TCS 1 YEAR DATA

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
tcs1=pd.read_csv('C:/Users/Uma Shankar/Downloads/TCS1.csv',index_col='Date')
tcs1.head()
```

Out[1]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-07-16	2244.0	2333.000000	2220.100098	2234.750000	2205.411621	8582118.0
2020-07-17	2237.0	2243.899902	2190.050049	2200.750000	2171.857910	4509135.0
2020-07-20	2201.0	2226.899902	2190.800049	2207.899902	2178.913818	2952646.0
2020-07-21	2230.0	2238.649902	2201.149902	2225.050049	2195.838867	2665286.0
2020-07-22	2231.0	2231.000000	2184.199951	2190.949951	2162.186523	2861534.0

In [2]:
```
tcs1.tail()
```

Out[2]:

Date	Open	High	Low	Close	Adj Close	Volume
2021-07-12	3235.000000	3236.000000	3188.350098	3193.100098	3186.146729	1892226.0
2021-07-13	3214.000000	3214.000000	3175.000000	3187.550049	3180.608887	1809339.0
2021-07-14	3187.000000	3222.000000	3185.000000	3214.550049	3207.550049	2044279.0
2021-07-15	3227.600098	3231.850098	3194.000000	3202.949951	3202.949951	2232968.0
2021-07-16	3213.000000	3219.850098	3192.000000	3197.850098	3197.850098	748832.0

In [3]:
```
tcs1.shape
```

Out[3]:
```
(252, 6)
```

In [4]:
```
tcs1.describe()
```

Out[4]:

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	2871.800399	2901.482070	2840.124912	2868.808174	2846.191865	3.565610e+06
std	358.747772	358.572987	355.051597	357.011762	362.436681	2.184349e+06
min	2154.500000	2163.000000	2125.100098	2157.399902	2129.076904	7.488320e+05
25%	2652.500000	2675.175049	2615.800049	2643.250000	2619.670288	2.271646e+06
50%	3006.949951	3050.750000	2968.449951	3006.949951	2985.796875	2.952646e+06
75%	3170.000000	3207.425049	3130.275024	3159.049926	3144.434448	4.150314e+06
max	3391.500000	3399.649902	3350.000000	3380.800049	3373.437988	1.983933e+07

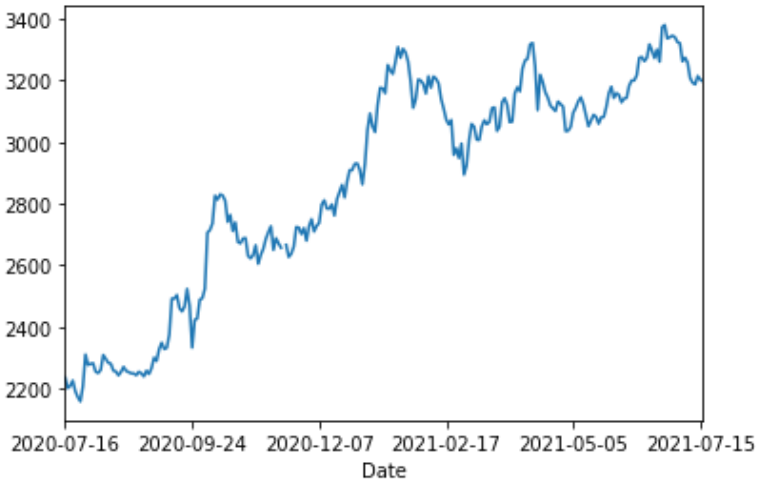
In [5]:
```
tcs1.loc['2021-06-16':'2021-07-16'].head()
```

Out[5]:

Date	Open	High	Low	Close	Adj Close	Volume
2021-06-16	3262.100098	3294.699951	3253.000000	3274.350098	3267.219971	1635552.0
2021-06-17	3265.500000	3336.050049	3260.000000	3317.750000	3310.525391	2273413.0
2021-06-18	3350.899902	3358.000000	3275.000000	3297.300049	3290.119873	3380431.0
2021-06-21	3265.000000	3286.000000	3251.699951	3273.100098	3265.972656	1130569.0
2021-06-22	3304.000000	3327.050049	3285.000000	3301.199951	3294.011230	1708688.0

In [6]:
```
tcs1['Close'].plot()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1273f39c148>
```



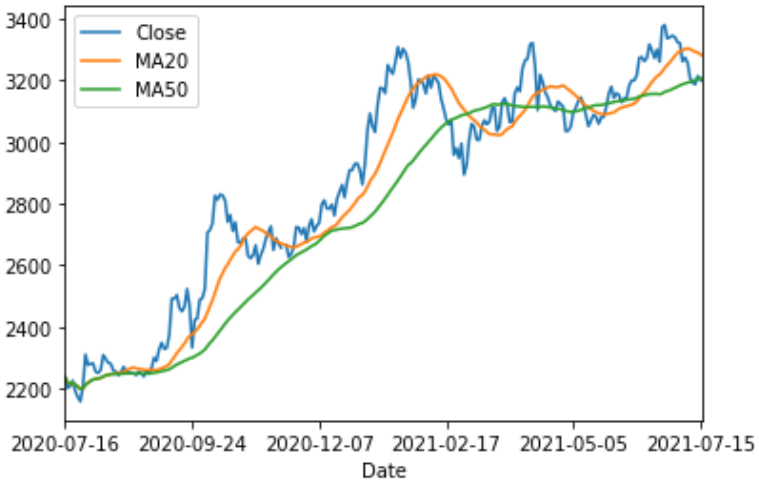
In [7]:
```
tcs1['Close-shift'] = tcs1['Close'].shift(-1)
tcs1['pricediff'] = tcs1['Close-shift'] - tcs1['Close']
tcs1['Return'] = tcs1['pricediff']/tcs1['Close']
tcs1['MA20'] =tcs1['Close'].rolling(20,min_periods=1).mean() #MA - MOVING AVERAGE N DAYS
tcs1['MA50'] =tcs1['Close'].rolling(50,min_periods=1).mean()
tcs1.head()
```

Out[7]:

Date	Open	High	Low	Close	Adj Close	Volume	Close-shift	pricediff	Return	
2020-07-16	2244.0	2333.000000	2220.100098	2234.750000	2205.411621	8582118.0	2200.750000	-34.000000	-0.015214	22
2020-07-17	2237.0	2243.899902	2190.050049	2200.750000	2171.857910	4509135.0	2207.899902	7.149902	0.003249	22
2020-07-20	2201.0	2226.899902	2190.800049	2207.899902	2178.913818	2952646.0	2225.050049	17.150147	0.007768	22
2020-07-21	2230.0	2238.649902	2201.149902	2225.050049	2195.838867	2665286.0	2190.949951	-34.100098	-0.015326	22
2020-07-22	2231.0	2231.000000	2184.199951	2190.949951	2162.186523	2861534.0	2171.199951	-19.750000	-0.009014	22

In [8]:
```
tcs1['Close'].plot()
tcs1['MA20'].plot()
tcs1['MA50'].plot()
plt.legend()
```

Out[8]:
```
<matplotlib.legend.Legend at 0x1273f7161c8>
```



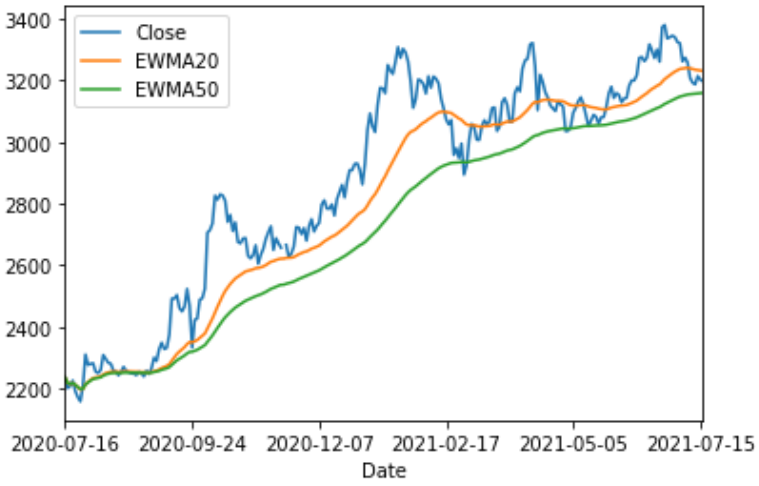
In [9]:
```
tcs1['EWMA20'] =tcs1['Close'].ewm(20,min_periods=1).mean() # exponential weighted moving average
tcs1['EWMA50'] =tcs1['Close'].ewm(50,min_periods=1).mean()
tcs1['EWMA200'] =tcs1['Close'].ewm(200,min_periods=1).mean()
tcs1.tail()
```

Out[9]:

Date	Open	High	Low	Close	Adj Close	Volume	Close-shift	pricediff	Returı
2021-07-12	3235.000000	3236.000000	3188.350098	3193.100098	3186.146729	1892226.0	3187.550049	-5.550049	-0.00173
2021-07-13	3214.000000	3214.000000	3175.000000	3187.550049	3180.608887	1809339.0	3214.550049	27.000000	0.00847
2021-07-14	3187.000000	3222.000000	3185.000000	3214.550049	3207.550049	2044279.0	3202.949951	-11.600098	-0.00360
2021-07-15	3227.600098	3231.850098	3194.000000	3202.949951	3202.949951	2232968.0	3197.850098	-5.099853	-0.00159
2021-07-16	3213.000000	3219.850098	3192.000000	3197.850098	3197.850098	748832.0	NaN	NaN	Nal

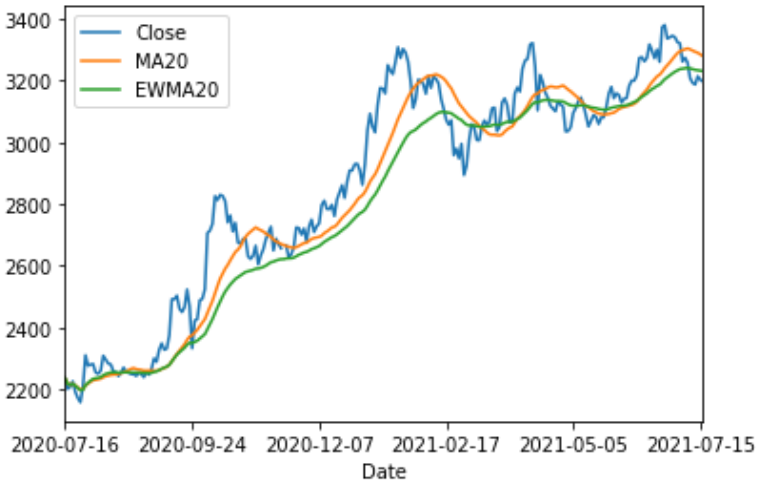
In [10]:
```
tcs1['Close'].plot()
tcs1['EWMA20'].plot()
tcs1['EWMA50'].plot()
plt.legend()
```

Out[10]: <matplotlib.legend.Legend at 0x1273f788d48>



In [11]:
```
tcs1['Close'].plot()
tcs1['MA20'].plot()
tcs1['EWMA20'].plot()
plt.legend()
```

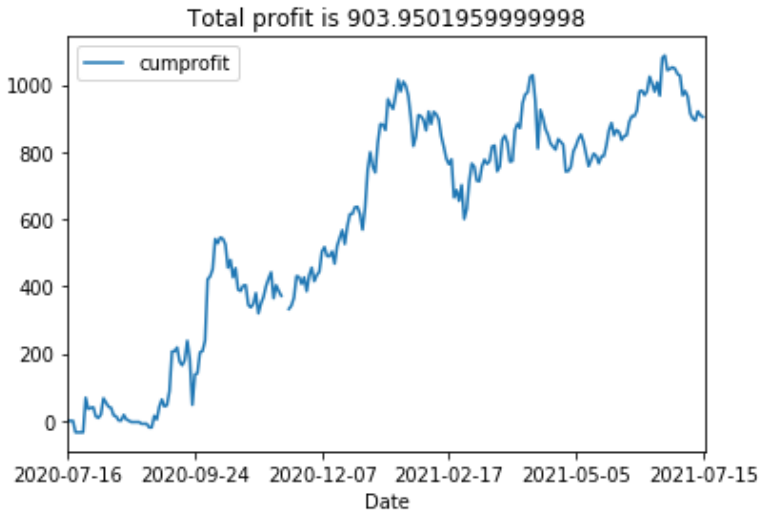
Out[11]: <matplotlib.legend.Legend at 0x1273f82ab08>



In [12]:
```
# Close price > EWMA (20)
tcs1['share'] = [1 if tcs1.loc[i,'Close'] > tcs1.loc[i,'EWMA200'] else 0 for i in tcs1.index]
tcs1['profit'] = tcs1['pricediff']*tcs1['share']
tcs1['cumprofit'] = tcs1['profit'].cumsum()
tcs1['cumprofit'].plot()
plt.title('Total profit is {}'.format(tcs1.loc[tcs1.index[-2],'cumprofit']))
plt.legend()
tcs1.head()
```

Out[12]:

Date	Open	High	Low	Close	Adj Close	Volume	Close-shift	pricediff	Return	
2020-07-16	2244.0	2333.000000	2220.100098	2234.750000	2205.411621	8582118.0	2200.750000	-34.000000	-0.015214	22
2020-07-17	2237.0	2243.899902	2190.050049	2200.750000	2171.857910	4509135.0	2207.899902	7.149902	0.003249	22
2020-07-20	2201.0	2226.899902	2190.800049	2207.899902	2178.913818	2952646.0	2225.050049	17.150147	0.007768	22
2020-07-21	2230.0	2238.649902	2201.149902	2225.050049	2195.838867	2665286.0	2190.949951	-34.100098	-0.015326	22
2020-07-22	2231.0	2231.000000	2184.199951	2190.949951	2162.186523	2861534.0	2171.199951	-19.750000	-0.009014	22



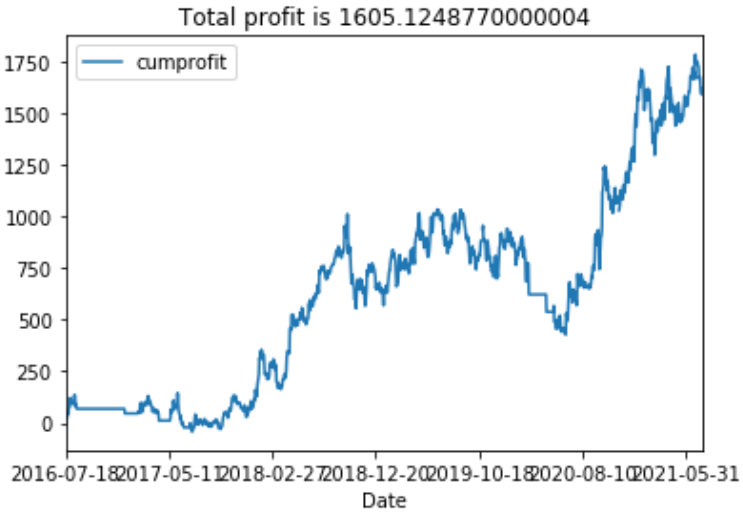
TCS 5 YEAR DATA

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
tcs5=pd.read_csv('C:/Users/Uma Shankar/Downloads/TCS5.csv',index_col='Date')
```

In [14]:
```
tcs5['Close-shift'] = tcs5['Close'].shift(-1)
tcs5['pricediff'] = tcs5['Close-shift'] - tcs5['Close']
tcs5['Return'] = tcs5['pricediff']/tcs5['Close']
tcs5['MA20'] =tcs5['Close'].rolling(20,min_periods=1).mean()
tcs5['MA50'] =tcs5['Close'].rolling(50,min_periods=1).mean()
tcs5['EWMA20'] =tcs5['Close'].ewm(20,min_periods=1).mean()
tcs5['EWMA50'] =tcs5['Close'].ewm(50,min_periods=1).mean()
tcs5['EWMA200'] =tcs5['Close'].ewm(200,min_periods=1).mean()
tcs5.head()
# Close price > EWMA (20)
tcs5['share'] = [1 if tcs5.loc[i,'Close'] > tcs5.loc[i,'EWMA200'] else 0 for i in tcs5.index]
tcs5['profit'] = tcs5['pricediff']*tcs5['share']
tcs5['cumprofit'] = tcs5['profit'].cumsum()
tcs5.tail()
tcs5['cumprofit'].plot()
plt.legend()
plt.title('Total profit is {}'.format(tcs5.loc[tcs5.index[-2],'cumprofit']))
tcs5.head()
```

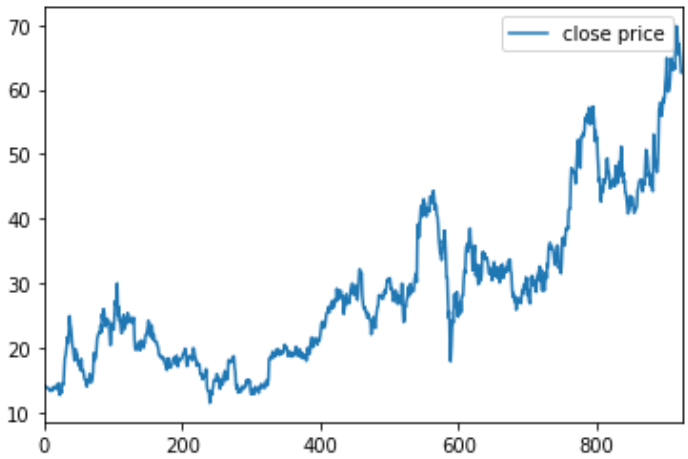
Out[14]:

Date	Open	High	Low	Close	Adj Close	Volume	Close-shift	pricediff	Return
2016-07-18	1223.500000	1234.500000	1215.300049	1216.724976	1101.211426	2598850.0	1233.099976	16.375000	0.013458
2016-07-19	1213.750000	1236.400024	1213.750000	1233.099976	1116.031982	2362896.0	1247.474976	14.375000	0.011658
2016-07-20	1225.000000	1251.250000	1224.074951	1247.474976	1129.042114	2815252.0	1253.025024	5.550048	0.004449
2016-07-21	1244.750000	1259.750000	1232.224976	1253.025024	1134.065430	2527148.0	1257.550049	4.525025	0.003611
2016-07-22	1249.449951	1262.000000	1246.000000	1257.550049	1138.160645	1517216.0	1279.474976	21.924927	0.017435



In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('C:/Users/Uma Shankar/Downloads/APLS.csv')
df['Close'].plot(label='close price')
plt.legend()
df.shape
```

Out[15]: (925, 7)



In [16]:
```
#Creating a new dataframe with index and the target variable(Close Price)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Close'])

for i in range(0,len(df)):
     new_data['Close'][i] = df['Close'][i]

# splitting into train and validation
# First 80% of total days for train data & remaining 20% for test (valid) data
m = int(0.8*df.shape[0])

train = new_data[:m]
test = new_data[m:]

# shapes of training set
print('\n Shape of training set:')
print(train.shape)

# shapes of test set
print('\n Shape of test set:')
print(test.shape)
```

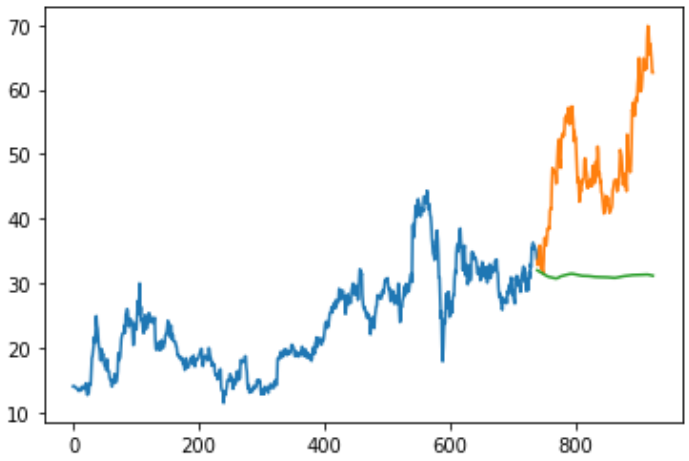
```
 Shape of training set:
(740, 1)

 Shape of test set:
(185, 1)
```

Method 1: Moving Average

In [17]:
```
# In the next step, we will create predictions for the validation set and check the
# RMSE using the actual values.
# making predictions
preds = []
for i in range(0,test.shape[0]):
    a = train['Close'][len(train)-test.shape[0]+i:].sum() + sum(preds)
    b = a/test.shape[0]
    preds.append(b)

# checking the results (RMSE value)
rms=np.sqrt(np.mean(np.power((np.array(test['Close'])-preds),2)))
print('\n RMSE value on validation set:')
print(rms)

# Ploting the prediction to visualize the results
test['Predictions'] = 0
test['Predictions'] = preds

plt.plot(train['Close'])
plt.plot(test[['Close', 'Predictions']])
```

```
 RMSE value on validation set:
19.64088798947

C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:16: SettingWithCopyWarni
ng:
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/user_guide/in
dexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  app.launch_new_instance()
C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: SettingWithCopyWarni
ng:
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/user_guide/in
dexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x127409efe08>,
 <matplotlib.lines.Line2D at 0x127409fa388>]
```



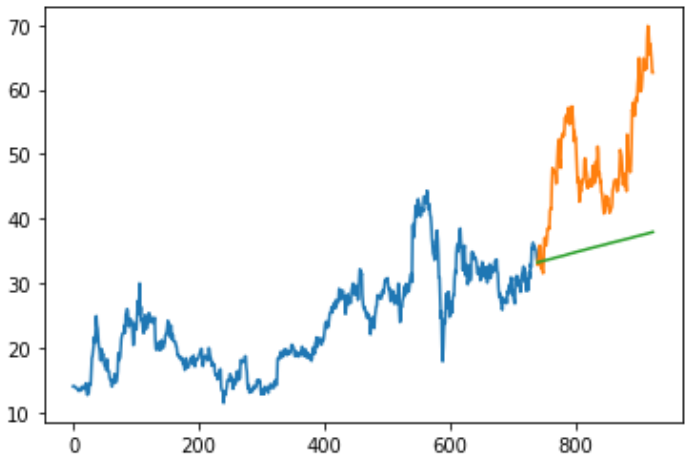
Method 2: Linear Regression

In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('C:/Users/Uma Shankar/Downloads/APLS.csv')
df.head()
#creating a separate dataset
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(df)):
    new_data['Date'][i] = df['Date'][i]
    new_data['Close'][i] = df['Close'][i]

new_data.head(10)
# First 80% of total days for train data & remaining 20% for test (valid) data
m = int(0.8*df.shape[0])

#split into train and validation
train = new_data[:m]
valid = new_data[m:]

x_train = train.drop('Close', axis=1)
x_train['Date'] = pd.to_numeric(pd.to_datetime(x_train['Date']))
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
x_valid['Date'] = pd.to_numeric(pd.to_datetime(x_valid['Date']))
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)

#make predictions and find the rmse
preds = model.predict(x_valid)

# checking the results (RMSE value)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
print('\n RMSE value on validation set:')
print(rms)


#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[m:].index
train.index = new_data[:m].index

plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

```
 RMSE value on validation set:
15.351438484899019

C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:43: SettingWithCopyWa
rning:
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/user_guid
e/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy)
```

```
C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:44: SettingWithCopyWa
rning:
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/user_guid
e/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x12741f71208>,
 <matplotlib.lines.Line2D at 0x12741f71748>]
```



Method 3: kNN Method

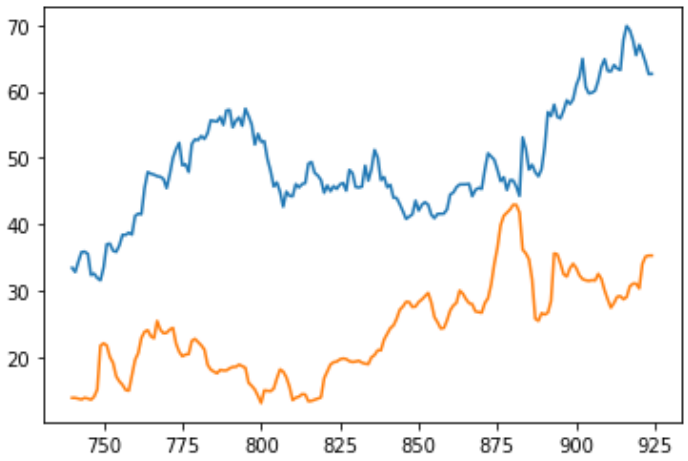
In [19]:
```
#importing libraries
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

In [20]:
```
#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)

#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

In [21]:
```
# checking the results (RMSE value)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
print('\n RMSE value on validation set:')
print(rms)

#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(valid[['Close', 'Predictions']])
#plt.plot(train['Close'])
```

```
 RMSE value on validation set:
26.436921304451886

C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarnin
g:
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/user_guide/in
dexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  import sys
C:\Users\Uma Shankar\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarnin
g:
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/user_guide/in
dexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x1274204df88>,
 <matplotlib.lines.Line2D at 0x127420b8f08>]
```



LSTM

In [2]:

```
#importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

import pandas as pd
import numpy as np
df = pd.read_csv('C:/Users/Uma Shankar/Downloads/APLS.csv')
df.head()

#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

#setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)
```

In [3]:

```
new_data.shape
```

Out[3]: (925, 1)

In [4]:
```
#creating train and test sets
dataset = new_data.values

# 80% data for train and 20% for valid
# First 80% of total days for train data & remaining 20% for test (valid) data
m = int(0.8*df.shape[0])

#train = dataset[0:m,:]
#valid = dataset[m:,:]
train = dataset[:m]
valid = dataset[m:]

#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

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, 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))

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=60, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=60))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

#predicting 554 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

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 = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

```
680/680 - 30s - loss: 0.0028
```

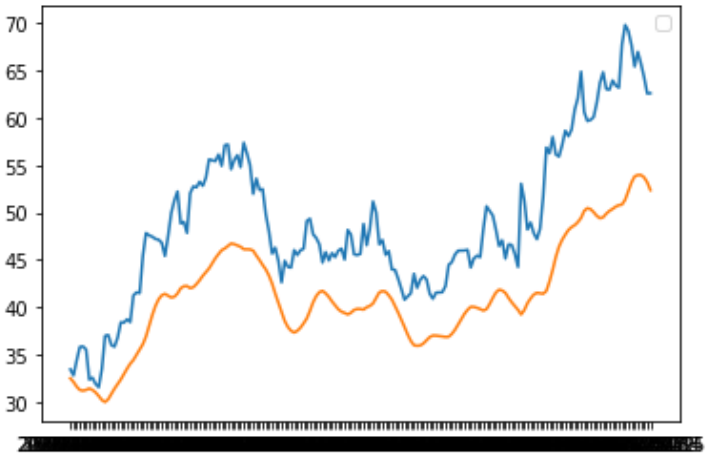
In [5]:
```
rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
rms
```

Out[5]:
```
8.431653438103904
```

In [7]:
```
#for plotting
import matplotlib.pyplot as plt
train = new_data[:m]
valid = new_data[m:]
valid['Predictions'] = closing_price
#plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend()
```

```
<ipython-input-7-c709f1d88ac1>:5: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  valid['Predictions'] = closing_price
No handles with labels found to put in legend.
```

Out[7]:
```
<matplotlib.legend.Legend at 0x21e4ea1e5b0>
```



In [ ]: