In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import datetime
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.metrics import r2_score
```

In [2]:

```
data = pd.read_csv('Google_Stock_Price_Train.csv',thousands=',')
print(data.head(10))
data.shape
```

```
        Date    Open    High     Low   Close    Volume
0   1/3/2012  325.25  332.83  324.97  663.59   7380500
1   1/4/2012  331.27  333.87  329.08  666.45   5749400
2   1/5/2012  329.83  330.75  326.89  657.21   6590300
3   1/6/2012  328.34  328.77  323.68  648.24   5405900
4   1/9/2012  322.04  322.29  309.46  620.76  11688800
5  1/10/2012  313.70  315.72  307.30  621.43   8824000
6  1/11/2012  310.59  313.52  309.40  624.25   4817800
7  1/12/2012  314.43  315.26  312.08  627.92   3764400
8  1/13/2012  311.96  312.30  309.37  623.28   4631800
9  1/17/2012  314.81  314.81  311.67  626.86   3832800
```

Out[2]:

```
(1258, 6)
```

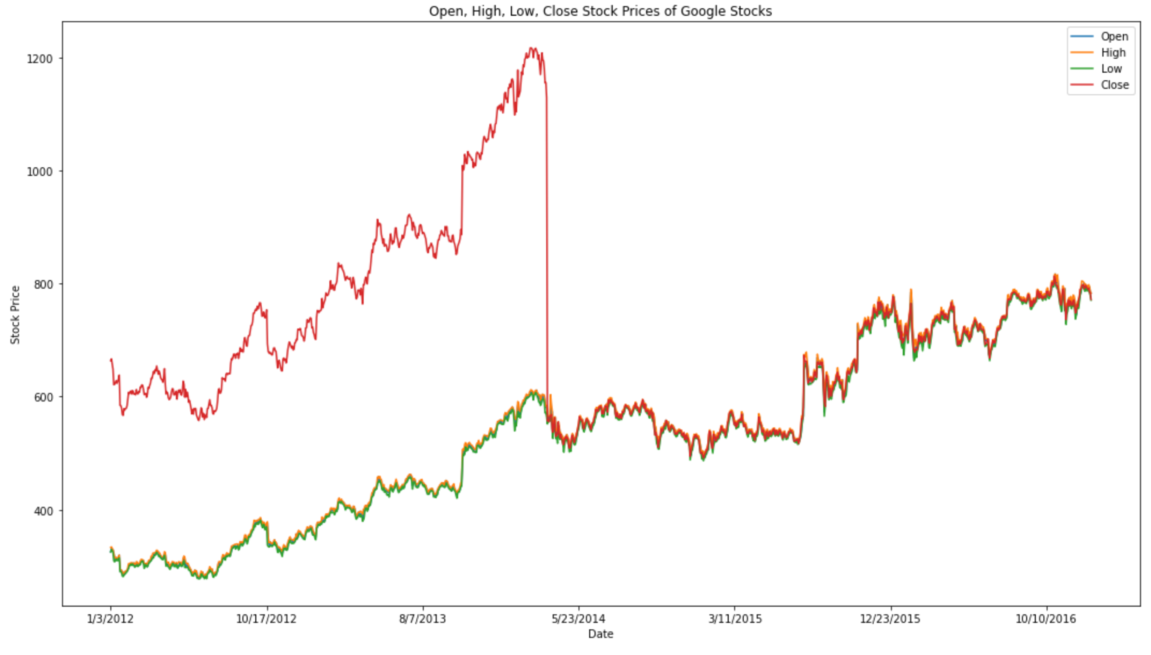
In [3]:

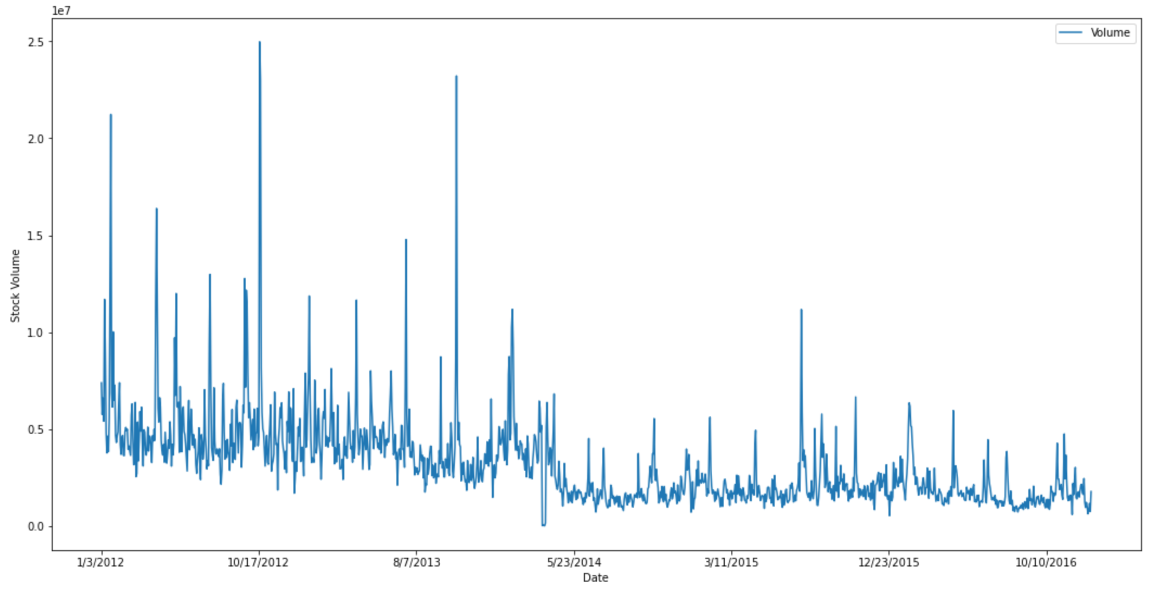
```
ax1 = data.plot(x="Date", y=["Open", "High", "Low", "Close"],  figsize=(18,10),title='Op
ax1.set_ylabel("Stock Price")

ax2 = data.plot(x="Date", y=["Volume"],  figsize=(18,9))
ax2.set_ylabel("Stock Volume")
```

Out[3]:

```
Text(0, 0.5, 'Stock Volume')
```



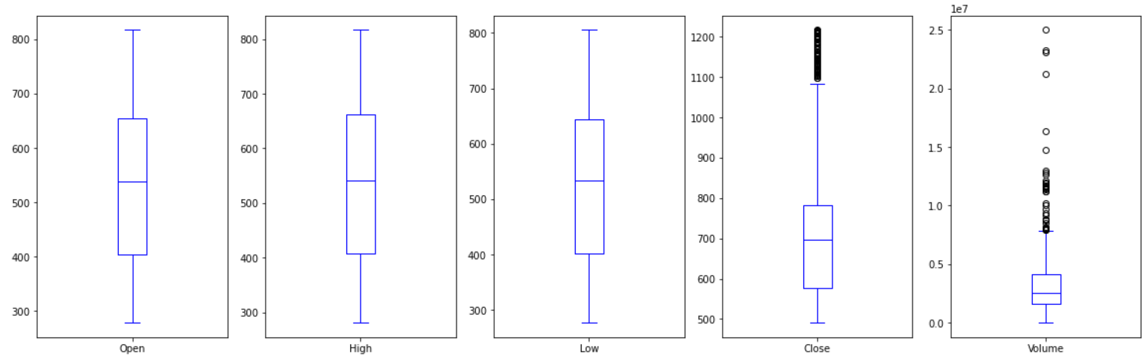
In [4]:

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

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

In [5]:

```
data[['Open','High','Low','Close','Volume']].plot(kind= 'box' ,layout=(1,5),subplots=Tru
plt.show()
```

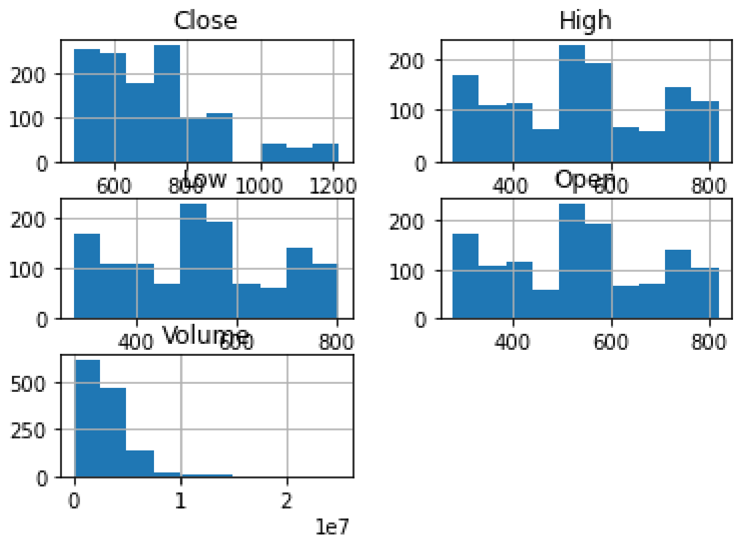


In [6]:

```python
data.hist()
```

Out[6]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FCE55BE
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD5E2C1
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD619F1
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD6523A
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD67E7F
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD6ACCA
0>]],
      dtype=object)
```




In [7]:

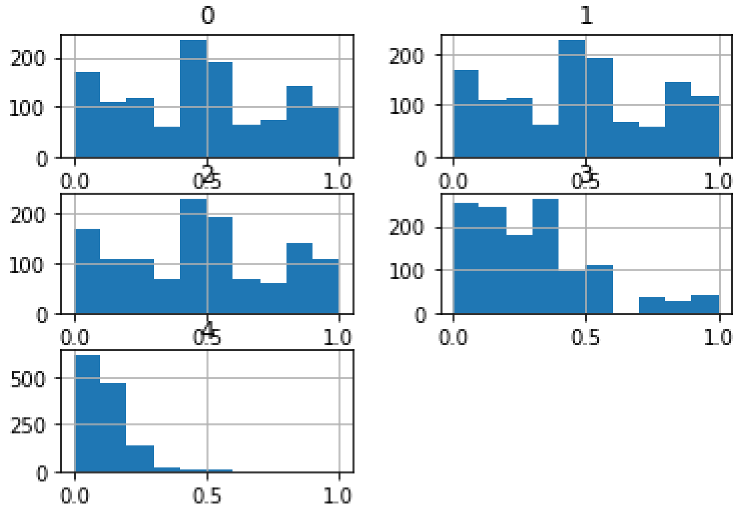
```python
scaler = MinMaxScaler()
data_without_date = data[['Open','High','Low','Close','Volume']]
data_scaled = pd.DataFrame(scaler.fit_transform(data_without_date))
```

In [8]:

```
data_scaled.hist()
```

Out[8]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FCA7146
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD7D36D
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD8454F
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD86EC7
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD8A343
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000204FD8CDC1
0>]],
      dtype=object)
```

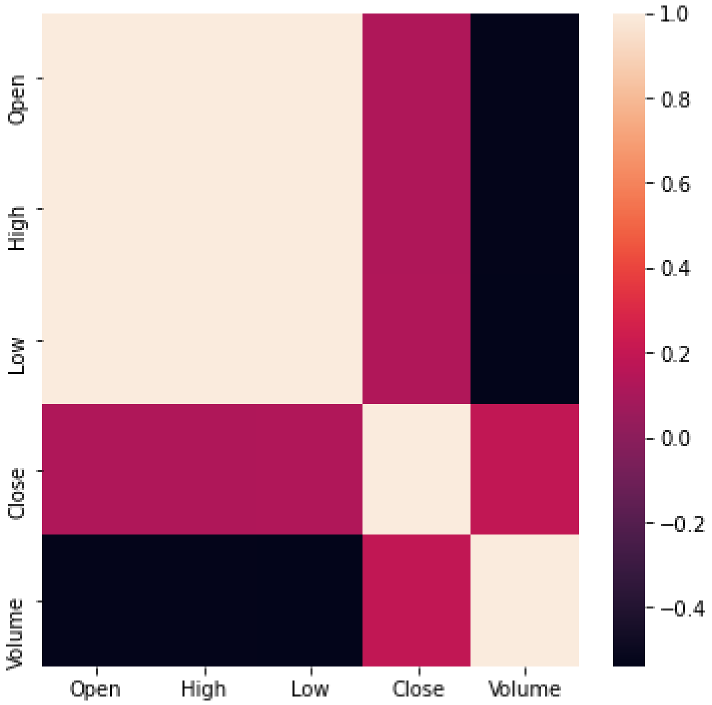


In [9]:

```
plt.figure(figsize=(6,6))
sns.heatmap(data.corr())
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x204f96e20a0>
```




In [10]:

```
data_scaled=data_scaled.drop([0,2,3], axis=1)
data_scaled
```

Out[10]:

| | 1 | 4 |
|---|---|---|
| **0** | 0.096401 | 0.295258 |
| **1** | 0.098344 | 0.229936 |
| **2** | 0.092517 | 0.263612 |
| **3** | 0.088819 | 0.216179 |
| **4** | 0.076718 | 0.467797 |
| **...** | ... | ... |
| **1253** | 0.955292 | 0.024650 |
| **1254** | 0.964853 | 0.031286 |
| **1255** | 0.958074 | 0.045891 |
| **1256** | 0.942574 | 0.029491 |
| **1257** | 0.936691 | 0.070569 |

1258 rows × 2 columns

In [11]:

```
def split_seq_multivariate(sequence, n_past, n_future):

    '''
    n_past ==> no of past observations
    n_future ==> no of future observations
    '''
    x, y = [], []
    for window_start in range(len(sequence)):
        past_end = window_start + n_past
        future_end = past_end + n_future
        if future_end > len(sequence):
            break
        # slicing the past and future parts of the window
        past   = sequence[window_start:past_end, :]
        future = sequence[past_end:future_end, -1]
        x.append(past)
        y.append(future)

    return np.array(x), np.array(y)
```

In [12]:

```
n_steps = 60

data_scaled = data_scaled.to_numpy()
data_scaled.shape
```

Out[12]:

```
(1258, 2)
```

In [13]:

```
X, y = split_seq_multivariate(data_scaled, n_steps,1)
```

In [14]:

```
print(X.shape)
print(y.shape)

# make y to the shape of [samples]
y=y[:,0]
y.shape
```

```
(1198, 60, 2)
(1198, 1)
```

Out[14]:

```
(1198,)
```

In [15]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=50)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(958, 60, 2) (240, 60, 2) (958,) (240,)
```

In [16]:

```
X_train, X_val, y_train, y_val = train_test_split(X_train,y_train,test_size=0.2,random_s

print(X_train.shape, X_val.shape, y_train.shape, y_val.shape)
```

```
(766, 60, 2) (192, 60, 2) (766,) (192,)
```

In [17]:

```
model = Sequential()
model.add(LSTM(612, input_shape=(n_steps,2)))
model.add(Dense(50, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(1))
```

In [18]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 612)               1505520   
                                                                 
 dense (Dense)               (None, 50)                30650     
                                                                 
 dense_1 (Dense)             (None, 50)                2550      
                                                                 
 dense_2 (Dense)             (None, 30)                1530      
                                                                 
 dense_3 (Dense)             (None, 1)                 31        
                                                                 
=================================================================
Total params: 1,540,281
Trainable params: 1,540,281
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

In [20]:

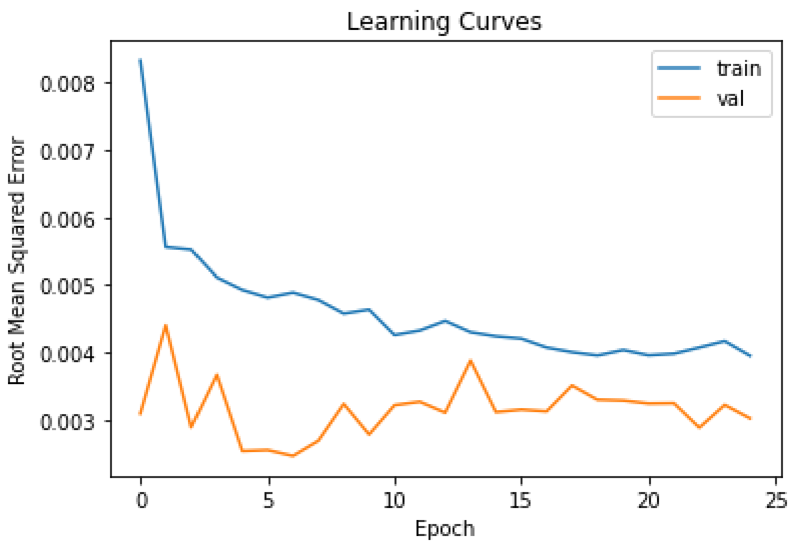
```
history = model.fit(X_train, y_train, epochs=25, batch_size=32, verbose=2, validation_da
```

```
Epoch 1/25
24/24 - 25s - loss: 0.0083 - mae: 0.0530 - val_loss: 0.0031 - val_mae: 0.0
382 - 25s/epoch - 1s/step
Epoch 2/25
24/24 - 18s - loss: 0.0056 - mae: 0.0415 - val_loss: 0.0044 - val_mae: 0.0
515 - 18s/epoch - 764ms/step
Epoch 3/25
24/24 - 19s - loss: 0.0055 - mae: 0.0418 - val_loss: 0.0029 - val_mae: 0.0
347 - 19s/epoch - 774ms/step
Epoch 4/25
24/24 - 18s - loss: 0.0051 - mae: 0.0391 - val_loss: 0.0037 - val_mae: 0.0
417 - 18s/epoch - 748ms/step
Epoch 5/25
24/24 - 18s - loss: 0.0049 - mae: 0.0377 - val_loss: 0.0025 - val_mae: 0.0
367 - 18s/epoch - 747ms/step
Epoch 6/25
24/24 - 18s - loss: 0.0048 - mae: 0.0377 - val_loss: 0.0025 - val_mae: 0.0
332 - 18s/epoch - 744ms/step
Epoch 7/25
24/24 - 18s - loss: 0.0049 - mae: 0.0383 - val_loss: 0.0025 - val_mae: 0.0
326 - 18s/epoch - 753ms/step
Epoch 8/25
24/24 - 18s - loss: 0.0048 - mae: 0.0367 - val_loss: 0.0027 - val_mae: 0.0
371 - 18s/epoch - 758ms/step
Epoch 9/25
24/24 - 18s - loss: 0.0046 - mae: 0.0374 - val_loss: 0.0032 - val_mae: 0.0
352 - 18s/epoch - 756ms/step
Epoch 10/25
24/24 - 18s - loss: 0.0046 - mae: 0.0389 - val_loss: 0.0028 - val_mae: 0.0
322 - 18s/epoch - 749ms/step
Epoch 11/25
24/24 - 18s - loss: 0.0043 - mae: 0.0353 - val_loss: 0.0032 - val_mae: 0.0
348 - 18s/epoch - 749ms/step
Epoch 12/25
24/24 - 18s - loss: 0.0043 - mae: 0.0360 - val_loss: 0.0033 - val_mae: 0.0
382 - 18s/epoch - 750ms/step
Epoch 13/25
24/24 - 18s - loss: 0.0045 - mae: 0.0391 - val_loss: 0.0031 - val_mae: 0.0
343 - 18s/epoch - 749ms/step
Epoch 14/25
24/24 - 18s - loss: 0.0043 - mae: 0.0361 - val_loss: 0.0039 - val_mae: 0.0
380 - 18s/epoch - 742ms/step
Epoch 15/25
24/24 - 18s - loss: 0.0042 - mae: 0.0376 - val_loss: 0.0031 - val_mae: 0.0
336 - 18s/epoch - 744ms/step
Epoch 16/25
24/24 - 18s - loss: 0.0042 - mae: 0.0363 - val_loss: 0.0031 - val_mae: 0.0
331 - 18s/epoch - 742ms/step
Epoch 17/25
24/24 - 18s - loss: 0.0041 - mae: 0.0341 - val_loss: 0.0031 - val_mae: 0.0
336 - 18s/epoch - 748ms/step
Epoch 18/25
24/24 - 18s - loss: 0.0040 - mae: 0.0339 - val_loss: 0.0035 - val_mae: 0.0
355 - 18s/epoch - 738ms/step
Epoch 19/25
24/24 - 18s - loss: 0.0040 - mae: 0.0337 - val_loss: 0.0033 - val_mae: 0.0
352 - 18s/epoch - 743ms/step
Epoch 20/25
24/24 - 18s - loss: 0.0040 - mae: 0.0344 - val_loss: 0.0033 - val_mae: 0.0
340 - 18s/epoch - 746ms/step
Epoch 21/25
```

```
24/24 - 18s - loss: 0.0040 - mae: 0.0336 - val_loss: 0.0032 - val_mae: 0.0
337 - 18s/epoch - 744ms/step
Epoch 22/25
24/24 - 18s - loss: 0.0040 - mae: 0.0353 - val_loss: 0.0032 - val_mae: 0.0
332 - 18s/epoch - 753ms/step
Epoch 23/25
24/24 - 18s - loss: 0.0041 - mae: 0.0360 - val_loss: 0.0029 - val_mae: 0.0
325 - 18s/epoch - 763ms/step
Epoch 24/25
24/24 - 18s - loss: 0.0042 - mae: 0.0351 - val_loss: 0.0032 - val_mae: 0.0
333 - 18s/epoch - 753ms/step
Epoch 25/25
24/24 - 18s - loss: 0.0039 - mae: 0.0337 - val_loss: 0.0030 - val_mae: 0.0
327 - 18s/epoch - 738ms/step
```

In [21]:

```
from matplotlib import pyplot
# plot learning curves
pyplot.title('Learning Curves')
pyplot.xlabel('Epoch')
pyplot.ylabel('Root Mean Squared Error')
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='val')
pyplot.legend()
pyplot.show()
```




In [22]:

```
mse, mae = model.evaluate(X_test, y_test, verbose=0)
print('MSE: %.3f, RMSE: %.3f, MAE: %.3f' % (mse, np.sqrt(mse), mae))
```

```
MSE: 0.002, RMSE: 0.049, MAE: 0.030
```

In [23]:

```
print(X_test.shape)
predicted_values = model.predict(X_test)
print(predicted_values.shape)
```

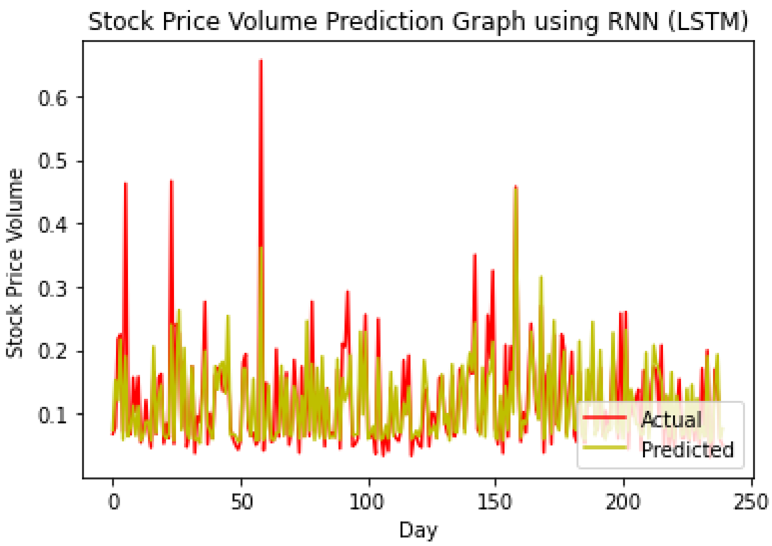
```
(240, 60, 2)
8/8 [==============================] - 3s 257ms/step
(240, 1)
```

In [24]:

```
plt.plot(y_test,c = 'r')
plt.plot(predicted_values,c = 'y')
plt.xlabel('Day')
plt.ylabel('Stock Price Volume')
plt.title('Stock Price Volume Prediction Graph using RNN (LSTM)')
plt.legend(['Actual','Predicted'],loc = 'lower right')
plt.figure(figsize=(10,6))
```

Out[24]:

```
<Figure size 720x432 with 0 Axes>
```




```
<Figure size 720x432 with 0 Axes>
```

In [25]:

```
R_square = r2_score(y_test, predicted_values)

print(R_square)
```

```
0.6469417657548717
```