Modelling and forecasting inventories

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
df=pd.read_csv(r'C:\Users\chumj\Downloads\INVENTORIES.csv',index_col='Date',parse_dates=True)
```

In [3]:

```
df.index
```

Out[3]:

```
DatetimeIndex(['1997-01-01', '1997-02-01', '1997-03-01', '1997-04-01',
               '1997-05-01', '1997-06-01', '1997-07-01', '1997-08-01',
               '1997-09-01', '1997-10-01',
               ...
               '2018-03-01', '2018-04-01', '2018-05-01', '2018-06-01',
               '2018-07-01', '2018-08-01', '2018-09-01', '2018-10-01',
               '2018-11-01', '2018-12-01'],
              dtype='datetime64[ns]', name='Date', length=264, freq=None)
```

In [4]:

```
# Our data seems to be a monthly,lets make the frequency monthly
df.index.freq='MS'
```

In [5]:

```
df.dropna(inplace=True)
```

In [6]:

```
df
```

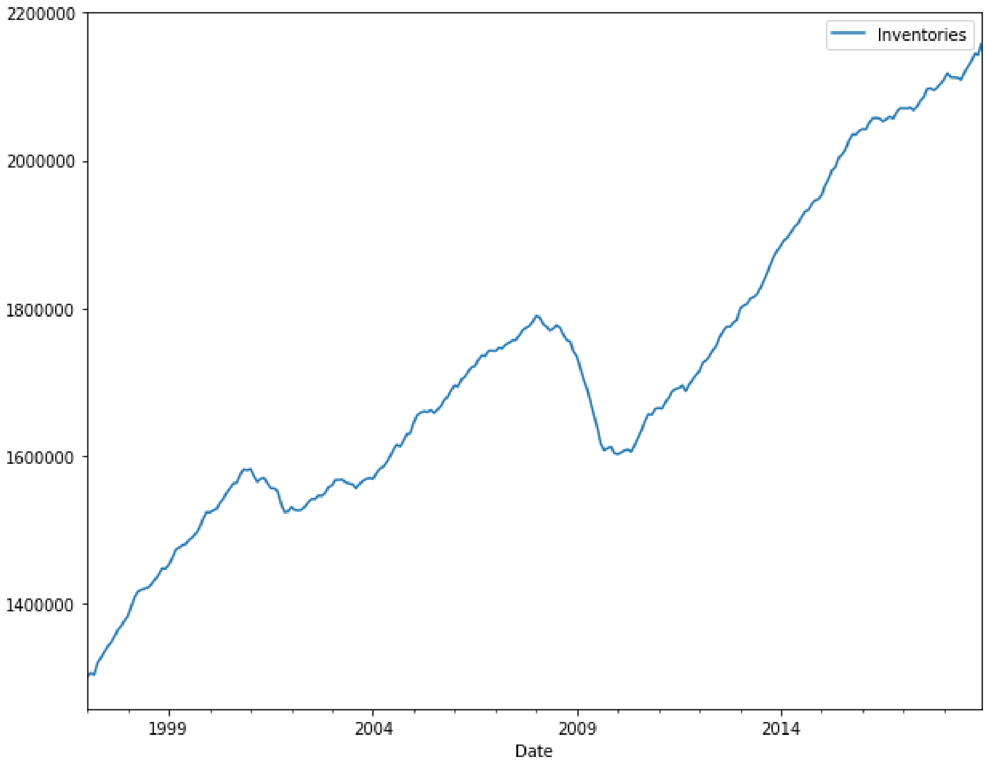
Out[6]:

	Inventories
Date	
1997-01-01	1301161
1997-02-01	1307080
1997-03-01	1303978
1997-04-01	1319740
1997-05-01	1327294
...	...
2018-08-01	2127170
2018-09-01	2134172
2018-10-01	2144639
2018-11-01	2143001
2018-12-01	2158115

264 rows × 1 columns

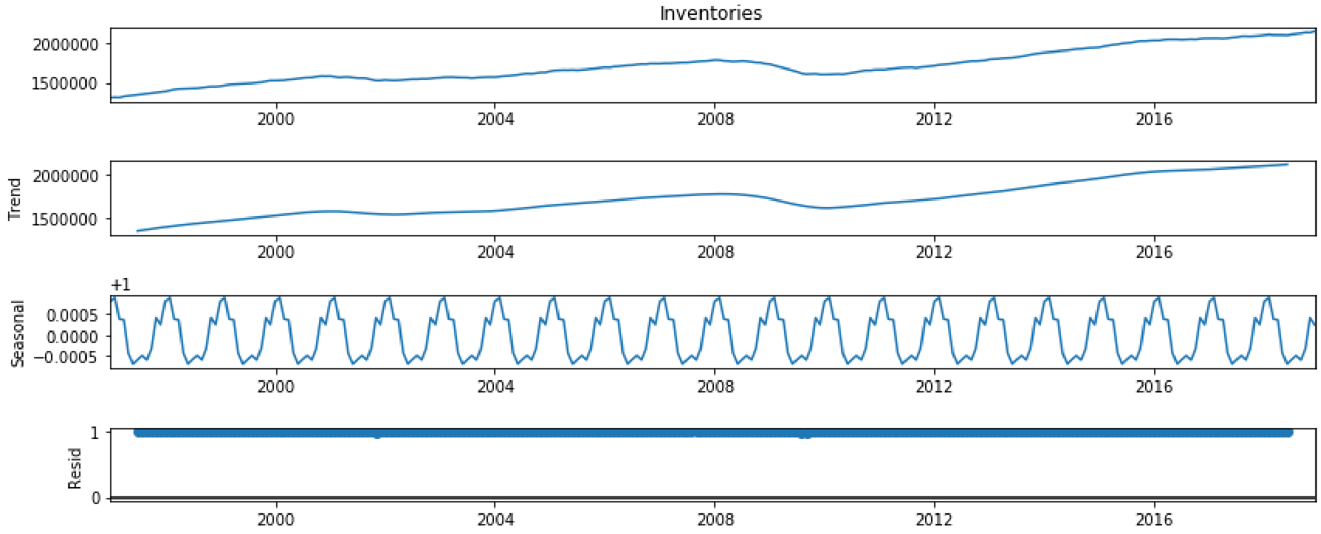
In [7]:

```
df.plot(figsize=(10,8));
```



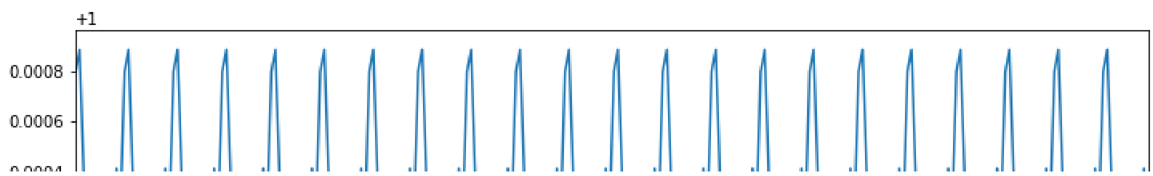
In [8]:

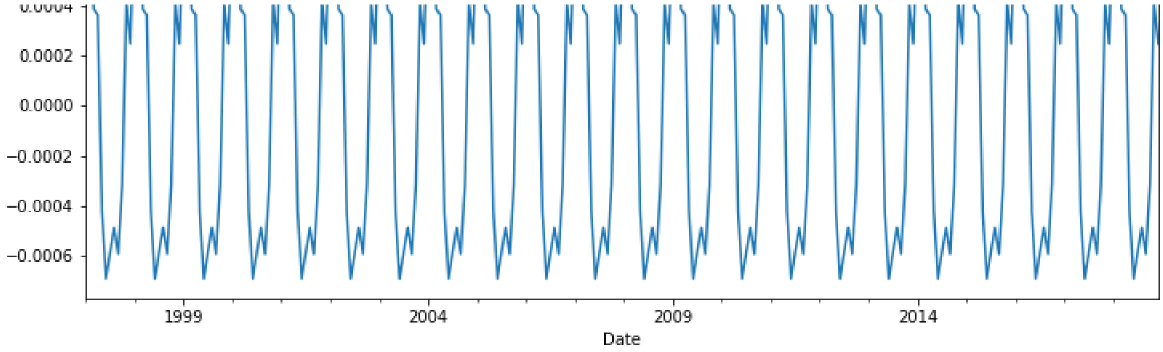
```
#Applying ETS decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
result=seasonal_decompose(df['Inventories'],model='mul')
from pylab import rcParams
rcParams['figure.figsize']=12,5
result.plot();
```



In [9]:

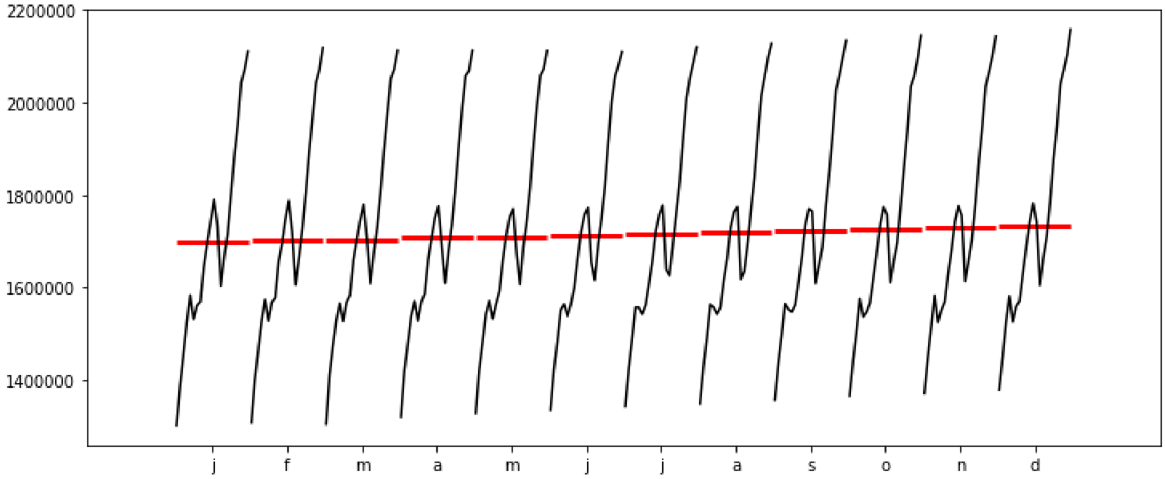
```
result.seasonal.plot();
```





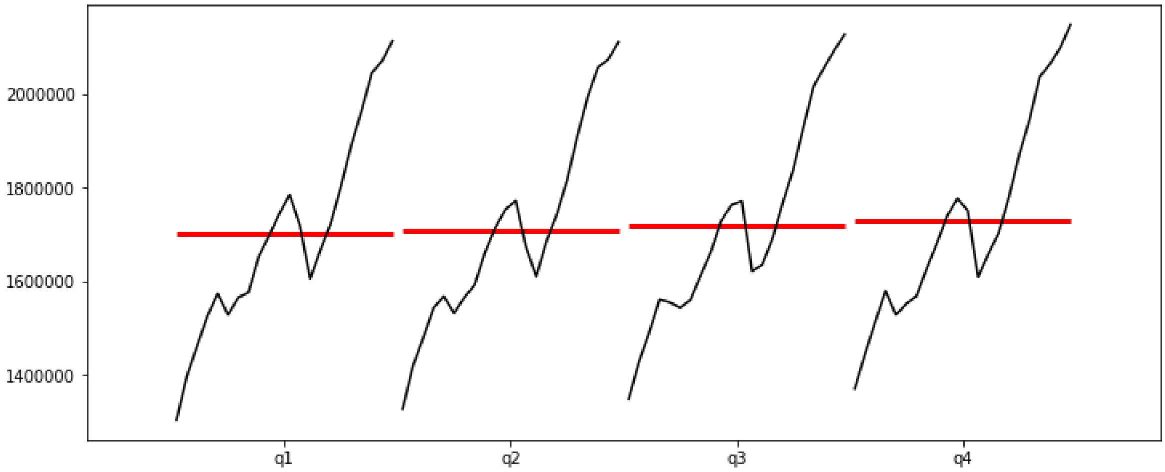
In [10]:

```
#Trying to expose Seasonality with Months and Quarter Plots
from statsmodels.graphics.tsaplots import month_plot,quarter_plot
month_plot(df['Inventories']);
```



In [11]:

```
dfq=df['Inventories'].resample(rule='Q').mean()
quarter_plot(dfq);
```



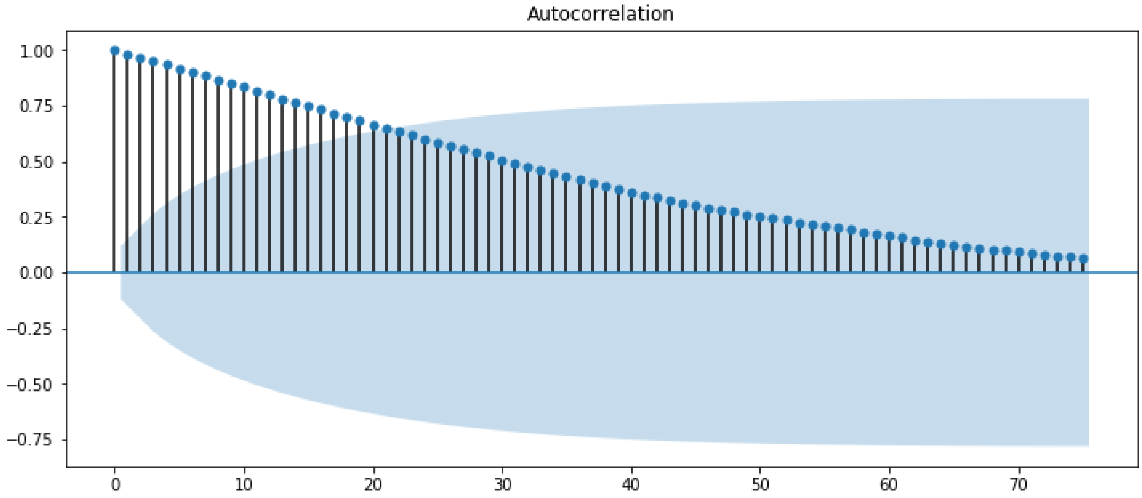
From all indications no seasonality but an upward trend in our data

In [12]:

```
#Tryin to visualized ACF and PACF plots
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

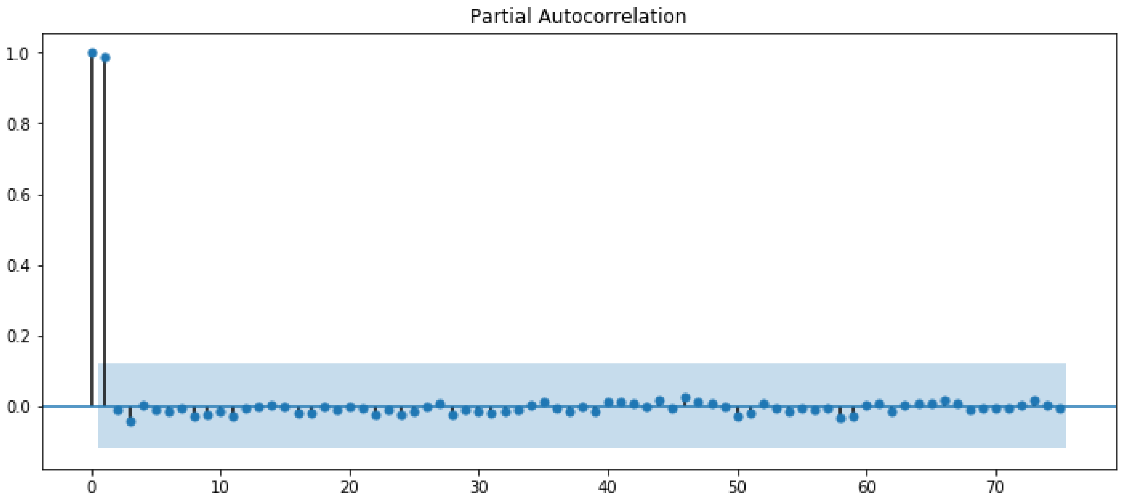
In [13]:

```
plot_acf(df,lags=75);
```



In [14]:

```
plot_pacf(df,lags=75);
```



Visually,AR portion really outbeats MA,base on the gradual decaying of the ACF plot and the sharp drop of PACF.

In [56]:

```
#choosing the best ARIMA for our data using auto_arima
from pmdarima import auto_arima
```

In [58]:

```
auto_arima(df['Inventories'],seasonal=False).summary()
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```

```

In [17]:

```
#test for statinarity with augment Dickey_Fuller
```

In [18]:

```
from statsmodels.tsa.stattools import adfuller


def adf_test(series,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series.dropna(),autolag='AIC')

    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    print(out.to_string())

    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has no unit root and is stationary")
    else:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has a unit root and is non-stationary")
```

In [19]:

```
adf_test(df['Inventories'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -0.087684
p-value                   0.950652
# lags used               5.000000
# observations          258.000000
critical value (1%)      -3.455953
critical value (5%)      -2.872809
critical value (10%)     -2.572775
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data has a unit root and is non-stationary
```

In [20]:

```
#which is totally true base on (0,1,0) best choosen parameter for our data,it shows we need differ
encing of 1, that is d=1,
#to make our dataset stationary
from statsmodels.tsa.statespace.tools import diff
df['df_k1']=diff(df['Inventories'],k_diff=1)
```

In [21]:

```
adf_test(df['df_k1'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -3.412249
p-value                   0.010548
# lags used               4.000000
# observations          258.000000
critical value (1%)      -3.455953
critical value (5%)      -2.872809
critical value (10%)     -2.572775
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [59]:

```
stepwise_fit=auto_arima(df['Inventories'],start_p=0,start_q=0,max_p=3,max_q=3,m=12,seasonal=False,
                        trace=True,stepwise=True,error_action='ignore')
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=5348.037, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=5399.843, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=5350.241, Time=0.12 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=5409.217, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=5378.835, Time=0.45 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0] intercept
Total fit time: 0.774 seconds
```

In [23]:

```
# Best Model,ARIMA(1,1,1) with the smallest AIC 5378.835
```

In [60]:

```
stepwise_fit.summary()
```

Out[60]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	264
Model:	SARIMAX(0, 1, 0)	**Log Likelihood**	-2672.018
Date:	Thu, 03 Sep 2020	**AIC**	5348.037
Time:	20:14:25	**BIC**	5355.181
Sample:	0	**HQIC**	5350.908
	- 264		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	3258.3802	470.991	6.918	0.000	2335.255	4181.506
sigma2	3.91e+07	2.95e+06	13.250	0.000	3.33e+07	4.49e+07

Ljung-Box (Q):	455.75	**Jarque-Bera (JB):**	100.74
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.86	**Skew:**	-1.15
Prob(H) (two-sided):	0.48	**Kurtosis:**	4.98

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

In [25]:

```
#splitting data into train/test sets and make forecasting for 1 year of 12months
len(df)
```

Out[25]:

```
264
```

In [26]:

```
train=df.iloc[:252]
test=df.iloc[252:]
```

In [27]:

```python
#fitting the ARIMA(0,1,0) Model
from statsmodels.tsa.arima_model import ARIMA,ARMA,ARIMAResults
```

In [61]:

```python
model=ARIMA(train['Inventories'],order=(1,1,1))
results=model.fit()
results.summary()
```

Out[61]:

ARIMA Model Results

Dep. Variable:	D.Inventories	**No. Observations:**	251
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-2486.395
Method:	css-mle	**S.D. of innovations**	4845.028
Date:	Thu, 03 Sep 2020	**AIC**	4980.790
Time:	20:16:02	**BIC**	4994.892
Sample:	02-01-1997	**HQIC**	4986.465
	- 12-01-2017		

	coef	std err	z	P>|z|	[0.025	0.975]
const	3197.5698	1344.866	2.378	0.017	561.681	5833.459
ar.L1.D.Inventories	0.9026	0.039	23.010	0.000	0.826	0.979
ma.L1.D.Inventories	-0.5581	0.079	-7.048	0.000	-0.713	-0.403

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.1080	+0.0000j	1.1080	0.0000
MA.1	1.7918	+0.0000j	1.7918	0.0000

In [62]:

```python
start=len(train)
end=len(train)+len(test)-1
```

In [63]:

```python
predictions=results.predict(start=start,end=end,dynamic=False,typ='levels').rename('ARIMA')
```

In [64]:

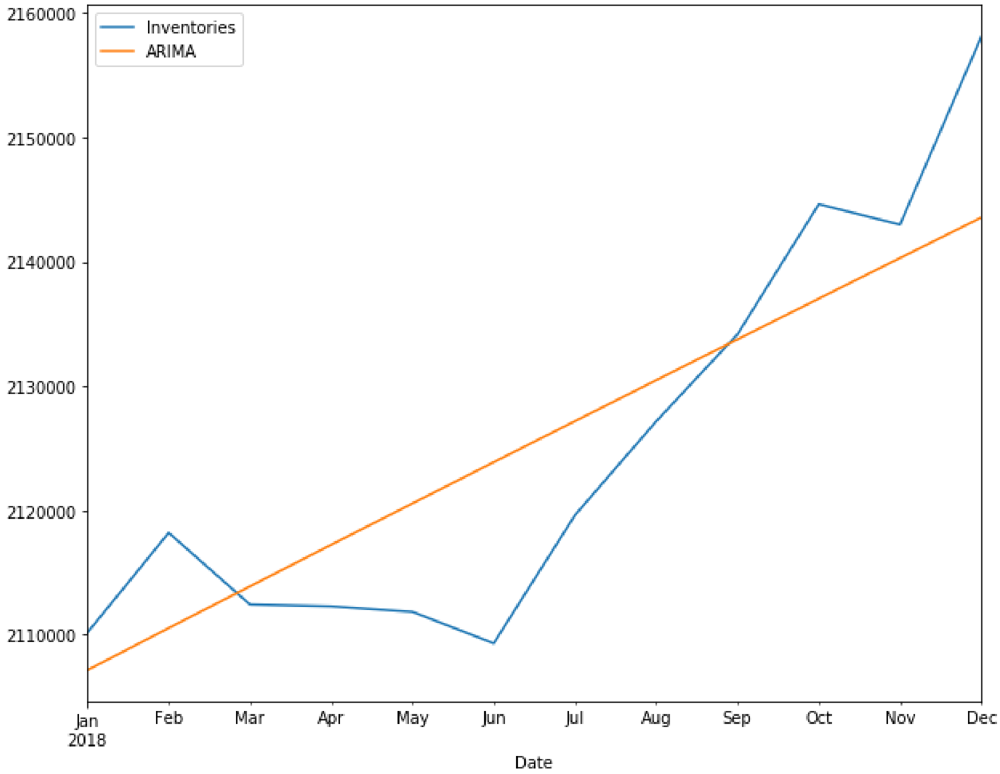
```python
predictions
```

Out[64]:

```
2018-01-01    2.107148e+06
2018-02-01    2.110526e+06
2018-03-01    2.113887e+06
2018-04-01    2.117231e+06
2018-05-01    2.120561e+06
2018-06-01    2.123878e+06
2018-07-01    2.127184e+06
2018-08-01    2.130479e+06
2018-09-01    2.133764e+06
2018-10-01    2.137041e+06
2018-11-01    2.140311e+06
2018-12-01    2.143573e+06
Freq: MS, Name: ARIMA, dtype: float64
```

In [65]:

```
test['Inventories'].plot(legend=True,figsize=(10,8))
predictions.plot(legend=True);
```



In [66]:

```
#Model Evaluation
from sklearn.metrics import mean_squared_error,mean_absolute_error
print(mean_squared_error(test['Inventories'],predictions))
```

60677830.18985107

In [67]:

```
print(np.sqrt(mean_squared_error(test['Inventories'],predictions)))
```

7789.597562766068

In [68]:

```
from statsmodels.tools.eval_measures import rmse
rmse(test['Inventories'],predictions)
```

Out[68]:

7789.597562766068

In [69]:

```
test['Inventories'].mean()
```

Out[69]:

2125075.6666666665

In [70]:

```
model=ARIMA(df['Inventories'],order=(1,1,0))
results=model.fit()
final_forecast=results.predict(len(df),len(df)+11,typ='levels').rename('ARIMA_FORECAST')
```

In [71]:

```
df['Inventories'].plot(legend=True,figsize=(10,8))
final_forecast.plot(legend=True)
```

Out[71]:

<matplotlib.axes._subplots.AxesSubplot at 0x26c11275308>