In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers,Sequential
from tensorflow.keras.layers import Dense,LSTM, Dropout
import matplotlib.pyplot as plt
```

In [2]:

```
#reading the given data set into our data frame named original_dataset
original_dataset = pd.read_csv("./q2_dataset.csv")
```

In [3]:

```
original_dataset
```

Out[3]:

	Date	Close/Last	Volume	Open	High	Low
0	07/08/20	$381.37	29272970	376.72	381.50	376.36
1	07/07/20	$372.69	28106110	375.41	378.62	372.23
2	07/06/20	$373.85	29663910	370.00	375.78	369.87
3	07/02/20	$364.11	28510370	367.85	370.47	363.64
4	07/01/20	$364.11	27684310	365.12	367.36	363.91
...	...	...	...	...	...	...
1254	07/15/2015	$126.82	33559770	125.72	127.15	125.58
1255	07/14/2015	$125.61	31695870	126.04	126.37	125.04
1256	07/13/2015	$125.66	41365600	125.03	125.76	124.32
1257	07/10/15	$123.28	61292800	121.94	123.85	121.21
1258	07/09/15	$120.07	78291510	123.85	124.06	119.22

1259 rows × 6 columns

In [4]:

```
#chagining the format of the date column for further sorting in ascendi
original_dataset['Date'] = pd.to_datetime(original_dataset.Date)
original_dataset = original_dataset.sort_values(by='Date')
original_dataset
```

Out[4]:

	Date	Close/Last	Volume	Open	High	Low
1258	2015-07-09	$120.07	78291510	123.85	124.06	119.22
1257	2015-07-10	$123.28	61292800	121.94	123.85	121.21
1256	2015-07-13	$125.66	41365600	125.03	125.76	124.32
1255	2015-07-14	$125.61	31695870	126.04	126.37	125.04
1254	2015-07-15	$126.82	33559770	125.72	127.15	125.58
...	...	...	...	...	...	...
4	2020-07-01	$364.11	27684310	365.12	367.36	363.91
3	2020-07-02	$364.11	28510370	367.85	370.47	363.64
2	2020-07-06	$373.85	29663910	370.00	375.78	369.87
1	2020-07-07	$372.69	28106110	375.41	378.62	372.23
0	2020-07-08	$381.37	29272970	376.72	381.50	376.36

1259 rows × 6 columns

In [5]:

```
#creating an empty array of shape 1258x13
new_df=np.zeros((1258,13))
new_df.shape
```

Out[5]:

```
(1258, 13)
```

In [6]:

```python
# Iterating through the for loop in order to create a a dataset in whic
# the open of the next day is calculated using the past 3 days Open, Hi

for i in range (len(new_df)-2):

    new_df[i][12] = original_dataset.iloc[i+3][3]

    new_df[i][0] = original_dataset.iloc[i+1][3]
    new_df[i][1] = original_dataset.iloc[i+2][3]
    new_df[i][2] = original_dataset.iloc[i][3]

    new_df[i][3] = original_dataset.iloc[i+2][4]
    new_df[i][4] = original_dataset.iloc[i+1][4]
    new_df[i][5] = original_dataset.iloc[i][4]

    new_df[i][6] = original_dataset.iloc[i+2][5]
    new_df[i][7] = original_dataset.iloc[i+1][5]
    new_df[i][8] = original_dataset.iloc[i][5]

    new_df[i][9] = original_dataset.iloc[i+2][2]
    new_df[i][10] = original_dataset.iloc[i+1][2]
    new_df[i][11] = original_dataset.iloc[i][2]
```

In [7]:

```python
column_names = ['Open_day1','Open_day2','Open_day3','High_day1','High_d
threeday_df = pd.DataFrame(new_df[:-2,:],columns=column_names)
```

In [8]:

```python
threeday_df
```

Out[8]:

	Open_day1	Open_day2	Open_day3	High_day1	High_day2	High_day3	Low_day1	Low_day2
0	121.94	125.03	123.85	125.76	123.85	124.06	124.32	121.21
1	125.03	126.04	121.94	126.37	125.76	123.85	125.04	124.32
2	126.04	125.72	125.03	127.15	126.37	125.76	125.58	125.04
3	125.72	127.74	126.04	128.57	127.15	126.37	127.35	125.58
4	127.74	129.08	125.72	129.62	128.57	127.15	128.31	127.35
...	...	...	...	...	...	...	...	...
1251	353.25	360.08	364.41	365.98	362.17	365.32	360.00	351.28
1252	360.08	365.12	353.25	367.36	365.98	362.17	363.91	360.00
1253	365.12	367.85	360.08	370.47	367.36	365.98	363.64	363.91
1254	367.85	370.00	365.12	375.78	370.47	367.36	369.87	363.64
1255	370.00	375.41	367.85	378.62	375.78	370.47	372.23	369.87

1256 rows × 13 columns

In [9]:

```
df_dropped_target = threeday_df.drop(['Target'],axis=1)
df_dropped_target
```

Out[9]:

	Open_day1	Open_day2	Open_day3	High_day1	High_day2	High_day3	Low_day1	Low_day2
0	121.94	125.03	123.85	125.76	123.85	124.06	124.32	121.21
1	125.03	126.04	121.94	126.37	125.76	123.85	125.04	124.32
2	126.04	125.72	125.03	127.15	126.37	125.76	125.58	125.04
3	125.72	127.74	126.04	128.57	127.15	126.37	127.35	125.58
4	127.74	129.08	125.72	129.62	128.57	127.15	128.31	127.35
...	...	...	...	...	...	...	...	...
1251	353.25	360.08	364.41	365.98	362.17	365.32	360.00	351.28
1252	360.08	365.12	353.25	367.36	365.98	362.17	363.91	360.00
1253	365.12	367.85	360.08	370.47	367.36	365.98	363.64	363.91
1254	367.85	370.00	365.12	375.78	370.47	367.36	369.87	363.64
1255	370.00	375.41	367.85	378.62	375.78	370.47	372.23	369.87

1256 rows × 12 columns

In [10]:

```
#splitting the dataset into 70% training and 30% Testing
x_train, x_test, y_train, y_test = train_test_split(df_dropped_target,
```

In [11]:

```
Train_Dataset = pd.concat([x_train,y_train],axis=1)
Train_Dataset
```

Out[11]:

	Open_day1	Open_day2	Open_day3	High_day1	High_day2	High_day3	Low_day1	Low_day2
388	119.55	120.42	120.00	122.10	120.10	120.81	120.28	119.50
1157	324.19	324.74	321.47	325.98	326.22	327.22	322.85	323.35
299	113.86	115.12	108.73	116.13	115.73	113.03	114.04	113.49
1133	307.24	310.60	297.16	312.67	310.43	304.44	308.25	306.20
583	167.90	169.87	163.89	169.94	169.65	168.07	165.61	166.94
...	...	...	...	...	...	...	...	...
555	155.80	152.02	157.90	152.27	155.80	158.26	150.56	152.75
289	105.66	106.14	105.80	106.80	106.57	106.50	105.62	105.64
1001	201.41	203.28	203.17	204.44	203.13	204.49	202.69	201.36
452	144.47	143.92	143.91	144.16	144.60	144.90	143.31	143.38
506	145.50	147.97	145.87	149.33	148.49	146.18	147.33	145.44

879 rows × 13 columns

In [12]:

```
#creating the training dataset
Train_Dataset.to_csv(r'./Train_Dataset_Rnn.csv', index = False, header=
```

In [13]:

```
y_test = y_test.reindex(x_test.index)
Test_Dataset=pd.concat([x_test,y_test],axis=1)
Test_Dataset
```

Out[13]:

	Open_day1	Open_day2	Open_day3	High_day1	High_day2	High_day3	Low_day1	Low_day2
477	153.17	153.58	153.97	155.45	153.33	154.17	152.89	152.22
1054	219.96	221.06	217.73	222.85	220.82	220.13	219.44	219.12
359	112.31	113.29	110.86	115.00	114.70	112.43	112.49	112.31
212	92.72	90.00	93.48	91.67	92.78	93.57	90.00	89.47
615	173.63	174.88	172.40	177.20	174.17	173.13	174.86	172.46
...	...	...	...	...	...	...	...	...
792	223.25	226.51	220.15	228.87	228.26	223.49	226.00	222.40
902	168.99	171.05	172.40	171.21	170.66	173.94	169.25	168.42
333	111.40	110.98	113.46	111.46	112.35	113.77	109.55	111.23
924	182.25	183.90	180.00	184.10	183.30	182.67	182.56	180.92
627	172.54	173.44	172.53	175.37	173.47	174.55	173.05	172.08

377 rows × 13 columns

In [14]:

```
#creating the testing dataset
Test_Dataset.to_csv(r'./Test_Dataset_Rnn.csv', index = False, header=Tr
```

In [15]:

```
#Normalizing using Min-Max scaler
scaler = MinMaxScaler(feature_range=(0,1))
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [16]:

```
#converting the data into numpy array for further reshaping
x_train,x_test = np.array(x_train), np.array(x_test)
```

In [17]:

```
#Reshaping the data from 2-Dimensions to 3-Dimesnions
x_train =x_train.reshape(x_train.shape[0],x_train.shape[1] , 1)
x_test = x_test.reshape(x_test.shape[0],x_test.shape[1] , 1)
print(f'shape of training data is : {x_train.shape}')
print(f'shape of testing data is : {x_test.shape}')

```

```
shape of training data is : (879, 12, 1)
shape of testing data is : (377, 12, 1)
```

LSTM network expects the network to be 3-Dimensional in the form of no.of samples, no.of time steps and no.of features, hence reshpaing is needed. As we can see from above, our data is 2-D, thus we need to reshape it to 3-D. We can pass the coloumn and row values to the reshape function to make it more robust.

In [18]:

```
# Building a Model
My_LSTM_model = Sequential()
#adding LSTM layer with 50 LSTM units
My_LSTM_model.add(LSTM(50,input_shape=(x_train.shape[1],1),return_seque
# My_LSTM_model.add(Dropout(0.2))
#adding LSTM layer with 150 LSTM units
My_LSTM_model.add(LSTM(150))
# My_LSTM_model.add(Dropout(0.2))
#adding dense layer
My_LSTM_model.add(Dense(1,activation='linear'))

#'mean_squared_error' has been used as loss function
# Optimizer: Here adam optimizer has been used.
# Adam is an adaptive learning rate optimization algorithm that’s been
# training deep neural networks.
My_LSTM_model.compile(loss='mean_squared_error',optimizer='adam',metric
```

```
Metal device set to: Apple M1

2022-07-08 12:41:27.174335: I tensorflow/core/common_runtime/pluggable_de
vice/pluggable_device_factory.cc:305] Could not identify NUMA node of pla
tform GPU ID 0, defaulting to 0. Your kernel may not have been built with
NUMA support.
2022-07-08 12:41:27.174596: I tensorflow/core/common_runtime/pluggable_de
vice/pluggable_device_factory.cc:271] Created TensorFlow device (/job:loc
alhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical Plugga
bleDevice (device: 0, name: METAL, pci bus id: <undefined>)
```

In [19]:

```
My_LSTM_model.summary()
```

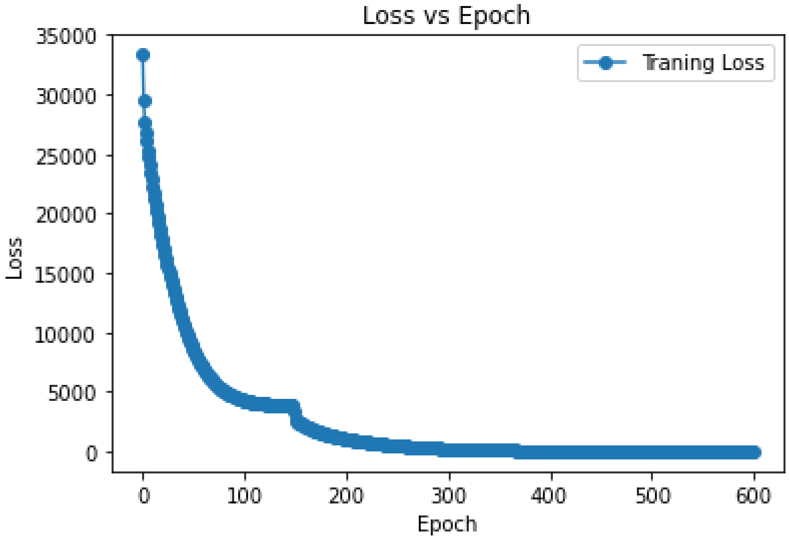
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 12, 50)            10400     
                                                                 
 lstm_1 (LSTM)               (None, 150)               120600    
                                                                 
 dense (Dense)               (None, 1)                 151       
                                                                 
=================================================================
Total params: 131,151
Trainable params: 131,151
Non-trainable params: 0
_________________________________________________________________
```

In [20]:
```
History = My_LSTM_model.fit(x_train,y_train,epochs=600,batch_size=64,ve
```

```
14/14 [==============================] - 0s 15ms/step - loss: 20.9394 - m
ae: 2.9228
Epoch 595/600
14/14 [==============================] - 0s 15ms/step - loss: 19.7900 - m
ae: 2.8910
Epoch 596/600
14/14 [==============================] - 0s 15ms/step - loss: 18.6902 - m
ae: 2.7372
Epoch 597/600
14/14 [==============================] - 0s 14ms/step - loss: 19.3814 - m
ae: 2.8125
Epoch 598/600
14/14 [==============================] - 0s 14ms/step - loss: 19.4626 - m
ae: 2.8295
Epoch 599/600
14/14 [==============================] - 0s 15ms/step - loss: 21.4228 - m
ae: 3.0346
Epoch 600/600
14/14 [==============================] - 0s 14ms/step - loss: 20.9278 - m
ae: 2.9301
```

In [21]:
```
def plot_loss(history):
  # summarize history for loss
  plt.plot(history.history['loss'],marker = 'o')
  plt.title('Loss vs Epoch')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Traning Loss'], loc='upper right')
  plt.show()
```

In [22]:
```
plot_loss(History)
```



In [23]:
```
y_test=np.array(y_test)
```

In [24]:

```
y_pred = My_LSTM_model.predict(x_test)
```

```
10/12 [========================>.....] - ETA: 0s

2022-07-08 12:43:34.166179: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.
2022-07-08 12:43:34.227043: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.
2022-07-08 12:43:34.257172: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.

12/12 [==============================] - 0s 8ms/step
```

In [25]:

```
scores=My_LSTM_model.evaluate(x_test,y_test)
```

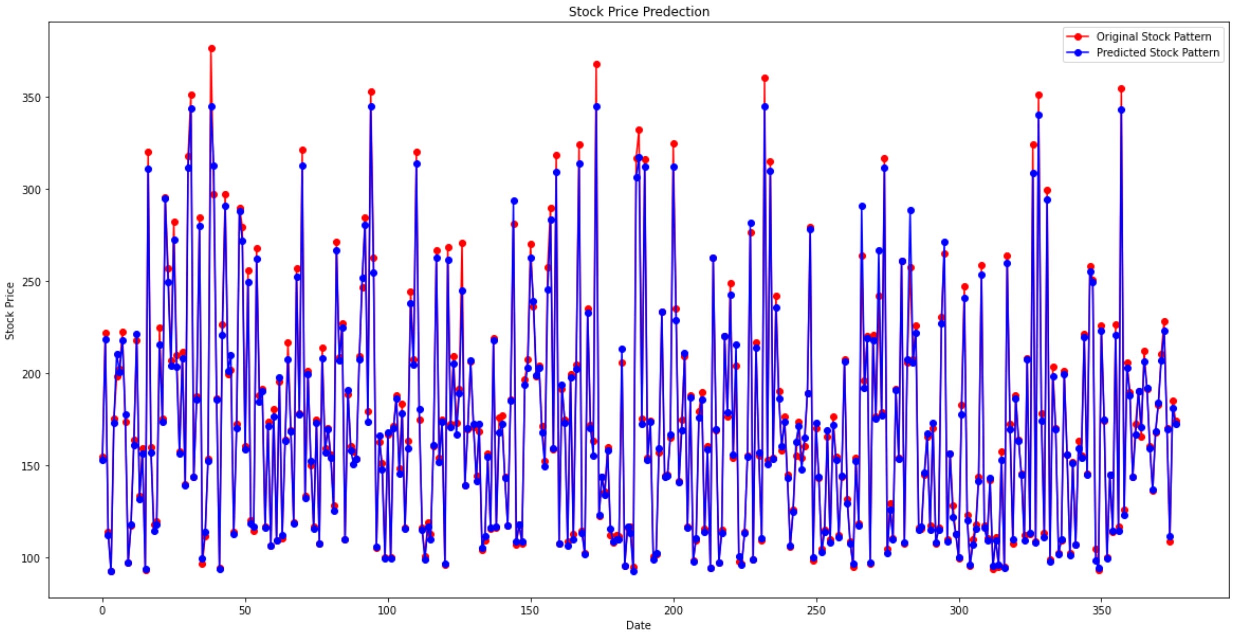
```
 8/12 [===================>..........] - ETA: 0s - loss: 26.7738 - mae:
3.3057

2022-07-08 12:43:34.647365: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.
2022-07-08 12:43:34.717793: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.
2022-07-08 12:43:34.744966: I tensorflow/core/grappler/optimizers/custom_
graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.

12/12 [==============================] - 0s 10ms/step - loss: 27.9825 - m
ae: 3.2033
```

In [26]:

```
plt.figure(figsize=(20,10))
plt.plot(y_test, color="red", marker='o', linestyle='solid', label="Ori
plt.plot(y_pred, color="blue", marker='o', linestyle='solid', label="Pr
plt.title("Stock Price Predection")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.legend()
plt.show()
```


Stock Price Predection
Original Stock Pattern
Predicted Stock Pattern
Stock Price
Date