Analysis and Prediction of the Price for the Agriculture Commodity Data

Data

```
    * Used data.csv from the mail.
    * Contains information about the different agriculture commodity.
```

In [133]:

```
## Importing all the necessary Packages
import pandas as pd
import re
import  matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from datetime import timedelta, date
```

In [42]:

```
## Importing the Data
data = pd.read_csv("data.csv")
data
```

Out[42]:

	priceDate	itemName	state	mandiName	arrivals	unitArrivals	variety	minPrice	maxPrice	modalPrice	priceUnit
0	2005-04-11	Potato	NCT of Delhi	Azadpur	106.0	Tonnes	F.A.Q.	300.0	663.0	475.0	Rs/Quintal
1	2005-04-12	Potato	NCT of Delhi	Azadpur	1745.0	Tonnes	F.A.Q.	325.0	688.0	500.0	Rs/Quintal
2	2005-04-13	Potato	NCT of Delhi	Azadpur	1233.0	Tonnes	F.A.Q.	300.0	688.0	488.0	Rs/Quintal
3	2005-04-16	Potato	NCT of Delhi	Azadpur	1654.0	Tonnes	F.A.Q.	350.0	650.0	475.0	Rs/Quintal
4	2005-04-18	Potato	NCT of Delhi	Azadpur	26.0	Tonnes	F.A.Q.	350.0	650.0	475.0	Rs/Quintal
...	...	...	...	...	...	...	...	...	...	...	...
3800	2018-08-27	Potato	NCT of Delhi	Azadpur	97.1	Tonnes	Potato	700.0	2400.0	1410.0	Rs/Quintal
3801	2018-08-28	Potato	NCT of Delhi	Azadpur	973.5	Tonnes	Potato	700.0	2400.0	1410.0	Rs/Quintal
3802	2018-08-29	Potato	NCT of Delhi	Azadpur	1317.9	Tonnes	Potato	700.0	2400.0	1410.0	Rs/Quintal
3803	2018-08-30	Potato	NCT of Delhi	Azadpur	1375.3	Tonnes	Potato	700.0	2400.0	1410.0	Rs/Quintal
3804	2018-08-31	Potato	NCT of Delhi	Azadpur	1121.0	Tonnes	Potato	700.0	2400.0	1411.0	Rs/Quintal

3805 rows × 11 columns

In [43]:

```
data.columns ## Getting all number of columns
```

Out[43]:

```
Index(['priceDate', 'itemName', 'state', 'mandiName', 'arrivals',
       'unitArrivals', 'variety', 'minPrice', 'maxPrice', 'modalPrice',
       'priceUnit'],
      dtype='object')
```

In [44]:

```python
## Getting the all the unique values in each column.
## Also getting the number of unique value in each column.
for i in data.columns:
    print(data[i].unique())
    print('Number of Unique values in {} is {}'.format(i,len(data[i].unique())))
    print()
    print()
```

```
['2005-04-11' '2005-04-12' '2005-04-13' ... '2018-08-29' '2018-08-30'
 '2018-08-31']
Number of Unique values in priceDate is 3745


['Potato']
Number of Unique values in itemName is 1


['NCT of Delhi']
Number of Unique values in state is 1


['Azadpur']
Number of Unique values in mandiName is 1


[ 106.  1745.  1233.   ... 1317.9 1375.3 1121. ]
Number of Unique values in arrivals is 3073


['Tonnes']
Number of Unique values in unitArrivals is 1


['F.A.Q.' 'Potato' 'Other']
Number of Unique values in variety is 3


[ 300.  325.  350.  313.  344.  375.  338.  438.  425.  406.  400.  413.
  388.  363.  450.  500.  563.  531.  725.  763.  594.  475.  625.  525.
  280.  463.  225.  422.  433.  444.  431.  275.  250.  600.  560.  590.
  630.  688.  750.  875.  281.  188.  213.  219.  200.  480.  538.  550.
  575.  700.  800.  900.  650.   nan  288.  150. 1300.  175.  160.  120.
  140.  180.  220.  238.  240.  260.  580.  588.  788.  775.  813.  844.
 1125.  938. 1000.  663.  640.  540.  520.  660.  440.  430.  460.  360.
  340.  320.  125.   60.  100.  760.  880.  840.  720.  960. 1050.  975.
 1040.  656.  680.  850.  668. 1100. 1200. 1240.  940.  950.  970.  740.
  490. 1625.  620.  920. 1160. 1280. 1260. 1250. 1063. 1350. 1360. 1188.
 1340. 1400. 1450. 1375. 1900. 1500. 1560. 1550. 1600. 1438. 1800. 1750.
 1875. 2000. 2100. 1700. 1650.  946. 1060.  470.  270.  469.  190.  380.]
Number of Unique values in minPrice is 144


[  663.   688.   650.   675.   700.   713.   750.   775.   788.   875.
   850.   938.  1000.  1063.  1075.   950.  1125.  1150.  1188.   900.
   975.   925.   906.  1250.  1156.  1025.  1200.  1031.   969.  1050.
  1094.   600.  1450.   813.   588.   575.   638.   625.  1225.   725.
   988.   350.   400.   963.   450.  1038.   500.   300.   550.  1088.
   590.   610.   620.   670.  1213.  1375.  1500.   363.   388.   513.
   463.   438.   425.   475.   800.   825.   535.  1163.  1350.  1313.
  1400.   531.   488.   519.   525.  1600.   290.   563.  1100.  1013.
  1175.   440.   360.   325.   320.   288.   280.   275.   380.   413.
   538.   640.   680.  2000.   780.   763.  1120.  1140.  1440.  1620.
  1630.  1563.  1680.  1625.  1750.  1800.  1850.  2375.  1688.  1640.
  1626.  1560.  1525.  1875.  1813.  2125.  2250.  2100.  2063.  2025.
  1975.  2425.  2313.  2200.  2128.  2050.  1700.  1650.  1160.   760.
   740.   660.   720.   520.    nan   530.   560.   540.   580.   480.
   460.   860.  1360.  1300.  1550.  1520.  1438.  1340.  1219.  1320.
  1260.  1420.   960.   880.   420.   430.   840.  1040.  1020.  1080.
  1656.  1900.  2400.  2350.  2300.  2224.  2500.  1950.  1938.  1240.
  2875.   830.   920.  1060.  1760.  2750.  1860.  1840.  2040.  2160.
  2650.  2900.  3000.  3400.  2813.  2625.  2800.  3125.  2600.  1720.
  1430.  1220.   980.  3500.   940.  1540.  2188.  2325. 20580.  2260.
  2700.  3100.  3200.  3300.  3800.  4000.  2850.  3750.   870.  7600.
  1880.  1920.  1480.  1960.  2360.  2420.  1380.  1940.]
Number of Unique values in maxPrice is 218


[ 475.  500.  488. ... 1305. 1392. 1411.]
Number of Unique values in modalPrice is 1078


['Rs/Quintal']
Number of Unique values in priceUnit is 1
```

Since the column values are same for the columns:

```
* Priceunit
* State
* itemName
* Mandiname
* unitarrivals
```

It means even if we remove those column , it won't affect our model.

We can generalize it as this data is of prices of Potato in Azadpur Mand in Delhi where Price Unit is mesured in Rs/Quintal and Unit of arrivals in in Tonnes.

And hence can be removed.

In [45]:

```
## So droping the columns accoring to the above analysis
p_data = data.drop(['priceUnit','state','itemName','mandiName','unitArrivals'],axis=1)
```

In [46]:

```
p_data
```

Out[46]:

	priceDate	arrivals	variety	minPrice	maxPrice	modalPrice
0	2005-04-11	106.0	F.A.Q.	300.0	663.0	475.0
1	2005-04-12	1745.0	F.A.Q.	325.0	688.0	500.0
2	2005-04-13	1233.0	F.A.Q.	300.0	688.0	488.0
3	2005-04-16	1654.0	F.A.Q.	350.0	650.0	475.0
4	2005-04-18	26.0	F.A.Q.	350.0	650.0	475.0
...	...	...	...	...	...	...
3800	2018-08-27	97.1	Potato	700.0	2400.0	1410.0
3801	2018-08-28	973.5	Potato	700.0	2400.0	1410.0
3802	2018-08-29	1317.9	Potato	700.0	2400.0	1410.0
3803	2018-08-30	1375.3	Potato	700.0	2400.0	1410.0
3804	2018-08-31	1121.0	Potato	700.0	2400.0	1411.0

3805 rows × 6 columns

Now we are left with 6 columns.

In [47]:

```
## Finding the number of null values in the data.
```

```
p_data.isnull().sum()
```

Out[47]:

```
priceDate      0
arrivals      27
variety        0
minPrice       1
maxPrice       1
modalPrice    10
dtype: int64
```

In [48]:

```
##Percentage of Null data for each major entity
print(f'percentage of null_value:{(p_data.arrivals.isnull().sum()/p_data.shape[0])*100}')
print()
print()
print(f'percentage of null_value:{(p_data.minPrice.isnull().sum()/p_data.minPrice.shape[0])*100}')
```

```
percentage of null_value:0.709592641261498


percentage of null_value:0.026281208935611037
```

In [49]:

```
## Since very less percentage of points are NULL , we can drop them.
```

In [50]:

```
## Dropping all the rows with NULL VALUES
p_data = p_data.dropna()
p_data
```

Out[50]:

	priceDate	arrivals	variety	minPrice	maxPrice	modalPrice
0	2005-04-11	106.0	F.A.Q.	300.0	663.0	475.0
1	2005-04-12	1745.0	F.A.Q.	325.0	688.0	500.0
2	2005-04-13	1233.0	F.A.Q.	300.0	688.0	488.0
3	2005-04-16	1654.0	F.A.Q.	350.0	650.0	475.0
4	2005-04-18	26.0	F.A.Q.	350.0	650.0	475.0
...	...	...	...	...	...	...
3800	2018-08-27	97.1	Potato	700.0	2400.0	1410.0
3801	2018-08-28	973.5	Potato	700.0	2400.0	1410.0
3802	2018-08-29	1317.9	Potato	700.0	2400.0	1410.0
3803	2018-08-30	1375.3	Potato	700.0	2400.0	1410.0
3804	2018-08-31	1121.0	Potato	700.0	2400.0	1411.0

3768 rows × 6 columns

In [51]:

```
## Hence how there are no null values
p_data.isnull().sum()
```

Out[51]:

```
priceDate     0
arrivals      0
variety       0
minPrice      0
maxPrice      0
```

```
modalPrice    0
dtype: int64
```

In [54]:

```
## Saving the cleaned data in another csv file.
p_data = p_data.drop(['variety'],axis=1)
```

In [52]:

```
## So here we have succefully removed all the NULL Values
```

In [55]:

```
p_data
```

Out[55]:

	priceDate	arrivals	minPrice	maxPrice	modalPrice
0	2005-04-11	106.0	300.0	663.0	475.0
1	2005-04-12	1745.0	325.0	688.0	500.0
2	2005-04-13	1233.0	300.0	688.0	488.0
3	2005-04-16	1654.0	350.0	650.0	475.0
4	2005-04-18	26.0	350.0	650.0	475.0
...	...	...	...	...	...
3800	2018-08-27	97.1	700.0	2400.0	1410.0
3801	2018-08-28	973.5	700.0	2400.0	1410.0
3802	2018-08-29	1317.9	700.0	2400.0	1410.0
3803	2018-08-30	1375.3	700.0	2400.0	1410.0
3804	2018-08-31	1121.0	700.0	2400.0	1411.0

3768 rows × 5 columns

In [85]:

```
## Loading the saved csv file.
p_data.to_csv('p_data.csv',index=False)
```

In [95]:

```
## Setting the priceDate as index
p_data = pd.read_csv('p_data.csv')
tdi = pd.DatetimeIndex(p_data.priceDate)
p_data.set_index(tdi,inplace=True)
p_data.drop(columns='priceDate',inplace=True)
p_data.index.name = 'priceDate'
```

In [96]:

```
p_data.head()
```

Out[96]:

	arrivals	minPrice	maxPrice	modalPrice
priceDate				
2005-04-11	106.0	300.0	663.0	475.0
2005-04-12	1745.0	325.0	688.0	500.0
2005-04-13	1233.0	300.0	688.0	488.0
2005-04-16	1654.0	350.0	650.0	475.0
2005-04-18	26.0	350.0	650.0	475.0

	arrivals	minPrice	maxPrice	modalPrice

In [97]:

```
## Creating 3 diferent tables for each Prediction
mindata = p_data[['minPrice']]
maxdata = p_data[['maxPrice']]
modalPrice = p_data[['modalPrice']]
```

Model For prediction of minPrice

In [99]:

```
mindata.head()
```

Out[99]:

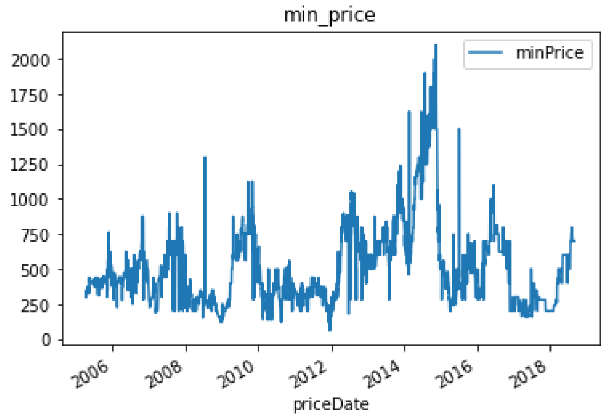
	minPrice
priceDate	
2005-04-11	300.0
2005-04-12	325.0
2005-04-13	300.0
2005-04-16	350.0
2005-04-18	350.0

In [119]:

```
mindata[['minPrice']].plot(title='min_price')
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a25cd9a808>
```



In [102]:

```
## Decoposing the tie series and will check its trends and Seasonality in data
```

In [115]:

```
fitted_model =
ExponentialSmoothing(mindata['minPrice'],trend='mul',seasonal='mul',seasonal_periods=12).fit()
min_score_predictions = fitted_model.forecast(30)
```

```
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:583:
ValueWarning: A date index has been provided, but it has no associated frequency information and s
o will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
```

```
packages\statsmodels\tsa\holtwinters\model.py:429: FutureWarning: After 0.13 initialization must b
e handled at model creation
  FutureWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:80: RuntimeWarning: overflow encountered in matmul
  return err.T @ err
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:922: ConvergenceWarning: Optimization failed to conv
erge. Check mle_retvals.
  ConvergenceWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:379:
ValueWarning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  ValueWarning)
```

Model for prediction of Maxprice

In [118]:

```
maxdata.head()
```

Out[118]:

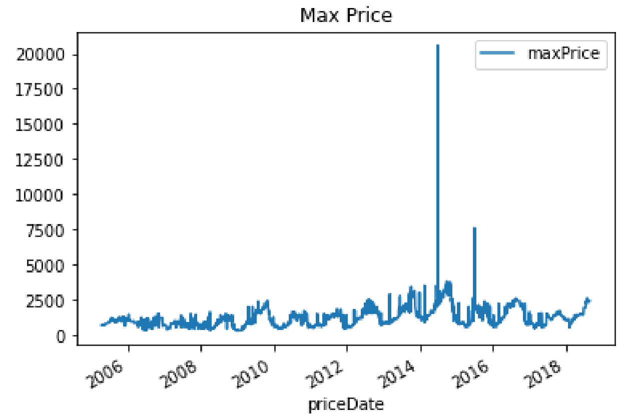
	maxPrice
priceDate	
2005-04-11	663.0
2005-04-12	688.0
2005-04-13	688.0
2005-04-16	650.0
2005-04-18	650.0

In [120]:

```
maxdata[['maxPrice']].plot(title = 'Max Price')
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a25aa437c8>
```



In [122]:

```
## Theres definately an outlier
```

In [124]:

```
fitted_model =
ExponentialSmoothing(maxdata['maxPrice'],trend='mul',seasonal='mul',seasonal_periods=12).fit()
max_score_predictions = fitted_model.forecast(30)
```

```
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:583:
ValueWarning: A date index has been provided, but it has no associated frequency information and s
o will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:429: FutureWarning: After 0.13 initialization must b
e handled at model creation
  FutureWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:80: RuntimeWarning: overflow encountered in matmul
  return err.T @ err
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:922: ConvergenceWarning: Optimization failed to conv
erge. Check mle_retvals.
  ConvergenceWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:379:
ValueWarning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  ValueWarning)
```

In [ ]:

Prediction for ModalPrice

In [125]:

```
modalPrice.head()
```

Out[125]:

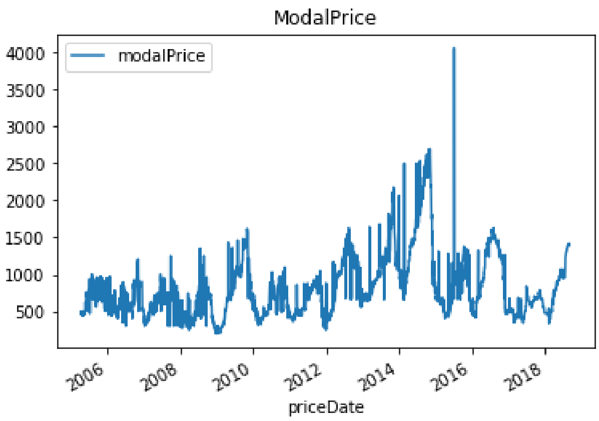
	modalPrice
priceDate	
2005-04-11	475.0
2005-04-12	500.0
2005-04-13	488.0
2005-04-16	475.0
2005-04-18	475.0

In [126]:

```
modalPrice[['modalPrice']].plot(title = 'ModalPrice')
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a25cf11348>
```



In [127]:

In [127]:

```
## There definately an outlier after 2015
```

In [132]:

```
fitted_model =
ExponentialSmoothing(modalPrice['modalPrice'],trend='mul',seasonal='mul',seasonal_periods=12).fit(
)
modal_score_predictions = fitted_model.forecast(30)
```

```
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:583:
ValueWarning: A date index has been provided, but it has no associated frequency information and s
o will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:429: FutureWarning: After 0.13 initialization must b
e handled at model creation
  FutureWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:80: RuntimeWarning: overflow encountered in matmul
  return err.T @ err
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:922: ConvergenceWarning: Optimization failed to conv
erge. Check mle_retvals.
  ConvergenceWarning,
C:\Users\Admin\miniconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:379:
ValueWarning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  ValueWarning)
```

Creating the Final Dataframe

In [145]:

```
## The Starting date will be last Date in the Data
## End date will be lastDate + 30 days
print('The starting date will be {}'.format(p_data.index[-1]))
print('The end date will be {}'.format(p_data.index[-1] + timedelta(days=30)))
```

```
The starting date will be 2018-08-31 00:00:00
The end date will be 2018-09-30 00:00:00
```

In [163]:

```
## Generating the timestamp of the predicted values
## https://www.w3resource.com/python-exercises/date-time-exercise/python-date-time-exercise-50.php
def daterange(date1, date2):
    for n in range(int ((date2 - date1).days)+1):
        yield date1 + timedelta(n)


Prediction_dates = []
start_dt = date(2018,8,31)
end_dt = date(2018, 9, 30)
for dt in daterange(start_dt, end_dt):
    Prediction_dates.append(dt.strftime("%Y-%m-%d"))
```

In [177]:

```
## Creating the Final Predicted Dataframe
data =
{'priceDate':Prediction_dates[1::],'minPrice':min_score_predictions,'maxPrice':max_score_prediction
s,'modalPrice':modal_score_predictions}
Predicted_Data = pd.DataFrame(data)
```

In [179]:

```
tdi = pd.DatetimeIndex(Predicted_Data.priceDate)
Predicted_Data.set_index(tdi,inplace=True)
Predicted_Data.drop(columns='priceDate',inplace=True)
Predicted_Data.index.name = 'priceDate'
```

In [185]:

```
Predicted_Data.head()
```

Out[185]:

	minPrice	maxPrice	modalPrice
priceDate			
2018-09-01	709.192480	2367.538476	1394.257760
2018-09-02	722.657934	2409.858004	1409.378100
2018-09-03	716.313258	2414.559900	1404.943079
2018-09-04	724.435470	2423.168117	1416.156648
2018-09-05	736.867935	2384.320828	1429.953853

In [187]:

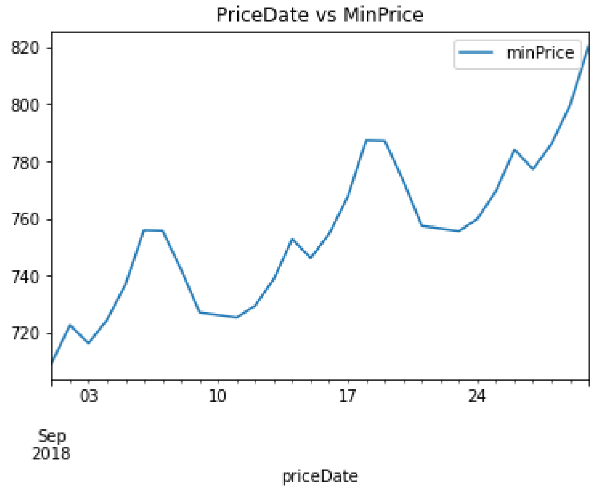
```
Predicted_Data.shape
```

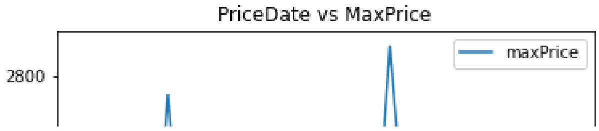
Out[187]:

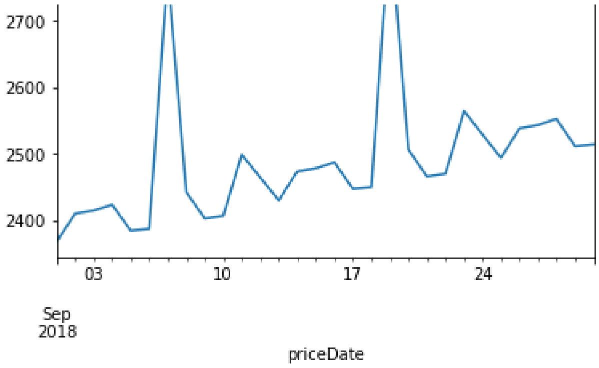
```
(30, 3)
```

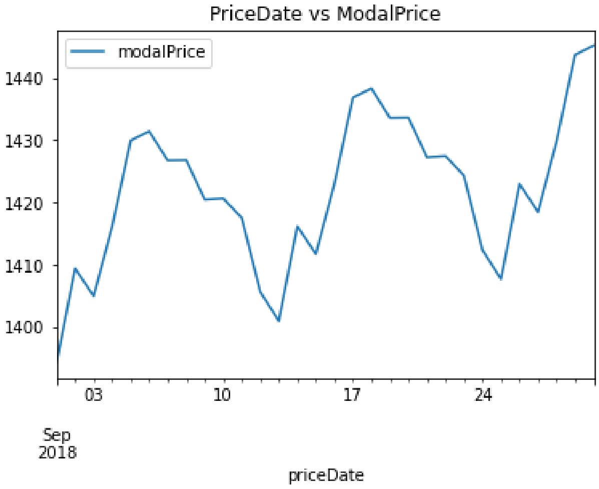
In [183]:

```
Predicted_Data[['minPrice']].plot(title = 'PriceDate vs MinPrice')
print()
print()
Predicted_Data[['maxPrice']].plot(title = 'PriceDate vs MaxPrice')
print()
print()
Predicted_Data[['modalPrice']].plot(title = 'PriceDate vs ModalPrice')
print()
print()
```









Observation

```
    * Minimum keeps increasing with the time
    * Maximum Price remains except for the first two days of 1st and 3rd week.
    * Model Price is at peak in first 2-3 days of 1st and 3rd week.
```