55	1288.00	1289.15	1295.81	1167010	1.512226e+09	32394	3
5	BAJFINANCE	EQ	22-May-2017	1289.15	1299.00	1304.00	1237.35	1240.00	1242.15	1264.99	873873	1.105444e+09	39772	3
			23-											

## QUERY 7-

1.7 Add another column 'Trend' whose values are: 'Slight or No change' for 'Day\_Perc\_Change' in between -0.5 and 0.5 'Slight positive' for 'Day\_Perc\_Change' in between 0.5 and 1 'Slight negative' for 'Day\_Perc\_Change' in between -0.5 and -1 'Positive' for 'Day\_Perc\_Change' in between 1 and 3 'Negative' for 'Day\_Perc\_Change' in between -1 and -3 'Among top gainers' for 'Day\_Perc\_Change' in between 3 and 7 'Among top losers' for 'Day\_Perc\_Change' in between -3 and -7 'Bull run' for 'Day\_Perc\_Change' >7 'Bear drop' for 'Day\_Perc\_Change' <-7



```
In [36]: data['Trend']=0

for i in np.arange(data.Day_Perc_Change.size):
    if(data.Day_Perc_Change[i]<0.5) and (data.Day_Perc_Change[i]>=-0.5):
        data['Trend'][i]='Slight or No Change'
    elif(data.Day_Perc_Change[i]<1) and (data.Day_Perc_Change[i]>=0.5):
        data['Trend'][i]='Slight Positive'
    elif(data.Day_Perc_Change[i]<=-0.5) and (data.Day_Perc_Change[i]>=-1):
        data['Trend'][i]='Slight Negative'
    elif(data.Day_Perc_Change[i]<3) and (data.Day_Perc_Change[i]>=1):
        data['Trend'][i]='Positive'
    elif(data.Day_Perc_Change[i]<=-1) and (data.Day_Perc_Change[i]>=-3):
        data['Trend'][i]='Negative'
    elif(data.Day_Perc_Change[i]<7) and (data.Day_Perc_Change[i]>=3):
        data['Trend'][i]='Among top gainers'
    elif(data.Day_Perc_Change[i]<=-3) and (data.Day_Perc_Change[i]>=-7):
        data['Trend'][i]='Among top losers'
    elif(data.Day_Perc_Change[i]>=7):
        data['Trend'][i]='Breakout Bull'
    elif(data.Day_Perc_Change[i]<=-3):
        data['Trend'][i]='Breakout Bear'
```

#### QUERY 8-

1.8 Find the average and median values of the column 'Total Traded Quantity' for each of the types of 'Trend'. {Hint : use 'groupby()' on the 'Trend' column and then calculate the average and median values of the column 'Total Traded Quantity'}

```
In [37]: data.Trend.value_counts()
```

```
Out[37]: Positive                121
Slight or No Change            120
Negative                       95
Slight Negative                58
Slight Positive                47
Among top gainers              25
Among top losers               23
Breakout Bull                   4
Breakout Bear                   1
Name: Trend, dtype: int64
```

```
In [38]: trend_vol=data.groupby('Trend')['Total Traded Quantity'].agg(['mean','median'])
```

```
In [39]: trend_vol.astype(int)
```

Out[39]:

	mean	median
Trend		
Among top gainers	2448044	1798461
Among top losers	2495378	1835002
Breakout Bear	3470441	3470441
Breakout Bull	7633932	7298615
Negative	1365531	1095372
Positive	1489973	1166483
Slight Negative	959566	796645
Slight Positive	1195705	1049457
Slight or No Change	1002898	832446

QUERY 9-

1.9 SAVE the dataframe with the additional columns computed as a csv file week2.csv.

```
In [40]: data.to_csv('week2.csv',index=False)
```