Made By- Parvej Siddiqui

B.Tech Cse - C

Enrollment-2019-310-127

PROJECT-1 (BTCSE-601)

Topic - Stock Price Prediction Using Linear Regression.

Import the libraries

In [349]:

```
import pandas as pd
import numpy as np
%matplotlib inline
from sklearn import metrics
import matplotlib.pyplot as plt
```

Read the Data

In [303]:

```
dataset=pd.read_csv( r"C:\Users\HP\Downloads\NSE-Tata-Global-Beverages-Limited.csv")
dataset.head()
```

Out[303]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	08-10-2018	208.00	222.25	206.85	216.00	215.15	4642146	10062.83
1	05-10-2018	217.00	218.60	205.90	210.25	209.20	3519515	7407.06
2	04-10-2018	223.50	227.80	216.15	217.25	218.20	1728786	3815.79
3	03-10-2018	230.00	237.50	225.75	226.45	227.60	1708590	3960.27
4	01-10-2018	234.55	234.60	221.05	230.30	230.90	1534749	3486.05

In [304]:

```
dataset['date']=pd.to_datetime(dataset.Date)
dataset.shape
```

Out[304]:

```
(1235, 9)
```

Drop The original date column

In [305]:

```
# drop The original date column
dataset = dataset.drop(['Date'], axis='columns')
dataset.head()
```

Out[305]:

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)	date
0	208.00	222.25	206.85	216.00	215.15	4642146	10062.83	2018-08-10
1	217.00	218.60	205.90	210.25	209.20	3519515	7407.06	2018-05-10
2	223.50	227.80	216.15	217.25	218.20	1728786	3815.79	2018-04-10
3	230.00	237.50	225.75	226.45	227.60	1708590	3960.27	2018-03-10
4	234.55	234.60	221.05	230.30	230.90	1534749	3486.05	2018-01-10

In [268]:

```
dataset.size
```

Out[268]:

```
9880
```

In [226]:

```
dataset.isnull().sum()
```

Out[226]:

```
Open                  0
High                  0
Low                   0
Last                  0
Close                 0
Total Trade Quantity  0
Turnover (Lacs)       0
date                  0
dtype: int64
```

In [227]:

```
dataset.isna().any()
```

Out[227]:

```
Open                   False
High                   False
Low                    False
Last                   False
Close                  False
Total Trade Quantity   False
Turnover (Lacs)        False
date                   False
dtype: bool
```

In [228]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1235 entries, 0 to 1234
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   Open                  1235 non-null   float64       
 1   High                  1235 non-null   float64       
 2   Low                   1235 non-null   float64       
 3   Last                  1235 non-null   float64       
 4   Close                 1235 non-null   float64       
 5   Total Trade Quantity  1235 non-null   int64         
 6   Turnover (Lacs)       1235 non-null   float64       
 7   date                  1235 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(6), int64(1)
memory usage: 77.3 KB
```

Describe the Dataset

In [230]:

```
dataset.describe()
```

Out[230]:

	Open	High	Low	Last	Close	Total Trade Quantity	
count	1235.000000	1235.000000	1235.000000	1235.000000	1235.000000	1.235000e+03	123
mean	168.954858	171.429069	166.402308	168.736356	168.731053	2.604151e+06	484
std	51.499145	52.436761	50.542919	51.587384	51.544928	2.277028e+06	534
min	103.000000	104.600000	100.000000	102.600000	102.650000	1.001800e+05	12
25%	137.550000	138.925000	135.250000	137.175000	137.225000	1.284482e+06	180
50%	151.500000	153.250000	149.500000	151.200000	151.100000	1.964885e+06	306
75%	169.000000	172.325000	166.700000	169.100000	169.500000	3.095788e+06	585
max	327.700000	328.750000	321.650000	325.950000	325.750000	2.919102e+07	5575

In [231]:

```
print(len(dataset))
```

```
1235
```

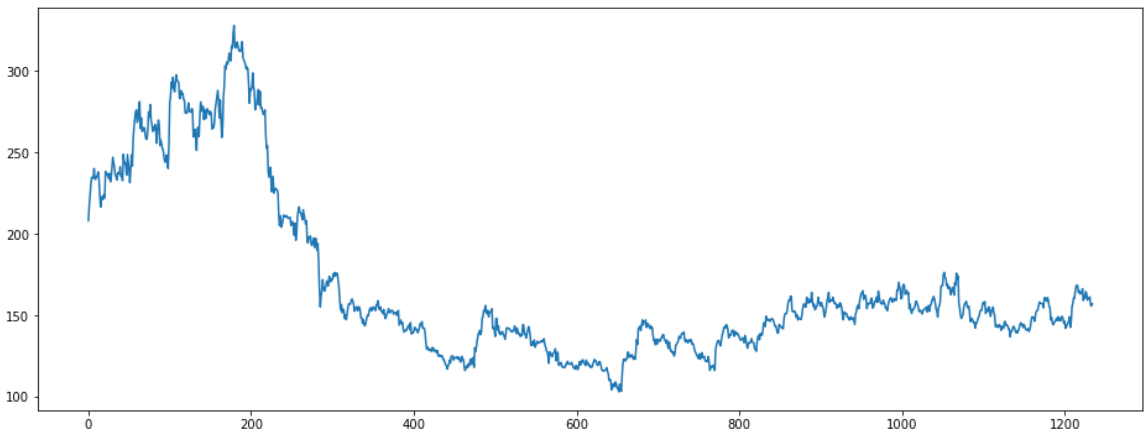
Plot the Data

In [232]:

```
dataset['Open'].plot(figsize=(16,6))
```

Out[232]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2875b31ea00>
```



In [306]:

```
x = dataset[['Open','Low','High','Last']]
y=dataset['Close']
```

In [308]:

```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size=0.2 ,random_st
ate=0)
```

In [315]:

```
x_train.size
```

Out[315]:

```
3952
```

In [269]:

```
x_test.size
```

Out[269]:

```
927
```

Linear Regression

In [309]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix , accuracy_score
```

In [310]:

```
regressor = LinearRegression()
```

In [311]:

```
regressor.fit(x_train,y_train)
```

Out[311]:

```
LinearRegression()
```

In [313]:

```
print(regressor.coef_)
```

```
[-0.06409941  0.08723566  0.09250705  0.88337498]
```

In [317]:

```
print(regressor.intercept_)
```

```
0.13094828296229366
```

In [318]:

```
predicted = regressor.predict(x_test)
```

In [319]:

```
print(x_test)
```

```
        Open     Low    High    Last
1083  152.95  151.05  154.85  153.35
18    221.00  219.10  224.50  223.15
1099  156.50  151.85  157.00  153.00
818   132.95  132.30  133.90  133.20
184   314.65  312.20  319.20  317.45
...      ...     ...     ...     ...
479   138.00  134.50  138.00  134.95
478   134.10  130.80  136.25  131.50
934   148.80  148.10  149.60  149.00
878   153.75  149.00  153.75  149.75
1120  144.15  143.20  144.15  143.70

[247 rows x 4 columns]
```

In [321]:

```
print(predicted)
```

```
[153.29416004 222.97127157 153.02610459 133.20245104 317.15268014
 122.50850585 161.6648302  127.72428918 141.17573772 163.37598322
 173.9338421  118.29087945 119.90494892 157.52292351 233.30522688
 119.89918349 243.6085481  302.80703227 153.39946606 159.44585008
 255.51252327 138.66621702 142.81356706 142.41013604 175.23301485
 126.69453256 152.84845688 132.20834753 118.03499796 109.03706961
 141.95571939 153.72420219 106.85587179 244.03218992 153.88066694
 146.06674464 135.93654428 278.43909678 124.04975839 132.82407432
 151.86629647 122.51731549 232.60004004 144.32689886 121.29347685
 151.91836163 148.1536198  169.02028853 277.91804261 119.40539204
 237.35617726 139.15891217 160.39850489 119.9298837  158.46730685
 137.21538305 155.2006375  155.31219794 140.21598373 125.31014432
 137.46314938 117.44596522 275.75513529 143.49039896 143.19122017
 137.11060673 156.75893012 233.89382236 154.7544676  147.11832539
 161.31006444 144.09537269 138.58606518 118.70383153 118.41731411
 236.48529621 106.14436825 167.03857066 147.00723139 153.52346096
 154.85907691 311.81405572 169.29962424 169.49229267 107.0733158
 146.9366878  168.86812088 137.26230269 157.02494144 155.74482616
 290.96623547 118.96008702 268.5753385  152.08924987 116.01816802
 156.62718873 171.57209619 305.50710574 155.75580268 160.08663641
 159.14873481 119.58981128 156.81863192 274.65459053 118.64760793
 154.21061584 127.04800379 153.92170078 292.89476829 276.84758242
 133.62951045 206.93313012 121.75942506 254.51154659 159.95539764
 287.83964496 124.24831726 150.88368596 132.52271282 205.47428586
 161.21994461 236.01361245 156.60603614 143.83846653 127.6366197
 140.10327923 208.32119664 146.7555225  125.23264671 121.34081713
 148.18046076 151.89060175 136.49799451 127.74630842 154.18297468
 151.68480372 128.64744735 145.71829813 207.37370408 287.54570993
 147.27835173 153.89626423 151.28230879 270.86282456 170.89737938
 123.45652816 151.95581946 126.59514698 127.70514262 196.95735522
 104.92289948 159.51418911 117.36413133 148.9810419  161.69307701
 152.82605999 152.60741511 133.9006188  153.2207782  154.70493586
 124.47606698 206.18560723 137.99034075 145.86692907 141.90159654
 153.61912824 157.35971245 193.84158673 237.344119   118.61788346
 146.16975843 154.57012235 144.18791658 142.49713209 226.50269691
 279.38583461 119.71864564 123.03724471 134.98467467 149.04640017
 121.71936313 169.0657199  155.79927665 137.34602945 157.36265531
 117.63497716 246.22101315 137.43752235 160.88005468 143.05098803
 151.0196907  140.99275977 155.12471606 142.08257583 144.77574116
 141.7472009  165.05859009 246.97290669 139.54325296 124.46516325
 148.12529762 158.36609124 157.3906969  143.86703926 269.83991956
 159.24984816 236.75326227 128.25783071 125.86359812 146.21328427
 228.14500592 116.78781404 138.23549701 115.22638805 120.47432318
 119.62823197 146.93857553 133.23683056 198.54820582 134.33700388
 133.14246793 156.63468304 135.7900567  151.54160993 264.54029519
 124.75807063 162.33160964 233.36602837 133.66073319 278.97842867
 153.21135495 174.13326098 131.31111856 156.34423461 244.70005688
 141.74068961 279.84505023 133.0498587  281.58261476 129.76066812
 149.7567593  160.82445611 134.9958526  131.71353734 148.9744842
 149.78213973 143.6590409 ]
```

In [325]:

```
dframe = pd.DataFrame({'Actual Price':y_test,'Predicted Price':predicted})
```

In [328]:

```
print(dframe)
```

```
      Actual Price  Predicted Price
1083        153.45       153.294160
18          222.95       222.971272
1099        152.95       153.026105
818         132.60       133.202451
184         317.60       317.152680
...            ...              ...
479         134.95       134.995853
478         131.35       131.713537
934         149.05       148.974484
878         149.95       149.782140
1120        143.65       143.659041

[247 rows x 2 columns]
```

In [333]:

```
dframe.head(20)
```

Out[333]:

	Actual Price	Predicted Price
1083	153.45	153.294160
18	222.95	222.971272
1099	152.95	153.026105
818	132.60	133.202451
184	317.60	317.152680
608	122.40	122.508506
1213	160.35	161.664830
717	128.00	127.724289
1137	141.20	141.175738
1009	162.20	163.375983
1068	174.85	173.933842
621	118.25	118.290879
761	120.45	119.904949
1018	157.10	157.522924
5	233.75	233.305227
472	118.65	119.899183
45	242.20	243.608548
194	302.75	302.807032
311	153.35	153.399466
989	158.80	159.445850

In [334]:

```
from sklearn.metrics import confusion_matrix , accuracy_score
```

In [336]:

```
regressor.score(x_test,y_test)
```

Out[336]:

```
0.9999279075814551
```

In [337]:

```
import math
```

In [339]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test , predicted))
```

```
Mean Absolute Error: 0.2971082325895538
```

In [341]:

```
print('Mean Squared Error:',metrics.mean_squared_error(y_test , predicted))
```

```
Mean Squared Error: 0.16763887471410333
```

In [342]:

```
print('Root Mean Squared Error:',math.sqrt(metrics.mean_squared_error(y_test , predicte
d)))
```

```
Root Mean Squared Error: 0.4094372659078596
```

Final Predicted Price

In [343]:

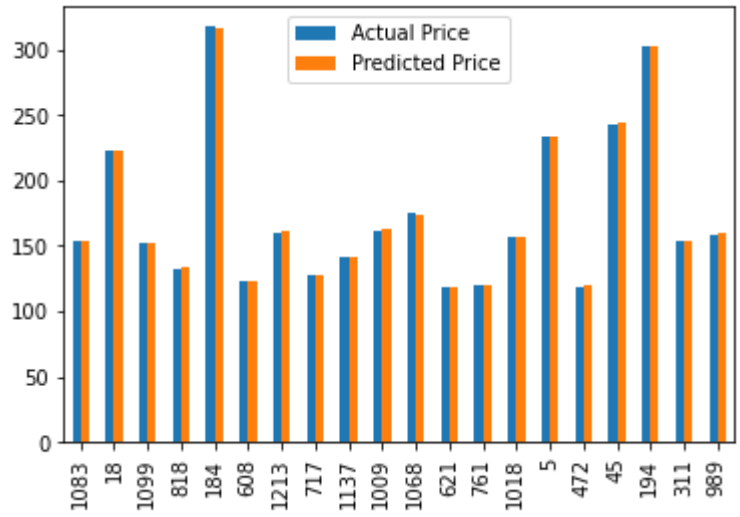
```
graph=dframe.head(20)
```

In [347]:

```
graph.plot(kind='bar')
```

Out[347]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2875cc764f0>
```



In [ ]:

```

```

In [ ]:

```

```