Importing Pandas, Matplotlib, Numpy Libraries

Dataset is loaded by linking via Google Drive and check for Missing Values

1. Upload the Dataset to Google Drive
2. Mount the Drive and Read Dataset using Pandas

In [37]:
```
import pandas as pd
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
df =pd.read_csv("drive/My Drive/IBM_Project/Dataset/Electricity.csv")
missing_values = df.isnull()
missing_values
```

```
<ipython-input-3-4383926d33ab>:1: DtypeWarning: Columns (9,10,11,14,15,16,17) have mixed types. Specify dtype option on import or
set low_memory=False.
  df =pd.read_csv("drive/My Drive/IBM_Project/Dataset/Electricity.csv")
```

Out [3]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Int
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
38009	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
38010	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
38011	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
38012	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
38013	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

38014 rows × 18 columns

In [ ]:
```
for column in df.columns:
    if df[column].dtype == 'object' and df[column].str.contains('\?').any():
        print(f"Column '{column}' contains '?'")
```

```
Column 'ForecastWindProduction' contains '?'
Column 'SystemLoadEA' contains '?'
Column 'SMPEA' contains '?'
Column 'ORKTemperature' contains '?'
Column 'ORKWindspeed' contains '?'
Column 'CO2Intensity' contains '?'
Column 'ActualWindProduction' contains '?'
Column 'SystemLoadEP2' contains '?'
Column 'SMPEP2' contains '?'
```

Replace the Missing Values using NaN values by Pandas library

In [ ]:
```
df.replace('?', np.nan, inplace=True)
df
```

Out [5]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2I
0	01/11/2011 00:00	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6.00	9.30	600.7
1	01/11/2011 00:30	None	0	1	44	1	11	2011	1	321.80	3196.66	49.26	6.00	11.10	605.4
2	01/11/2011 01:00	None	0	1	44	1	11	2011	2	328.57	3060.71	49.10	5.00	11.10	589.9
3	01/11/2011 01:30	None	0	1	44	1	11	2011	3	335.60	2945.56	48.04	6.00	9.30	585.9
4	01/11/2011 02:00	None	0	1	44	1	11	2011	4	342.90	2849.34	33.75	6.00	11.10	571.5
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
38009	31/12/2013 21:30	New Year's	1	1	1	31	12	2013	43	1179.14	3932.22	34.51	6.00	22.20	285.3

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2I
		Eve													
38010	31/12/2013 22:00	New Year's Eve	1	1	1	31	12	2013	44	1152.01	3821.44	33.83	5.00	24.10	278.3
38011	31/12/2013 22:30	New Year's Eve	1	1	1	31	12	2013	45	1123.67	3724.21	31.75	4.00	20.40	280.9
38012	31/12/2013 23:00	New Year's Eve	1	1	1	31	12	2013	46	1094.24	3638.16	33.83	5.00	14.80	302.4
38013	31/12/2013 23:30	New Year's Eve	1	1	1	31	12	2013	47	1064.0	3624.25	33.83	5.00	16.70	308.0

38014 rows × 18 columns

Convert the Datatype of the columns in the Dataset as per their Requirements

In [ ]:
```
df["DateTime"] = df['DateTime'].astype('datetime64')
df["ForecastWindProduction"] = df['ForecastWindProduction'].astype('float64')
df["SystemLoadEA"] = df['SystemLoadEA'].astype('float64')
df["SMPEA"] = df['SMPEA'].astype('float64')
df["ORKTemperature"] = df['ORKTemperature'].astype('float64')
df["ORKWindspeed"] = df['ORKWindspeed'].astype('float64')
df["CO2Intensity"] = df['CO2Intensity'].astype('float64')
df["ActualWindProduction"] = df['ActualWindProduction'].astype('float64')
df["SystemLoadEP2"] = df['SystemLoadEP2'].astype('float64')
df["SMPEP2"] = df['SMPEP2'].astype('float64')
df.dtypes
```

Out [6]:
```
DateTime                 datetime64[ns]
Holiday                          object
HolidayFlag                       int64
DayOfWeek                         int64
WeekOfYear                        int64
Day                               int64
Month                             int64
Year                              int64
PeriodOfDay                       int64
ForecastWindProduction          float64
SystemLoadEA                    float64
SMPEA                           float64
ORKTemperature                  float64
ORKWindspeed                    float64
CO2Intensity                    float64
ActualWindProduction            float64
SystemLoadEP2                   float64
SMPEP2                          float64
dtype: object
```

In [ ]:
```
print ("\nMissing values :  ", df.isnull().any())
```

```
Missing values :   DateTime                 False
Holiday                  False
HolidayFlag              False
DayOfWeek                False
WeekOfYear               False
Day                      False
Month                    False
Year                     False
PeriodOfDay              False
ForecastWindProduction    True
SystemLoadEA              True
SMPEA                     True
ORKTemperature            True
ORKWindspeed              True
CO2Intensity              True
ActualWindProduction      True
SystemLoadEP2             True
SMPEP2                    True
dtype: bool
```

Handle Missing Values using ffill method to replace NaN Values

In [ ]:
```
df['ForecastWindProduction']=df['ForecastWindProduction'].fillna(method='ffill')
df['SystemLoadEA']=df['SystemLoadEA'].fillna(method='ffill')
df['SMPEA']=df['SMPEA'].fillna(method='ffill')
df['ORKTemperature']=df['ORKTemperature'].fillna(method='ffill')
df['ORKWindspeed']=df['ORKWindspeed'].fillna(method='ffill')
df['CO2Intensity']=df['CO2Intensity'].fillna(method='ffill')
df['ActualWindProduction']=df['ActualWindProduction'].fillna(method='ffill')
df['SystemLoadEP2']=df['SystemLoadEP2'].fillna(method='ffill')
df['SMPEP2']=df['SMPEP2'].fillna(method='ffill')
```

In [ ]:
```
print ("\nMissing values :  ", df.isnull().any())
```

```
Missing values :   DateTime                 False
Holiday                  False
HolidayFlag              False
DayOfWeek                False
WeekOfYear               False
Day                      False
Month                    False
Year                     False
PeriodOfDay              False
ForecastWindProduction   False
SystemLoadEA             False
SMPEA                    False
ORKTemperature           False
ORKWindspeed             False
CO2Intensity             False
ActualWindProduction     False
SystemLoadEP2            False
SMPEP2                   False
dtype: bool
```

Import Plotly Library and Plot the Target Column

In [ ]:
```
import plotly.express as px
```

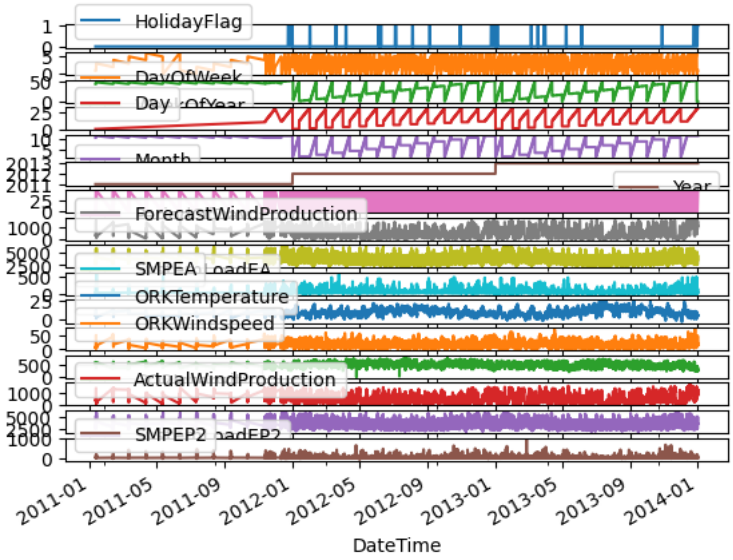
In [ ]:
```
fig = px.line(df, x='DateTime', y='SMPEP2', title='Electricity Price')
fig.update_xaxes(
    rangeslider_visible=True,
    rangeselector=dict(
        buttons=list([
            dict(step="all")
        ])
    )
)
fig.show()
```

Set DateTime column as Index and plot the Subplots

In [ ]:
```
el_df=df.set_index('DateTime')
```

In [ ]:
```
el_df.plot(subplots=True)
```

Out [13]:
```
array([<Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>], dtype=object)
```



Resample the Dataset and Plot the New SubPlots

In [ ]:
```
el_df.resample('M').mean()
```

```
<ipython-input-14-421011436e0d>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False.
Either specify numeric_only or select only columns which should be valid for the function.
```

Out [14]:

DateTime	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Int
2011-01-31	0.000000	2.000000	46.000000	1.000000	11.500000	2011.0	23.500000	567.916771	4433.788125	61.189167	6.520833	17.005208	530.58
2011-02-28	0.000000	3.000000	46.000000	2.000000	11.500000	2011.0	23.500000	1054.793229	4456.451979	57.025104	9.427083	32.725000	462.59
2011-03-31	0.000000	4.000000	46.000000	3.000000	11.500000	2011.0	23.500000	723.956667	4259.082917	53.261458	9.895833	21.303125	460.77
2011-04-30	0.000000	5.000000	46.000000	4.000000	11.500000	2011.0	23.500000	474.091979	4156.697708	52.314063	6.885417	13.996875	529.71
2011-05-31	0.000000	2.500000	46.500000	5.000000	11.500000	2011.0	23.500000	621.892292	4302.408125	57.051979	4.916667	15.882292	515.88
2011-06-30	0.000000	3.500000	46.500000	6.000000	11.500000	2011.0	23.500000	613.782917	4283.031042	53.654792	6.229167	12.566667	497.78
2011-07-31	0.000000	1.000000	47.000000	7.000000	11.500000	2011.0	23.500000	608.573958	4540.860104	60.758438	6.875000	17.119792	463.94
2011-08-31	0.000000	2.000000	47.000000	8.000000	11.500000	2011.0	23.500000	817.929271	4652.903854	57.753750	8.718750	26.794792	431.22
2011-09-30	0.000000	3.000000	47.000000	9.000000	11.500000	2011.0	23.500000	691.819792	4587.447917	62.579792	6.927083	17.117708	476.58
2011-10-31	0.000000	4.000000	47.000000	10.000000	11.500000	2011.0	23.500000	698.289688	4367.038229	56.049062	7.718750	19.370833	491.56
2011-11-30	0.000000	3.050000	46.700000	20.450000	11.050000	2011.0	23.500000	850.949271	4263.230042	59.390302	9.831250	24.163854	451.29
2011-12-31	0.190476	2.952381	50.666667	21.047619	11.952381	2011.0	23.500000	929.351746	4433.362411	57.791230	6.905754	23.289385	429.59
2012-01-31	0.000000	3.258065	13.161290	13.870968	3.129032	2012.0	23.500000	622.410491	4239.651028	58.881573	8.049059	19.163038	488.01
2012-02-29	0.034483	2.724138	14.689655	13.137931	3.862069	2012.0	23.500000	579.290014	4193.543807	60.356042	8.714799	19.009267	515.13
2012-03-31	0.032301	2.866756	16.909825	14.631225	4.356662	2012.0	23.528264	448.149764	4053.695128	60.550249	8.679677	17.698250	517.13
2012-04-30	0.033333	3.133333	19.700000	14.500000	5.000000	2012.0	23.500000	555.787521	3923.846694	63.923271	7.341667	21.949583	520.84
2012-05-31	0.000000	2.870968	22.354839	15.419355	5.580645	2012.0	23.500000	313.707782	3889.033226	62.993333	10.315860	17.036761	524.22
2012-06-30	0.066667	2.966667	25.066667	15.300000	6.200000	2012.0	23.500000	386.359576	3838.298840	58.928333	11.109722	17.109653	528.99
2012-07-31	0.064516	3.096774	27.774194	16.193548	6.806452	2012.0	23.500000	387.225820	3735.246472	60.961633	12.519489	17.986290	542.28
2012-08-31	0.032258	3.064516	30.548387	16.580645	7.419355	2012.0	23.500000	508.129772	3775.707446	62.362406	12.930780	18.959409	461.66
2012-09-30	0.033333	3.133333	33.100000	16.500000	8.000000	2012.0	23.500000	488.393299	3869.488743	64.967847	10.982639	17.460833	483.24
2012-10-31	0.032258	2.838710	36.032258	17.354839	8.645161	2012.0	23.500000	358.515094	4046.332890	64.296216	9.354167	16.447043	528.25

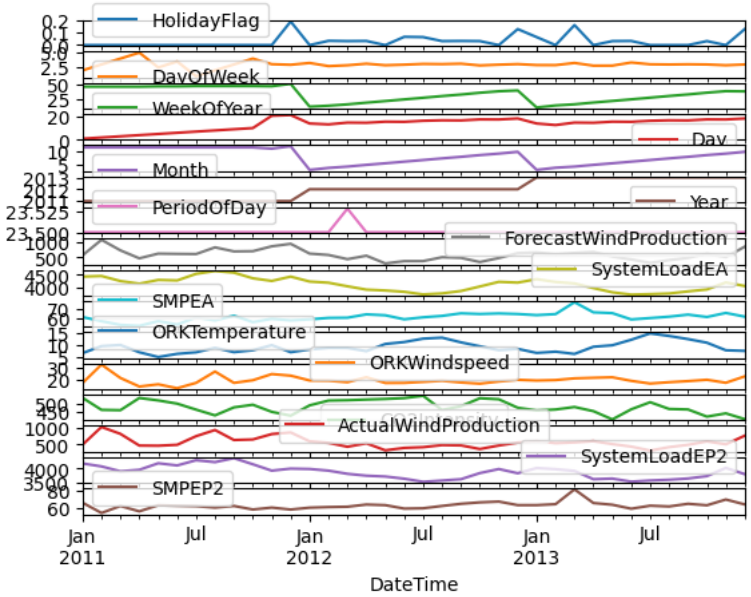
DateTime	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Int
2012-11-30	0.000000	2.966667	38.466667	17.300000	9.200000	2012.0	23.500000	483.443924	4227.898431	64.853924	7.747222	18.307153	522.37
2012-12-31	0.129032	3.064516	39.774194	18.129032	9.870968	2012.0	23.500000	636.899046	4198.231176	64.269603	8.289651	19.745699	474.50
2013-01-31	0.064516	2.903226	11.774194	13.870968	3.129032	2013.0	23.500000	657.181277	4332.598804	63.152151	6.613575	19.139852	459.27
2013-02-28	0.000000	2.892857	15.107143	12.571429	3.928571	2013.0	23.500000	594.745432	4223.081563	64.182232	7.135417	19.515402	464.62
2013-03-31	0.161290	3.258065	17.000000	14.645161	4.354839	2013.0	23.500000	640.569395	4165.431680	76.235067	6.194892	20.978427	478.74
2013-04-30	0.000000	2.800000	19.900000	14.500000	5.000000	2013.0	23.500000	672.551028	3982.307542	65.969576	9.164583	21.460903	455.19
2013-05-31	0.032258	2.806452	22.516129	15.419355	5.580645	2013.0	23.500000	531.812681	3818.493199	65.162923	9.782930	21.887500	408.54
2013-06-30	0.033333	3.333333	25.166667	15.300000	6.200000	2013.0	23.500000	438.145396	3726.758576	58.789444	12.061111	18.835208	465.72
2013-07-31	0.000000	3.032258	27.935484	16.193548	6.806452	2013.0	23.500000	330.536169	3746.369745	60.184698	14.692204	16.550403	505.90
2013-08-31	0.000000	3.000000	30.709677	16.580645	7.419355	2013.0	23.500000	417.096781	3771.842628	61.584362	13.627016	17.690726	467.59
2013-09-30	0.000000	3.033333	33.266667	16.500000	8.000000	2013.0	23.500000	488.688042	3853.882000	63.664826	12.334028	18.619375	464.82
2013-10-31	0.032258	3.000000	36.161290	17.354839	8.645161	2013.0	23.500000	615.541324	3929.673038	61.264684	10.837366	19.771505	424.23
2013-11-30	0.000000	2.866667	38.633333	17.300000	9.200000	2013.0	23.500000	513.089451	4204.551132	65.365604	7.708333	17.207292	442.72
2013-12-31	0.129032	3.000000	38.258065	18.129032	9.870968	2013.0	23.500000	839.975887	4064.858831	61.603918	7.449597	22.673925	409.93

In [ ]:
```
el_df.resample('M').mean().plot(subplots=True)
```

```
<ipython-input-15-052b9850bc35>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False.
Either specify numeric_only or select only columns which should be valid for the function.
```

Out [15]:
```
array([<Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>,
       <Axes: xlabel='DateTime'>, <Axes: xlabel='DateTime'>], dtype=object)
```



In [ ]:
```
final_df=el_df.resample('M').mean()
final_df
```

```
<ipython-input-16-262a0f12b9cd>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False.
Either specify numeric_only or select only columns which should be valid for the function.
```

Out [16]:

DateTime	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Int
2011-01-31	0.000000	2.000000	46.000000	1.000000	11.500000	2011.0	23.500000	567.916771	4433.788125	61.189167	6.520833	17.005208	530.58
2011-02-28	0.000000	3.000000	46.000000	2.000000	11.500000	2011.0	23.500000	1054.793229	4456.451979	57.025104	9.427083	32.725000	462.59
2011-03-31	0.000000	4.000000	46.000000	3.000000	11.500000	2011.0	23.500000	723.956667	4259.082917	53.261458	9.895833	21.303125	460.77
2011-04-30	0.000000	5.000000	46.000000	4.000000	11.500000	2011.0	23.500000	474.091979	4156.697708	52.314063	6.885417	13.996875	529.71
2011-05-31	0.000000	2.500000	46.500000	5.000000	11.500000	2011.0	23.500000	621.892292	4302.408125	57.051979	4.916667	15.882292	515.88
2011-06-30	0.000000	3.500000	46.500000	6.000000	11.500000	2011.0	23.500000	613.782917	4283.031042	53.654792	6.229167	12.566667	497.78
2011-07-31	0.000000	1.000000	47.000000	7.000000	11.500000	2011.0	23.500000	608.573958	4540.860104	60.758438	6.875000	17.119792	463.94
2011-08-31	0.000000	2.000000	47.000000	8.000000	11.500000	2011.0	23.500000	817.929271	4652.903854	57.753750	8.718750	26.794792	431.22
2011-09-30	0.000000	3.000000	47.000000	9.000000	11.500000	2011.0	23.500000	691.819792	4587.447917	62.579792	6.927083	17.117708	476.58
2011-10-31	0.000000	4.000000	47.000000	10.000000	11.500000	2011.0	23.500000	698.289688	4367.038229	56.049062	7.718750	19.370833	491.56
2011-11-30	0.000000	3.050000	46.700000	20.450000	11.050000	2011.0	23.500000	850.949271	4263.230042	59.390302	9.831250	24.663854	451.29
2011-12-31	0.190476	2.952381	50.666667	21.047619	11.952381	2011.0	23.500000	929.351746	4433.362411	57.791230	6.905754	23.289385	429.59
2012-01-31	0.000000	3.258065	13.161290	13.870968	3.129032	2012.0	23.500000	622.410491	4239.651028	58.881573	8.049059	19.163038	488.01
2012-02-29	0.034483	2.724138	14.689655	13.137931	3.862069	2012.0	23.500000	579.290014	4193.543807	60.356042	8.714799	19.009267	515.13
2012-03-31	0.032301	2.866756	16.909825	14.631225	4.356662	2012.0	23.528264	448.149764	4053.695128	60.550249	8.679677	17.698250	517.13
2012-04-30	0.033333	3.133333	19.700000	14.500000	5.000000	2012.0	23.500000	555.787521	3923.846694	63.923771	7.341667	21.949583	520.84
2012-05-31	0.000000	2.870968	22.354839	15.419355	5.580645	2012.0	23.500000	313.707782	3889.033226	62.993333	10.315860	17.036761	524.22
2012-06-30	0.066667	2.966667	25.066667	15.300000	6.200000	2012.0	23.500000	386.359576	3838.298840	58.928333	11.109722	17.109653	528.99
2012-07-31	0.064516	3.096774	27.774194	16.193548	6.806452	2012.0	23.500000	387.225820	3735.246472	60.961633	12.519489	17.986290	542.28
2012-08-31	0.032258	3.064516	30.548387	16.580645	7.419355	2012.0	23.500000	508.129772	3775.707446	62.362406	12.930780	18.959409	461.66
2012-09-30	0.033333	3.133333	33.100000	16.500000	8.000000	2012.0	23.500000	488.393299	3869.488743	64.967847	10.982639	17.460833	483.24
2012-10-31	0.032258	2.838710	36.032258	17.354839	8.645161	2012.0	23.500000	358.515094	4046.332890	64.296216	9.354167	16.447043	528.25
2012-11-30	0.000000	2.966667	38.466667	17.300000	9.200000	2012.0	23.500000	483.443924	4227.898431	64.853924	7.747222	18.307153	522.37
2012-12-31	0.129032	3.064516	39.774194	18.129032	9.870968	2012.0	23.500000	636.899046	4198.231176	64.269603	8.289651	19.745699	474.50
2013-01-31	0.064516	2.903226	11.774194	13.870968	3.129032	2013.0	23.500000	657.181277	4332.598804	63.152151	6.613575	19.139852	459.27
2013-02-28	0.000000	2.892857	15.107143	12.571429	3.928571	2013.0	23.500000	594.745432	4223.081563	64.182232	7.135417	19.515402	464.62
2013-03-31	0.161290	3.258065	17.000000	14.645161	4.354839	2013.0	23.500000	640.569395	4165.431680	76.235067	6.194892	20.978427	478.74
2013-04-30	0.000000	2.800000	19.900000	14.500000	5.000000	2013.0	23.500000	672.551028	3982.307542	65.969576	9.164583	21.460903	455.19
2013-05-31	0.032258	2.806452	22.516129	15.419355	5.580645	2013.0	23.500000	531.812681	3818.493199	65.162923	9.782930	21.887500	408.54
2013-06-30	0.033333	3.333333	25.166667	15.300000	6.200000	2013.0	23.500000	438.145396	3726.758576	58.789444	12.061111	18.835208	465.72
2013-07-31	0.000000	3.032258	27.935484	16.193548	6.806452	2013.0	23.500000	330.536169	3746.369745	60.184698	14.692204	16.550403	505.90
2013-08-31	0.000000	3.000000	30.709677	16.580645	7.419355	2013.0	23.500000	417.096781	3771.842628	61.584362	13.627016	17.690726	467.59
2013-09-30	0.000000	3.033333	33.266667	16.500000	8.000000	2013.0	23.500000	488.688042	3853.882000	63.664826	12.334028	18.619375	464.82
2013-10-31	0.032258	3.000000	36.161290	17.354839	8.645161	2013.0	23.500000	615.541324	3929.673038	61.264684	10.837366	19.771505	424.23
2013-11-30	0.000000	2.866667	38.633333	17.300000	9.200000	2013.0	23.500000	513.089451	4204.551132	65.365604	7.708333	17.207292	442.72
2013-12-31	0.129032	3.000000	38.258065	18.129032	9.870968	2013.0	23.500000	839.975887	4064.858831	61.603918	7.449597	22.673925	409.93

Implementing auto ARIMA and finding P,D,Q Values

In [ ]:

```
!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
[2K     [90m[0m [32m1.8/1.8 MB[0m [31m18.6 MB/s[0m eta [36m0:00:00[0m
[?25hRequirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.2)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.3)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.5)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.3
```

In [ ]:
```
import pmdarima as pm
```

In [ ]:
```
model = pm.auto_arima(final_df['SMPEP2'],
                      m=12, seasonal=True,
                     start_p=0, start_q=0, max_order=4, test='adf',error_action='ignore',
                         suppress_warnings=True,
                     stepwise=True, trace=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=155.224, Time=0.20 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=155.274, Time=0.03 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=152.621, Time=0.10 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.26 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=150.834, Time=0.04 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=152.615, Time=0.09 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=0.54 sec
 ARIMA(2,1,0)(0,1,0)[12]             : AIC=152.235, Time=0.07 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=inf, Time=0.45 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=inf, Time=0.05 sec
 ARIMA(2,1,1)(0,1,0)[12]             : AIC=inf, Time=0.29 sec
 ARIMA(1,1,0)(0,1,0)[12] intercept   : AIC=152.812, Time=0.06 sec

Best model:  ARIMA(1,1,0)(0,1,0)[12]
Total fit time: 2.243 seconds
```

Train and Test the Arima Model by Splitting the Time Series dataset

In [ ]:
```
train=final_df[(final_df.index.get_level_values(0) >= '2011-01-31') & (final_df.index.get_level_values(0) <=
```

In [ ]:
```
test=final_df[(final_df.index.get_level_values(0) > '2013-08-31')]
```

In [ ]:
```
test
```

Out [22]:

DateTime	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Inten
2013-09-30	0.000000	3.033333	33.266667	16.500000	8.000000	2013.0	23.5	488.688042	3853.882000	63.664826	12.334028	18.619375	464.8217
2013-10-31	0.032258	3.000000	36.161290	17.354839	8.645161	2013.0	23.5	615.541324	3929.673038	61.264684	10.837366	19.771505	424.2392
2013-11-30	0.000000	2.866667	38.633333	17.300000	9.200000	2013.0	23.5	513.089451	4204.551132	65.365604	7.708333	17.207292	442.7208
2013-12-31	0.129032	3.000000	38.258065	18.129032	9.870968	2013.0	23.5	839.975887	4064.858831	61.603918	7.449597	22.673925	409.9391

Fit the Target Data into Auto ARIMA model and Predict the Future Values

In [ ]:
```
model.fit(train['SMPEP2'])
```

Out [23]:

ARIMA
ARIMA(1,1,0)(0,1,0)[12]

In [ ]:
```
forecast=model.predict(n_periods=4, return_conf_int=True)
```

In [ ]:
```
forecast
```

Out [25]:
```
(2013-09-30    66.416461
 2013-10-31    67.124461
 2013-11-30    68.461771
 2013-12-31    64.248716
 Freq: M, dtype: float64,
 array([[54.02176249, 78.81115899],
        [53.33136586, 80.91755651],
        [51.82754598, 85.09599564],
        [45.94536605, 82.55206688]]))
```

In [ ]:
```
forecast_df = pd.DataFrame(forecast[0],index = test.index,columns=['Prediction'])
```

In [ ]:
```
forecast_df
```

Out [27]:

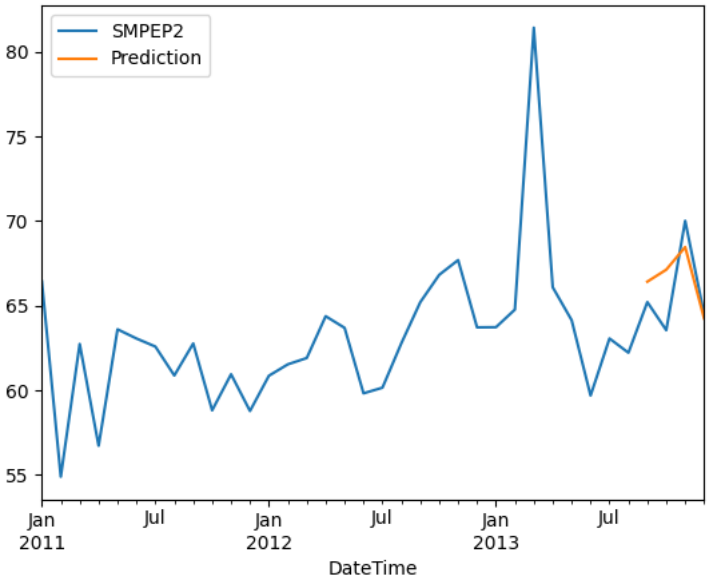
DateTime	Prediction
2013-09-30	66.416461
2013-10-31	67.124461
2013-11-30	68.461771
2013-12-31	64.248716

Using Matplotlib library, Plot the Predicted Target Data

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
pd.concat([final_df['SMPEP2'],forecast_df],axis=1).plot()
```

Out [29]: <Axes: xlabel='DateTime'>



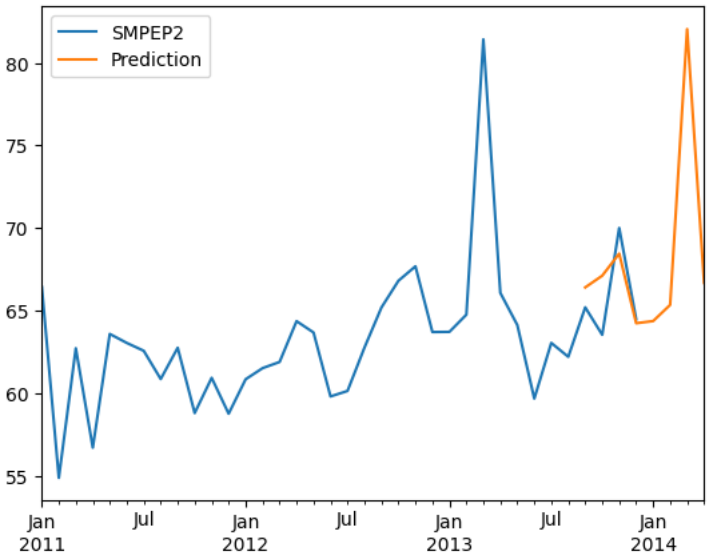
Plot the Predicted Target Data for the Future Unseen Values

In [ ]:
```
forecast1=model.predict(n_periods=8, return_conf_int=True)
forecast_range=pd.date_range(start='2013-09-30', periods=8,freq='M')
```

In [ ]:
```
forecast1_df = pd.DataFrame(forecast1[0],index =forecast_range,columns=['Prediction'])
```

In [ ]:
```
pd.concat([final_df['SMPEP2'],forecast1_df],axis=1).plot()
```

Out [36]: <Axes: >


SMPEP2
Prediction
80
75
70
65
60
55
Jan
2011
Jul
Jan
2012
Jul
Jan
2013
Jul
Jan
2014