In [1]:

```
import pandas as pd
```

In [2]:

```
data = pd.read_csv('Google_Stock_Price_Train.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	Date	Open	High	Low	Close	Volume
0	01-03-2012	325.25	332.83	324.97	663.59	73,80,500
1	01-04-2012	331.27	333.87	329.08	666.45	57,49,400
2	01-05-2012	329.83	330.75	326.89	657.21	65,90,300
3	01-06-2012	328.34	328.77	323.68	648.24	54,05,900
4	01-09-2012	322.04	322.29	309.46	620.76	1,16,88,800

In [4]:

```
data.tail()
```

Out[4]:

	Date	Open	High	Low	Close	Volume
1253	12/23/2016	790.90	792.74	787.28	789.91	6,23,400
1254	12/27/2016	790.68	797.86	787.66	791.55	7,89,100
1255	12/28/2016	793.70	794.23	783.20	785.05	11,53,800
1256	12/29/2016	783.33	785.93	778.92	782.79	7,44,300
1257	12/30/2016	782.75	782.78	770.41	771.82	17,70,000

In [5]:

```
data.shape
```

Out[5]:

```
(1258, 6)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume'], dtype='object')
```

In [7]:

```
data.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    1258 non-null   object 
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object 
 5   Volume  1258 non-null   object 
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

In [10]:

```
data.describe()
```

Out[10]:

	Open	High	Low
count	1258.000000	1258.000000	1258.000000
mean	533.709833	537.880223	529.007409
std	151.904442	153.008811	150.552807
min	279.120000	281.210000	277.220000
25%	404.115000	406.765000	401.765000
50%	537.470000	540.750000	532.990000
75%	654.922500	662.587500	644.800000
max	816.680000	816.680000	805.140000

In [11]:

```
data.nunique()
```

Out[11]:

```
Date      1258
Open      1215
High      1219
Low       1223
Close     1241
Volume    1240
dtype: int64
```

In [12]:

```
data.corr()
```

Out[12]:

	Open	High	Low
Open	1.000000	0.999692	0.999498
High	0.999692	1.000000	0.999480
Low	0.999498	0.999480	1.000000

In [13]:

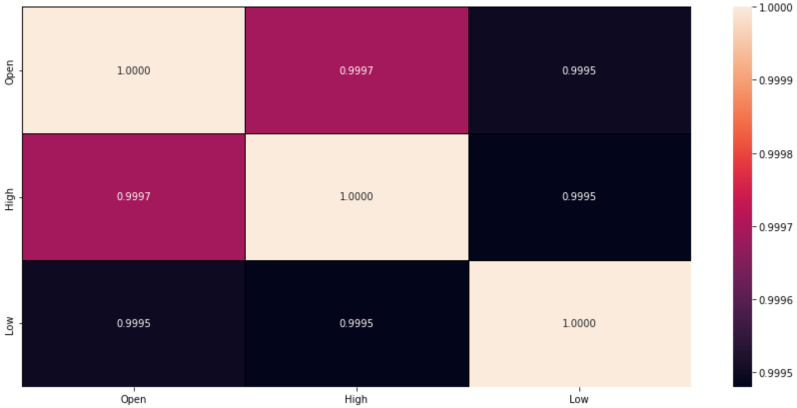
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [15]:

```
f, ax = plt.subplots(figsize = (15,7))
sns.heatmap(data.corr(), annot = True, linewidths=0.5, linecolor = "black",
            fmt = ".4f", ax = ax)
plt.show()
```



In [16]:

```
data_set = data.loc[:, ["Open"]].values
data_set
```

Out[16]:

```
array([[325.25],
       [331.27],
       [329.83],
       ...,
       [793.7 ],
       [783.33],
       [782.75]])
```

In [17]:

```
train = data_set[:len(data_set) - 50]
test = data_set[len(train):]

train.reshape(train.shape[0],1)
train.shape
```

Out[17]:

```
(1208, 1)
```

In [18]:

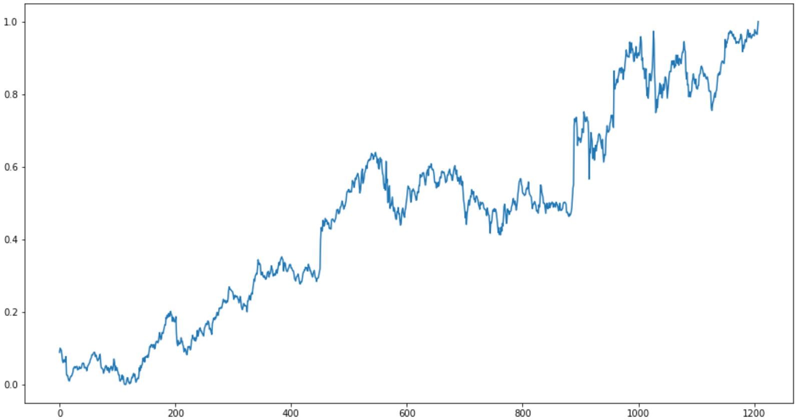
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
train_scaler = scaler.fit_transform(train)
train_scaler
```

Out[18]:

```
array([[0.08886192],
       [0.10045847],
       [0.09768454],
       ...,
       [0.96447835],
       [0.97998536],
       [1.        ]])
```

In [19]:

```
plt.figure(figsize=[15,8],)
plt.plot(train_scaler)
plt.show()
```



In [20]:

```
import numpy as np
```

In [21]:

```
X_train = []
Y_train = []
timesteps = 50

for i in range(timesteps, len(train_scaler)):
    X_train.append(train_scaler[i - timesteps:i, 0])
    Y_train.append(train_scaler[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))
```

In [22]:

```
X_train.shape
```

Out[22]:

```
(1158, 50, 1)
```

In [23]:

```
X_train
```

Out[23]:

```
array([[[0.08886192],
        [0.10045847],
        [0.09768454],
        ...,
        [0.03806442],
        [0.04646325],
        [0.05245415]],

       [[0.10045847],
        [0.09768454],
        [0.0948143 ],
        ...,
        [0.04646325],
        [0.05245415],
        [0.05399522]],

       [[0.09768454],
        [0.0948143 ],
        [0.08267838],
        ...,
        [0.05245415],
        [0.05399522],
        [0.05811758]],

       ...,

       [[0.9528818 ],
        [0.96871629],
        [0.96698259],
        ...,
        [0.97210664],
        [0.96721375],
        [0.96804207]],

       [[0.96871629],
        [0.96698259],
        [0.97208738],
        ...,
        [0.96721375],
        [0.96804207],
        [0.96447835]],

       [[0.96698259],
        [0.97208738],
        [0.9744953 ],
        ...,
        [0.96804207],
        [0.96447835],
        [0.97998536]]])
```

In [24]:

```
Y_train.shape
```

Out[24]:

```
(1158,)
```

In [25]:

```
Y_train
```

Out[25]:

```
array([0.05399522, 0.05811758, 0.06025582, ..., 0.96447835, 0.97998536,
       1.        ])
```

In [26]:

```
# Import Library
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN, Dropout

# Initialising the RNN
regressor = Sequential()

# Add the first RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True, input_shap
regressor.add(Dropout(0.2))

# Second RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
regressor.add(Dropout(0.2))

# Third RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
regressor.add(Dropout(0.2))

# Fourth RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))

# Add the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = "adam", loss = "mean_squared_error")

# Fitting the RNN to the training set
regressor.fit(X_train, Y_train, epochs = 100, batch_size = 32)
```

```
Epoch 44/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0079
Epoch 45/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0074
Epoch 46/100
37/37 [==============================] - 1s 27ms/step - loss: 0.0070
Epoch 47/100
37/37 [==============================] - 1s 29ms/step - loss: 0.0070
Epoch 48/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0064
Epoch 49/100
37/37 [==============================] - 1s 27ms/step - loss: 0.0065
Epoch 50/100
37/37 [==============================] - 1s 27ms/step - loss: 0.0063
Epoch 51/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0060
Epoch 52/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0057
Epoch 53/100
37/37 [==============================] - 1s 28ms/step - loss: 0.0059
```

In [27]:

```
inputs = data_set[len(data_set) - len(test) - timesteps:]
inputs = scaler.transform(inputs)
```

In [28]:

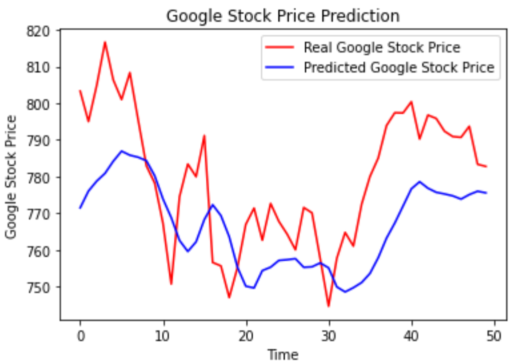
```
X_test = []
for i in range(timesteps, inputs.shape[0]):
    X_test.append(inputs[i - timesteps:i, 0])

X_test_rnn = np.array(X_test)
X_test_rnn = np.reshape(X_test_rnn, (X_test_rnn.shape[0], X_test_rnn.shape[1], 1))
predicted_stock_price = regressor.predict(X_test_rnn)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

```
2/2 [==============================] - 0s 23ms/step
```

In [29]:

```
# visualising the results
plt.plot(test, color = "red", label = "Real Google Stock Price")
plt.plot(predicted_stock_price, color = "blue", label = "Predicted Google Stock Price")
plt.title("Google Stock Price Prediction")
plt.xlabel("Time")
plt.ylabel("Google Stock Price")
plt.legend()
plt.show()
```



In [30]:

```
trainX = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
```

In [31]:

```
import math
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

In [32]:

```
model = Sequential()
model.add(LSTM(10, input_shape = (1, timesteps)))
model.add(Dense(1))
model.compile(loss = "mean_squared_error", optimizer = "adam")
model.fit(trainX, Y_train, epochs = 50, batch_size = 1)
```

```
Epoch 1/50
1158/1158 [==============================] - 5s 3ms/step - loss: 0.0068
Epoch 2/50
1158/1158 [==============================] - 4s 3ms/step - loss: 0.0012
Epoch 3/50
1158/1158 [==============================] - 3s 3ms/step - loss: 0.0011
Epoch 4/50
1158/1158 [==============================] - 4s 3ms/step - loss: 9.5732e-04
Epoch 5/50
1158/1158 [==============================] - 4s 3ms/step - loss: 9.9990e-04
Epoch 6/50
1158/1158 [==============================] - 4s 3ms/step - loss: 9.5506e-04
Epoch 7/50
1158/1158 [==============================] - 4s 3ms/step - loss: 8.9754e-04
Epoch 8/50
1158/1158 [==============================] - 4s 3ms/step - loss: 7.6320e-04
Epoch 9/50
1158/1158 [==============================] - 4s 3ms/step - loss: 7.9522e-04
Epoch 10/50
1158/1158 [==============================] - 4s 3ms/step - loss: 7.6238e-04
Epoch 11/50
1158/1158 [==============================] - 4s 3ms/step - loss: 8.3610e-04
Epoch 12/50
1158/1158 [==============================] - 4s 3ms/step - loss: 7.2668e-04
Epoch 13/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.8871e-04
Epoch 14/50
1158/1158 [==============================] - 3s 3ms/step - loss: 7.0677e-04
Epoch 15/50
1158/1158 [==============================] - 3s 3ms/step - loss: 6.7835e-04
Epoch 16/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.7848e-04
Epoch 17/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.7202e-04
Epoch 18/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.3054e-04
Epoch 19/50
1158/1158 [==============================] - 3s 3ms/step - loss: 6.3273e-04
Epoch 20/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.8427e-04
Epoch 21/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.1246e-04
Epoch 22/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.5473e-04
Epoch 23/50
1158/1158 [==============================] - 3s 3ms/step - loss: 6.4019e-04
Epoch 24/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.5982e-04
Epoch 25/50
1158/1158 [==============================] - 3s 3ms/step - loss: 6.2733e-04
Epoch 26/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.9205e-04
Epoch 27/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.9020e-04
Epoch 28/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.6921e-04
Epoch 29/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.5375e-04
Epoch 30/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.6322e-04
Epoch 31/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.4449e-04
```

```
Epoch 32/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.5258e-04
Epoch 33/50
1158/1158 [==============================] - 4s 3ms/step - loss: 6.0507e-04
Epoch 34/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.6692e-04
Epoch 35/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.2528e-04
Epoch 36/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.6085e-04
Epoch 37/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.6375e-04
Epoch 38/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.5883e-04
Epoch 39/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.2316e-04
Epoch 40/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.4303e-04
Epoch 41/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.7524e-04
Epoch 42/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.2709e-04
Epoch 43/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.0030e-04
Epoch 44/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.4262e-04
Epoch 45/50
1158/1158 [==============================] - 4s 3ms/step - loss: 4.8127e-04
Epoch 46/50
1158/1158 [==============================] - 4s 3ms/step - loss: 5.1596e-04
Epoch 47/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.3422e-04
Epoch 48/50
1158/1158 [==============================] - 3s 3ms/step - loss: 4.9169e-04
Epoch 49/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.7332e-04
Epoch 50/50
1158/1158 [==============================] - 3s 3ms/step - loss: 5.1144e-04
```

Out[32]:

```
<keras.callbacks.History at 0x166380ecbb0>
```

In [33]:

```
testX = np.array(X_test)
testX = testX.reshape(testX.shape[0], 1, testX.shape[1])

predict_lstm = model.predict(testX)
predict_lstm = scaler.inverse_transform(predict_lstm)
```

```
2/2 [==============================] - 0s 8ms/step
```

In [34]:

```
plt.plot(test, color = "red", label = "Real Google Stock Price")
plt.plot(predict_lstm, color = "blue", label = "Predicted Google Stock Price")
plt.title("Google Stock Price Prediction")
plt.xlabel("Time")
plt.ylabel("Google Stock Price")
plt.legend()
plt.show()
```