In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data=pd.read_excel(r'C:\Users\Debabrata\Downloads\STOCKPRICEDATASET.xlsx')
data.head()
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       1009 non-null   datetime64[ns]
 1   Open       1009 non-null   float64       
 2   High       1009 non-null   float64       
 3   Low        1009 non-null   float64       
 4   Close      1009 non-null   float64       
 5   Adj Close  1009 non-null   float64       
 6   Volume     1009 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 55.3 KB
```

In [4]:
```
data.shape
```

Out[4]:
```
(1009, 7)
```

In [5]:
```
data.count()
```

Out[5]:
```
Date         1009
Open         1009
High         1009
Low          1009
Close        1009
Adj Close    1009
Volume       1009
dtype: int64
```

In [6]:
```
data.isnull().sum
```

Out[6]:
```
<bound method NDFrame._add_numeric_operations.<locals>.sum of          Date   Open   High    Low  Close  Adj Close  Volume
0       False  False  False  False  False      False   False
1       False  False  False  False  False      False   False
2       False  False  False  False  False      False   False
3       False  False  False  False  False      False   False
4       False  False  False  False  False      False   False
...       ...    ...    ...    ...    ...        ...     ...
1004    False  False  False  False  False      False   False
1005    False  False  False  False  False      False   False
1006    False  False  False  False  False      False   False
1007    False  False  False  False  False      False   False
1008    False  False  False  False  False      False   False

[1009 rows x 7 columns]>
```

In [7]:
```
data.describe()
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

In [8]:
```
data.dtypes
```

Out[8]:
```
Date         datetime64[ns]
Open                float64
High                float64
Low                 float64
Close               float64
Adj Close           float64
Volume                int64
dtype: object
```

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

Out[9]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [12]:
```
data['Date']=pd.to_datetime(data['Date'])
data.head()
```

Out[12]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [13]:
```
data['Day']=data['Date'].dt.day
data['Month']=data['Date'].dt.month
data['Year']=data['Date'].dt.year
```

In [14]:
```
data.head()
```

Out[14]:

	Date	Open	High	Low	Close	Adj Close	Volume	Day	Month	Year
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100	5	2	2018
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800	6	2	2018
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500	7	2	2018
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700	8	2	2018
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900	9	2	2018

In [15]:
```
data.drop('Date',axis=1,inplace=True)
```

In [16]:
```
data.head()
```

Out[16]:

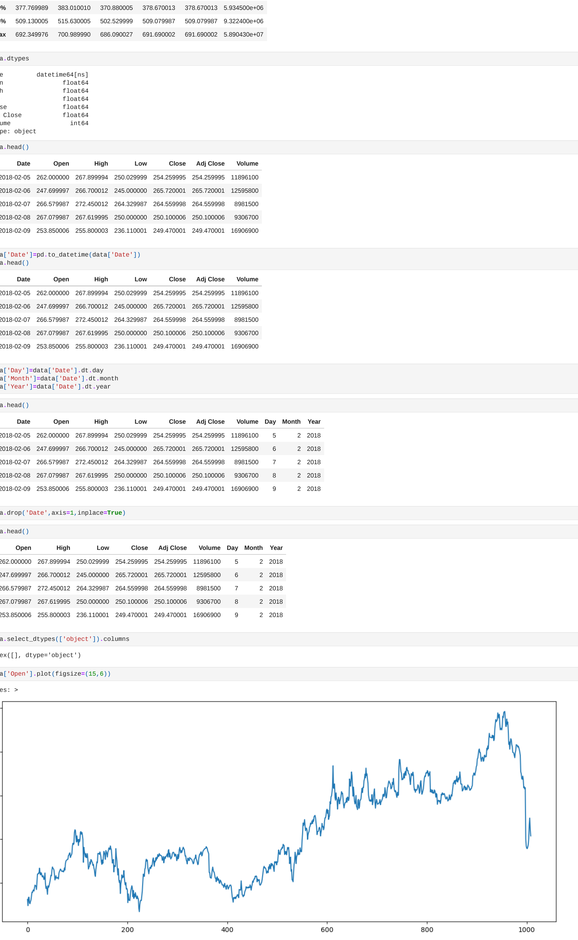
	Open	High	Low	Close	Adj Close	Volume	Day	Month	Year
0	262.000000	267.899994	250.029999	254.259995	254.259995	11896100	5	2	2018
1	247.699997	266.700012	245.000000	265.720001	265.720001	12595800	6	2	2018
2	266.579987	272.450012	264.329987	264.559998	264.559998	8981500	7	2	2018
3	267.079987	267.619995	250.000000	250.100006	250.100006	9306700	8	2	2018
4	253.850006	255.800003	236.110001	249.470001	249.470001	16906900	9	2	2018

In [17]:
```
data.select_dtypes(['object']).columns
```

Out[17]:
```
Index([], dtype='object')
```

In [19]:
```
data['Open'].plot(figsize=(15,6))
```

Out[19]:
```
<Axes: >
```



In [20]:
```
x=data[['Open','High','Low','Volume']]
y=data['Close']
```

In [23]:
```
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test=tts(x,y,test_size=0.1, random_state=0)
```

In [25]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(908, 4)
(101, 4)
(908,)
(101,)
```

In [28]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

In [36]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [38]:
```
print(model.coef_)
```

```
[-6.10157277e-01  7.66273579e-01  8.45491412e-01  9.89377614e-08]
```

In [40]:
```
print(model.intercept_)
```

```
-0.6658384475784374
```

In [44]:
```
y_pred=model.predict(x_test)
```

In [46]:
```
from sklearn.metrics import mean_squared_error,r2_score,absolute_error
def metrics(y_true,y_pred):
    print(f'RMSE:',mean_squared_error(y_true,y_pred)**0.5)
    print(f'R_squared value:',r2_score(y_true,y_pred))
    print(f'Absolute value:',)

def accuracy(y_true,y_pred):
    errors=abs(y_true-y_pred)
    mape=100*np.mean(errors/y_true)
    accuracy=100-mape
    return accuracy
```

In [47]:
```
accuracy(y_test,y_pred)
```

Out[47]:
```
99.2640812586824
```

In [48]:
```
metrics(y_test,y_pred)
```

```
RMSE: 4.084887524477271S
R_squared value: 0.9984149741460959
```

In [50]:
```
model.score(x_test,y_test)
```

Out[50]:
```
0.9984149741460959
```

In [60]:
```
dframe=pd.DataFrame(y_test,y_pred)
dfrm=pd.DataFrame({'Actual Price':y_test,'Predicted Price_by Linear Regression':y_pred})
```

In [61]:
```
print(dfr)
```

```
     Actual Price  Predicted Price_by Linear Regression
801         553.729980                           553.941248
311         379.059998                           379.739711
85          361.399994                           360.327695
435         281.859985                           283.757864
204         261.429993                           260.200337
..                 ...                                  ...
363         362.440002                           363.909305
787         535.090027                           540.364223
175         333.130005                           330.780499
603         447.239990                           436.561459
859         541.640015                           541.021081

[101 rows x 2 columns]
```

In [57]:
```
from sklearn.ensemble import RandomForestRegressor
model_random_forest=RandomForestRegressor(n_estimators=500)
model_random_forest.fit(x_train,y_train)
```

Out[57]:
```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=500)
```

In [63]:
```
pred_rf=model_random_forest.predict(x_test)
```

In [65]:
```
dframeRF=pd.DataFrame(y_test,pred_rf)
dfrRF=pd.DataFrame({'Actual Price':y_test,'Predicted Price_by Random Forest Regressor':pred_rf})
```

In [66]:
```
print(dfrRF)
```

```
     Actual Price  Predicted Price_by Random Forest Regressor
801         553.729980                                554.526434
311         379.059998                                380.038079
85          361.399994                                360.938181
435         281.859985                                281.384553
204         261.429993                                260.957578
..                 ...                                       ...
363         362.440002                                361.993480
787         535.090027                                538.375211
175         333.130005                                332.768535
603         447.239990                                436.861640
859         541.640015                                539.378658

[101 rows x 2 columns]
```

In [67]:
```
accuracy(y_test,pred_rf)
```

Out[67]:
```
99.22496428354569
```

In [ ]:
```

```