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\KIIT\Downloads\1729258-1613615-Stock_Price_data_set_(1) (3).xlsx')
```

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

Out[3]:

	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 [4]:
```
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 [5]:
```
data.shape
```

Out[5]:
```
(1009, 7)
```

In [6]:
```
data.count()
```

Out[6]:
```
Date         1009
Open         1009
High         1009
Low          1009
Close        1009
Adj Close    1009
Volume       1009
dtype: int64
```

In [7]:
```
data.dtypes
```

Out[7]:
```
Date         datetime64[ns]
Open                float64
High                float64
Low                 float64
Close               float64
Adj Close           float64
Volume                int64
dtype: object
```

In [8]:
```
data['Date'].dt.year.unique()
```

Out[8]:
```
array([2018, 2019, 2020, 2021, 2022], dtype=int64)
```

In [9]:
```
data['Date'].dt.month.unique()
```

Out[9]:
```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,  1], dtype=int64)
```

In [10]:
```
data['Day']=data['Date'].dt.day
data['Month']=data['Date'].dt.month
data['Year']=data['Date'].dt.month
```

In [11]:
```
data.drop('Date',axis=1,inplace=True)
```

In [12]:
```
data.head()
```

Out[12]:

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

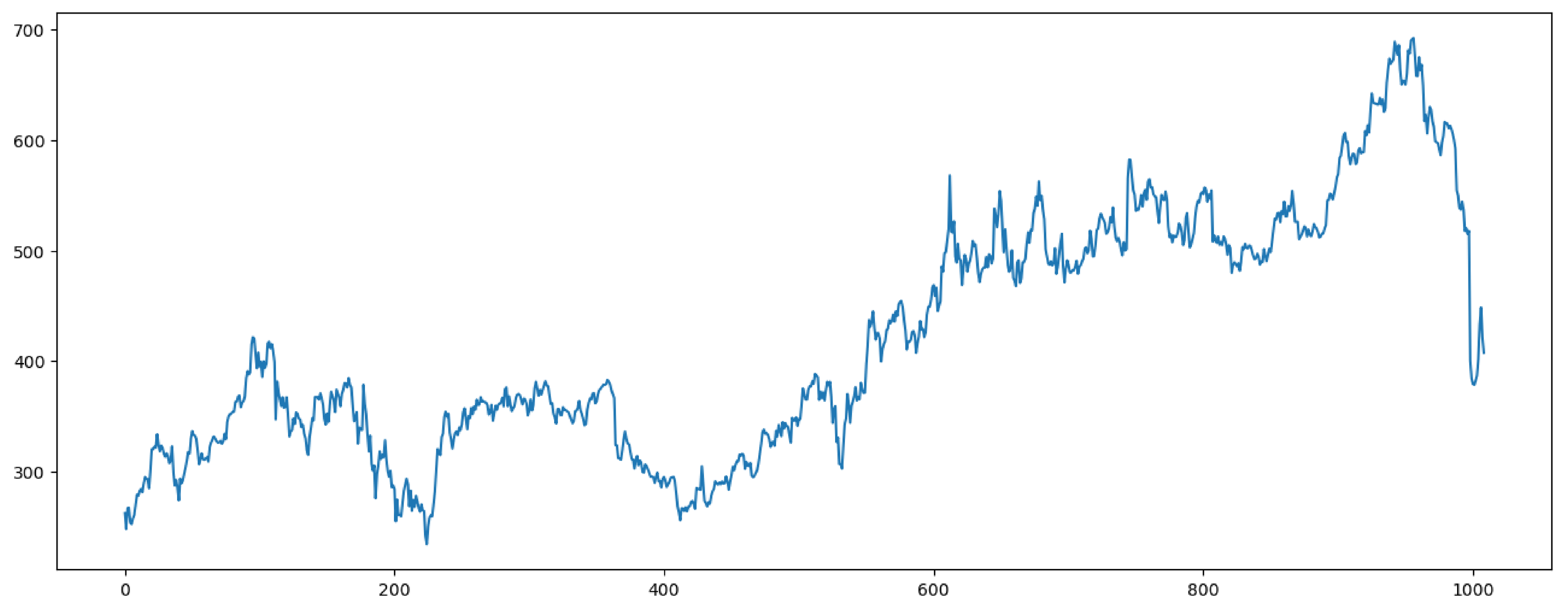
In [13]:
```
data.drop('Adj Close',axis=1,inplace=True)
```

In [16]:
```
print (len(data))
```

```
1009
```

In [17]:
```
data['Open'].plot(figsize=(16,6))
```

Out[17]:
```
<Axes: >
```



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

In [21]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0)
```

In [22]:
```
X_train.shape
```

Out[22]:
```
(756, 4)
```

In [23]:
```
X_test.shape
```

Out[23]:
```
(253, 4)
```

In [24]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix,accuracy_score
regressor=LinearRegression()
```

In [25]:
```
regressor.fit(X_train,y_train)
```

Out[25]:
```
▾ LinearRegression
LinearRegression()
```

In [26]:
```
print(regressor.coef_)
```

```
[-5.98637670e-01  7.42752459e-01  8.57948723e-01  9.68159262e-08]
```

In [27]:
```
print(regressor.intercept_)
```

```
-0.7077595564785497
```

In [28]:
```
predicted=regressor.predict(X_test)
```

In [29]:
```
print(X_test)
```

```
            Open        High         Low    Volume
801   557.000000  559.750000  550.299988   2720300
311   378.000000  383.500000  374.510010   5398200
85    368.540009  368.700012  357.799988   8278000
435   278.049988  285.750000  277.350006   6248400
204   260.549988  266.250000  253.800003  12498600
...          ...         ...         ...       ...
583   418.829987  426.720001  415.980011   3743700
200   283.790009  285.089996  269.149994  12993800
767   525.000000  548.539978  518.280029   4136500
1000  379.140015  387.709991  365.130005  15145800
385   298.859985  303.549988  296.269989   6905800

[253 rows x 4 columns]
```

In [30]:
```
predicted.shape
```

Out[30]:
```
(253,)
```

In [31]:
```
dframe=pd.DataFrame(y_test,predicted)
```

In [32]:
```
dfr=pd.DataFrame({'Actualprice':y_test,'Predictedprice':predicted})
```

In [33]:
```
print(dfr)
```

```
      Actualprice  Predictedprice
801    553.729980      553.999288
311    379.059998      379.685786
85     361.399994      360.298634
435    281.859985      283.639587
204    261.429993      260.032497
...           ...             ...
583    425.920013      422.764132
200    270.600006      273.331047
767    546.150024      537.495050
1000   366.420013      375.026471
385    302.799988      300.698946

[253 rows x 2 columns]
```

In [34]:
```
dfr.head()
```

Out[34]:

	Actualprice	Predictedprice
801	553.729980	553.999288
311	379.059998	379.685786
85	361.399994	360.298634
435	281.859985	283.639587
204	261.429993	260.032497

In [35]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [36]:
```
regressor.score(X_test,y_test)
```

Out[36]:
```
0.9982601041694569
```

In [50]:
```
import math
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
def metrics(y_test,predicted):
    print(f'Mean Absolute Error: ',mean_absolute_error(y_test,predicted))
    print('Mean squared Error',mean_squared_error(y_test,predicted)**0.5)
    print('Root mean squared error:',math.sqrt(mean_squared_error(y_test,predicted)**0.5))
metrics(y_test,predicted)
```

```
Mean Absolute Error:  3.124088127337689
Mean squared Error 4.377626336434151
Root mean squared error: 2.0922777866321076
```

In [52]:
```
def accuracy(y_test,predicted):
    error=abs(y_test-predicted)
    mape=100*np.mean(error/y_test)
    accuracy=100-mape
    return accuracy
accuracy(y_test,predicted)
```

Out[52]:
```
99.23154051624365
```

In [53]:
```
regressor.score(X_test,y_test)
```

Out[53]:
```
0.9982601041694569
```

In [54]:
```
from sklearn.ensemble import RandomForestRegressor
model_random_forest=RandomForestRegressor(n_estimators=500,min_samples_split=3)
model_random_forest.fit(X_train,y_train)
```

Out[54]:
```
▾ RandomForestRegressor
RandomForestRegressor(min_samples_split=3, n_estimators=500)
```

In [55]:
```
pred=model_random_forest.predict(X_test)
```

In [56]:
```
metrics(y_test,pred)
```

```
Mean Absolute Error:  4.013913202416633
Mean squared Error 5.6309167366228055
Root mean squared error: 2.3729552748888474
```

In [57]:
```
accuracy(y_test,pred)
```

Out[57]:
```
99.02981200686772
```

In [ ]: