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('EICHERMOT.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	% C Tra
0	EICHERMOT	EQ	15-May-2017	29178.35	29403.10	29455.00	28831.00	28915.05	28930.60	29015.64	20026	5.810673e+08	8495	8193	4(
1	EICHERMOT	EQ	16-May-2017	28930.60	28920.45	29450.00	28820.00	29365.95	29362.50	29107.91	36189	1.053386e+09	11584	16359	4!
2	EICHERMOT	EQ	17-May-2017	29362.50	29262.00	29300.00	28971.35	28982.05	29093.55	29116.26	18545	5.399610e+08	7918	10611	5'
3	EICHERMOT	EQ	18-May-2017	29093.55	28900.00	28998.85	28213.00	28213.00	28357.30	28521.49	30153	8.600085e+08	11283	15761	5!
4	EICHERMOT	EQ	19-May-2017	28357.30	28698.00	28698.00	27756.00	27915.00	27936.05	28024.04	44251	1.240092e+09	21721	20153	4!

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	% Tr
490	EICHERMOT	EQ	07-May-2019	20349.90	20470.0	20579.90	20270.05	20345.00	20353.20	20366.88	48978	9.975293e+08	12756	19053	
491	EICHERMOT	EQ	08-May-2019	20353.20	20280.0	20470.00	20250.00	20350.00	20336.55	20365.19	43021	8.761307e+08	12698	13056	
492	EICHERMOT	EQ	09-May-2019	20336.55	20315.0	20490.85	20227.60	20390.00	20360.90	20353.58	43718	8.898180e+08	11556	16312	
493	EICHERMOT	EQ	10-May-2019	20360.90	20390.0	20417.55	20301.00	20361.35	20354.00	20346.97	54755	1.114098e+09	11925	25079	
494	EICHERMOT	EQ	13-May-2019	20354.00	20050.0	20121.00	18600.00	18700.00	18751.75	19178.44	342973	6.577686e+09	87269	86209	

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

Out[5]:
```
25-Mar-2019    2
21-Sep-2017    1
11-Dec-2018    1
28-Jul-2017    1
06-Jul-2018    1
08-Mar-2019    1
05-Nov-2018    1
25-Jun-2018    1
08-Sep-2017    1
09-Nov-2018    1
07-Jun-2018    1
18-Dec-2017    1
16-May-2018    1
27-Feb-2018    1
24-Dec-2018    1
20-Feb-2019    1
29-Nov-2017    1
04-Oct-2018    1
26-Oct-2017    1
08-Nov-2017    1
27-Sep-2017    1
04-Jul-2017    1
16-Jan-2018    1
02-Jul-2018    1
06-Feb-2019    1
20-Mar-2018    1
10-Apr-2018    1
25-Jan-2019    1
31-May-2018    1
13-Aug-2018    1
              ..
02-Feb-2018    1
09-Jan-2019    1
15-Sep-2017    1
22-Feb-2018    1
14-Jul-2017    1
26-Mar-2018    1
13-Jun-2017    1
21-Nov-2018    1
01-Jun-2017    1
15-Mar-2019    1
12-Nov-2018    1
15-Dec-2017    1
03-Jul-2017    1
14-Sep-2018    1
```

```
24-Jan-2019    1
06-Sep-2017    1
31-Oct-2017    1
17-Jul-2017    1
14-Sep-2017    1
04-Jul-2018    1
06-Mar-2019    1
04-Dec-2018    1
30-Jun-2017    1
01-Sep-2017    1
23-Apr-2018    1
28-Nov-2018    1
22-Apr-2019    1
10-Jan-2018    1
23-Feb-2018    1
01-Apr-2019    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	Deliverable Qty	T
462	EICHERMOT	BL	25-Mar-2019	249.95	21446.85	21446.85	21446.85	21446.85	21446.85	21446.85	5004	1.073200e+08	1	5004	
463	EICHERMOT	EQ	25-Mar-2019	21446.85	21390.00	21390.00	21050.95	21191.40	21151.35	21146.55	99423	2.102454e+09	24569	48245	

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	% C Tra
0	EICHERMOT	EQ	15-May-2017	29178.35	29403.10	29455.00	28831.00	28915.05	28930.60	29015.64	20026	5.810673e+08	8495	8193	4(
1	EICHERMOT	EQ	16-May-2017	28930.60	28920.45	29450.00	28820.00	29365.95	29362.50	29107.91	36189	1.053386e+09	11584	16359	4!
2	EICHERMOT	EQ	17-May-2017	29362.50	29262.00	29300.00	28971.35	28982.05	29093.55	29116.26	18545	5.399610e+08	7918	10611	5'
3	EICHERMOT	EQ	18-May-2017	29093.55	28900.00	28998.85	28213.00	28213.00	28357.30	28521.49	30153	8.600085e+08	11283	15761	5!
4	EICHERMOT	EQ	19-May-2017	28357.30	28698.00	28698.00	27756.00	27915.00	27936.05	28024.04	44251	1.240092e+09	21721	20153	4!

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	% Tr
489	EICHERMOT	EQ	07-May-2019	20349.90	20470.0	20579.90	20270.05	20345.00	20353.20	20366.88	48978	9.975293e+08	12756	19053	
490	EICHERMOT	EQ	08-May-2019	20353.20	20280.0	20470.00	20250.00	20350.00	20336.55	20365.19	43021	8.761307e+08	12698	13056	
491	EICHERMOT	EQ	09-May-2019	20336.55	20315.0	20490.85	20227.60	20390.00	20360.90	20353.58	43718	8.898180e+08	11556	16312	
492	EICHERMOT	EQ	10-May-2019	20360.90	20390.0	20417.55	20301.00	20361.35	20354.00	20346.97	54755	1.114098e+09	11925	25079	
493	EICHERMOT	EQ	13-May-2019	20354.00	20050.0	20121.00	18600.00	18700.00	18751.75	19178.44	342973	6.577686e+09	87269	86209	

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 Trade
count	494.000000	494.000000	494.000000	494.000000	494.000000	494.000000	494.000000	494.000000	4.940000e+02	494.00000
mean	26983.775607	27025.830263	27328.877935	26642.895445	26953.491296	26962.669130	26978.380040	67862.196356	1.689181e+09	20987.63360
std	3838.978448	3862.087199	3848.221813	3857.188964	3860.120164	3855.515134	3854.897878	62758.208662	1.310023e+09	16248.49469
min	18877.250000	18900.000000	19094.900000	18600.000000	18700.000000	18751.750000	18939.760000	3589.000000	1.131677e+08	1231.00000
25%	23236.837500	23305.000000	23730.225000	22952.287500	23165.725000	23216.962500	23322.735000	30865.500000	8.955348e+08	11488.25000
50%	28166.300000	28235.350000	28508.250000	27884.150000	28160.175000	28159.150000	28168.550000	46515.500000	1.280253e+09	15794.50000
75%	29940.425000	29943.937500	30272.100000	29609.500000	29915.700000	29940.425000	29924.602500	83441.250000	2.080619e+09	24599.00000
max	32861.950000	33399.950000	33480.000000	32468.100000	32849.000000	32861.950000	32975.240000	481497.000000	1.040866e+10	127954.00000

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
count	90	90	90	90	90	90	90
mean	20767	20779	21042	20468	20705	20719	20740
std	891	880	870	854	876	880	846
min	18877	18900	19094	18600	18700	18751	18939
25%	20217	20247	20476	19993	20271	20204	20255
50%	20530	20560	20774	20300	20502	20518	20499
75%	21147	21246	21455	20844	21109	21129	21123
max	23195	23350	23390	23007	23170	23195	23228

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	% Q Tra
0	EICHERMOT	EQ	15-May-2017	29178.35	29403.10	29455.00	28831.00	28915.05	28930.60	29015.64	20026	5.810673e+08	8495	8193	4(
1	EICHERMOT	EQ	16-May-2017	28930.60	28920.45	29450.00	28820.00	29365.95	29362.50	29107.91	36189	1.053386e+09	11584	16359	4!
2	EICHERMOT	EQ	17-May-2017	29362.50	29262.00	29300.00	28971.35	28982.05	29093.55	29116.26	18545	5.399610e+08	7918	10611	5'
3	EICHERMOT	EQ	18-May-2017	29093.55	28900.00	28998.85	28213.00	28213.00	28357.30	28521.49	30153	8.600085e+08	11283	15761	5!
4	EICHERMOT	EQ	19-May-2017	28357.30	28698.00	28698.00	27756.00	27915.00	27936.05	28024.04	44251	1.240092e+09	21721	20153	4!

In [ ]:
```

```

QUERY 4-

1.4 In a separate array , calculate the monthwise VWAP (Volume Weighted Average Price) of the stock. (VWAP = sum(price*volume)/sum(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	% C Tra
0	EICHERMOT	EQ	15-May-2017	29178.35	29403.10	29455.00	28831.00	28915.05	28930.60	29015.64	20026	5.810673e+08	8495	8193	4(
1	EICHERMOT	EQ	16-May-2017	28930.60	28920.45	29450.00	28820.00	29365.95	29362.50	29107.91	36189	1.053386e+09	11584	16359	4
2	EICHERMOT	EQ	17-May-2017	29362.50	29262.00	29300.00	28971.35	28982.05	29093.55	29116.26	18545	5.399610e+08	7918	10611	5
3	EICHERMOT	EQ	18-May-2017	29093.55	28900.00	28998.85	28213.00	28213.00	28357.30	28521.49	30153	8.600085e+08	11283	15761	5
4	EICHERMOT	EQ	19-May-2017	28357.30	28698.00	28698.00	27756.00	27915.00	27936.05	28024.04	44251	1.240092e+09	21721	20153	4

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	28114.232251
Jun-2017	28128.339586
Jul-2017	28531.195465
Aug-2017	31062.755945
Sep-2017	31783.750818
Oct-2017	31547.585796
Nov-2017	30715.931559
Dec-2017	29419.214690
Jan-2018	27713.985491
Feb-2018	27552.492338
Mar-2018	28054.987774
Apr-2018	30309.613464
May-2018	30343.433752
Jun-2018	29314.238397
Jul-2018	27821.948923
Aug-2018	28131.782389
Sep-2018	27160.190187
Oct-2018	22272.419028
Nov-2018	23443.709064
Dec-2018	23170.789043
Jan-2019	20389.272869
Feb-2019	20530.878892
Mar-2019	21627.118898

	VWAP
Apr-2019	20640.422955
May-2019	19818.603538

QUERY 5-

1.5Write 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)
```

```
20031.28
```

Out[22]:
```
20031.28
```

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

```
20212.269999999997
```

Out[23]:
```
20212.27
```

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

```
20553.178571428572
```

Out[24]:
```
20553.18
```

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

```
20990.43690476191
```

Out[25]:
```
20990.44
```

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

```
21482.62499999999
```

Out[26]:
```
21482.62
```

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

```
24468.13174603176
```

Out[27]:
```
24468.13
```

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'],
    return
profitloss(data,5)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -7.85 %
```

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'],
    return
profitloss(data,10)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -7.81 %
```

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'],
    return
profitloss(data,21)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -9.03 %
```

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'],
    return
profitloss(data,42)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -12.87 %
```

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'],
    return
profitloss(data,126)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -16.85 %
```

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'],
    return
profitloss(data,252)
```

```
<module 'numpy' from 'F:\\Users\\Nikita\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'> -37.24 %
```

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
```

Out[35]:

	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
0	EICHERMOT	EQ	15-May-2017	29178.35	29403.10	29455.00	28831.00	28915.05	28930.60	29015.64	20026	5.810673e+08	8495	8193
1	EICHERMOT	EQ	16-May-2017	28930.60	28920.45	29450.00	28820.00	29365.95	29362.50	29107.91	36189	1.053386e+09	11584	16359
2	EICHERMOT	EQ	17-May-2017	29362.50	29262.00	29300.00	28971.35	28982.05	29093.55	29116.26	18545	5.399610e+08	7918	10611
3	EICHERMOT	EQ	18-May-2017	29093.55	28900.00	28998.85	28213.00	28213.00	28357.30	28521.49	30153	8.600085e+08	11283	15761
			19-											

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]:
```
Slight or No Change    138
Negative               122
Positive                93
Slight Negative         53
Slight Positive         48
Among top gainers       22
Among top losers        13
Breakout Bear            3
Breakout Bull            2
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]:

Trend	mean	median
Among top gainers	129801	128967
Among top losers	182270	150490
Breakout Bear	321435	342973
Breakout Bull	300627	300627
Negative	64012	50169
Positive	69132	47279
Slight Negative	55232	35856
Slight Positive	56096	45222
Slight or No Change	49815	42650

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)
```