In [156]:
```
import numpy as np
import pandas as pd

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

In [157]:
```
df = pd.read_csv(r'file:///C:\Users\Acer\AppData\Local\Temp\Temp1_archive%20(1).z
df.head(5)
```

Out[157]:

	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 [158]:
```
df.columns = ['Months','Sales']
df.head(5)
```

Out[158]:

	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 [159]:
```
df.isnull()
```

Out[159]:

	Months	Sales
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
102	False	False
103	False	False
104	False	False
105	True	True
106	False	True

107 rows × 2 columns

In [160]:
```
df.drop(105,axis=0,inplace=True)
df.drop(106,axis=0,inplace=True)
df.isnull().sum()
```

Out[160]:
```
Months    0
Sales     0
dtype: int64
```

In [161]:
```
df.dtypes
```

Out[161]:
```
Months     object
Sales     float64
dtype: object
```

In [162]:
```
df['Months'] = pd.to_datetime(df['Months'])
df.dtypes
```

Out[162]:
```
Months    datetime64[ns]
Sales            float64
dtype: object
```

In [163]:
```
df.set_index('Months',inplace=True)
df.head(5)
```

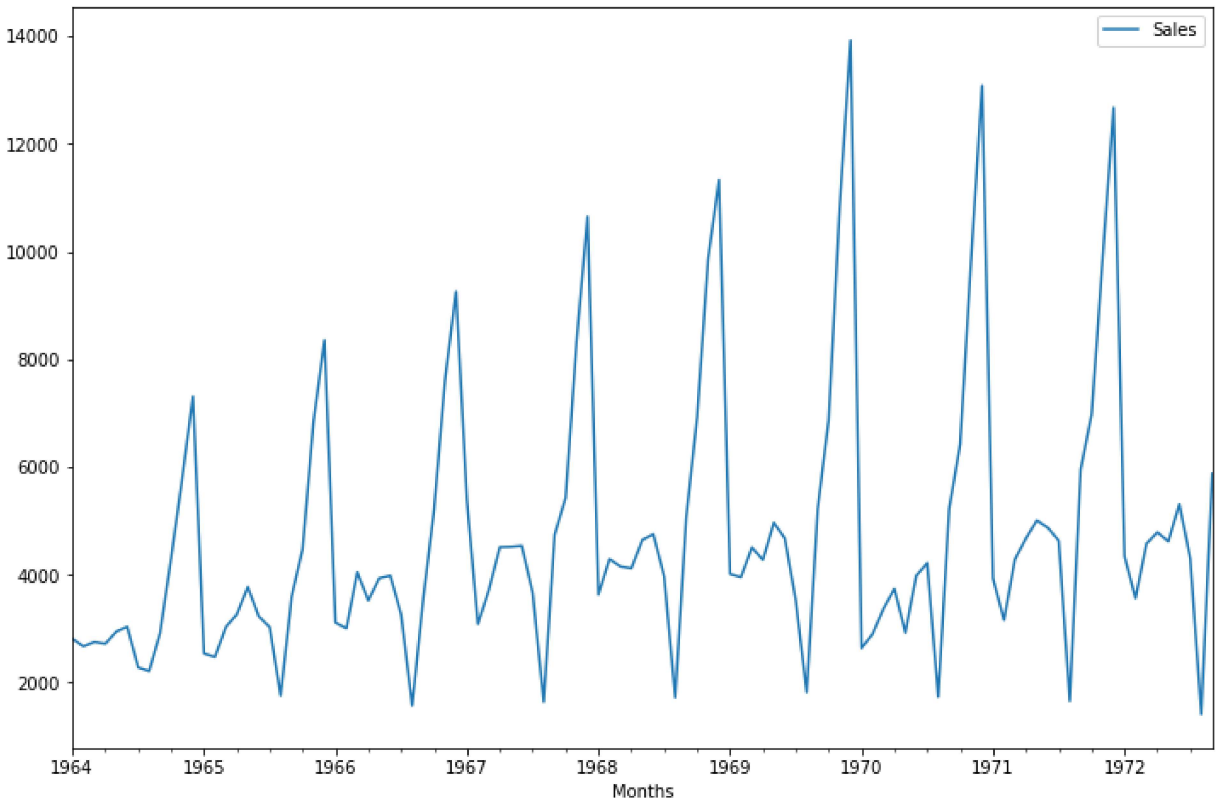
Out[163]:

Months	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

#Visualize the Data

In [164]:
```
df.plot(figsize= (12,8))
```

Out[164]: <AxesSubplot:xlabel='Months'>



In [165]:
```
from statsmodels.tsa.stattools import adfuller
```

In [166]:
```
# Accept Null Hpyo means dataset is Not Stationary
# Reject Null Hypo Means dataset is Stationary
def adfuller_test(sales):
    result = adfuller(sales)
    labels = ['ADF Test statistic','p-value','#lags used','Number of Observations
    for values, label in zip(result,labels):
        print(label +':'+ str(values))
    if(result[1] <= 0.05):
        print("The Dataset is stationary, Reject Null Hypothesis")
    else:
        print("The Dataset is Not Stationary, Accept Null Hypothesis")
```

In [167]:
```
adfuller_test(df['Sales'])
```

```
ADF Test statistic:-1.8335930563276197
p-value:0.3639157716602465
#lags used:11
Number of Observations:93
The Dataset is Not Stationary, Accept Null Hypothesis
```

Differencing

In [168]:
```
df['Seasonal Sales Diff'] = df['Sales'] - df['Sales'].shift(12)
df
```

Out[168]:

Months	Sales	Seasonal Sales Diff
1964-01-01	2815.0	NaN
1964-02-01	2672.0	NaN
1964-03-01	2755.0	NaN
1964-04-01	2721.0	NaN
1964-05-01	2946.0	NaN
...	...	...
1972-05-01	4618.0	-392.0
1972-06-01	5312.0	438.0
1972-07-01	4298.0	-335.0
1972-08-01	1413.0	-246.0
1972-09-01	5877.0	-74.0

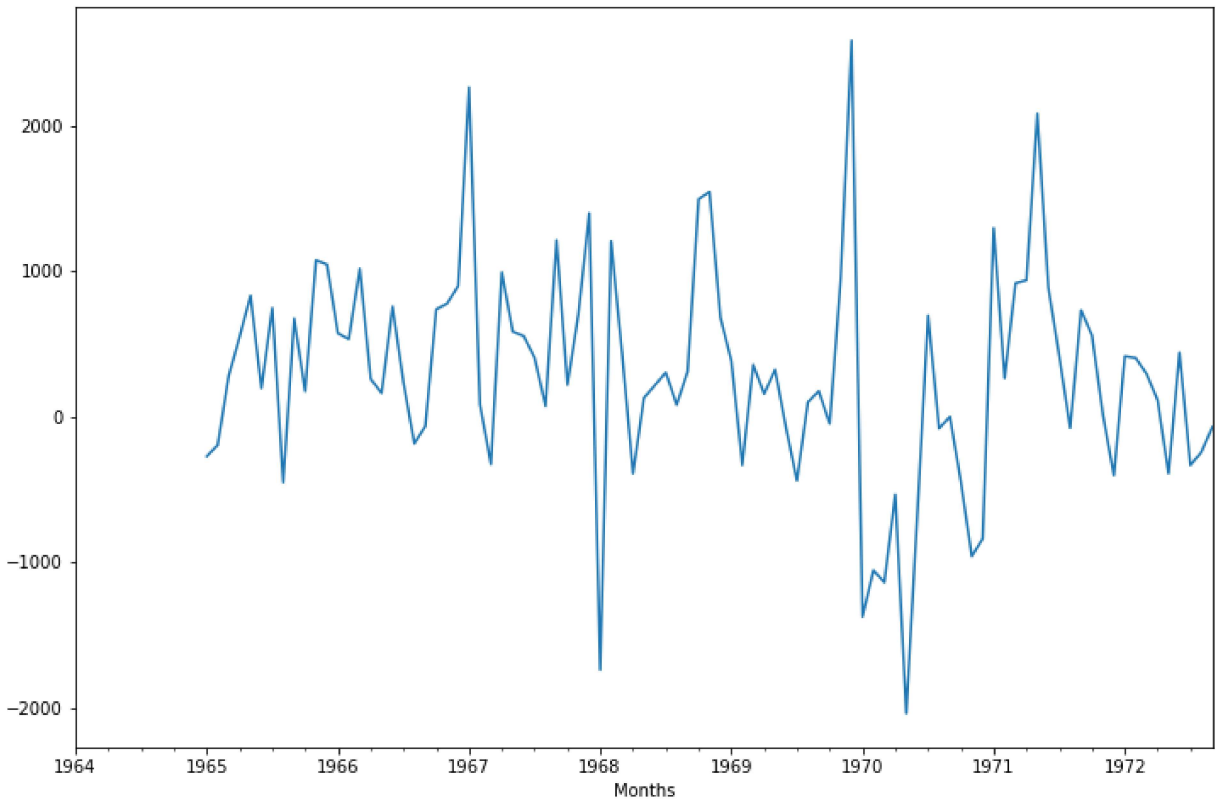
105 rows × 2 columns

In [169]:
```
adfuller_test(df['Seasonal Sales Diff'].dropna())
```

```
ADF Test statistic:-7.626619157213162
p-value:2.060579696813685e-11
#lags used:0
Number of Observations:92
The Dataset is stationary, Reject Null Hypothesis
```

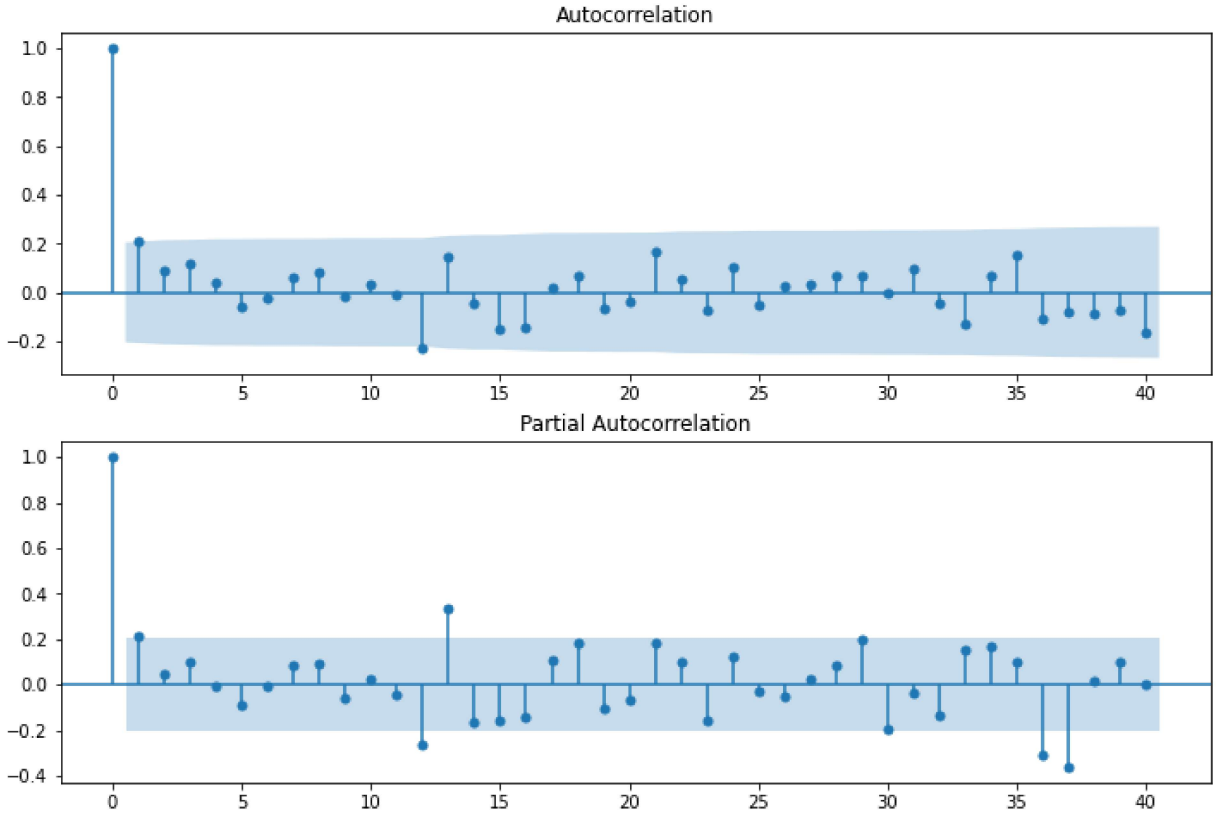
In [170]:
```
df['Seasonal Sales Diff'].plot(figsize=(12,8))
```

Out[170]:
```
<AxesSubplot:xlabel='Months'>
```



In [171]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

In [172]:
```
fig =plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = plot_acf(df['Seasonal Sales Diff'].iloc[13:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(df['Seasonal Sales Diff'].iloc[13:],lags= 40,ax=ax2)
```



In [173]:
```
# For Non Seasonal Data
#p=1 d=1 q can be 0 or 1
from statsmodels.tsa.arima_model import ARIMA
```

In [174]:
```
# For Non Seasonal Data
#p=1 d=1 q can be 0 or 1
from statsmodels.tsa.arima_model import ARIMA
```

In [175]:
```
model = ARIMA(df['Sales'],order=(1,1,1))
model_fit = model.fit()
```

```
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472: F
utureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:52
4: ValueWarning: No frequency information was provided, so inferred frequency M
S will be used.
  warnings.warn('No frequency information was'
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:52
4: ValueWarning: No frequency information was provided, so inferred frequency M
S will be used.
  warnings.warn('No frequency information was'
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472: F
utureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

In [176]:
```
model_fit.summary()
```

Out[176]:

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:	Sun, 05 Dec 2021	**AIC**	1910.251
Time:	12:39:28	**BIC**	1920.829
Sample:	02-01-1964	**HQIC**	1914.536
	- 09-01-1972		

	coef	std err	z	P>|z|	[0.025	0.975]
const	22.7853	12.405	1.837	0.066	-1.529	47.099
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

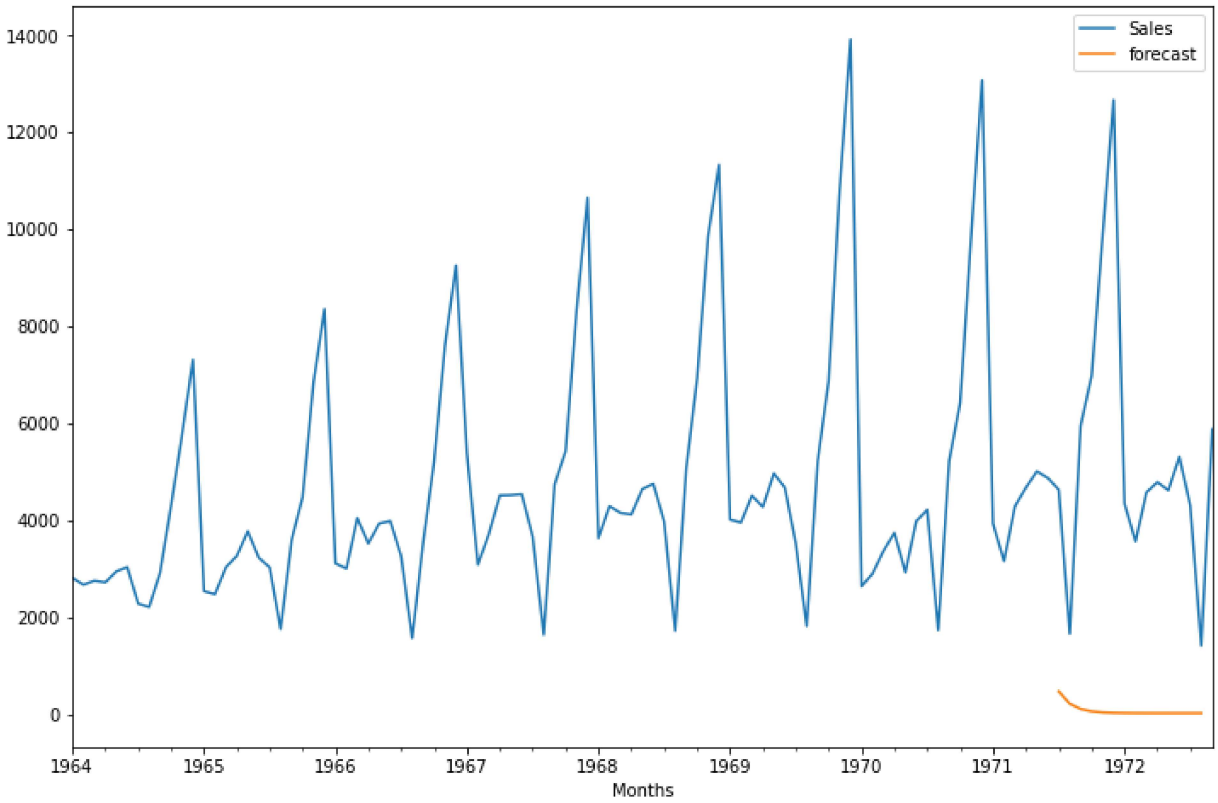
In [177]:
```
df.tail(20)
```

Out[177]:

Months	Sales	Seasonal Sales Diff
1971-02-01	3162.0	263.0
1971-03-01	4286.0	916.0
1971-04-01	4676.0	936.0
1971-05-01	5010.0	2083.0
1971-06-01	4874.0	888.0
1971-07-01	4633.0	416.0
1971-08-01	1659.0	-79.0
1971-09-01	5951.0	730.0
1971-10-01	6981.0	557.0
1971-11-01	9851.0	9.0
1971-12-01	12670.0	-406.0
1972-01-01	4348.0	414.0
1972-02-01	3564.0	402.0
1972-03-01	4577.0	291.0
1972-04-01	4788.0	112.0
1972-05-01	4618.0	-392.0
1972-06-01	5312.0	438.0
1972-07-01	4298.0	-335.0
1972-08-01	1413.0	-246.0
1972-09-01	5877.0	-74.0

In [178]:
```
df['forecast'] = model_fit.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[178]: <AxesSubplot:xlabel='Months'>



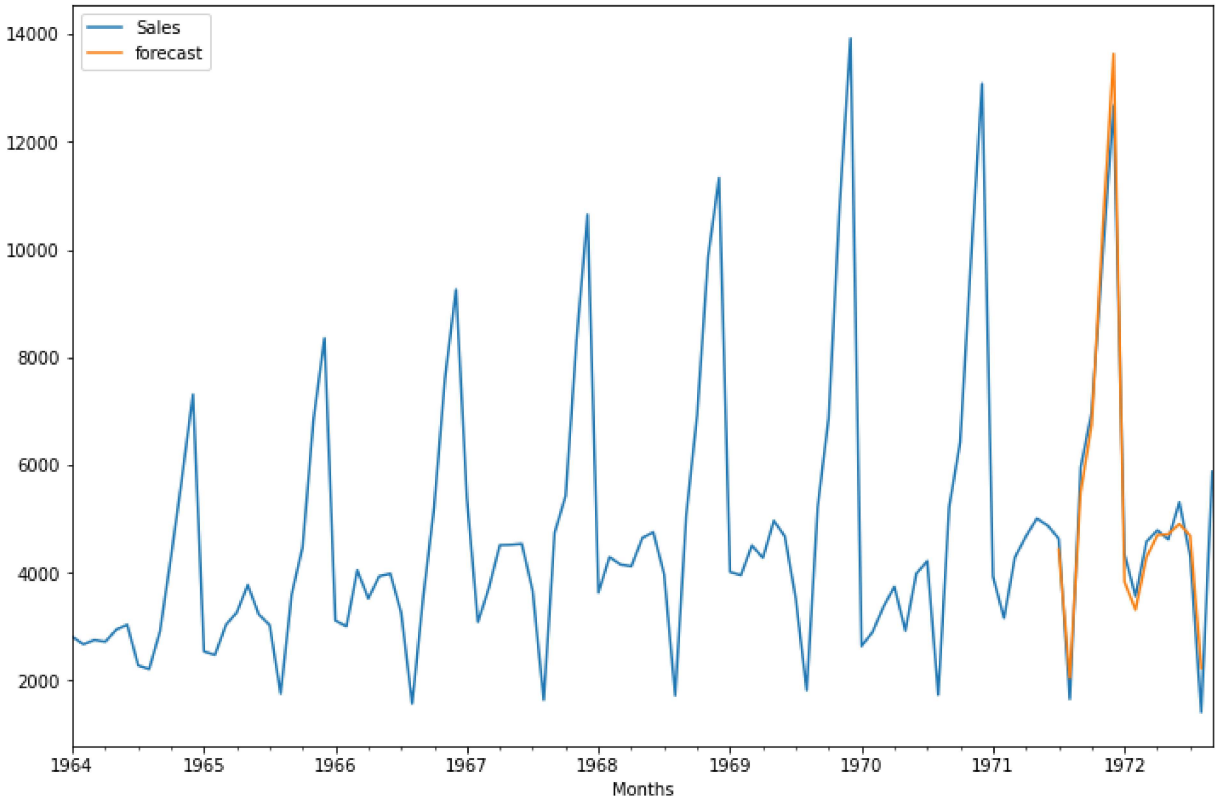
In [179]:
```
import statsmodels.api as sm
```

In [180]:
```
model = sm.tsa.statespace.SARIMAX(df['Sales'],order=(1,1,1),seasonal_order=(1,1,1
results = model.fit()
```

```
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:52
4: ValueWarning: No frequency information was provided, so inferred frequency M
S will be used.
  warnings.warn('No frequency information was'
C:\Users\Acer\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:52
4: ValueWarning: No frequency information was provided, so inferred frequency M
S will be used.
  warnings.warn('No frequency information was'
```

In [181]:
```
df['forecast'] = results.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[181]: <AxesSubplot:xlabel='Months'>



In [182]:
```
from pandas.tseries.offsets import DateOffset
```

In [183]:
```
future_dates = [df.index[-1]+ DateOffset(months=x) for x in range(0,24)]
```

In [184]:
```
future_dataset_df = pd.DataFrame(index=future_dates[1:],columns=df.columns)
```

In [185]:
```
future_dataset_df.tail(20)
```

Out[185]:

	Sales	Seasonal Sales Diff	forecast
1973-01-01	NaN	NaN	NaN
1973-02-01	NaN	NaN	NaN
1973-03-01	NaN	NaN	NaN
1973-04-01	NaN	NaN	NaN
1973-05-01	NaN	NaN	NaN
1973-06-01	NaN	NaN	NaN
1973-07-01	NaN	NaN	NaN
1973-08-01	NaN	NaN	NaN
1973-09-01	NaN	NaN	NaN
1973-10-01	NaN	NaN	NaN
1973-11-01	NaN	NaN	NaN
1973-12-01	NaN	NaN	NaN
1974-01-01	NaN	NaN	NaN
1974-02-01	NaN	NaN	NaN
1974-03-01	NaN	NaN	NaN
1974-04-01	NaN	NaN	NaN
1974-05-01	NaN	NaN	NaN
1974-06-01	NaN	NaN	NaN
1974-07-01	NaN	NaN	NaN
1974-08-01	NaN	NaN	NaN

In [186]:
```
future_dataset_df.shape
```

Out[186]:
```
(23, 3)
```

In [187]:
```
future_df = pd.concat([df,future_dataset_df])
```

In [188]:
```
future_df['forecast'] = results.predict(start=104,end=120,dynamic=True)
future_df[['Sales','forecast']].plot(figsize=(12,8))
```

Out[188]: <AxesSubplot:>