IMPORTING LIBRARIES AND DATA TO BE USED

In [2]:
```
#importing libraries to be used
import numpy as np # for linear algebra
import pandas as pd # data preprocessing
import matplotlib.pyplot as plt # data visualization library
import seaborn as sns # data visualization library
%matplotlib inline
import warnings
warnings.filterwarnings('ignore') # ignore warnings

from sklearn.preprocessing import MinMaxScaler # for normalization
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Bidirectional
```

In [3]:
```
df = pd.read_csv('C:/Users/KAVYA/Downloads/GOOG.csv') # data_importing
df.head(10) # fetching first 10 rows of dataset
```

Out[3]:

	symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
0	GOOG	2016-06-14 00:00:00+00:00	718.27	722.47	713.1200	716.48	1306065	718.27	722.47	713.1200	716.48	1306065	0.0	1.0
1	GOOG	2016-06-15 00:00:00+00:00	718.92	722.98	717.3100	719.00	1214517	718.92	722.98	717.3100	719.00	1214517	0.0	1.0
2	GOOG	2016-06-16 00:00:00+00:00	710.36	716.65	703.2600	714.91	1982471	710.36	716.65	703.2600	714.91	1982471	0.0	1.0
3	GOOG	2016-06-17 00:00:00+00:00	691.72	708.82	688.4515	708.65	3402357	691.72	708.82	688.4515	708.65	3402357	0.0	1.0
4	GOOG	2016-06-20 00:00:00+00:00	693.71	702.48	693.4100	698.77	2082538	693.71	702.48	693.4100	698.77	2082538	0.0	1.0
5	GOOG	2016-06-21 00:00:00+00:00	695.94	702.77	692.0100	698.40	1465634	695.94	702.77	692.0100	698.40	1465634	0.0	1.0
6	GOOG	2016-06-22 00:00:00+00:00	697.46	700.86	693.0819	699.06	1184318	697.46	700.86	693.0819	699.06	1184318	0.0	1.0
7	GOOG	2016-06-23 00:00:00+00:00	701.87	701.95	687.0000	697.45	2171415	701.87	701.95	687.0000	697.45	2171415	0.0	1.0
8	GOOG	2016-06-24 00:00:00+00:00	675.22	689.40	673.4500	675.17	4449022	675.22	689.40	673.4500	675.17	4449022	0.0	1.0
9	GOOG	2016-06-27 00:00:00+00:00	668.26	672.30	663.2840	671.00	2641085	668.26	672.30	663.2840	671.00	2641085	0.0	1.0

GATHERING INSIGHTS

In [4]:
```
# shape of data
print("Shape of data:",df.shape)
```

```
Shape of data: (1258, 14)
```

In [5]:
```
# statistical description of data
df.describe()
```

Out[5]:

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume
count	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03
mean	1216.317067	1227.430934	1204.176430	1215.260779	1.601590e+06	1216.317067	1227.430936	1204.176436	1215.260779	1.601590e+06
std	383.333358	387.570872	378.777094	382.446995	6.960172e+05	383.333358	387.570873	378.777099	382.446995	6.960172e+05
min	668.260000	672.300000	663.284000	671.000000	3.467530e+05	668.260000	672.300000	663.284000	671.000000	3.467530e+05
25%	960.802500	968.757500	952.182500	959.005000	1.173522e+06	960.802500	968.757500	952.182500	959.005000	1.173522e+06
50%	1132.460000	1143.935000	1117.915000	1131.150000	1.412588e+06	1132.460000	1143.935000	1117.915000	1131.150000	1.412588e+06
75%	1360.595000	1374.345000	1348.557500	1361.075000	1.812156e+06	1360.595000	1374.345000	1348.557500	1361.075000	1.812156e+06
max	2521.600000	2526.990000	2498.290000	2524.920000	6.207027e+06	2521.600000	2526.990000	2498.290000	2524.920000	6.207027e+06

In [6]:
```
# summary of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   symbol       1258 non-null   object
 1   date         1258 non-null   object
 2   close        1258 non-null   float64
 3   high         1258 non-null   float64
 4   low          1258 non-null   float64
 5   open         1258 non-null   float64
 6   volume       1258 non-null   int64
 7   adjClose     1258 non-null   float64
 8   adjHigh      1258 non-null   float64
 9   adjLow       1258 non-null   float64
 10  adjOpen      1258 non-null   float64
 11  adjVolume    1258 non-null   int64
 12  divCash      1258 non-null   float64
 13  splitFactor  1258 non-null   float64
dtypes: float64(10), int64(2), object(2)
memory usage: 137.7+ KB
```

In [7]:
```
# checking null values
df.isnull().sum()
```

Out[7]:
```
symbol         0
date           0
close          0
high           0
low            0
open           0
volume         0
adjClose       0
adjHigh        0
adjLow         0
adjOpen        0
adjVolume      0
divCash        0
splitFactor    0
dtype: int64
```

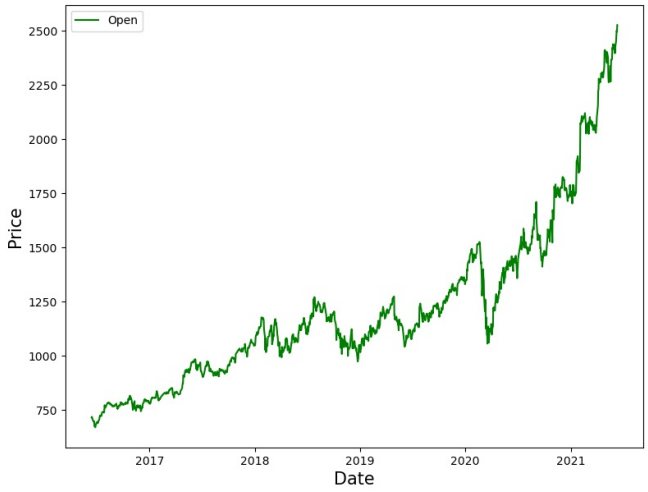
There are no null values in the dataset

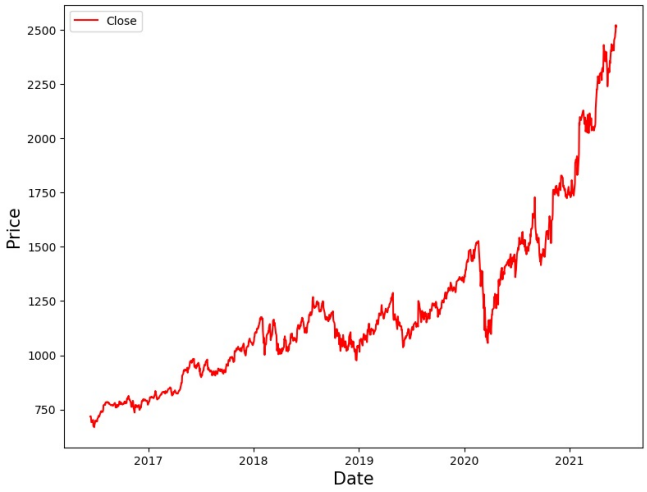
In [8]:
```
df = df[['date','open','close']] # Extracting required columns
df['date'] = pd.to_datetime(df['date'].apply(lambda x: x.split()[0])) # converting object dtype of date column
df.set_index('date',drop=True,inplace=True) # Setting date column as index
df.head(10)
```

Out[8]:

date	open	close
2016-06-14	716.48	718.27
2016-06-15	719.00	718.92
2016-06-16	714.91	710.36
2016-06-17	708.65	691.72
2016-06-20	698.77	693.71
2016-06-21	698.40	695.94
2016-06-22	699.06	697.46
2016-06-23	697.45	701.87
2016-06-24	675.17	675.22
2016-06-27	671.00	668.26

In [9]:
```
# plotting open and closing price on date index
fig, ax =plt.subplots(1,2,figsize=(20,7))
ax[0].plot(df['open'],label='Open',color='green')
ax[0].set_xlabel('Date',size=15)
ax[0].set_ylabel('Price',size=15)
ax[0].legend()

ax[1].plot(df['close'],label='Close',color='red')
ax[1].set_xlabel('Date',size=15)
ax[1].set_ylabel('Price',size=15)
ax[1].legend()

fig.show()
```





DATA PRE-PROCESSING

In [10]:
```
# normalizing all the values of all columns using MinMaxScaler
MMS = MinMaxScaler()
df[df.columns] = MMS.fit_transform(df)
df.head(10)
```

Out[10]:

date	open	close
2016-06-14	0.024532	0.026984
2016-06-15	0.025891	0.027334
2016-06-16	0.023685	0.022716
2016-06-17	0.020308	0.012658
2016-06-20	0.014979	0.013732
2016-06-21	0.014779	0.014935
2016-06-22	0.015135	0.015755
2016-06-23	0.014267	0.018135
2016-06-24	0.002249	0.003755
2016-06-27	0.000000	0.000000

In [11]:
```
# splitting the data into training and test set
training_size = round(len(df) * 0.75) # Selecting 75 % for training and 25 % for testing
training_size
```

Out[11]:
```
944
```

In [12]:
```
train_data = df[:training_size]
test_data  = df[training_size:]

train_data.shape, test_data.shape
```

Out[12]:
```
((944, 2), (314, 2))
```

In [13]:
```
# Function to create sequence of data for training and testing

def create_sequence(dataset):
  sequences = []
  labels = []

  start_idx = 0

  for stop_idx in range(50,len(dataset)): # Selecting 50 rows at a time
    sequences.append(dataset.iloc[start_idx:stop_idx])
    labels.append(dataset.iloc[stop_idx])
    start_idx += 1
  return (np.array(sequences),np.array(labels))
```

In [14]:
```
train_seq, train_label = create_sequence(train_data)
test_seq, test_label = create_sequence(test_data)
train_seq.shape, train_label.shape, test_seq.shape, test_label.shape
```

Out[14]:
```
((894, 50, 2), (894, 2), (264, 50, 2), (264, 2))
```

CREATING LSTM MODEL

In [15]:
```
# imported Sequential from keras.models
model = Sequential()
# importing Dense, Dropout, LSTM, Bidirectional from keras.layers
model.add(LSTM(units=50, return_sequences=True, input_shape = (train_seq.shape[1], train_seq.shape[2])))

model.add(Dropout(0.1))
model.add(LSTM(units=50))

model.add(Dense(2))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_error'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50, 50)            10600

 dropout (Dropout)           (None, 50, 50)            0

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 2)                 102

=================================================================
Total params: 30,902
Trainable params: 30,902
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
# fitting the model by iterating the dataset over 100 times(100 epochs)
model.fit(train_seq, train_label, epochs=100,validation_data=(test_seq, test_label), verbose=1)
```

```
Epoch 1/100
28/28 [==============================] - 13s 162ms/step - loss: 0.0084 - mean_absolute_error: 0.0648 - val_loss
: 0.0221 - val_mean_absolute_error: 0.1258
Epoch 2/100
28/28 [==============================] - 2s 76ms/step - loss: 9.1405e-04 - mean_absolute_error: 0.0240 - val_lo
ss: 0.0063 - val_mean_absolute_error: 0.0644
Epoch 3/100
28/28 [==============================] - 2s 72ms/step - loss: 4.7781e-04 - mean_absolute_error: 0.0161 - val_lo
ss: 0.0052 - val_mean_absolute_error: 0.0592
Epoch 4/100
28/28 [==============================] - 2s 73ms/step - loss: 4.6453e-04 - mean_absolute_error: 0.0158 - val_lo
ss: 0.0032 - val_mean_absolute_error: 0.0433
Epoch 5/100
28/28 [==============================] - 2s 75ms/step - loss: 4.4336e-04 - mean_absolute_error: 0.0153 - val_lo
ss: 0.0035 - val_mean_absolute_error: 0.0460
Epoch 6/100
28/28 [==============================] - 2s 76ms/step - loss: 4.2706e-04 - mean_absolute_error: 0.0151 - val_lo
ss: 0.0044 - val_mean_absolute_error: 0.0533
Epoch 7/100
28/28 [==============================] - 2s 76ms/step - loss: 4.6415e-04 - mean_absolute_error: 0.0158 - val_lo
ss: 0.0059 - val_mean_absolute_error: 0.0625
Epoch 8/100
28/28 [==============================] - 2s 77ms/step - loss: 4.1368e-04 - mean_absolute_error: 0.0149 - val_lo
ss: 0.0039 - val_mean_absolute_error: 0.0485
Epoch 9/100
28/28 [==============================] - 2s 79ms/step - loss: 4.1483e-04 - mean_absolute_error: 0.0149 - val_lo
ss: 0.0037 - val_mean_absolute_error: 0.0469
Epoch 10/100
28/28 [==============================] - 2s 78ms/step - loss: 4.0273e-04 - mean_absolute_error: 0.0147 - val_lo
ss: 0.0032 - val_mean_absolute_error: 0.0432
Epoch 11/100
28/28 [==============================] - 2s 77ms/step - loss: 4.3347e-04 - mean_absolute_error: 0.0153 - val_lo
ss: 0.0062 - val_mean_absolute_error: 0.0650
Epoch 12/100
28/28 [==============================] - 2s 78ms/step - loss: 3.8743e-04 - mean_absolute_error: 0.0145 - val_lo
ss: 0.0054 - val_mean_absolute_error: 0.0592
Epoch 13/100
28/28 [==============================] - 2s 78ms/step - loss: 3.8162e-04 - mean_absolute_error: 0.0142 - val_lo
ss: 0.0053 - val_mean_absolute_error: 0.0587
Epoch 14/100
28/28 [==============================] - 2s 78ms/step - loss: 3.6166e-04 - mean_absolute_error: 0.0137 - val_lo
ss: 0.0031 - val_mean_absolute_error: 0.0419
Epoch 15/100
28/28 [==============================] - 2s 78ms/step - loss: 3.6923e-04 - mean_absolute_error: 0.0141 - val_lo
ss: 0.0053 - val_mean_absolute_error: 0.0578
Epoch 16/100
28/28 [==============================] - 2s 78ms/step - loss: 3.6820e-04 - mean_absolute_error: 0.0142 - val_lo
ss: 0.0070 - val_mean_absolute_error: 0.0684
Epoch 17/100
28/28 [==============================] - 2s 78ms/step - loss: 3.6576e-04 - mean_absolute_error: 0.0140 - val_lo
ss: 0.0053 - val_mean_absolute_error: 0.0589
Epoch 18/100
```

```
28/28 [==============================] - 2s 77ms/step - loss: 3.4324e-04 - mean_absolute_error: 0.0135 - val_lo
ss: 0.0064 - val_mean_absolute_error: 0.0648
Epoch 19/100
28/28 [==============================] - 2s 78ms/step - loss: 3.2163e-04 - mean_absolute_error: 0.0131 - val_lo
ss: 0.0057 - val_mean_absolute_error: 0.0607
Epoch 20/100
28/28 [==============================] - 2s 77ms/step - loss: 3.2105e-04 - mean_absolute_error: 0.0131 - val_lo
ss: 0.0049 - val_mean_absolute_error: 0.0559
Epoch 21/100
28/28 [==============================] - 2s 78ms/step - loss: 3.1451e-04 - mean_absolute_error: 0.0128 - val_lo
ss: 0.0088 - val_mean_absolute_error: 0.0798
Epoch 22/100
28/28 [==============================] - 2s 77ms/step - loss: 3.2065e-04 - mean_absolute_error: 0.0134 - val_lo
ss: 0.0043 - val_mean_absolute_error: 0.0524
Epoch 23/100
28/28 [==============================] - 2s 78ms/step - loss: 3.0431e-04 - mean_absolute_error: 0.0129 - val_lo
ss: 0.0047 - val_mean_absolute_error: 0.0551
Epoch 24/100
28/28 [==============================] - 2s 77ms/step - loss: 3.0412e-04 - mean_absolute_error: 0.0129 - val_lo
ss: 0.0063 - val_mean_absolute_error: 0.0658
Epoch 25/100
28/28 [==============================] - 2s 77ms/step - loss: 2.9042e-04 - mean_absolute_error: 0.0125 - val_lo
ss: 0.0051 - val_mean_absolute_error: 0.0583
Epoch 26/100
28/28 [==============================] - 2s 77ms/step - loss: 2.9271e-04 - mean_absolute_error: 0.0125 - val_lo
ss: 0.0029 - val_mean_absolute_error: 0.0426
Epoch 27/100
28/28 [==============================] - 2s 77ms/step - loss: 3.0456e-04 - mean_absolute_error: 0.0128 - val_lo
ss: 0.0023 - val_mean_absolute_error: 0.0367
Epoch 28/100
28/28 [==============================] - 2s 78ms/step - loss: 3.0565e-04 - mean_absolute_error: 0.0128 - val_lo
ss: 0.0048 - val_mean_absolute_error: 0.0558
Epoch 29/100
28/28 [==============================] - 2s 76ms/step - loss: 2.7566e-04 - mean_absolute_error: 0.0121 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0353
Epoch 30/100
28/28 [==============================] - 2s 76ms/step - loss: 2.6982e-04 - mean_absolute_error: 0.0120 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0358
Epoch 31/100
28/28 [==============================] - 2s 77ms/step - loss: 2.6578e-04 - mean_absolute_error: 0.0120 - val_lo
ss: 0.0036 - val_mean_absolute_error: 0.0491
Epoch 32/100                                                                                          Type
28/28 [==============================] - 2s 89ms/step - loss: 2.9437e-04 - mean_absolute_error: 0.0129 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0358
Epoch 33/100
28/28 [==============================] - 2s 64ms/step - loss: 2.8215e-04 - mean_absolute_error: 0.0123 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0361
Epoch 34/100
28/28 [==============================] - 2s 78ms/step - loss: 2.6206e-04 - mean_absolute_error: 0.0121 - val_lo
ss: 0.0029 - val_mean_absolute_error: 0.0423
Epoch 35/100
28/28 [==============================] - 2s 77ms/step - loss: 2.5252e-04 - mean_absolute_error: 0.0116 - val_lo
ss: 0.0034 - val_mean_absolute_error: 0.0473
Epoch 36/100
28/28 [==============================] - 2s 76ms/step - loss: 2.5735e-04 - mean_absolute_error: 0.0118 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0368
Epoch 37/100
28/28 [==============================] - 2s 75ms/step - loss: 2.5474e-04 - mean_absolute_error: 0.0116 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0372
Epoch 38/100
28/28 [==============================] - 2s 68ms/step - loss: 2.3630e-04 - mean_absolute_error: 0.0113 - val_lo
ss: 0.0050 - val_mean_absolute_error: 0.0586
Epoch 39/100
28/28 [==============================] - 2s 74ms/step - loss: 2.4119e-04 - mean_absolute_error: 0.0112 - val_lo
ss: 0.0059 - val_mean_absolute_error: 0.0644
Epoch 40/100
28/28 [==============================] - 2s 70ms/step - loss: 2.3362e-04 - mean_absolute_error: 0.0112 - val_lo
ss: 0.0035 - val_mean_absolute_error: 0.0474
Epoch 41/100
28/28 [==============================] - 2s 75ms/step - loss: 2.3213e-04 - mean_absolute_error: 0.0112 - val_lo
ss: 0.0027 - val_mean_absolute_error: 0.0415
Epoch 42/100
28/28 [==============================] - 2s 74ms/step - loss: 2.2727e-04 - mean_absolute_error: 0.0111 - val_lo
ss: 0.0024 - val_mean_absolute_error: 0.0396
Epoch 43/100
28/28 [==============================] - 2s 75ms/step - loss: 2.2635e-04 - mean_absolute_error: 0.0111 - val_lo
ss: 0.0043 - val_mean_absolute_error: 0.0545
Epoch 44/100
28/28 [==============================] - 2s 72ms/step - loss: 2.3254e-04 - mean_absolute_error: 0.0111 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0367
Epoch 45/100
28/28 [==============================] - 2s 73ms/step - loss: 2.4993e-04 - mean_absolute_error: 0.0118 - val_lo
ss: 0.0034 - val_mean_absolute_error: 0.0475
Epoch 46/100
28/28 [==============================] - 2s 76ms/step - loss: 2.2562e-04 - mean_absolute_error: 0.0109 - val_lo
ss: 0.0030 - val_mean_absolute_error: 0.0448
Epoch 47/100
28/28 [==============================] - 2s 69ms/step - loss: 2.1701e-04 - mean_absolute_error: 0.0108 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0365
```

```
Epoch 48/100
28/28 [==============================] - 2s 77ms/step - loss: 2.2386e-04 - mean_absolute_error: 0.0110 - val_lo
ss: 0.0050 - val_mean_absolute_error: 0.0607
Epoch 49/100
28/28 [==============================] - 2s 83ms/step - loss: 1.9933e-04 - mean_absolute_error: 0.0104 - val_lo
ss: 0.0033 - val_mean_absolute_error: 0.0472
Epoch 50/100
28/28 [==============================] - 2s 83ms/step - loss: 2.1095e-04 - mean_absolute_error: 0.0106 - val_lo
ss: 0.0038 - val_mean_absolute_error: 0.0511
Epoch 51/100
28/28 [==============================] - 2s 76ms/step - loss: 2.0873e-04 - mean_absolute_error: 0.0106 - val_lo
ss: 8.6034e-04 - val_mean_absolute_error: 0.0221
Epoch 52/100
28/28 [==============================] - 2s 74ms/step - loss: 1.9468e-04 - mean_absolute_error: 0.0103 - val_lo
ss: 0.0045 - val_mean_absolute_error: 0.0552
Epoch 53/100
28/28 [==============================] - 2s 72ms/step - loss: 1.9288e-04 - mean_absolute_error: 0.0102 - val_lo
ss: 0.0035 - val_mean_absolute_error: 0.0480
Epoch 54/100
28/28 [==============================] - 2s 76ms/step - loss: 2.1123e-04 - mean_absolute_error: 0.0108 - val_lo
ss: 0.0051 - val_mean_absolute_error: 0.0598
Epoch 55/100
28/28 [==============================] - 2s 72ms/step - loss: 2.0095e-04 - mean_absolute_error: 0.0101 - val_lo
ss: 0.0037 - val_mean_absolute_error: 0.0490
Epoch 56/100
28/28 [==============================] - 2s 76ms/step - loss: 1.8876e-04 - mean_absolute_error: 0.0101 - val_lo
ss: 0.0026 - val_mean_absolute_error: 0.0418
Epoch 57/100
28/28 [==============================] - 2s 76ms/step - loss: 1.8531e-04 - mean_absolute_error: 0.0099 - val_lo
ss: 0.0014 - val_mean_absolute_error: 0.0285
Epoch 58/100
28/28 [==============================] - 2s 75ms/step - loss: 1.7605e-04 - mean_absolute_error: 0.0096 - val_lo
ss: 0.0029 - val_mean_absolute_error: 0.0436
Epoch 59/100
28/28 [==============================] - 2s 76ms/step - loss: 2.0312e-04 - mean_absolute_error: 0.0103 - val_lo
ss: 0.0026 - val_mean_absolute_error: 0.0403
Epoch 60/100
28/28 [==============================] - 2s 75ms/step - loss: 2.0422e-04 - mean_absolute_error: 0.0104 - val_lo
ss: 9.0172e-04 - val_mean_absolute_error: 0.0228
Epoch 61/100
28/28 [==============================] - 2s 70ms/step - loss: 1.8638e-04 - mean_absolute_error: 0.0100 - val_lo
ss: 9.2860e-04 - val_mean_absolute_error: 0.0230
Epoch 62/100
28/28 [==============================] - 2s 70ms/step - loss: 1.8993e-04 - mean_absolute_error: 0.0101 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0363
Epoch 63/100
28/28 [==============================] - 2s 71ms/step - loss: 1.7831e-04 - mean_absolute_error: 0.0097 - val_lo
ss: 0.0040 - val_mean_absolute_error: 0.0520
Epoch 64/100
28/28 [==============================] - 2s 66ms/step - loss: 1.8095e-04 - mean_absolute_error: 0.0098 - val_lo
ss: 0.0050 - val_mean_absolute_error: 0.0579
Epoch 65/100
28/28 [==============================] - 2s 71ms/step - loss: 1.7173e-04 - mean_absolute_error: 0.0095 - val_lo
ss: 0.0025 - val_mean_absolute_error: 0.0395
Epoch 66/100
28/28 [==============================] - 2s 68ms/step - loss: 1.6561e-04 - mean_absolute_error: 0.0092 - val_lo
ss: 0.0030 - val_mean_absolute_error: 0.0441
Epoch 67/100
28/28 [==============================] - 2s 74ms/step - loss: 1.7168e-04 - mean_absolute_error: 0.0096 - val_lo
ss: 0.0039 - val_mean_absolute_error: 0.0525
Epoch 68/100
28/28 [==============================] - 2s 66ms/step - loss: 1.7348e-04 - mean_absolute_error: 0.0097 - val_lo
ss: 0.0031 - val_mean_absolute_error: 0.0450
Epoch 69/100
28/28 [==============================] - 2s 77ms/step - loss: 1.6919e-04 - mean_absolute_error: 0.0096 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0373
Epoch 70/100
28/28 [==============================] - 2s 77ms/step - loss: 1.5923e-04 - mean_absolute_error: 0.0092 - val_lo
ss: 0.0013 - val_mean_absolute_error: 0.0267
Epoch 71/100
28/28 [==============================] - 2s 69ms/step - loss: 1.5958e-04 - mean_absolute_error: 0.0091 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0365
Epoch 72/100
28/28 [==============================] - 2s 73ms/step - loss: 1.5513e-04 - mean_absolute_error: 0.0090 - val_lo
ss: 0.0015 - val_mean_absolute_error: 0.0304
Epoch 73/100
28/28 [==============================] - 2s 67ms/step - loss: 1.5671e-04 - mean_absolute_error: 0.0090 - val_lo
ss: 0.0027 - val_mean_absolute_error: 0.0416
Epoch 74/100
28/28 [==============================] - 2s 74ms/step - loss: 1.6093e-04 - mean_absolute_error: 0.0091 - val_lo
ss: 0.0030 - val_mean_absolute_error: 0.0454
Epoch 75/100
28/28 [==============================] - 2s 76ms/step - loss: 1.4674e-04 - mean_absolute_error: 0.0088 - val_lo
ss: 0.0030 - val_mean_absolute_error: 0.0453
Epoch 76/100
28/28 [==============================] - 2s 76ms/step - loss: 1.4612e-04 - mean_absolute_error: 0.0087 - val_lo
ss: 0.0034 - val_mean_absolute_error: 0.0485
Epoch 77/100
28/28 [==============================] - 2s 77ms/step - loss: 1.4620e-04 - mean_absolute_error: 0.0087 - val_lo
```

```
ss: 0.0020 - val_mean_absolute_error: 0.0356
Epoch 78/100
28/28 [==============================] - 2s 75ms/step - loss: 1.6000e-04 - mean_absolute_error: 0.0092 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0358
Epoch 79/100
28/28 [==============================] - 2s 75ms/step - loss: 1.5322e-04 - mean_absolute_error: 0.0090 - val_lo
ss: 0.0039 - val_mean_absolute_error: 0.0522
Epoch 80/100
28/28 [==============================] - 2s 76ms/step - loss: 1.5051e-04 - mean_absolute_error: 0.0091 - val_lo
ss: 0.0036 - val_mean_absolute_error: 0.0513
Epoch 81/100
28/28 [==============================] - 2s 76ms/step - loss: 1.6520e-04 - mean_absolute_error: 0.0095 - val_lo
ss: 0.0022 - val_mean_absolute_error: 0.0365
Epoch 82/100
28/28 [==============================] - 2s 75ms/step - loss: 1.4147e-04 - mean_absolute_error: 0.0085 - val_lo
ss: 0.0019 - val_mean_absolute_error: 0.0343
Epoch 83/100
28/28 [==============================] - 2s 73ms/step - loss: 1.3314e-04 - mean_absolute_error: 0.0083 - val_lo
ss: 0.0041 - val_mean_absolute_error: 0.0543
Epoch 84/100
28/28 [==============================] - 2s 71ms/step - loss: 1.3426e-04 - mean_absolute_error: 0.0084 - val_lo
ss: 0.0028 - val_mean_absolute_error: 0.0432
Epoch 85/100
28/28 [==============================] - 2s 75ms/step - loss: 1.2755e-04 - mean_absolute_error: 0.0080 - val_lo
ss: 0.0031 - val_mean_absolute_error: 0.0454
Epoch 86/100
28/28 [==============================] - 2s 76ms/step - loss: 1.3597e-04 - mean_absolute_error: 0.0084 - val_lo
ss: 0.0033 - val_mean_absolute_error: 0.0465
Epoch 87/100
28/28 [==============================] - 2s 68ms/step - loss: 1.3157e-04 - mean_absolute_error: 0.0082 - val_lo
ss: 0.0027 - val_mean_absolute_error: 0.0432
Epoch 88/100
28/28 [==============================] - 2s 72ms/step - loss: 1.3880e-04 - mean_absolute_error: 0.0085 - val_lo
ss: 0.0019 - val_mean_absolute_error: 0.0354
Epoch 89/100
28/28 [==============================] - 2s 72ms/step - loss: 1.3528e-04 - mean_absolute_error: 0.0085 - val_lo
ss: 0.0021 - val_mean_absolute_error: 0.0366
Epoch 90/100
28/28 [==============================] - 2s 72ms/step - loss: 1.5266e-04 - mean_absolute_error: 0.0090 - val_lo
ss: 0.0012 - val_mean_absolute_error: 0.0265
Epoch 91/100
28/28 [==============================] - 2s 76ms/step - loss: 1.3766e-04 - mean_absolute_error: 0.0085 - val_lo
ss: 0.0015 - val_mean_absolute_error: 0.0286
Epoch 92/100
28/28 [==============================] - 2s 76ms/step - loss: 1.3259e-04 - mean_absolute_error: 0.0084 - val_lo
ss: 0.0025 - val_mean_absolute_error: 0.0402
Epoch 93/100
28/28 [==============================] - 2s 76ms/step - loss: 1.2394e-04 - mean_absolute_error: 0.0082 - val_lo
ss: 0.0019 - val_mean_absolute_error: 0.0354
Epoch 94/100
28/28 [==============================] - 2s 75ms/step - loss: 1.2022e-04 - mean_absolute_error: 0.0078 - val_lo
ss: 0.0013 - val_mean_absolute_error: 0.0282
Epoch 95/100
28/28 [==============================] - 2s 72ms/step - loss: 1.3169e-04 - mean_absolute_error: 0.0083 - val_lo
ss: 0.0010 - val_mean_absolute_error: 0.0254
Epoch 96/100
28/28 [==============================] - 2s 74ms/step - loss: 1.4499e-04 - mean_absolute_error: 0.0087 - val_lo
ss: 0.0027 - val_mean_absolute_error: 0.0430
Epoch 97/100
28/28 [==============================] - 2s 72ms/step - loss: 1.1930e-04 - mean_absolute_error: 0.0080 - val_lo
ss: 0.0039 - val_mean_absolute_error: 0.0529
Epoch 98/100
28/28 [==============================] - 2s 71ms/step - loss: 1.1981e-04 - mean_absolute_error: 0.0079 - val_lo
ss: 0.0023 - val_mean_absolute_error: 0.0386
Epoch 99/100
28/28 [==============================] - 2s 72ms/step - loss: 1.2681e-04 - mean_absolute_error: 0.0081 - val_lo
ss: 0.0028 - val_mean_absolute_error: 0.0441
Epoch 100/100
28/28 [==============================] - 2s 71ms/step - loss: 1.2679e-04 - mean_absolute_error: 0.0081 - val_lo
ss: 0.0028 - val_mean_absolute_error: 0.0424
```

Out[16]:
```
<keras.callbacks.History at 0x23b61997760>
```

In [17]:
```
# predicting the values after running the model
test_predicted = model.predict(test_seq)
test_predicted[:5]
```

```
9/9 [==============================] - 2s 19ms/step
```

Out[17]:
```
array([[0.3987828 , 0.40509373],
       [0.39797232, 0.40461183],
       [0.39309195, 0.3999965 ],
       [0.3959207 , 0.4024696 ],
       [0.39985454, 0.4062231 ]], dtype=float32)
```

In [18]:
```
# Inversing normalization/scaling on predicted data
test_inverse_predicted = MMS.inverse_transform(test_predicted)
test_inverse_predicted[:5]
```

Out[18]:
```
array([[1410.3114, 1419.0365],
       [1408.8088, 1418.1433],
       [1399.761 , 1409.5895],
       [1405.0052, 1414.173 ],
       [1412.2983, 1421.1295]], dtype=float32)
```

VISUALIZING ACTUAL VS PREDICTED DATA

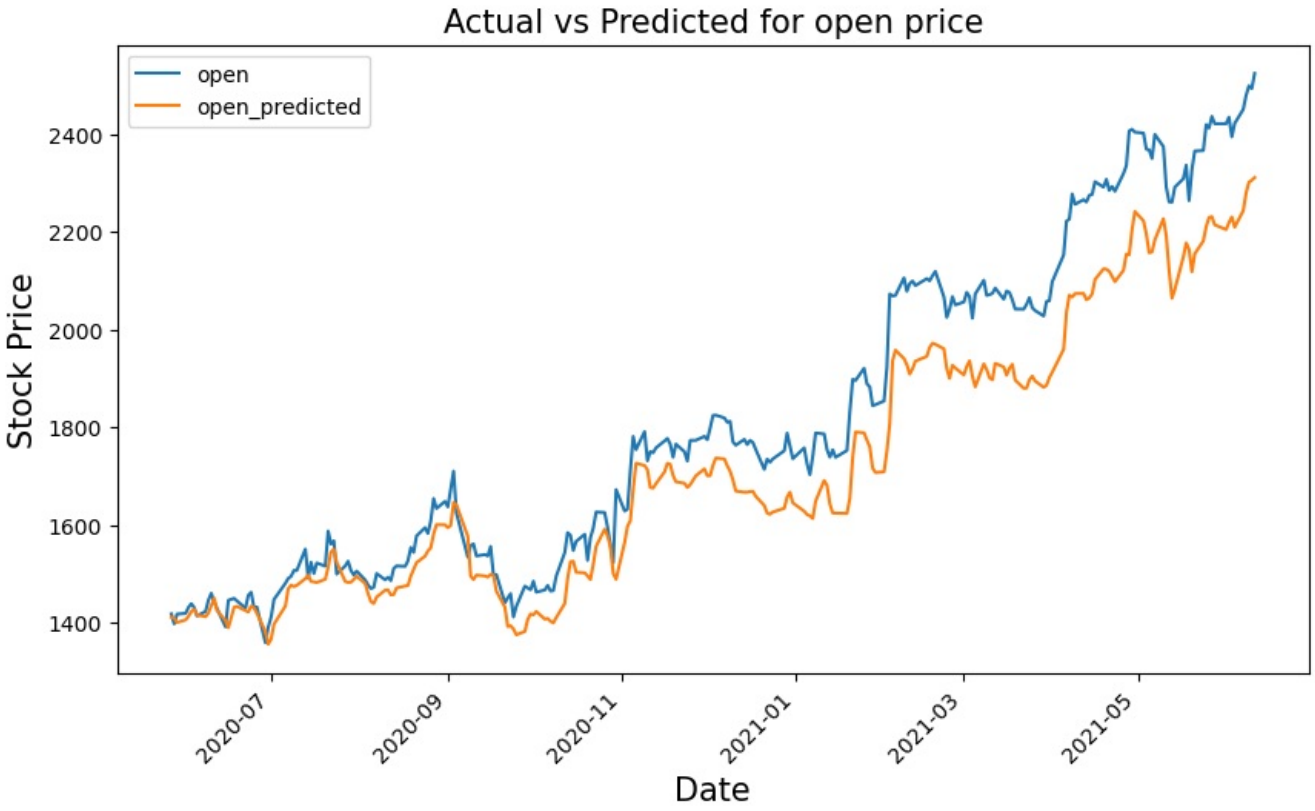
In [19]:
```
# Merging actual and predicted data for better visualization
df_merge = pd.concat([df.iloc[-264:].copy(),
                      pd.DataFrame(test_inverse_predicted,columns=['open_predicted','close_predicted'],
                                   index=df.iloc[-264:].index)], axis=1)
```

In [20]:
```
# Inversing normalization/scaling
df_merge[['open','close']] = MMS.inverse_transform(df_merge[['open','close']])
df_merge.head()
```

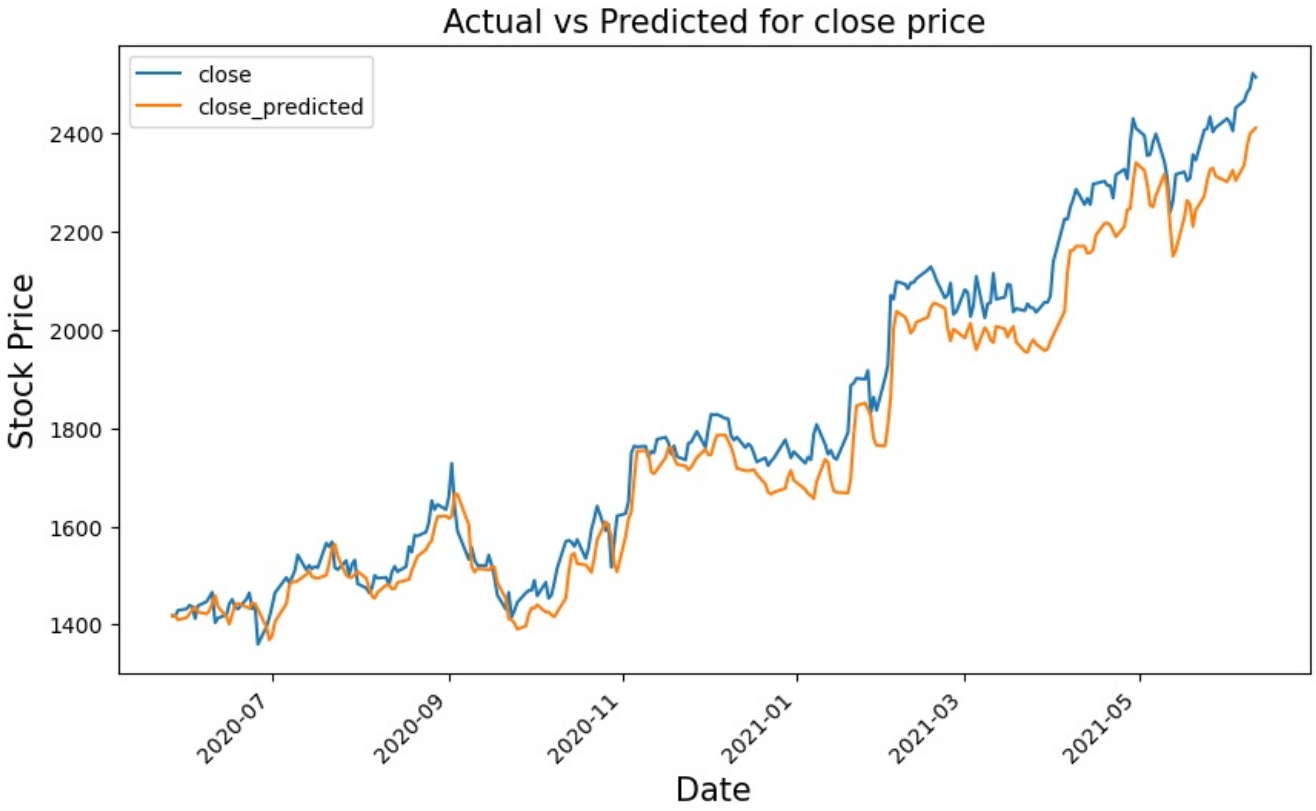
Out[20]:

date	open	close	open_predicted	close_predicted
2020-05-27	1417.25	1417.84	1410.311401	1419.036499
2020-05-28	1396.86	1416.73	1408.808838	1418.143311
2020-05-29	1416.94	1428.92	1399.760986	1409.589478
2020-06-01	1418.39	1431.82	1405.005249	1414.172974
2020-06-02	1430.55	1439.22	1412.298340	1421.129517

In [21]:
```
# plotting the actual open and predicted open prices on date index
df_merge[['open','open_predicted']].plot(figsize=(10,6))
plt.xticks(rotation=45)
plt.xlabel('Date',size=15)
plt.ylabel('Stock Price',size=15)
plt.title('Actual vs Predicted for open price',size=15)
plt.show()
```



In [22]:
```
# plotting the actual close and predicted close prices on date index
df_merge[['close','close_predicted']].plot(figsize=(10,6))
plt.xticks(rotation=45)
plt.xlabel('Date',size=15)
plt.ylabel('Stock Price',size=15)
plt.title('Actual vs Predicted for close price',size=15)
plt.show()
```



PREDICTING UPCOMING 10 DAYS

In [23]:
```
# Creating a dataframe and adding 10 days to existing index

df_merge = df_merge.append(pd.DataFrame(columns=df_merge.columns,
                                        index=pd.date_range(start=df_merge.index[-1], periods=11, freq='D', clo
df_merge['2021-06-09':'2021-06-16']
```

Out[23]:

	open	close	open_predicted	close_predicted
2021-06-09	2499.50	2491.40	2301.978027	2398.498047
2021-06-10	2494.01	2521.60	2306.009033	2404.821045
2021-06-11	2524.92	2513.93	2311.519531	2410.579346
2021-06-12	NaN	NaN	NaN	NaN
2021-06-13	NaN	NaN	NaN	NaN
2021-06-14	NaN	NaN	NaN	NaN
2021-06-15	NaN	NaN	NaN	NaN
2021-06-16	NaN	NaN	NaN	NaN

In [24]:
```
# creating a DataFrame and filling values of open and close column
upcoming_prediction = pd.DataFrame(columns=['open','close'],index=df_merge.index)
upcoming_prediction.index=pd.to_datetime(upcoming_prediction.index)
```

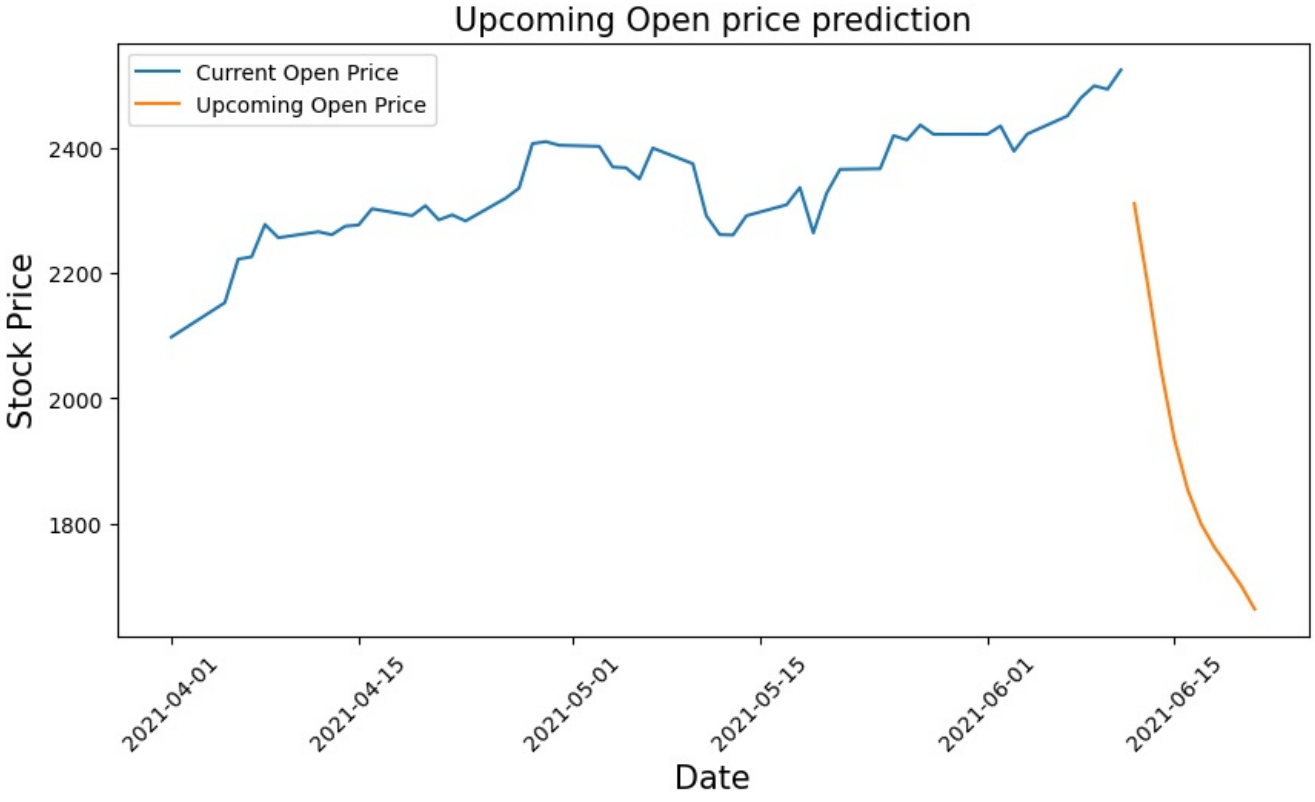
In [25]:
```
curr_seq = test_seq[-1:]

for i in range(-10,0):
  up_pred = model.predict(curr_seq)
  upcoming_prediction.iloc[i] = up_pred
  curr_seq = np.append(curr_seq[0][1:],up_pred,axis=0)
  curr_seq = curr_seq.reshape(test_seq[-1:].shape)
```

```
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 50ms/step
```

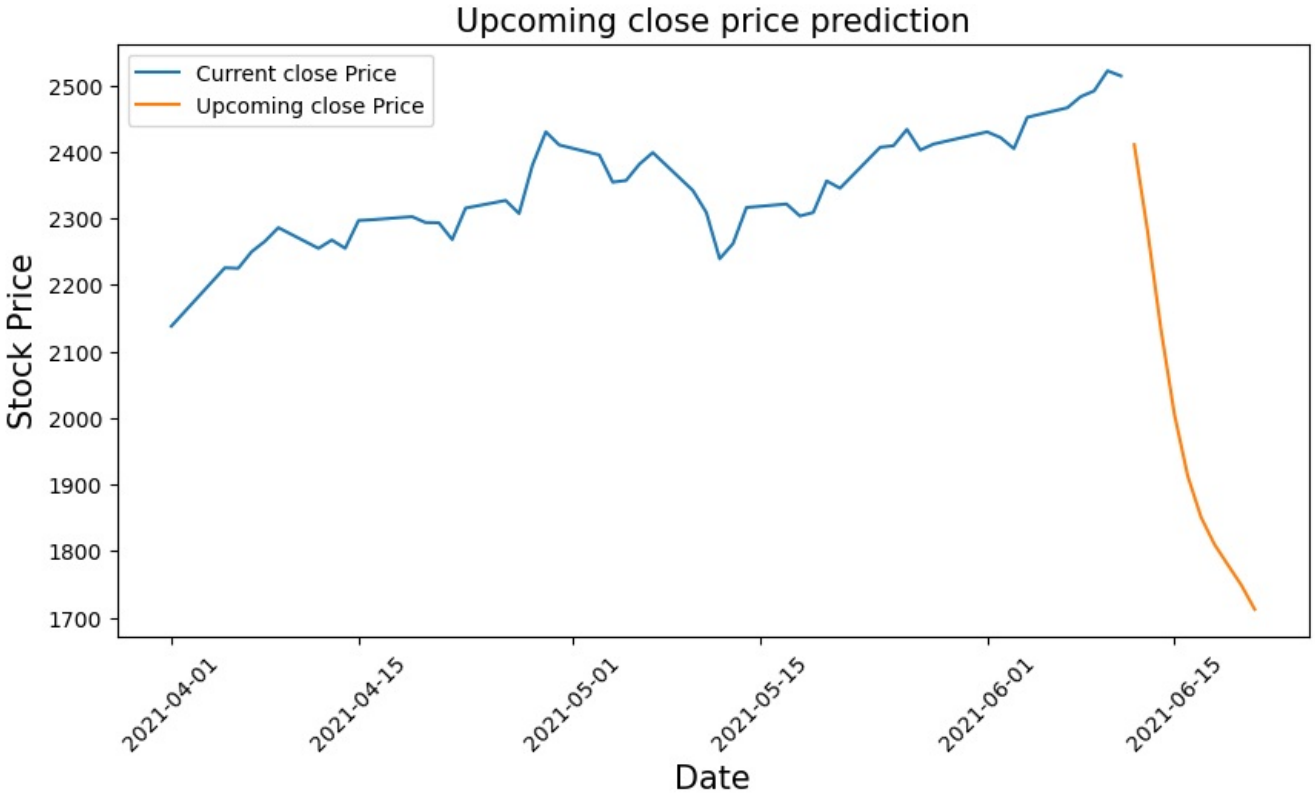
In [26]:
```
# inversing Normalization/scaling
upcoming_prediction[['open','close']] = MMS.inverse_transform(upcoming_prediction[['open','close']])
```

In [27]:
```
# plotting Upcoming Open price on date index
```

```
fig,ax=plt.subplots(figsize=(10,5))
ax.plot(df_merge.loc['2021-04-01':,'open'],label='Current Open Price')
ax.plot(upcoming_prediction.loc['2021-04-01':,'open'],label='Upcoming Open Price')
plt.setp(ax.xaxis.get_majorticklabels(), rotation=45)
ax.set_xlabel('Date',size=15)
ax.set_ylabel('Stock Price',size=15)
ax.set_title('Upcoming Open price prediction',size=15)
ax.legend()
fig.show()
```



In [28]:
```
# plotting Upcoming Close price on date index
fig,ax=plt.subplots(figsize=(10,5))
ax.plot(df_merge.loc['2021-04-01':,'close'],label='Current close Price')
ax.plot(upcoming_prediction.loc['2021-04-01':,'close'],label='Upcoming close Price')
plt.setp(ax.xaxis.get_majorticklabels(), rotation=45)
ax.set_xlabel('Date',size=15)
ax.set_ylabel('Stock Price',size=15)
ax.set_title('Upcoming close price prediction',size=15)
ax.legend()
fig.show()
```



THANK YOU!

In [ ]: