In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading Data

In [ ]:
```python
df = pd.read_csv('Seasonal Data.csv', encoding="ISO-8859-1")
df.head()
```

Out[15]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	P
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FU 10
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FU 10
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	O 10
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	F 10
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	C 10

5 rows × 21 columns

Data Cleaning

In [ ]:
```
df = df.drop_duplicates()

df['Order Date'] = pd.to_datetime(df['Order Date'])

print(df.isnull().sum())
```

```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [ ]:
```
monthly_data = df[['Order Date', 'Profit']].copy()
monthly_data.set_index('Order Date', inplace=True)

# Resample to monthly frequency and mean profit
monthly_profit = monthly_data.resample('ME').mean()
print(monthly_profit)
```

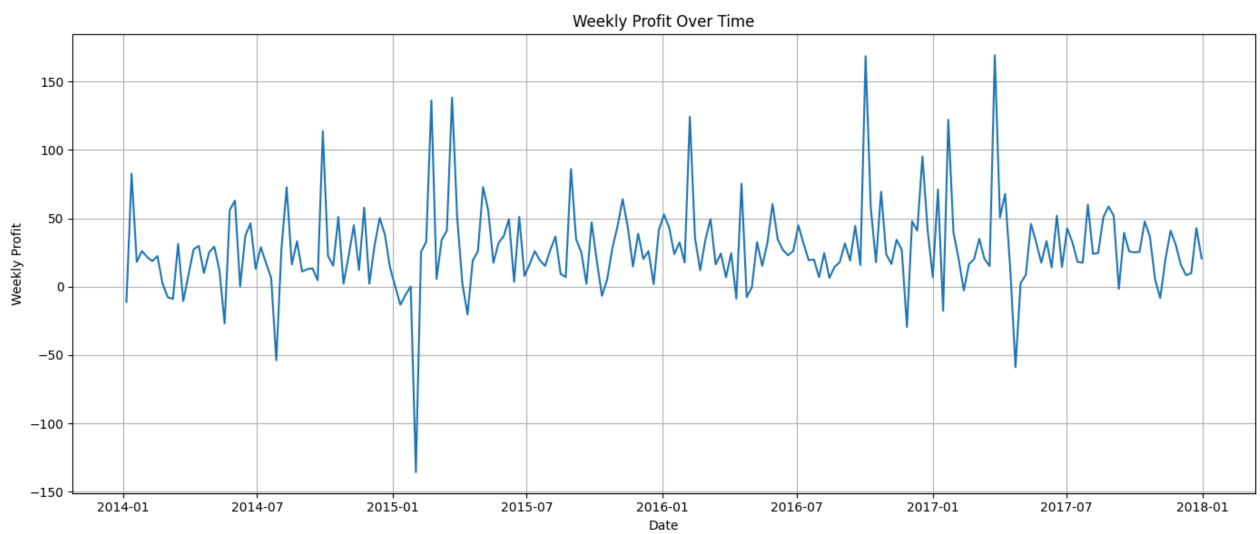
```
               Profit
Order Date
2014-01-31  31.015072
2014-02-28  18.745835
2014-03-31   3.176624
2014-04-30  25.843224
2014-05-31  22.448439
2014-06-30  36.863144
2014-07-31  -5.884494
2014-08-31  34.758856
2014-09-30  31.074998
2014-10-31  21.687153
2014-11-30  29.220525
2014-12-31  32.315000
2015-01-31 -56.569086
2015-02-28  43.966419
2015-03-31  70.522448
2015-04-30  26.171851
2015-05-31  31.971705
2015-06-30  24.170704
2015-07-31  23.490345
2015-08-31  33.684330
2015-09-30  28.017620
2015-10-31  16.972084
2015-11-30  38.502433
2015-12-31  25.370145
2016-01-31  31.739588
2016-02-29  60.296139
2016-03-31  22.159313
2016-04-30  17.516558
2016-05-31  38.498428
2016-06-30  23.871247
2016-07-31  22.054119
2016-08-31  11.716303
2016-09-30  25.698781
2016-10-31  82.873176
2016-11-30  10.841642
2016-12-31  50.810538
2017-01-31  46.067349
2017-02-28  15.082916
2017-03-31  61.982737
2017-04-30   4.597488
2017-05-31  26.209020
2017-06-30  33.564636
2017-07-31  30.763811
2017-08-31  41.472274
2017-09-30  23.946744
2017-10-31  31.125086
2017-11-30  21.111337
2017-12-31  18.362223
```

In [ ]:
```python
weekly_data = df[['Order Date', 'Profit']].copy()
weekly_data.set_index('Order Date', inplace=True)

#Coverting into weekly profit
weekly_profit = weekly_data.resample('W').mean()
print(weekly_profit)
```

```
                Profit
Order Date
2014-01-05 -11.110980
2014-01-12  82.671462
2014-01-19  18.131195
2014-01-26  26.139231
2014-02-02  21.776973
...               ...
2017-12-03  16.013970
2017-12-10   8.390592
2017-12-17   9.856660
2017-12-24  42.883730
2017-12-31  20.707991

[209 rows x 1 columns]
```

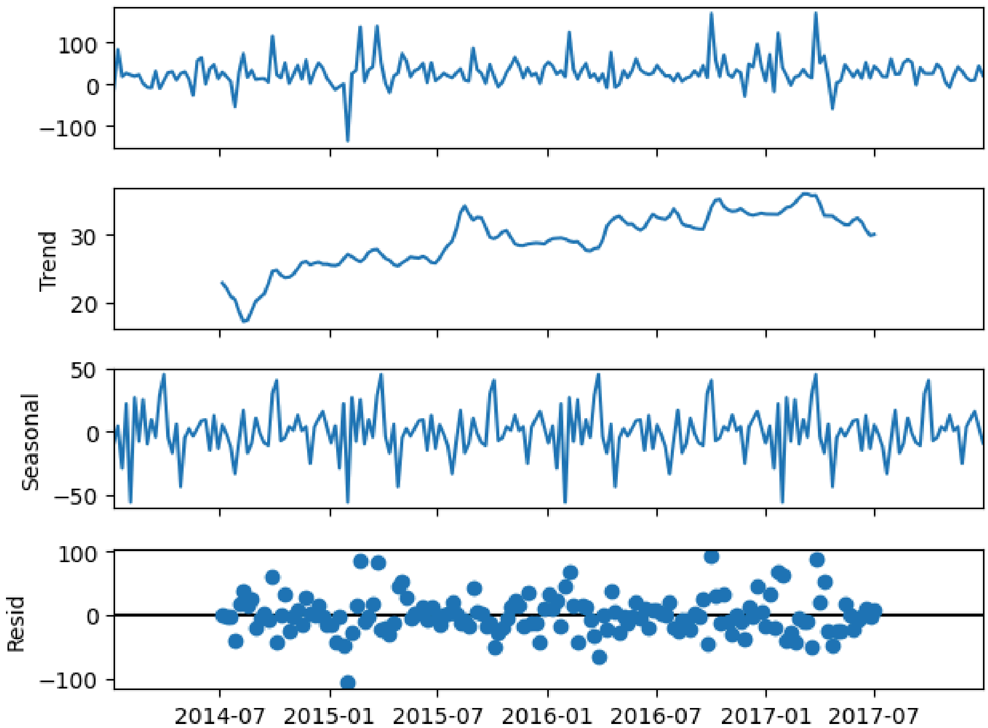
Plot

In [ ]:
```python
plt.figure(figsize=(14, 6))
sns.lineplot(data=weekly_profit, x=weekly_profit.index, y='Profit')
plt.title('Weekly Profit Over Time')
plt.xlabel('Date')
plt.ylabel('Weekly Profit')
plt.grid(True)
plt.tight_layout()
plt.show()

```



In [ ]:

```python
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series
decomposition = seasonal_decompose(weekly_profit, model='additive')
decomposition.plot()
plt.figure(figsize=(12, 8))
plt.tight_layout()
plt.suptitle('Seasonal Decomposition of Weekly profit')
plt.show()

```




```
<Figure size 1200x800 with 0 Axes>
```

From the graph we can see that this data is seasonal dataset

ADF test

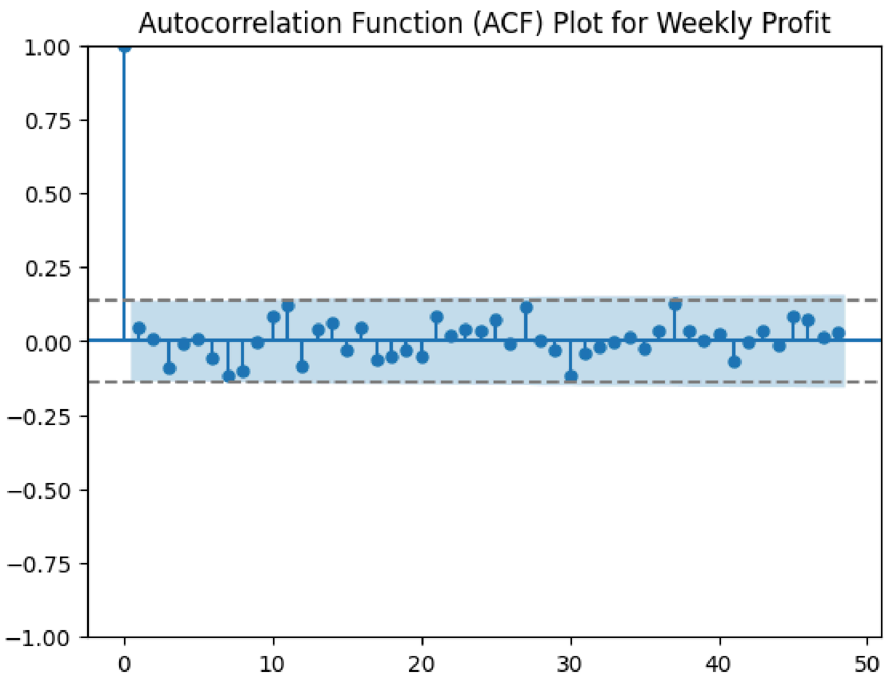
In [23]:

```python
from statsmodels.tsa.stattools import adfuller

result = adfuller(weekly_profit['Profit'])

print("ADF Statistic:", result[0])
print("p-value:", result[1])
for key, value in result[4].items():
    print(f'Critical Value ({key}): {value}')

if result[1] < 0.05:
    print("The series is stationary.")
else:
    print("The series is not stationary.")

```

```
ADF Statistic: -13.790688189145076
p-value: 8.938098275917793e-26
Critical Value (1%): -3.4621857592784546
Critical Value (5%): -2.875537986778846
Critical Value (10%): -2.574231080806213
The series is stationary.
```

In [24]:
```
from statsmodels.graphics.tsaplots import plot_acf

# ACF
plt.figure(figsize=(12, 6))
acf_plot_close = plot_acf(weekly_profit.dropna(), lags=48, alpha=0.05)  # confidence inter

# Add a dotted line at the significance threshold
plt.axhline(y=-1.96/np.sqrt(len(weekly_profit)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(weekly_profit)), linestyle='--', color='gray')

plt.title('Autocorrelation Function (ACF) Plot for Weekly Profit')
plt.show()
```

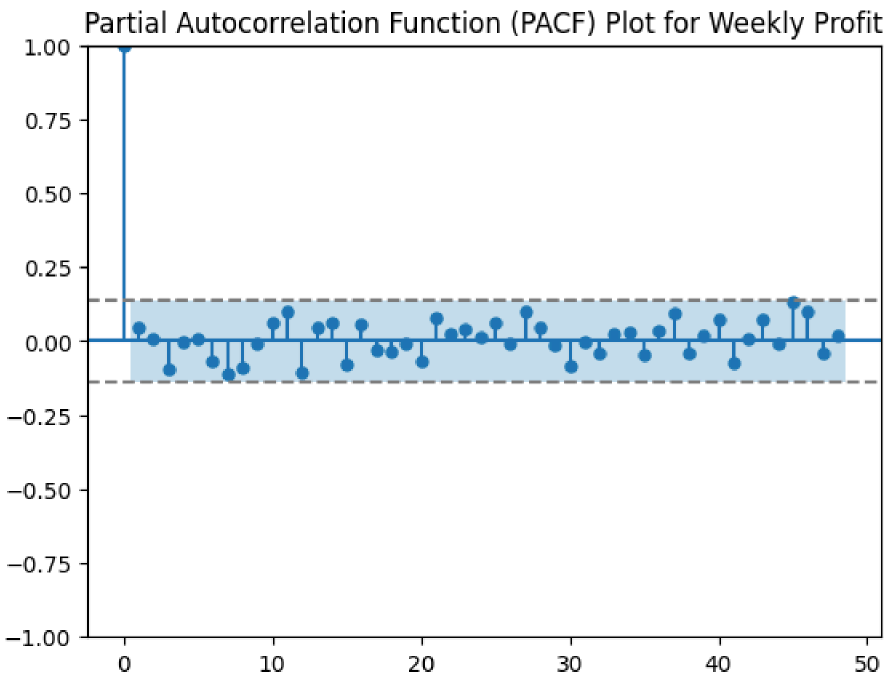
```
<Figure size 1200x600 with 0 Axes>
```



In [25]:

```
from statsmodels.graphics.tsaplots import plot_pacf

#PACF
plt.figure(figsize=(12, 6))
pacf_plot_temp = plot_pacf(weekly_profit.dropna(), lags=48, alpha=0.05)

# Add a dotted line at the significance threshold
plt.axhline(y=-1.96/np.sqrt(len(weekly_profit)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(weekly_profit)), linestyle='--', color='gray')

plt.title('Partial Autocorrelation Function (PACF) Plot for Weekly Profit')
plt.show()
```

<Figure size 1200x600 with 0 Axes>



In [26]:
```
import statsmodels.api as sm
import itertools
from statsmodels.tsa.statespace.sarimax import SARIMAX

p=d=q= range(0,2)
# Generate all different combinatins of p,d and q triplets
pdq= list(itertools.product(p,d,q))

seasonal_pdq = [(x[0], x[1], x[2],12) for x in list(itertools.product(p,d,q))]

for param in pdq:
  for param_seasonal in seasonal_pdq:
    try:
      mod= sm.tsa.statespace.SARIMAX(weekly_profit,
                                     order= param,
                                     seasonal_order= param_seasonal,
                                     enforce_stationarity= False,
                                     enforce_invertibility= False)
      results= mod.fit()
      print(f'SARIMA{param}x{param_seasonal}12- AIC: {results.aic}')
    except:
      continue
```

```
SARIMA(0, 0, 0)x(0, 0, 0, 12)12- AIC: 2156.888558218995
SARIMA(0, 0, 0)x(0, 0, 1, 12)12- AIC: 2023.6058502594185
SARIMA(0, 0, 0)x(0, 1, 0, 12)12- AIC: 2082.773519163945
SARIMA(0, 0, 0)x(0, 1, 1, 12)12- AIC: 1831.74461055686
SARIMA(0, 0, 0)x(1, 0, 0, 12)12- AIC: 2020.414576116892
SARIMA(0, 0, 0)x(1, 0, 1, 12)12- AIC: 1956.241045990632
SARIMA(0, 0, 0)x(1, 1, 0, 12)12- AIC: 1902.7050713545368
SARIMA(0, 0, 0)x(1, 1, 1, 12)12- AIC: 1832.8439613077053
SARIMA(0, 0, 1)x(0, 0, 0, 12)12- AIC: 2116.09439095827
SARIMA(0, 0, 1)x(0, 0, 1, 12)12- AIC: 1999.3926962331466
SARIMA(0, 0, 1)x(0, 1, 0, 12)12- AIC: 2073.628199570655
SARIMA(0, 0, 1)x(0, 1, 1, 12)12- AIC: 1824.027617179786
SARIMA(0, 0, 1)x(1, 0, 0, 12)12- AIC: 2014.0152562383623
SARIMA(0, 0, 1)x(1, 0, 1, 12)12- AIC: 1947.2197180130447
SARIMA(0, 0, 1)x(1, 1, 0, 12)12- AIC: 1904.6318308299383
SARIMA(0, 0, 1)x(1, 1, 1, 12)12- AIC: 1825.2991589565547
SARIMA(0, 1, 0)x(0, 0, 0, 12)12- AIC: 2162.826826167911
SARIMA(0, 1, 0)x(0, 0, 1, 12)12- AIC: 2040.7427160622137
SARIMA(0, 1, 0)x(0, 1, 0, 12)12- AIC: 2214.0270070590245
SARIMA(0, 1, 0)x(0, 1, 1, 12)12- AIC: 1948.530725560736
SARIMA(0, 1, 0)x(1, 0, 0, 12)12- AIC: 2050.346362077836
SARIMA(0, 1, 0)x(1, 0, 1, 12)12- AIC: 2042.250745279631
SARIMA(0, 1, 0)x(1, 1, 0, 12)12- AIC: 2023.860774296481
SARIMA(0, 1, 0)x(1, 1, 1, 12)12- AIC: 1948.3401034852745
SARIMA(0, 1, 1)x(0, 0, 0, 12)12- AIC: 2027.8506774817458
SARIMA(0, 1, 1)x(0, 0, 1, 12)12- AIC: 1916.9678966805159
SARIMA(0, 1, 1)x(0, 1, 0, 12)12- AIC: 2068.4491286542234
SARIMA(0, 1, 1)x(0, 1, 1, 12)12- AIC: 1818.8481096506332
SARIMA(0, 1, 1)x(1, 0, 0, 12)12- AIC: 1935.3719500244456
SARIMA(0, 1, 1)x(1, 0, 1, 12)12- AIC: 1918.090667048636
SARIMA(0, 1, 1)x(1, 1, 0, 12)12- AIC: 1900.6347254313573
SARIMA(0, 1, 1)x(1, 1, 1, 12)12- AIC: 1820.7979412546983
SARIMA(1, 0, 0)x(0, 0, 0, 12)12- AIC: 2111.8269457836213
SARIMA(1, 0, 0)x(0, 0, 1, 12)12- AIC: 1996.5681732446599
SARIMA(1, 0, 0)x(0, 1, 0, 12)12- AIC: 2084.5290075328003
SARIMA(1, 0, 0)x(0, 1, 1, 12)12- AIC: 1833.7425419029564
SARIMA(1, 0, 0)x(1, 0, 0, 12)12- AIC: 1996.4727577019762
SARIMA(1, 0, 0)x(1, 0, 1, 12)12- AIC: 1958.2406970346601
SARIMA(1, 0, 0)x(1, 1, 0, 12)12- AIC: 1893.9546472056595
SARIMA(1, 0, 0)x(1, 1, 1, 12)12- AIC: 1834.8431386212847
SARIMA(1, 0, 1)x(0, 0, 0, 12)12- AIC: 2044.7281486556337
SARIMA(1, 0, 1)x(0, 0, 1, 12)12- AIC: 1927.629601302803
SARIMA(1, 0, 1)x(0, 1, 0, 12)12- AIC: 2067.723159881378
SARIMA(1, 0, 1)x(0, 1, 1, 12)12- AIC: 1827.7507321199432
SARIMA(1, 0, 1)x(1, 0, 0, 12)12- AIC: 1936.2279471303855
SARIMA(1, 0, 1)x(1, 0, 1, 12)12- AIC: 1948.9929445060156
SARIMA(1, 0, 1)x(1, 1, 0, 12)12- AIC: 1892.7456574364264
SARIMA(1, 0, 1)x(1, 1, 1, 12)12- AIC: 1826.3679193106734
SARIMA(1, 1, 0)x(0, 0, 0, 12)12- AIC: 2110.561223941431
SARIMA(1, 1, 0)x(0, 0, 1, 12)12- AIC: 1993.8679026671948
SARIMA(1, 1, 0)x(0, 1, 0, 12)12- AIC: 2163.7330649773203
SARIMA(1, 1, 0)x(0, 1, 1, 12)12- AIC: 1898.838509817877
SARIMA(1, 1, 0)x(1, 0, 0, 12)12- AIC: 1993.0632123291407
SARIMA(1, 1, 0)x(1, 0, 1, 12)12- AIC: 1994.9138377783102
SARIMA(1, 1, 0)x(1, 1, 0, 12)12- AIC: 1959.350151846037
SARIMA(1, 1, 0)x(1, 1, 1, 12)12- AIC: 1899.8898627935496
SARIMA(1, 1, 1)x(0, 0, 0, 12)12- AIC: 2029.2632925014368
SARIMA(1, 1, 1)x(0, 0, 1, 12)12- AIC: 1918.3422950270237
SARIMA(1, 1, 1)x(0, 1, 0, 12)12- AIC: 2070.2848094978062
SARIMA(1, 1, 1)x(0, 1, 1, 12)12- AIC: 1820.8416215680381
SARIMA(1, 1, 1)x(1, 0, 0, 12)12- AIC: 1926.7033175519075
SARIMA(1, 1, 1)x(1, 0, 1, 12)12- AIC: 1919.4519746027413
SARIMA(1, 1, 1)x(1, 1, 0, 12)12- AIC: 1890.0128082223362
SARIMA(1, 1, 1)x(1, 1, 1, 12)12- AIC: 1822.792107378268
```

In [27]:

```
Best_model=  sm.tsa.statespace.SARIMAX(weekly_profit,
                                      order= (1,0,1),
                                      seasonal_order= (0,1,1,12),
                                      enforce_stationarity= False,
                                      enforce_invertibility= False)
results= Best_model.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9491      0.042    -22.667      0.000      -1.031      -0.867
ma.L1          0.9877      0.124      7.965      0.000       0.745       1.231
ma.S.L12      -1.0000      0.110     -9.120      0.000      -1.215      -0.785
sigma2      1033.9243      0.000   9.75e+06      0.000    1033.924    1033.924
==============================================================================
```

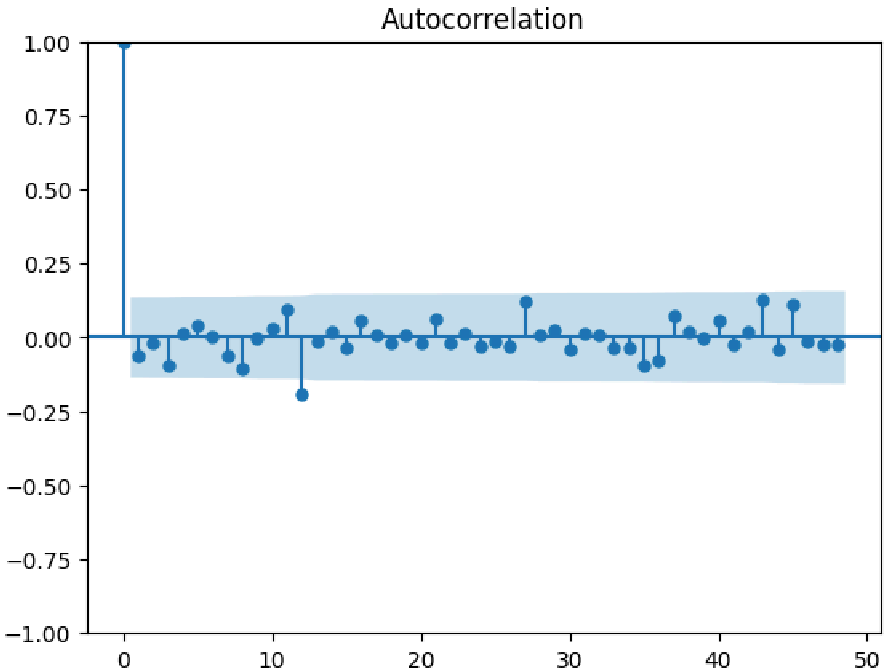
In [28]:

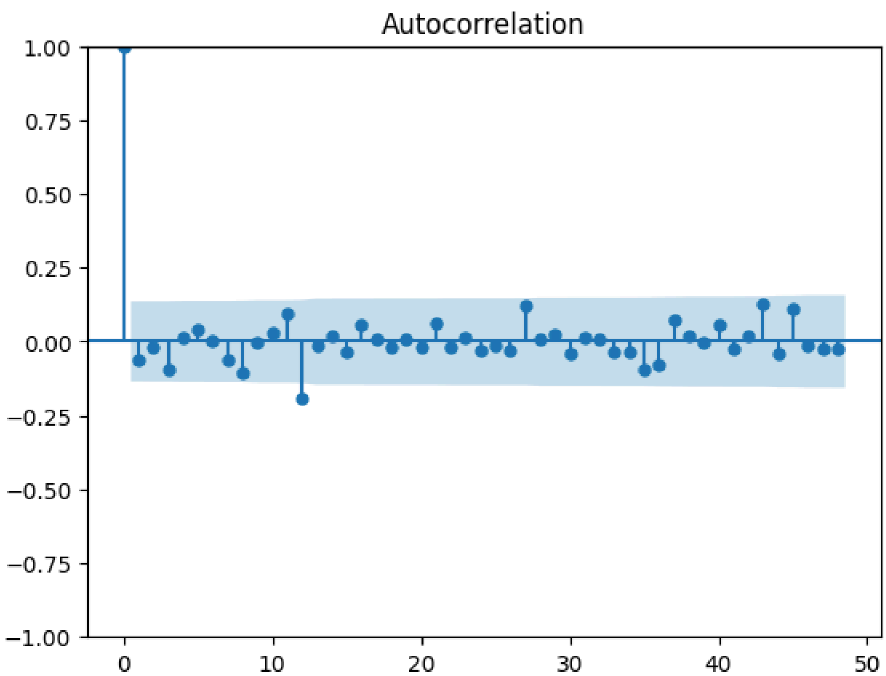
```
Best_model=  sm.tsa.statespace.SARIMAX(weekly_profit,
                                      order= (0,1,1),
                                      seasonal_order= (0,1,1,12),
                                      enforce_stationarity= False,
                                      enforce_invertibility= False)
results= Best_model.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.0000    223.237     -0.004      0.996    -438.536     436.536
ma.S.L12      -1.0000    223.237     -0.004      0.996    -438.536     436.536
sigma2      1043.1985      0.208   5004.156      0.000    1042.790    1043.607
==============================================================================
```

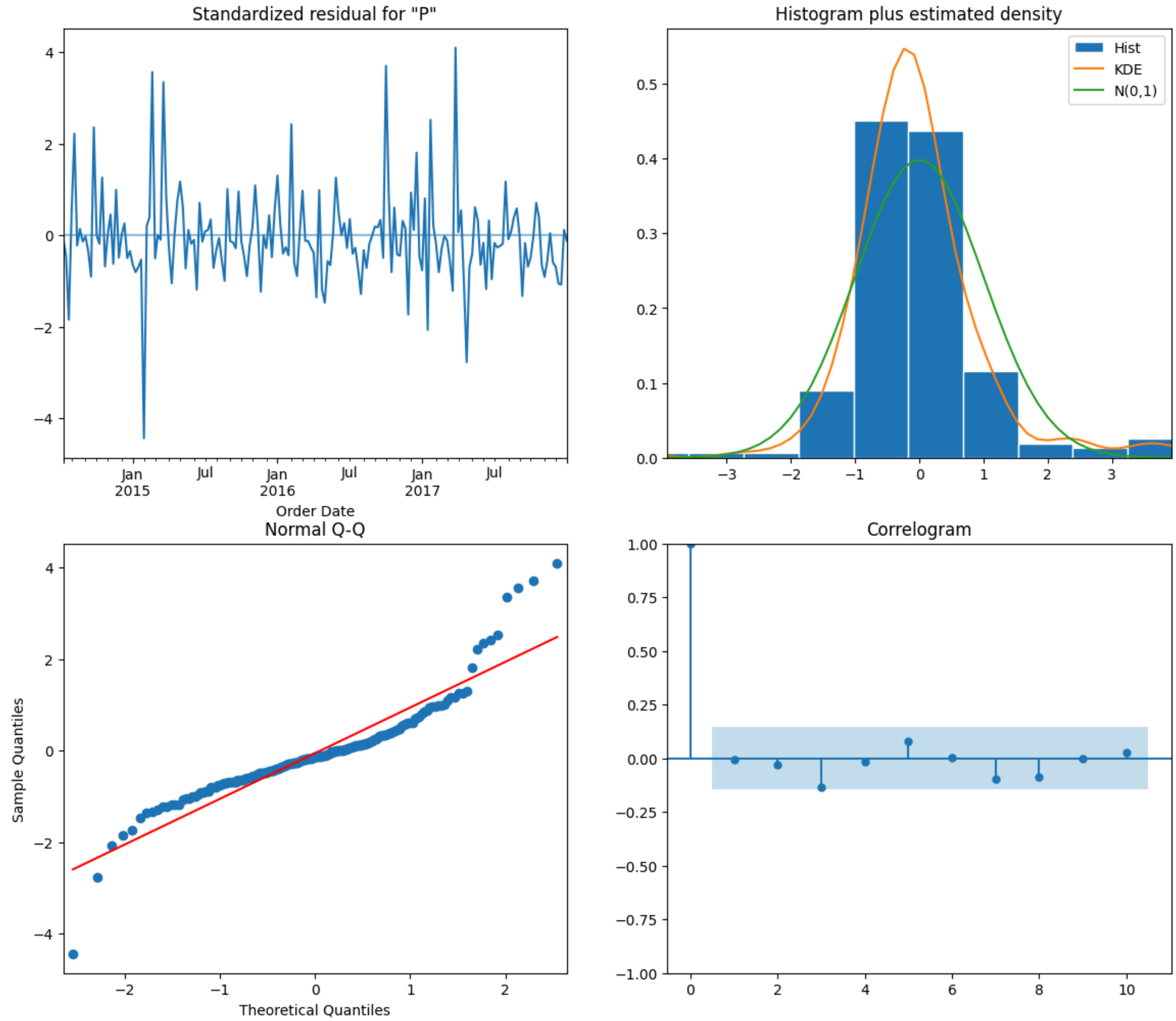
In [29]:
```
residuals = results.resid
plot_acf(residuals, lags=48)
```

Out[29]:



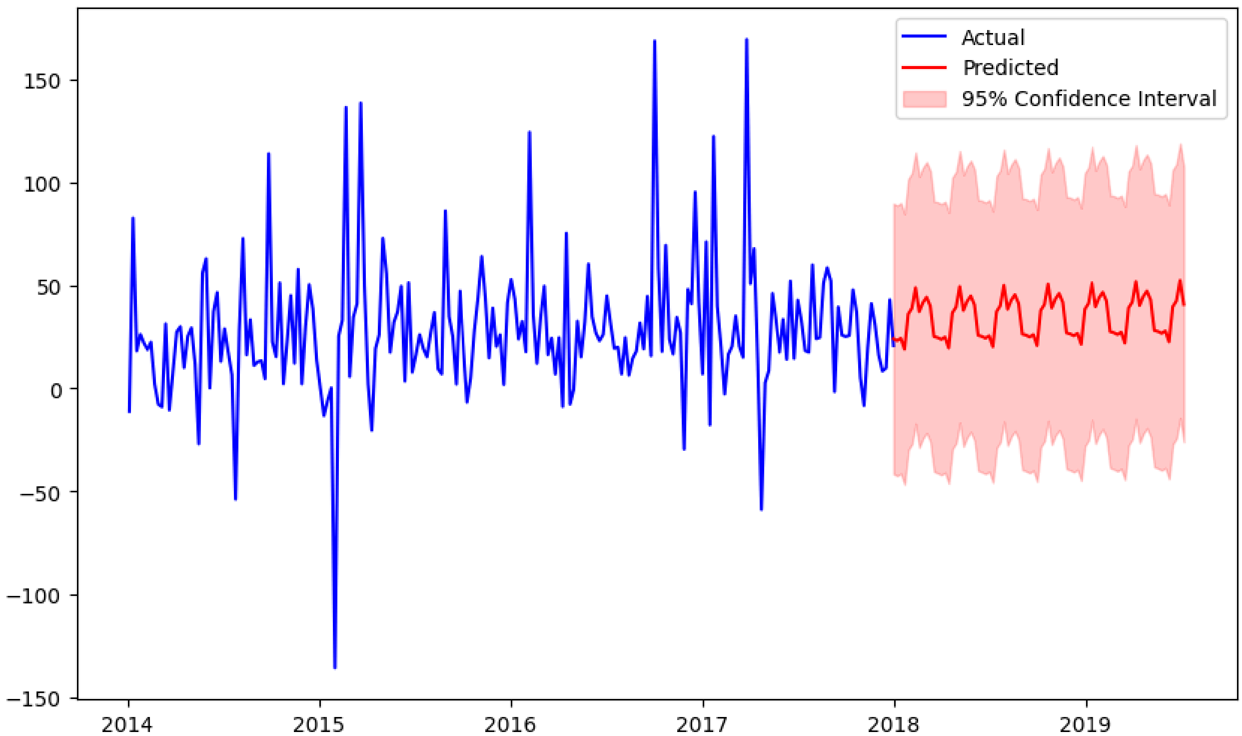


In [30]:
```
results.plot_diagnostics(figsize=(14,12))
plt.show()
```



In [31]:

```
from statsmodels.tsa.arima.model import ARIMA

# Forecast with confidence intervals
forecast = results.get_forecast(steps=80)
forecast_ci = forecast.conf_int()

# Forecast index
forecast_index = pd.date_range(start=weekly_profit.index[-1], periods=80, freq=weekly_prof

# Actual vs Predicted with Confidence Intervals plot
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(weekly_profit, label='Actual', color='blue')
ax.plot(forecast_index, forecast.predicted_mean, label='Predicted', color='red')

ax.fill_between(forecast_index,
                forecast_ci.iloc[:, 0],
                forecast_ci.iloc[:, 1], color='red', alpha=0.2, label='95% Confidence Inte

ax.legend()
plt.show()

# Print the summary
print(results.summary())
```



```
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                              Profit   No. Observations:                  209
Model:             SARIMAX(0, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -906.424
Date:                             Mon, 28 Apr 2025   AIC                           1818.848
Time:                                     21:45:43   BIC                           1828.460
Sample:                                 01-05-2014   HQIC                          1822.745
                                      - 12-31-2017
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.0000    223.237     -0.004      0.996    -438.536     436.536
ma.S.L12      -1.0000    223.237     -0.004      0.996    -438.536     436.536
sigma2      1043.1985      0.208   5004.156      0.000    1042.790    1043.607
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               222.96
Prob(Q):                              0.92   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.82
Prob(H) (two-sided):                  0.45   Kurtosis:                         8.17
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.72e+22. Standard
errors may be unstable.
```

In [32]:

```
df2= weekly_profit.copy()

df2.describe()
```

Out[32]:

	Profit
count	209.000000
mean	27.913498
std	32.653447
min	-135.678708
25%	12.506919
50%	24.582379
75%	41.007081
max	169.309043

In [33]:

```
from sklearn.metrics import mean_squared_error

n = int(len(df2) * 0.8)
train = weekly_profit[:n]
test = weekly_profit[n:]


print(len(train))
print(len(test))
```

```
167
42
```

In [34]:
```
import warnings
warnings.filterwarnings("ignore")

model=  sm.tsa.statespace.SARIMAX(train,
                                     order= (1,0,1),
                                     seasonal_order= (0,1,1,12),
                                     enforce_stationarity= False,
                                     enforce_invertibility= False)
result= model.fit()

print(result.summary())

```

```
                                      SARIMAX Results                                       
==========================================================================================
Dep. Variable:                              Profit   No. Observations:                  167
Model:             SARIMAX(1, 0, 1)x(0, 1, 1, 12)   Log Likelihood                -702.744
Date:                             Mon, 28 Apr 2025   AIC                           1413.488
Time:                                     21:46:57   BIC                           1425.283
Sample:                                 01-05-2014   HQIC                          1418.282
                                      - 03-12-2017                                         
Covariance Type:                               opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9616      0.061     15.893      0.000       0.843       1.080
ma.L1         -1.0000    364.549     -0.003      0.998    -715.503     713.503
ma.S.L12      -1.0000    364.583     -0.003      0.998    -715.569     713.569
sigma2      1019.3438      0.266   3839.267      0.000    1018.823    1019.864
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):               149.46
Prob(Q):                              0.75   Prob(JB):                         0.00
Heteroskedasticity (H):               0.67   Skew:                             0.45
Prob(H) (two-sided):                  0.17   Kurtosis:                         7.96
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.53e+22. Standard errors may be unstable.
```

In [35]:
```
# Forecast using the trained model
forecast_result = result.forecast(steps=len(test))

mse = mean_squared_error(test, forecast_result)
rmse = mse**0.5

print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')

```

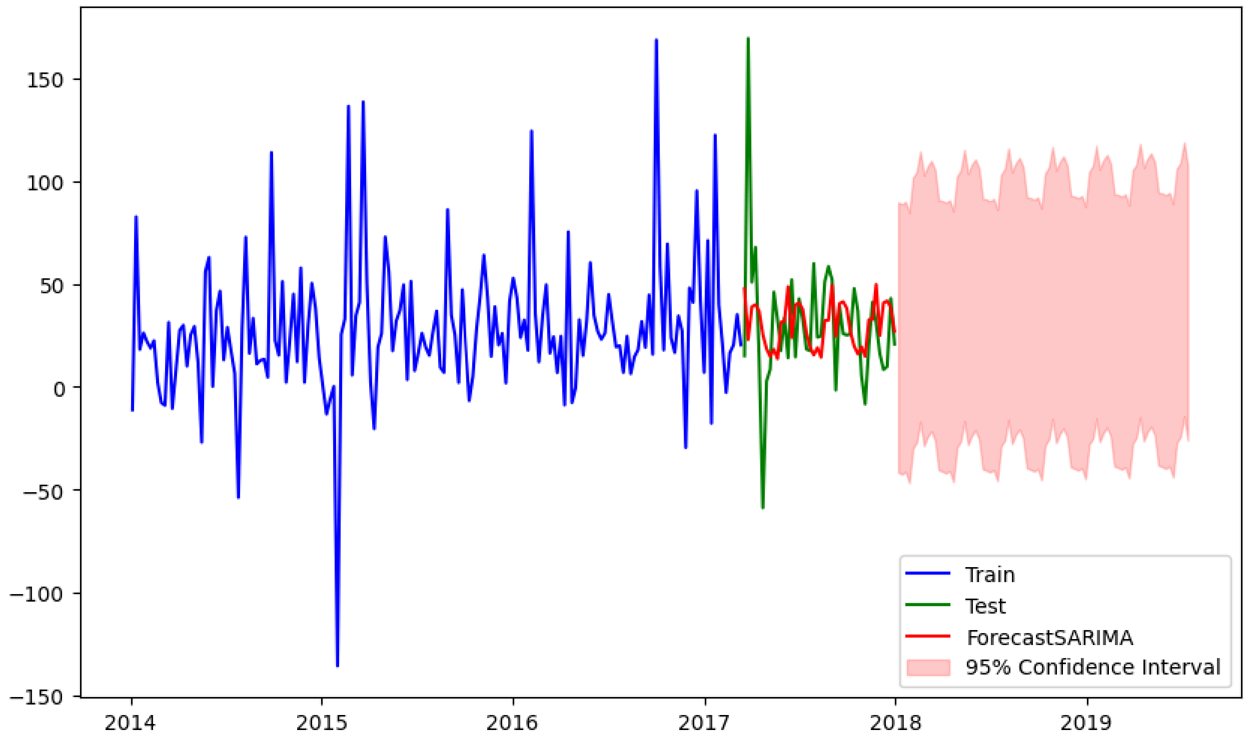
```
Mean Squared Error: 1056.9852814141088
Root Mean Squared Error: 32.5113100537968
```

In [36]:
```
# For cheking the model is good or not
test.mean(), np.sqrt(test.var())
```

Out[36]:
```
(Profit    29.619405
 dtype: float64,
 Profit    31.46318
 dtype: float64)
```

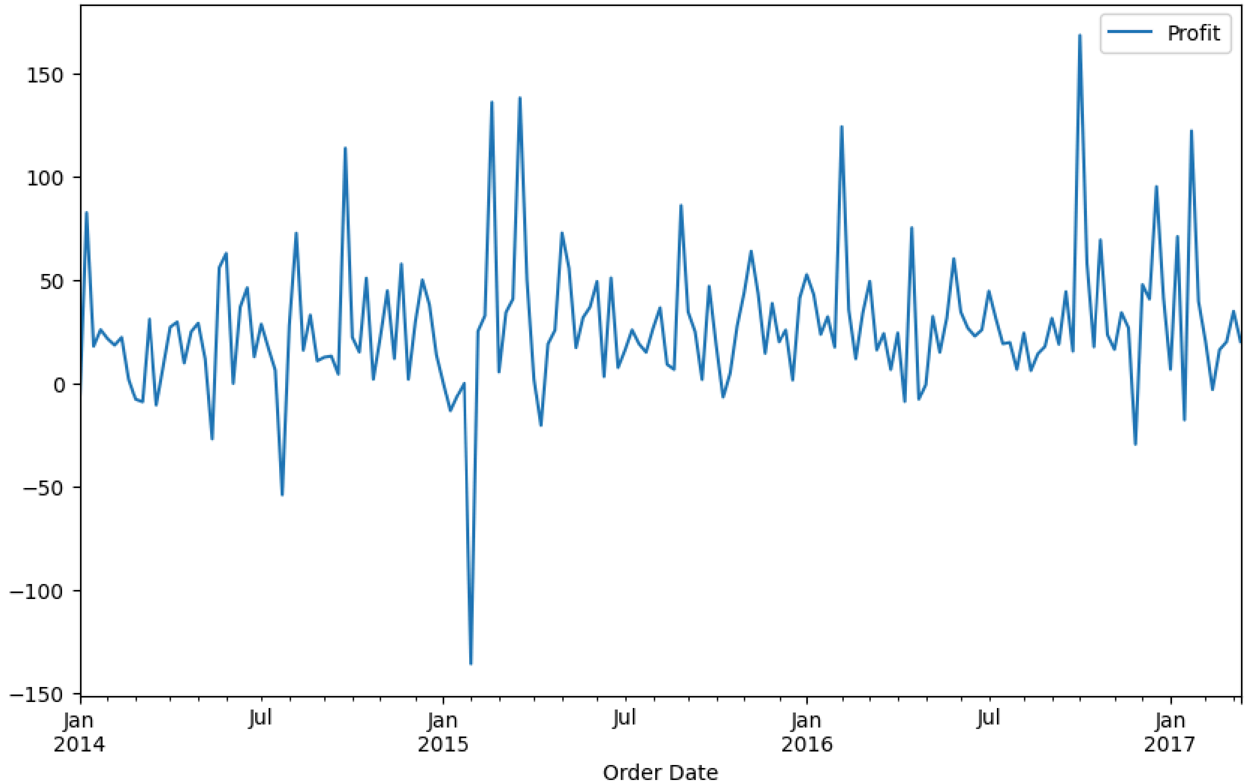
In [37]:

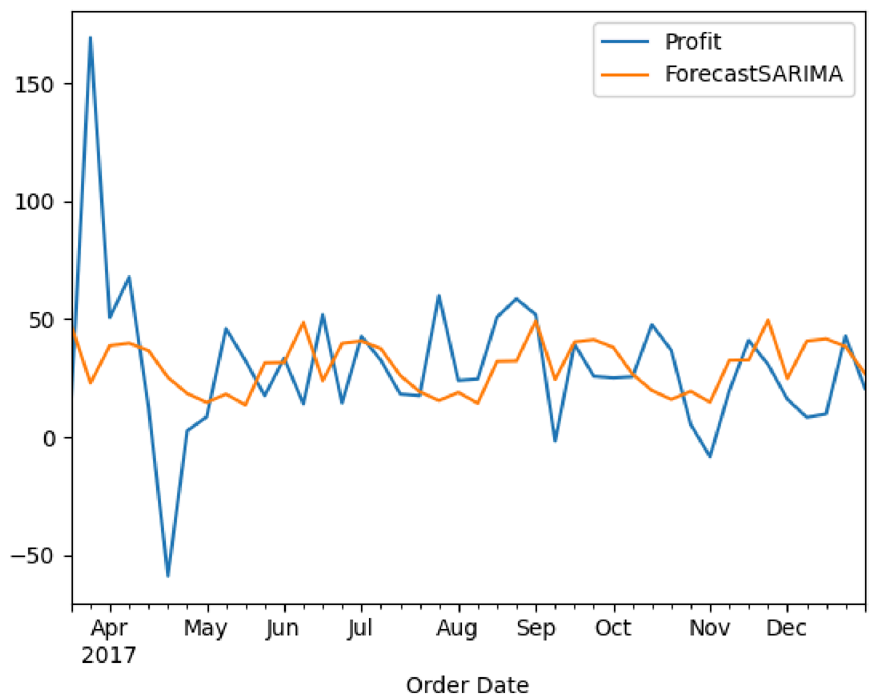
```
# Forecast with confidence intervals
forecast = results.get_forecast(steps=80)
forecast_ci = forecast.conf_int()

# Plot Train, Test, and Forecast with Confidence Intervals
plt.figure(figsize=(10, 6))
plt.plot(train, label='Train', color='blue')
plt.plot(test, label='Test', color='green')
plt.plot(forecast_result, label='ForecastSARIMA', color='red')

plt.fill_between(forecast_ci.index,
                 forecast_ci.iloc[:, 0],
                 forecast_ci.iloc[:, 1], color='red', alpha=0.2, label='95% Confidence Int

plt.legend()
plt.show()
```



In [ ]:
```
train.plot(legend= True, label= 'Train', figsize= (10,6))
test.plot(legend= True, label= 'Test')
forecast_result.plot(legend= True, label= 'ForecastSARIMA')
```

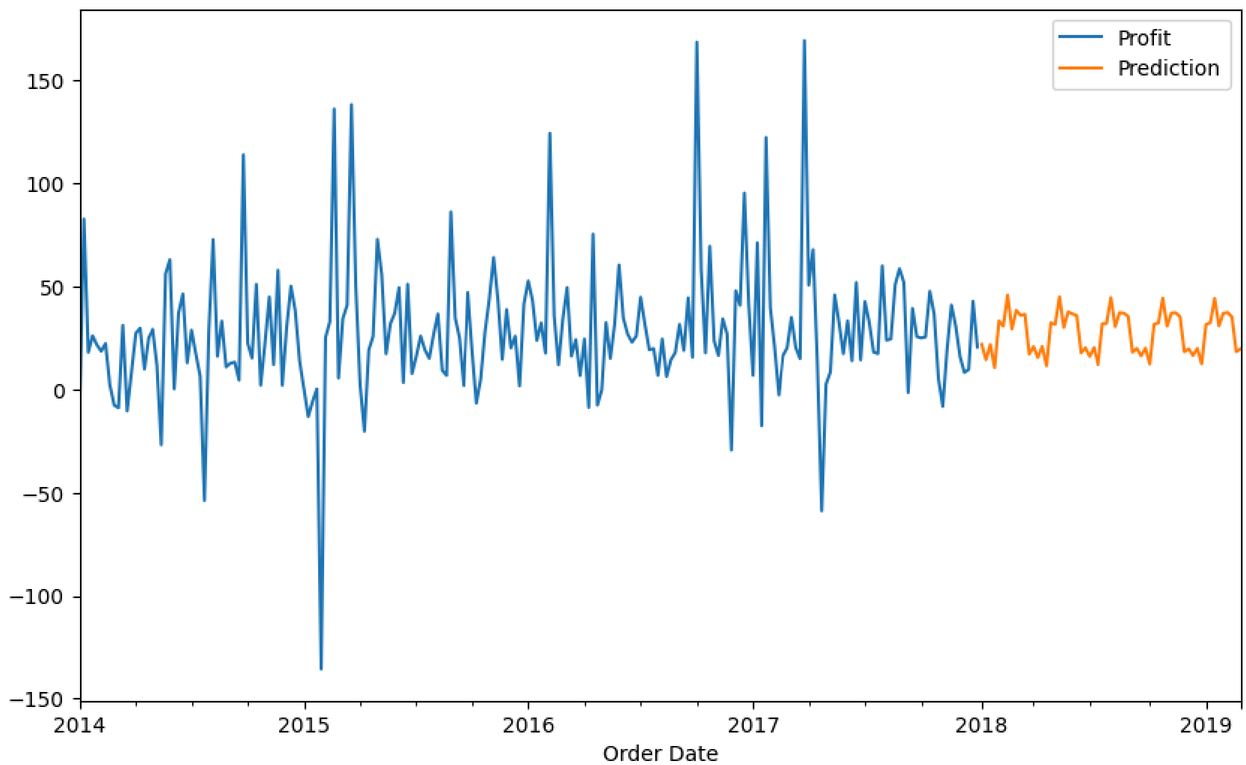
Out[29]: <Axes: xlabel='Order Date'>





In [38]:
```python
final_model= sm.tsa.statespace.SARIMAX(weekly_profit,
                                       order= (1,0,1),
                                       seasonal_order= (0,1,1,12),
                                       enforce_stationarity= False,
                                       enforce_invertibility= False).fit()


predication= final_model.predict(len(df2),len(df2)+60)


df2.plot(legend= True, label= 'Train', figsize= (10,6))
predication.plot(legend= True, label= 'Prediction')
```

Out[38]: <Axes: xlabel='Order Date'>



In [ ]:
```python

```