In [1]:

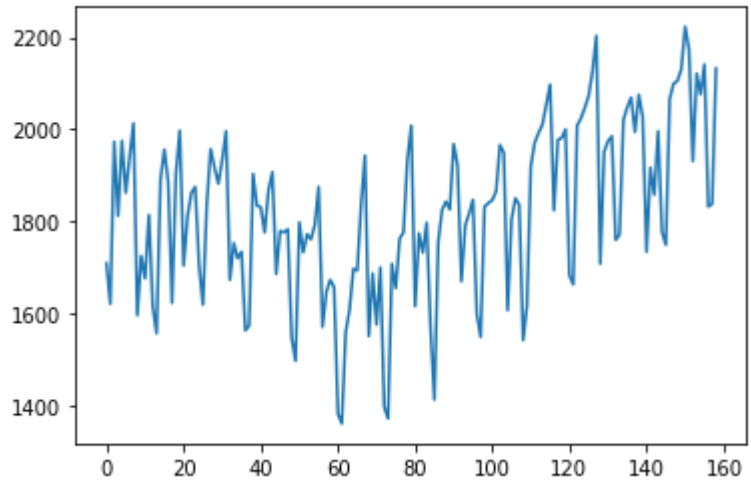
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [2]:

```
Walmart = pd.read_csv("footfalls.csv")
Walmart.Footfalls.plot()
```

Out[2]:

<AxesSubplot:>



Splitting data

In [3]:

```
Train = Walmart.head(147)
Test = Walmart.tail(12)
```

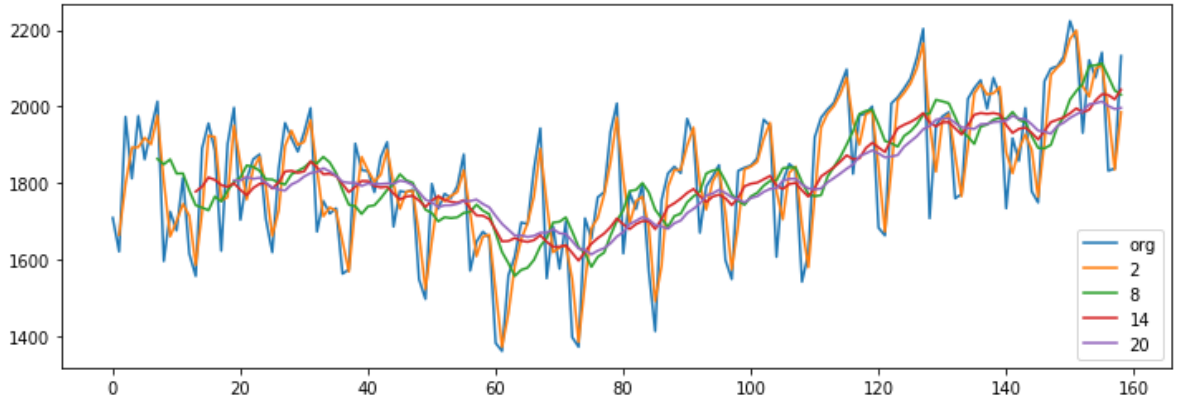
Moving Average

In [4]:

```
plt.figure(figsize=(12,4))
Walmart.Footfalls.plot(label="org")
for i in range(2,24,6):
    Walmart["Footfalls"].rolling(i).mean().plot(label=str(i))
plt.legend(loc='best')
```

Out[4]:

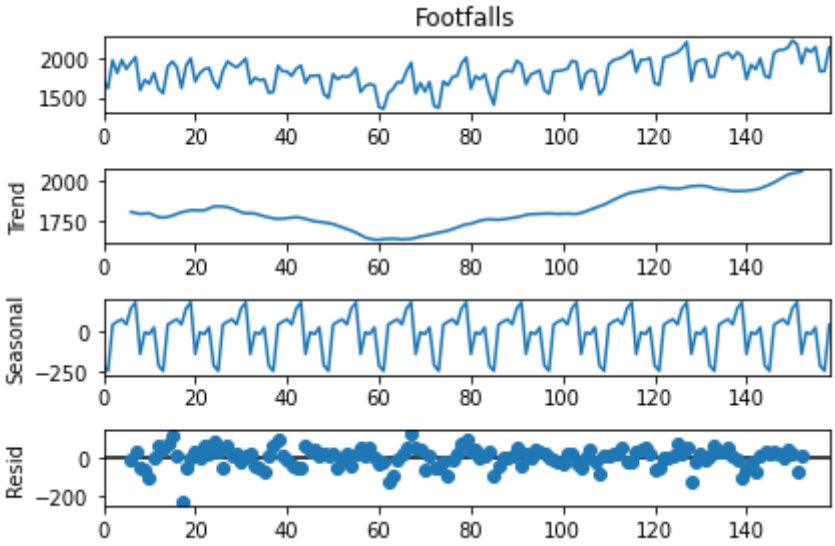
```
<matplotlib.legend.Legend at 0x11f701210>
```



Time series decomposition plot

In [5]:

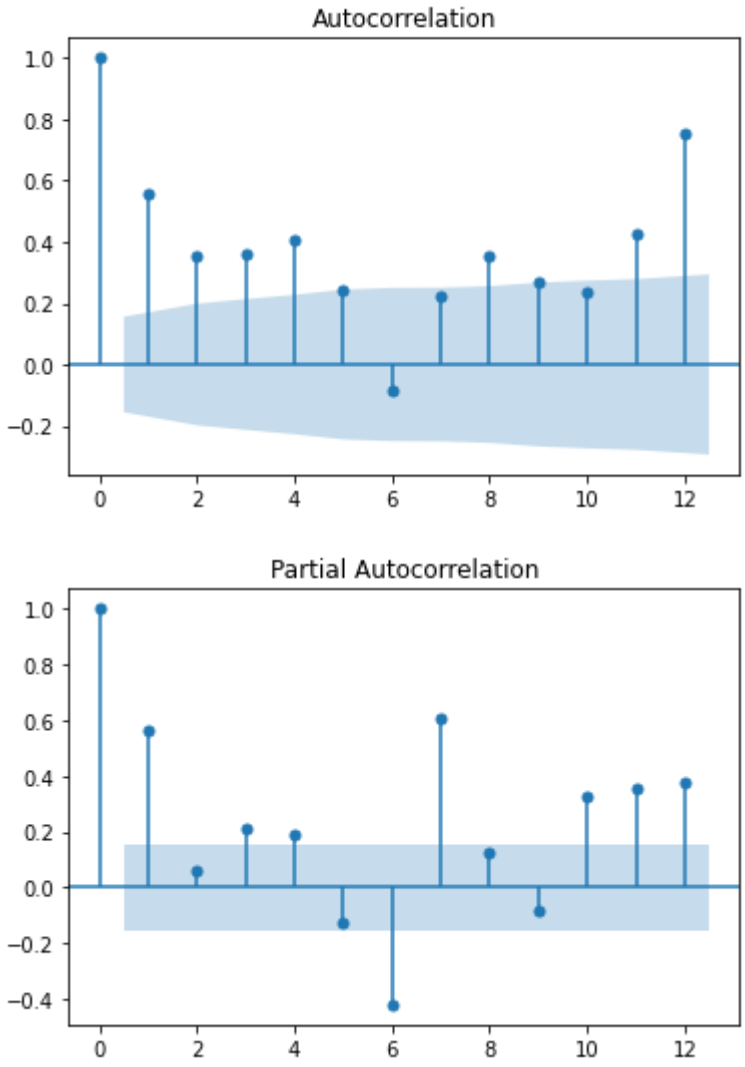
```
decompose_ts_add = seasonal_decompose(Walmart.Footfalls,period=12)
decompose_ts_add.plot()
plt.show()
```



ACF plots and PACF plots

In [6]:

```
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(Walmart.Footfalls,lags=12)
tsa_plots.plot_pacf(Walmart.Footfalls,lags=12)
plt.show()
```



Evaluation Metric MAPE

In [7]:

```
def MAPE(pred,org):
    temp = np.abs((pred-org)/org)*100
    return np.mean(temp)
```

Simple Exponential Method

In [8]:

```
ses_model = SimpleExpSmoothing(Train["Footfalls"]).fit(smoothing_level=0.2)
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_ses,Test.Footfalls)
```

Out[8]:

```
8.490637057828362
```

Holt method

In [9]:

```
# Holt method
hw_model = Holt(Train["Footfalls"]).fit(smoothing_level=0.8, smoothing_slope=0.2)
pred_hw = hw_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_hw,Test.Footfalls)
```

Out[9]:

```
7.546249229496066
```

Holts winter exponential smoothing with additive seasonality and additive trend

In [10]:

```
hwe_model_add_add = ExponentialSmoothing(Train["Footfalls"],seasonal="add",trend="add",seas
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_hwe_add_add,Test.Footfalls)
```

```
/Users/srinivasgurrala/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/tsa/holtwinters.py:744: ConvergenceWarning: Optimization failed to converg
e. Check mle_retvals.
  ConvergenceWarning)
```

Out[10]:

```
3.094698688707736
```

Holts winter exponential smoothing with multiplicative seasonality and additive trend

In [11]:

```
hwe_model_mul_add = ExponentialSmoothing(Train["Footfalls"],seasonal="mul",trend="add",seas
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_hwe_mul_add,Test.Footfalls)
```

Out[11]:

```
3.1183252705860736
```

Final Model by combining train and test

In [12]:

```
hwe_model_add_add = ExponentialSmoothing(Walmart["Footfalls"],seasonal="add",trend="add",se
```

```
/Users/srinivasgurrala/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/tsa/holtwinters.py:744: ConvergenceWarning: Optimization failed to converg
e. Check mle_retvals.
  ConvergenceWarning)
```

In [13]:

```
#Forecasting for next 10 time periods
hwe_model_add_add.forecast(10)
```

Out[13]:

```
159    2138.461839
160    2167.954404
161    2131.322143
162    2234.340471
163    2272.568439
164    1946.715567
165    2090.642628
166    2077.783348
167    2123.698233
168    1880.317615
dtype: float64
```

In [ ]:

```

```