In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

In [2]:
```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(15)
```

Out[2]:

	Sales
Order Date	
2014-01-06	2573.820
2014-01-07	76.728
2014-01-10	51.940
2014-01-11	9.940
2014-01-13	879.939
2014-01-14	61.960
2014-01-16	127.104
2014-01-19	181.470
2014-01-20	1413.510
2014-01-21	25.248
2014-01-26	217.200
2014-01-27	333.000
2014-01-31	290.666
2014-02-08	14.560
2014-02-11	1650.050

In [3]:

```
df = df.resample('MS').sum()

df.head(15)
```

Out[3]:

Order Date	Sales
2014-01-01	6242.5250
2014-02-01	1839.6580
2014-03-01	14573.9560
2014-04-01	7944.8370
2014-05-01	6912.7870
2014-06-01	13206.1256
2014-07-01	10821.0510
2014-08-01	7320.3465
2014-09-01	23816.4808
2014-10-01	12304.2470
2014-11-01	21564.8727
2014-12-01	30645.9665
2015-01-01	11739.9416
2015-02-01	3134.3740
2015-03-01	12499.7830

In [4]:

```
df.shape
```

Out[4]: (48, 1)

In [5]:

```
n = len(df)
m = int(n*0.8)

train_data = df.iloc[0:m]
```

```
test_data = df.iloc[m:n]

print(f"Total df size {len(df)}")
print(f"Total train data size {len(train_data)}")
print(f"Total test data size {len(test_data)}")
```

```
Total df size 48
Total train data size 38
Total test data size 10
```

In [6]:
```
train_data.tail()
```

Out[6]:

Order Date	Sales
2016-10-01	11872.5770
2016-11-01	31783.6288
2016-12-01	36678.7150
2017-01-01	5964.0320
2017-02-01	6866.3374

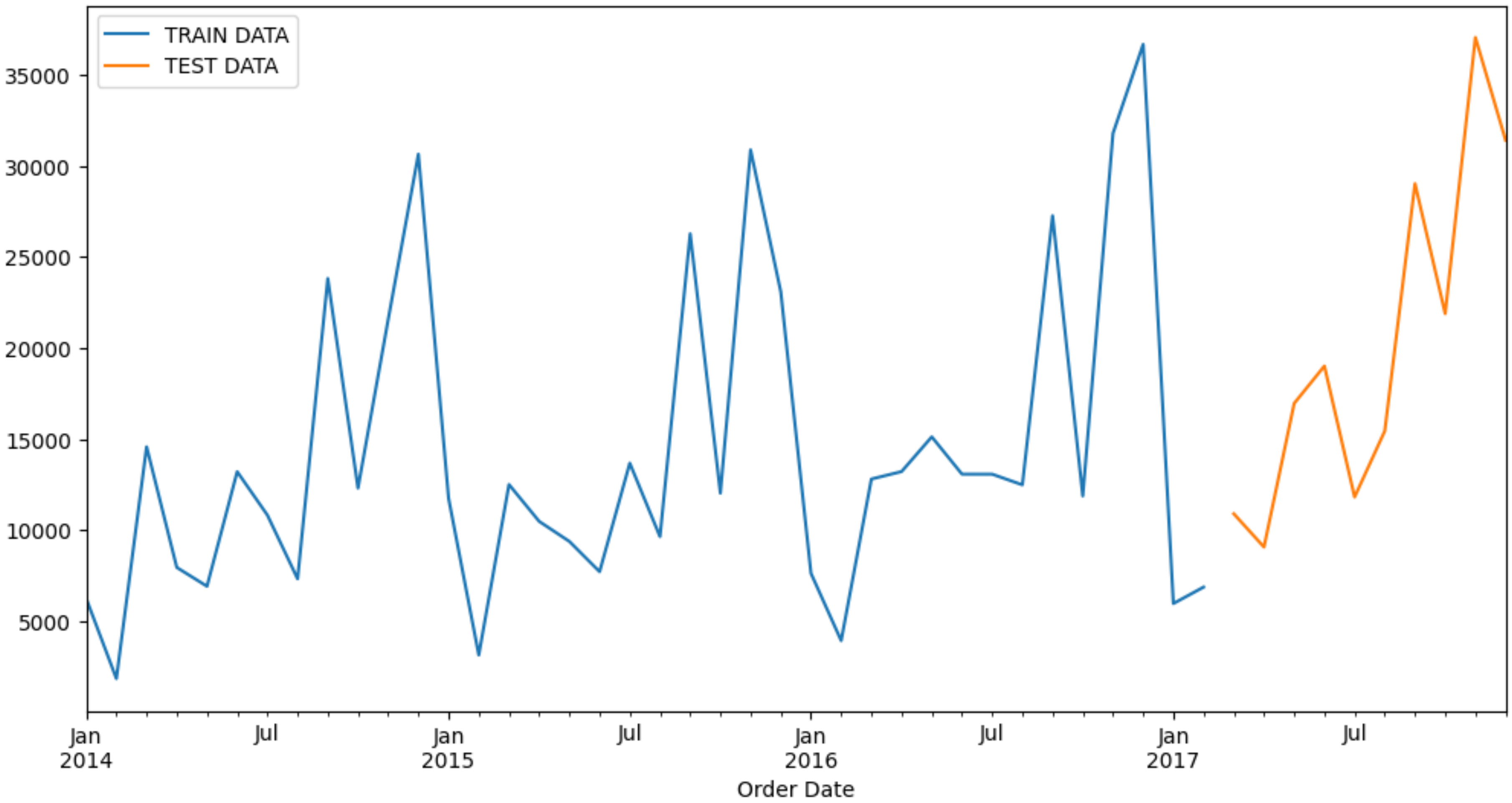
In [7]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

fitted_model = ExponentialSmoothing(train_data['Sales'], trend='mul', seasonal='mul', seasonal_periods=12).fit()
```

```
C:\Users\saswa\AppData\Roaming\Python\Python311\site-packages\statsmodels\tsa\holtwinters\model.py:83: RuntimeWarning: overflow
encountered in matmul
  return err.T @ err
C:\Users\saswa\AppData\Roaming\Python\Python311\site-packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning: Opt
imization failed to converge. Check mle_retvals.
  warnings.warn(
```

In [8]:
```
test_predictions = fitted_model.forecast(len(test_data))
test_predictions
```

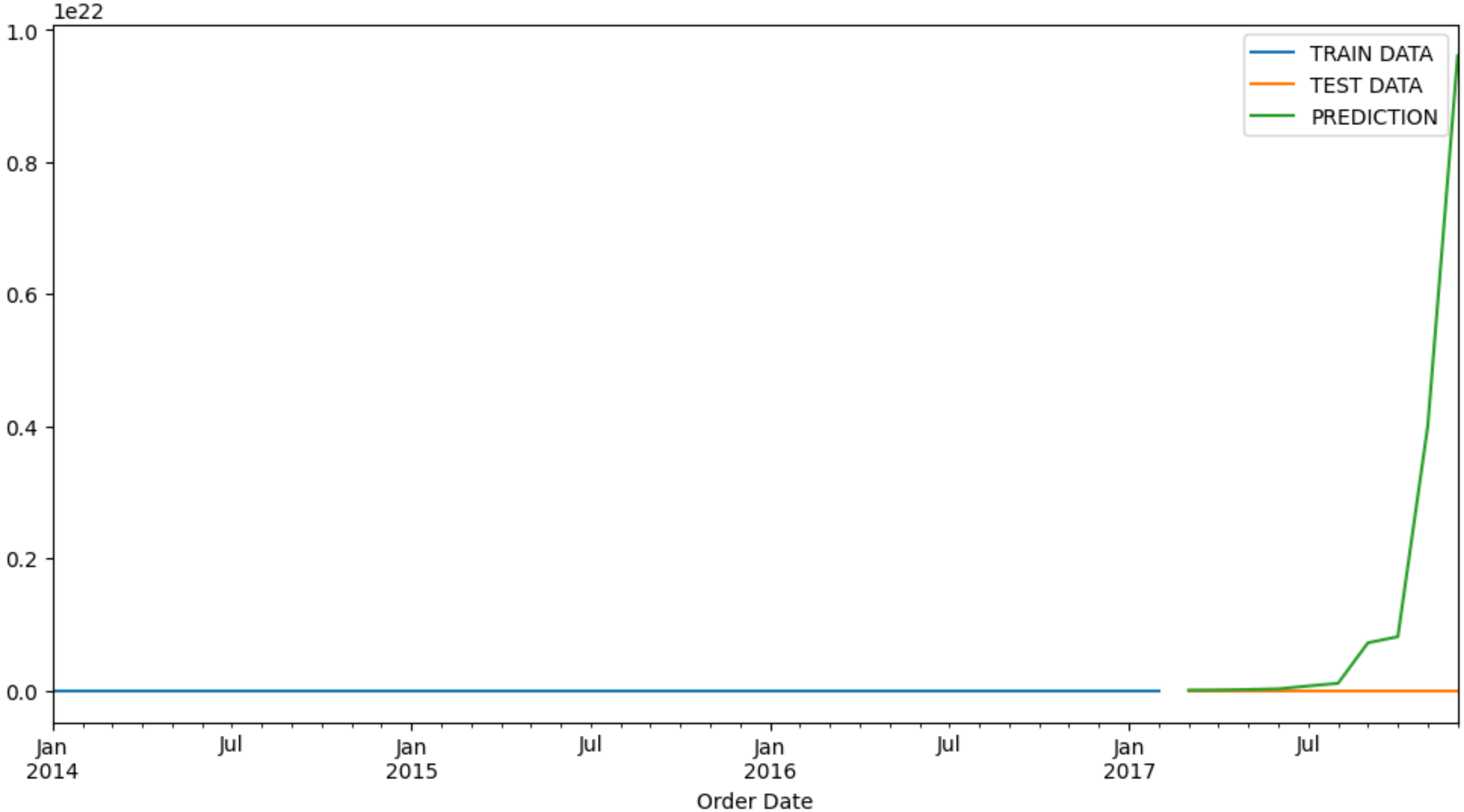
Out[8]:
```
2017-03-01    2.139283e+18
2017-04-01    4.589978e+18
2017-05-01    1.088440e+19
2017-06-01    2.139505e+19
2017-07-01    6.488781e+19
2017-08-01    1.055809e+20
2017-09-01    7.192510e+20
2017-10-01    8.114941e+20
2017-11-01    4.000376e+21
2017-12-01    9.608036e+21
Freq: MS, dtype: float64
```

In [9]:
```
train_data['Sales'].plot(legend=True, label='TRAIN DATA')
test_data['Sales'].plot(legend=True, label='TEST DATA', figsize=(12, 6)).autoscale(axis='x', tight=True)
```



In [10]:
```
train_data['Sales'].plot(legend=True, label='TRAIN DATA')
test_data['Sales'].plot(legend=True, label='TEST DATA', figsize=(12, 6)).autoscale(axis='x', tight=True)
test_predictions.plot(legend=True, label='PREDICTION')
```

Out[10]:
```
<Axes: xlabel='Order Date'>
```



In [11]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
mae_error = mean_absolute_error(test_data, test_predictions)

print(f"Mean absolute error of the above model is {mae_error}")
```

```
Mean absolute error of the above model is 1.5348634368862705e+21
```

In [12]:

```
mse_error = mean_squared_error(test_data, test_predictions)
```

```
print(f"Mean squared error of the above model is {mse_error}")
```

```
Mean squared error of the above model is 1.0950916485965921e+43
```

In [13]:
```
rmse_error = np.sqrt(mean_squared_error(test_data, test_predictions))

print(f"Root mean squared error of the above model is {rmse_error}")
```

```
Root mean squared error of the above model is 3.309216899202275e+21
```

In [14]:
```
test_data.describe()
```

Out[14]:

	Sales
count	10.000000
mean	20255.689980
std	9463.329001
min	9065.958100
25%	12720.235000
50%	17983.072450
75%	27242.171550
max	37056.715000

In [15]:
```
fitted_model = ExponentialSmoothing(df['Sales'], trend='mul', seasonal='mul', seasonal_periods=12).fit()
```

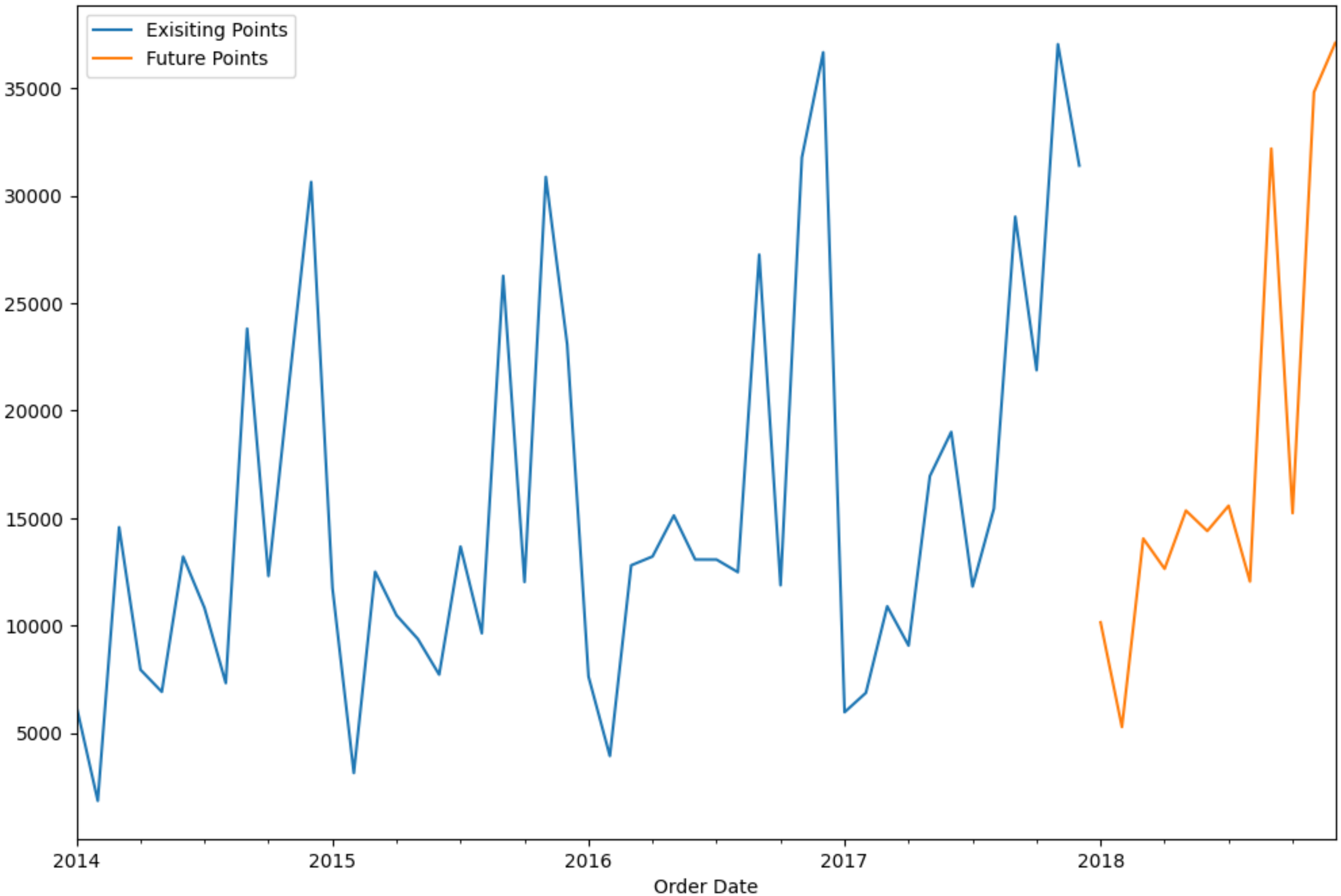
```
C:\Users\saswa\AppData\Roaming\Python\Python311\site-packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning: Opt
imization failed to converge. Check mle_retvals.
  warnings.warn(
```

In [16]:
```
future_preds = fitted_model.forecast(12)
future_preds
```

Out[16]:
```
2018-01-01    10145.196758
2018-02-01     5276.646045
2018-03-01    14046.143518
2018-04-01    12638.938600
2018-05-01    15347.806822
2018-06-01    14398.498944
2018-07-01    15575.177149
2018-08-01    12050.471323
2018-09-01    32198.645769
2018-10-01    15228.732215
2018-11-01    34823.955051
2018-12-01    37118.533234
Freq: MS, dtype: float64
```

In [17]:
```
df['Sales'].plot(figsize=(12, 8), legend=True, label='Exisiting Points')
future_preds.plot(figsize=(12, 8), legend=True, label='Future Points')
```

Out[17]:
```
<Axes: xlabel='Order Date'>
```



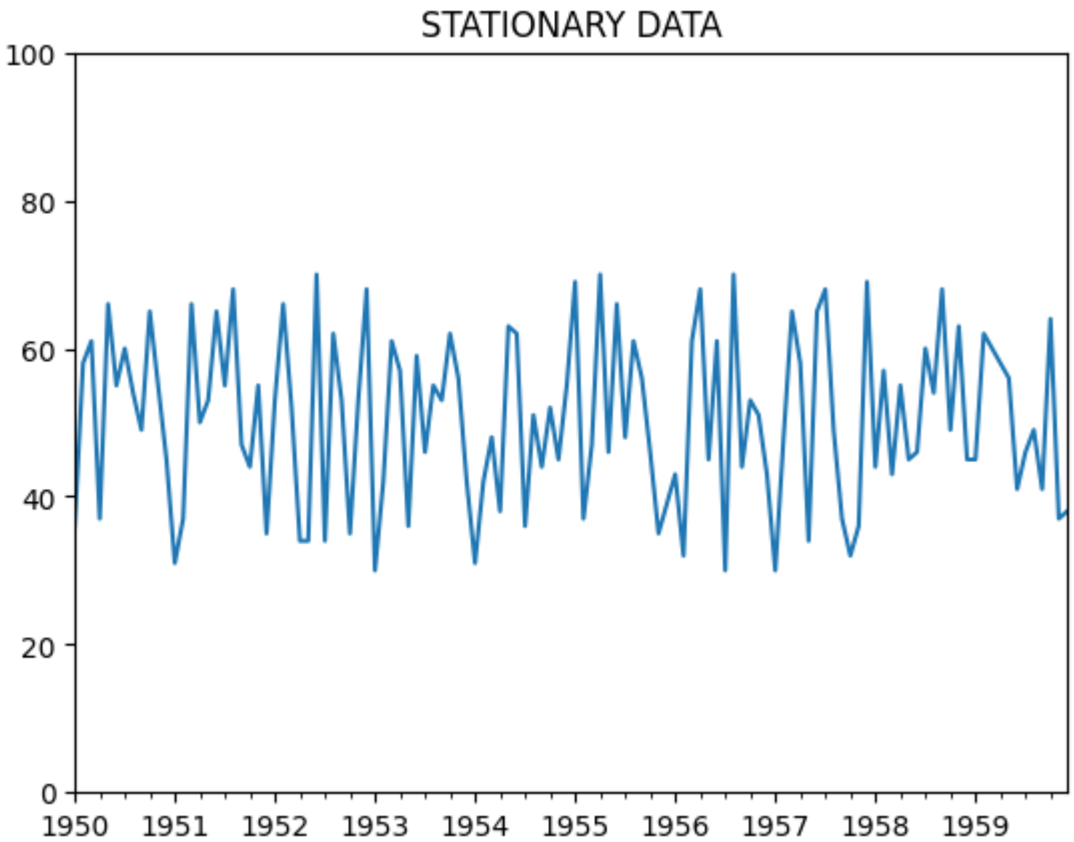
In [18]:
```
df = pd.read_csv('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Pinaki-samples.csv', index_col=0, parse
df.head()
```

Out[18]:

	a	b	c	d
1950-01-01	36	27	0	67
1950-02-01	58	22	3	31
1950-03-01	61	17	5	67
1950-04-01	37	15	8	47
1950-05-01	66	13	8	62

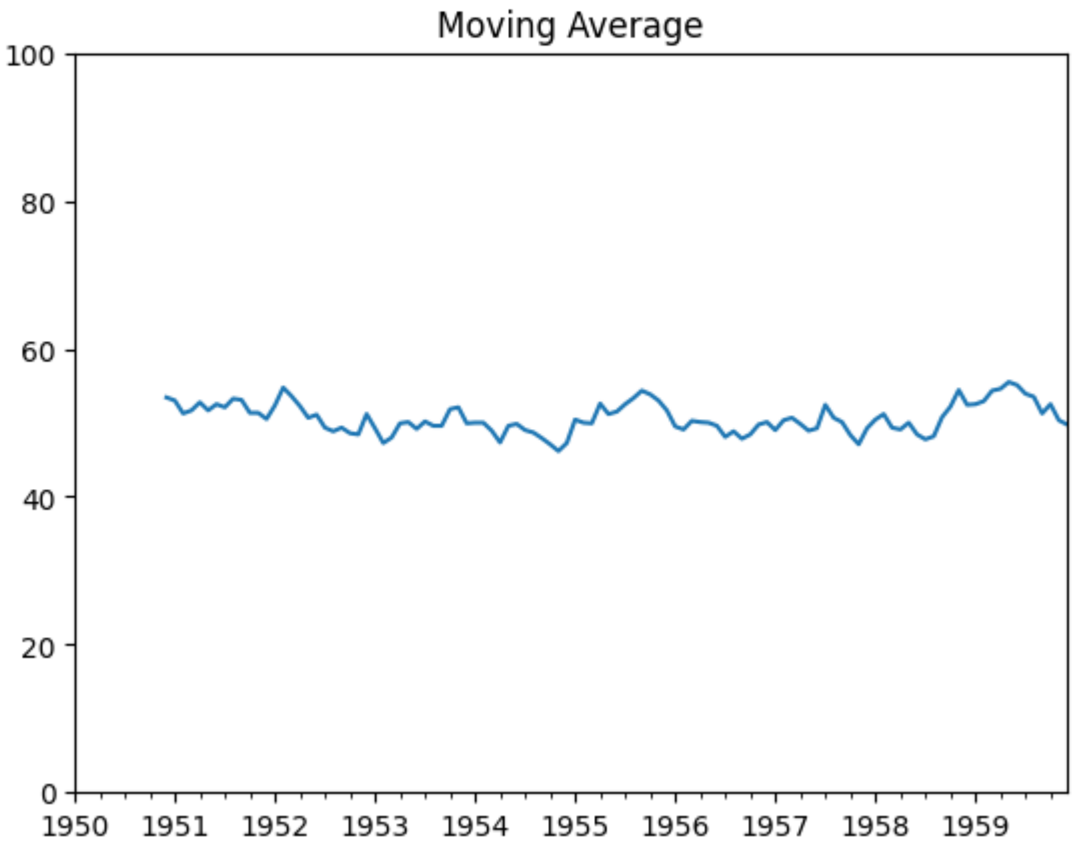
In [19]:

```
df['a'].plot(ylim=[0, 100], title='STATIONARY DATA').autoscale(axis='x', tight=True)
```

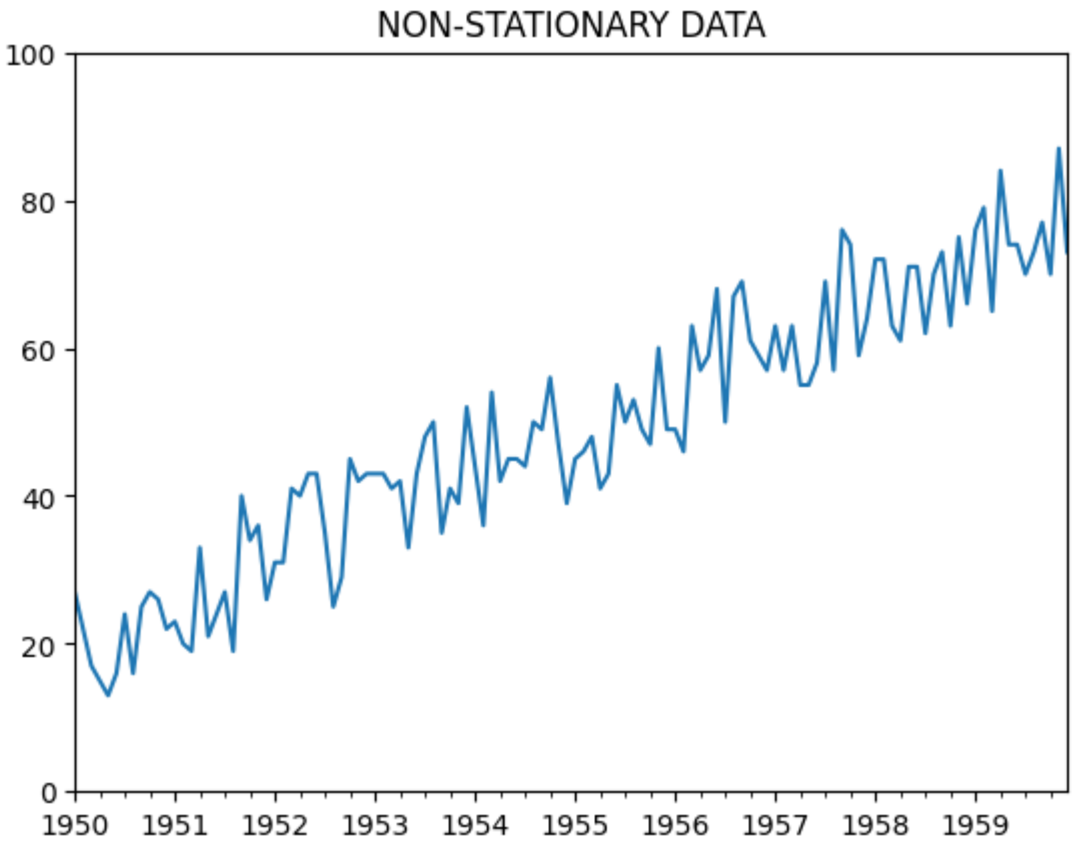


In [20]:
```
df['a_ma'] = df['a'].rolling(12).mean()

df['a_ma'].plot(ylim=[0, 100], title='Moving Average')
```

Out[20]: <Axes: title={'center': 'Moving Average'}>

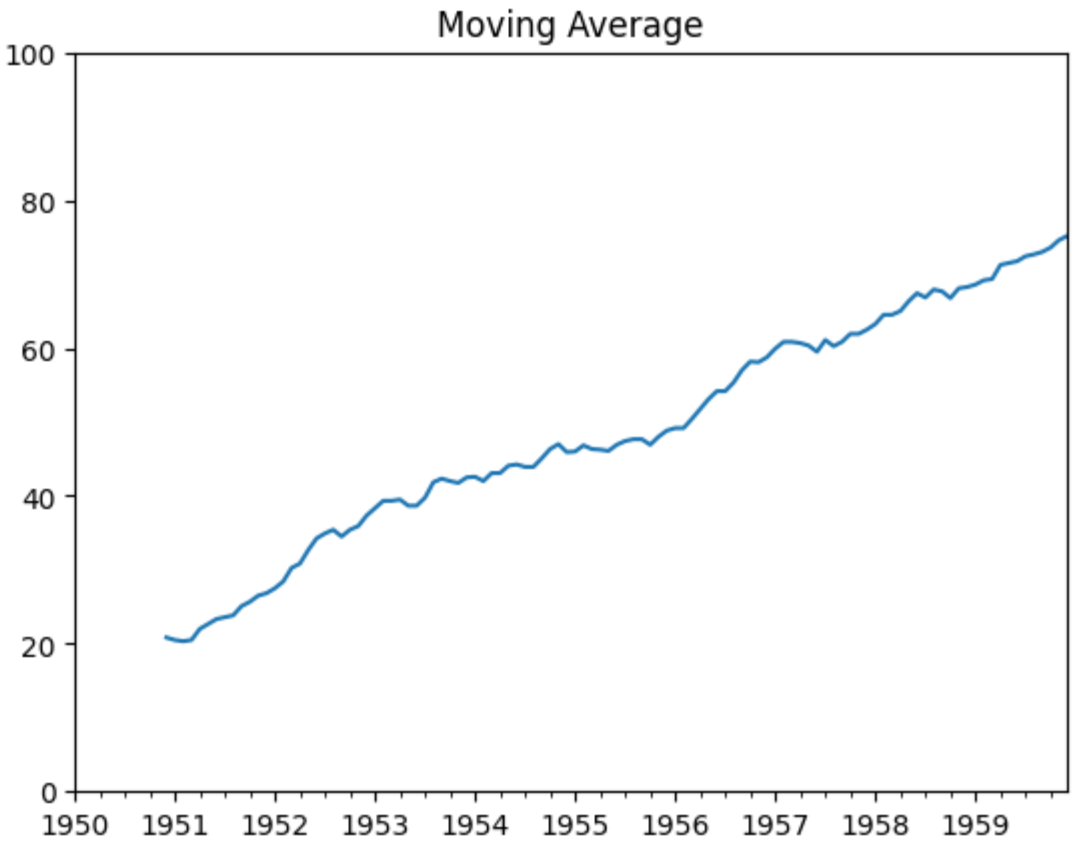


In [21]:
```
df['b'].plot(ylim=[0,100],title="NON-STATIONARY DATA").autoscale(axis='x',tight=True)
```

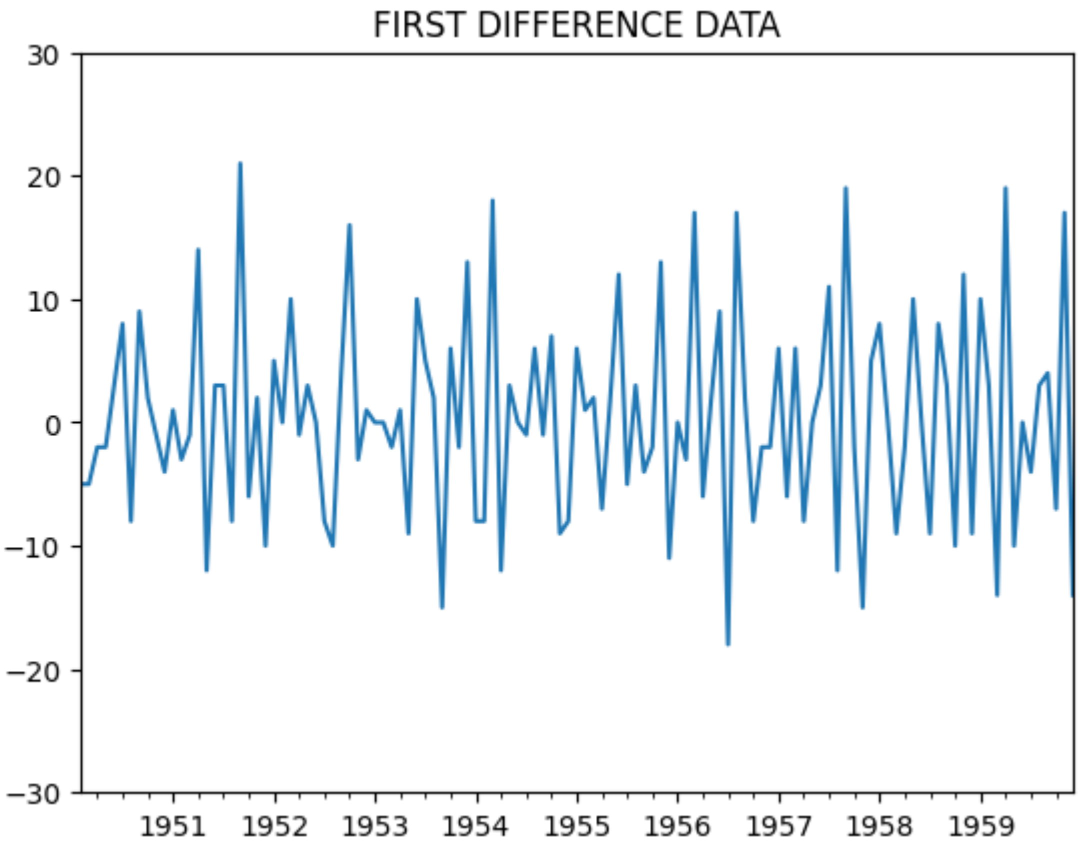


In [22]:
```
df['b_ma'] = df['b'].rolling(12).mean()
df['b_ma'].plot(ylim=[0, 100], title='Moving Average')
```

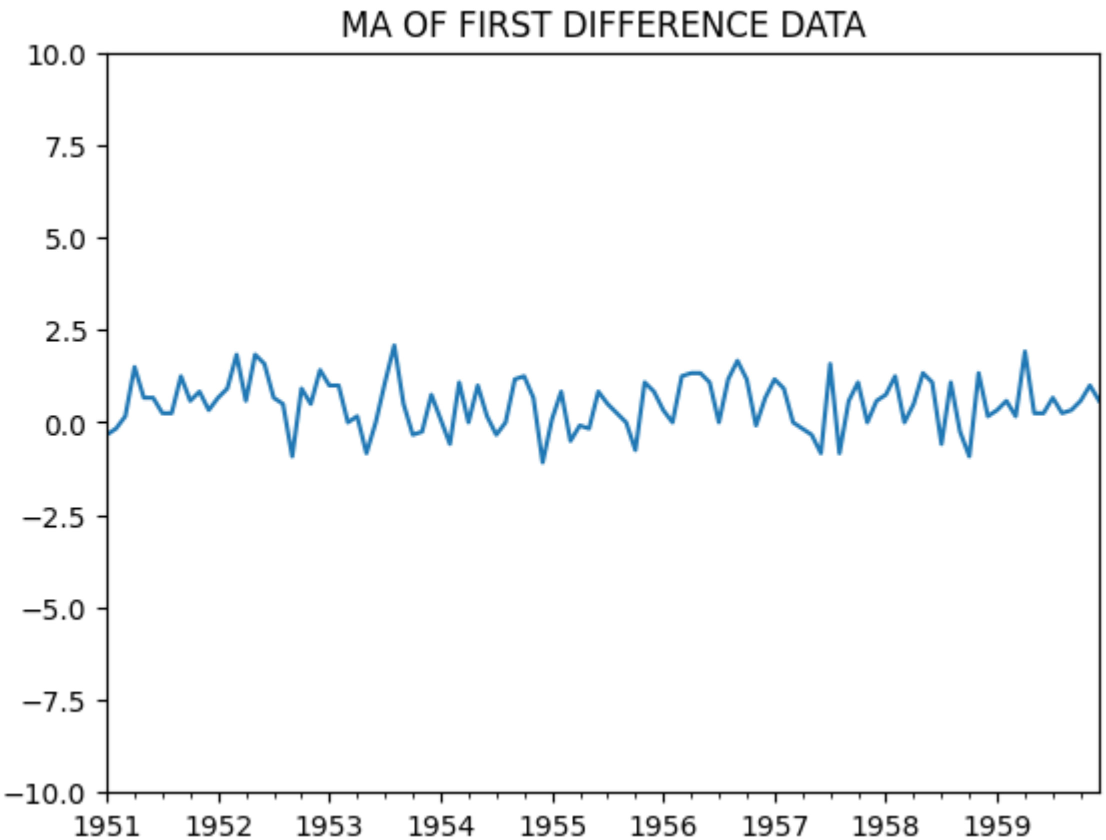
Out[22]:
```
<Axes: title={'center': 'Moving Average'}>
```



In [23]:
```
from statsmodels.tsa.statespace.tools import diff

df['d1'] = diff(df['b'],k_diff=1)

df['d1'].plot(title="FIRST DIFFERENCE DATA", ylim=[-30, 30]).autoscale(axis='x',tight=True)
```



In [24]:
```
df['d1_ma'] = df['d1'].rolling(12).mean()

df['d1_ma'].plot(title="MA OF FIRST DIFFERENCE DATA", ylim=[-10, 10]).autoscale(axis='x',tight=True)
```

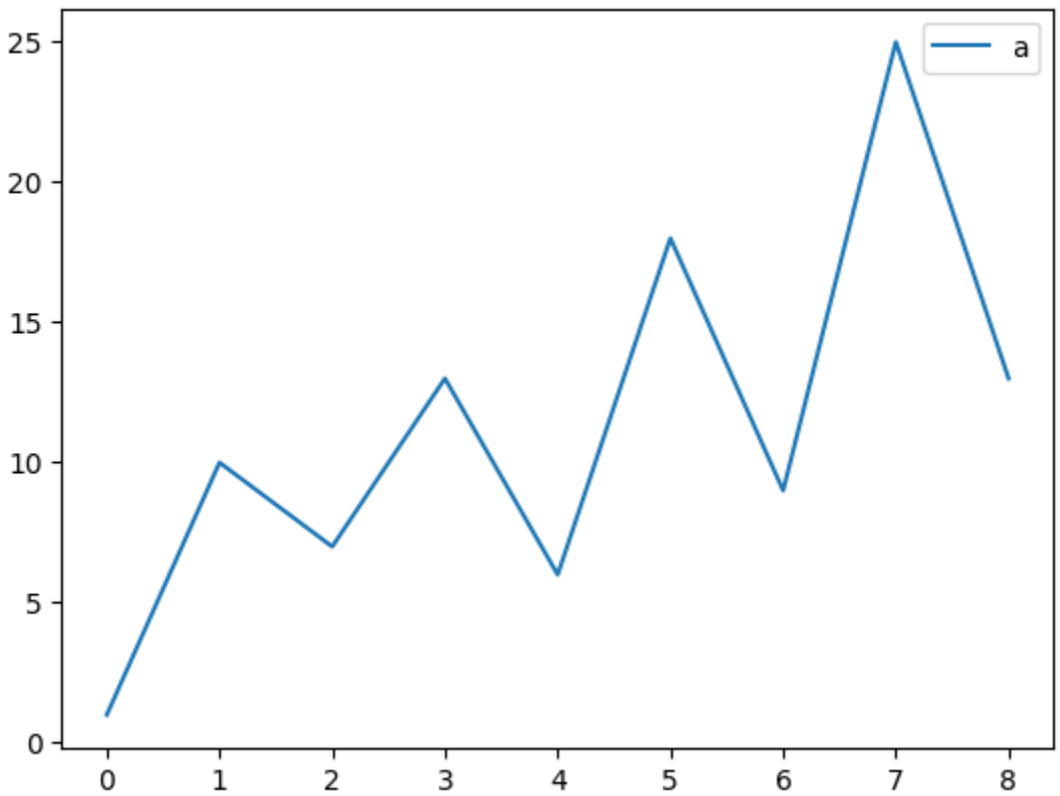


In [25]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import statsmodels.api as sm
```

In [26]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [27]:
```
df = pd.DataFrame({'a':[1, 10, 7, 13, 6, 18, 9, 25, 13]})

df.plot()
```

Out[27]: <Axes: >



In [28]:
```
from statsmodels.tsa.stattools import acovf, acf
arr = acovf(df['a'])

arr
```

Out[28]:
```
array([ 44.22222222,  -6.30864198,  19.60493827, -11.62962963,
         7.80246914, -13.58024691,  -0.14814815, -15.9382716 ,
        -1.91358025])
```

In [29]:
```
arr1 = acovf(df['a'])

arr1
```

Out[29]:
```
array([ 44.22222222,  -6.30864198,  19.60493827, -11.62962963,
         7.80246914, -13.58024691,  -0.14814815, -15.9382716 ,
        -1.91358025])
```

In [30]:
```
arr3 = acf(df['a'])

arr3
```

Out[30]:
```
array([ 1.        , -0.14265773,  0.44332775, -0.26298157,  0.17643774,
       -0.30709101, -0.00335008, -0.36041318, -0.04327192])
```

In [31]:
```
from statsmodels.tsa.stattools import pacf_yw
arr4 = pacf_yw(df['a'],nlags=4,method='mle')
arr4
```

Out[31]:
```
array([ 1.        , -0.14265773,  0.43176344, -0.20758442, -0.04572862])
```

In [32]:
```
arr4 = pacf_yw(df['a'],nlags=4,method='adjusted')
arr4
```

Out[32]:
```
array([ 1.        , -0.16048995,  0.5586243 , -0.39456104,  0.01906252])
```

In [33]:
```
from statsmodels.tsa.stattools import pacf_ols
arr5 = pacf_ols(df['a'],nlags=4)
arr5
```

Out[33]:
```
array([ 1.        , -0.13833492,  1.13495418, -0.04476691,  0.5979815 ])
```

In [34]:
```
from pandas.plotting import lag_plot
```

In [36]:
```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df['Category'].value_counts()
```

Out[36]:
```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

In [37]:
```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Office Supplies']
```

```
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(4)
```

Out[37]:

	Sales
Order Date	
2014-01-03	16.448
2014-01-04	288.060
2014-01-05	19.536
2014-01-06	685.340

In [38]:

```
df = df.resample('W').sum()

df.head()
```

Out[38]:

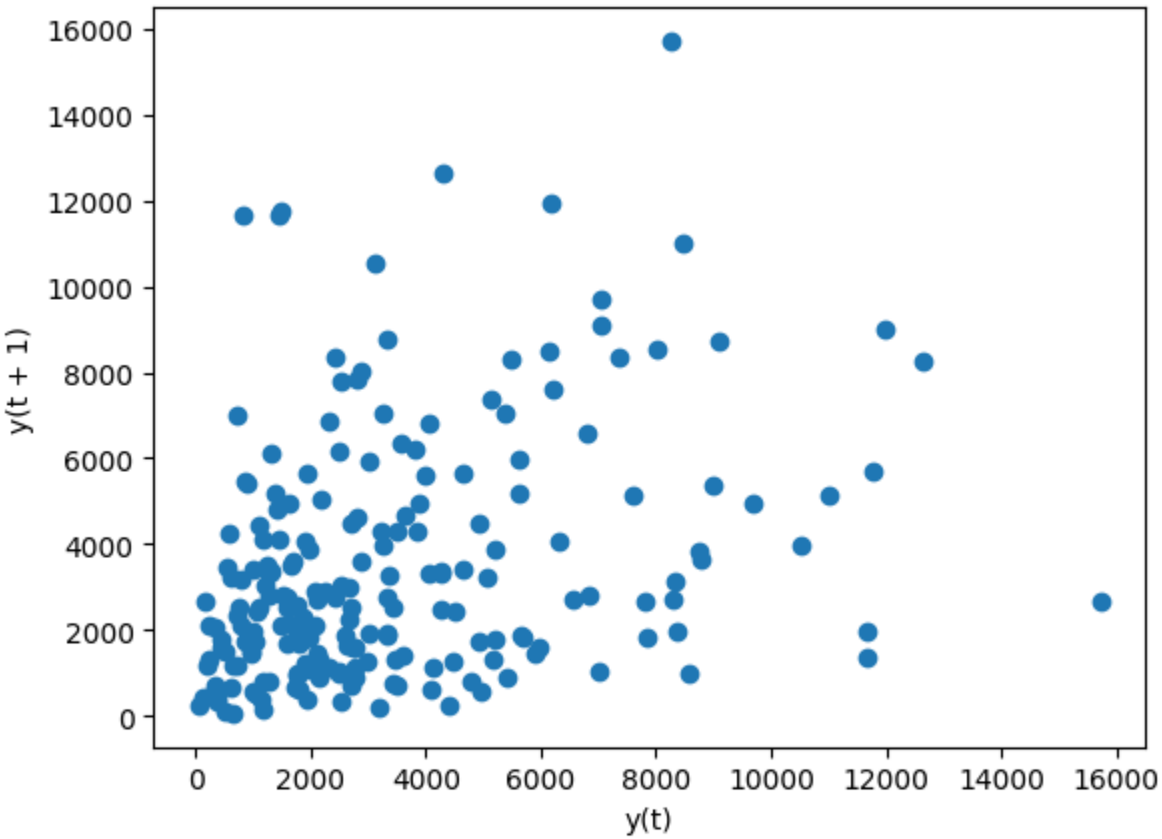
	Sales
Order Date	
2014-01-05	324.044
2014-01-12	708.004
2014-01-19	2337.764
2014-01-26	1143.170
2014-02-02	368.784

In [39]:

```
lag_plot(df['Sales'])
```

Out[39]:

```
<Axes: xlabel='y(t)', ylabel='y(t + 1)'>
```

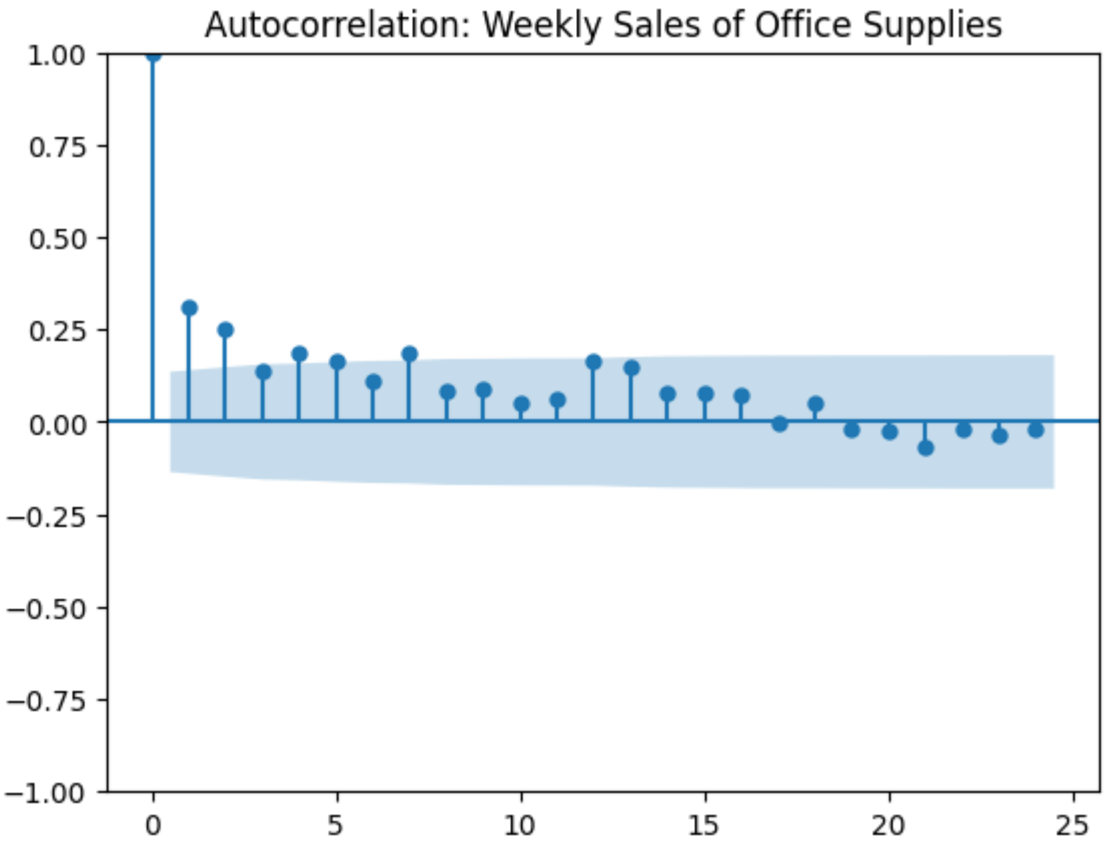


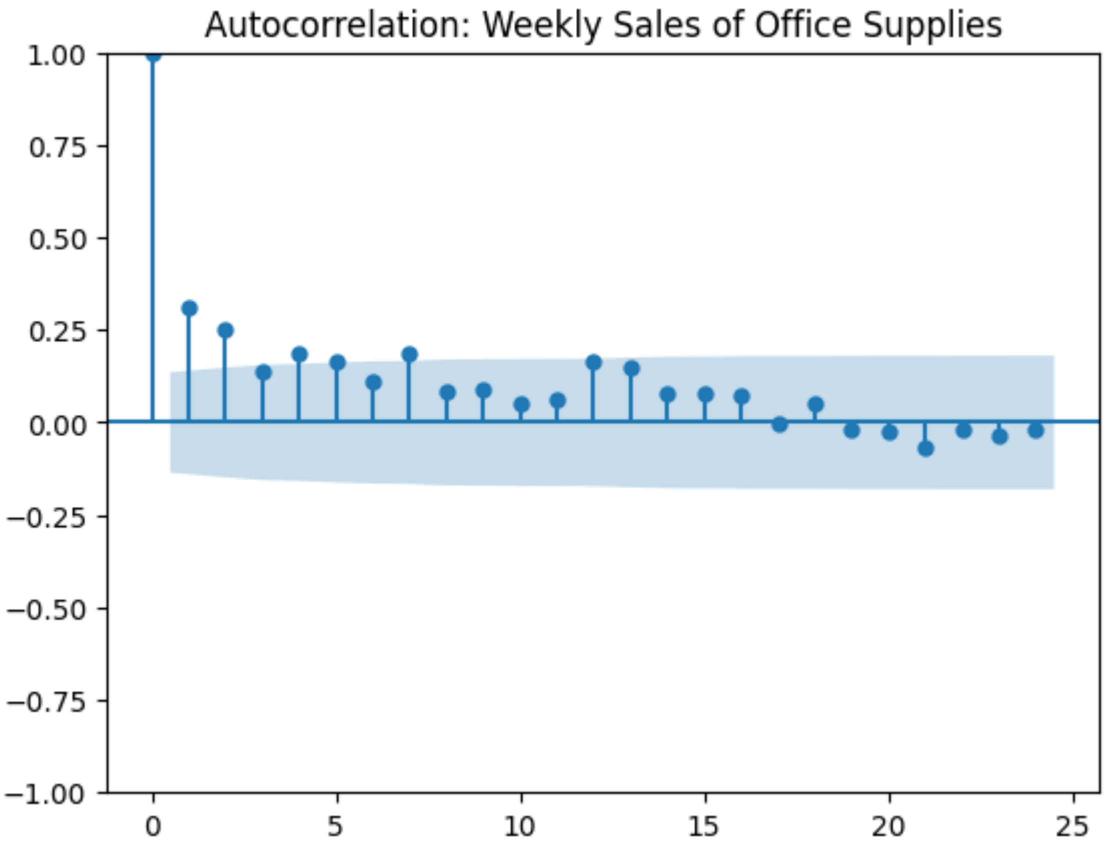
In [40]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf(df['Sales'])
```

Out[40]:
```
array([ 1.00000000e+00,  3.12440082e-01,  2.49173025e-01,  1.39336867e-01,
        1.84489786e-01,  1.67531418e-01,  1.11379734e-01,  1.84930768e-01,
        8.24198806e-02,  8.74684994e-02,  4.89924777e-02,  6.22098365e-02,
        1.65448013e-01,  1.49072360e-01,  7.97573259e-02,  7.65911202e-02,
        7.55594141e-02, -7.46386648e-04,  4.97054222e-02, -2.10402529e-02,
       -2.49880823e-02, -6.68093251e-02, -1.79718855e-02, -3.42941758e-02])
```

In [41]:
```
title = 'Autocorrelation: Weekly Sales of Office Supplies'
lags = 40
plot_acf(df['Sales'], title=title)
```

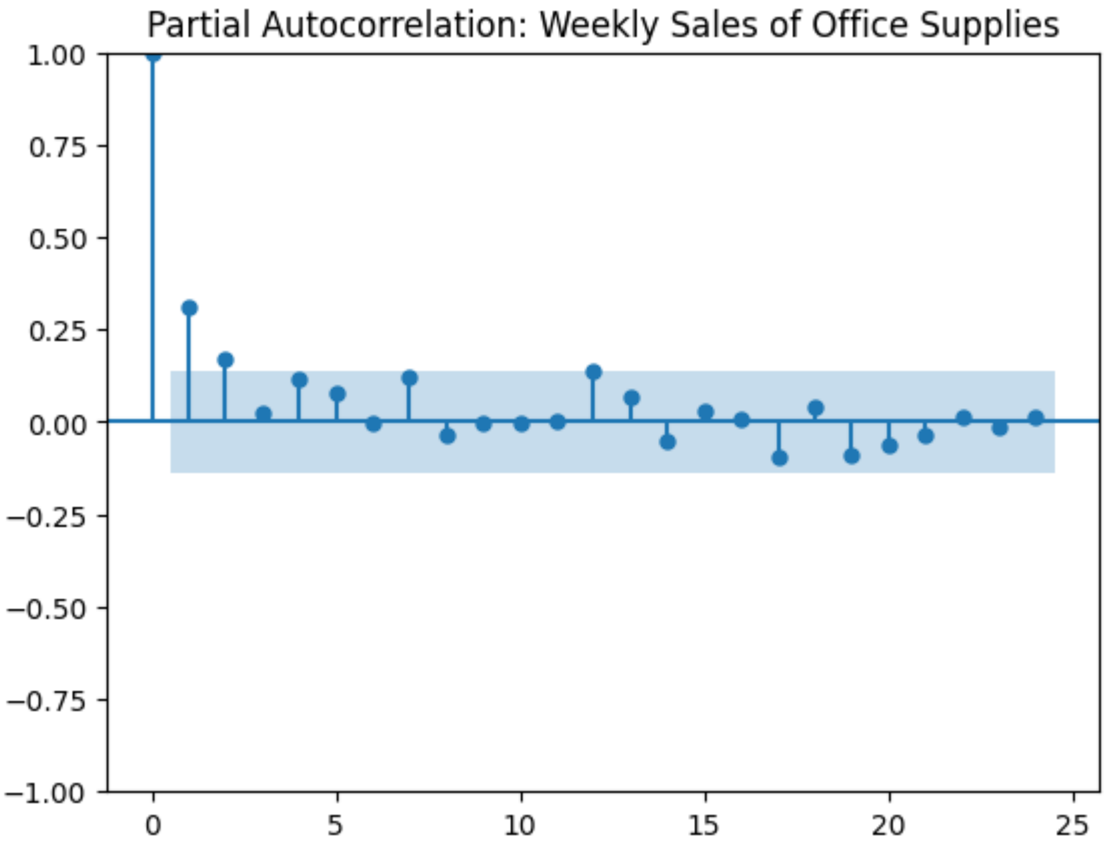
Out[41]:

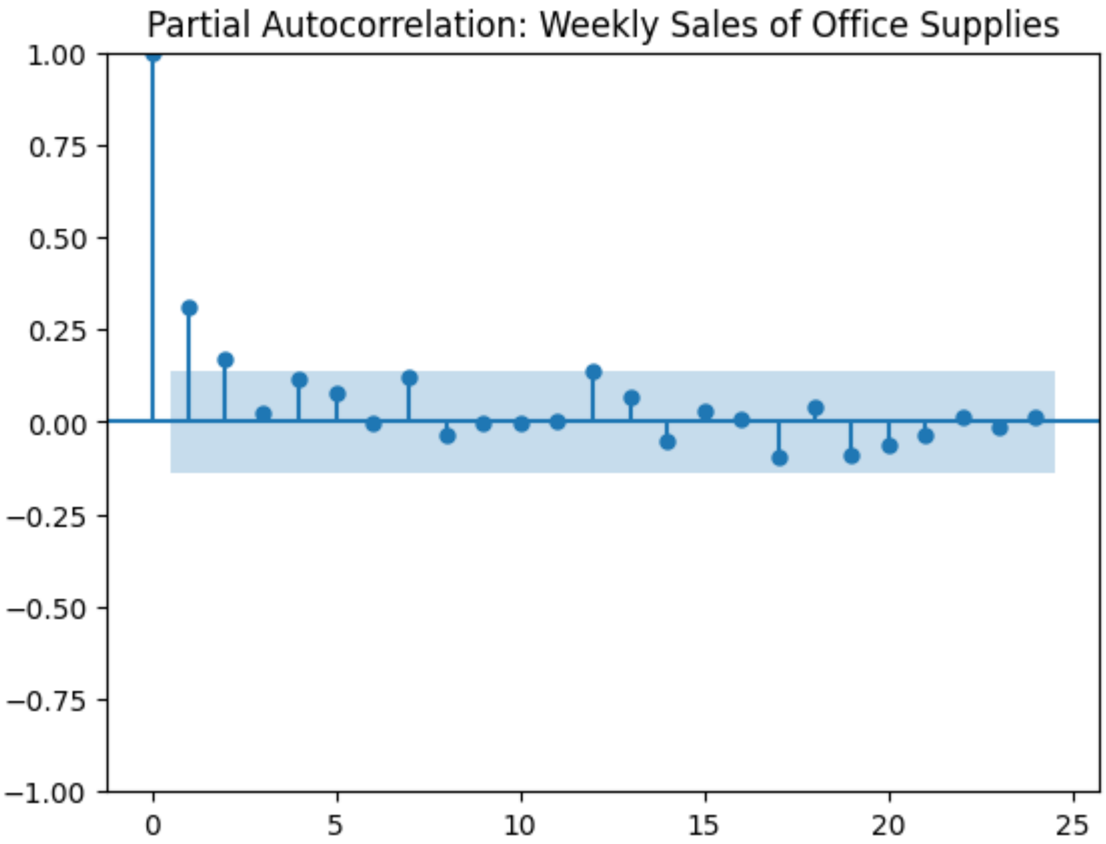

Autocorrelation: Weekly Sales of Office Supplies
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25



In [42]:
```
title = 'Partial Autocorrelation: Weekly Sales of Office Supplies'
lags = 40
plot_pacf(df['Sales'], title=title)
```

Out[42]:


Partial Autocorrelation: Weekly Sales of Office Supplies
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25



In [43]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

In [44]:
```
from statsmodels.tsa.ar_model import AR,ARResults

#plot the sales data of Office Supplies
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(4)
```

Out[44]:

	Sales
Order Date	
2014-01-06	2573.820
2014-01-07	76.728
2014-01-10	51.940
2014-01-11	9.940

In [45]:

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

df.head()
```

Out[45]:

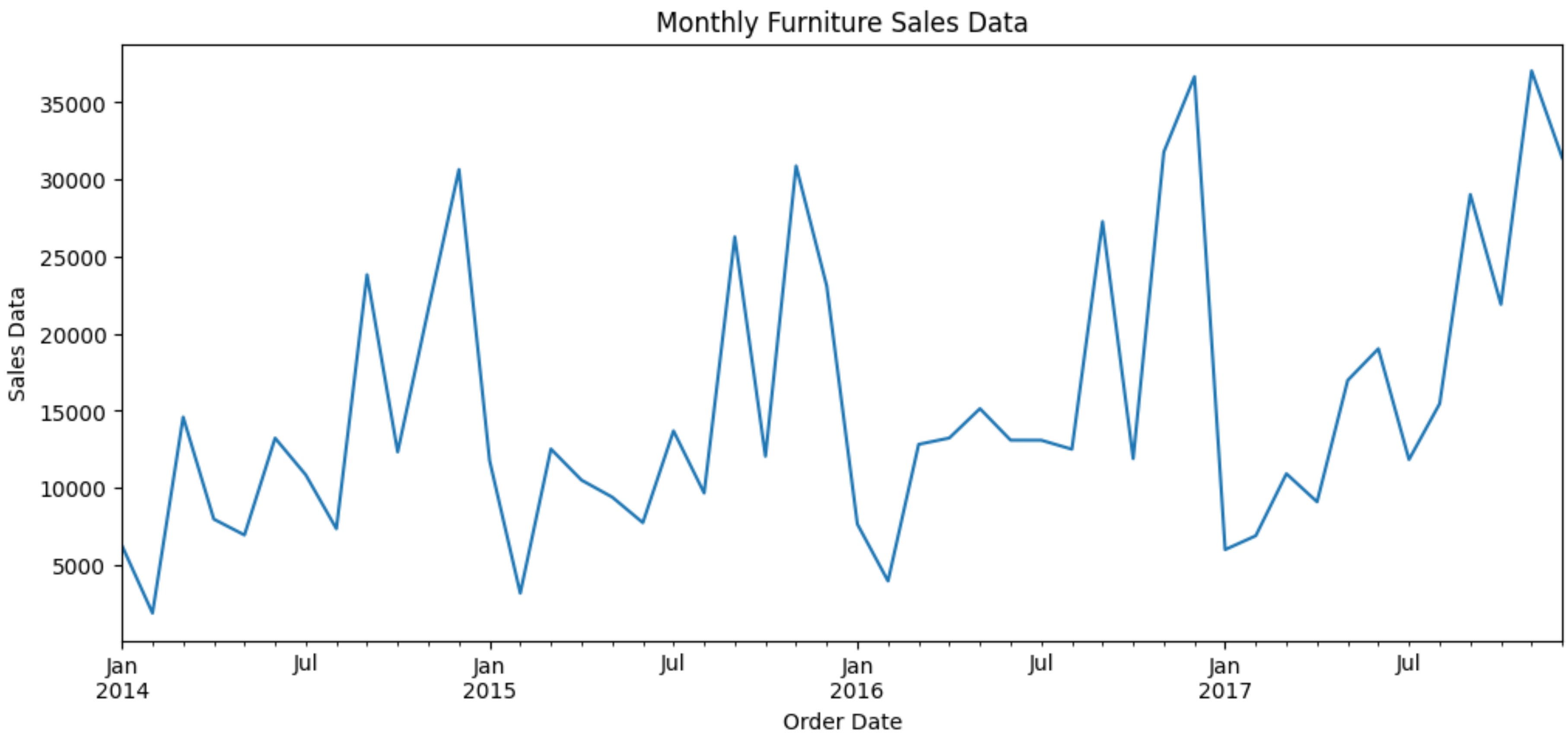
	Sales
Order Date	
2014-01-31	6242.525
2014-02-28	1839.658
2014-03-31	14573.956
2014-04-30	7944.837
2014-05-31	6912.787

In [46]:

```
title='Monthly Furniture Sales Data'
ylabel='Sales Data'
xlabel='Order Date'

ax = df['Sales'].plot(figsize=(12,5),title=title)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[46]:

```
[Text(0.5, 0, 'Order Date'), Text(0, 0.5, 'Sales Data')]
```



In [47]:
```
len(df)
```

Out[47]: 48

In [48]:
```
train = df.iloc[:len(df)-6]
test = df.iloc[len(df)-6:]

print(f"Train size is {len(train)}")
print(f"Test size is {len(test)}")
```

```
Train size is 42
Test size is 6
```

In [49]:
```
import warnings
```

```
warnings.filterwarnings("ignore")

from statsmodels.tsa.ar_model import AutoReg
```

In [50]:
```
mod1 = AutoReg(train['Sales'], 1, old_names=False)
res1 = mod1.fit()
print(res1.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Sales   No. Observations:                   42
Model:                     AutoReg(1)   Log Likelihood                -427.247
Method:               Conditional MLE   S.D. of innovations           8116.944
Date:                Thu, 21 Sep 2023   AIC                            860.493
Time:                        00:07:29   BIC                            865.634
Sample:                    02-28-2014   HQIC                           862.365
                         - 06-30-2017
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.172e+04   2487.380      4.712      0.000    6845.693    1.66e+04
Sales.L1       0.1884      0.152      1.237      0.216      -0.110       0.487
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            5.3086           +0.0000j            5.3086            0.0000
-----------------------------------------------------------------------------
```

In [51]:
```
print(f'Lag: {res1.arfreq}')
print(f'Coefficients:\n{res1.params}')
```

```
Lag: [0.]
Coefficients:
const       11720.868755
Sales.L1        0.188372
dtype: float64
```

In [52]:
```
start=len(train)
end=len(train)+len(test)-1
predictions1 = res1.predict(start=start, end=end, dynamic=False).rename('AR(1) Predictions')
```

In [53]:
```
predictions1
```

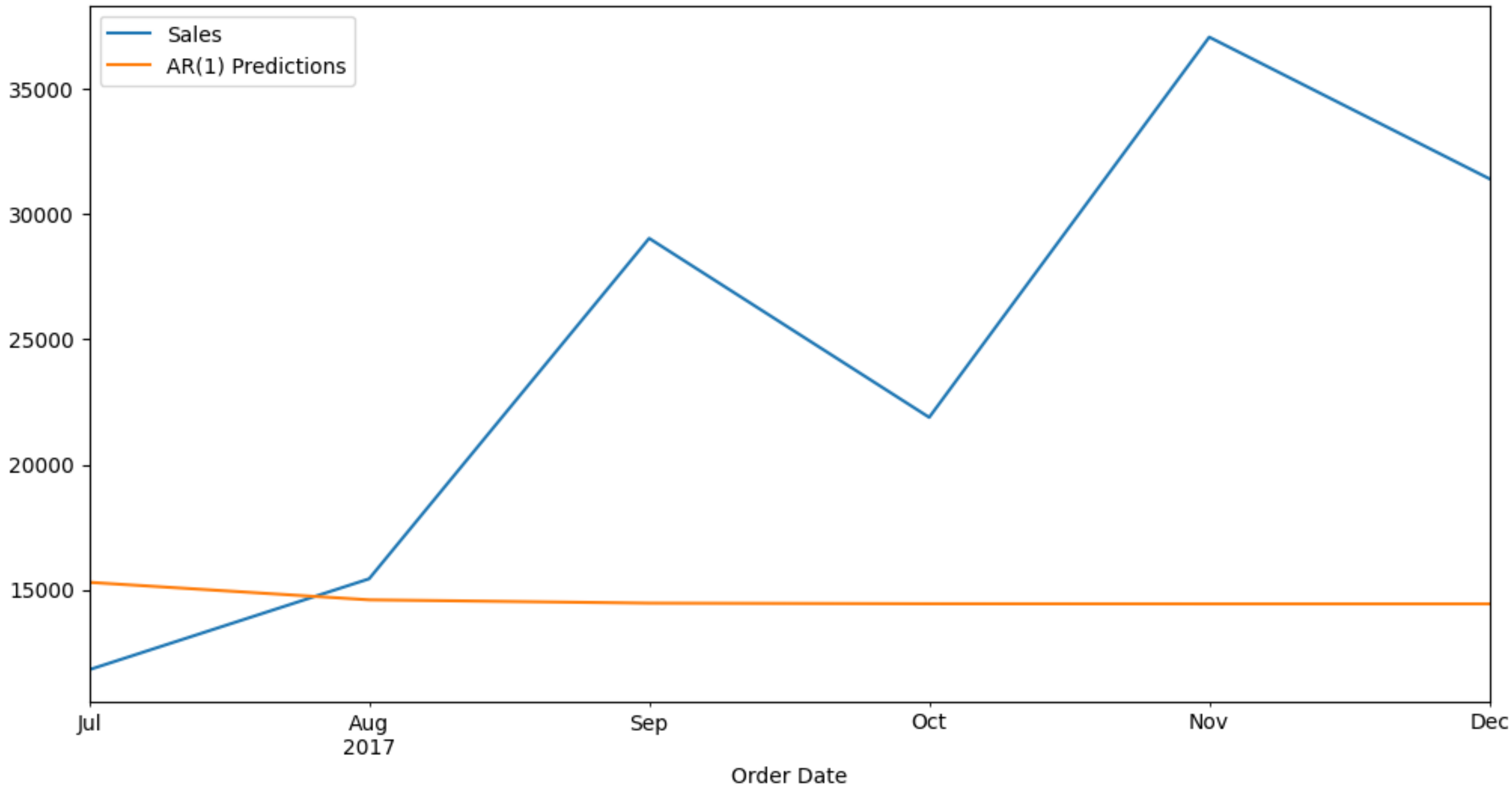
Out[53]:
```
2017-07-31    15301.562179
2017-08-31    14603.261009
2017-09-30    14471.720330
2017-10-31    14446.941694
2017-11-30    14442.274083
2017-12-31    14441.394833
Freq: M, Name: AR(1) Predictions, dtype: float64
```

In [54]:
```
for i in range(len(predictions1)):
    print(f"predicted={predictions1[i]:<11.10}, expected={test['Sales'][i]}")
```

```
predicted=15301.56218, expected=11813.021999999999
predicted=14603.26101, expected=15441.874
predicted=14471.72033, expected=29028.206000000002
predicted=14446.94169, expected=21884.0682
predicted=14442.27408, expected=37056.715
predicted=14441.39483, expected=31407.4668
```

In [55]:
```
test['Sales'].plot(legend=True)
predictions1.plot(legend=True,figsize=(12,6))
```

Out[55]:
```
<Axes: xlabel='Order Date'>
```



In [56]:
```
mod6 = AutoReg(train['Sales'], 6)
res6 = mod6.fit()
print(res6.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Sales   No. Observations:                   42
Model:                     AutoReg(6)   Log Likelihood                -373.443
Method:               Conditional MLE   S.D. of innovations           7742.555
Date:                Thu, 21 Sep 2023   AIC                            762.887
Time:                        00:08:29   BIC                            775.555
Sample:                    07-31-2014   HQIC                           767.308
                         - 06-30-2017
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        1.93e+04   5788.110      3.334      0.001    7950.646    3.06e+04
Sales.L1       0.1277      0.165      0.774      0.439      -0.196       0.451
Sales.L2      -0.0805      0.162     -0.496      0.620      -0.399       0.238
Sales.L3       0.0557      0.159      0.350      0.726      -0.256       0.367
Sales.L4      -0.2548      0.158     -1.610      0.107      -0.565       0.055
Sales.L5      -0.1875      0.160     -1.175      0.240      -0.500       0.125
Sales.L6       0.0541      0.160      0.339      0.735      -0.259       0.367
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.5872           -1.1332j            1.2763           -0.3261
AR.2           -0.5872           +1.1332j            1.2763            0.3261
AR.3           -1.6817           -0.0000j            1.6817           -0.5000
AR.4            0.9181           -0.8134j            1.2266           -0.1154
AR.5            0.9181           +0.8134j            1.2266            0.1154
AR.6            4.4871           -0.0000j            4.4871           -0.0000
-----------------------------------------------------------------------------
```

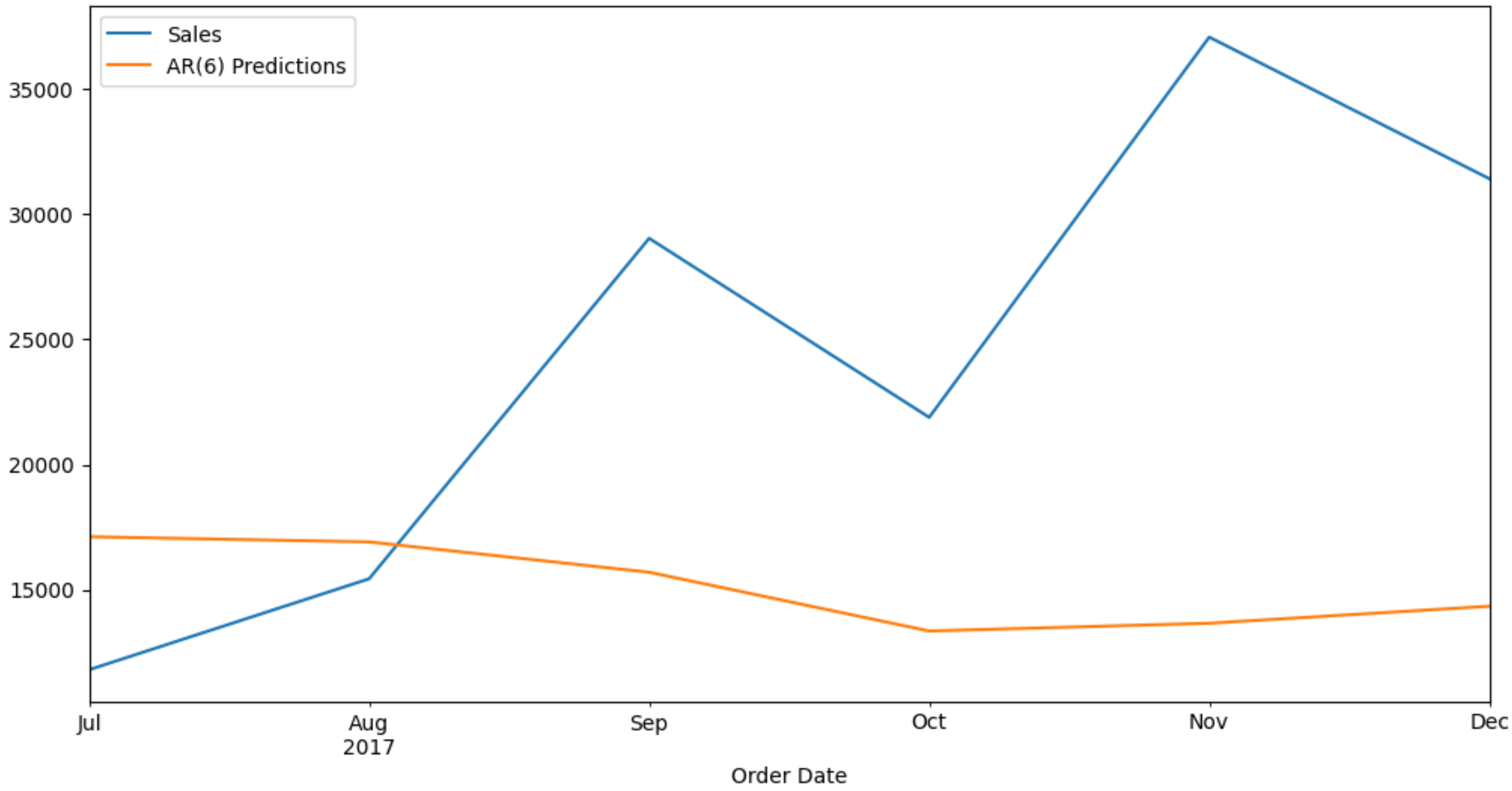
In [57]:

```
start=len(train)
end=len(train)+len(test)-1
predictions6 = res6.predict(start=start, end=end, dynamic=False).rename('AR(6) Predictions')
predictions6
```

Out[57]:
```
2017-07-31    17121.629045
2017-08-31    16914.370614
2017-09-30    15703.623235
2017-10-31    13359.612451
2017-11-30    13669.199630
2017-12-31    14347.505858
Freq: M, Name: AR(6) Predictions, dtype: float64
```

In [58]:
```
test['Sales'].plot(legend=True)
predictions6.plot(legend=True,figsize=(12,6))
```

Out[58]:
```
<Axes: xlabel='Order Date'>
```



In [59]:
```
from statsmodels.tsa.ar_model import ar_select_order
p = ar_select_order(train['Sales'], maxlag=15)
p.ar_lags
```

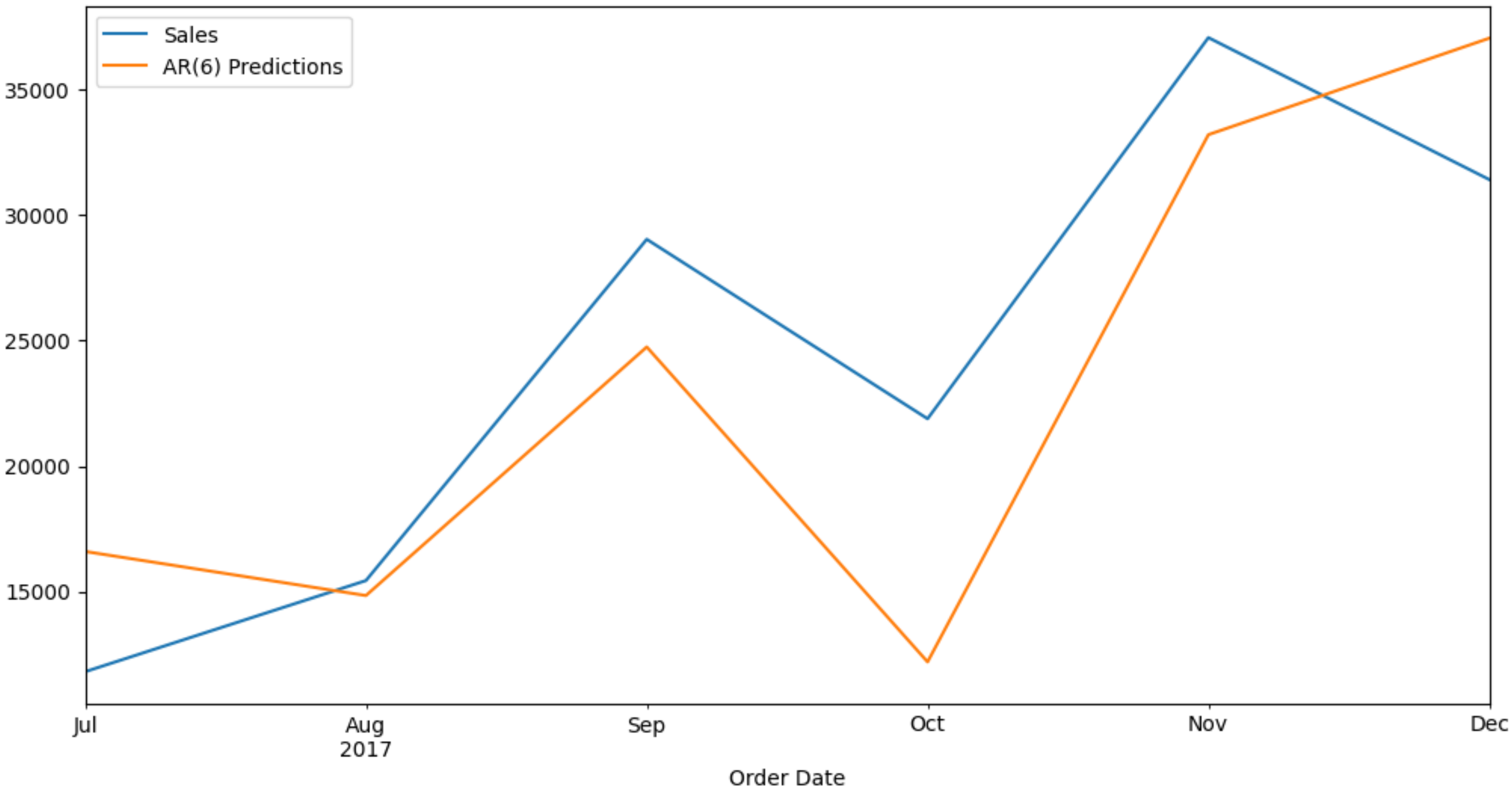
Out[59]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

In [60]:
```
mod12 = AutoReg(train['Sales'], 12)
res12 = mod12.fit()
print(res12.summary())
```

```
                            AutoReg Model Results                             
==============================================================================
Dep. Variable:                  Sales   No. Observations:                   42
Model:                    AutoReg(12)   Log Likelihood                -290.930
Method:               Conditional MLE   S.D. of innovations           3939.183
Date:                Thu, 21 Sep 2023   AIC                            609.860
Time:                        00:09:12   BIC                            629.477
Sample:                    01-31-2015   HQIC                           616.136
                         - 06-30-2017                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       6179.8244   9948.341      0.621      0.534   -1.33e+04    2.57e+04
Sales.L1       0.0175      0.102      0.172      0.864      -0.182       0.217
Sales.L2      -0.0058      0.104     -0.056      0.956      -0.209       0.197
Sales.L3      -0.0491      0.102     -0.481      0.631      -0.249       0.151
Sales.L4      -0.1194      0.103     -1.163      0.245      -0.320       0.082
Sales.L5      -0.0596      0.101     -0.593      0.553      -0.257       0.138
Sales.L6      -0.0091      0.101     -0.091      0.928      -0.206       0.188
Sales.L7       0.0419      0.114      0.369      0.712      -0.181       0.265
Sales.L8       0.0262      0.118      0.221      0.825      -0.206       0.258
Sales.L9      -0.0595      0.115     -0.516      0.606      -0.285       0.166
Sales.L10     -0.0724      0.123     -0.590      0.555      -0.313       0.168
Sales.L11      0.0335      0.118      0.284      0.776      -0.198       0.265
Sales.L12      0.9403      0.115      8.143      0.000       0.714       1.167
                                    Roots                                     
==============================================================================
                   Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1             1.0270           -0.0000j            1.0270           -0.0000
AR.2             0.8601           -0.5085j            0.9991           -0.0850
AR.3             0.8601           +0.5085j            0.9991            0.0850
AR.4             0.5128           -0.8447j            0.9882           -0.1632
AR.5             0.5128           +0.8447j            0.9882            0.1632
AR.6            -0.0108           -1.0048j            1.0049           -0.2517
AR.7            -0.0108           +1.0048j            1.0049            0.2517
AR.8            -0.5131           -0.8690j            1.0091           -0.3349
AR.9            -0.5131           +0.8690j            1.0091            0.3349
AR.10           -0.8731           -0.5059j            1.0091           -0.4164
AR.11           -0.8731           +0.5059j            1.0091            0.4164
AR.12           -1.0144           -0.0000j            1.0144           -0.5000
```

In [61]:
```
start=len(train)
end=len(train)+len(test)-1
predictions12 = res12.predict(start=start, end=end, dynamic=False).rename('AR(6) Predictions')
test['Sales'].plot(legend=True)
predictions12.plot(legend=True,figsize=(12,6))
```

Out[61]: `<Axes: xlabel='Order Date'>`



In [62]:
```
from sklearn.metrics import mean_squared_error
```

```
labels = ['AR(1)','AR(6)','AR(12)']
preds = [predictions1, predictions6, predictions12]  # these are variables, not strings!

for i in range(3):
    error = mean_squared_error(test['Sales'], preds[i])
    print(f'{labels[i]} Error: {error:11.10}')
```

```
AR(1) Error: 179889307.7
AR(6) Error: 186429758.2
AR(12) Error: 30320630.65
```

In [63]:
```
modls = [res1,res6,res12]

for i in range(3):
    print(f'{labels[i]} AIC: {modls[i].aic:6.5}')
```

```
AR(1) AIC: 860.49
AR(6) AIC: 762.89
AR(12) AIC: 609.86
```

In [64]:
```
mod12 = AutoReg(df['Sales'], 12)
res12 = mod12.fit()
print(res12.summary())
```

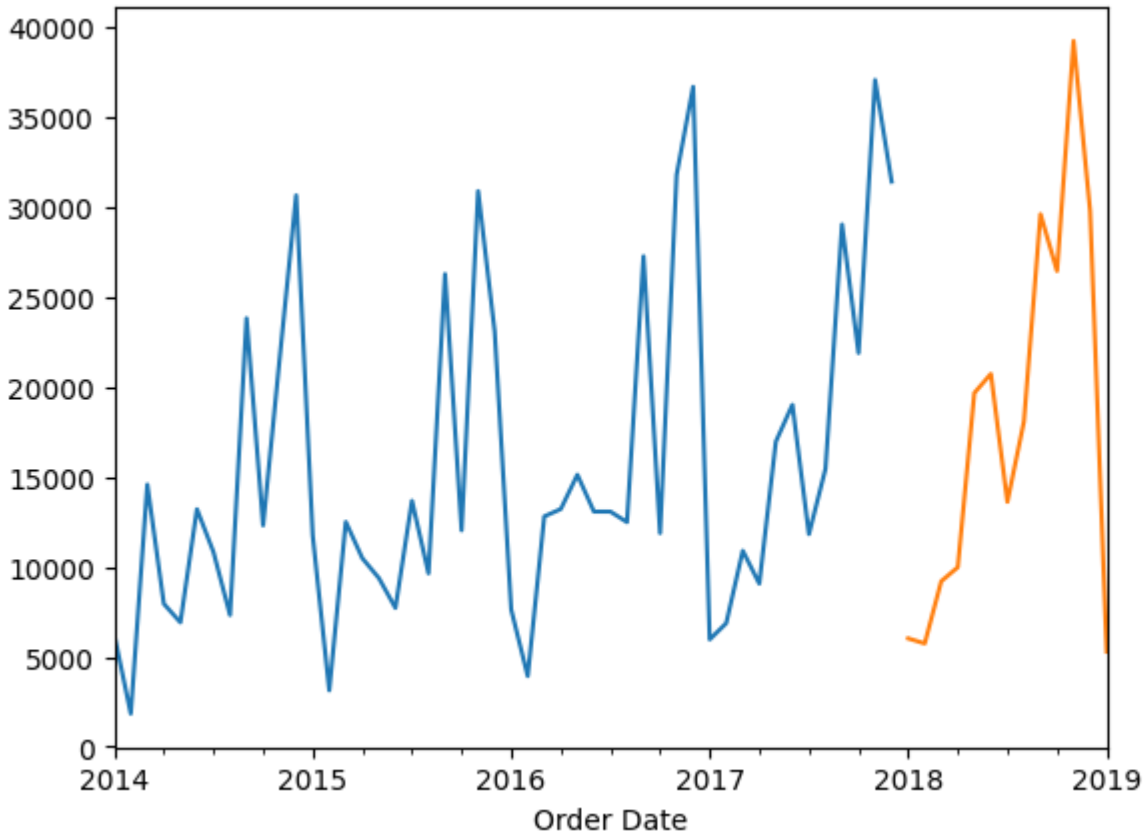
```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Sales   No. Observations:                   48
Model:                    AutoReg(12)   Log Likelihood                -349.076
Method:               Conditional MLE   S.D. of innovations           3934.790
Date:                Thu, 21 Sep 2023   AIC                            726.152
Time:                        00:10:01   BIC                            748.321
Sample:                    01-31-2015   HQIC                           733.889
                         - 12-31-2017
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       4356.0138   7562.612      0.576      0.565   -1.05e+04    1.92e+04
Sales.L1       0.0321      0.088      0.366      0.715      -0.140       0.204
Sales.L2      -0.0097      0.094     -0.102      0.918      -0.195       0.175
Sales.L3      -0.0670      0.092     -0.727      0.467      -0.248       0.114
Sales.L4      -0.0820      0.097     -0.846      0.398      -0.272       0.108
Sales.L5      -0.0348      0.094     -0.369      0.712      -0.219       0.150
Sales.L6       0.0304      0.095      0.320      0.749      -0.155       0.216
Sales.L7      -0.0261      0.093     -0.279      0.780      -0.209       0.157
Sales.L8      -0.0317      0.093     -0.339      0.734      -0.215       0.151
Sales.L9      -0.0284      0.094     -0.303      0.762      -0.212       0.155
Sales.L10      0.0041      0.093      0.044      0.965      -0.178       0.186
Sales.L11      0.1429      0.090      1.592      0.111      -0.033       0.319
Sales.L12      0.8895      0.089      9.993      0.000       0.715       1.064
                                    Roots
==============================================================================
                   Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1             1.0149           -0.0000j            1.0149           -0.0000
AR.2             0.8510           -0.5063j            0.9902           -0.0854
AR.3             0.8510           +0.5063j            0.9902            0.0854
AR.4             0.4907           -0.8578j            0.9882           -0.1673
AR.5             0.4907           +0.8578j            0.9882            0.1673
AR.6            -0.0094           -1.0212j            1.0212           -0.2515
AR.7            -0.0094           +1.0212j            1.0212            0.2515
AR.8            -0.5178           -0.8730j            1.0150           -0.3352
AR.9            -0.5178           +0.8730j            1.0150            0.3352
AR.10           -1.0205           -0.0000j            1.0205           -0.5000
AR.11           -0.8920           -0.5093j            1.0272           -0.4174
AR.12           -0.8920           +0.5093j            1.0272            0.4174
```

In [65]:
```
start = len(df)
end = len(df)+12

pred_future = res12.predict(start=start, end=end, dynamic=False)

pred_future
```

Out[65]:
```
2018-01-31     6022.626972
2018-02-28     5739.250407
2018-03-31     9199.016790
2018-04-30     9976.759362
2018-05-31    19661.613494
2018-06-30    20722.318428
2018-07-31    13609.961837
2018-08-31    18080.731150
2018-09-30    29604.129926
2018-10-31    26437.359412
2018-11-30    39227.722049
2018-12-31    29781.186936
2019-01-31     5288.024913
Freq: M, dtype: float64
```

In [66]:
```
df['Sales'].plot()
pred_future.plot()
```

Out[66]:
```
<Axes: xlabel='Order Date'>
```



In [67]:
```
pip install pmdarima
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pmdarima in c:\users\saswa\appdata\roaming\python\python311\site-packages (2.0.3)
Requirement already satisfied: joblib>=0.11 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdarima)
(1.3.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\users\saswa\appdata\roaming\python\python311\site-package
s (from pmdarima) (3.0.2)
Requirement already satisfied: numpy>=1.21.2 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdarima)
(1.24.2)
Requirement already satisfied: pandas>=0.19 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdarima)
(1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdari
ma) (1.3.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdarima)
(1.11.2)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdar
ima) (0.14.0)
Requirement already satisfied: urllib3 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pmdarima) (2.0.4)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\program files\python311\lib\site-packages (from pmdarima) (65.
5.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pa
ndas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from pandas>=0.19
->pmdarima) (2022.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from scik
it-learn>=0.22->pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from statsmodels>
=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\saswa\appdata\roaming\python\python311\site-packages (from statsmode
ls>=0.13.2->pmdarima) (23.1)
Requirement already satisfied: six in c:\users\saswa\appdata\roaming\python\python311\site-packages (from patsy>=0.5.2->statsmo
dels>=0.13.2->pmdarima) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [68]:

```python
import pandas as pd
import numpy as np
%matplotlib inline

# Load specific forecasting tools
from statsmodels.tsa.arima_model import ARMA,ARMAResults,ARIMA,ARIMAResults
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf # for determining (p,q) orders
from pmdarima import auto_arima # for determining ARIMA orders
```

```
# Ignore harmless warnings
import warnings
warnings.filterwarnings("ignore")
```

In [69]:

```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(4)
```

Out[69]:

Order Date	Sales
2014-01-06	2573.820
2014-01-07	76.728
2014-01-10	51.940
2014-01-11	9.940

In [70]:

```
df = df.resample('MS').sum()
df.head()
```

Out[70]:

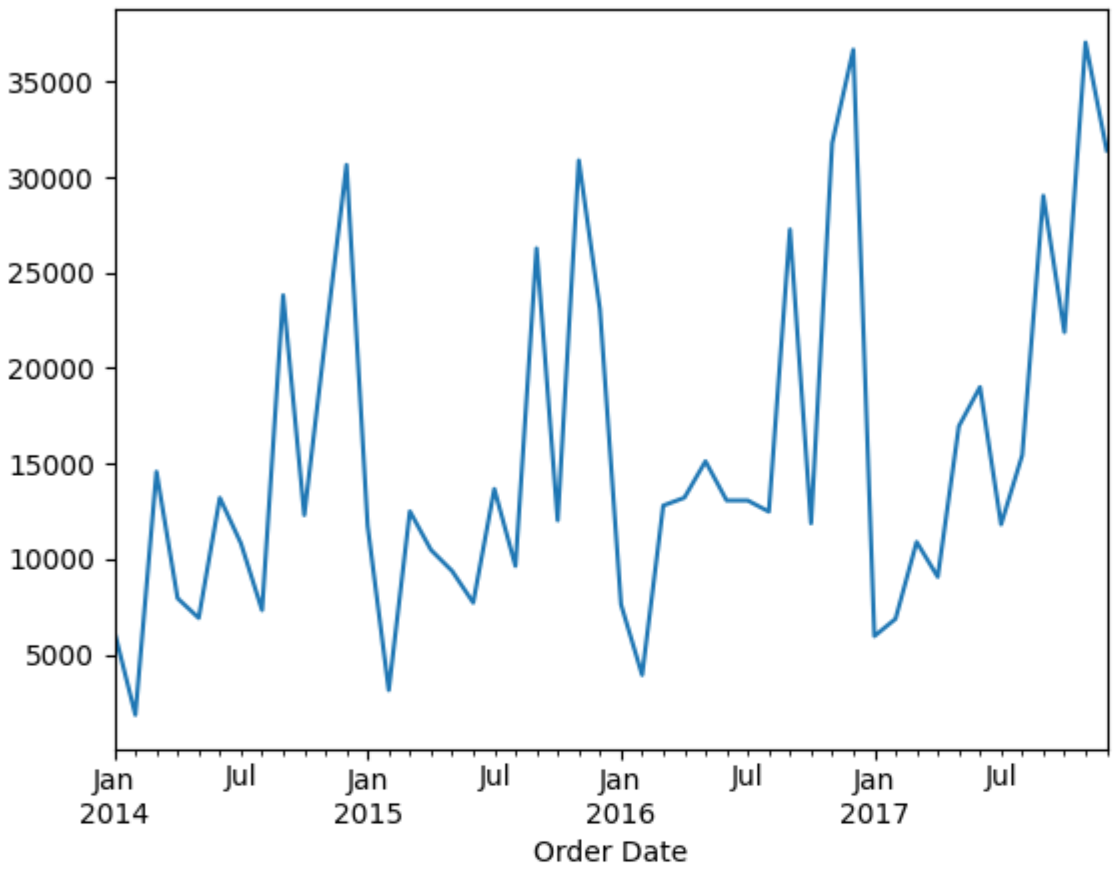
Order Date	Sales
2014-01-01	6242.525
2014-02-01	1839.658
2014-03-01	14573.956
2014-04-01	7944.837
2014-05-01	6912.787

In [71]:
```
from statsmodels.tsa.stattools import adfuller
```

In [72]:
```
def adf_test(series,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series.dropna(),autolag='AIC')

    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    print(out.to_string())

    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has no unit root and is stationary")
    else:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has a unit root and is non-stationary")
```

In [73]:
```
df['Sales'].plot()
```

Out[73]:
```
<Axes: xlabel='Order Date'>
```



In [74]:
```
adf_test(df['Sales'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic      -4.699026
p-value                  0.000085
# lags used              0.000000
# observations          47.000000
critical value (1%)     -3.577848
critical value (5%)     -2.925338
critical value (10%)    -2.600774
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [75]:
```
auto_arima(df['Sales'], seasonal=True).summary()
```

Out[75]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	48
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	-502.820
Date:	Thu, 21 Sep 2023	**AIC**	1011.640
Time:	00:14:34	**BIC**	1017.253
Sample:	01-01-2014	**HQIC**	1013.761
	- 12-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	1.084e+04	2695.066	4.021	0.000	5554.237	1.61e+04
ar.L1	0.3056	0.131	2.328	0.020	0.048	0.563
sigma2	7.318e+07	0.160	4.56e+08	0.000	7.32e+07	7.32e+07

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	3.70
Prob(Q):	0.98	**Prob(JB):**	0.16
Heteroskedasticity (H):	1.88	**Skew:**	0.64
Prob(H) (two-sided):	0.22	**Kurtosis:**	2.54

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 2.09e+24. Standard errors may be unstable.

In [76]:
```
train = df.iloc[:len(df)-6]
test = df.iloc[len(df)-6:]

len(train), len(test)
```

Out[76]: (42, 6)

In [77]:
```
help(ARMA)
```

```
Help on class ARMA in module statsmodels.tsa.arima_model:

class ARMA(builtins.object)
 |  ARMA(*args, **kwargs)
 |
 |  ARMA has been deprecated in favor of the new implementation
 |
 |  See Also
 |  --------
 |  statsmodels.tsa.arima.model.ARIMA
 |      ARIMA models with a variety of parameter estimators
 |  statsmodels.tsa.statespace.SARIMAX
 |      SARIMAX models estimated using MLE
 |
 |  Methods defined here:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

In [78]:
```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(train['Sales'],order=(2,0,2))
results = model.fit()
```

```
results.summary()
```

Out[78]:

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	42
Model:	ARIMA(2, 0, 2)	**Log Likelihood**	-436.980
Date:	Thu, 21 Sep 2023	**AIC**	885.960
Time:	00:15:08	**BIC**	896.386
Sample:	01-01-2014	**HQIC**	889.782
	- 06-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.418e+04	1427.017	9.934	0.000	1.14e+04	1.7e+04
ar.L1	0.6732	0.810	0.831	0.406	-0.914	2.260
ar.L2	0.1773	0.842	0.211	0.833	-1.474	1.828
ma.L1	-0.5278	0.739	-0.714	0.475	-1.977	0.921
ma.L2	-0.4528	0.771	-0.587	0.557	-1.964	1.059
sigma2	7.04e+07	0.014	4.87e+09	0.000	7.04e+07	7.04e+07

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	8.53
Prob(Q):	0.93	**Prob(JB):**	0.01
Heteroskedasticity (H):	1.31	**Skew:**	1.10
Prob(H) (two-sided):	0.62	**Kurtosis:**	3.27

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

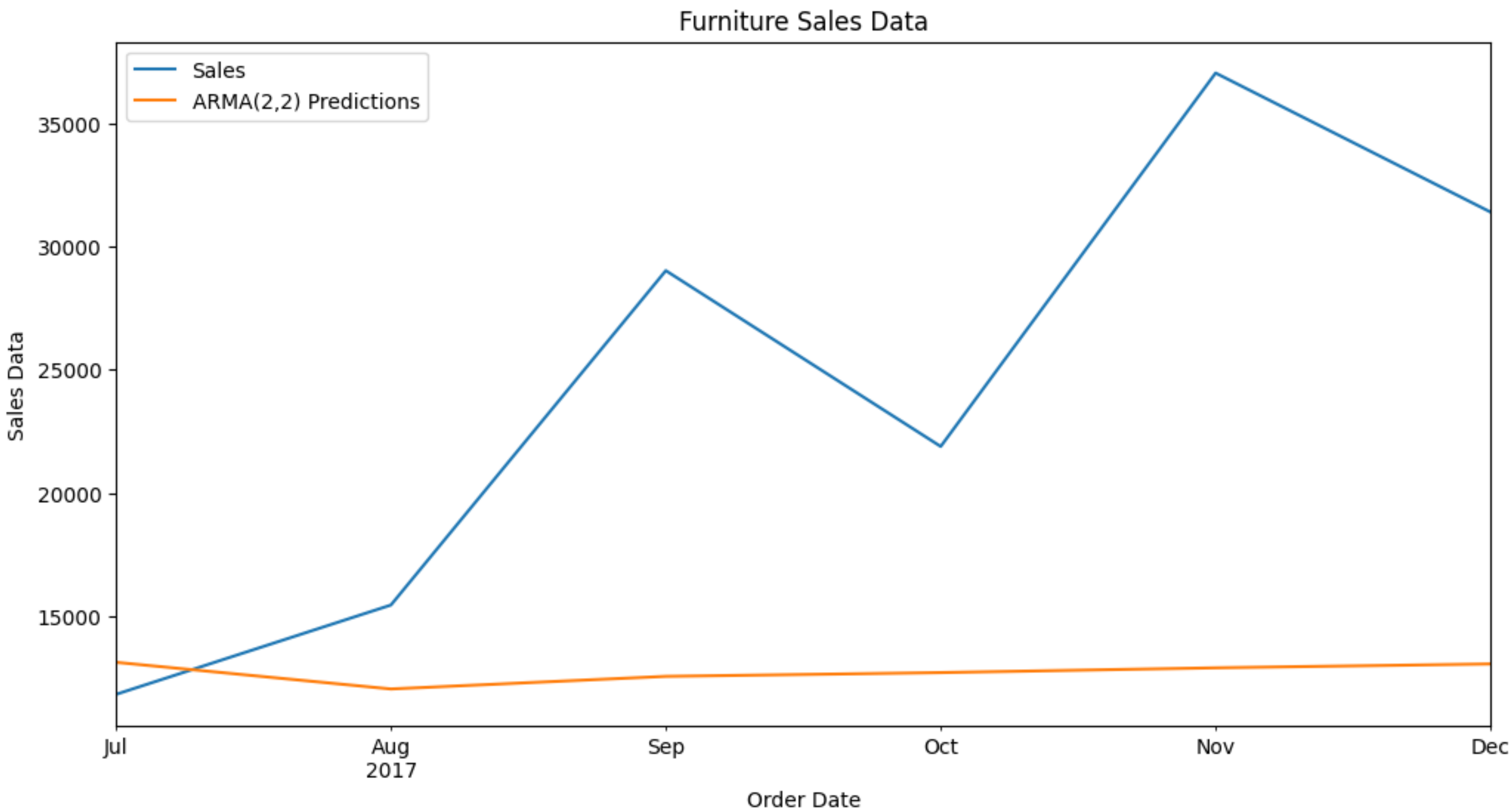
[2] Covariance matrix is singular or near-singular, with condition number 2.64e+25. Standard errors may be unstable.

In [79]:
```
start=len(train)
end=len(train)+len(test)-1
predictions = results.predict(start=start, end=end).rename('ARMA(2,2) Predictions')
predictions
```

Out[79]:
```
2017-07-01    13115.074287
2017-08-01    12031.112128
2017-09-01    12543.910534
2017-10-01    12696.894212
2017-11-01    12890.811988
2017-12-01    13048.480070
Freq: MS, Name: ARMA(2,2) Predictions, dtype: float64
```

In [80]:
```
title = 'Furniture Sales Data'
ylabel='Sales Data'
xlabel='Order Date'

ax = test['Sales'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[80]:
```
[Text(0.5, 0, 'Order Date'), Text(0, 0.5, 'Sales Data')]
```



In [81]:
```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(4)
```

Out[81]:

	Sales
Order Date	
2014-01-06	2573.820
2014-01-07	76.728
2014-01-10	51.940
2014-01-11	9.940

In [82]:

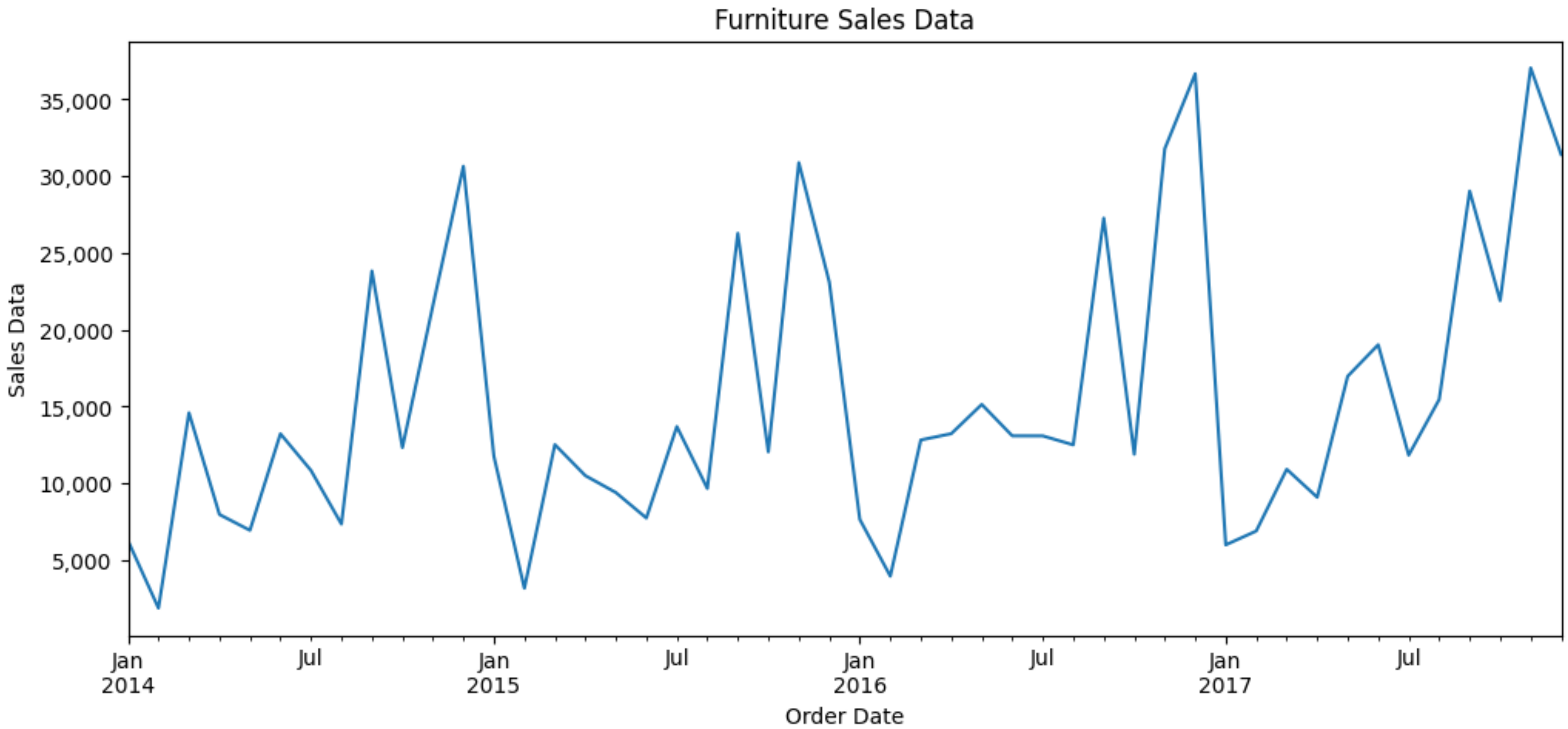
```
df = df.resample('MS').sum()
df.head()
```

Out[82]:

	Sales
Order Date	
2014-01-01	6242.525
2014-02-01	1839.658
2014-03-01	14573.956
2014-04-01	7944.837
2014-05-01	6912.787

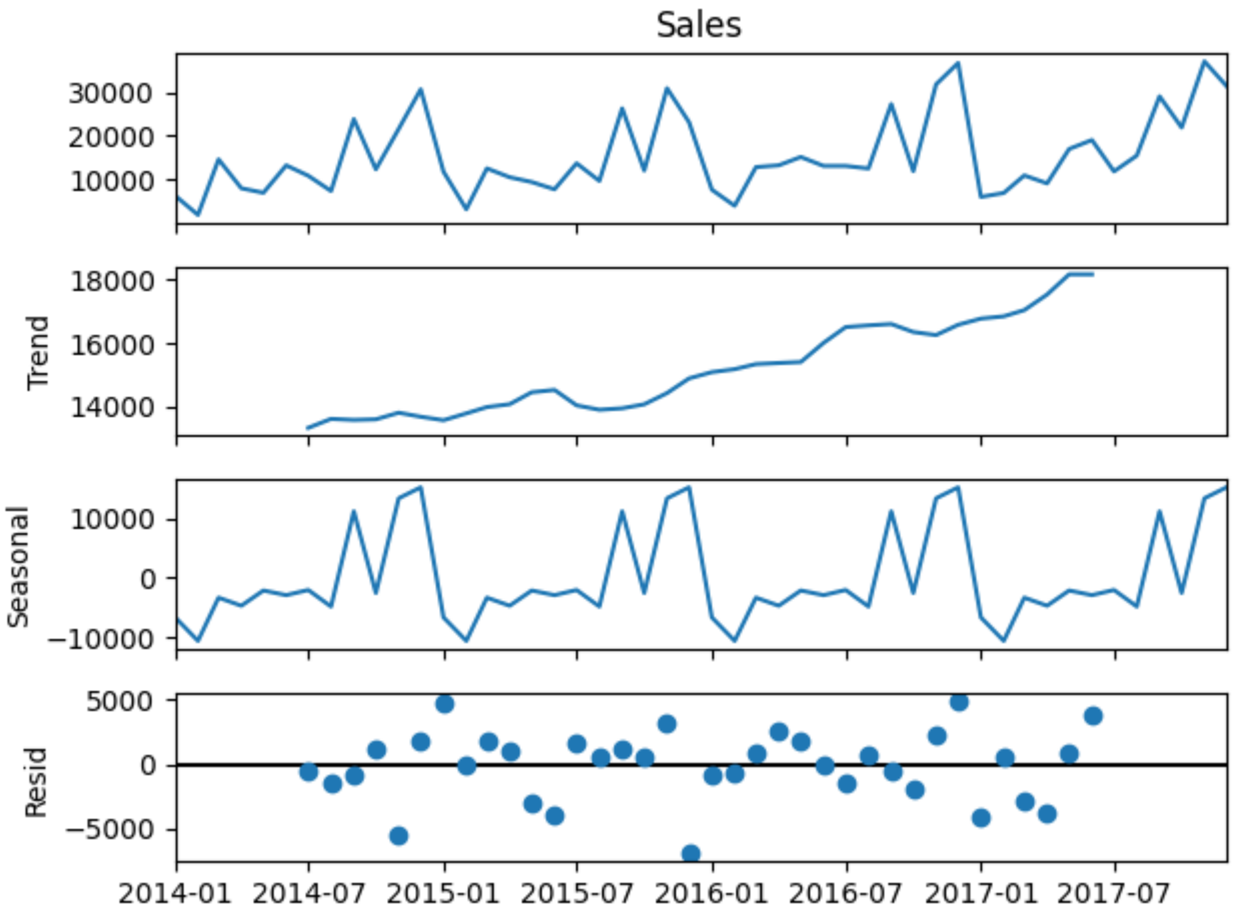
In [83]:

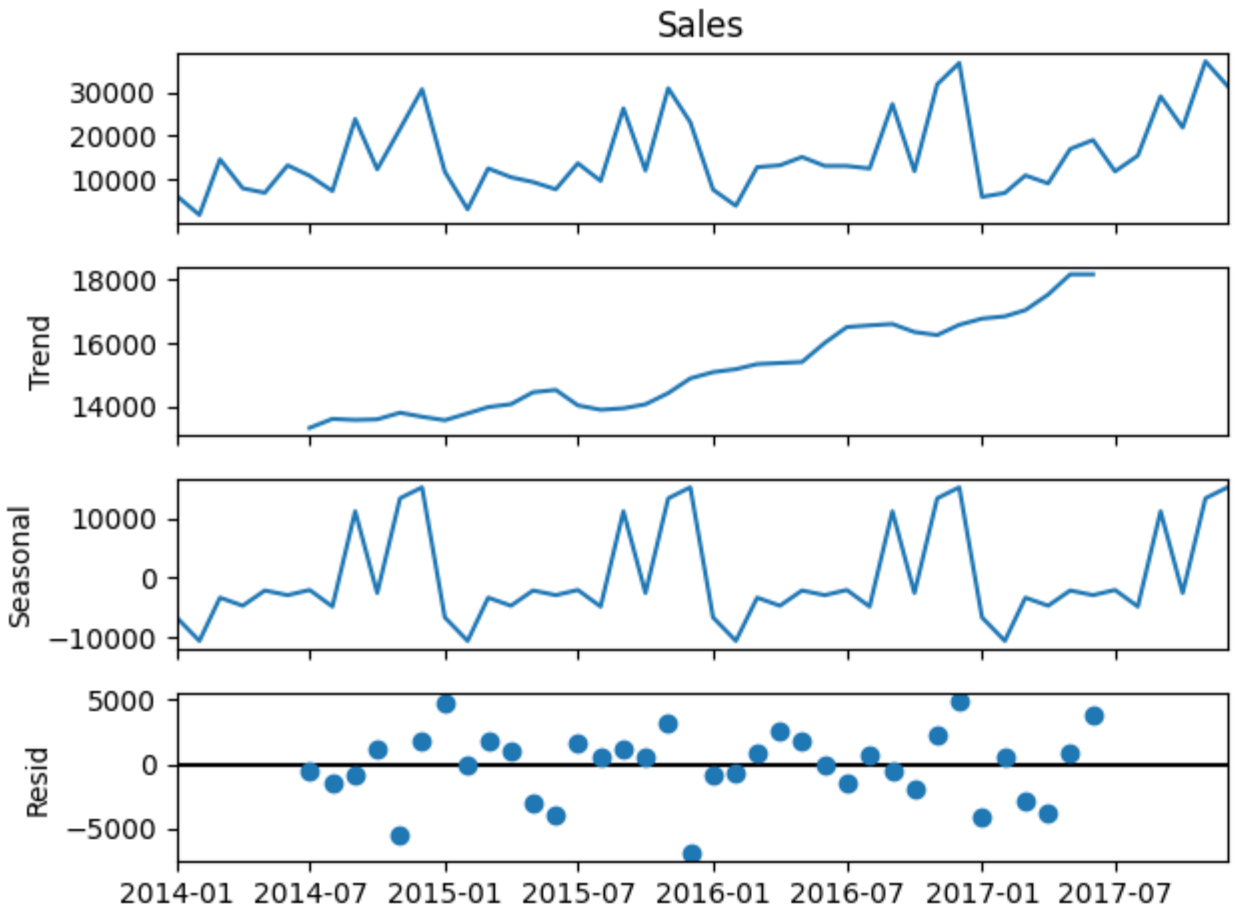
```
import matplotlib.ticker as ticker
formatter = ticker.StrMethodFormatter('{x:,.0f}')

title = 'Furniture Sales Data'
ylabel='Sales Data'
xlabel='Order Date'

ax = df['Sales'].plot(figsize=(12,5),title=title)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter);
```



In [84]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(df['Sales'], model='additive')
result.plot()
```

Out[84]:


Sales
30000
20000
10000
Trend
18000
16000
14000
Seasonal
10000
0
−10000
Resid
5000
0
−5000
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07



In [85]:
```
auto_arima(df['Sales'], seasonal=False).summary()
```

Out[85]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	48
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	-502.820
Date:	Thu, 21 Sep 2023	**AIC**	1011.640
Time:	00:17:17	**BIC**	1017.253
Sample:	01-01-2014	**HQIC**	1013.761
	- 12-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	1.084e+04	2695.066	4.021	0.000	5554.237	1.61e+04
ar.L1	0.3056	0.131	2.328	0.020	0.048	0.563
sigma2	7.318e+07	0.160	4.56e+08	0.000	7.32e+07	7.32e+07

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	3.70
Prob(Q):	0.98	**Prob(JB):**	0.16
Heteroskedasticity (H):	1.88	**Skew:**	0.64
Prob(H) (two-sided):	0.22	**Kurtosis:**	2.54

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 2.09e+24. Standard errors may be unstable.

In [86]:

```
from statsmodels.tsa.statespace.tools import diff
```

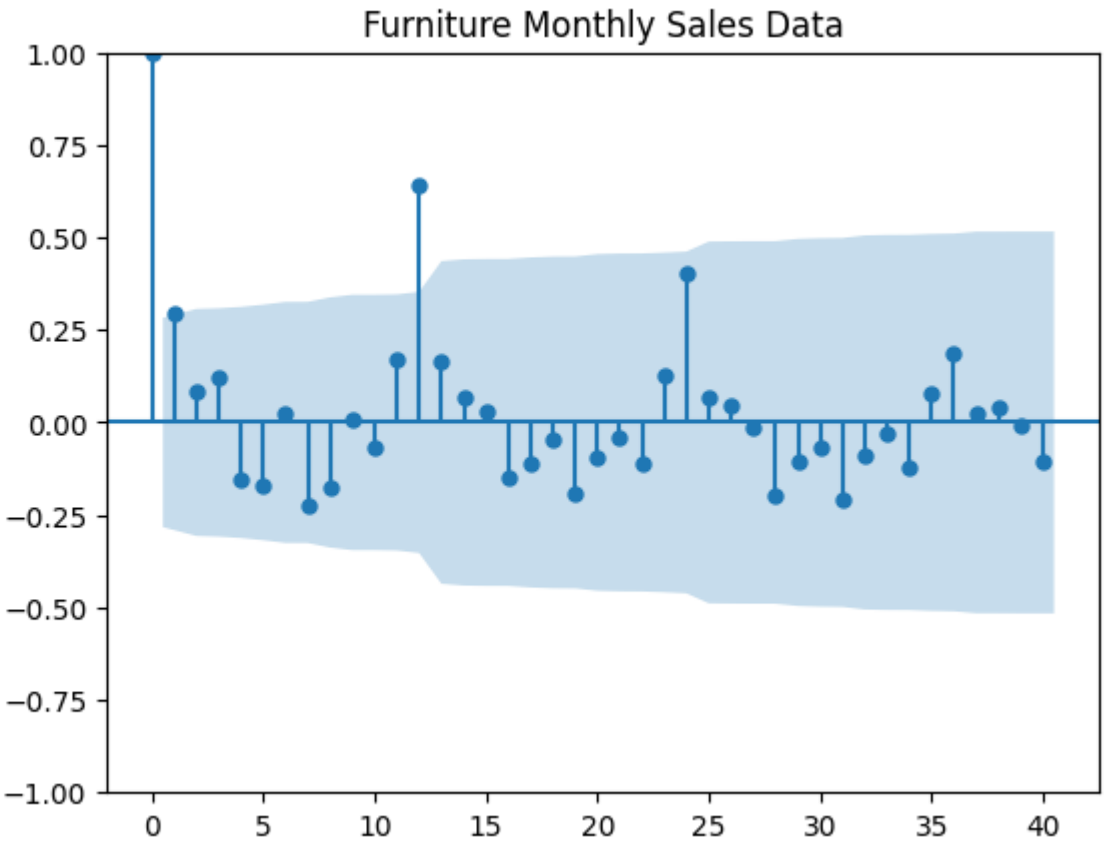
```
df['d1'] = diff(df['Sales'], k_diff=1)

adf_test(df['d1'], 'Furniture Sales Data')
```

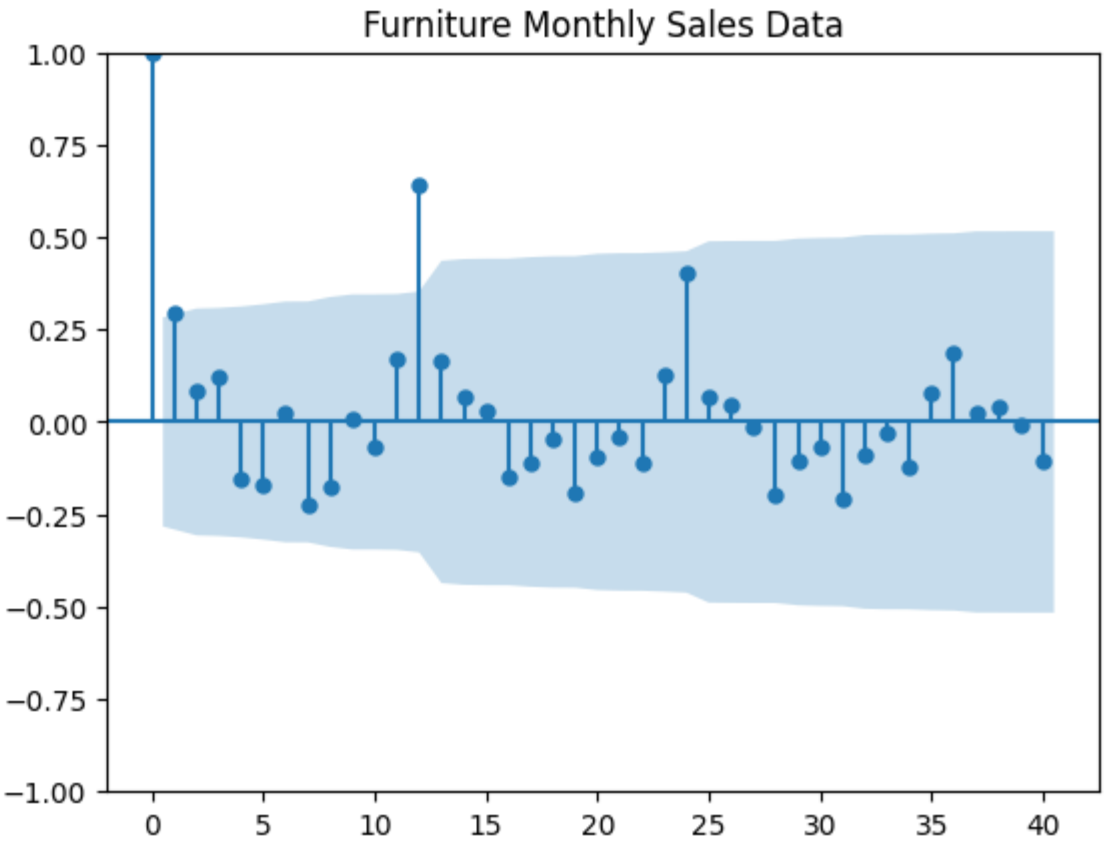
```
Augmented Dickey-Fuller Test: Furniture Sales Data
ADF test statistic     -1.147459e+01
p-value                 5.167971e-21
# lags used             1.000000e+01
# observations          3.600000e+01
critical value (1%)    -3.626652e+00
critical value (5%)    -2.945951e+00
critical value (10%)   -2.611671e+00
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [87]:

```
title = 'Furniture Monthly Sales Data'
lags = 40
plot_acf(df['Sales'],title=title,lags=lags)
```

Out[87]:

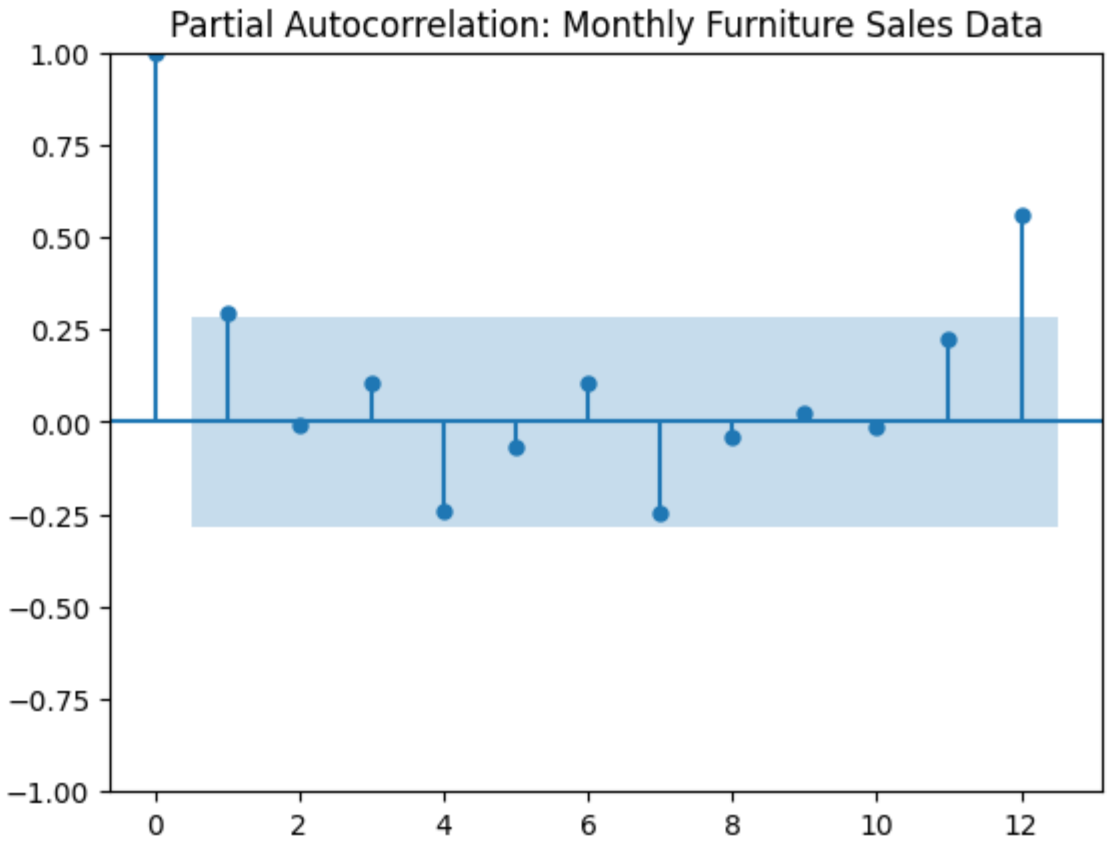

Furniture Monthly Sales Data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40

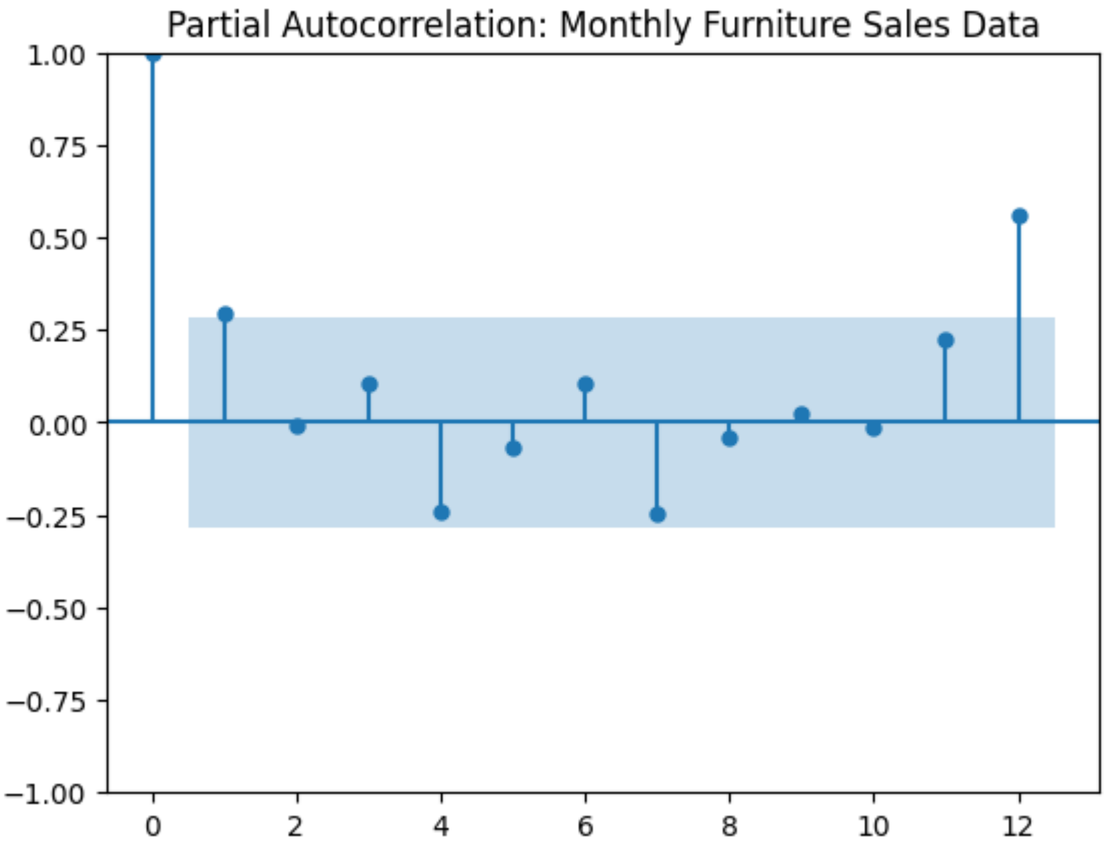


In [88]:

```
title = 'Partial Autocorrelation: Monthly Furniture Sales Data'
lags = 12
plot_pacf(df['Sales'],title=title,lags=lags)
```

Out[88]:


Partial Autocorrelation: Monthly Furniture Sales Data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
12



In [89]:
```
stepwise_fit = auto_arima(df['Sales'], start_p=0, start_q=0,
                          max_p=2, max_q=2, m=12,
                          seasonal=False,
                          d=None, trace=True,
                          error_action='ignore',   # we don't want to know if an order does not work
                          suppress_warnings=True,  # we don't want convergence warnings
                          stepwise=True)           # set to stepwise

stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=1078.259, Time=0.02 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=1026.129, Time=0.04 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=1059.367, Time=0.05 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=1022.619, Time=0.06 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=1018.040, Time=0.18 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=1018.347, Time=0.14 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=1020.706, Time=0.44 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=1019.172, Time=0.15 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=1015.561, Time=0.15 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=1013.580, Time=0.08 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=1011.909, Time=0.15 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=1014.386, Time=0.08 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=1013.938, Time=0.09 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=1011.640, Time=0.06 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=1013.680, Time=0.18 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 1.890 seconds
```

Out[89]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	48
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	-502.820
Date:	Thu, 21 Sep 2023	**AIC**	1011.640
Time:	00:18:13	**BIC**	1017.253
Sample:	01-01-2014	**HQIC**	1013.761
	- 12-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	1.084e+04	2695.066	4.021	0.000	5554.237	1.61e+04
ar.L1	0.3056	0.131	2.328	0.020	0.048	0.563
sigma2	7.318e+07	0.160	4.56e+08	0.000	7.32e+07	7.32e+07

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	3.70
Prob(Q):	0.98	**Prob(JB):**	0.16
Heteroskedasticity (H):	1.88	**Skew:**	0.64
Prob(H) (two-sided):	0.22	**Kurtosis:**	2.54

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 2.09e+24. Standard errors may be unstable.

In [90]:

```
n = len(df)
test_size = 12
```

```
train = df.iloc[:n-test_size]
test = df.iloc[n-test_size:]
```

In [91]:
```
model = ARIMA(train['Sales'], order=(1, 0, 0))

results = model.fit()
results.summary()
```

Out[91]:

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	36
Model:	ARIMA(1, 0, 0)	**Log Likelihood**	-376.058
Date:	Thu, 21 Sep 2023	**AIC**	758.116
Time:	00:18:37	**BIC**	762.867
Sample:	01-01-2014	**HQIC**	759.774
	- 12-01-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.463e+04	1934.689	7.561	0.000	1.08e+04	1.84e+04
ar.L1	0.2825	0.170	1.658	0.097	-0.051	0.616
sigma2	6.929e+07	0.134	5.15e+08	0.000	6.93e+07	6.93e+07

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	4.47
Prob(Q):	0.92	**Prob(JB):**	0.11
Heteroskedasticity (H):	1.40	**Skew:**	0.83
Prob(H) (two-sided):	0.57	**Kurtosis:**	2.53

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 5.41e+25. Standard errors may be unstable.

In [92]:

```
start=len(train)
end=len(train)+len(test)-1
```

```
predictions = results.predict(start=start, end=end, dynamic=False, typ='levels').rename('ARIMA(1,0,0) Predictions')
predictions
```

Out[92]:

```
2017-01-01    20856.524035
2017-02-01    16387.395968
2017-03-01    15125.048311
2017-04-01    14768.486230
2017-05-01    14667.771885
2017-06-01    14639.324162
2017-07-01    14631.288832
2017-08-01    14629.019177
2017-09-01    14628.378091
2017-10-01    14628.197011
2017-11-01    14628.145863
2017-12-01    14628.131415
Freq: MS, Name: ARIMA(1,0,0) Predictions, dtype: float64
```

In [93]:

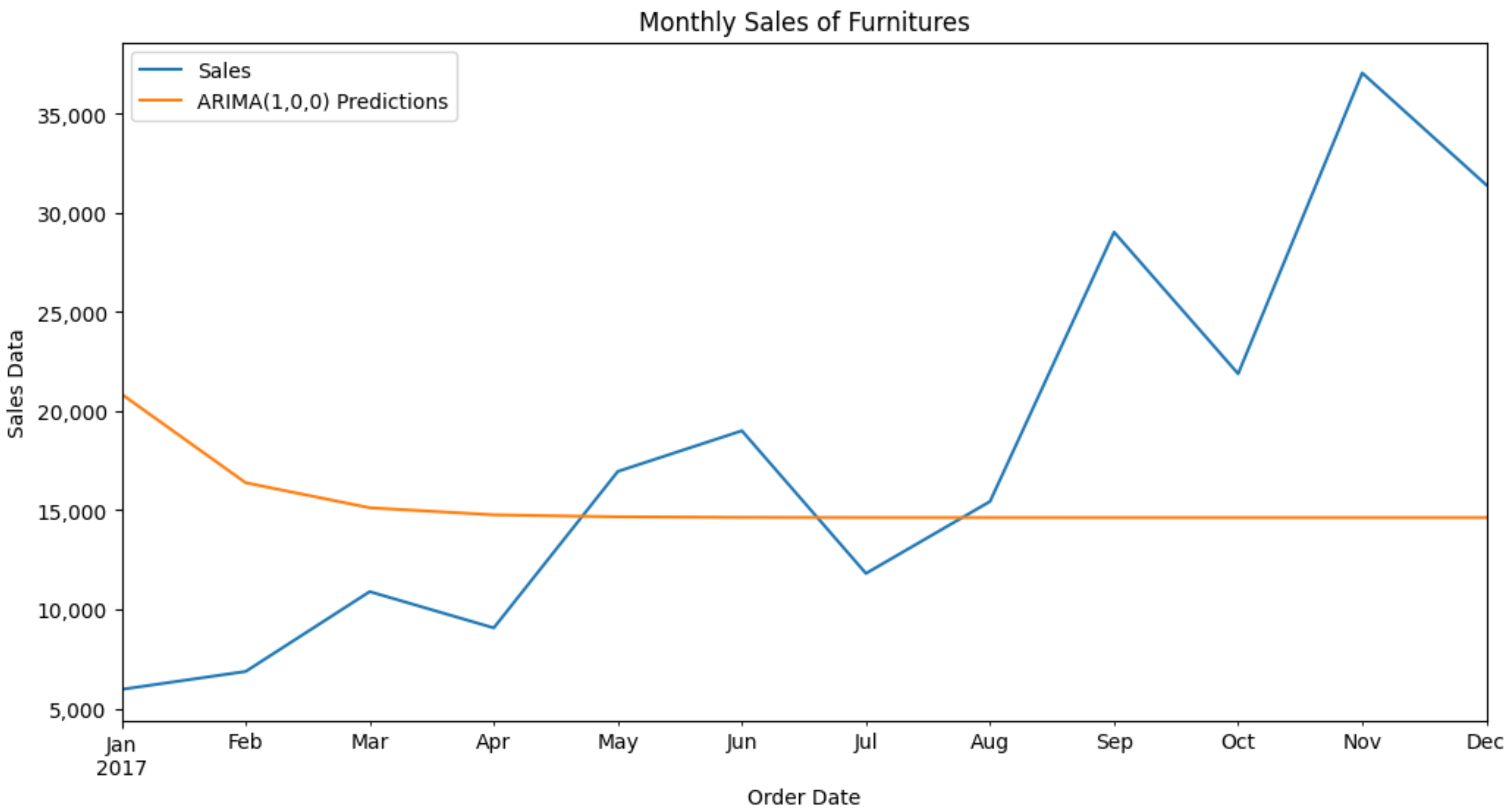
```
for i in range(len(predictions)):
    print(f"predicted={predictions[i]:<11.10}, expected={test['Sales'][i]}")
```

```
predicted=20856.52404, expected=5964.032
predicted=16387.39597, expected=6866.3374
predicted=15125.04831, expected=10893.4448
predicted=14768.48623, expected=9065.9581
predicted=14667.77188, expected=16957.5582
predicted=14639.32416, expected=19008.5867
predicted=14631.28883, expected=11813.021999999999
predicted=14629.01918, expected=15441.874
predicted=14628.37809, expected=29028.206000000002
predicted=14628.19701, expected=21884.0682
predicted=14628.14586, expected=37056.715
predicted=14628.13142, expected=31407.4668
```

In [94]:

```
title = 'Monthly Sales of Furnitures'
ylabel='Sales Data'
xlabel='Order Date'

ax = test['Sales'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

```
ax.yaxis.set_major_formatter(formatter)
```



In [95]:
```
from sklearn.metrics import mean_squared_error

error = mean_squared_error(test['Sales'], predictions)
print(f'ARIMA(1,0,0) MSE Error: {error:11.10}')
```

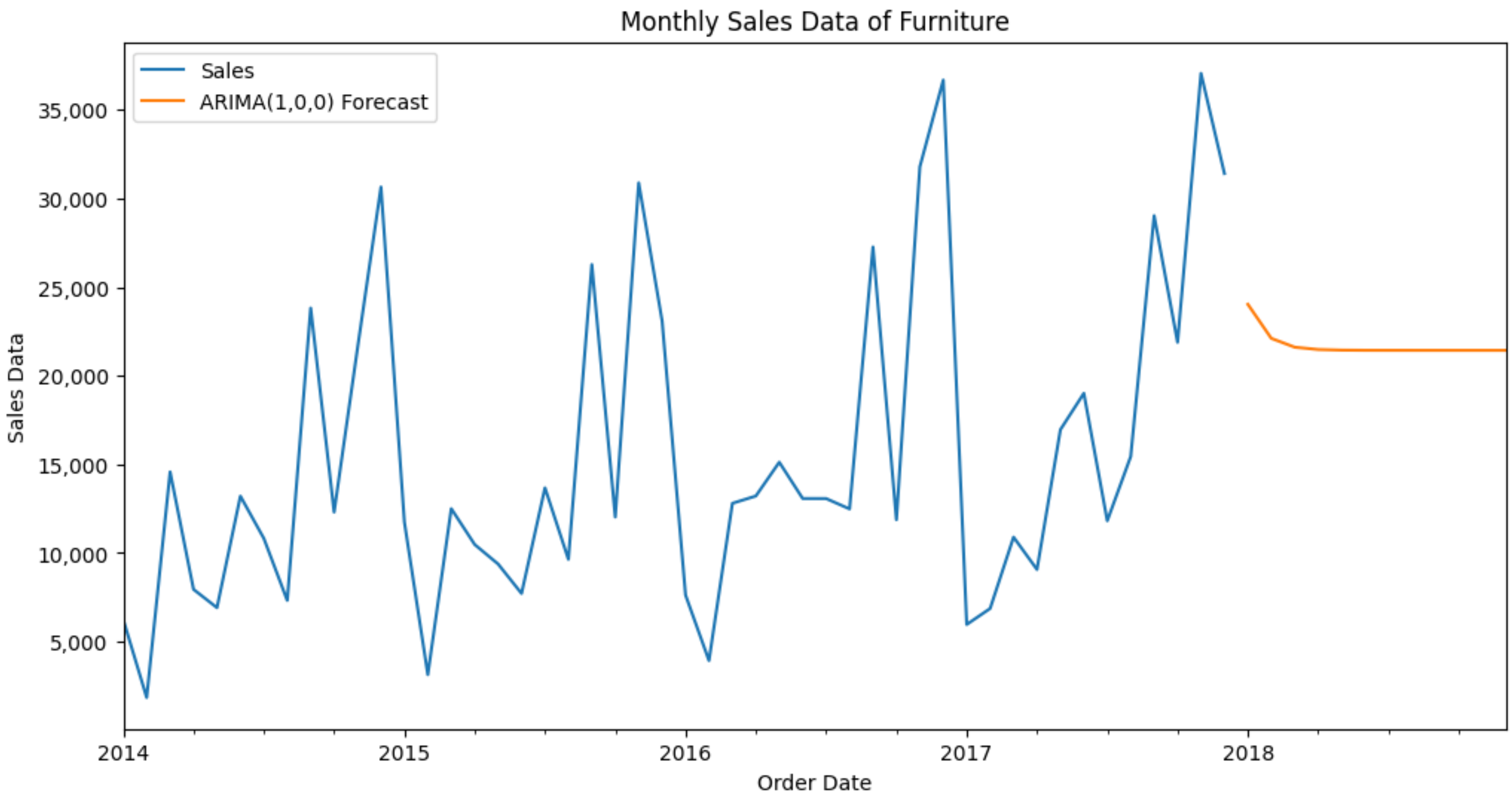
```
ARIMA(1,0,0) MSE Error: 120032385.7
```

In [96]:
```
from statsmodels.tools.eval_measures import rmse
```

```
error = rmse(test['Sales'], predictions)
print(f'ARIMA(1,0,0) RMSE Error: {error:11.10}')
```

```
ARIMA(1,0,0) RMSE Error: 10955.92925
```

In [97]:
```
model = ARIMA(df['Sales'],order=(1,1,1))
results = model.fit()
fcast = results.predict(len(df),len(df)+11,typ='levels').rename('ARIMA(1,0,0) Forecast')
title = 'Monthly Sales Data of Furniture'
ylabel='Sales Data'
xlabel='Order Date'

ax = df['Sales'].plot(legend=True,figsize=(12,6),title=title)
fcast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter)
```



In [98]:
```
import pandas as pd
import numpy as np
%matplotlib inline

from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima import auto_arima

import warnings
```

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

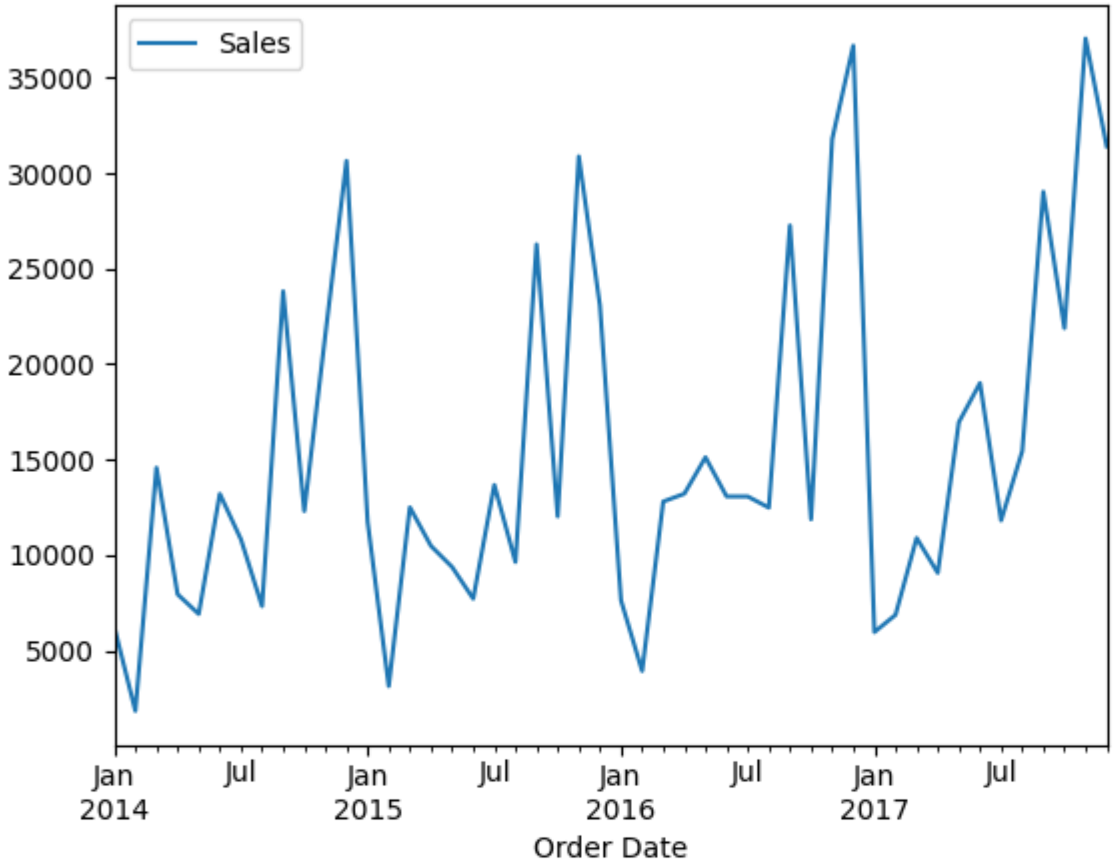
In [99]:
```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum})
df.sort_index(inplace=True)
df.head(15)
```

Out[99]:

	Sales
Order Date	
2014-01-06	2573.820
2014-01-07	76.728
2014-01-10	51.940
2014-01-11	9.940
2014-01-13	879.939
2014-01-14	61.960
2014-01-16	127.104
2014-01-19	181.470
2014-01-20	1413.510
2014-01-21	25.248
2014-01-26	217.200
2014-01-27	333.000
2014-01-31	290.666
2014-02-08	14.560
2014-02-11	1650.050

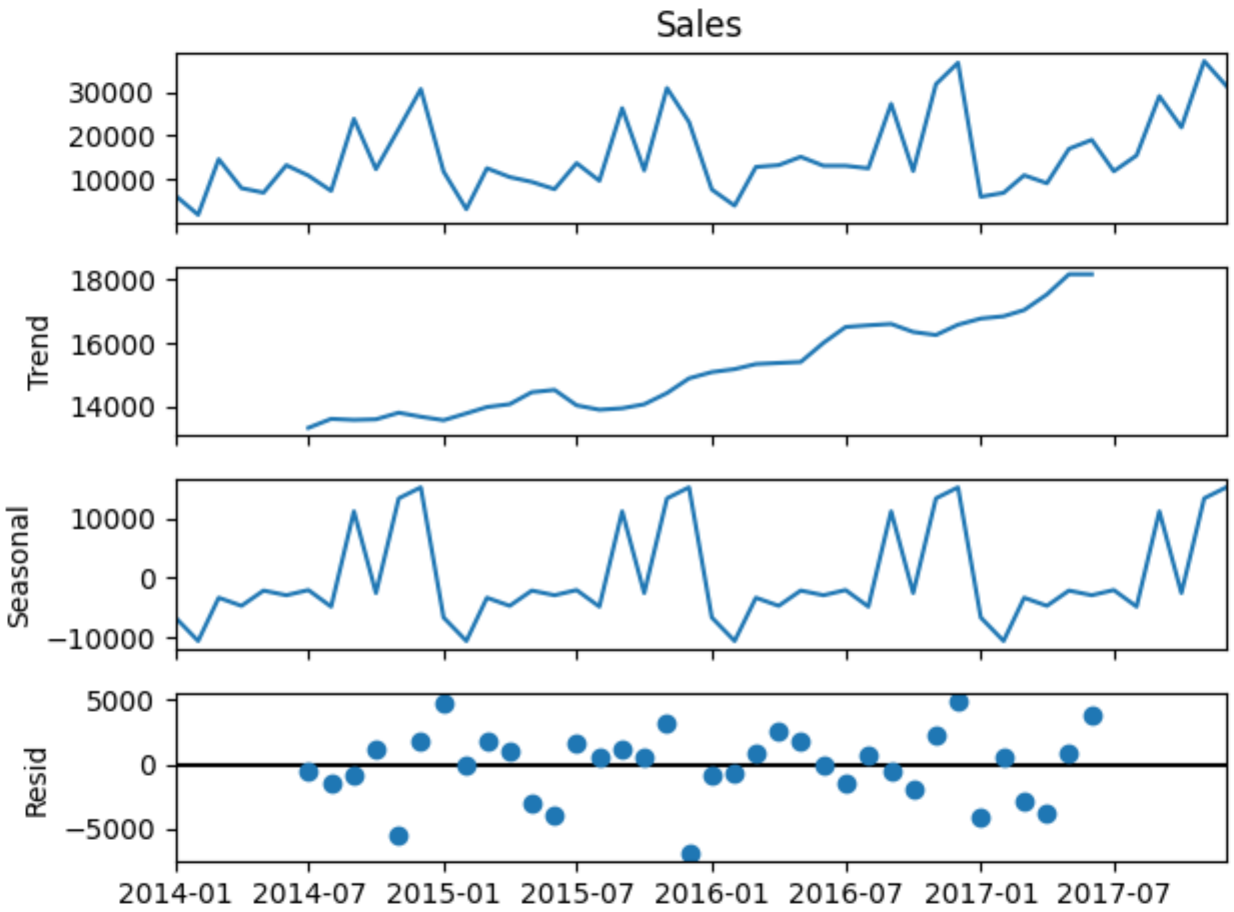
In [100…
```
df = df.resample('MS').sum()
df.plot()
```

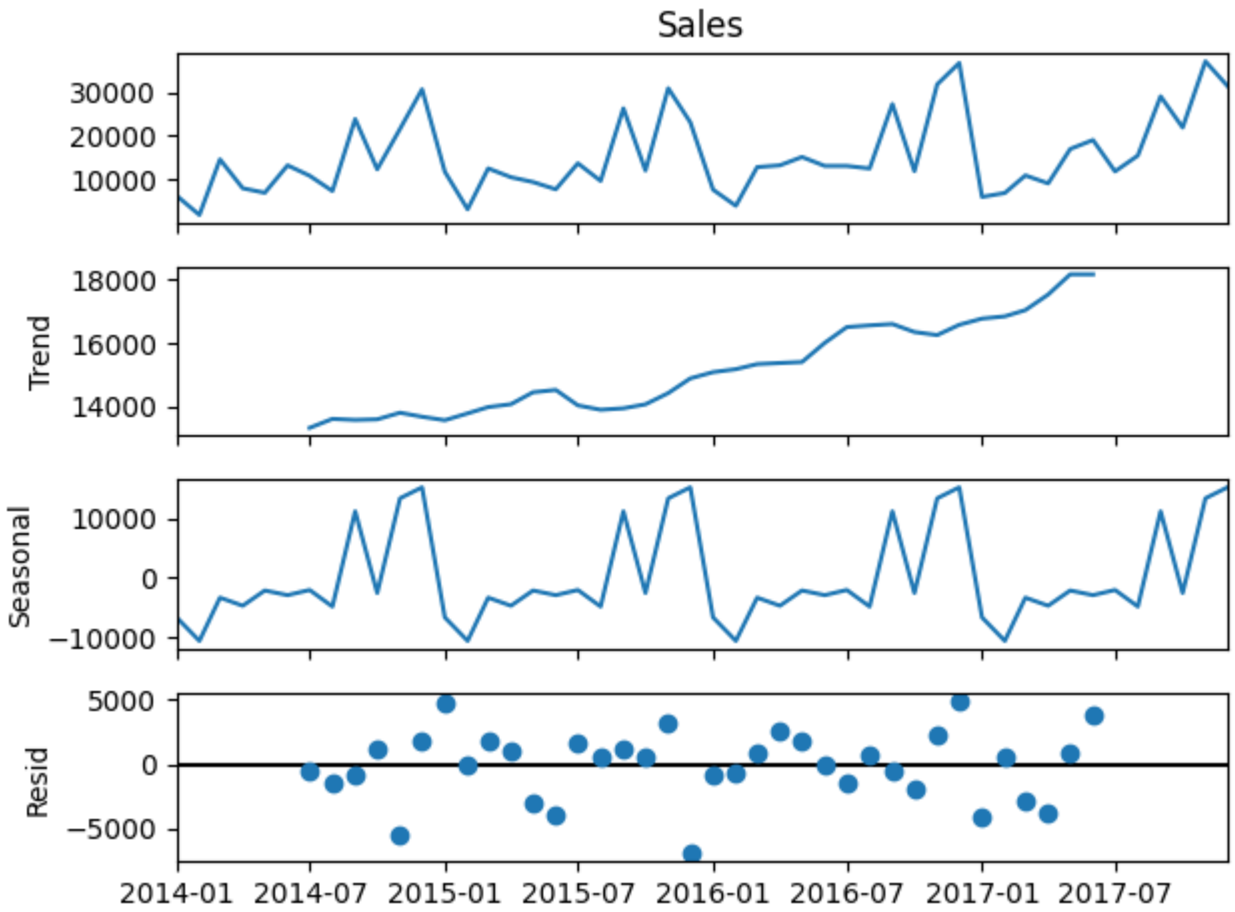
Out[100…
```
<Axes: xlabel='Order Date'>
```



In [101…
```
result = seasonal_decompose(df['Sales'], model='add')
result.plot()
```

Out[101…


Sales
30000
20000
10000
Trend
18000
16000
14000
Seasonal
10000
0
−10000
Resid
5000
0
−5000
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07



In [102…

```
auto_arima(df['Sales'],seasonal=True,m=12).summary()
```

Out[102…

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	48
Model:	SARIMAX(2, 1, 0, 12)	**Log Likelihood**	-349.872
Date:	Thu, 21 Sep 2023	**AIC**	707.744
Time:	00:21:27	**BIC**	714.078
Sample:	01-01-2014	**HQIC**	709.955
	- 12-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	1616.5115	1257.009	1.286	0.198	-847.181	4080.205
ar.S.L12	-0.1196	0.087	-1.378	0.168	-0.290	0.051
ar.S.L24	0.1214	0.098	1.240	0.215	-0.070	0.313
sigma2	1.862e+07	0.036	5.1e+08	0.000	1.86e+07	1.86e+07

Ljung-Box (L1) (Q):	0.71	**Jarque-Bera (JB):**	1.87
Prob(Q):	0.40	**Prob(JB):**	0.39
Heteroskedasticity (H):	0.96	**Skew:**	0.55
Prob(H) (two-sided):	0.95	**Kurtosis:**	2.84

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 4.96e+24. Standard errors may be unstable.

In [103…

```
train = df.iloc[:len(df)-6]
test = df.iloc[len(df)-6:]
model = SARIMAX(train['Sales'],order=(0,1,3),seasonal_order=(2,1,0,12))
results = model.fit()
results.summary()
```

Out[103…

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	42
Model:	SARIMAX(0, 1, 3)x(2, 1, [], 12)	**Log Likelihood**	-289.730
Date:	Thu, 21 Sep 2023	**AIC**	591.461
Time:	00:21:48	**BIC**	599.664
Sample:	01-01-2014	**HQIC**	594.030
	- 06-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.3976	0.146	-2.722	0.006	-0.684	-0.111
ma.L2	-0.1639	0.156	-1.053	0.292	-0.469	0.141
ma.L3	-0.1625	0.167	-0.972	0.331	-0.490	0.165
ar.S.L12	-0.1664	0.118	-1.404	0.160	-0.399	0.066
ar.S.L24	0.2224	0.130	1.710	0.087	-0.033	0.477
sigma2	2.461e+07	5.24e-10	4.7e+16	0.000	2.46e+07	2.46e+07

Ljung-Box (L1) (Q):	2.15	**Jarque-Bera (JB):**	0.87
Prob(Q):	0.14	**Prob(JB):**	0.65
Heteroskedasticity (H):	0.75	**Skew:**	0.17
Prob(H) (two-sided):	0.65	**Kurtosis:**	2.22

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 3.91e+32. Standard errors may be unstable.

In [104…

```
start=len(train)
end=len(train)+len(test)-1
predictions = results.predict(start=start, end=end, dynamic=False, typ='levels').rename('SARIMA(0,1,3)(1,0,1,12) Predictions')
```

In [105…

```
for i in range(len(predictions)):
    print(f"predicted={predictions[i]:<11.10}, expected={test['Sales'][i]}")
```

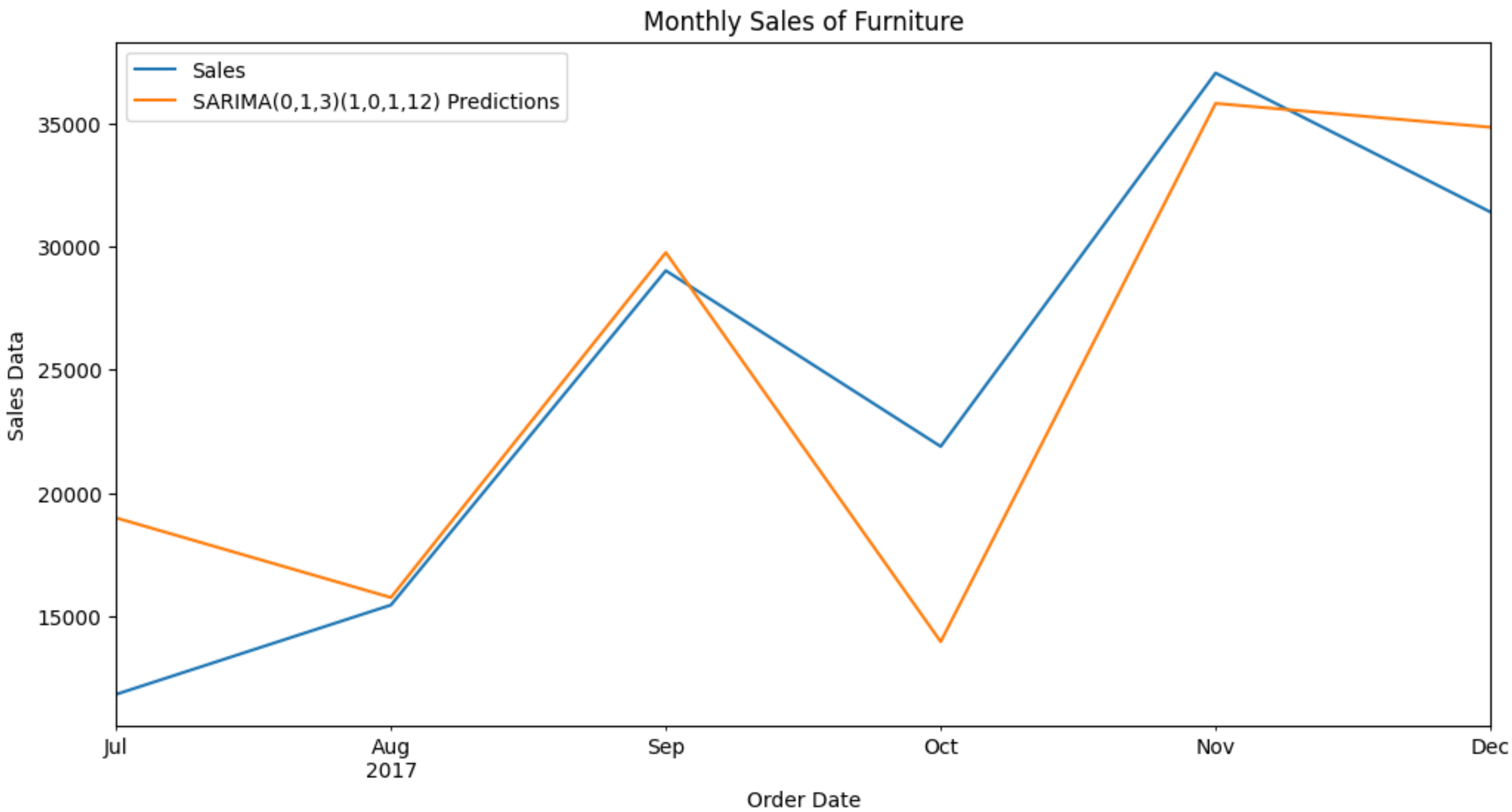
```
predicted=18982.66877, expected=11813.021999999999
predicted=15746.93545, expected=15441.874
predicted=29761.95657, expected=29028.206000000002
predicted=13953.797  , expected=21884.0682
predicted=35822.90086, expected=37056.715
predicted=34853.24127, expected=31407.4668
```

In [106…

```
title = 'Monthly Sales of Furniture'
ylabel='Sales Data'
xlabel='Order Date'

ax = test['Sales'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[106…

```
[Text(0.5, 0, 'Order Date'), Text(0, 0.5, 'Sales Data')]
```



In [107…

```
from sklearn.metrics import mean_squared_error
error = mean_squared_error(test['Sales'], predictions)
print(f'SARIMA(2,0,1,12) MSE Error: {error:11.10}')
```

```
SARIMA(2,0,1,12) MSE Error: 21386691.26
```

In [108…

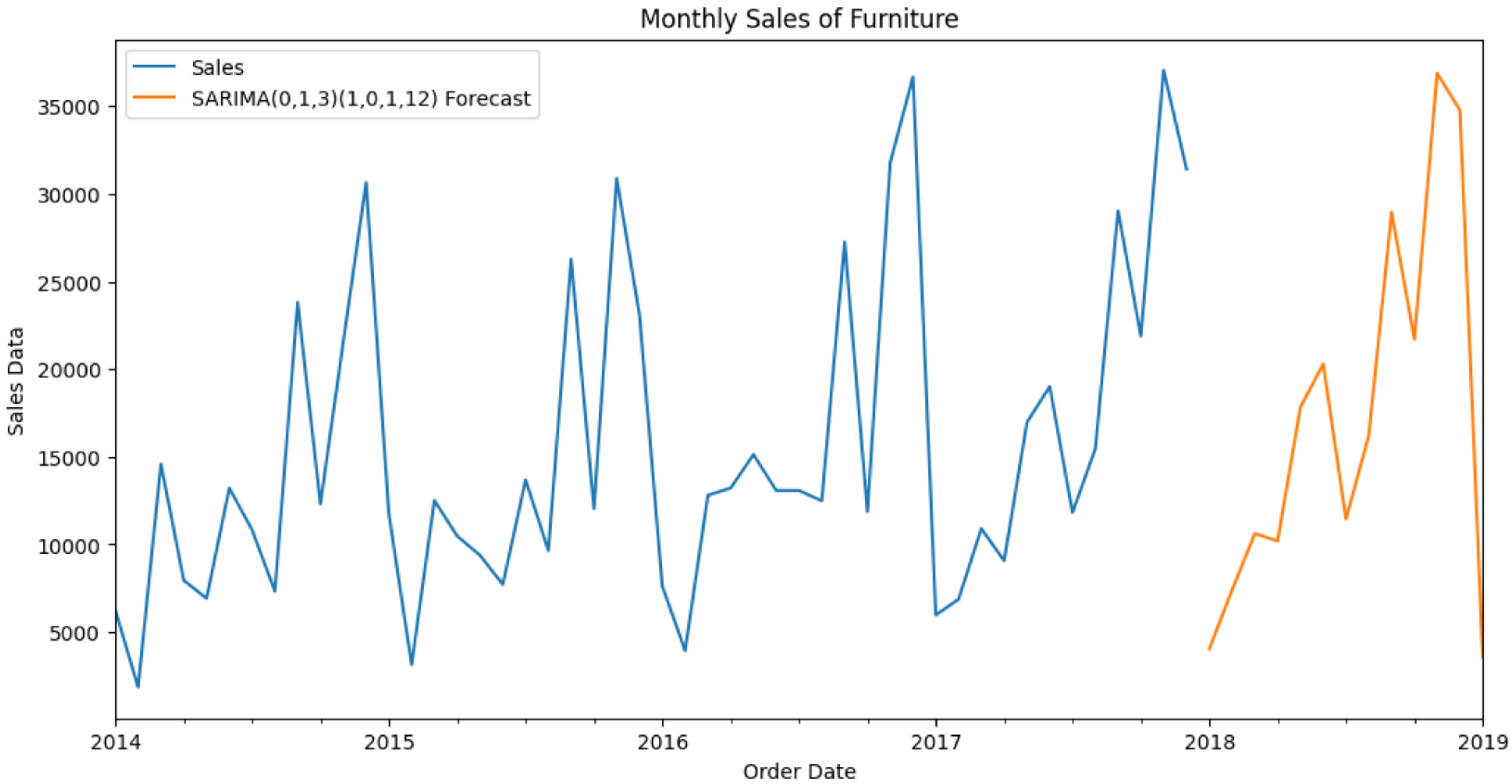
```
model = SARIMAX(df['Sales'],order=(1,0,2),seasonal_order=(2,1,0,12))
results = model.fit()
fcast = results.predict(len(df),len(df)+12,typ='levels').rename('SARIMA(0,1,3)(1,0,1,12) Forecast')
```

In [109…

```
title = 'Monthly Sales of Furniture'
ylabel='Sales Data'
xlabel='Order Date'

ax = df['Sales'].plot(legend=True,figsize=(12,6),title=title)
fcast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[109…

```
[Text(0.5, 0, 'Order Date'), Text(0, 0.5, 'Sales Data')]
```



In [110…

```
import pandas as pd
import numpy as np
%matplotlib inline

from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima import auto_arima
import warnings
warnings.filterwarnings("ignore")
```

In [111…

```
df = pd.read_excel('C://Users//saswa//OneDrive//Desktop//Pinaki-Time-series-forecasting//Superstore_Sales_Records.xls', index_
df = df[df['Category']=='Furniture']
df = df.groupby(by='Order Date').agg({'Sales':sum, 'Quantity':sum})
df.sort_index(inplace=True)
df.head(24)
```

Out[111…

	Sales	Quantity
Order Date		
2014-01-06	2573.820	9
2014-01-07	76.728	3
2014-01-10	51.940	1
2014-01-11	9.940	2
2014-01-13	879.939	9
2014-01-14	61.960	4
2014-01-16	127.104	6
2014-01-19	181.470	5
2014-01-20	1413.510	15
2014-01-21	25.248	3
2014-01-26	217.200	8
2014-01-27	333.000	3
2014-01-31	290.666	2
2014-02-08	14.560	2
2014-02-11	1650.050	10
2014-02-12	129.568	2
2014-02-18	25.160	5
2014-02-20	20.320	4
2014-03-01	1893.995	23
2014-03-03	928.802	8
2014-03-07	966.984	9

	Sales	Quantity
Order Date		
2014-03-11	8.320	5
2014-03-14	1139.920	4
2014-03-15	45.696	3

In [112…

```
df = df.resample('W').sum()

df.head()
```

Out[112…

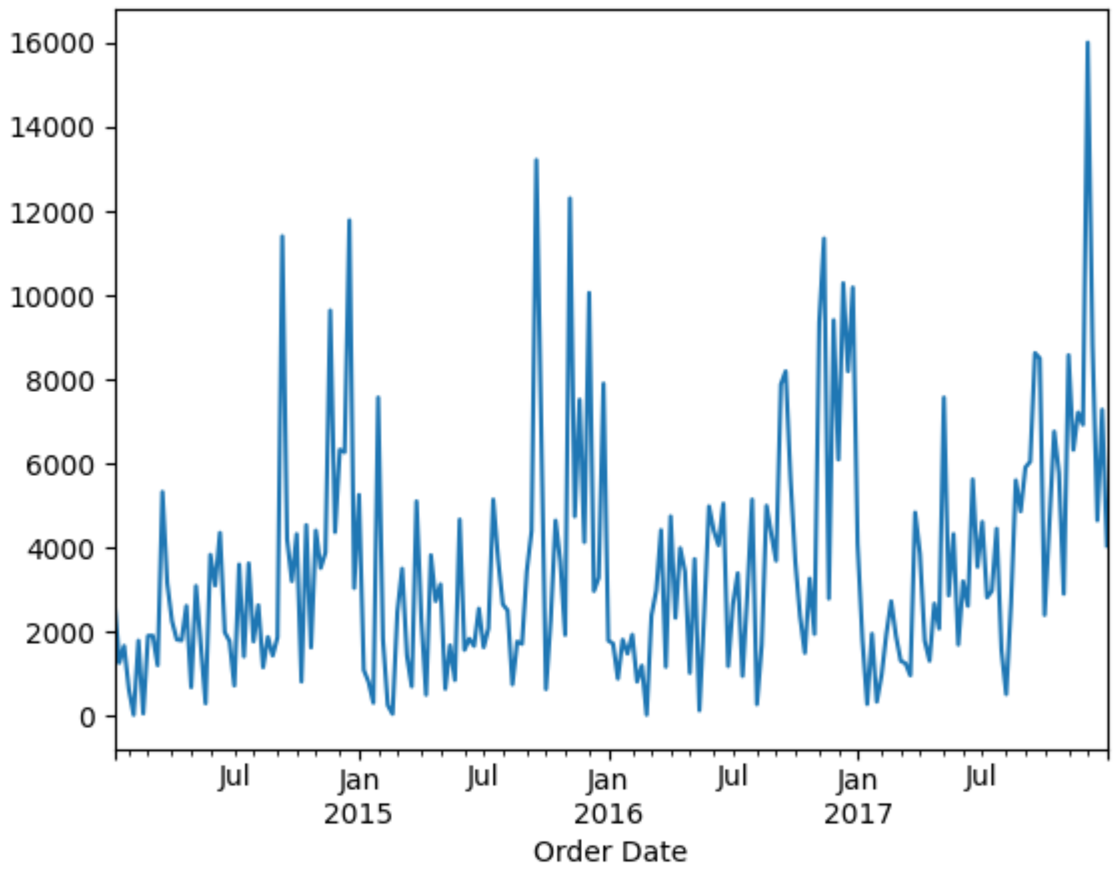
	Sales	Quantity
Order Date		
2014-01-12	2712.428	15
2014-01-19	1250.473	24
2014-01-26	1655.958	26
2014-02-02	623.666	5
2014-02-09	14.560	2

In [113…

```
df['Sales'].plot()
```

Out[113…

```
<Axes: xlabel='Order Date'>
```

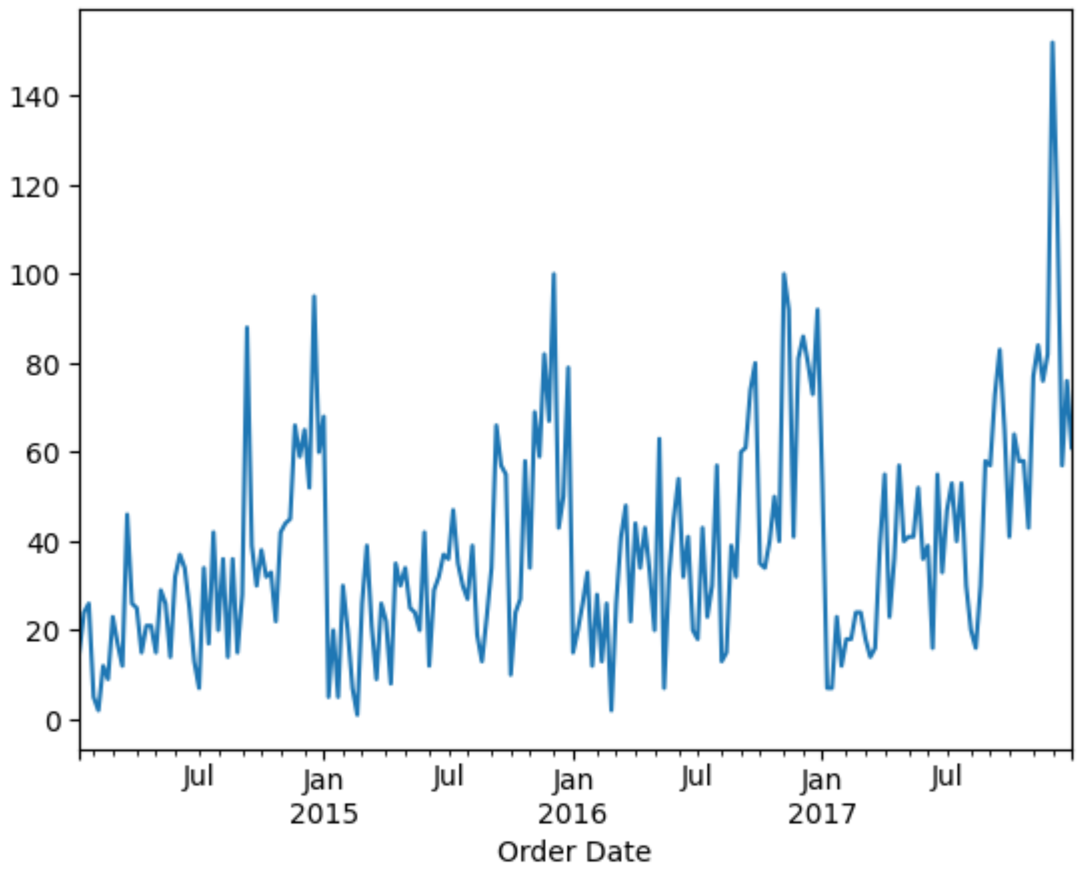


In [114…

```
df['Quantity'].plot()
```

Out[114…

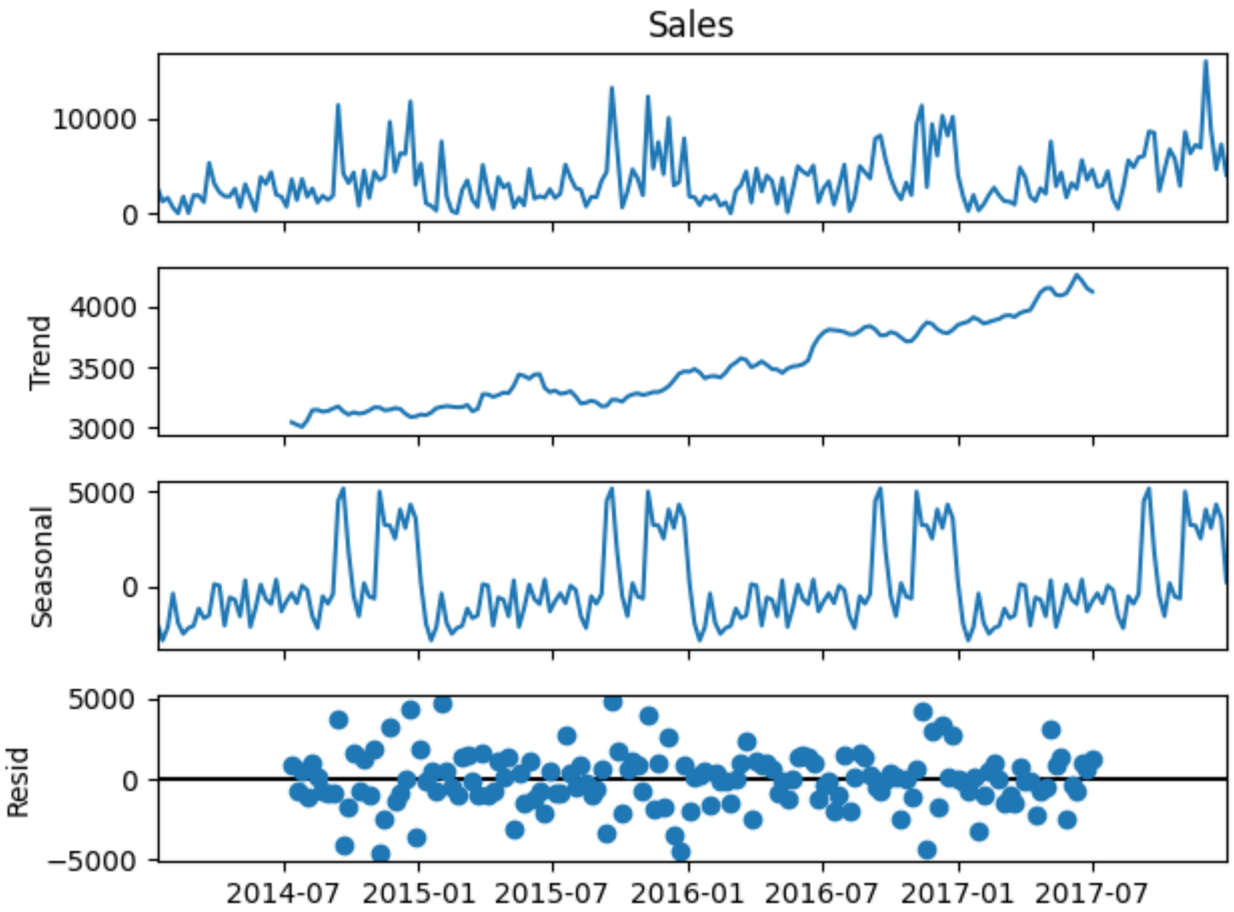
```
<Axes: xlabel='Order Date'>
```

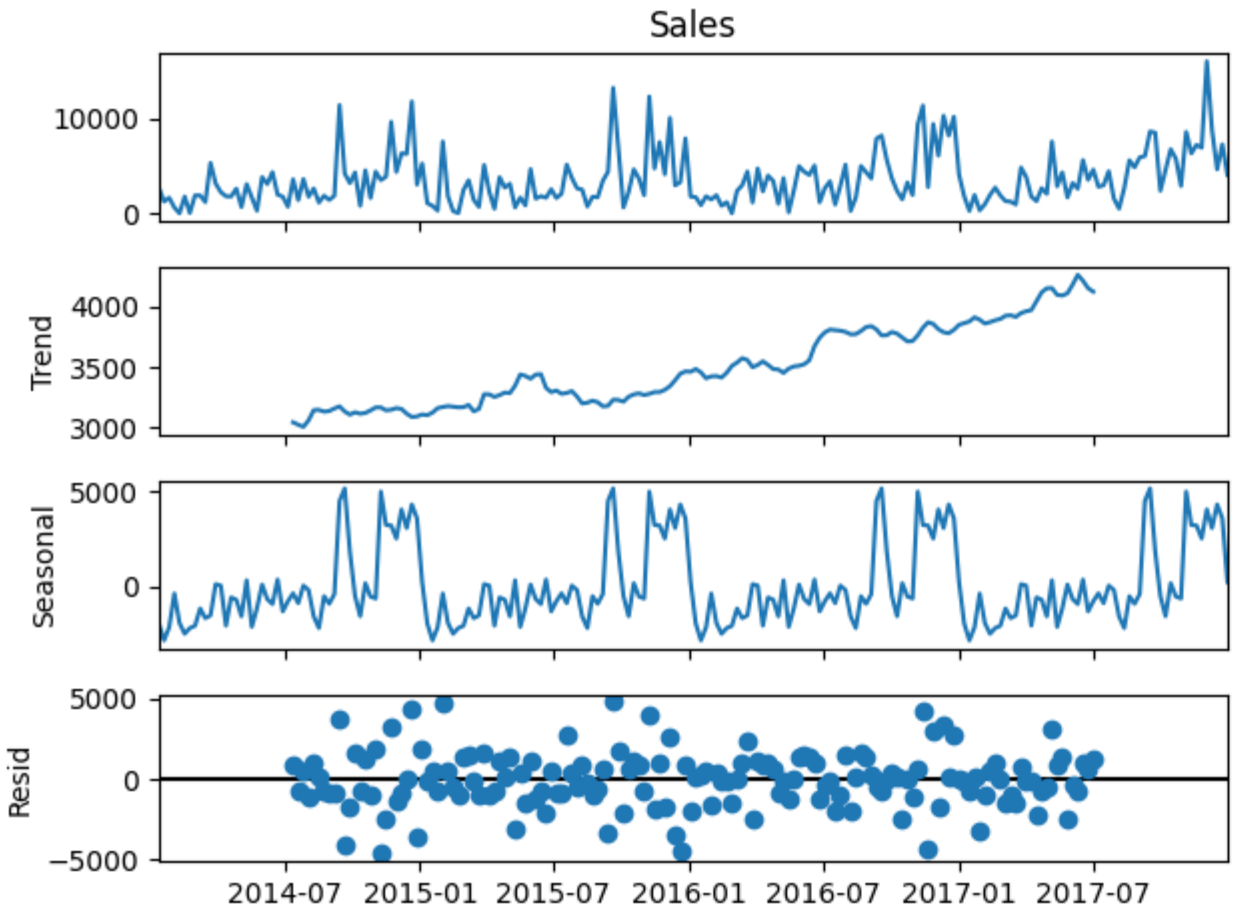


In [115…

```
result_sales = seasonal_decompose(df['Sales'])

result_sales.plot()
```

Out[115…

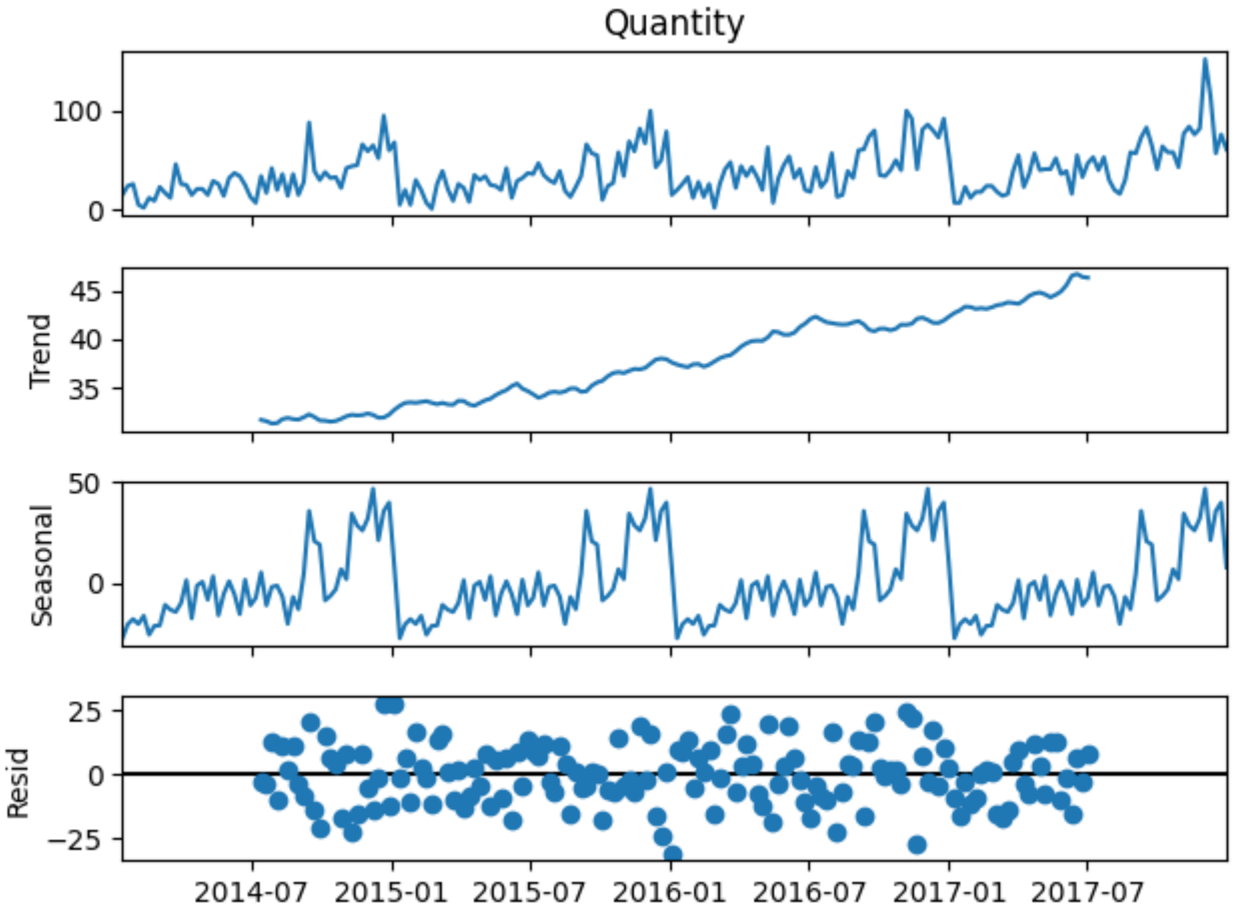

Sales
10000
0
Trend
4000
3500
3000
Seasonal
5000
0
Resid
5000
0
−5000
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07

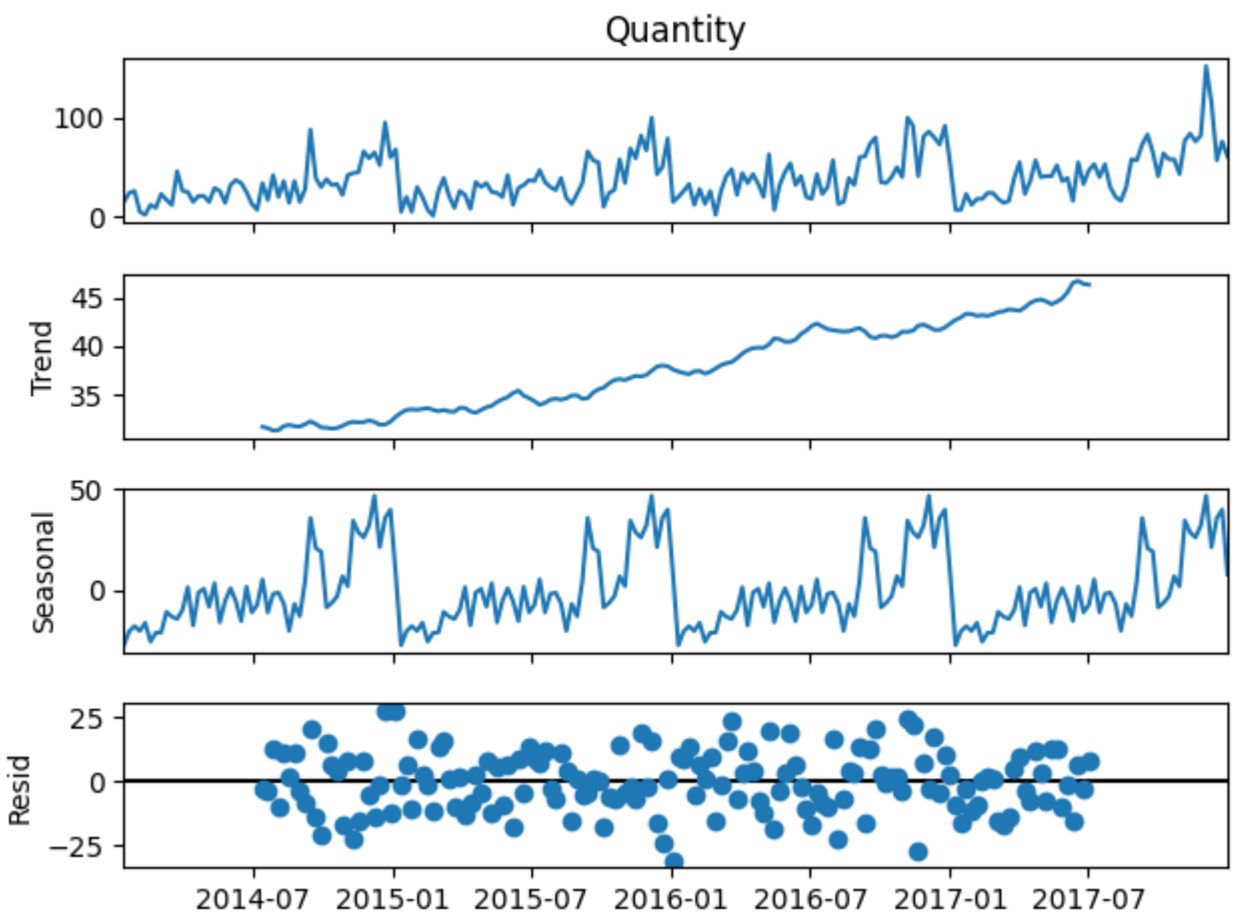


In [116…

```
result_sales = seasonal_decompose(df['Quantity'])

result_sales.plot()
```

Out[116…


Quantity
100
0
Trend
45
40
35
Seasonal
50
0
Resid
25
0
−25
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07



In [117…

```
from statsmodels.tsa.stattools import adfuller

def adf_test(series,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series.dropna(),autolag='AIC') # .dropna() handles differenced data

    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)
```

```
    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    print(out.to_string())          # .to_string() removes the line "dtype: float64"

    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has no unit root and is stationary")
    else:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has a unit root and is non-stationary")
adf_test(df['Sales'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -3.444275
p-value                   0.009539
# lags used               6.000000
# observations          201.000000
critical value (1%)      -3.463309
critical value (5%)      -2.876029
critical value (10%)     -2.574493
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [118…

```
adf_test(df['Quantity'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -4.457632
p-value                   0.000234
# lags used               4.000000
# observations          203.000000
critical value (1%)      -3.462980
critical value (5%)      -2.875885
critical value (10%)     -2.574416
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [119…

```
auto_arima(df['Sales'],bseasonal=True, m=12).summary()
```

Out[119…

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	208
Model:	SARIMAX(1, 1, 2)	**Log Likelihood**	-1917.198
Date:	Thu, 21 Sep 2023	**AIC**	3842.396
Time:	00:26:55	**BIC**	3855.727
Sample:	01-12-2014	**HQIC**	3847.787
	- 12-31-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7864	0.166	4.731	0.000	0.461	1.112
ma.L1	-1.4945	0.201	-7.446	0.000	-1.888	-1.101
ma.L2	0.5044	0.183	2.763	0.006	0.147	0.862
sigma2	6.442e+06	4.47e-08	1.44e+14	0.000	6.44e+06	6.44e+06

Ljung-Box (L1) (Q):	0.12	**Jarque-Bera (JB):**	130.12
Prob(Q):	0.73	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.43	**Skew:**	1.32
Prob(H) (two-sided):	0.14	**Kurtosis:**	5.85

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near-singular, with condition number 7.8e+29. Standard errors may be unstable.

In [120…
```
len(df)
```

Out[120…
```
208
```

In [121…
```
train = df.iloc[:170]
test = df.iloc[170:]
```

In [122…
```
model = SARIMAX(train['Sales'],order=(1,1,2) ,enforce_invertibility=False)
results = model.fit()
results.summary()
```

Out[122...

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	170
Model:	SARIMAX(1, 1, 2)	**Log Likelihood**	-1573.035
Date:	Thu, 21 Sep 2023	**AIC**	3154.069
Time:	00:27:21	**BIC**	3166.589
Sample:	01-12-2014	**HQIC**	3159.150
	- 04-09-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7910	0.528	-1.499	0.134	-1.825	0.243
ma.L1	-0.2794	0.479	-0.583	0.560	-1.218	0.660
ma.L2	-0.9240	0.525	-1.759	0.079	-1.953	0.105
sigma2	6.972e+06	9.33e+05	7.472	0.000	5.14e+06	8.8e+06

Ljung-Box (L1) (Q):	5.41	**Jarque-Bera (JB):**	70.94
Prob(Q):	0.02	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.23	**Skew:**	1.14
Prob(H) (two-sided):	0.44	**Kurtosis:**	5.21

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

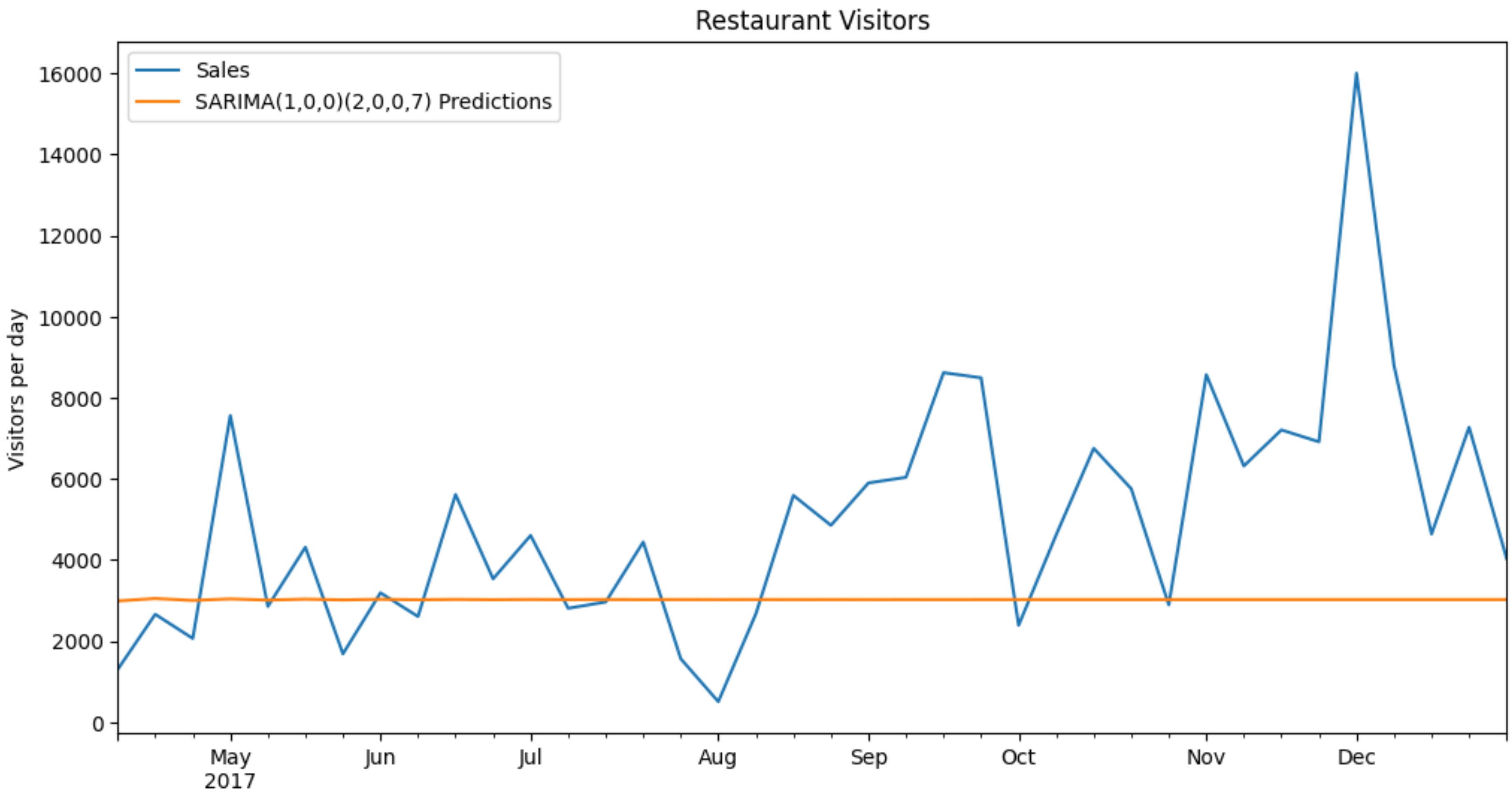
In [123...

```
start=len(train)
end=len(train)+len(test)-1
```

```
predictions = results.predict(start=start, end=end, dynamic=False).rename('SARIMA(1,0,0)(2,0,0,7) Predictions')
title='Restaurant Visitors'
ylabel='Visitors per day'
xlabel=''

ax = test['Sales'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[123…

```
[Text(0.5, 0, ''), Text(0, 0.5, 'Visitors per day')]
```



In [124…

```
from statsmodels.tools.eval_measures import mse,rmse

error1 = mse(test['Sales'], predictions)
error2 = rmse(test['Sales'], predictions)

print(f'SARIMA(1,0,0)(2,0,0,7) MSE Error: {error1:11.10}')
print(f'SARIMA(1,0,0)(2,0,0,7) RMSE Error: {error2:11.10}')
```

```
SARIMA(1,0,0)(2,0,0,7) MSE Error: 12077452.14
SARIMA(1,0,0)(2,0,0,7) RMSE Error: 3475.262888
```

In [125…

```
df.columns
```

Out[125…

```
Index(['Sales', 'Quantity'], dtype='object')
```

In [126…

```
model = SARIMAX(train['Sales'],exog=train['Quantity'],order=(1,0,0),seasonal_order=(2,0,0,7),enforce_invertibility=False)
results = model.fit()
results.summary()
```

Out[126...

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	170
Model:	SARIMAX(1, 0, 0)x(2, 0, 0, 7)	**Log Likelihood**	-1466.114
Date:	Thu, 21 Sep 2023	**AIC**	2942.229
Time:	00:28:10	**BIC**	2957.908
Sample:	01-12-2014	**HQIC**	2948.591
	- 04-09-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Quantity	97.6600	2.403	40.643	0.000	92.950	102.369
ar.L1	-0.0783	0.072	-1.086	0.277	-0.220	0.063
ar.S.L7	0.2222	0.057	3.874	0.000	0.110	0.335
ar.S.L14	-0.0828	0.094	-0.879	0.379	-0.267	0.102
sigma2	1.903e+06	1.58e+05	12.024	0.000	1.59e+06	2.21e+06

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	295.39
Prob(Q):	0.88	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.16	**Skew:**	1.56
Prob(H) (two-sided):	0.58	**Kurtosis:**	8.66

Warnings:

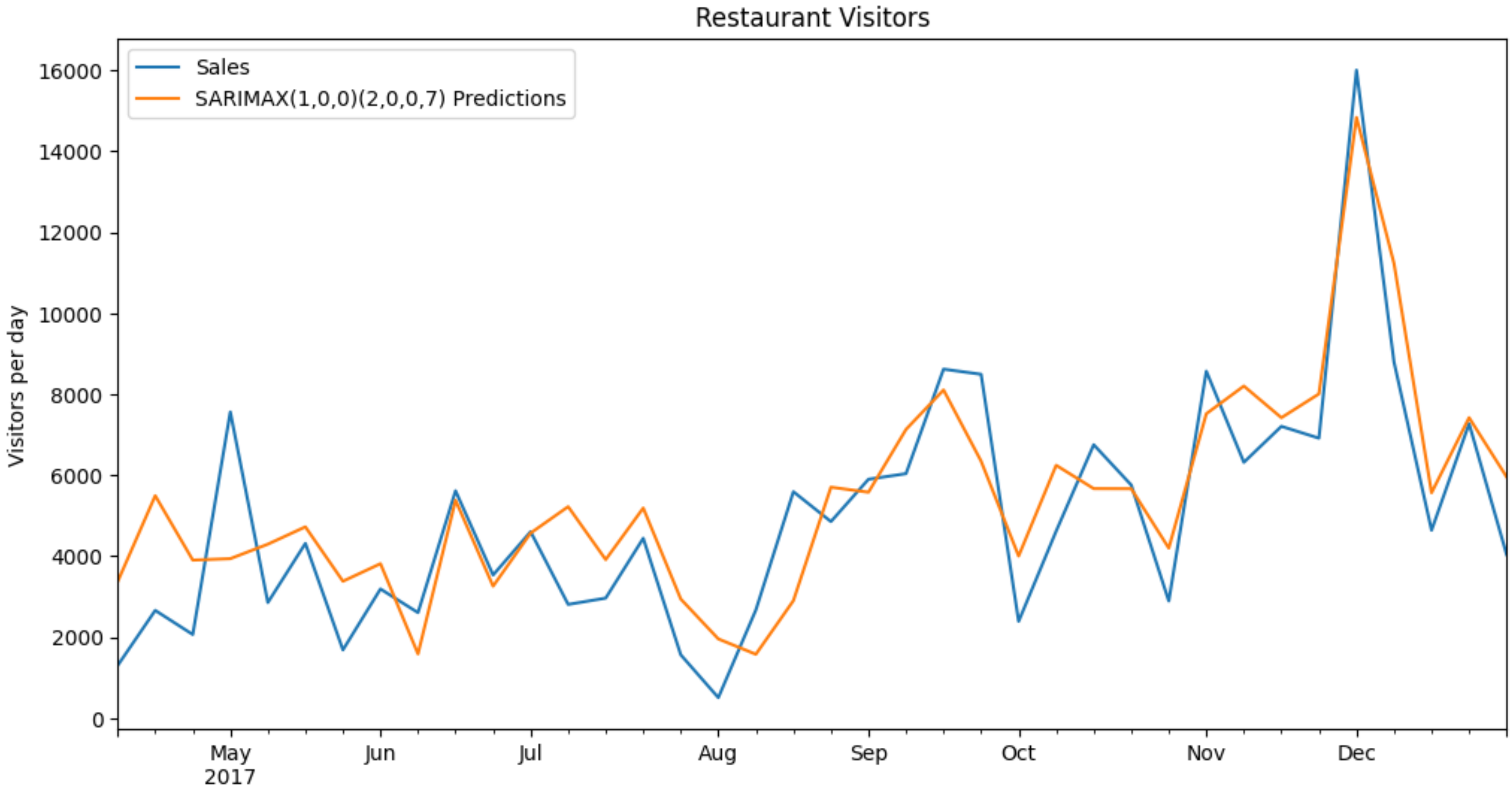
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [127…

```
start=len(train)
end=len(train)+len(test)-1
exog_forecast = test[['Quantity']]
predictions = results.predict(start=start, end=end, exog=exog_forecast).rename('SARIMAX(1,0,0)(2,0,0,7) Predictions')
title='Restaurant Visitors'
ylabel='Visitors per day'
xlabel=''

ax = test['Sales'].plot(legend=True,figsize=(12,6),title=title)
predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[127…

```
[Text(0.5, 0, ''), Text(0, 0.5, 'Visitors per day')]
```



In [128…

```
print(f'SARIMA(1,0,0)(2,0,0,7) MSE Error: {error1:11.10}')
print(f'SARIMA(1,0,0)(2,0,0,7) RMSE Error: {error2:11.10}')
print()

error1x = mse(test['Sales'], predictions)
error2x = rmse(test['Sales'], predictions)

print(f'SARIMAX(1,0,0)(2,0,0,7) MSE Error: {error1x:11.10}')
print(f'SARIMAX(1,0,0)(2,0,0,7) RMSE Error: {error2x:11.10}')
```

```
SARIMA(1,0,0)(2,0,0,7) MSE Error: 12077452.14
SARIMA(1,0,0)(2,0,0,7) RMSE Error: 3475.262888

SARIMAX(1,0,0)(2,0,0,7) MSE Error: 2317778.515
SARIMAX(1,0,0)(2,0,0,7) RMSE Error: 1522.425208
```

In [129…

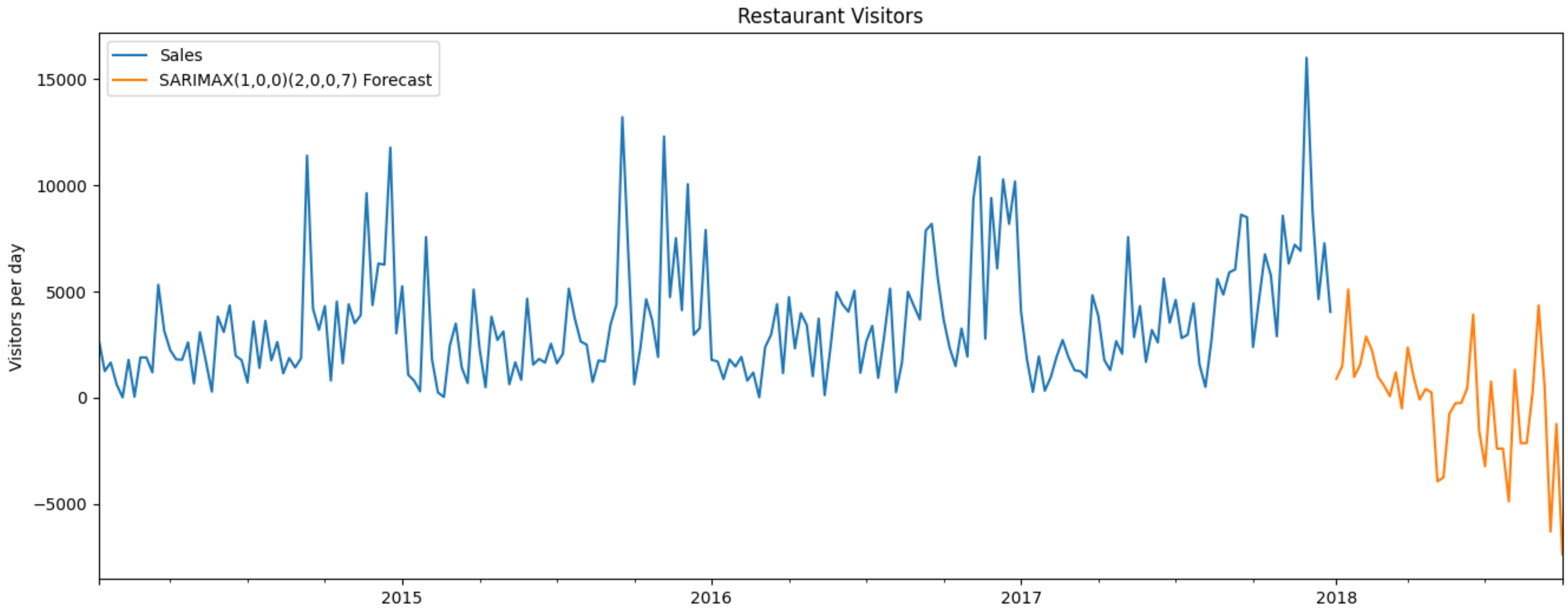
```
model = SARIMAX(df['Sales'],exog=df['Quantity'],order=(1,2,1),seasonal_order=(1,2,2,7),enforce_invertibility=False)
results = model.fit()
exog_forecast = df[169:][['Quantity']]
fcast = results.predict(len(df),len(df)+38,exog=exog_forecast).rename('SARIMAX(1,0,0)(2,0,0,7) Forecast')
```

In [130…

```
title='Restaurant Visitors'
ylabel='Visitors per day'
xlabel=''

ax = df['Sales'].plot(legend=True,figsize=(16,6),title=title)
fcast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
```

Out[130…

```
[Text(0.5, 0, ''), Text(0, 0.5, 'Visitors per day')]
```


Restaurant Visitors
Sales
SARIMAX(1,0,0)(2,0,0,7) Forecast
Visitors per day
15000
10000
5000
0
−5000
2015
2016
2017
2018

In [ ]: