Import all necessary libraries

In [1]:
```python
import numpy as np
import pandas as pd

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

As data is in the form of excel

In [8]:
```python
df=pd.read_csv(r'perrin freres monthly champagne.csv')
```

In [9]:
```python
df.head()
```

Out[9]:

	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 [10]:
```python
df.tail()
```

Out[10]:

	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

Perform Exploratory Data Analysis

In [11]:
```python
df.shape
```

Out[11]:
```
(107, 2)
```

In [13]:
```python
## Cleaning up the data
df.columns=["Month","Sales"]
df.head()
```

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

In [14]:
```python
## Drop last 2 rows
df.drop(106,axis=0,inplace=True)
```

In [15]:
```python
df.tail()
```

Out[15]:

	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 [16]:
```python
df.drop(105,axis=0,inplace=True)
```

In [17]:
```python
df.tail()
```

Out[17]:

	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

In [18]:
```python
# Convert Month into Datetime
df['Month']=pd.to_datetime(df['Month'])
```

In [19]:
```python
df.head()
```

Out[19]:

	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

In [20]:
```python
df.set_index('Month',inplace=True)
```

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

Out[21]:

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 [22]:
```python
df.describe()
```

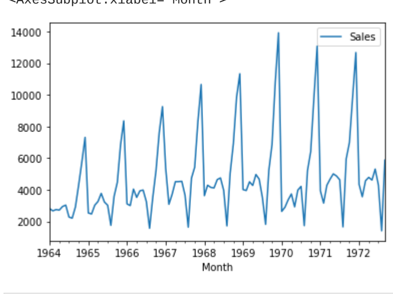
Out[22]:

	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

Visualize the Data

In [23]:
```python
df.plot()
```

Out[23]: <AxesSubplot:xlabel='Month'>



In [24]:
```python
#looking this graph,we can say this data is seasonal
#(seasonal is all about suppose in each yr in christmas sales goes up) and then down
```

In [25]:
```python
# we plot,whether test is stationary or not
#if not stationery then how to make it stationery
```

In [26]:
```python
### Testing For Stationarity

from statsmodels.tsa.stattools import adfuller
```

In [27]:
```python
test_result=adfuller(df['Sales'])
```

In [28]:
```python
#adfuller gives 5 values ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
#most imp is p-value
#this is almost similar to Hypothesis testing,whereas Null hypo which says Data is not Stationery
#whereas Alternate hypo says data is stationery
```

In [29]:
```python
###if Pvalue<0.05 ,we reject Null hypo ie alternate hypo is true,ie data is stationery
```

In [30]:
```python
#zip basically combines result,labels

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 [31]:
```python
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
```

In [32]:
```python
#if our data is non-stationery,we have to make it stationery using various techniques such as Differencing & many more
```

Differencing

In [33]:
```python
df
```

Out[33]:

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 [34]:
```python
df['Sales'].shift(1)
```

Out[34]:
```
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 [35]:
```python
df['Sales First Difference'] = df['Sales'] - df['Sales'].shift(1)
```

In [36]:
```python
#why taken shift(12),bcz basically year has 12 month cycle
df['Seasonal First Difference']=df['Sales']-df['Sales'].shift(12)
```

In [37]:
```python
df.head(14)
```

Out[37]:

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

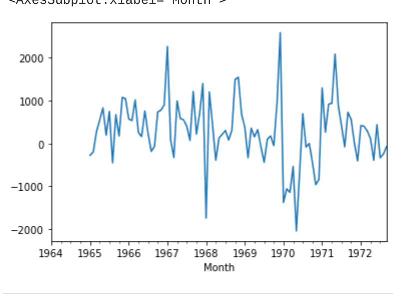
In [38]:
```python
## Again test dickey fuller test on df['Sales First Difference']
adfuller_test(df['Seasonal First Difference'].dropna())
```

```
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 [39]:
```python
#now our p-value is less than 0.05 which basically says we are rejecting null hypo and accepting alternate hypo
#ie data is stationery
#if p-value is almost 0,then we have a wonderful stationery graph
```

In [40]:
```python
df['Seasonal First Difference'].plot()
```

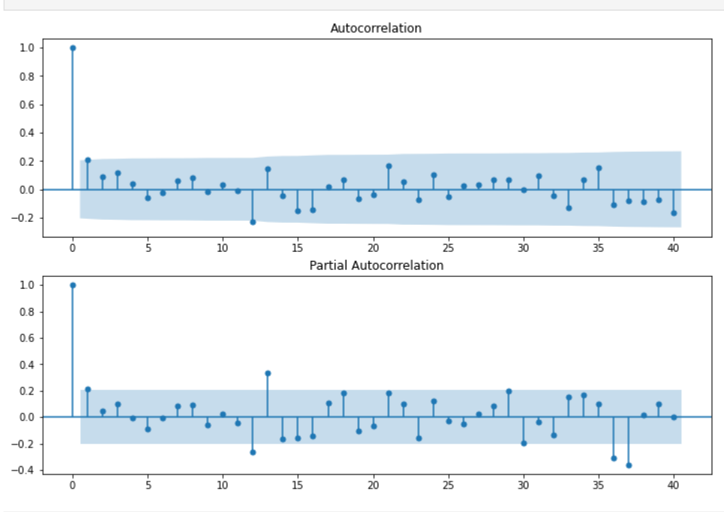
Out[40]: <AxesSubplot:xlabel='Month'>



In [41]:
```python
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
#Autocorrelation(plot_acf),Partial Autocorrelation(plot_pacf)
```

In [42]:
```python
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

In [44]:
```python
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = plot_acf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax2)
```



In [ ]:

In [50]:
```python
# For non-seasonal data
#p=1, d=1, q=0 or 1
import statsmodels.tsa.arima.model as stats
```

In [53]:
```python
model=ARIMA(df['Sales'],order=(1,1,1))
model_fit=model.fit()
import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA', FutureWarning)
```

In [54]:
```python
model_fit.summary()
```

Out[54]:

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	2227.263
Date:	Sun, 01 Aug 2021	AIC	1910.251
Time:	19:35:33	BIC	1920.829
Sample:	02-01-1964	HQIC	1914.536
	- 09-01-1972		

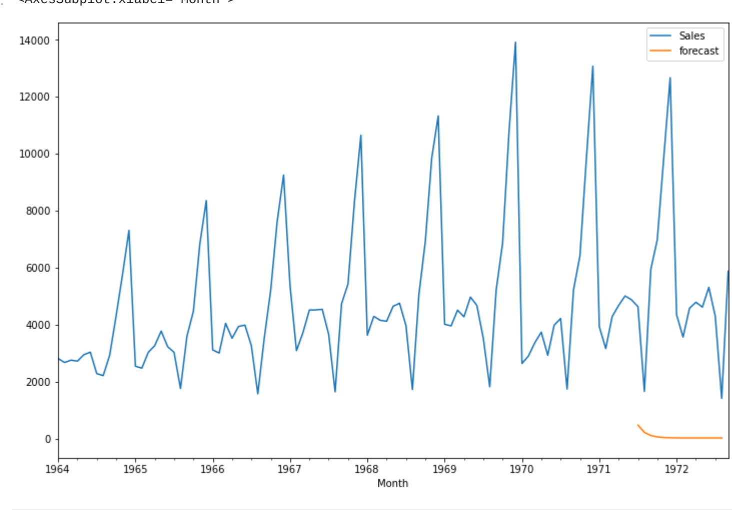
	coef	std err	z	P>|z|	[0.025	0.975]
const	22.7844	12.405	1.837	0.066	-1.529	47.098
ar.L1.D.Sales	0.4343	0.089	4.866	0.000	0.259	0.609
ma.L1.D.Sales	-1.0000	0.026	-38.503	0.000	-1.051	-0.949

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 [55]:
```python
df['forecast']=model_fit.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[55]: <AxesSubplot:xlabel='Month'>



In [56]:
```python
## note: when u have seasonal data, use SARIMAX over there..
```

In [57]:
```python
import statsmodels.api as sm
```

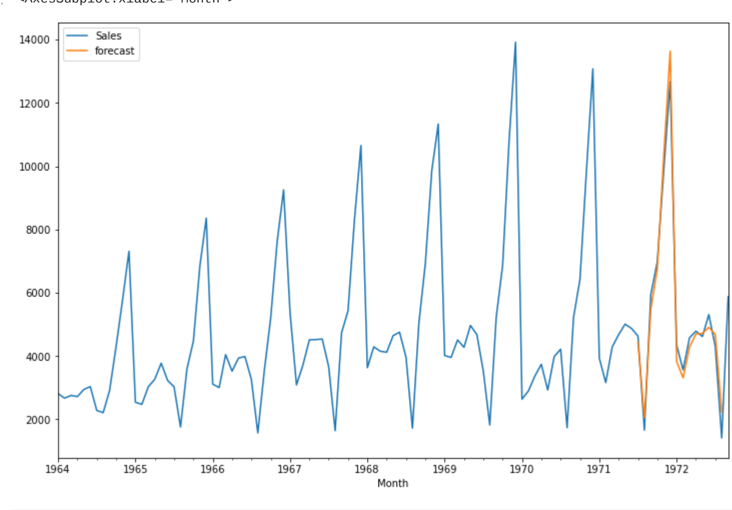
In [58]:
```python
#seasonal_order=(1,1,1,12)
#(p,d,q,shift_value)
```

In [59]:
```python
model=sm.tsa.statespace.SARIMAX(df['Sales'],order=(1, 1, 1),seasonal_order=(1,1,1,12))
results=model.fit()
```

In [60]:
```python
#predicting from index 90 to 103
df['forecast']=results.predict(start=90,end=103,dynamic=True)
```

In [61]:
```python
#blue line is original data anf orange is a forecasted data
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[61]: <AxesSubplot:xlabel='Month'>



In [62]:
```python
### to see how future predictions/projections looks like
```

In [63]:
```python
from pandas.tseries.offsets import DateOffset
```

In [64]:
```python
df.index[-1]
```

Out[64]:
```
Timestamp('1972-09-01 00:00:00')
```

In [65]:
```python
future_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]
```

In [66]:
```python
future_dates
```

Out[66]:
```
[Timestamp('1972-09-01 00:00:00'),
 Timestamp('1972-10-01 00:00:00'),
 Timestamp('1972-11-01 00:00:00'),
 Timestamp('1972-12-01 00:00:00'),
 Timestamp('1973-01-01 00:00:00'),
 Timestamp('1973-02-01 00:00:00'),
 Timestamp('1973-03-01 00:00:00'),
 Timestamp('1973-04-01 00:00:00'),
 Timestamp('1973-05-01 00:00:00'),
 Timestamp('1973-06-01 00:00:00'),
 Timestamp('1973-07-01 00:00:00'),
 Timestamp('1973-08-01 00:00:00'),
 Timestamp('1973-09-01 00:00:00'),
 Timestamp('1973-10-01 00:00:00'),
 Timestamp('1973-11-01 00:00:00'),
 Timestamp('1973-12-01 00:00:00'),
 Timestamp('1974-01-01 00:00:00'),
 Timestamp('1974-02-01 00:00:00'),
 Timestamp('1974-03-01 00:00:00'),
 Timestamp('1974-04-01 00:00:00'),
 Timestamp('1974-05-01 00:00:00'),
 Timestamp('1974-06-01 00:00:00'),
 Timestamp('1974-07-01 00:00:00'),
 Timestamp('1974-08-01 00:00:00')]
```

In [77]:
```python
from pandas.tseries.offsets import DateOffset

#creating additional dataset for 24 months,
future_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]
```

In [80]:
```python
future_datest_df=pd.DataFrame(index=future_dates[1:],columns=df.columns)
```

In [81]:
```python
future_datest_df.tail()
```

Out[81]:

	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 [82]:
```python
future_df=pd.concat([df,future_datest_df])
```

In [83]:
```python
#then we can see future predictions
```

In [84]:
```python
future_df['forecast'] = results.predict(start = 104, end = 120, dynamic= True)
future_df[['Sales', 'forecast']].plot(figsize=(12, 8))
```

Out[84]: <AxesSubplot:>