TASK-5 Stock-Price-Prediction

In [31]:

```
#import packages
import pandas as pd
import numpy as np

#to plot within notebook
import matplotlib.pyplot as plt
%matplotlib inline

#setting figure size
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

In [32]:

```
#read the file
df = pd.read_csv('NSE-TATAGLOBAL11.csv')
```

In [33]:

```
df.info()
```

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

In [34]:

```
df.describe()
```

Out[34]:

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
count	1235.000000	1235.000000	1235.000000	1235.000000	1235.000000	1.235000e+03	1235.000000
mean	168.954858	171.429069	166.402308	168.736356	168.731053	2.604151e+06	4843.166502
std	51.499145	52.436761	50.542919	51.587384	51.544928	2.277028e+06	5348.919832
min	103.000000	104.600000	100.000000	102.600000	102.650000	1.001800e+05	128.040000
25%	137.550000	138.925000	135.250000	137.175000	137.225000	1.284482e+06	1801.035000
50%	151.500000	153.250000	149.500000	151.200000	151.100000	1.964885e+06	3068.510000

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
75%	169.000000	172.325000	166.700000	169.100000	169.500000	3.095788e+06	5852.600000
max	327.700000	328.750000	321.650000	325.950000	325.750000	2.919102e+07	55755.080000

In [35]:

```
#print the head
df.head()
```

Out[35]:

	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 [36]:

```
df.shape
```

Out[36]:

```
(1235, 8)
```

In [37]:

```
df.isnull().values.any()
```

Out[37]:

```
False
```

In [38]:

```
df.isnull().sum()
```

Out[38]:

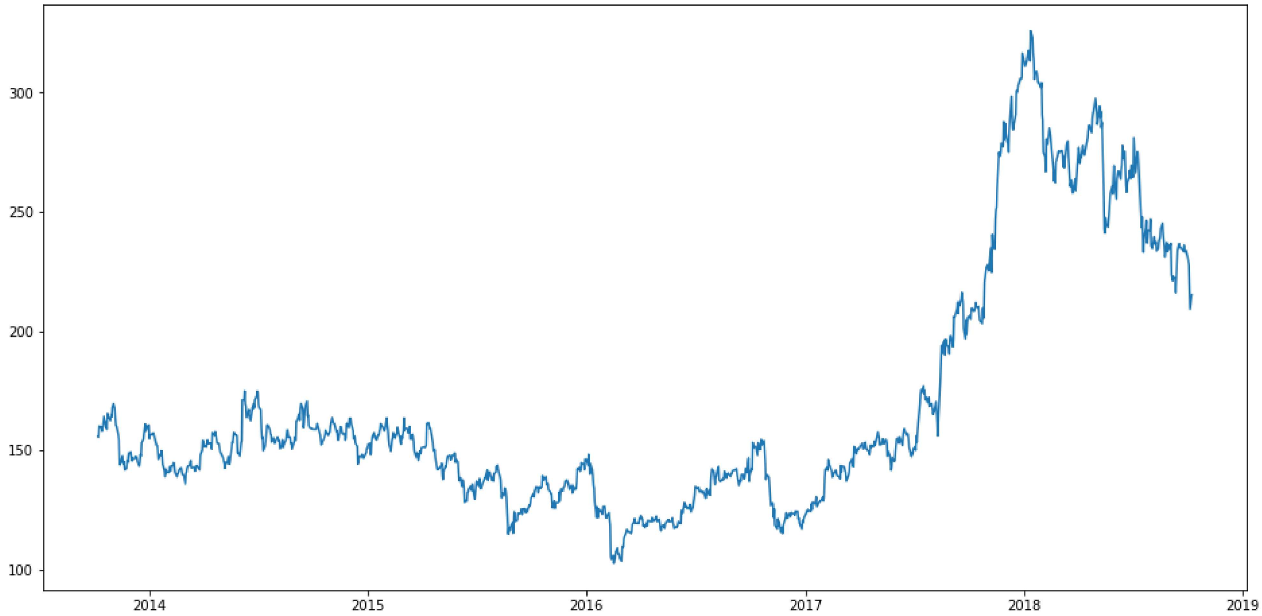
```
Date                    0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
Total Trade Quantity    0
Turnover (Lacs)         0
dtype: int64
```

In [39]:

```
#setting index as date
df['Date'] = pd.to_datetime(df.Date,format='%d-%m-%Y')
df.index = df['Date']

#plot
plt.figure(figsize=(16,8))
plt.plot(df['Close'], label='Close Price history')
```

Out[39]:

```
[<matplotlib.lines.Line2D at 0x1c1fa873208>]
```



In [40]:

```
#creating dataframe with date and the target variable
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
     new_data['Date'][i] = data['Date'][i]
     new_data['Close'][i] = data['Close'][i]
```

In [41]:

```
#splitting into train and validation
train = new_data[:987]
valid = new_data[987:]
```

In [42]:

```
new_data.shape, train.shape, valid.shape
```

Out[42]:

```
((1235, 2), (987, 2), (248, 2))
```

In [43]:

```
train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()
```

Out[43]:

```
(Timestamp('2013-10-08 00:00:00'),
 Timestamp('2017-10-06 00:00:00'),
 Timestamp('2017-10-09 00:00:00'),
 Timestamp('2018-10-08 00:00:00'))
```

In [44]:

```
#make predictions
preds = []
for i in range(0,248):
    a = train['Close'][len(train)-248+i:].sum() + sum(preds)
    b = a/248
    preds.append(b)
```

In [45]:

```
#calculate rmse
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
rms
```

Out[45]:

```
104.51415465984348
```

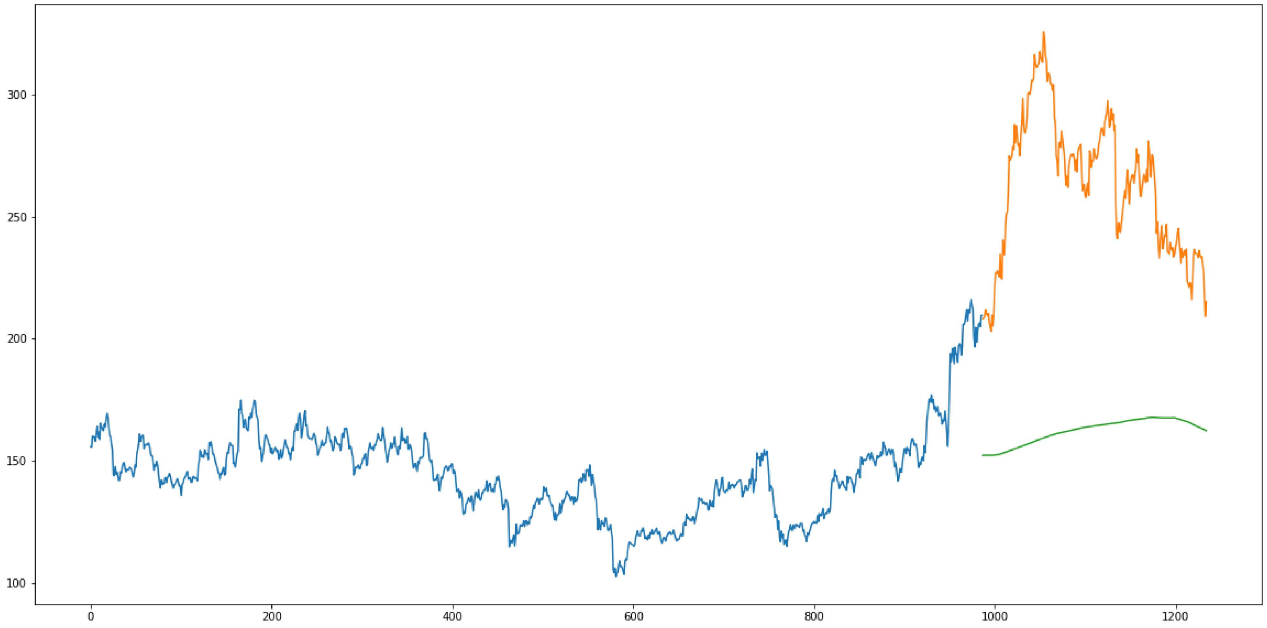
In [46]:

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy

c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[46]:

```
[<matplotlib.lines.Line2D at 0x1c1a8f7e5f8>,
 <matplotlib.lines.Line2D at 0x1c1a8f7e6a0>]
```



LINEAR REGRESSION

In [47]:

```
#setting index as date values
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

#sorting
data = df.sort_index(ascending=True, axis=0)

#creating a separate dataset
```

```
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

In [48]:

```
new_data.drop('Date', axis=1, inplace=True)  #elapsed will be the time stamp
```

In [49]:

```
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

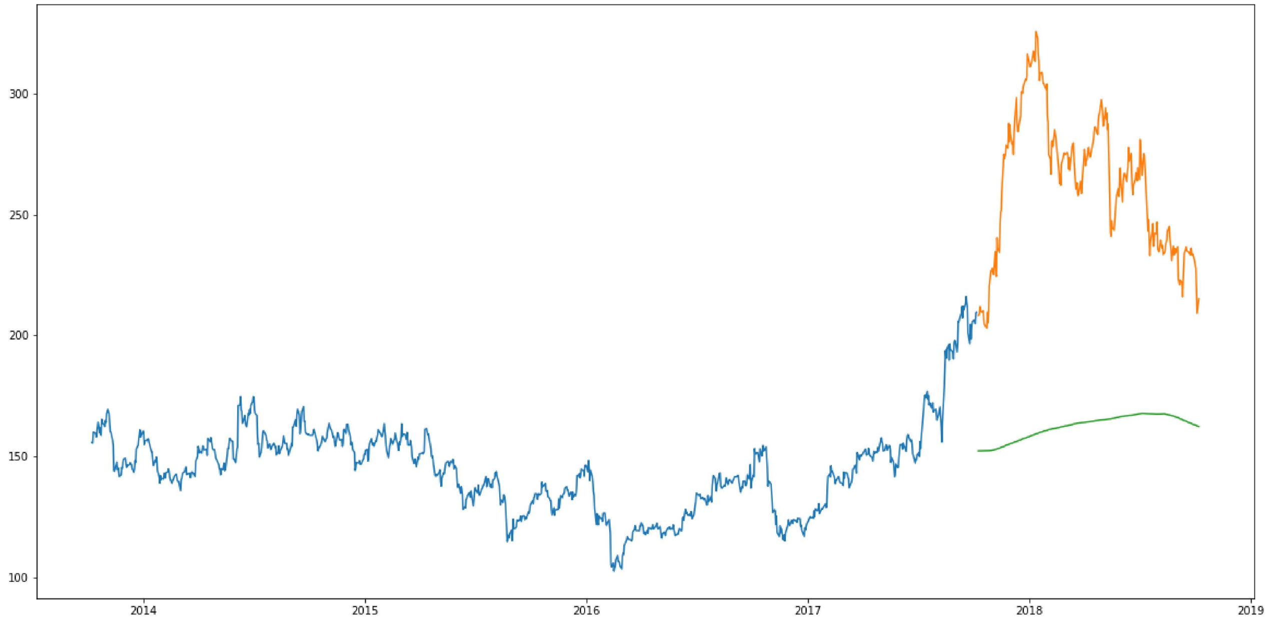
In [50]:

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = data[987:].index
train.index = data[:987].index

plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy

c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[50]:

```
[<matplotlib.lines.Line2D at 0x1c1a8ff1c18>,
 <matplotlib.lines.Line2D at 0x1c1a8ff1cc0>]
```



LONG SHORT TERM MEMORY(LSTM)

In [51]:

```
#importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

#setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)

#creating train and test sets
dataset = new_data.values

train = dataset[0:987,:]
valid = dataset[987:,:]

#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

# create and fit the LSTM network
model = Sequential()
```

```
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

```
927/927 - 13s - loss: 0.0012
```

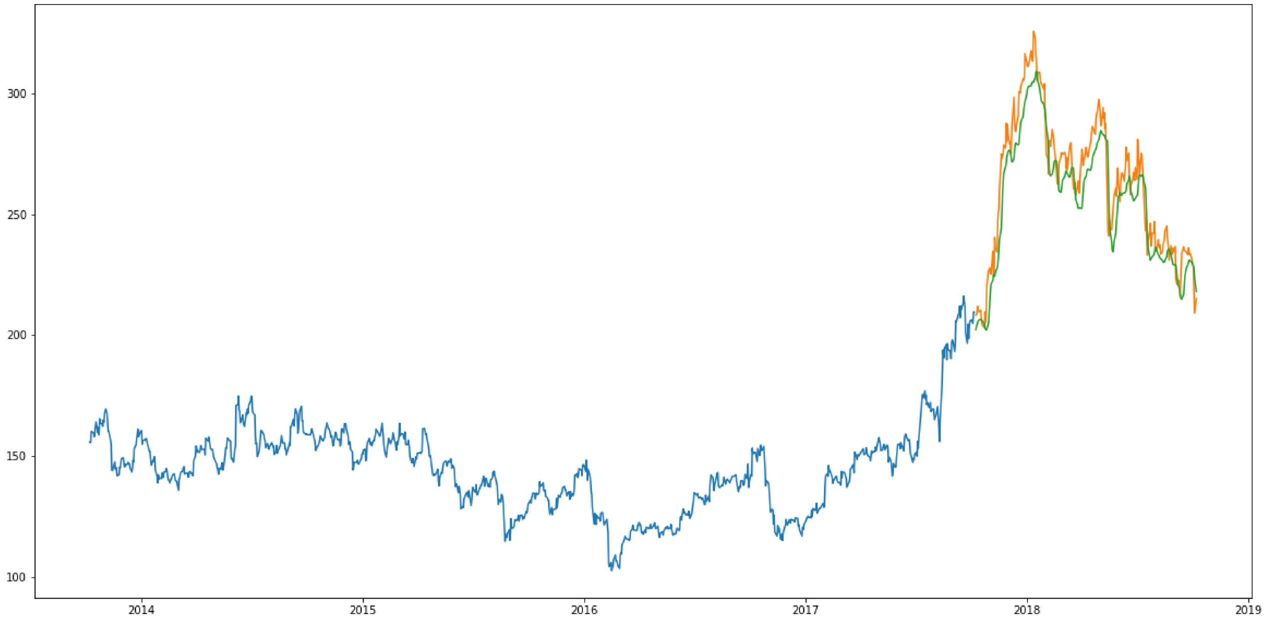
In [52]:

```
#for plotting
train = new_data[:987]
valid = new_data[987:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:4: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x1c1a99baa20>,
 <matplotlib.lines.Line2D at 0x1c1a998dbe0>]
```



In [ ]: