In [2]:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import itertools
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.api as sm
```

In [3]:

```
def check_stationarity(series):
    result = adfuller(series)
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    if result[1]<0.05:
        print('Time Series is stationary')
    else:
        print('Time Series is not stationary')
```

In [4]:

```
def series_decomposition(series, method='additive'):
    result = seasonal_decompose(series, model=method)
    result.plot()
    plt.show()
```

In [5]:

```
def plot_acf_pacf_graphs(series):
    fig, ax = plt.subplots(2,1)
    fig = sm.graphics.tsa.plot_acf(series, lags=25, ax=ax[0])
    fig = sm.graphics.tsa.plot_pacf(series, lags=25, ax=ax[1])
    plt.tight_layout()
    plt.show()
```

In [6]:

```
def arima_modeling(series, params):
    mod = sm.tsa.arima.ARIMA(series,order=params)
    results = mod.fit()
    print('ARIMA{} - AIC:{}'.format(params,results.aic))
    print(results.summary())
    results.plot_diagnostics(figsize=(18, 8))
    plt.show()
```

In [7]:

```
def arima_prediction(series, params, start_point):
    model = sm.tsa.arima.ARIMA(series,order=params).fit()
    pred = model.get_prediction(start=start_point, dynamic=False)
    pred_ci = pred.conf_int()
    ax = series.plot(label='observed')
    pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(
14, 4))
    ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', al
pha=.2)
    ax.set_xlabel('Date')
    ax.set_ylabel('Quantity')
    plt.legend()
```

```
    plt.show()
```

In [8]:

```
def arima_walk_forward_validation(series, params, test_size):
    n_train = int(len(series) * (1-test_size))
    train, test = series.values[0:n_train], series.values[n_train:len(series)]
    history = [x for x in train]
    predictions = list()
    # walk-forward validation
    for t in range(len(test)):
        model = sm.tsa.arima.ARIMA(history, order=params)
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
    # evaluate forecasts
    rmse = np.sqrt(mean_squared_error(test, predictions))
    print('Test RMSE: %.3f' % rmse)
    # plot forecasts against actual outcomes
    plt.plot(test)
    plt.plot(predictions, color='red')
    plt.show()
```

In [9]:

```
def arima_walk_forward_forecast(series, params, steps=5):
    history = series.copy()
    predictions = [history.iloc[-1]]
    predictions_ci_min = [history.iloc[-1]]
    predictions_ci_max = [history.iloc[-1]]
    predictions_ci_index = [history.index[-1]]
    for t in range(steps):
        model = sm.tsa.arima.ARIMA(history, order=params)
        model_fit = model.fit()
        predictions.append(model_fit.get_forecast().predicted_mean[0])
        predictions_ci_min.append(model_fit.get_forecast().conf_int().values[0,0])
        predictions_ci_max.append(model_fit.get_forecast().conf_int().values[0,1])
        predictions_ci_index.append(model_fit.get_forecast().conf_int().index.tolist()[0
])
        history = history.append(model_fit.get_forecast().predicted_mean)
    plt.figure(figsize=(14, 4))
    plt.plot(predictions_ci_index, predictions, label='Walk-Forward ahead Forecast', alp
ha=.7, color='red')
    plt.plot(series, label='observed', color='blue')
    plt.fill_between(predictions_ci_index, predictions_ci_min, predictions_ci_max, color=
'k', alpha=.2)
    plt.xlabel('Date')
    plt.ylabel('Quantity')
    plt.legend()
    plt.show()
```

In [10]:

```
def sarimax_modeling(series, params, s_params):
    model = sm.tsa.statespace.SARIMAX(series, order=params,
                                      seasonal_order=s_params).fit(max_iter=50, method='
powell')
    print('SARIMAX{}x{} - AIC:{}'.format(params, s_params, model.aic))
    print(model.summary())
    model.plot_diagnostics(figsize=(18, 8))
    plt.show()
```

In [11]:

```
def sarimax_prediction(series, params, s_params, start_point):
    model = sm.tsa.statespace.SARIMAX(series, order=params,
                                      seasonal_order=s_params).fit(max_iter=50, method='
powell', disp=False)
```

```
    pred = model.get_prediction(start=start_point, dynamic=False)
    pred_ci = pred.conf_int()
    ax = series.plot(label='observed')
    pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(
14, 4))
    ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', al
pha=.2)
    ax.set_xlabel('Date')
    ax.set_ylabel('Quantity')
    plt.legend()
    plt.show()
```

In [12]:

```
def sarimax_walk_forward_validation(series, params, s_params, test_size):
    n_train = int(len(series) * (1-test_size))
    train, test = series.values[0:n_train], series.values[n_train:len(series)]
    history = [x for x in train]
    predictions = list()
    # walk-forward validation
    for t in range(len(test)):
        model = sm.tsa.statespace.SARIMAX(history, order=params, seasonal_order=s_params
)
        model_fit = model.fit(max_iter=50, method='powell', disp=False)
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
    # evaluate forecasts
    rmse = np.sqrt(mean_squared_error(test, predictions))
    print('Test RMSE: %.3f' % rmse)
    # plot forecasts against actual outcomes
    plt.plot(test)
    plt.plot(predictions, color='red')
    plt.show()
```

In [13]:

```
def sarimax_walk_forward_forecast(series, params, s_params, steps=5):
    history = series.copy()
    predictions = [history.iloc[-1]]
    predictions_ci_min = [history.iloc[-1]]
    predictions_ci_max = [history.iloc[-1]]
    predictions_ci_index = [history.index[-1]]
    for t in range(steps):
        model = sm.tsa.statespace.SARIMAX(history, order=params, seasonal_order=s_params
)
        model_fit = model.fit(max_iter=50, method='powell', disp=False)
        predictions.append(model_fit.get_forecast().predicted_mean[0])
        predictions_ci_min.append(model_fit.get_forecast().conf_int().values[0,0])
        predictions_ci_max.append(model_fit.get_forecast().conf_int().values[0,1])
        predictions_ci_index.append(model_fit.get_forecast().conf_int().index.tolist()[0
])
        history = history.append(model_fit.get_forecast().predicted_mean)
    plt.figure(figsize=(14, 4))
    plt.plot(predictions_ci_index, predictions, label='Walk-Forward ahead Forecast', alp
ha=.7, color='red')
    plt.plot(series, label='observed', color='blue')
    plt.fill_between(predictions_ci_index, predictions_ci_min, predictions_ci_max, color=
'k', alpha=.2)
    plt.xlabel('Date')
    plt.ylabel('Quantity')
    plt.legend()
    plt.show()
```

(ARIMA)when dealing with non-seasonal time series data without a repeating pattern at fixed intervals.

In [14]:

```
df = pd.read_csv('Annual_changes_in_the_earths_rotation_day_length_sec105_18211970.csv')
```

```
df.head()
```

Out[14]:

	Unnamed: 0	x
0	1	-217
1	2	-177
2	3	-166
3	4	-136
4	5	-110

In [15]:

```
df.drop(columns=['Unnamed: 0'], inplace=True)
```

In [16]:

```
len(pd.date_range('1821-01-01','1971-01-01' , freq='1Y')), len(df)
```

Out[16]:

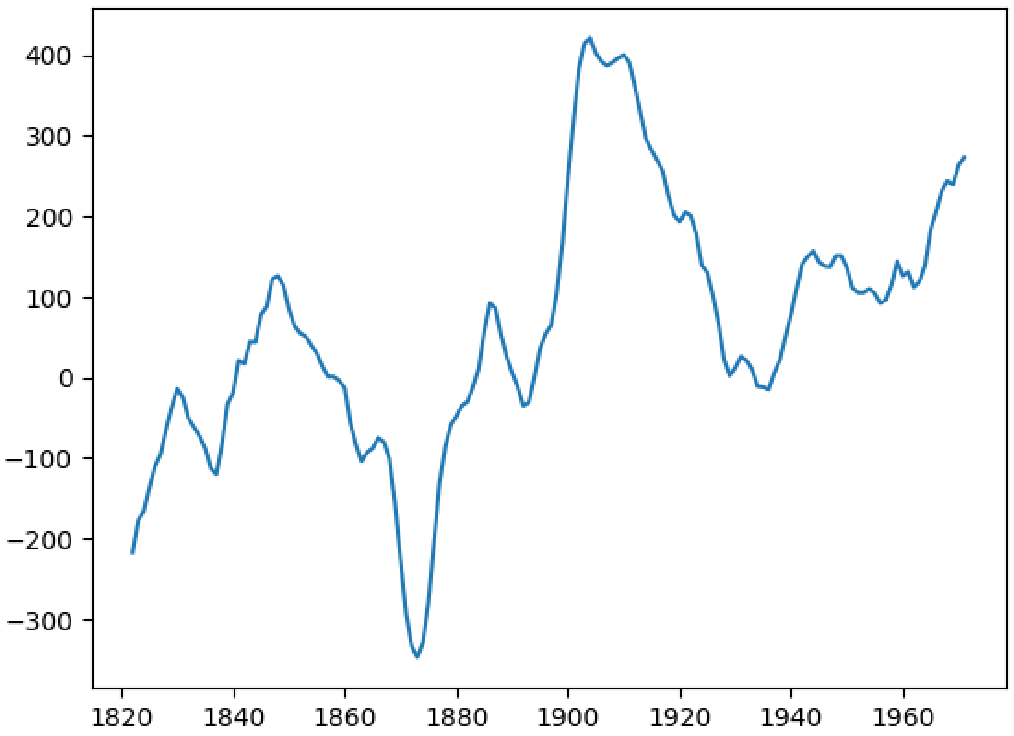
```
(150, 150)
```

In [17]:

```
df['date'] = pd.date_range('1821-01-01','1971-01-01', freq='1Y')
df = df.set_index('date')
df.index = pd.DatetimeIndex(df.index.values, freq=df.index.inferred_freq)
```

In [18]:

```
plt.plot(df['x'])
plt.show()
```



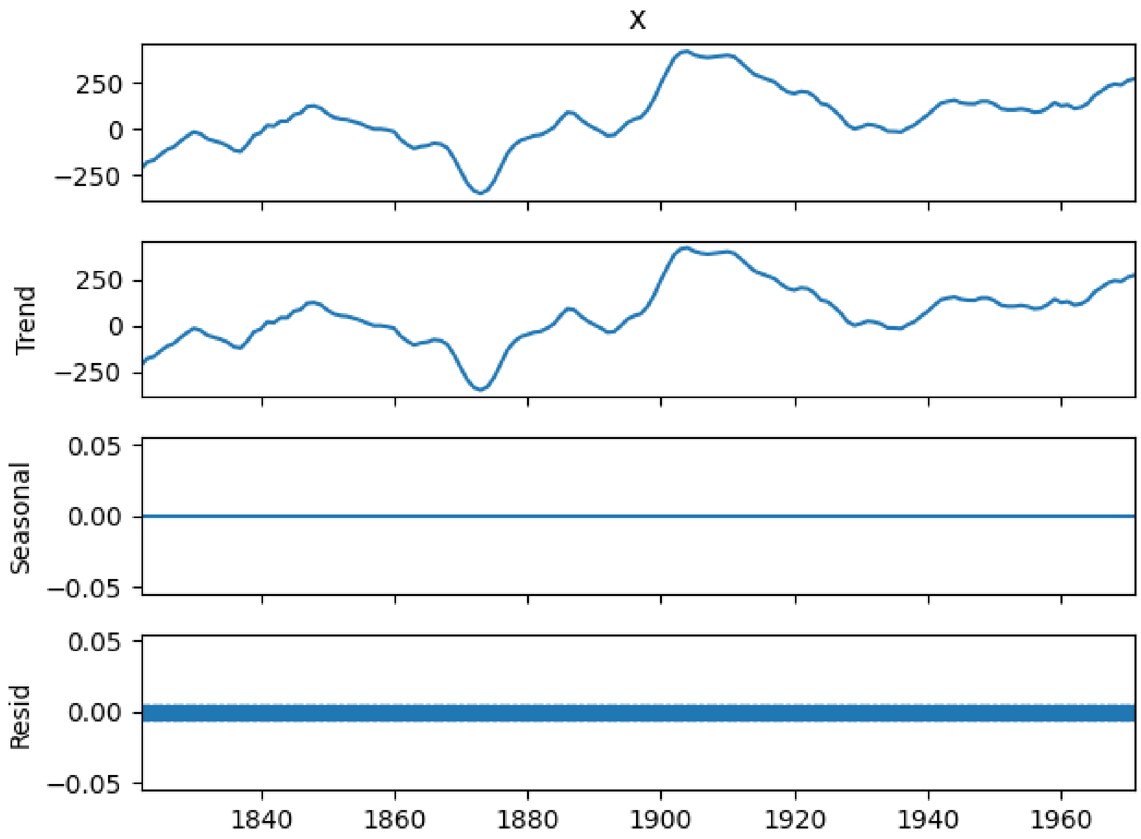
In [19]:

```
check_stationarity(df['x'])
```

```
ADF Statistic: -2.033183
p-value: 0.272215
Time Series is not stationary
```

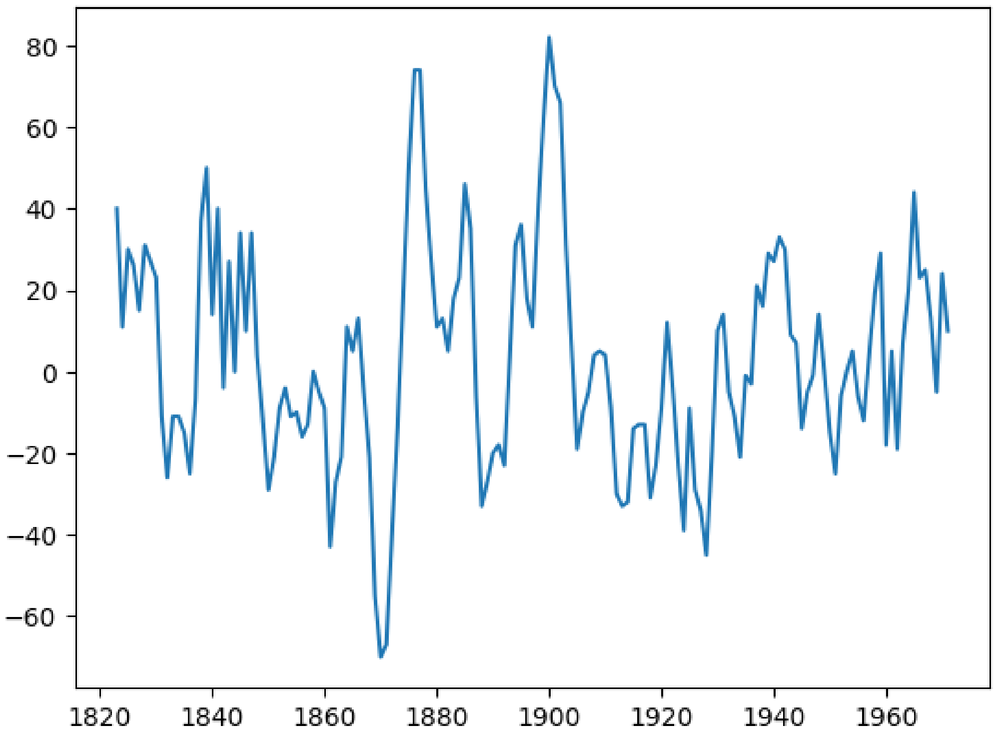
In [20]:

```
series_decomposition(df['x'])
```



In [21]:

```
plt.plot(df['x'].diff(1))
plt.show()
```



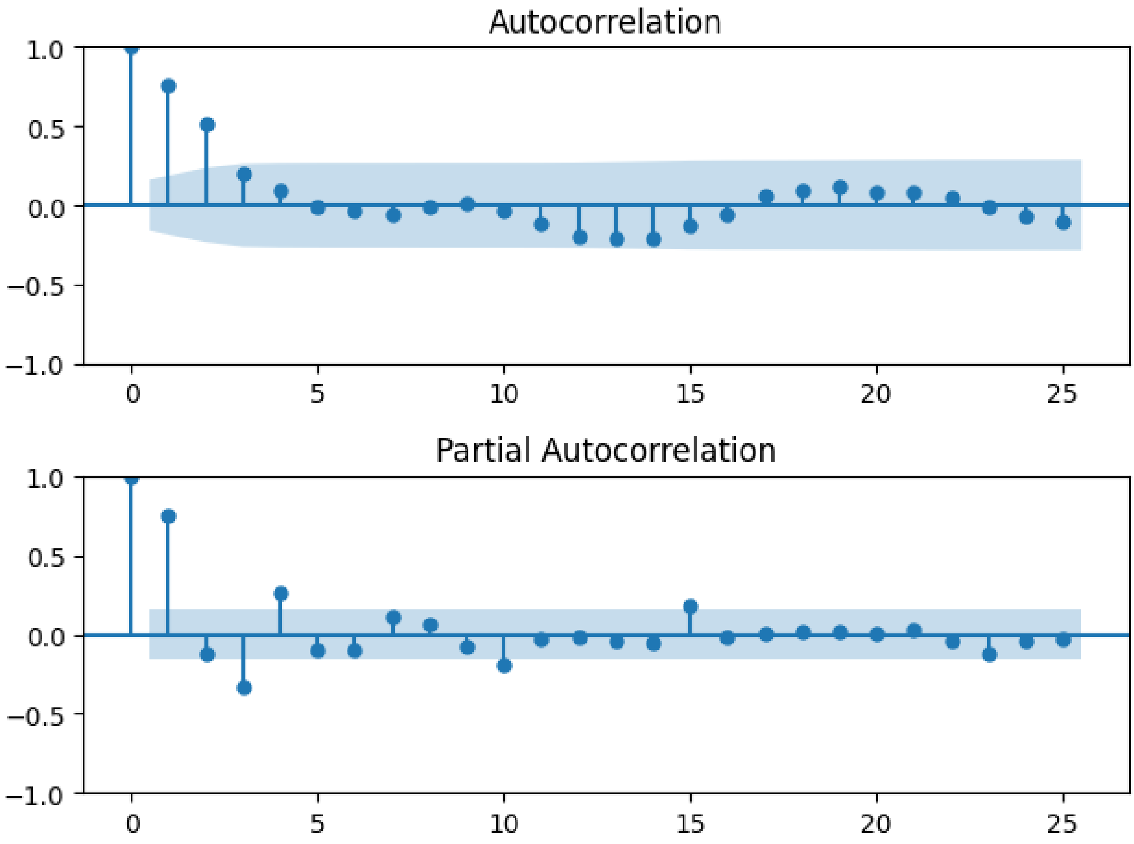
In [22]:

```
check_stationarity(df['x'].diff(1).dropna())
```

```
ADF Statistic: -3.835409
p-value: 0.002565
Time Series is stationary
```

In [23]:

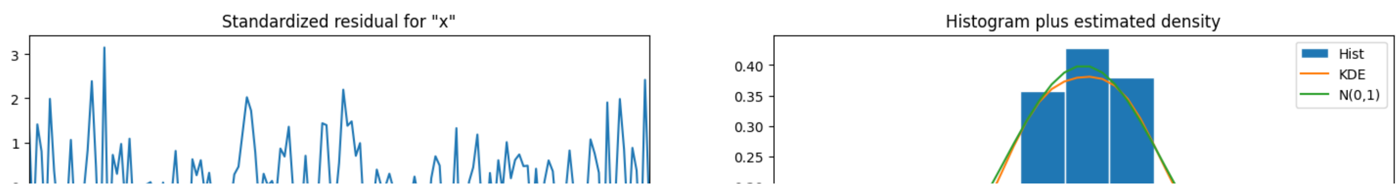
```
plot_acf_pacf_graphs(df['x'].diff(1).dropna())
```

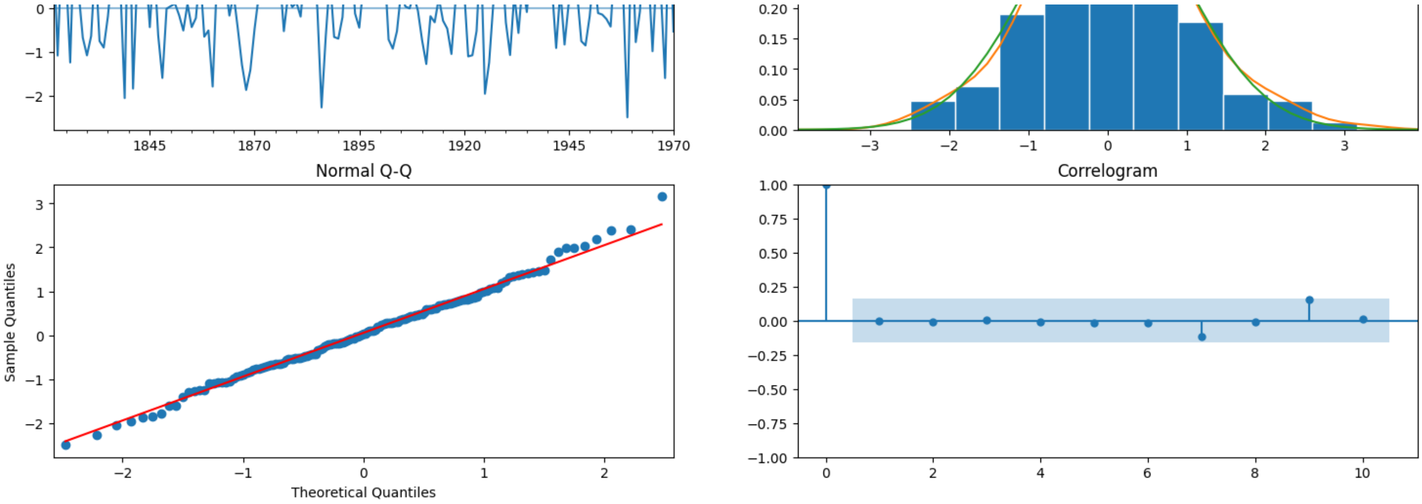


In [24]:

```
arima_modeling(df['x'], (4,1,2))
```

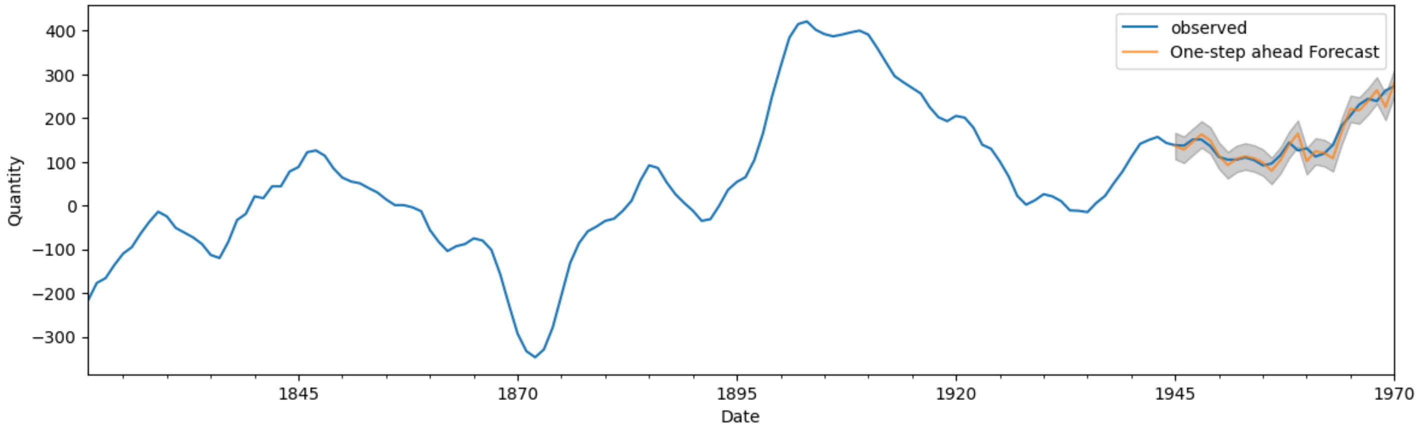
```
ARIMA(4, 1, 2) - AIC:1254.8485263818966
                               SARIMAX Results
==============================================================================
Dep. Variable:                      x   No. Observations:                  150
Model:                 ARIMA(4, 1, 2)   Log Likelihood                -620.424
Date:                Mon, 13 Nov 2023   AIC                           1254.849
Time:                        15:32:15   BIC                           1275.876
Sample:                    12-31-1821   HQIC                          1263.392
                         - 12-31-1970
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3924      0.256      1.531      0.126      -0.110       0.895
ar.L2          0.2000      0.166      1.208      0.227      -0.124       0.524
ar.L3         -0.2344      0.216     -1.083      0.279      -0.659       0.190
ar.L4          0.1313      0.153      0.860      0.390      -0.168       0.431
ma.L1          0.5572      0.241      2.315      0.021       0.085       1.029
ma.L2          0.4518      0.206      2.196      0.028       0.049       0.855
sigma2       239.3068     29.110      8.221      0.000     182.252     296.362
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.94
Prob(Q):                              0.97   Prob(JB):                         0.63
Heteroskedasticity (H):               0.80   Skew:                             0.16
Prob(H) (two-sided):                  0.43   Kurtosis:                         3.22
===================================================================================

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





In [25]:

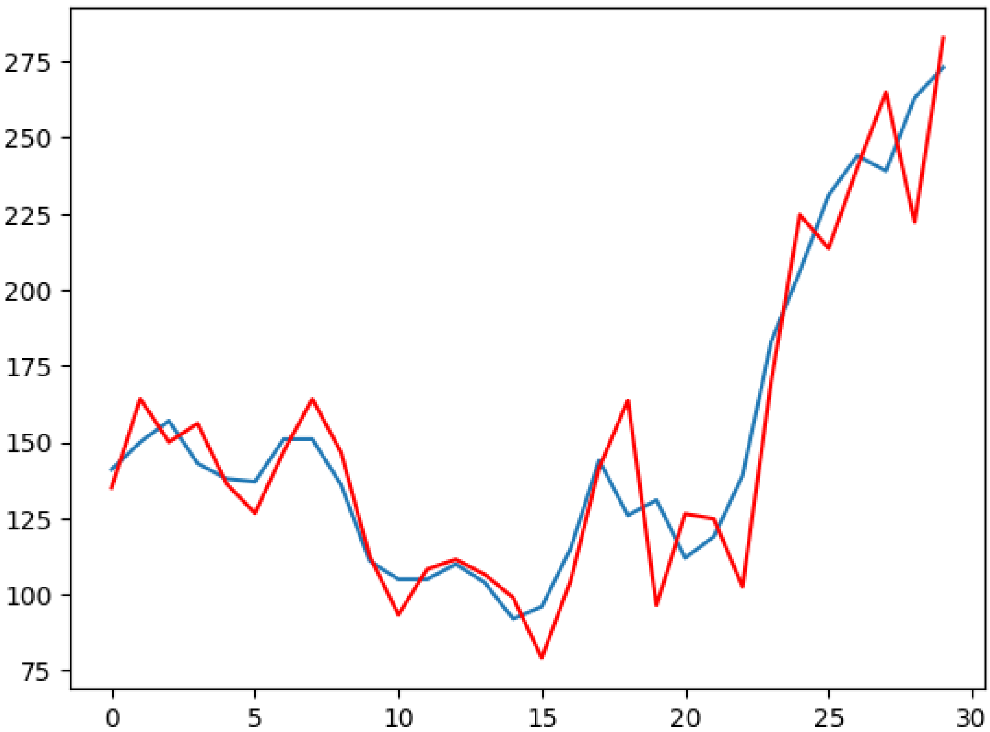
```
arima_prediction(df['x'], (4,1,2), start_point=pd.to_datetime('1945-12-31'))
```



In [26]:

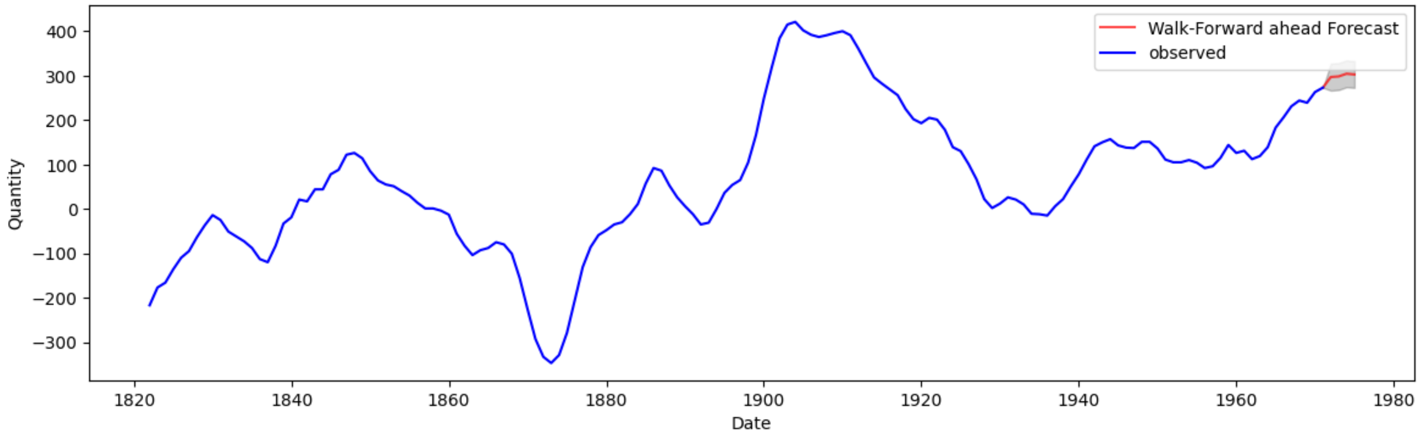
```
arima_walk_forward_validation(df['x'], (4,1,2), test_size=0.2)
```

```
Test RMSE: 17.260
```



In [27]:

```
arima_walk_forward_forecast(df['x'], (4,1,2), steps=4)
```



SARIMA(when dealing with time series data that exhibits a repeating pattern at fixed intervals, indicating a seasonal influence)

In [28]:

```
beer = pd.read_csv('BeerWineLiquor.csv')
beer.head()
```

Out[28]:

	date	beer
0	1/1/1992	1509
1	2/1/1992	1541
2	3/1/1992	1597
3	4/1/1992	1675
4	5/1/1992	1822

In [29]:

```
beer['date'] = pd.to_datetime(beer['date'])
beer = beer.set_index('date')
beer.index = pd.DatetimeIndex(beer.index.values, freq=beer.index.inferred_freq)
```

In [30]:

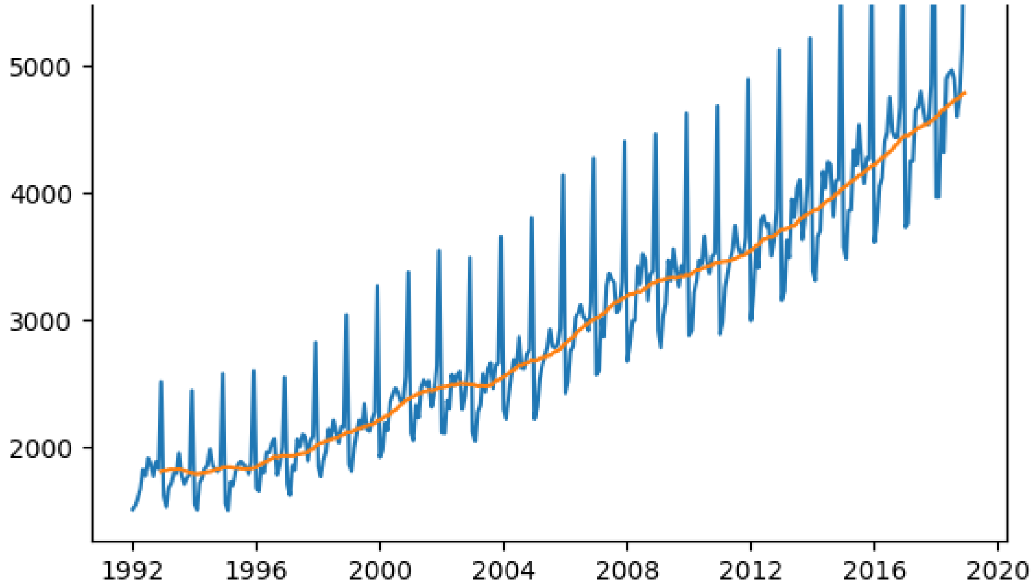
```
beer.head()
```

Out[30]:

	beer
1992-01-01	1509
1992-02-01	1541
1992-03-01	1597
1992-04-01	1675
1992-05-01	1822

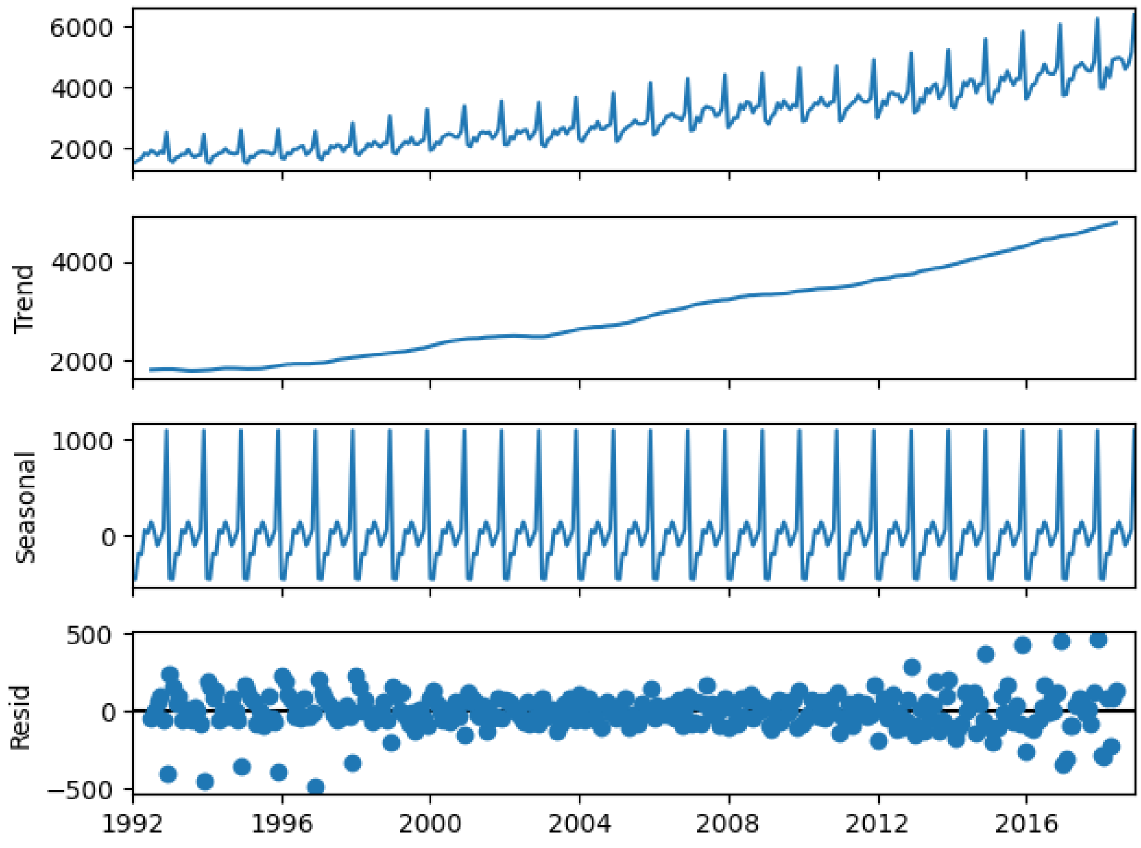
In [31]:

```
plt.plot(beer)
plt.plot(beer.rolling(window=12).mean())
plt.show()
```



In [32]:

```
series_decomposition(beer)
```



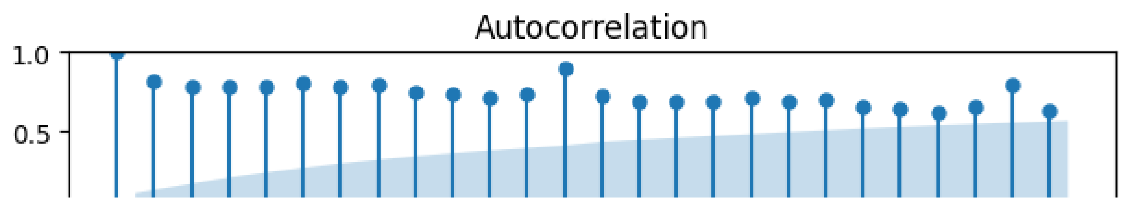
In [33]:

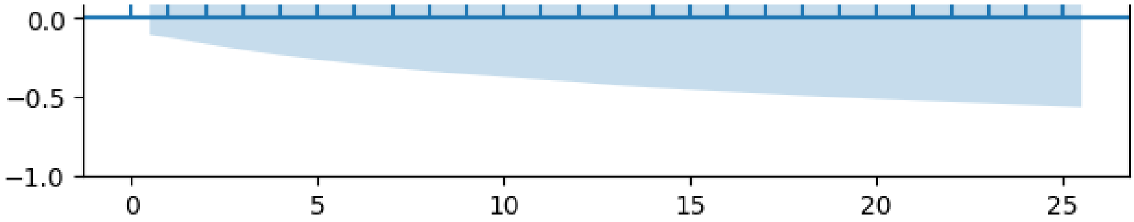
```
check_stationarity(beer)
```

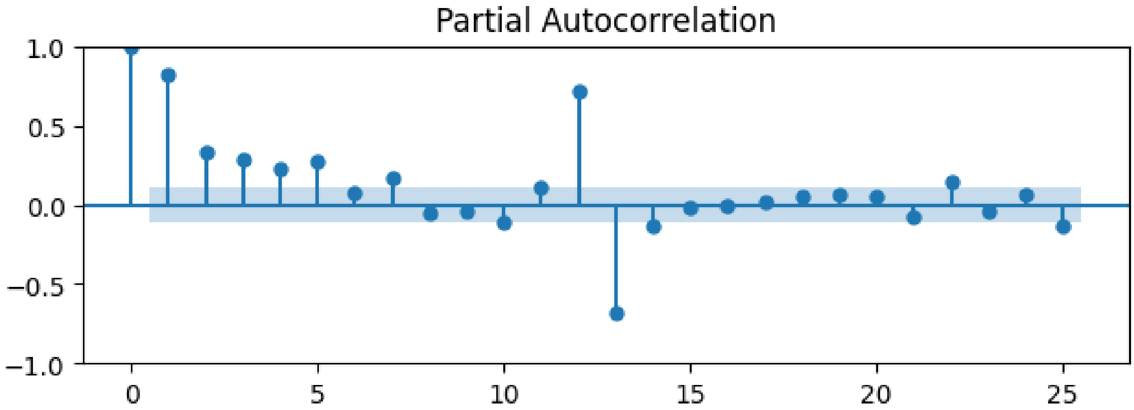
```
ADF Statistic: 2.864309
p-value: 1.000000
Time Series is not stationary
```

In [34]:

```
plot_acf_pacf_graphs(beer)
```

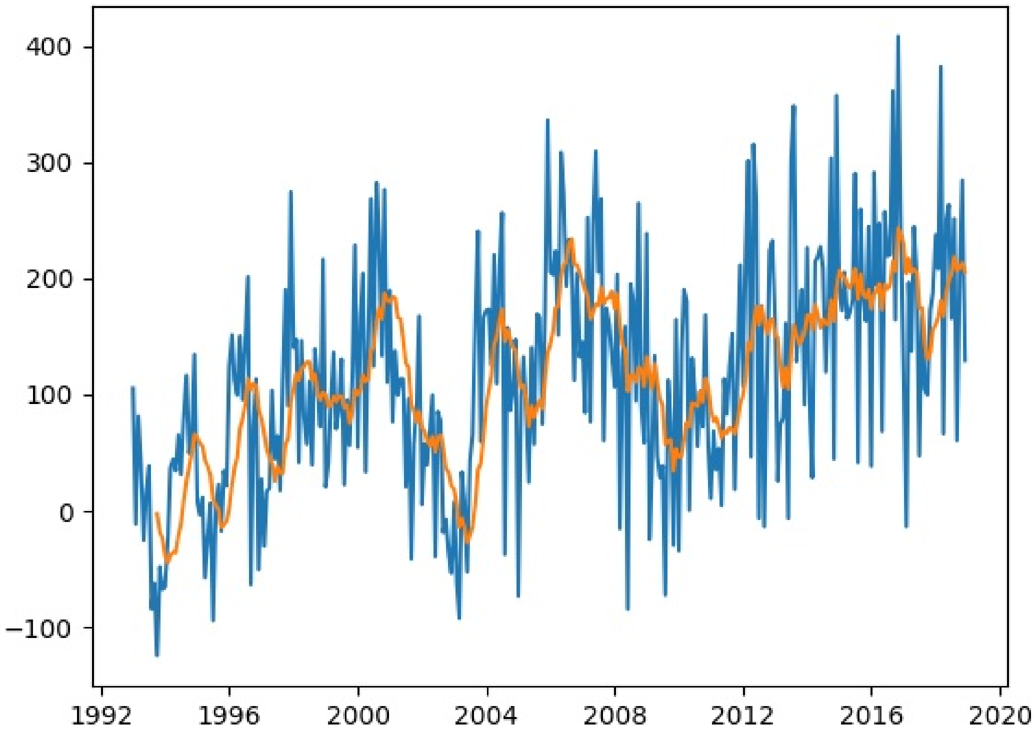






In [35]:

```
plt.plot(beer.diff(12))
plt.plot(beer.diff(12).rolling(window=10).mean())
plt.show()
```



In [38]:

```
check_stationarity(beer.diff(12).diff(1).dropna())
```

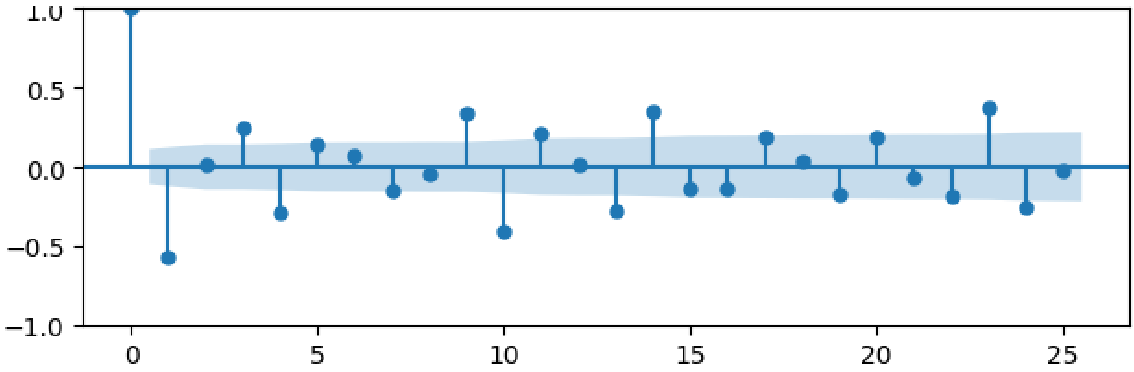
```
ADF Statistic: -8.639802
p-value: 0.000000
Time Series is stationary
```

In [39]:

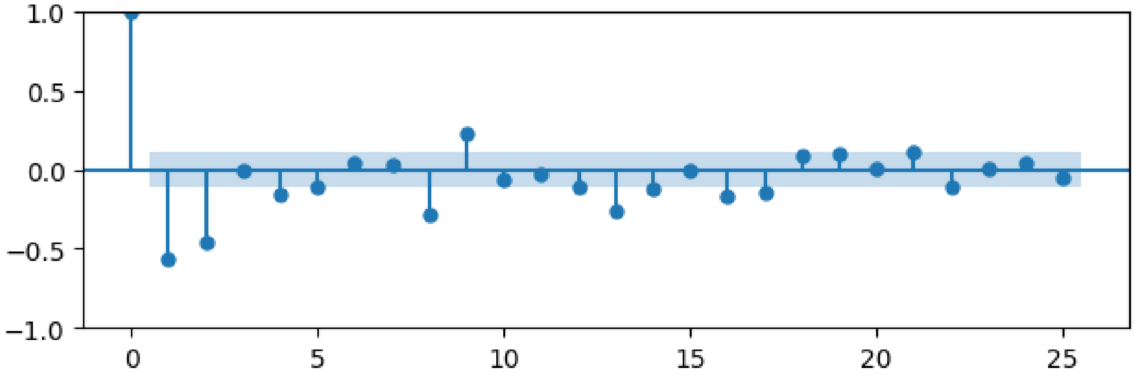
```
diff_beer = beer.diff(12).diff(1).dropna()
```

In [40]:

```
plot_acf_pacf_graphs(diff_beer)
```



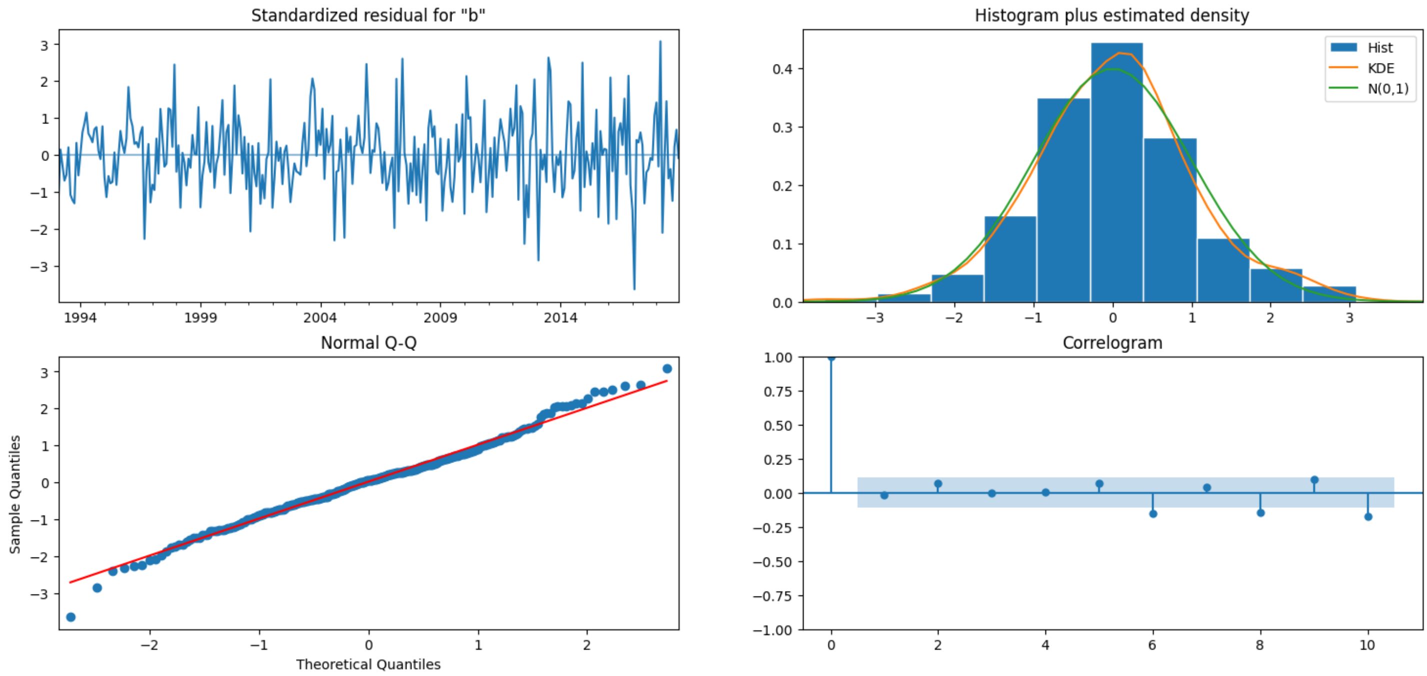
Partial Autocorrelation



In [41]:

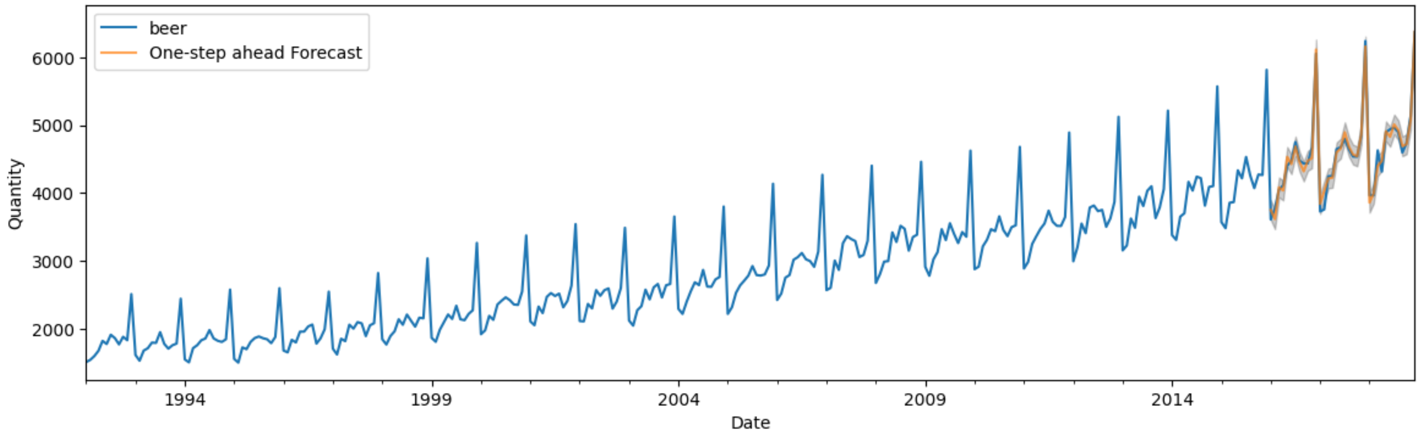
```
sarimax_modeling(beer, params=(2,1,3), s_params=(0,1,0,12))
```

```
Optimization terminated successfully.
         Current function value: 5.511796
         Iterations: 11
         Function evaluations: 725
SARIMAX(2, 1, 3)x(0, 1, 0, 12) - AIC:3583.643916978222
                                      SARIMAX Results

==========================================================================================
==
Dep. Variable:                                 beer   No. Observations:
324
Model:             SARIMAX(2, 1, 3)x(0, 1, [], 12)   Log Likelihood               -1785.
822
Date:                              Mon, 13 Nov 2023   AIC                           3583.
644
Time:                                      15:40:52   BIC                           3606
.083
Sample:                                  01-01-1992   HQIC                          3592.
613
                                       - 12-01-2018

Covariance Type:                                opg

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.1547      0.008   -150.100      0.000      -1.170      -1.140
ar.L2         -0.9923      0.006   -153.065      0.000      -1.005      -0.980
ma.L1          0.3964      0.042      9.392      0.000       0.314       0.479
ma.L2          0.2243      0.049      4.612      0.000       0.129       0.320
ma.L3         -0.6791      0.049    -13.873      0.000      -0.775      -0.583
sigma2      5576.6337    400.281     13.932      0.000    4792.098    6361.170
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):                 6.08
Prob(Q):                              0.75   Prob(JB):                         0.05
Heteroskedasticity (H):               1.93   Skew:                             0.03
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.68
===================================================================================

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



In [44]:

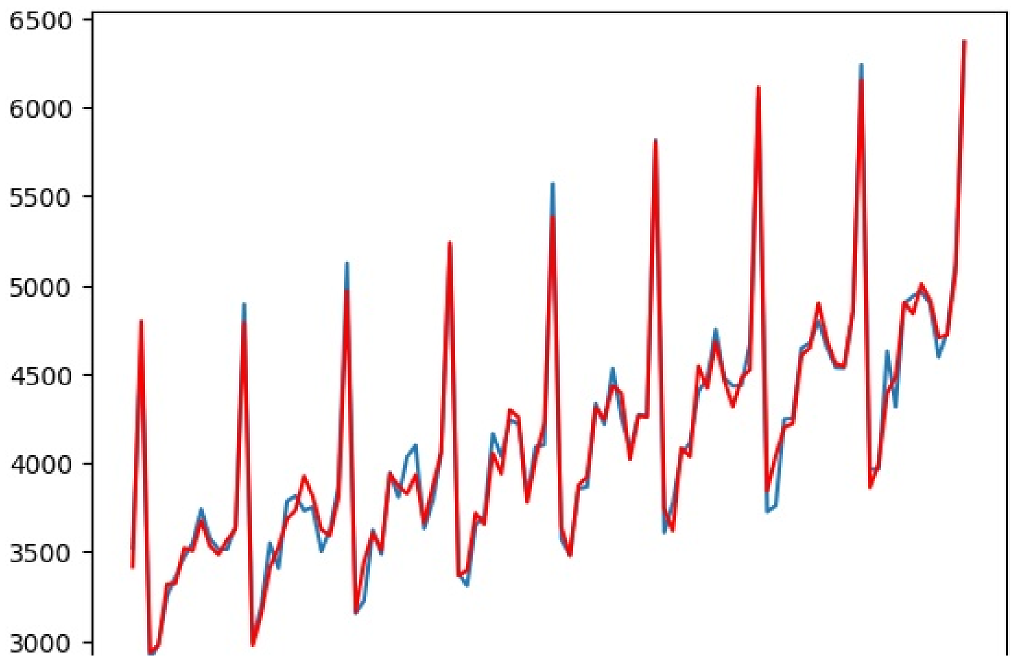
```
sarimax_prediction(beer, params=(2,1,3), s_params=(0,1,0,12), start_point=pd.to_datetime
('2016-01-01'))
```

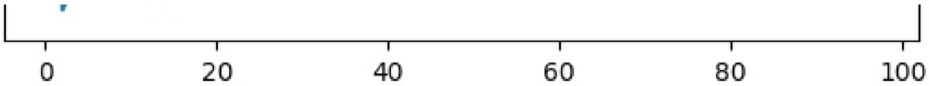


In [45]:

```
sarimax_walk_forward_validation(beer,params=(2,1,3), s_params=(0,1,0,12), test_size=0.3)
```

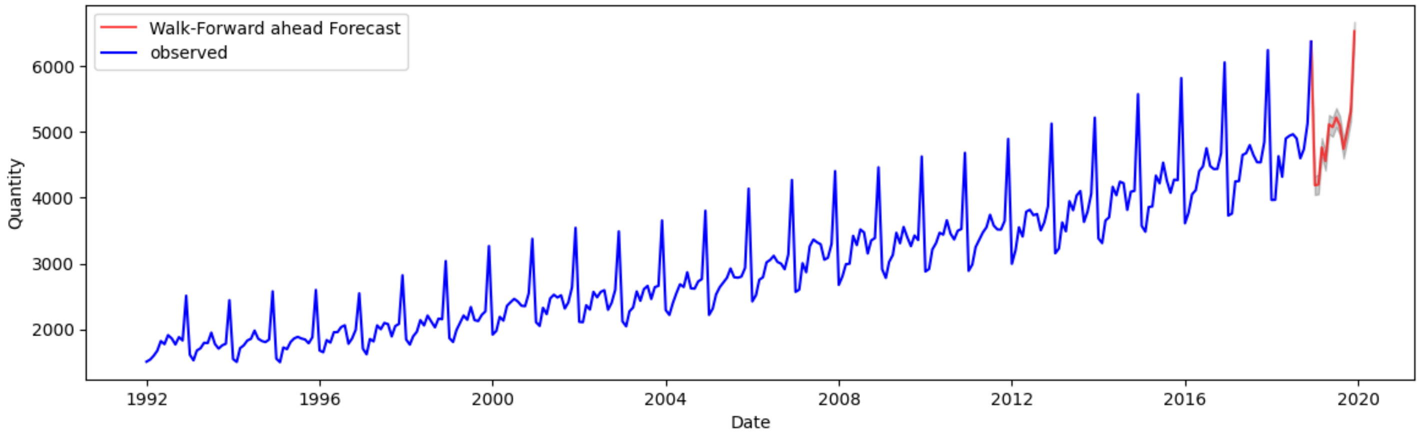
Test RMSE: 91.535





In [46]:

```
sarimax_walk_forward_forecast(beer['beer'],params=(2,1,3), s_params=(0,1,0,12), steps=12
)
```



In [ ]: