In [14]:
```
#import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [166]:
```
df = pd.read_csv('C://Users/shash/Downloads/NSE-TATAGLOBAL11.csv')
df
```

Out[166]:

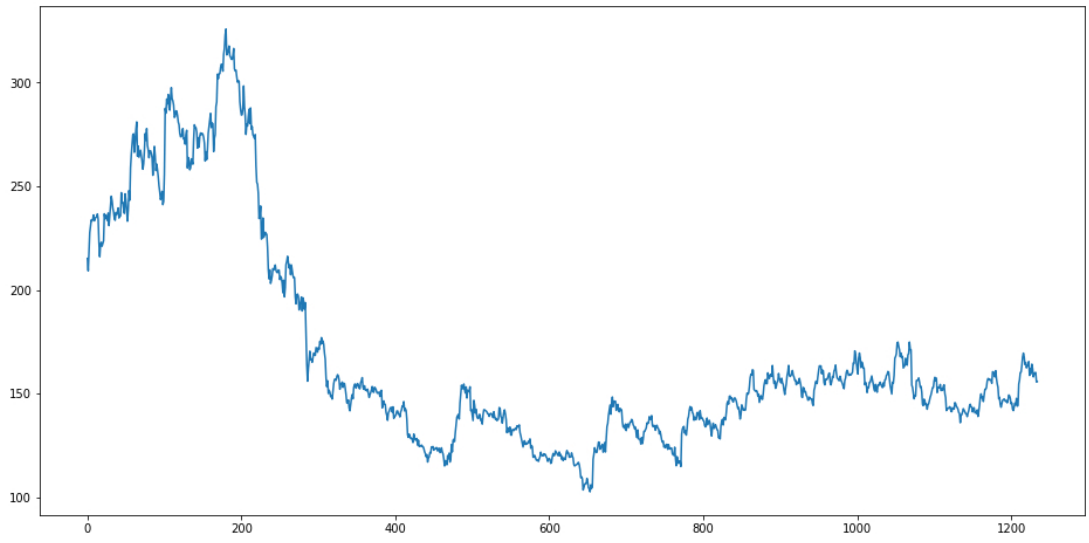
	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05
5	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914.0	7162.35
6	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859.0	11859.95
7	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909.0	5248.60
8	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368.0	5503.90
9	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509.0	7999.55
10	2018-09-21	235.00	237.00	227.95	233.75	234.60	5395319.0	12589.59

In [167]:
```
#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

In [168]:
```
#setting figure size
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10
```

In [169]:
```
#plot
plt.figure(figsize=(16,8))
plt.plot(df['Close'], label='Close Price history')
```

Out[169]:
```
[<matplotlib.lines.Line2D at 0x280f65718d0>]
```



In [170]:
```
#setting index as date
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']
df.head()
```

Out[170]:

Date	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2018-10-08	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
2018-10-05	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2018-10-04	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
2018-10-03	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
2018-10-01	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

In [171]:
```
#plot
```

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

Out[171]: [<matplotlib.lines.Line2D at 0x280f65bada0>]

In [12]:
```
import tensorflow as tf
```

In [20]:
```
import sklearn
from sklearn.preprocessing import MinMaxScaler
```

In [21]:
```
#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

In [172]:
```
Sequential=tf.keras.models.Sequential()
Dense=tf.keras.layers.Dense
Dropout=tf.keras.layers.Dropout
LSTM=tf.keras.layers.LSTM
```

In [177]:
```
#arranging in ascending order of index value
data = df.sort_index(ascending=True, axis=0)
#making a data frame with only date and close as columns
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
```

In [178]:
```
new_data[0:5]
```

Out[178]:

	Date	Close
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

In [179]:
```
#entering data to our new empty data frame
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

In [180]:
```
new_data
```

Out[180]:

	Date	Close
0	2013-10-08 00:00:00	155.8
1	2013-10-09 00:00:00	155.55
2	2013-10-10 00:00:00	160.15
3	2013-10-11 00:00:00	160.05
4	2013-10-14 00:00:00	159.45
5	2013-10-15 00:00:00	158.05
6	2013-10-17 00:00:00	162
7	2013-10-18 00:00:00	164.2
8	2013-10-21 00:00:00	159.6
9	2013-10-22 00:00:00	161.85
10	2013-10-23 00:00:00	158.75
11	2013-10-24 00:00:00	165.45
12	2013-10-25 00:00:00	163.85
13	2013-10-28 00:00:00	163.25
14	2013-10-29 00:00:00	162.4

15	2013-10-30 00:00:00	165
16	2013-10-31 00:00:00	164
17	2013-11-01 00:00:00	167.7
18	2013-11-03 00:00:00	169.5
19	2013-11-05 00:00:00	167.6
20	2013-11-06 00:00:00	163.55
21	2013-11-07 00:00:00	160.35
22	2013-11-08 00:00:00	160.1
23	2013-11-11 00:00:00	156.55
24	2013-11-12 00:00:00	154.55
25	2013-11-13 00:00:00	144.3
26	2013-11-14 00:00:00	143.95
27	2013-11-18 00:00:00	147.7
28	2013-11-19 00:00:00	144.55
29	2013-11-20 00:00:00	145.35
...	...	...
1205	2018-08-23 00:00:00	235.45
1206	2018-08-24 00:00:00	231
1207	2018-08-27 00:00:00	237.05
1208	2018-08-28 00:00:00	233.35
1209	2018-08-29 00:00:00	234.55
1210	2018-08-30 00:00:00	236
1211	2018-08-31 00:00:00	234.3
1212	2018-09-03 00:00:00	236.7
1213	2018-09-04 00:00:00	223.7
1214	2018-09-05 00:00:00	222.4
1215	2018-09-06 00:00:00	221.05
1216	2018-09-07 00:00:00	222.95
1217	2018-09-10 00:00:00	222
1218	2018-09-11 00:00:00	216
1219	2018-09-12 00:00:00	222.65
1220	2018-09-14 00:00:00	233.95
1221	2018-09-17 00:00:00	236.6
1222	2018-09-18 00:00:00	235.05
1223	2018-09-19 00:00:00	234.9
1224	2018-09-21 00:00:00	234.6
1225	2018-09-24 00:00:00	233.3
1226	2018-09-25 00:00:00	236.1
1227	2018-09-26 00:00:00	234.25
1228	2018-09-27 00:00:00	233.25
1229	2018-09-28 00:00:00	233.75
1230	2018-10-01 00:00:00	230.9
1231	2018-10-03 00:00:00	227.6
1232	2018-10-04 00:00:00	218.2
1233	2018-10-05 00:00:00	209.2
1234	2018-10-08 00:00:00	215.15

1235 rows × 2 columns

In [181]:
```
#setting date as index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)
```

In [182]:
```
new_data
```

Out[182]:

Date	Close
2013-10-08	155.8
2013-10-09	155.55
2013-10-10	160.15
2013-10-11	160.05
2013-10-14	159.45
2013-10-15	158.05
2013-10-17	162
2013-10-18	164.2
2013-10-21	159.6
2013-10-22	161.85
2013-10-23	158.75
2013-10-24	165.45

2013-10-24	165.45
2013-10-25	163.85
2013-10-28	163.25
2013-10-29	162.4
2013-10-30	165
2013-10-31	164
2013-11-01	167.7
2013-11-03	169.5
2013-11-05	167.6
2013-11-06	163.55
2013-11-07	160.35
2013-11-08	160.1
2013-11-11	156.55
2013-11-12	154.55
2013-11-13	144.3
2013-11-14	143.95
2013-11-18	147.7
2013-11-19	144.55
2013-11-20	145.35
...	...
2018-08-23	235.45
2018-08-24	231
2018-08-27	237.05
2018-08-28	233.35
2018-08-29	234.55
2018-08-30	236
2018-08-31	234.3
2018-09-03	236.7
2018-09-04	223.7
2018-09-05	222.4
2018-09-06	221.05
2018-09-07	222.95
2018-09-10	222
2018-09-11	216
2018-09-12	222.65
2018-09-14	233.95
2018-09-17	236.6
2018-09-18	235.05
2018-09-19	234.9
2018-09-21	234.6
2018-09-24	233.3
2018-09-25	236.1
2018-09-26	234.25
2018-09-27	233.25
2018-09-28	233.75
2018-10-01	230.9
2018-10-03	227.6
2018-10-04	218.2
2018-10-05	209.2
2018-10-08	215.15

1235 rows × 1 columns

In [183]:
```
#creating training and testing sets
dataset = new_data.values

train = dataset[0:987,:]
test = dataset[987:,:]
print(train.shape,test.shape)
test
```

```
       [222.95],
       [222.0],
       [216.0],
       [222.65],
       [233.95],
       [236.6],
       [235.05],
       [234.9],
       [234.6],
       [233.3],
       [236.1],
       [234.25],
       [233.25],
       [233.75],
```

```
       [230.9],
       [227.6],
       [218.2],
       [209.2],
       [215.15]], dtype=object)
```

In [184]:
```
#converting dataset into a given range
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

print(scaled_data.shape,scaled_data)
scaled_data[60,0]
```

```
(1235, 1) [[0.23823398]
 [0.2371134 ]
 [0.25773196]
 ...
 [0.51792918]
 [0.47758853]
 [0.50425818]]
```

Out[184]: 0.24069923800986098

In [185]:
```
#converting dataset into x_train and y_train
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
```

```
       0.24831914, 0.25616316, 0.25907665, 0.25930076, 0.23442403,
       0.24069924, 0.24271627, 0.24137158, 0.24406096, 0.2447333 ]),
 array([0.2660242 , 0.27588525, 0.2552667 , 0.26535186, 0.25145675,
       0.28148812, 0.27431645, 0.27162707, 0.26781712, 0.27947109,
       0.27498879, 0.29157329, 0.29964142, 0.29112506, 0.27297176,
       0.25862842, 0.25750784, 0.2415957 , 0.23263111, 0.18668758,
       0.18511878, 0.20192739, 0.18780816, 0.19139399, 0.18377409,
       0.17548185, 0.17817122, 0.19206634, 0.19004931, 0.20170327,
       0.20730614, 0.20909906, 0.19789332, 0.19229045, 0.19722098,
       0.19520394, 0.19744509, 0.20080681, 0.19878978, 0.19766921,
       0.19049753, 0.18287763, 0.18870462, 0.20416853, 0.20103093,
       0.22613178, 0.23330345, 0.2447333 , 0.26176602, 0.24831914,
       0.25616316, 0.25907665, 0.25930076, 0.23442403, 0.24069924,
       0.24271627, 0.24137158, 0.24406096, 0.2447333 , 0.23890632]),
 array([0.27588525, 0.2552667 , 0.26535186, 0.25145675, 0.28148812,
       0.27431645, 0.27162707, 0.26781712, 0.27947109, 0.27498879,
       0.29157329, 0.29964142, 0.29112506, 0.27297176, 0.25862842,
       0.25750784, 0.2415957 , 0.23263111, 0.18668758, 0.18511878,
       0.20192739, 0.18780816, 0.19139399, 0.18377409, 0.17548185,
       0.17817122, 0.19206634, 0.19004931, 0.20170327, 0.20730614,
```

In [186]:
```
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [187]:
```
x_train
```

Out[187]:
```
array([[0.23823398, 0.2371134 , 0.25773196, ..., 0.25907665, 0.25930076,
        0.23442403],
       [0.2371134 , 0.25773196, 0.25728373, ..., 0.25930076, 0.23442403,
        0.24069924],
       [0.25773196, 0.25728373, 0.25459435, ..., 0.23442403, 0.24069924,
        0.24271627],
       ...,
       [0.24697445, 0.28641865, 0.29964142, ..., 0.45047064, 0.46234872,
        0.46526221],
       [0.28641865, 0.29964142, 0.31869117, ..., 0.46234872, 0.46526221,
        0.45853877],
       [0.29964142, 0.31869117, 0.32608696, ..., 0.46526221, 0.45853877,
        0.4787091 ]])
```

In [188]:
```
x_train.shape
```

Out[188]: (927, 60)

In [189]:
```
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

In [190]:
```
x_train.shape
```

Out[190]: (927, 60, 1)

In [191]:
```
# 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))
```

In [192]:
```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=5, batch_size=1, verbose=2)
```

```
Epoch 1/5
 - 232s - loss: 0.0015
Epoch 2/5
 - 191s - loss: 5.3661e-04
Epoch 3/5
 - 198s - loss: 4.0068e-04
Epoch 4/5
 - 195s - loss: 3.0068e-04
Epoch 5/5
 - 185s - loss: 3.0284e-04
```

Out[192]: <tensorflow.python.keras.callbacks.History at 0x280f86ab080>

In [193]:
```
print(len(new_data),len(test))
```

```
1235 248
```

In [194]:
```
new_data[800:810]
```

Out[194]:

Date	Close
2017-01-05	125.2
2017-01-06	124.65
2017-01-09	124.75
2017-01-10	124.45
2017-01-11	127.55
2017-01-12	125.15
2017-01-13	128.35
2017-01-16	127.65
2017-01-17	127.6
2017-01-18	128.65

In [195]:
```
#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(test) - 60:].values
inputs
```

```
       [171.3],
       [172.2],
       [170.0],
       [171.15],
       [172.2],
       [168.4],
       [169.15],
       [169.55],
       [167.5],
       [165.1],
       [166.65],
       [166.45],
       [170.45],
       [166.0],
       [162.1],
       [156.0],
       [164.2],
       [178.75],
       [193.85],
```

In [196]:
```
inputs = inputs.reshape(-1,1)
inputs
```

Out[196]:
```
array([[173.75],
       [175.4],
       [174.2],
       [177.0],
       [173.75],
       [175.15],
       [171.3],
       [172.2],
       [170.0],
       [171.15],
       [172.2],
       [168.4],
       [169.15],
       [169.55],
       [167.5],
       [165.1],
       [166.65],
       [166.45],
       [170.45],
```

In [197]:
```
inputs  = scaler.transform(inputs)
inputs
```

Out[197]:
```
array([[0.31869117],
       [0.32608696],
       [0.3207082 ],
       [0.33325863],
       [0.31869117],
       [0.32496638],
       [0.30770955],
       [0.31174361],
       [0.30188256],
       [0.3070372 ],
       [0.31174361],
       [0.29471089],
       [0.29807261],
       [0.29986553],
       [0.29067683],
       [0.27991932],
       [0.28686688],
       [0.28597042],
       [0.3038996 ],
```

In [198]:
```
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
```

Out[198]:
```
array([[0.31869117, 0.32608696, 0.3207082 , ..., 0.45853877, 0.4787091 ,
        0.47938144],
       [0.32608696, 0.3207082 , 0.33325863, ..., 0.4787091 , 0.47938144,
        0.47355446],
       [0.3207082 , 0.33325863, 0.31869117, ..., 0.47938144, 0.47355446,
        0.4742268 ],
       ...,
       [0.75571493, 0.73352757, 0.77364411, ..., 0.58762887, 0.57485433,
        0.56006275],
       [0.73352757, 0.77364411, 0.77140296, ..., 0.57485433, 0.56006275,
        0.51792918],
       [0.77364411, 0.77140296, 0.7492156 , ..., 0.56006275, 0.51792918,
        0.47758853]])
```

In [199]:
```
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
X_test
```

Out[199]:
```
array([[[0.31869117],
        [0.32608696],
        [0.3207082 ],
        ...,
        [0.45853877],
        [0.4787091 ],
        [0.47938144]],

       [[0.32608696],
        [0.3207082 ],
        [0.33325863],
        ...,
        [0.4787091 ],
        [0.47938144],
        [0.47355446]],

       [[0.3207082 ],
        [0.33325863],
        [0.31869117],
        ...,
        [0.47938144],
        [0.47355446],
        [0.4742268 ]],

       ...,

       [[0.75571493],
        [0.73352757],
        [0.77364411],
        ...,
        [0.58762887],
        [0.57485433],
        [0.56006275]],

       [[0.73352757],
        [0.77364411],
        [0.77140296],
        ...,
        [0.57485433],
        [0.56006275],
        [0.51792918]],

       [[0.77364411],
        [0.77140296],
        [0.7492156 ],
        ...,
        [0.56006275],
        [0.51792918],
        [0.47758853]]])
```

In [200]:
```
closing_price = model.predict(X_test)
closing_price
```

```
       [0.6319312 ],
       [0.6289451 ],
       [0.6067512 ],
       [0.66325206],
       [0.69676346],
       [0.6969016 ],
       [0.7312305 ],
       [0.8148812 ],
       [0.8051879 ],
       [0.7916597 ],
       [0.79434216],
       [0.8193891 ],
       [0.8142036 ],
       [0.87057185],
       [0.82833827],
       [0.8581103 ],
       [0.8444979 ],
       [0.81359357],
       [0.8220129 ],
       [0.80019534],
```

In [201]:
```
closing_price = scaler.inverse_transform(closing_price)
closing_price
```

Out[201]:
```
array([[212.16121],
       [210.594  ],
       [210.08562],
```

```
[211.08871],
[214.03137],
[212.88107],
[211.49576],
[211.98015],
[208.85457],
[205.94345],
[204.29813],
[210.84212],
[208.6133 ],
[211.53862],
[223.19499],
[232.01456],
[231.95163],
[230.8679 ],
[228.68845],
```

In [202]:
```
#tells how much our predicted value is away from line of best fit
rms=np.sqrt(np.mean(np.power((test-closing_price),2)))
rms
```

Out[202]:
```
9.123841630467266
```

In [203]:
```
#for plotting
trainp = new_data[:987]
testp = new_data[987:]
testp['Predictions'] = closing_price
plt.plot(trainp['Close'])
plt.plot(testp[['Close','Predictions']])
```

```
C:\ProgramData\Anaconda3\envs\myenv\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
```

Out[203]:
```
[<matplotlib.lines.Line2D at 0x280f8735748>,
 <matplotlib.lines.Line2D at 0x280f8735a20>]
```

In [ ]: