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

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

```
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The text.latex.preview rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The mathtext.fallback_to_cm rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases lat
er.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle: Support for setting the 'mathtext.fallback_to_cm' rcParam is deprecated since 3.3 and will
be removed two minor releases later; use 'mathtext.fallback : 'cm' instead.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The validate_bool_maybe_none function was deprecated in Matplotlib 3.3 and will be removed two minor releases l
ater.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The savefig.jpeg_quality rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The keymap.all_axes rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The animation.avconv_path rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later
.
In C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_cla
ssic_test.mplstyle:
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later
.
```

In [2]:
```
df=pd.read_excel("Collection LOB'S Day Level DF Data_21_10.xlsx")
```

In [3]:
```
df.head()
```

Out[3]:

	date	volume	WEEKDAY
0	2019-01-01	NaN	3
1	2019-01-02	557.0	4
2	2019-01-03	560.0	5
3	2019-01-04	791.0	6
4	2019-01-05	470.0	7

In [4]:
```
df.tail()
```

Out[4]:

	date	volume	WEEKDAY
560	2020-10-15	463.0	5
561	2020-10-16	550.0	6
562	2020-10-17	282.0	7
563	2020-10-19	530.0	2
564	2020-10-20	426.0	3

In [5]:
```
df.shape
```

Out[5]:
```
(565, 3)
```

In [6]:
```
# Check the data type of data columns
df.dtypes
```

Out[6]:
```
date       datetime64[ns]
volume            float64
WEEKDAY             int64
dtype: object
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['date', 'volume', 'WEEKDAY'], dtype='object')
```

In [8]:
```
df.describe()
```

Out[8]:

	volume	WEEKDAY
count	562.000000	565.000000
mean	451.606762	4.497345
std	149.556850	1.708991
min	0.000000	2.000000
25%	365.250000	3.000000
50%	470.000000	4.000000
75%	546.000000	6.000000
max	815.000000	7.000000

Data Cleaning

In [8]:
```
# Removing Weekday
# df_new=df[['date', 'volume']]
# df_new.head()
```

In [9]:
```
# Renaming Columns
# df_new.columns=['Date', 'Volume']
# df_new.head()
```

In [9]:
```
df_new=df
```

In [10]:
```
df_new.isnull().sum()
```

Out[10]:
```
date       0
volume     3
WEEKDAY    0
dtype: int64
```

In [11]:
```
# rows with missing values with weekday mean
df_new[df_new.isnull().any(axis=1)]
```

Out[11]:

	date	volume	WEEKDAY
0	2019-01-01	NaN	3
313	2020-01-01	NaN	4
521	2020-08-31	NaN	2

In [12]:
```
# Get weekday mean
df_new.groupby(['WEEKDAY'])['volume'].mean()
```

Out[12]:
```
WEEKDAY
2    520.817204
3    447.957447
4    482.333333
5    475.553191
6    531.819149
7    252.223404
Name: volume, dtype: float64
```

In [13]:
```
# Replace missing values with weekday mean
#df_new['volume']=df_new.groupby('WEEKDAY').transform(lambda x: x.fillna(x.mean()))
#df_new['volume']=df['volume'].fillna(df.groupby('WEEKDAY')['volume']).transform(lambda x: x.fillna(x.mean()))
df_new['volume']=df_new.groupby('WEEKDAY')['volume'].transform(lambda x: x.fillna(x.mean()))
```

In [14]:
```
df_new.isnull().sum()
```

Out[14]:
```
date       0
volume     0
WEEKDAY    0
dtype: int64
```

In [15]:
```
# Print specific rows
df_new.loc[[0,313,521]]
```

Out[15]:

	date	volume	WEEKDAY
0	2019-01-01	447.957447	3
313	2020-01-01	482.333333	4
521	2020-08-31	520.817204	2

In [16]:
```
df_new.dtypes
```

Out[16]:
```
date       datetime64[ns]
volume            float64
WEEKDAY             int64
dtype: object
```

In [17]:
```
# Convert date into Datetime format
import datetime
df_new['date']=pd.to_datetime(df_new['date'])
```

In [18]:
```
df_new.head()
```

Out[18]:

	date	volume	WEEKDAY
0	2019-01-01	447.957447	3
1	2019-01-02	557.000000	4
2	2019-01-03	560.000000	5
3	2019-01-04	791.000000	6
4	2019-01-05	470.000000	7

In [46]:
```
df_new.tail()
```

Out[46]:

date	volume	WEEKDAY	volume First Difference	Seasonal First Difference	forecast
2020-10-15	463.0	5	22.0	-26.0	-0.298755
2020-10-16	550.0	6	87.0	-125.0	-0.298755
2020-10-17	282.0	7	-268.0	53.0	-0.298755
2020-10-19	530.0	2	248.0	5.0	-0.298755
2020-10-20	426.0	3	-104.0	52.0	-0.298755

In [19]:
```
# Set Date column as Index
df_new.set_index('date', inplace=True)
df_new.head()
```

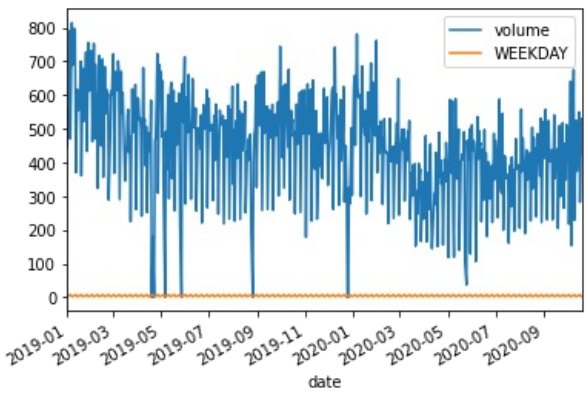
Out[19]:

date	volume	WEEKDAY
2019-01-01	447.957447	3
2019-01-02	557.000000	4
2019-01-03	560.000000	5
2019-01-04	791.000000	6
2019-01-05	470.000000	7

Visualize the data

In [20]:
```
df_new.plot()
```

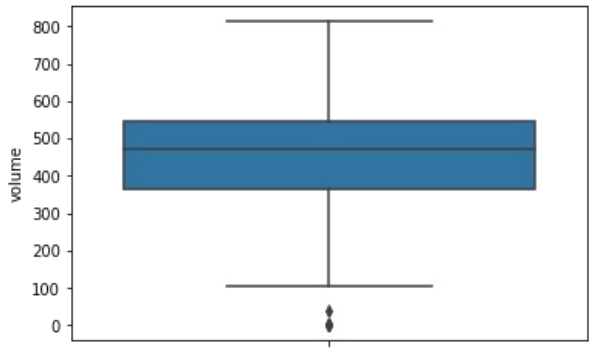
Out[20]:
```
<AxesSubplot:xlabel='date'>
```



Treating Outliers

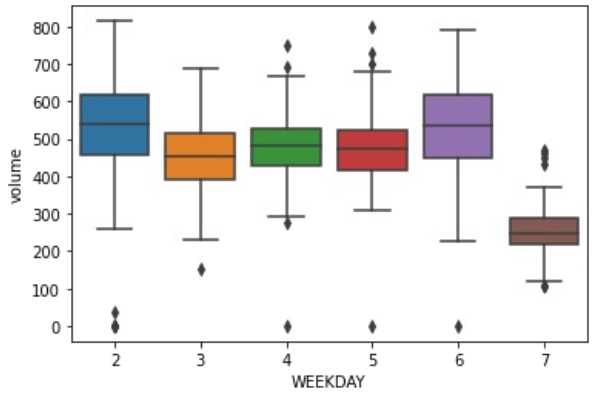
In [21]:
```
# Outlier Plot
import seaborn as sns
sns.boxplot(y=df_new['volume'])
```

Out[21]:
```
<AxesSubplot:ylabel='volume'>
```



In [22]:
```
# Vertical Boxplot grouped by Week
sns.boxplot(x='WEEKDAY', y='volume', data=df_new)
```

Out[22]:
```
<AxesSubplot:xlabel='WEEKDAY', ylabel='volume'>
```



In [23]:
```
df_new[df_new['volume']<130]
```

Out[23]:

date	volume	WEEKDAY
2019-04-19	0.0	6
2019-04-22	0.0	2
2019-05-06	0.0	2
2019-05-27	0.0	2
2019-08-26	0.0	2
2019-12-25	0.0	4
2019-12-26	0.0	5
2020-05-02	118.0	7
2020-05-09	119.0	7
2020-05-23	108.0	7
2020-05-25	37.0	2
2020-05-30	129.0	7
2020-06-06	106.0	7

In [24]:
```
df_new[df_new['volume']>750]
```

Out[24]:

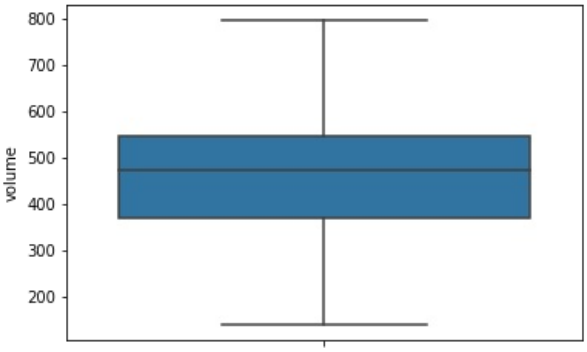
date	volume	WEEKDAY
2019-01-04	791.0	6
2019-01-07	815.0	2
2019-01-10	798.0	5
2019-01-11	783.0	6
2019-01-28	756.0	2
2019-02-04	753.0	2
2020-01-06	781.0	2
2020-01-31	762.0	6

In [25]:
```
# replacing outliers with weekday mean
#df_new['volume']=np.where(df_new['volume']==0, df_new.groupby('WEEKDAY')['volume'].transform(lambda x: x.filln
```

```
df_new['volume']=np.where(df_new['volume']<130, df_new.groupby('WEEKDAY')['volume'].transform(lambda x: np.wher
df_new['volume']=np.where(df_new['volume']>800, df_new.groupby('WEEKDAY')['volume'].transform(lambda x: np.wher
```

In [26]:
```
# After removing Outliers
sns.boxplot(y=df_new['volume'])
```

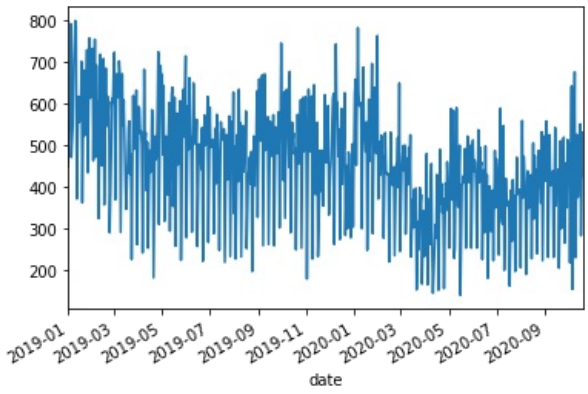
Out[26]:
```
<AxesSubplot:ylabel='volume'>
```



In [27]:
```
# df_new['volume']=np.where(df_new['volume']==0, np.where(df_new['WEEKDAY']==2, 520,
#                                                  np.where(df_new['WEEKDAY']==3, 447,
#                                                           np.where(df_new['WEEKDAY']==4,482,
#                                                                    np.where(df_new['WEEKDAY']==5, 475,
#                                                                             np.where(df_new['WEEKDAY']==6,
#                                                                                      np.where(df_new['WEEKD
```

In [28]:
```
# Plot After Removing Outliers
df_new['volume'].plot()
```

Out[28]:
```
<AxesSubplot:xlabel='date'>
```



Time Series

In [29]:
```
### Testing For Stationarity

from statsmodels.tsa.stattools import adfuller
```

In [30]:
```
test_result=adfuller(df_new['volume'])
```

In [31]:
```
#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(volume):
    result=adfuller(volume)
    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 ro
    else:
        print("weak evidence against null hypothesis, time series has a unit root, indicating it is non-station
```

In [32]:
```
adfuller_test(df_new['volume'])
```

```
ADF Test Statistic : -2.516127469329454
p-value : 0.11159948551730764
#Lags Used : 18
Number of Observations Used : 546
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Differencing

In [33]:
```
df_new['volume First Difference'] = df_new['volume'] - df_new['volume'].shift(1)
```

In [33]:
```
df_new['volume First Difference'] = df_new['volume'] - df_new['volume'].shift(1)
```

In [34]:
```
df_new['volume'].shift(1)
```

Out[34]:
```
date
2019-01-01           NaN
2019-01-02    447.957447
2019-01-03    557.000000
2019-01-04    560.000000
2019-01-05    791.000000
                 ...
2020-10-15    441.000000
2020-10-16    463.000000
2020-10-17    550.000000
2020-10-19    282.000000
2020-10-20    530.000000
Name: volume, Length: 565, dtype: float64
```

In [35]:
```
# For Seasonality
df_new['Seasonal First Difference']=df_new['volume']-df_new['volume'].shift(6)
```

In [36]:
```
df_new.head()
```

Out[36]:

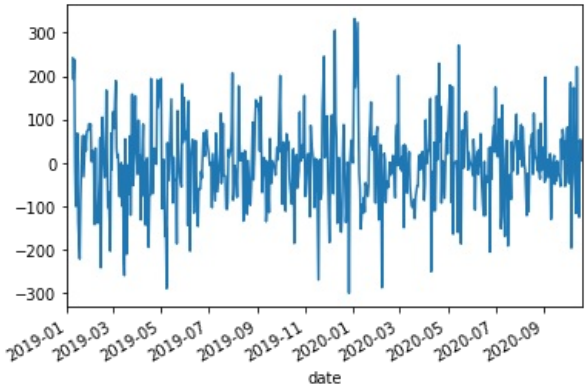
date	volume	WEEKDAY	volume First Difference	Seasonal First Difference
2019-01-01	447.957447	3	NaN	NaN
2019-01-02	557.000000	4	109.042553	NaN
2019-01-03	560.000000	5	3.000000	NaN
2019-01-04	791.000000	6	231.000000	NaN
2019-01-05	470.000000	7	-321.000000	NaN

In [37]:
```
# Again test dickey fuller test
adfuller_test(df_new['Seasonal First Difference'].dropna())
```

```
ADF Test Statistic : -9.490097179554372
p-value : 3.6723739424988657e-16
#Lags Used : 17
Number of Observations Used : 541
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stati
onary
```

In [38]:
```
df['Seasonal First Difference'].plot()
```

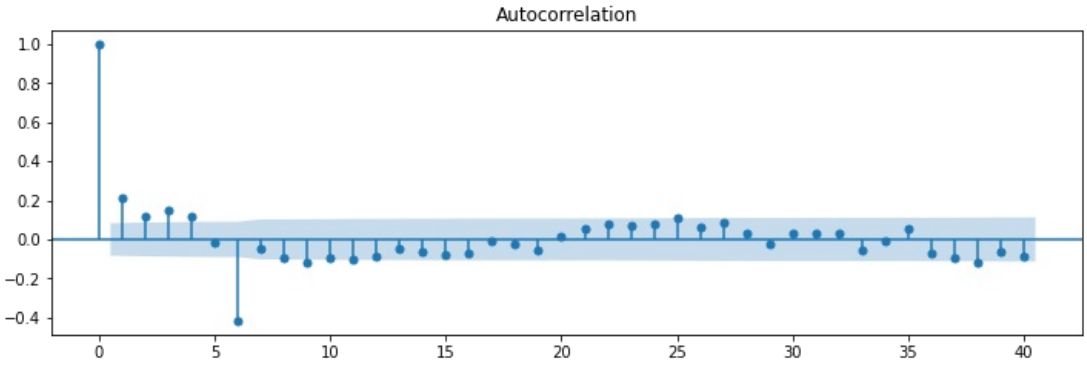
Out[38]:
```
<AxesSubplot:xlabel='date'>
```

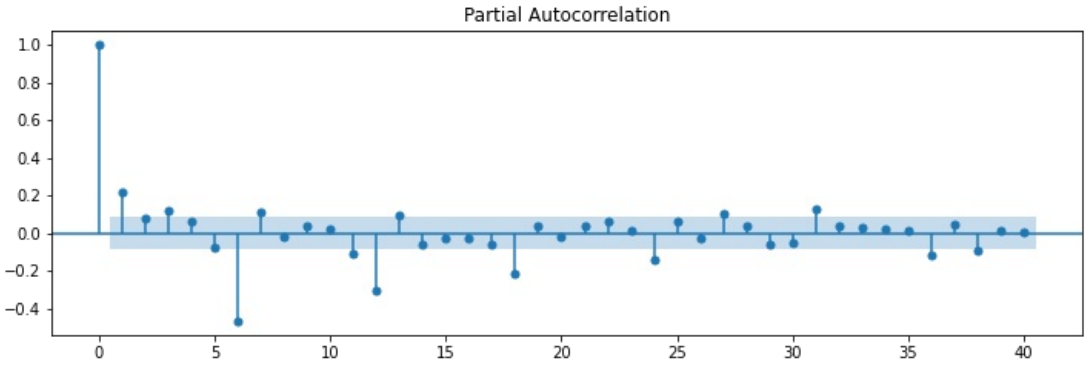


Auto Regressive Model

In [39]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
```

In [40]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Seasonal First Difference'].iloc[6:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Seasonal First Difference'].iloc[6:],lags=40,ax=ax2)
```





In [41]:
```
# For non-seasonal data
#p=1, d=1, q=0 or 1
from statsmodels.tsa.arima_model import ARIMA
```

In [42]:
```
model=ARIMA(df_new['volume'],order=(1,1,1))
model_fit=model.fit()
```

```
C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:219: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:219: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

In [43]:
```
model_fit.summary()
```

Out[43]:

ARIMA Model Results

Dep. Variable:	D.volume	**No. Observations:**	564
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-3481.658
Method:	css-mle	**S.D. of innovations**	115.844
Date:	Thu, 05 Nov 2020	**AIC**	6971.316
Time:	19:10:22	**BIC**	6988.656
Sample:	1	**HQIC**	6978.084

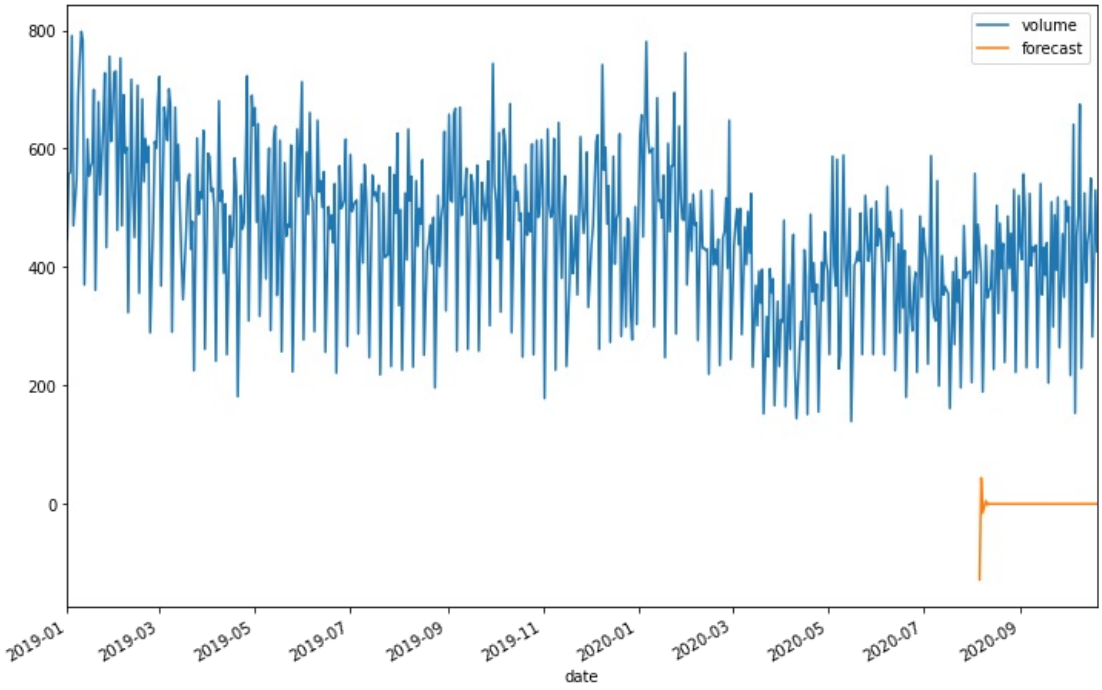
	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.2988	0.410	-0.730	0.466	-1.101	0.504
ar.L1.D.volume	-0.3443	0.043	-8.035	0.000	-0.428	-0.260
ma.L1.D.volume	-0.8888	0.025	-35.906	0.000	-0.937	-0.840

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-2.9046	+0.0000j	2.9046	0.5000
MA.1	1.1251	+0.0000j	1.1251	0.0000

In [44]:
```
# Without considering seasonality ARIMA prediction
df_new['forecast']=model_fit.predict(start=500,end=564,dynamic=True)
df_new[['volume','forecast']].plot(figsize=(12,8))
```

Out[44]:
```
<AxesSubplot:xlabel='date'>
```



SARIMA on Dialler Freestyle Data

In [45]:
```
# Considering Seasonality in Arima it becomes SARIMA
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(df_new['volume'],order=(1, 1, 1),seasonal_order=(1,1,1,6))
results=model.fit()
```

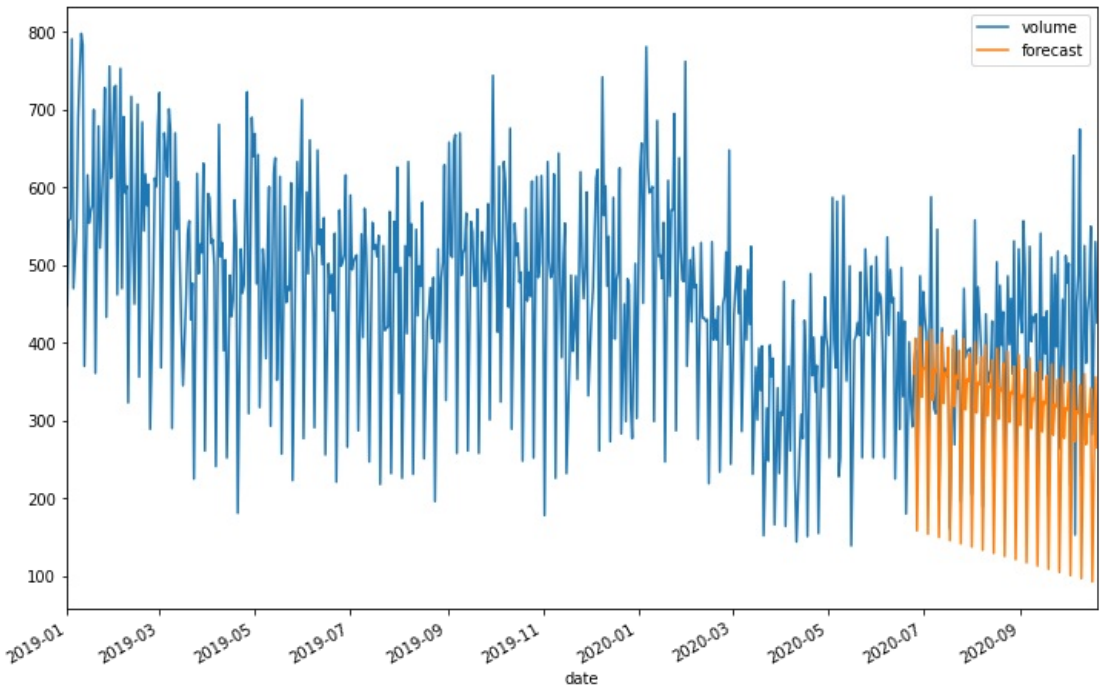
```
C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:219: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

In [47]:
```
565-100
```

Out[47]:
```
465
```

In [50]:
```
df_new['forecast']=results.predict(start=464,end=564,dynamic=True)
df_new[['volume','forecast']].plot(figsize=(12,8))
```

Out[50]:
```
<AxesSubplot:xlabel='date'>
```



In [51]:
```
df_new['forecast'][495:505]
```

Out[51]:
```
date
2020-07-31    386.168385
2020-08-01    137.502283
2020-08-03    400.921460
2020-08-04    310.019730
2020-08-05    348.996234
2020-08-06    346.360795
2020-08-07    382.095961
2020-08-08    133.429859
2020-08-10    396.849036
2020-08-11    305.947307
Name: forecast, dtype: float64
```

In [52]:
```
df_new['forecast'][464:]
```

Out[52]:
```
date
2020-06-25    360.131315
2020-06-26    405.925703
2020-06-27    158.224150
2020-06-29    420.623563
2020-06-30    330.102813
                 ...
2020-10-15    305.636559
2020-10-16    341.371726
2020-10-17     92.705624
2020-10-19    356.124801
2020-10-20    265.223071
Name: forecast, Length: 101, dtype: float64
```

In [53]:
```
# Writing it to a file as csv
df_new['forecast'][464:].to_csv('SARIMA on DF100daystest.csv')
```

```
C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarni
ng: The signature of `Series.to_csv` was aligned to that of `DataFrame.to_csv`, and argument 'header' will chan
ge its default value from False to True: please pass an explicit value to suppress this warning.
```

Future prediction

In [174…
```
from pandas.tseries.offsets import DateOffset
future_dates=[df_new.index[-1]+ DateOffset(days=x)for x in range(0,90)]
```

In [163…
```
future_datest_df=pd.DataFrame(index=future_dates[1:],columns=df_new.columns)
```

In [164…
```
future_datest_df.head()
```

Out[164]:

	volume	WEEKDAY	volume First Difference	Seasonal First Difference	forecast
2020-10-21	NaN	NaN	NaN	NaN	NaN
2020-10-22	NaN	NaN	NaN	NaN	NaN
2020-10-23	NaN	NaN	NaN	NaN	NaN
2020-10-24	NaN	NaN	NaN	NaN	NaN
2020-10-25	NaN	NaN	NaN	NaN	NaN

In [165…
```
future_datest_df.shape
```

Out[165]:
```
(89, 5)
```

In [213…
```
future_df=pd.concat([df_new,future_datest_df])
```

In [214…
```
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(df_new['volume'],order=(1, 1, 1),seasonal_order=(1,1,1,6))
results=model.fit()
```

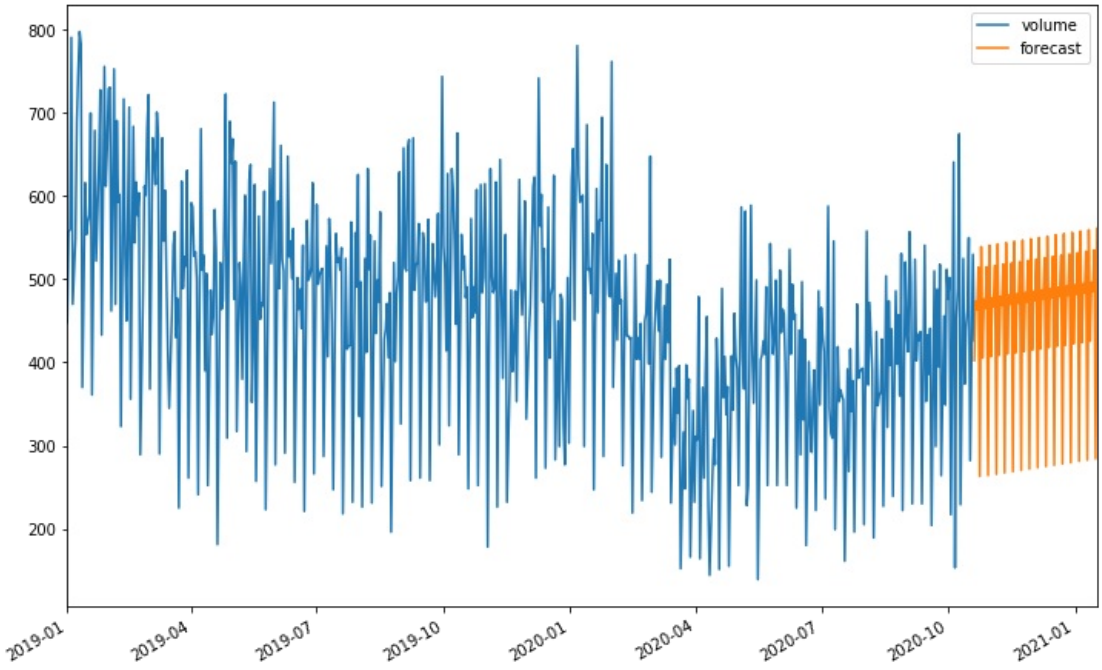
```
C:\Users\suma.s.huddar\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:21
9: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be
ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

In [1]:
```
564+89
```

Out[1]:
```
653
```

In [221…
```
future_df['forecast'] = results.predict(start = 564,end = 653, dynamic= True)
future_df[['volume', 'forecast']].plot(figsize=(12, 8))
```

Out[221]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d726cae208>
```



In [222…
```
future_df['forecast'].shape
```

Out[222]:
```
(743,)
```

In [224…
```
future_df[560:654]
```

Out[224]:

	volume	WEEKDAY	volume First Difference	Seasonal First Difference	forecast
2020-10-15	463.0	5	22.0	-26.0	NaN
2020-10-16	550.0	6	87.0	-125.0	NaN
2020-10-17	282.0	7	-268.0	53.0	NaN
2020-10-19	530.0	2	248.0	5.0	NaN
2020-10-20	426.0	3	-104.0	52.0	402.367095
...	...	...	...	...	...
2021-01-13	NaN	NaN	NaN	NaN	496.350959
2021-01-14	NaN	NaN	NaN	NaN	485.594053
2021-01-15	NaN	NaN	NaN	NaN	535.156204
2021-01-16	NaN	NaN	NaN	NaN	284.152444
2021-01-17	NaN	NaN	NaN	NaN	561.129468

94 rows × 5 columns

In [181…
```
565+89
```

Out[181]:
```
654
```

In [228…
```
future_df[564:577]
```

Out[228]:

	volume	WEEKDAY	volume First Difference	Seasonal First Difference	forecast
2019-01-01	447.957447	3	NaN	NaN	NaN
2019-01-02	557.000000	4	109.042553	NaN	NaN
2019-01-03	560.000000	5	3.000000	NaN	NaN
2019-01-04	791.000000	6	231.000000	NaN	NaN
2019-01-05	470.000000	7	-321.000000	NaN	NaN

Extracting Forecasted Values

In [230…
```
# Forecasted values for 89 days
future_df['forecast'][565:654]
```

Out[230]:
```
2020-10-21    473.777410
2020-10-22    463.947700
2020-10-23    514.479859
2020-10-24    263.041335
2020-10-25    539.275343
                 ...
2021-01-13    496.350959
2021-01-14    485.594053
2021-01-15    535.156204
2021-01-16    284.152444
2021-01-17    561.129468
Name: forecast, Length: 89, dtype: float64
```

In [233…
```
# Writing it to a file as csv
future_df[565:654].to_csv('ARIMA_and_SARIMA on DF.csv')
```

In [234…
```
import os
os.getcwd()
```

Out[234]:
```
'C:\\Users\\suma.s.huddar\\Documents\\Python Scripts\\Dialler FreeStyle'
```

In [ ]:
```

```