In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv(r'F:\Prthon Programming\Time Series Modelling\AirlinePassengers_data.csv')
df.head()
```

Out[1]:

	ID	Datetime	Count
0	0	25-08-2012 00:00	8
1	1	25-08-2012 01:00	2
2	2	25-08-2012 02:00	6
3	3	25-08-2012 03:00	2
4	4	25-08-2012 04:00	2

In [2]:
```
df.dtypes
```

Out[2]:
```
ID            int64
Datetime     object
Count         int64
dtype: object
```

In [3]:
```
df.Timestamp = pd.to_datetime(df.Datetime,format='%d-%m-%Y %H:%M')
df.index = df.Timestamp
df = df.resample('D').sum()
```

```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  """Entry point for launching an IPython kernel.
```

In [5]:
```
df.tail()
```

Out[5]:

	ID	Count
Datetime		
2013-12-27	281940	3868
2013-12-28	282516	3084
2013-12-29	283092	2330
2013-12-30	283668	4928
2013-12-31	284244	4860

In [6]:
```
df.sort_index(axis=0)
```

Out[6]:

	ID	Count
Datetime		
2012-08-25	276	76
2012-08-26	852	88
2012-08-27	1428	62
2012-08-28	2004	58
2012-08-29	2580	60
2012-08-30	3156	74
2012-08-31	3732	78
2012-09-01	4308	112
2012-09-02	4884	118
2012-09-03	5460	108
2012-09-04	6036	66
2012-09-05	6612	104
2012-09-06	7188	100
2012-09-07	7764	68
2012-09-08	8340	100
2012-09-09	8916	68
2012-09-10	9492	64
2012-09-11	10068	58
2012-09-12	10644	84
2012-09-13	11220	72
2012-09-14	11796	84
2012-09-15	12372	92
2012-09-16	12948	70
2012-09-17	13524	92
2012-09-18	14100	188
2012-09-19	14676	158
2012-09-20	15252	84
2012-09-21	15828	90
2012-09-22	16404	78
2012-09-23	16980	102
...	...	...
2013-12-02	267540	4042
2013-12-03	268116	6186
2013-12-04	268692	5078

Datetime	ID	Count
2013-12-05	269268	4948
2013-12-06	269844	4558
2013-12-07	270420	2344
2013-12-08	270996	2414
2013-12-09	271572	4394
2013-12-10	272148	3850
2013-12-11	272724	4808
2013-12-12	273300	4772
2013-12-13	273876	3624
2013-12-14	274452	2166
2013-12-15	275028	2722
2013-12-16	275604	5252
2013-12-17	276180	4508
2013-12-18	276756	4708
2013-12-19	277332	4344
2013-12-20	277908	3474
2013-12-21	278484	1988
2013-12-22	279060	1862
2013-12-23	279636	3642
2013-12-24	280212	3198
2013-12-25	280788	3676
2013-12-26	281364	3376
2013-12-27	281940	3868
2013-12-28	282516	3084
2013-12-29	283092	2330
2013-12-30	283668	4928
2013-12-31	284244	4860

494 rows × 2 columns

In [7]:
```
train=df.loc[:'2013-10-31']
train.tail()
```

Out[7]:

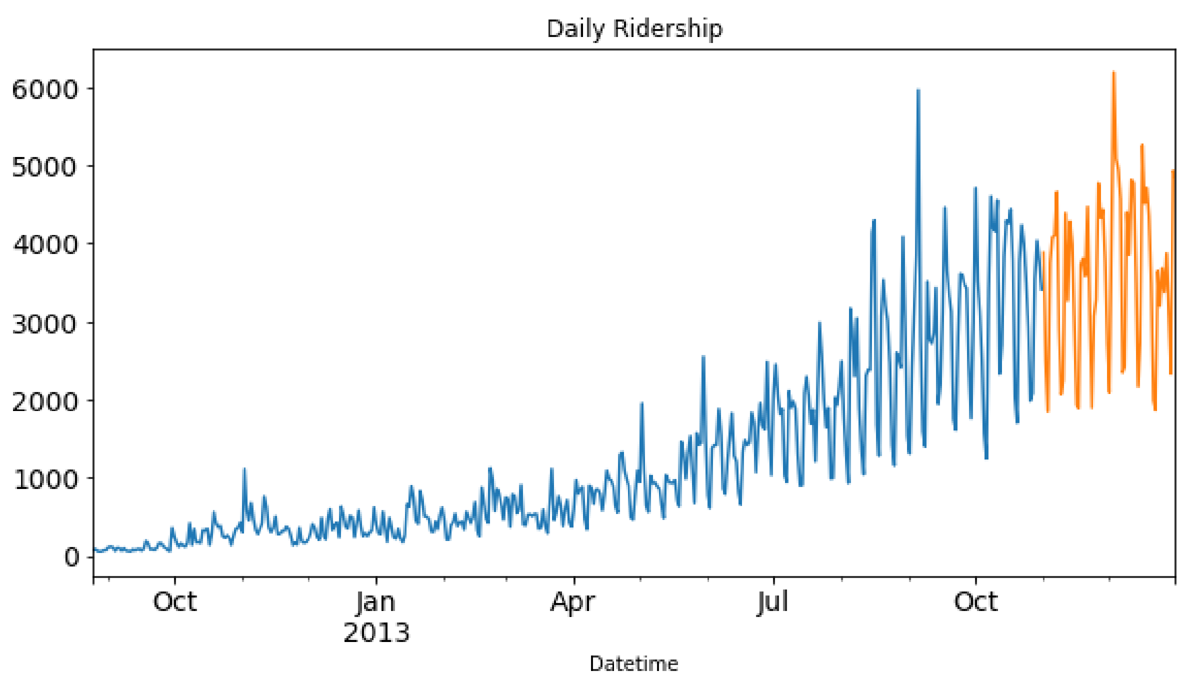
	ID	Count
Datetime		
2013-10-27	246804	2082
2013-10-28	247380	3536
2013-10-29	247956	4030
2013-10-30	248532	3774
2013-10-31	249108	3408

In [8]:
```
test=df.loc['2013-11-01':]
test.tail()
```

Out[8]:

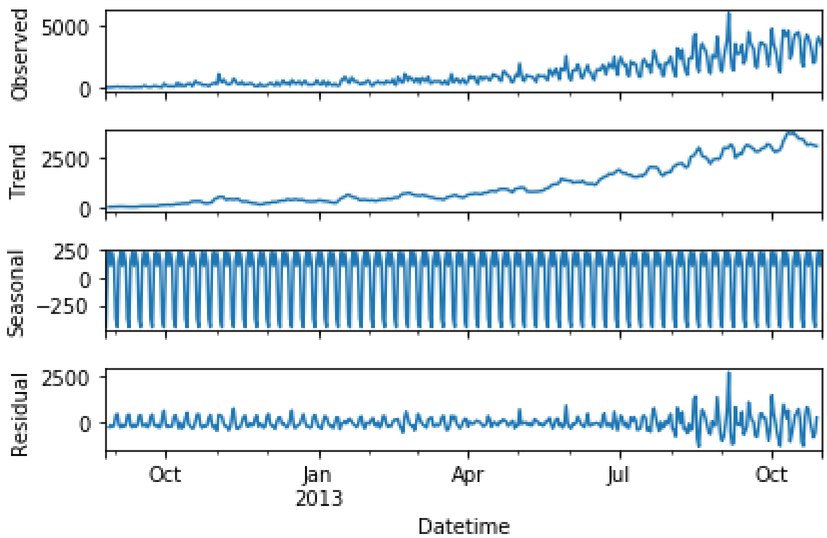
	ID	Count
Datetime		
2013-12-27	281940	3868
2013-12-28	282516	3084
2013-12-29	283092	2330
2013-12-30	283668	4928
2013-12-31	284244	4860

In [13]:
```
#Plotting data
train.Count.plot(figsize=(10,5), title= 'Daily Ridership', fontsize=14)
test.Count.plot(figsize=(10,5), title= 'Daily Ridership', fontsize=14)
plt.show()
```



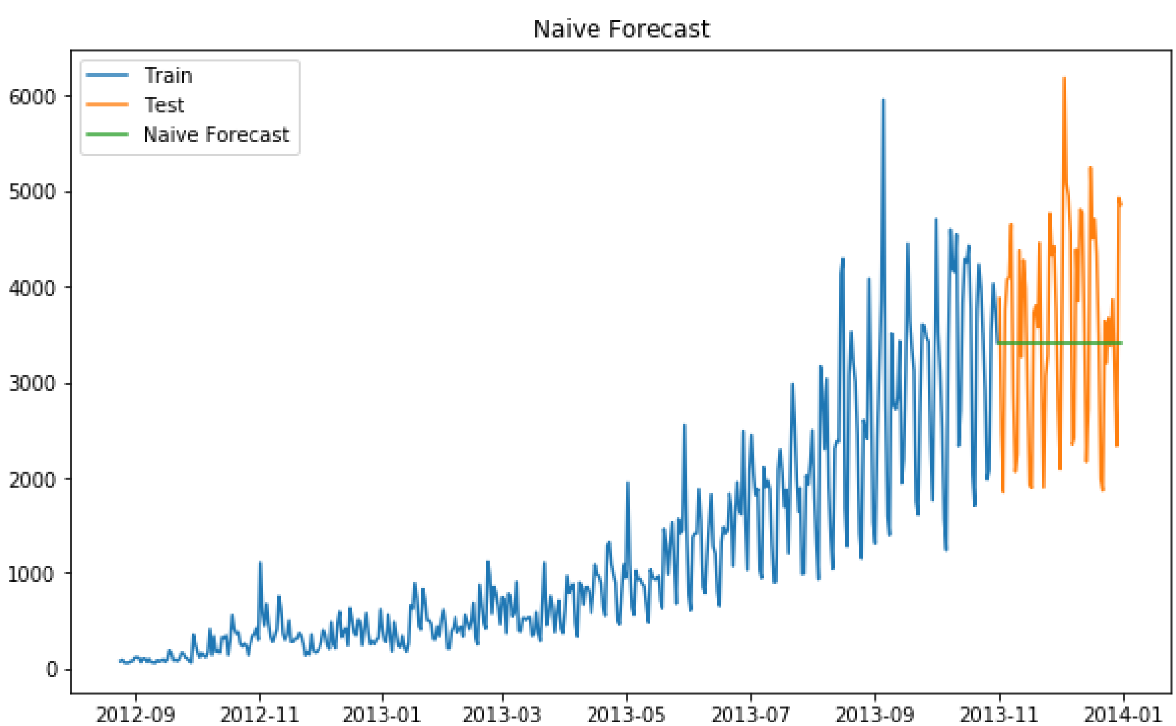
In [14]:

```
import statsmodels.api as sm
sm.tsa.seasonal_decompose(train.Count).plot()
plt.show()
```



In [15]:

```
dd= np.asarray(train.Count)
y_hat = test.copy()
#print(dd)
y_hat['naive'] = dd[-1] #3408.0
#print(y_hat.head())
plt.figure(figsize=(10,6))
plt.plot(train.index, train['Count'], label='Train')
plt.plot(test.index,test['Count'], label='Test')
plt.plot(y_hat.index,y_hat['naive'], label='Naive Forecast')
plt.legend(loc='best')
plt.title("Naive Forecast")
plt.show()
```

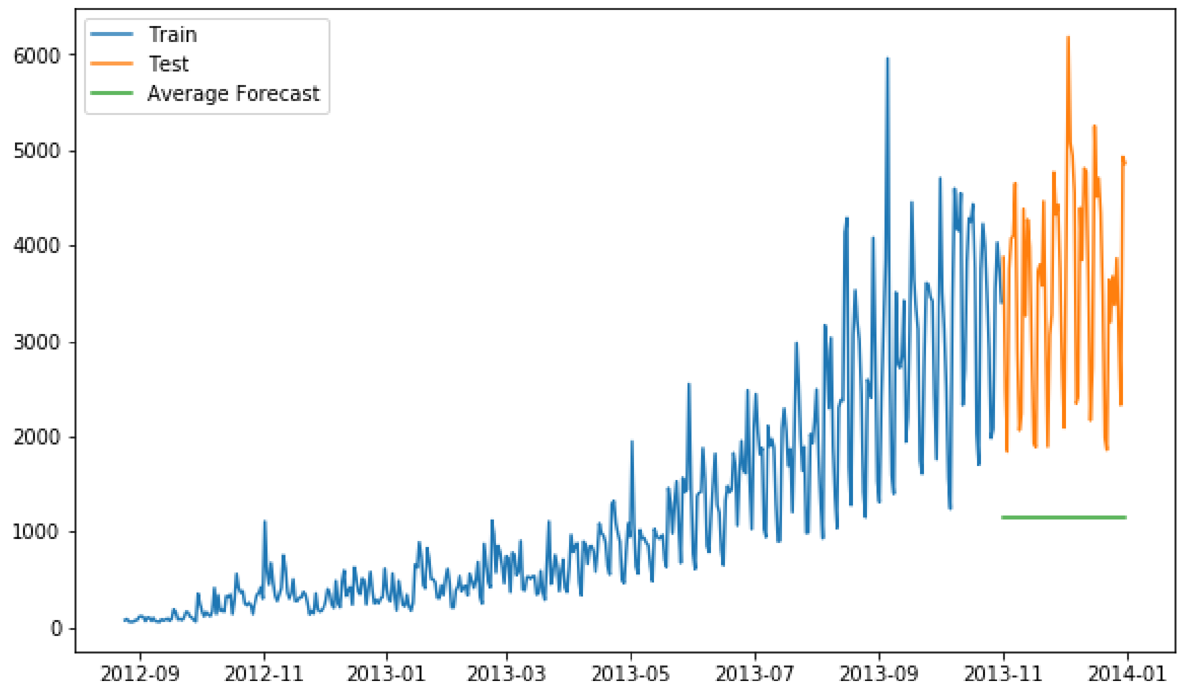


In [16]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat.Count, y_hat.naive))
print(rms)
```

1053.9937474540022

In [18]:
```
y_hat_avg = test.copy()
y_hat_avg['avg_forecast'] = train['Count'].mean() #1160.00
#print(y_hat_avg.head())

plt.figure(figsize=(10,6))
plt.plot(train['Count'], label='Train')
plt.plot(test['Count'], label='Test')
plt.plot(y_hat_avg['avg_forecast'], label='Average Forecast')
plt.legend(loc='best')
plt.show()
```



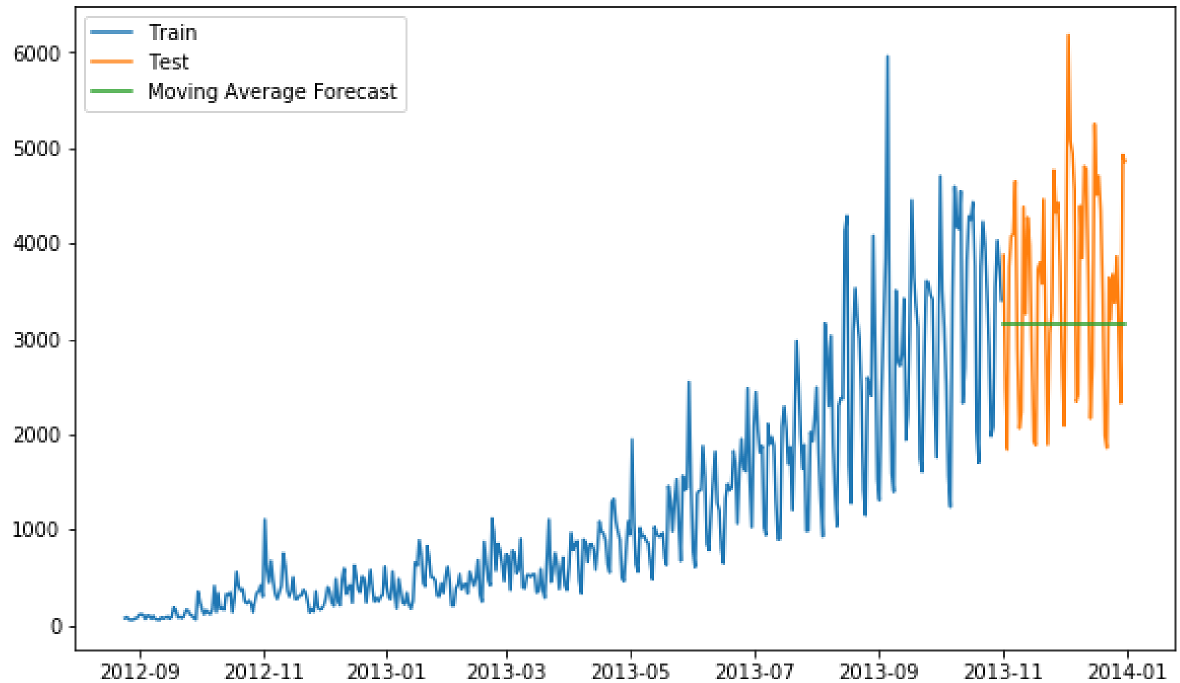
In [19]:
```
train['Count'].mean()
```

Out[19]: 1160.0092378752886

In [20]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat_avg.Count, y_hat_avg.avg_forecast))
print(rms)
```

2637.2463664998872

In [21]:
```
y_hat_avg = test.copy()
y_hat_avg['moving_avg_forecast'] = train['Count'].rolling(60).mean().iloc[-1]
#3162.
plt.figure(figsize=(10,6))
plt.plot(train['Count'], label='Train')
plt.plot(test['Count'], label='Test')
plt.plot(y_hat_avg['moving_avg_forecast'], label='Moving Average Forecast')
plt.legend(loc='best')
plt.show()
```

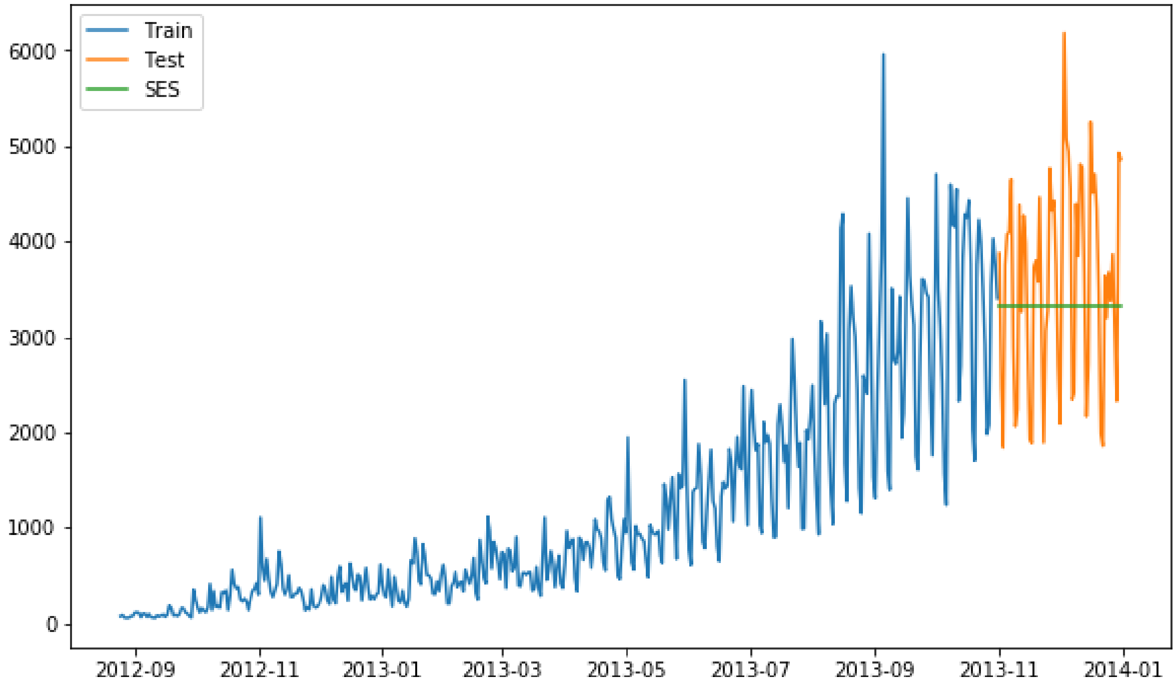


In [22]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat_avg.Count, y_hat_avg.moving_avg_forecast))
print(rms)
```

```
1121.4817740256713
```

In [23]:
```
from statsmodels.tsa.api import SimpleExpSmoothing
```

In [24]:
```
y_hat_avg = test.copy()
fit2 = SimpleExpSmoothing(np.asarray(train['Count'])).fit(smoothing_level=0.1)
y_hat_avg['SES'] = fit2.forecast(len(test))
plt.figure(figsize=(10,6))
plt.plot(train['Count'], label='Train')
plt.plot(test['Count'], label='Test')
plt.plot(y_hat_avg['SES'], label='SES')
plt.legend(loc='best')
plt.show()
```



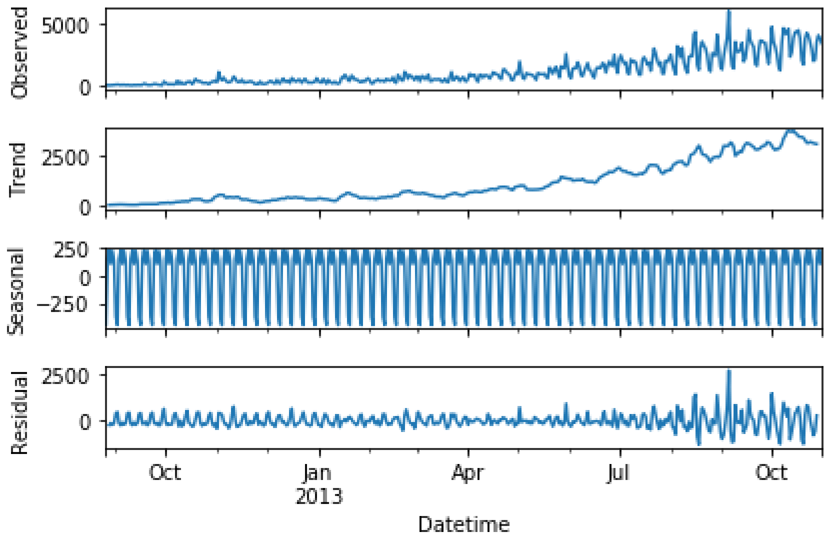
In [25]:
```
fit2.aic
```

Out[25]: 5436.843107674888

In [26]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat_avg.Count, y_hat_avg.SES))
print(rms)
```

1071.2120259800895

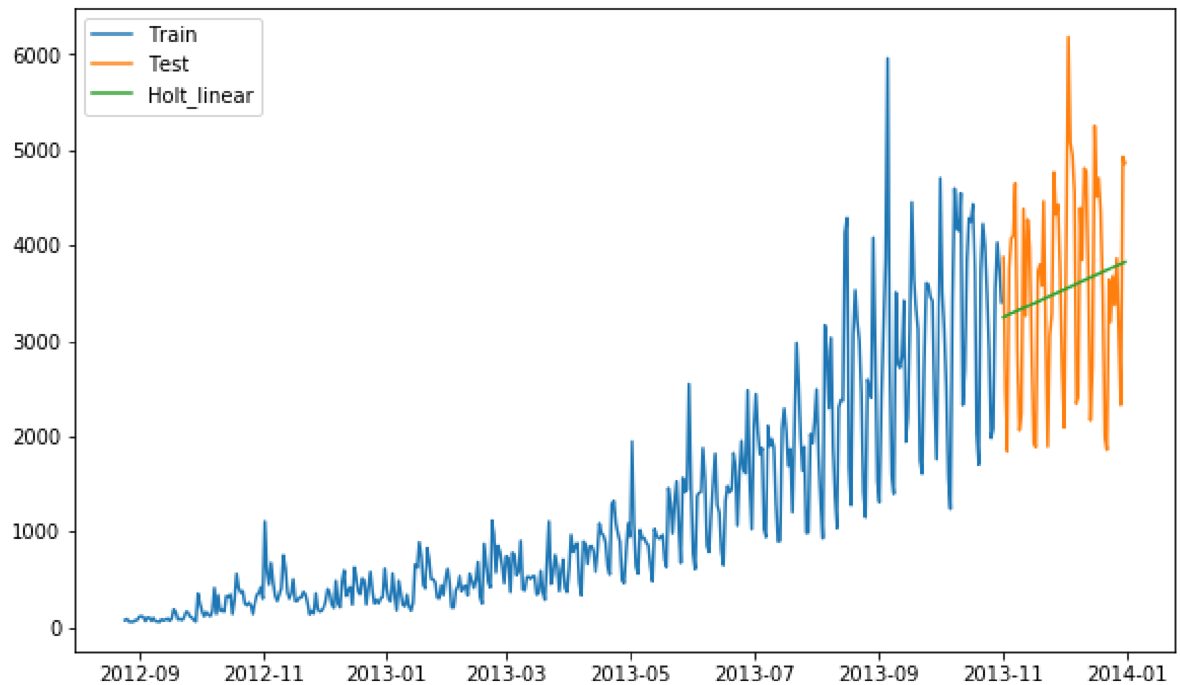
In [27]:
```
import statsmodels.api as sm
sm.tsa.seasonal_decompose(train.Count).plot()
plt.show()
```



In [28]:
```
from statsmodels.tsa.api import Holt
y_hat_avg = test.copy()

fit1 = Holt(np.asarray(train['Count'])).fit(smoothing_level = 0.23,smoothing_s
lope = 0.19)
y_hat_avg['Holt_linear'] = fit1.forecast(len(test))

plt.figure(figsize=(10,6))
plt.plot(train['Count'], label='Train')
plt.plot(test['Count'], label='Test')
plt.plot(y_hat_avg['Holt_linear'], label='Holt_linear')
plt.legend(loc='best')
plt.show()
```



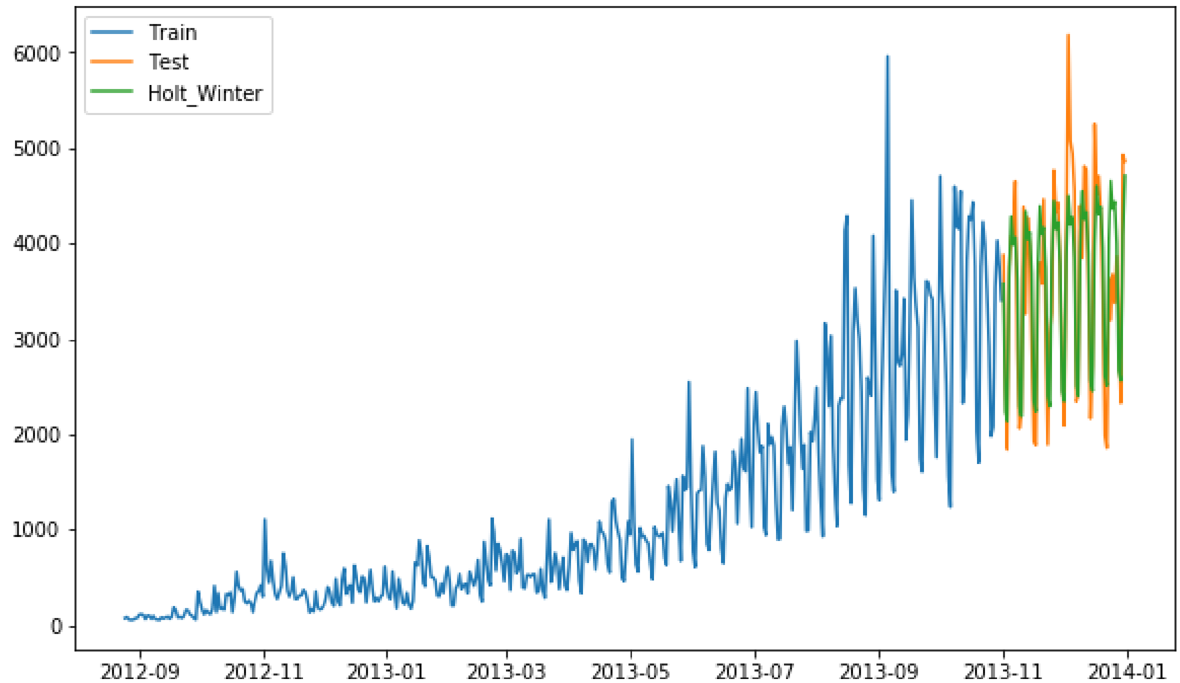
In [29]:
```
fit1.aic
```

Out[29]: 5509.380903223064

In [30]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat_avg.Count, y_hat_avg.Holt_linear))
print(rms)
```

1029.6846993193717

In [31]:
```
from statsmodels.tsa.api import ExponentialSmoothing
y_hat_avg = test.copy()
fit1 = ExponentialSmoothing(np.asarray(train['Count']) ,seasonal_periods=7
                            ,trend='add',
                            seasonal='add').fit()
y_hat_avg['Holt_Winter'] = fit1.forecast(len(test))
plt.figure(figsize=(10,6))
plt.plot( train['Count'], label='Train')
plt.plot(test['Count'], label='Test')
plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
plt.legend(loc='best')
plt.show()
```



In [32]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_hat_avg.Count, y_hat_avg.Holt_Winter))
print(rms)
```

```
575.0758215878233
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: