In [2]:
```python
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

In [3]:
```python
df = pd.read_excel("Sample - Superstore.xls")
df
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	0.00
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	0.00
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	0.45
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2	0.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9989	9990	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami	...	33180	South	FUR-FU-10001889	Furniture	Furnishings	Ultra Door Pull Handle	25.2480	3	0.20
9990	9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	FUR-FU-10000747	Furniture	Furnishings	Tenex B1-RE Series Chair Mats for Low Pile Car...	91.9600	2	0.00
9991	9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	TEC-PH-10003645	Technology	Phones	Aastra 57i VoIP phone	258.5760	2	0.20
9992	9993	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	OFF-PA-10004041	Office Supplies	Paper	It's Hot Message Books with Stickers, 2 3/4" x 5"	29.6000	4	0.00
9993	9994	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster	...	92683	West	OFF-AP-10002684	Office Supplies	Appliances	Acco 7-Outlet Masterpiece Power Center, Wihtou...	243.1600	2	0.00

9994 rows × 21 columns

In [4]:
```python
furniture = df.loc[df['Category'] == 'Furniture']
furniture['Order Date'].min(), furniture['Order Date'].max()
```

Out[4]:
```
(Timestamp('2014-01-06 00:00:00'), Timestamp('2017-12-30 00:00:00'))
```

In [5]:
```python
cols = ['Row ID', 'Order ID', 'Ship Date', 'Ship Mode', 'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region', 'Produ
furniture.drop(cols, axis=1, inplace=True)
furniture = furniture.sort_values('Order Date')

furniture.isnull().sum()
```

Out[5]:
```
Order Date    0
Sales         0
dtype: int64
```

In [6]:
```python
furniture = furniture.groupby('Order Date')['Sales'].sum().reset_index()
furniture
```

Out[6]:

	Order Date	Sales
0	2014-01-06	2573.8200
1	2014-01-07	76.7280
2	2014-01-10	51.9400
3	2014-01-11	9.9400
4	2014-01-13	879.9390
...	...	...
884	2017-12-24	1393.4940
885	2017-12-25	832.4540
886	2017-12-28	551.2568
887	2017-12-29	2330.7180
888	2017-12-30	323.1360

889 rows × 2 columns

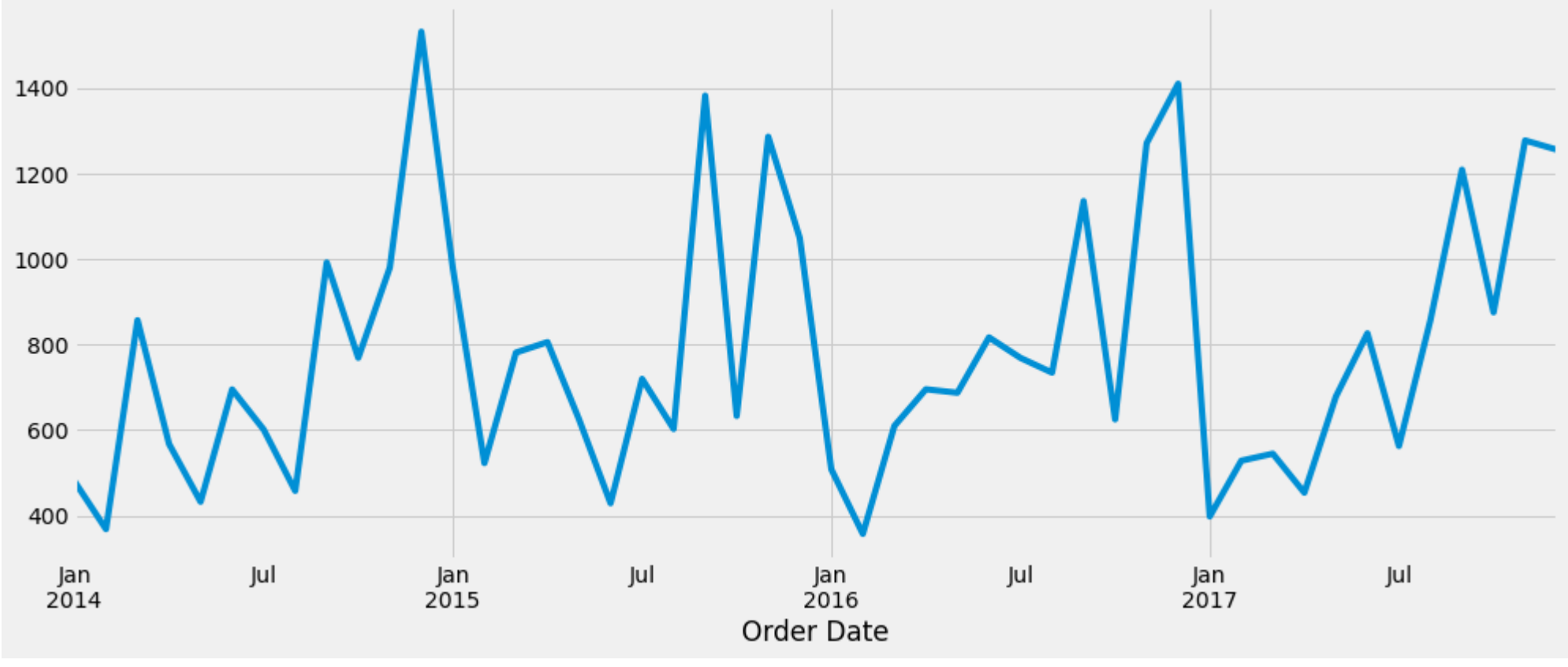
In [7]:
```python
furniture = furniture.set_index('Order Date')
furniture.index
```

Out[7]:
```
DatetimeIndex(['2014-01-06', '2014-01-07', '2014-01-10', '2014-01-11',
               '2014-01-13', '2014-01-14', '2014-01-16', '2014-01-19',
               '2014-01-20', '2014-01-21',
               ...
               '2017-12-18', '2017-12-19', '2017-12-21', '2017-12-22',
               '2017-12-23', '2017-12-24', '2017-12-25', '2017-12-28',
               '2017-12-29', '2017-12-30'],
              dtype='datetime64[ns]', name='Order Date', length=889, freq=None)
```

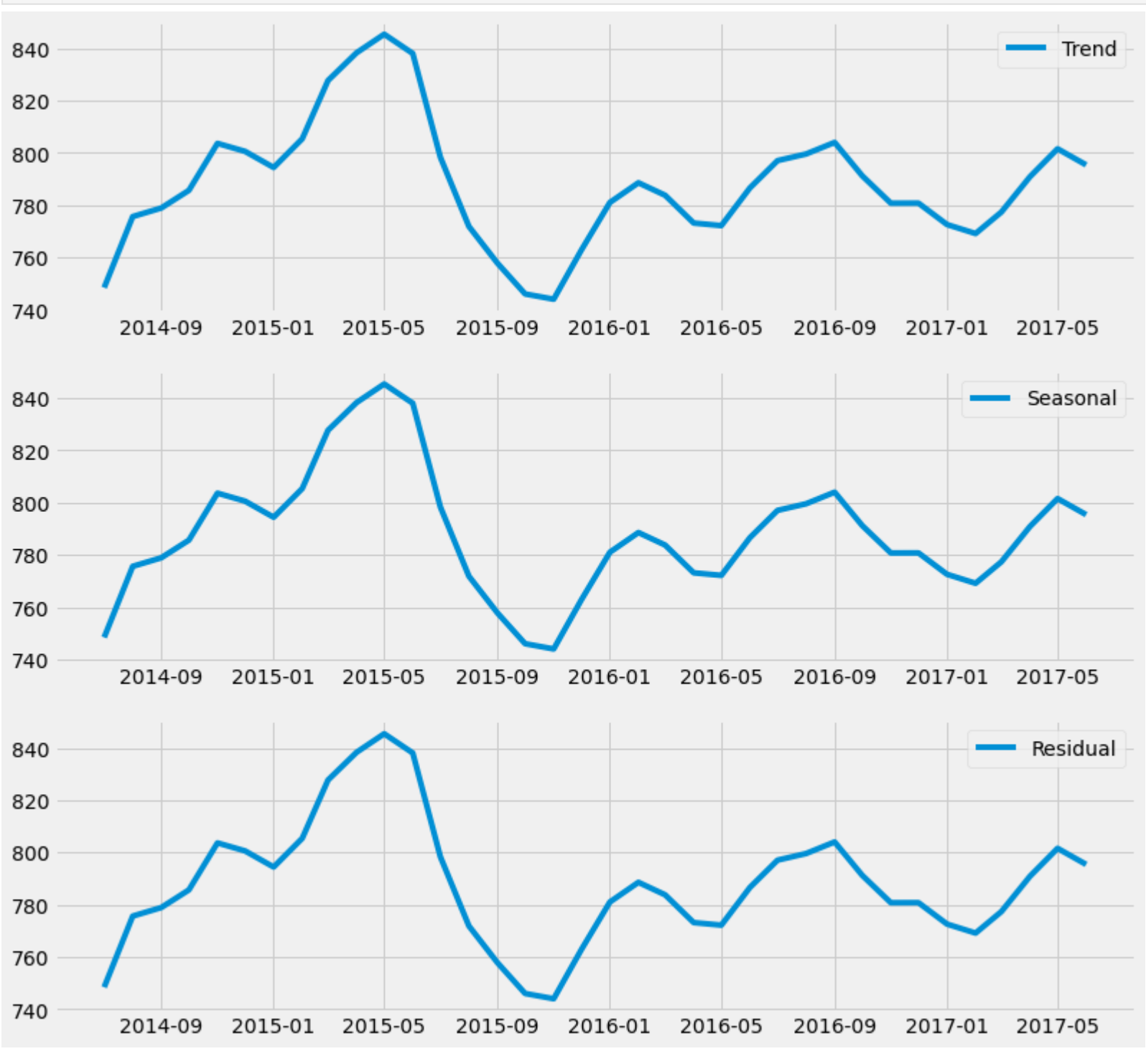
In [8]:
```python
y_useful = furniture['Sales'].resample('MS').mean()
y_useful['2015':]
```

Out[8]:
```
Order Date
2015-01-01     978.328467
2015-02-01     522.395667
2015-03-01     781.236437
2015-04-01     805.822962
2015-05-01     624.996700
2015-06-01     428.565500
2015-07-01     719.706316
2015-08-01     602.412012
2015-09-01    1382.790684
2015-10-01     632.980184
2015-11-01    1286.701354
2015-12-01    1049.355418
2016-01-01     508.182867
2016-02-01     356.868273
2016-03-01     609.575810
2016-04-01     695.373158
2016-05-01     687.265227
2016-06-01     816.910750
2016-07-01     768.736412
2016-08-01     734.307782
2016-09-01    1135.953371
2016-10-01     624.872474
2016-11-01    1271.345152
2016-12-01    1410.719808
2017-01-01     397.602133
2017-02-01     528.179800
2017-03-01     544.672240
2017-04-01     453.297905
2017-05-01     678.302328
2017-06-01     826.460291
2017-07-01     562.524857
2017-08-01     857.881889
2017-09-01    1209.508583
2017-10-01     875.362728
2017-11-01    1277.817759
2017-12-01    1256.298672
Freq: MS, Name: Sales, dtype: float64
```

In [9]:
```python
y_useful.plot(figsize=(15, 6))
plt.show()
```



In [10]:
```python
decomposition = sm.tsa.seasonal_decompose(y_useful, model='additive')
trend = decomposition.trend
plt.figure(figsize=(12,16))
plt.subplot(411)
plt.plot(trend,label="Trend")
plt.legend()
Seasonal = decomposition.trend
plt.subplot(412)
plt.plot(trend,label="Seasonal")
plt.legend()
Residual = decomposition.trend
plt.subplot(413)
plt.plot(trend,label="Residual")
plt.legend()
plt.show()
```

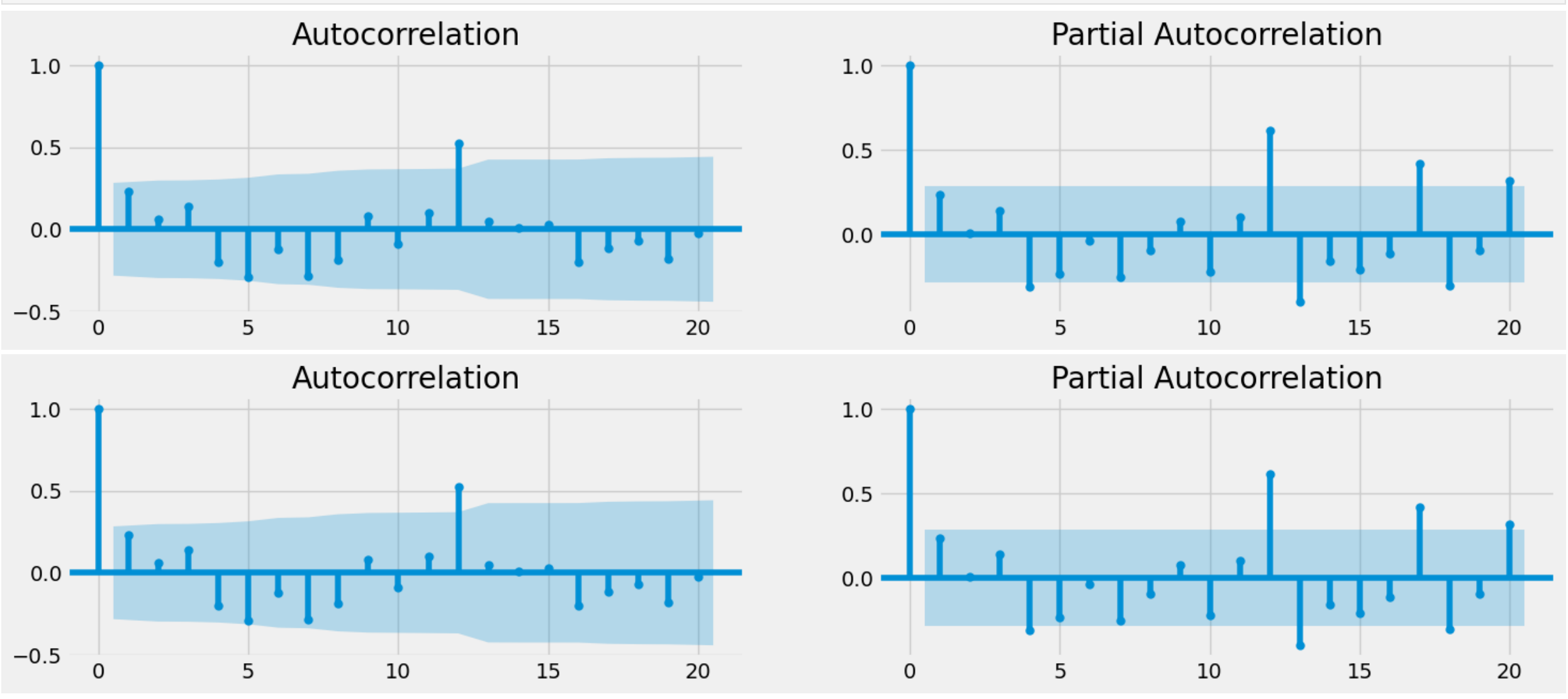


In [11]:
```python
from statsmodels.tsa.stattools import adfuller
y_test = adfuller(y_useful,autolag="AIC")
y_test
```

Out[11]:
```
(-5.191070187339267,
 9.168756655665896e-06,
 10,
 37,
 {'1%': -3.6209175221605827,
  '5%': -2.9435394610388332,
  '10%': -2.6104002410518627},
 521.9616303121272)
```

In [12]:
```python
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# fig, (ax1,ax2) = plt.subplots(1,2,figsize=(10,5))
# fig = sm.graphics.tsa.plot_acf(y_useful.values.squeeze(),lags=20,ax=ax1)
# ax1.axhline(y==-1.96/np.sqrt(len(y_useful)))
# ax1.axhline(y==1.96/np.sqrt(len(y_useful)))
# fig = sm.graphics.tsa.plot_pacf(y_useful,lags=20,ax=ax2)
# ax2.axhline(y==-1.96/np.sqrt(len(y_useful)))
# ax2.axhline(y==1.96/np.sqrt(len(y_useful)))
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(y_useful, lags=20, ax=axes[0])
plot_pacf(y_useful, lags=20, ax=axes[1])
```

Out[12]:



In [13]:
```python
model = sm.tsa.ARIMA(y_useful,order=(2,0,2))
result = model.fit(disp=-1)
result
```

Out[13]:
```
<statsmodels.tsa.arima_model.ARMAResultsWrapper at 0x211cf4494f0>
```

In [14]:
```python
print(result.aic)
print(result.bic)
```

```
693.5725330243508
704.7997390897982
```

In [15]:
```python
from sklearn import metrics
results_pred = result.predict()
metrics.mean_absolute_error(y_useful.values,results_pred.values)
```

Out[15]:
```
231.642903843485
```

In [16]:
```python
result.summary()
```

Out[16]:

ARMA Model Results

Dep. Variable:	Sales	**No. Observations:**	48
Model:	ARMA(2, 2)	**Log Likelihood**	-340.786
Method:	css-mle	**S.D. of innovations**	292.913
Date:	Sun, 25 Apr 2021	**AIC**	693.573
Time:	10:30:04	**BIC**	704.800
Sample:	01-01-2014	**HQIC**	697.815
	- 12-01-2017		

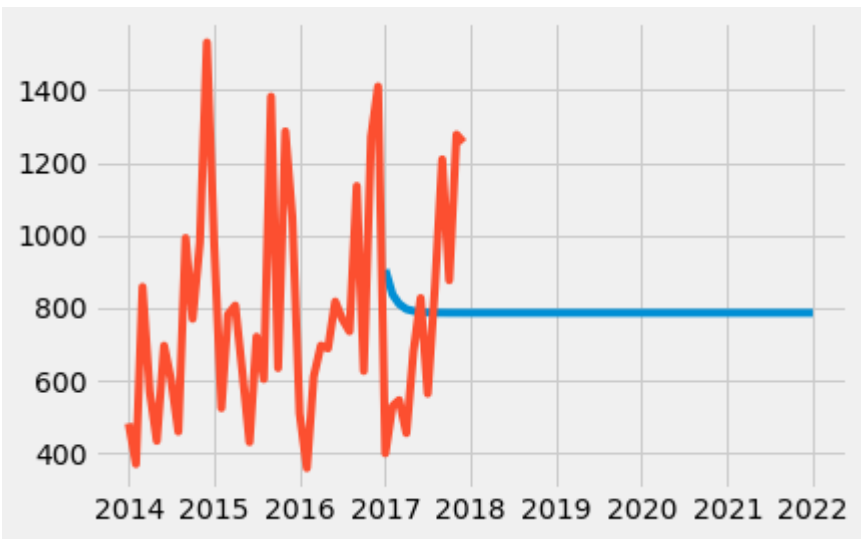
	coef	std err	z	P>|z|	[0.025	0.975]
const	785.4806	57.423	13.679	0.000	672.933	898.029
ar.L1.Sales	-0.0880	0.633	-0.139	0.889	-1.329	1.153
ar.L2.Sales	0.2451	0.311	0.787	0.431	-0.365	0.856
ma.L1.Sales	0.3534	0.618	0.572	0.568	-0.859	1.565
ma.L2.Sales	-0.1988	0.309	-0.643	0.520	-0.805	0.408

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.8483	+0.0000j	1.8483	0.5000
AR.2	2.2072	+0.0000j	2.2072	0.0000
MA.1	-1.5236	+0.0000j	1.5236	0.5000
MA.2	3.3015	+0.0000j	3.3015	0.0000

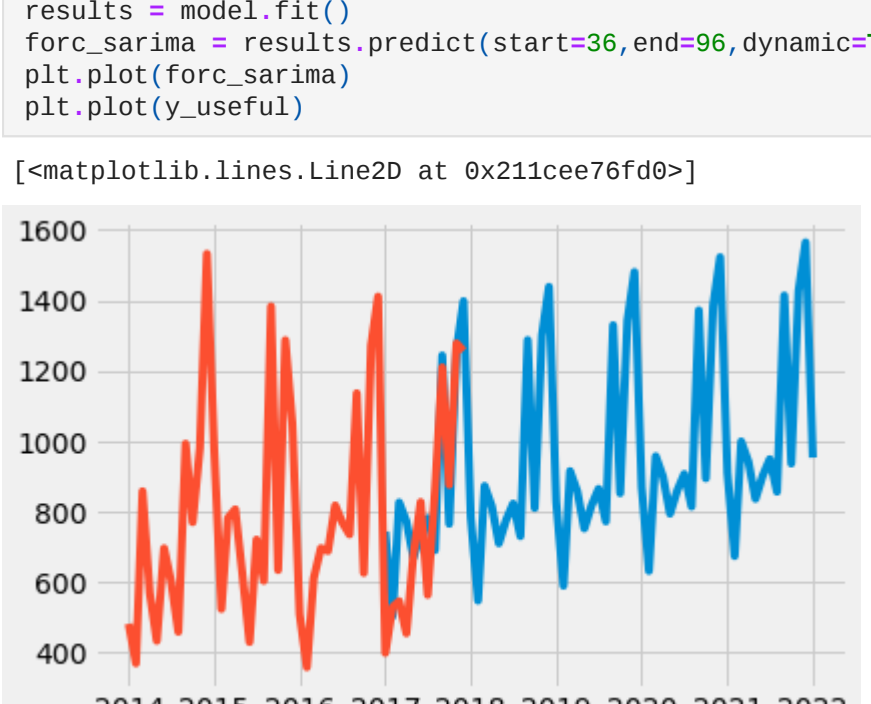
In [17]:
```python
forc_arima = result.predict(start=36,end=96,dynamic=True)
plt.plot(forc_arima)
plt.plot(y_useful)
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x211d014e130>]
```



In [18]:
```python
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(y_useful,order=(1, 1, 1),seasonal_order=(1,1,1,12))
results = model.fit()
forc_sarima = results.predict(start=36,end=96,dynamic=True)
plt.plot(forc_sarima)
plt.plot(y_useful)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x211cee76fd0>]
```



In [19]:
```python
results.summary()
```

Out[19]:

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	48
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-238.291
Date:	Sun, 25 Apr 2021	**AIC**	486.582
Time:	10:30:18	**BIC**	494.359
Sample:	01-01-2014	**HQIC**	489.267
	- 12-01-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0854	0.250	0.342	0.732	-0.404	0.575
ma.L1	-0.9984	13.948	-0.072	0.943	-28.337	26.340
ar.S.L12	0.0273	0.588	0.046	0.963	-1.124	1.179
ma.S.L12	-0.9921	63.664	-0.016	0.988	-125.772	123.788
sigma2	2.741e+04	1.84e+06	0.015	0.988	-3.58e+06	3.64e+06

Ljung-Box (L1) (Q):	0.05	**Jarque-Bera (JB):**	2.10
Prob(Q):	0.82	**Prob(JB):**	0.35
Heteroskedasticity (H):	0.54	**Skew:**	-0.37
Prob(H) (two-sided):	0.30	**Kurtosis:**	2.05

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

In [ ]: