In [1]:

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

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:

```
data = pd.read_csv("D:/Dixant/CDAC/Machine Learning/LAB GRADED/HCLTECH.csv")
```

In [4]:

```
data
```

Out[4]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	
0	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	
1	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	
2	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	
3	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	
4	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	
...	...	...	...	...	...	...	...	...	...	...	
5193	2020-11-23	HCLTECH	EQ	819.25	825.0	842.00	816.25	838.50	839.20	832.35	
5194	2020-11-24	HCLTECH	EQ	839.20	843.9	857.40	835.35	841.00	840.50	847.95	
5195	2020-11-25	HCLTECH	EQ	840.50	840.5	846.00	822.50	825.00	824.70	829.08	
5196	2020-11-26	HCLTECH	EQ	824.70	824.1	845.00	819.60	841.20	842.05	834.43	
5197	2020-11-27	HCLTECH	EQ	842.05	842.0	847.80	814.35	823.15	822.10	827.29	1

5198 rows × 15 columns

In [5]:

```
df=pd.DataFrame(data['Prev Close'])
```

In [6]:

```
df
```

Out[6]:

```
      Prev Close
   0      580.00
   1     1554.45
   2     1678.85
   3     1813.20
   4     1958.30
 ...         ...
5193      819.25
5194      839.20
5195      840.50
5196      824.70
5197      842.05

5198 rows × 1 columns
```

In [7]:

```
data.Timestamp = pd.to_datetime(data.Date,format='%Y-%m-%d')
df.index = data.Timestamp
```

In [8]:

```
df = df.resample('M').mean()
```

In [9]:

```
df
```

Out[9]:

	Prev Close
Date	
2000-01-31	1773.614286
2000-02-29	2292.078571
2000-03-31	2043.504762
2000-04-30	1575.988889
2000-05-31	1283.884091
...	...
2020-07-31	622.628261
2020-08-31	705.233333
2020-09-30	763.259091
2020-10-31	849.233333
2020-11-30	828.007500

251 rows × 1 columns

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 251 entries, 2000-01-31 to 2020-11-30
Freq: M
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Prev Close  251 non-null    float64
dtypes: float64(1)
memory usage: 3.9 KB
```

In [11]:

```
df.describe()
```

Out[11]:

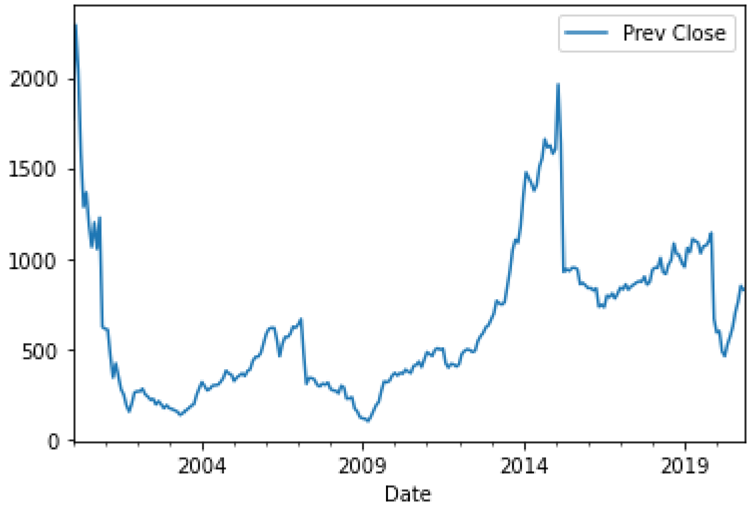
	Prev Close
count	251.000000
mean	639.493365
std	413.390164
min	99.035000
25%	315.706607
50%	524.128947
75%	871.507798
max	2292.078571

In [12]:

```
df.plot()
```

Out[12]:

```
<AxesSubplot:xlabel='Date'>
```



In [13]:

```
from statsmodels.tsa.stattools import adfuller
```

In [14]:

```
test_result=adfuller(df['Prev Close'])
test_result
```

Out[14]:

```
(-2.4715316944098067,
 0.12255549748663425,
 5,
 245,
 {'1%': -3.4573260719088132,
  '5%': -2.873410402808354,
  '10%': -2.573095980841316},
 2684.378436140892)
```

In [15]:

```
df['Seasonal_Difference']=df['Prev Close']-df['Prev Close'].shift(1)
## Again test dickey fuller test
test_result=adfuller(df['Seasonal_Difference'].dropna())
test_result
```

Out[15]:

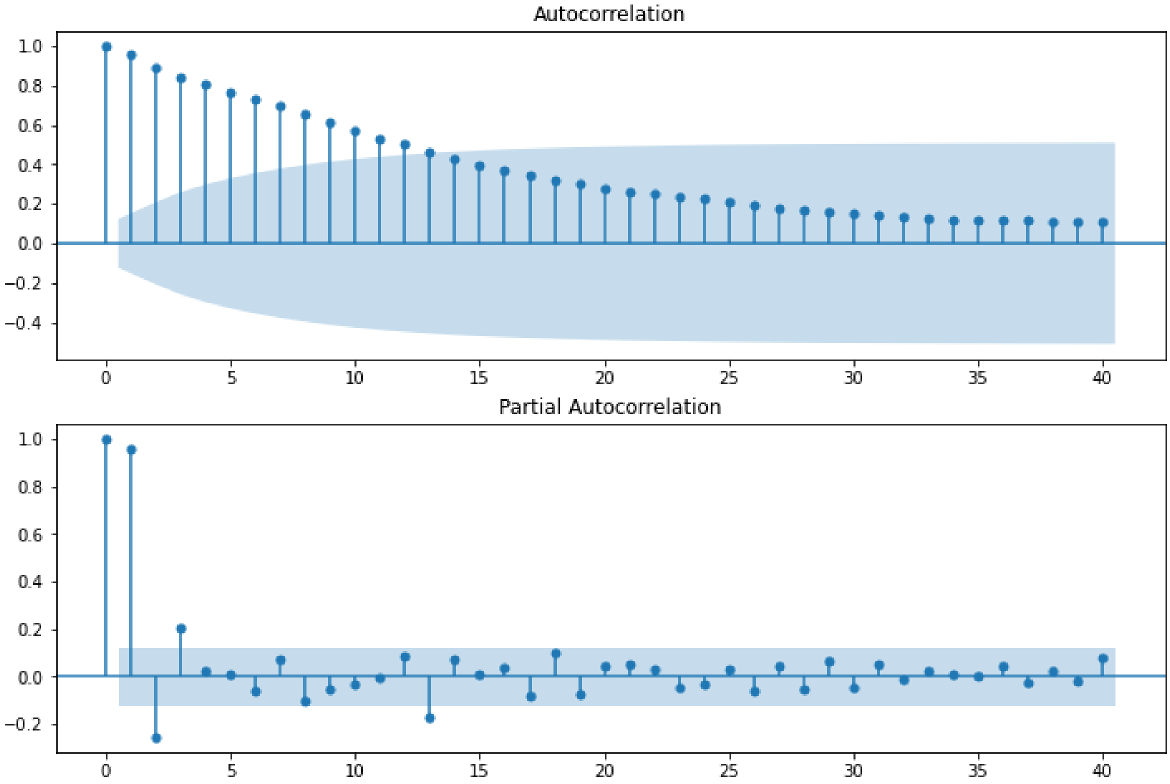
```
(-6.461448659819052,
 1.4398566680376188e-08,
 4,
 245,
 {'1%': -3.4573260719088132,
  '5%': -2.873410402808354,
  '10%': -2.573095980841316},
 2673.5323739735854)
```

In [16]:

```
import statsmodels.api as sm
```

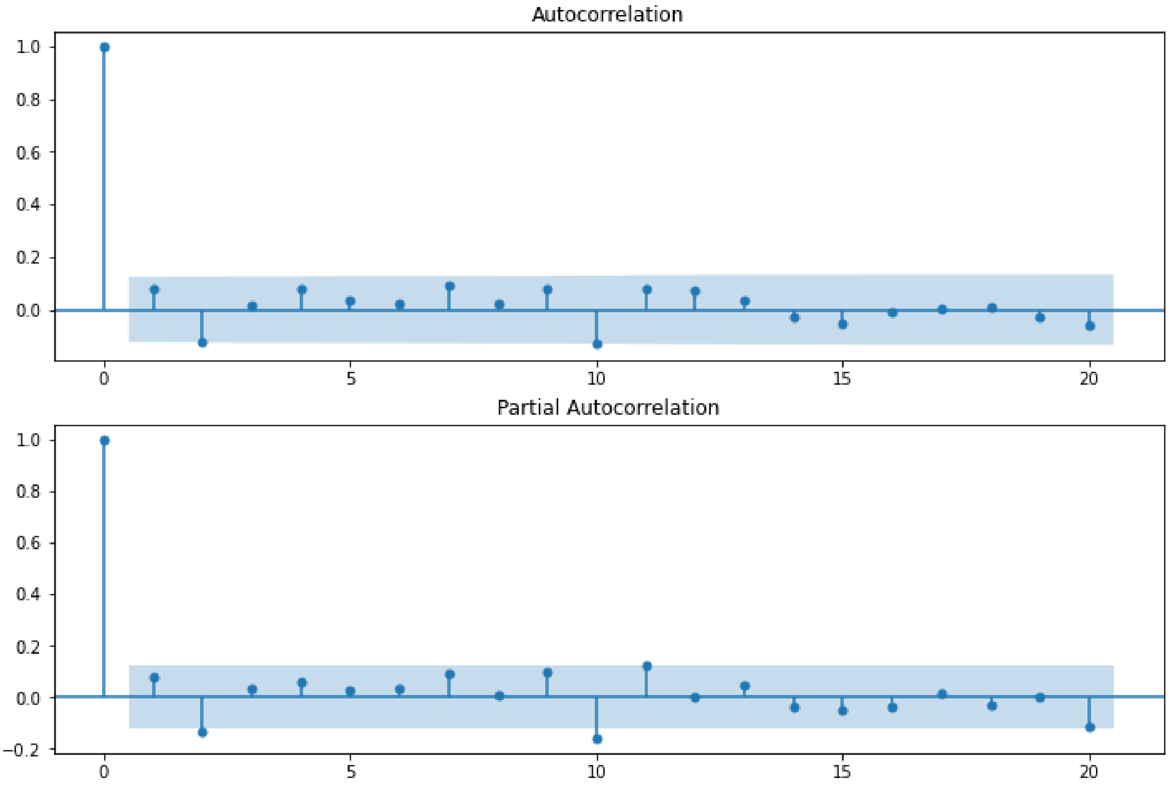
In [17]:

```
%matplotlib inline
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Prev Close'], lags=40, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Prev Close'], lags=40, ax=ax2)
```



In [18]:

```
%matplotlib inline
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Seasonal_Difference'].dropna(), lags=20, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Seasonal_Difference'].dropna(), lags=20, ax=ax2)
```



In [19]:

```
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARMA
# fit model
ARMAmodel = ARMA(df['Prev Close'], order=(1, 1))
ARmodel_fit = ARMAmodel.fit(disp=False)
```

In [20]:

```
actuals = df['Prev Close'][245:251]
actuals
```

Out[20]:

```
Date
2020-06-30    569.379545
2020-07-31    622.628261
2020-08-31    705.233333
2020-09-30    763.259091
2020-10-31    849.233333
2020-11-30    828.007500
Freq: M, Name: Prev Close, dtype: float64
```

In [21]:

```
ypredicted = ARmodel_fit.predict(245,250) # end point included
print(ypredicted)
```

```
Date
2020-06-30    540.692456
2020-07-31    579.961325
2020-08-31    633.676278
2020-09-30    718.126138
2020-10-31    770.173854
2020-11-30    858.668825
Freq: M, dtype: float64
```

In [22]:

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(actuals, ypredicted)
print('MAE: %f' % mae)
```

```
MAE: 49.627473
```

In [23]:

```
import itertools
i = j = range(0, 4)
ij = itertools.product(i,j)
for parameters in ij:
    try:
        mod = ARMA(df['Prev Close'],order=parameters)
        results = mod.fit()
        ypredicted = results.predict(245,250) # end point included
        mae = mean_absolute_error(actuals, ypredicted)
        print('ARMA{} - MAE:{}'.format(parameters, mae))
        #print('ARMA{} - AIC:{}'.format(parameters, results.aic))
    except:
        continue
```

```
ARMA(0, 0) - MAE:112.4564535605707
ARMA(0, 1) - MAE:79.01932583348815
ARMA(1, 0) - MAE:54.39842035955305
ARMA(1, 1) - MAE:49.62747294377228
ARMA(1, 2) - MAE:53.14149261928704
ARMA(1, 3) - MAE:53.40182856564545
ARMA(2, 0) - MAE:50.41343496667482
ARMA(2, 1) - MAE:48.723956275824754
ARMA(2, 2) - MAE:53.24381386106381
ARMA(3, 0) - MAE:53.65274849506577
ARMA(3, 1) - MAE:59.63536651918648
```

In [24]:

```
ARMAmodel = ARMA(df['Prev Close'], order=(2, 1))
ARmodel_fit = ARMAmodel.fit()
ypredicted = ARmodel_fit.predict(245,250)
print(ypredicted)
mae = mean_absolute_error(actuals, ypredicted)
print('MAE: %f' % mae)
print(ARmodel_fit.aic)
```

```
Date
2020-06-30    548.313110
2020-07-31    561.430993
2020-08-31    644.641265
2020-09-30    697.921383
2020-10-31    779.152553
2020-11-30    842.076978
Freq: M, dtype: float64
MAE: 48.723956
3032.6025444755005
```

In [25]:

```
actuals
```

Out[25]:

```
Date
2020-06-30    569.379545
2020-07-31    622.628261
2020-08-31    705.233333
2020-09-30    763.259091
2020-10-31    849.233333
2020-11-30    828.007500
Freq: M, Name: Prev Close, dtype: float64
```

In [26]:

```
# make prediction
ypredicted = ARmodel_fit.predict(len(df), len(df)+3)
print(ypredicted)
```

```
2020-12-31    831.682395
2021-01-31    827.556960
2021-02-28    830.298940
2021-03-31    827.033187
Freq: M, dtype: float64
```

In [27]:

```
from statsmodels.tsa.arima_model import ARIMA
```

In [28]:

```
ARIMAmodel = ARIMA(df['Prev Close'], order=(1, 1, 1))
ARIMA_model_fit = ARIMAmodel.fit(disp=False)

ypredicted = ARIMA_model_fit.predict(len(df)-6, len(df)-1, typ='levels')
print(ypredicted)
```

```
Date
2020-06-30    538.462629
2020-07-31    552.512231
2020-08-31    636.969824
2020-09-30    691.901295
2020-10-31    774.603529
2020-11-30    839.145865
Freq: M, dtype: float64
```

In [29]:

```
mae = mean_absolute_error(actuals, ypredicted)
print('MAE: %f' % mae)
print(ARIMA_model_fit.aic)
```

```
MAE: 54.403737
3017.428962222136
```

In [30]:

```
# make prediction
ypredicted = ARIMA_model_fit.predict(len(df), len(df)+3)
print(ypredicted)
```

```
2020-12-31    0.661140
2021-01-31   -7.752929
2021-02-28   -0.339359
2021-03-31   -6.871397
Freq: M, dtype: float64
```

In [31]:

```
import itertools
p= d = q = range(0, 4)
pdq = itertools.product(p,d,q)
for parameters in pdq:
    try:
        ARIMAmodel = ARIMA(df['Prev Close'], order=parameters)
        results = ARIMAmodel.fit()
        ypredicted = results.predict(245,250) # end point included
        mae = mean_absolute_error(actuals, ypredicted)
        print('ARIMA{} - MAE:{}'.format(parameters, mae))
    #print('ARMA{} - AIC:{}'.format(parameters, results.aic))
    except:
        continue
```

```
ARIMA(0, 0, 0) - MAE:112.4564535605707
ARIMA(0, 0, 1) - MAE:79.01932583348815
ARIMA(0, 1, 0) - MAE:726.7392711261685
ARIMA(0, 1, 1) - MAE:718.8696634563181
ARIMA(0, 1, 2) - MAE:725.6268096638801
ARIMA(0, 1, 3) - MAE:725.6730427737581
ARIMA(0, 2, 0) - MAE:725.1242741802898
ARIMA(0, 2, 1) - MAE:776.6085148306483
ARIMA(0, 2, 2) - MAE:771.4072996065757
ARIMA(0, 2, 3) - MAE:800.9592325485969
ARIMA(1, 0, 0) - MAE:54.39842035955305
ARIMA(1, 0, 1) - MAE:49.62747294377228
ARIMA(1, 0, 2) - MAE:53.14149261928704
ARIMA(1, 0, 3) - MAE:53.40182856564545
ARIMA(1, 1, 0) - MAE:720.7360516381938
ARIMA(1, 1, 1) - MAE:723.0013670364191
ARIMA(1, 1, 2) - MAE:725.6456472037179
ARIMA(1, 2, 0) - MAE:731.7079250514474
ARIMA(1, 2, 1) - MAE:772.7572416415072
ARIMA(1, 2, 2) - MAE:773.4854097062538
ARIMA(1, 2, 3) - MAE:802.5477133351736
ARIMA(2, 0, 0) - MAE:50.41343496667482
ARIMA(2, 0, 1) - MAE:48.723956275824754
ARIMA(2, 0, 2) - MAE:53.24381386106381
ARIMA(2, 1, 0) - MAE:727.1808800371963
ARIMA(2, 1, 1) - MAE:719.1762883497955
ARIMA(2, 1, 2) - MAE:720.9287155293206
ARIMA(2, 1, 3) - MAE:757.0668882052481
ARIMA(2, 2, 0) - MAE:746.1094587782953
ARIMA(2, 2, 1) - MAE:802.1589641220171
ARIMA(2, 2, 2) - MAE:804.646987580595
ARIMA(2, 2, 3) - MAE:776.3051832902329
ARIMA(3, 0, 0) - MAE:53.65274849506577
ARIMA(3, 0, 1) - MAE:59.63536651918648
ARIMA(3, 1, 0) - MAE:726.4607595504107
ARIMA(3, 1, 1) - MAE:740.0123181329737
ARIMA(3, 1, 2) - MAE:748.2158835685942
```

```
C:\Users\divya\anaconda3\lib\site-packages\statsmodels\base\model.py:547:
HessianInversionWarning: Inverting hessian failed, no bse or cov_params av
ailable
  warnings.warn('Inverting hessian failed, no bse or cov_params '
C:\Users\divya\anaconda3\lib\site-packages\statsmodels\base\model.py:566:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Ch
```

```
eck mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "

ARIMA(3, 1, 3) - MAE:725.6714464853422
ARIMA(3, 2, 0) - MAE:757.8151397964543
ARIMA(3, 2, 1) - MAE:805.424456323712
ARIMA(3, 2, 2) - MAE:794.1215291783145
ARIMA(3, 2, 3) - MAE:802.7321919281849
```

In [32]:

```
ARIMAmodel = ARIMA(df['Prev Close'], order=(2, 0, 1))
ARIMA_model_fit = ARIMAmodel.fit()

ypredicted = ARIMA_model_fit.predict(len(df)-6, len(df)-1, typ='levels')
print(ypredicted)
```

```
Date
2020-06-30    548.313110
2020-07-31    561.430993
2020-08-31    644.641265
2020-09-30    697.921383
2020-10-31    779.152553
2020-11-30    842.076978
Freq: M, dtype: float64
```

In [33]:

```
mae = mean_absolute_error(actuals, ypredicted)
print('MAE: %f' % mae)
print(ARIMA_model_fit.aic)
```

```
MAE: 48.723956
3032.6025444755005
```

In [34]:

```
# make prediction
ypredicted = ARIMA_model_fit.predict(len(df), len(df)+3)
print(ypredicted)
```

```
2020-12-31    831.682395
2021-01-31    827.556960
2021-02-28    830.298940
2021-03-31    827.033187
Freq: M, dtype: float64
```

In [ ]: