ARIMA and SEASONAL ARIMA

AUTO REGRESSIVE INTEGRATED MOVING AVERAGE

The general purpose of the arima model is as follows

- visualize the time series data
- make the time series data stationary
- plot the corelation and auto corelation plots
- construct the arima or seasonal arima model
- use the model to make predistions ### lets go through the steps

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```
```
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [4]:
```
df=pd.read_csv("~/downloads/perrin-freres-monthly-champagne-.csv")
```

In [5]:
```
df
```
Out[5]:

	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
...	...	...
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 [6]:
```
df.head()
```
Out[6]:

	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 [7]:
```
df.tail()
```
Out[7]:

	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

we have observed some nan values so we need to remove them

cleaning up the data

In [11]:
```
df.columns=['Month','sales']
df
```
Out[11]:

	Month	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 [12]:
```
df.drop(106,axis=0,inplace=True)
```

In [13]:
```
df
```
Out[13]:

	Month	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
...	...	...
101	1972-06	5312.0
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN

106 rows × 2 columns

In [14]:
```
df.tail()
```
Out[14]:

	Month	sales
101	1972-06	5312.0
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN

In [15]:
```
df.drop(105,axis=0,inplace=True)
```

In [16]:
```
df.tail()
```
Out[16]:

	Month	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

convert the month into date time variable

In [21]:
```
df['Month']=pd.to_datetime(df['Month'])
```

In [22]:
```
df
```
Out[22]:

	Month	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
...	...	...
100	1972-05-01	4618.0
101	1972-06-01	5312.0
102	1972-07-01	4298.0
103	1972-08-01	1413.0
104	1972-09-01	5877.0

105 rows × 2 columns

In [23]:
```
df.tail()
```
Out[23]:

	Month	sales
100	1972-05-01	4618.0
101	1972-06-01	5312.0
102	1972-07-01	4298.0
103	1972-08-01	1413.0
104	1972-09-01	5877.0

In [24]:
```
df.set_index('Month',inplace=True)
```

In [25]:
```
df
```
Out[25]:

Month	sales
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

105 rows × 1 columns

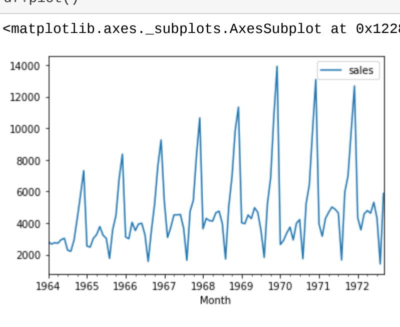
In [26]:
```
df.head()
```
Out[26]:

Month	sales
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 [27]:
```
df.describe()
```
Out[27]:

	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

In [28]:
```
df.plot()
```
Out[28]: `<matplotlib.axes._subplots.AxesSubplot at 0x1228ade80>`



testing for stationary

In [29]:
```
from statsmodels.tsa.stattools import adfuller
```

In [30]:
```
test_result=adfuller(df['sales'])
```

H0 is not stationary

h1 is stationary

In [31]:
```
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, indicatin
g it is non-stationary ")
```

In [33]:
```
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-stat
ionary
```

Differencing

In [35]:
```
df['Sales First Difference'] = df['sales'] - df['sales'].shift(1)
```

In [37]:
```
df['sales'].shift(1)
```
Out[37]:
```
Month
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
```

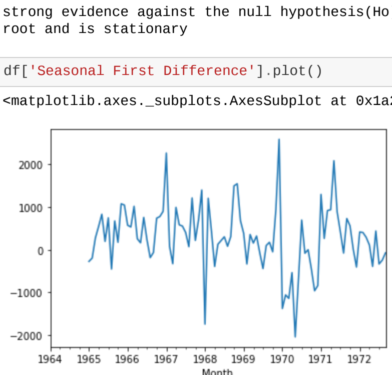
In [39]:
```
df['Seasonal First Difference']=df['sales']-df['sales'].shift(12)
```

In [42]:
```
df.head(14)
```
Out[42]:

Month	sales	Sales First Difference	Seasonal First Difference
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

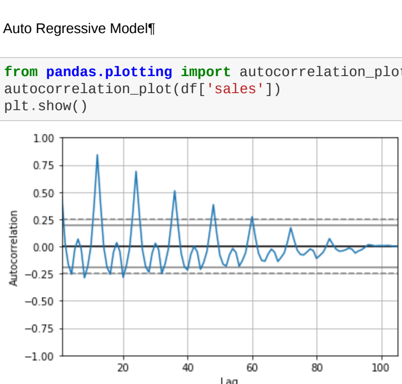
In [43]:
```
## Again test dickey fuller test
adfuller_test(df['Seasonal First Difference'].dropna())
```
```
ADF Test Statistic : -7.626619157213163
p-value : 2.060579696813685e-11
#Lags Used : 0
Number of Observations Used : 92
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit
root and is stationary
```

In [44]:
```
df['Seasonal First Difference'].plot()
```
Out[44]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a25e6fa58>`



Auto Regressive Model¶

In [48]:
```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(df['sales'])
plt.show()
```



Final Thoughts on Autocorrelation and Partial Autocorrelation

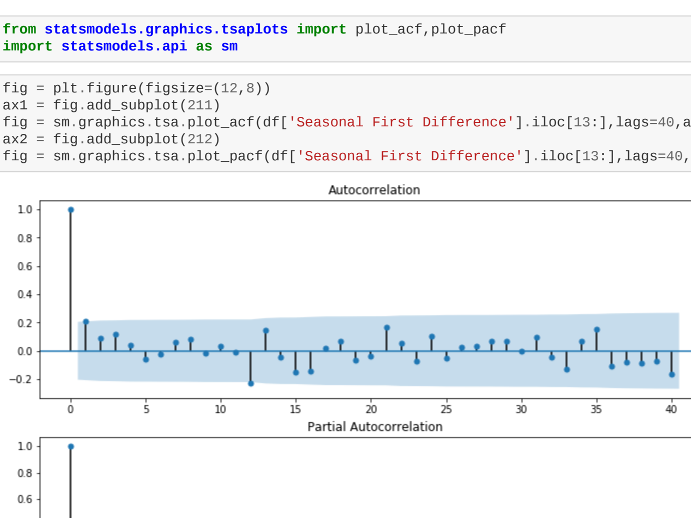
Identification of an AR model is often best done with the PACF. For an AR model, the theoretical PACF "shuts off" past the order of the model. The phrase "shuts off" means that in theory the partial autocorrelations are equal to 0 beyond that point. Put another way, the number of non-zero partial autocorrelations gives the order of the AR model. By the "order of the model" we mean the most extreme lag of x that is used as a predictor. Identification of an MA model is often best done with the ACF rather than the PACF.

For an MA model, the theoretical PACF does not shut off, but instead tapers toward 0 in some manner. A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.

p,d,q p AR model lags d differencing q MA lags

In [74]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
```

In [76]:
```
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)
```



In [57]:
```
# For non-seasonal data
#p=1, d=1, q=0 or 1
from statsmodels.tsa.arima_model import ARIMA
```

In [59]:
```
model=ARIMA(df['sales'],order=(1,1,1))
model_fit=model.fit()
```
```
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/site-packages/statsmodels/tsa/bas
e/tsa_model.py:162: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  % freq, ValueWarning)
```

In [60]:
```
model_fit.summary()
```
Out[60]:

ARIMA Model Results

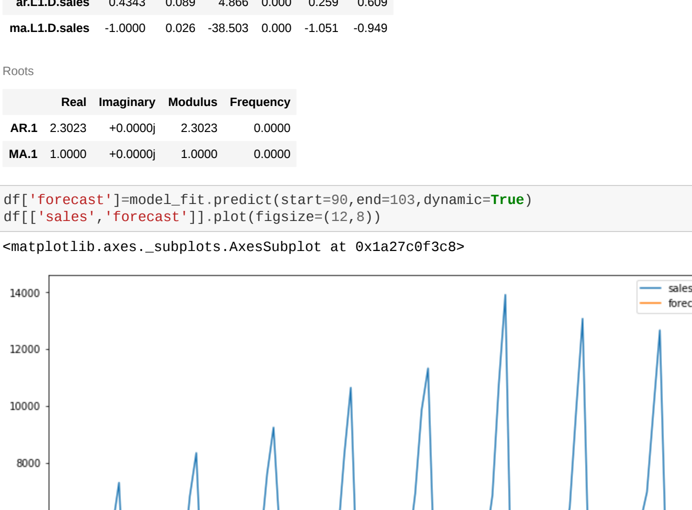
Dep. Variable:	D.sales	No. Observations:	104
Model:	ARIMA(1, 1, 1)	Log Likelihood	-951.126
Method:	css-mle	S.D. of innovations	2227.262
Date:	Sat, 21 Mar 2020	AIC	1910.251
Time:	11:34:02	BIC	1920.829
Sample:	02-01-1964	HQIC	1914.536
	- 09-01-1972		

	coef	std err	z	P>|z|	[0.025	0.975]
const	22.7821	12.405	1.836	0.066	-1.532	47.096
ar.L1.D.sales	0.4343	0.089	4.866	0.000	0.259	0.609
ma.L1.D.sales	-1.0000	0.026	-38.503	0.000	-1.051	-0.949

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	2.3023	+0.0000j	2.3023	0.0000
MA.1	1.0000	+0.0000j	1.0000	0.0000

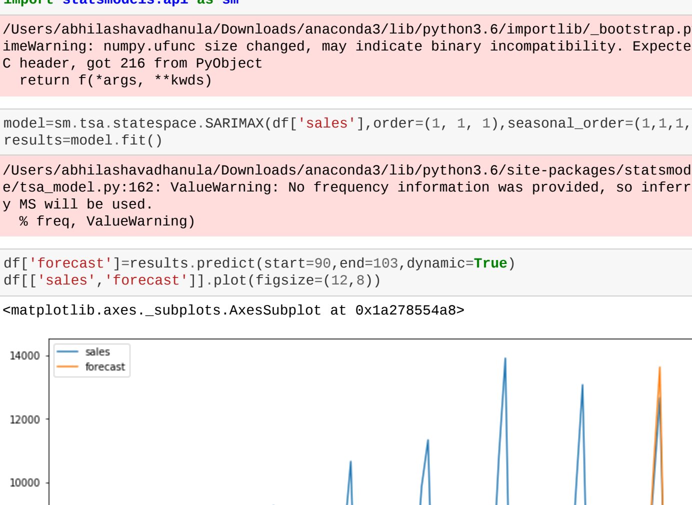
In [62]:
```
df['forecast']=model_fit.predict(start=90,end=103,dynamic=True)
df[['sales','forecast']].plot(figsize=(12,8))
```
Out[62]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a27c0f3c8>`



In [63]:
```
import statsmodels.api as sm
```
```
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [65]:
```
model=sm.tsa.statespace.SARIMAX(df['sales'],order=(1, 1, 1),seasonal_order=(1,1,1,12))
results=model.fit()
```
```
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/site-packages/statsmodels/tsa/bas
e/tsa_model.py:162: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  % freq, ValueWarning)
```

In [67]:
```
df['forecast']=results.predict(start=90,end=103,dynamic=True)
df[['sales','forecast']].plot(figsize=(12,8))
```
Out[67]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a278554a8>`



In [68]:
```
from pandas.tseries.offsets import DateOffset
future_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]
```

In [69]:
```
future_datest_df=pd.DataFrame(index=future_dates[1:],columns=df.columns)
```

In [70]:
```
future_datest_df.tail()
```
Out[70]:

	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 [71]:
```
future_df=pd.concat([df,future_datest_df])
```

In [73]:
```
future_df['forecast'] = results.predict(start = 104, end = 120, dynamic= True)
future_df[['sales', 'forecast']].plot(figsize=(12, 8))
```
Out[73]: `<matplotlib.axes._subplots.AxesSubplot at 0x122c65208>`