ARIMA and Seasonal ARIMA

Autoregressive Integrated Moving Averages

The general process for ARIMA models is the following:

Visualize the Time Series Data

Make the time series data stationary

Plot the Correlation and AutoCorrelation Charts

Construct the ARIMA Model or Seasonal ARIMA based on the data

Use the model to make predictions

Let's go through these steps!

In [19]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

In [20]:
```
df=pd.read_csv('perrin-freres-monthly-champagne-.csv')
```

In [21]:
```
df.head()
```

Out[21]:

	Month	Perrin Freres monthly champagne sales millions ?64-?72
0	1964-01	2815.0
1	1964-02	2672.0
2	1964-03	2755.0
3	1964-04	2721.0
4	1964-05	2946.0

In [22]:
```
df.tail()
```

Out[22]:

	Month	Perrin Freres monthly champagne sales millions ?64-?72
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN
106	Perrin Freres monthly champagne sales millions...	NaN

In [23]:
```
#cleanin the data set, changing the column name
df.columns=["Months","Sales"]
df.head()
```

Out[23]:

	Months	Sales
0	1964-01	2815.0
1	1964-02	2672.0
2	1964-03	2755.0
3	1964-04	2721.0
4	1964-05	2946.0

In [24]:
```
df
```

Out[24]:

	Months	Sales
0	1964-01	2815.0
1	1964-02	2672.0
2	1964-03	2755.0
3	1964-04	2721.0
4	1964-05	2946.0
...	...	...
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN
106	Perrin Freres monthly champagne sales millions...	NaN

107 rows × 2 columns

In [25]:
```
##dropin last two null rows
df.drop(106,axis=0, inplace=True)
df.drop(105,axis=0, inplace=True)
```

In [26]:
```
df.tail()
```

Out[26]:

	Months	Sales
100	1972-05	4618.0
101	1972-06	5312.0
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0

In [28]:
```
# Convert Month into Datetime
df['Months']=pd.to_datetime(df['Months'])
```

In [29]:
```
df.head()
```

Out[29]:

	Months	Sales
0	1964-01-01	2815.0
1	1964-02-01	2672.0
2	1964-03-01	2755.0
3	1964-04-01	2721.0
4	1964-05-01	2946.0

In [30]:
```
df.set_index('Months',inplace=True)
```

In [31]:
```
df.head()
```

Out[31]:

	Sales
Months	
1964-01-01	2815.0
1964-02-01	2672.0
1964-03-01	2755.0
1964-04-01	2721.0
1964-05-01	2946.0

In [32]:
```
df.describe()
```

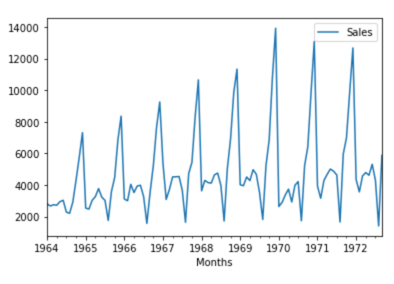
Out[32]:

	Sales
count	105.000000
mean	4761.152381
std	2553.502601
min	1413.000000
25%	3113.000000
50%	4217.000000
75%	5221.000000
max	13916.000000

Step 2: Visualize the Data

In [33]:
```
df.plot()
```

Out[33]:
```
<AxesSubplot:xlabel='Months'>
```



In [34]:
```
# testing for stationarity

from statsmodels.tsa.stattools import adfuller
```

In [37]:
```
test_result = adfuller(df['Sales'])
```

In [38]:
```
#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary")
    else:
        print("weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary ")
```

In [39]:
```
adfuller_test(df['Sales'])
```

```
ADF Test Statistic : -1.8335930563276237
p-value : 0.3639157716602447
#Lags Used : 11
Number of Observations Used : 93
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Differencing

In [40]:
```
df['Sales First Difference'] = df['Sales'] - df['Sales'].shift(1)
```

In [41]:
```
df['Sales'].shift(1)
```

Out[41]:
```
Months
1964-01-01       NaN
1964-02-01    2815.0
1964-03-01    2672.0
1964-04-01    2755.0
1964-05-01    2721.0
               ...
1972-05-01    4788.0
1972-06-01    4618.0
1972-07-01    5312.0
1972-08-01    4298.0
1972-09-01    1413.0
Name: Sales, Length: 105, dtype: float64
```

In [42]:
```
df['Seasonal First Difference']=df['Sales']-df['Sales'].shift(12)
```

In [43]:
```
df.head(14)
```

Out[43]:

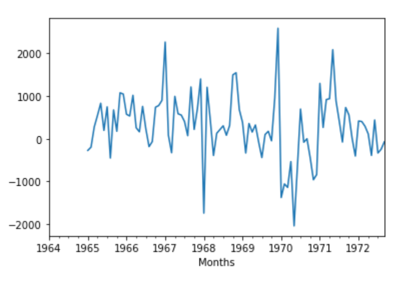
	Sales	Sales First Difference	Seasonal First Difference
Months			
1964-01-01	2815.0	NaN	NaN
1964-02-01	2672.0	-143.0	NaN
1964-03-01	2755.0	83.0	NaN
1964-04-01	2721.0	-34.0	NaN
1964-05-01	2946.0	225.0	NaN
1964-06-01	3036.0	90.0	NaN
1964-07-01	2282.0	-754.0	NaN
1964-08-01	2212.0	-70.0	NaN
1964-09-01	2922.0	710.0	NaN
1964-10-01	4301.0	1379.0	NaN
1964-11-01	5764.0	1463.0	NaN
1964-12-01	7312.0	1548.0	NaN
1965-01-01	2541.0	-4771.0	-274.0
1965-02-01	2475.0	-66.0	-197.0

In [44]:
```
## Again test dickey fuller test
adfuller_test(df['Seasonal First Difference'].dropna())
```

```
ADF Test Statistic : -7.626619157213163
p-value : 2.060579696813685e-11
#Lags Used : 0
Number of Observations Used : 92
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary
```

In [45]:
```
df['Seasonal First Difference'].plot()
```

Out[45]:
```
<AxesSubplot:xlabel='Months'>
```



Auto Regressive Model

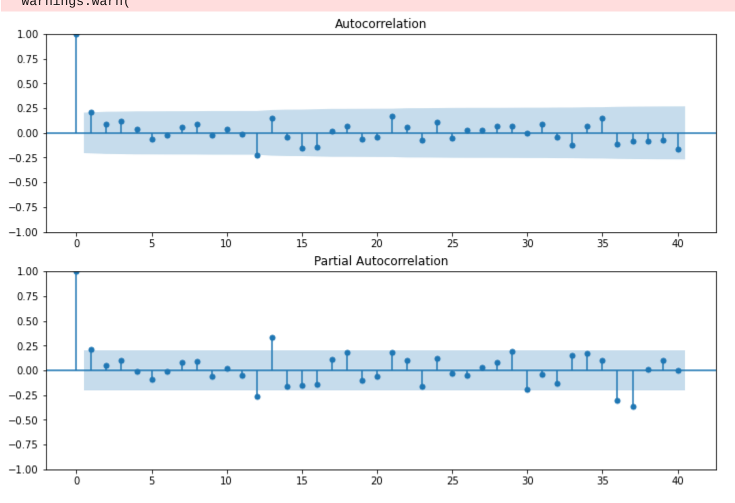
In [62]:
```
df['Sales']
```

Out[62]:
```
Months
1964-01-01    2815.0
1964-02-01    2672.0
1964-03-01    2755.0
1964-04-01    2721.0
1964-05-01    2946.0
               ...
1972-05-01    4618.0
1972-06-01    5312.0
1972-07-01    4298.0
1972-08-01    1413.0
1972-09-01    5877.0
Name: Sales, Length: 105, dtype: float64
```

In [71]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
```

In [72]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```



In [75]:
```
# For non-seasonal data
#p=1, d=1, q=0 or 1
from statsmodels.tsa.arima.model import ARIMA
```

In [76]:
```
model=ARIMA(df['Sales'],order=(1,1,1))
model_fit=model.fit()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

In [77]:
```
model_fit.summary()
```

Out[77]:

SARIMAX Results

Dep. Variable:	Sales	No. Observations:	105
Model:	ARIMA(1, 1, 1)	Log Likelihood	-952.814
Date:	Tue, 11 Oct 2022	AIC	1911.627
Time:	15:10:12	BIC	1919.560
Sample:	01-01-1964	HQIC	1914.841
	- 09-01-1972		
Covariance Type:	opg		

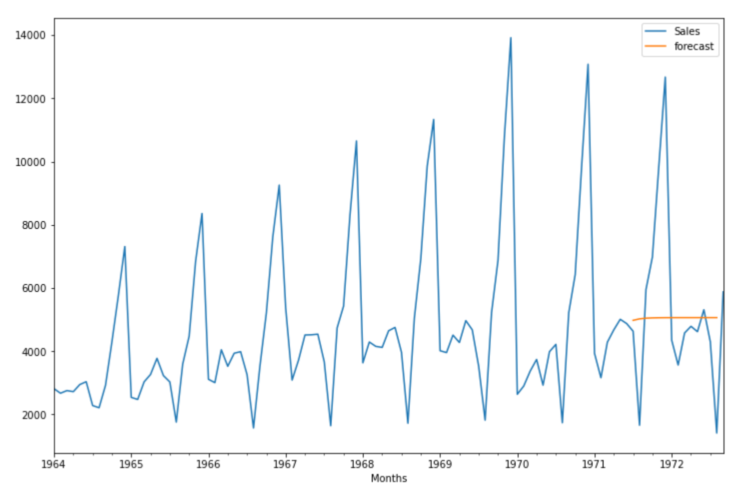
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4545	0.114	3.999	0.000	0.232	0.677
ma.L1	-0.9666	0.056	-17.314	0.000	-1.076	-0.857
sigma2	5.226e+06	6.17e+05	8.473	0.000	4.02e+06	6.43e+06

Ljung-Box (L1) (Q):	0.91	Jarque-Bera (JB):	2.59
Prob(Q):	0.34	Prob(JB):	0.27
Heteroskedasticity (H):	3.40	Skew:	0.05
Prob(H) (two-sided):	0.00	Kurtosis:	3.77

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

In [78]:
```
df['forecast']=model_fit.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[78]:
```
<AxesSubplot:xlabel='Months'>
```

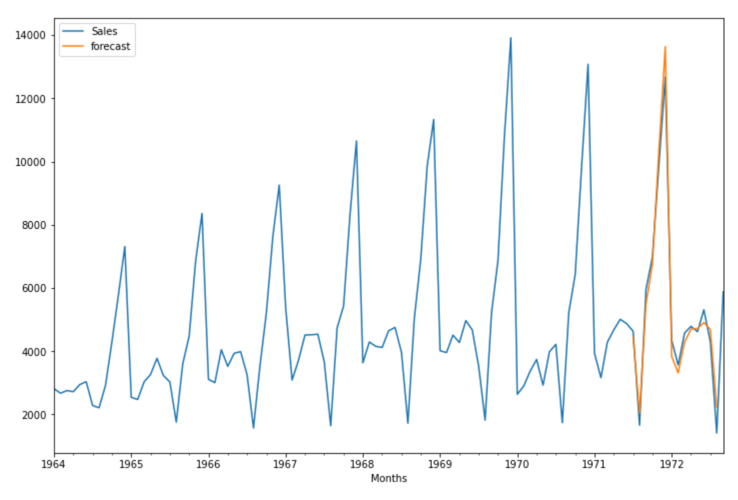


In [79]:
```
model=sm.tsa.statespace.SARIMAX(df['Sales'],order=(1, 1, 1),seasonal_order=(1,1,1,12))
results=model.fit()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

In [80]:
```
df['forecast']=results.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[80]:
```
<AxesSubplot:xlabel='Months'>
```



In [81]:
```
from pandas.tseries.offsets import DateOffset
future_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]
```

In [82]:
```
future_datest_df=pd.DataFrame(index=future_dates[1:],columns=df.columns)
```

In [83]:
```
future_datest_df.tail()
```

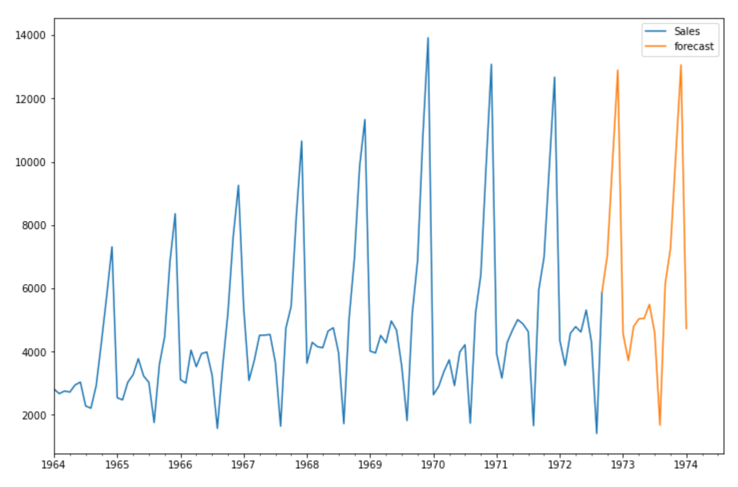
Out[83]:

	Sales	Sales First Difference	Seasonal First Difference	forecast
1974-04-01	NaN	NaN	NaN	NaN
1974-05-01	NaN	NaN	NaN	NaN
1974-06-01	NaN	NaN	NaN	NaN
1974-07-01	NaN	NaN	NaN	NaN
1974-08-01	NaN	NaN	NaN	NaN

In [84]:
```
future_df=pd.concat([df,future_datest_df])
```

In [85]:
```
future_df['forecast'] = results.predict(start = 104, end = 120, dynamic= True)
future_df[['Sales', 'forecast']].plot(figsize=(12, 8))
```

Out[85]:
```
<AxesSubplot:>
```



In [ ]: