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
```

In [2]:
```
df=pd.read_csv("Nat_Gas.csv")
df
```

Out[2]:

| | Dates | Prices |
|---|---|---|
| **0** | 10/31/20 | 10.10 |
| **1** | 11/30/20 | 10.30 |
| **2** | 12/31/20 | 11.00 |
| **3** | 1/31/21 | 10.90 |
| **4** | 2/28/21 | 10.90 |
| **5** | 3/31/21 | 10.90 |
| **6** | 4/30/21 | 10.40 |
| **7** | 5/31/21 | 9.84 |
| **8** | 6/30/21 | 10.00 |
| **9** | 7/31/21 | 10.10 |
| **10** | 8/31/21 | 10.30 |
| **11** | 9/30/21 | 10.20 |
| **12** | 10/31/21 | 10.10 |
| **13** | 11/30/21 | 11.20 |
| **14** | 12/31/21 | 11.40 |
| **15** | 1/31/22 | 11.50 |
| **16** | 2/28/22 | 11.80 |
| **17** | 3/31/22 | 11.50 |
| **18** | 4/30/22 | 10.70 |
| **19** | 5/31/22 | 10.70 |
| **20** | 6/30/22 | 10.40 |
| **21** | 7/31/22 | 10.50 |
| **22** | 8/31/22 | 10.40 |
| **23** | 9/30/22 | 10.80 |
| **24** | 10/31/22 | 11.00 |
| **25** | 11/30/22 | 11.60 |
| **26** | 12/31/22 | 11.60 |
| **27** | 1/31/23 | 12.10 |
| **28** | 2/28/23 | 11.70 |
| **29** | 3/31/23 | 12.00 |

| | Dates | Prices |
|---|---|---|
| **30** | 4/30/23 | 11.50 |
| **31** | 5/31/23 | 11.20 |
| **32** | 6/30/23 | 10.90 |
| **33** | 7/31/23 | 11.40 |
| **34** | 8/31/23 | 11.10 |
| **35** | 9/30/23 | 11.50 |
| **36** | 10/31/23 | 11.80 |
| **37** | 11/30/23 | 12.20 |
| **38** | 12/31/23 | 12.80 |
| **39** | 1/31/24 | 12.60 |
| **40** | 2/29/24 | 12.40 |
| **41** | 3/31/24 | 12.70 |
| **42** | 4/30/24 | 12.10 |
| **43** | 5/31/24 | 11.40 |
| **44** | 6/30/24 | 11.50 |
| **45** | 7/31/24 | 11.60 |
| **46** | 8/31/24 | 11.50 |
| **47** | 9/30/24 | 11.80 |

In [3]:
```
df['Dates']=pd.to_datetime(df['Dates'])
```

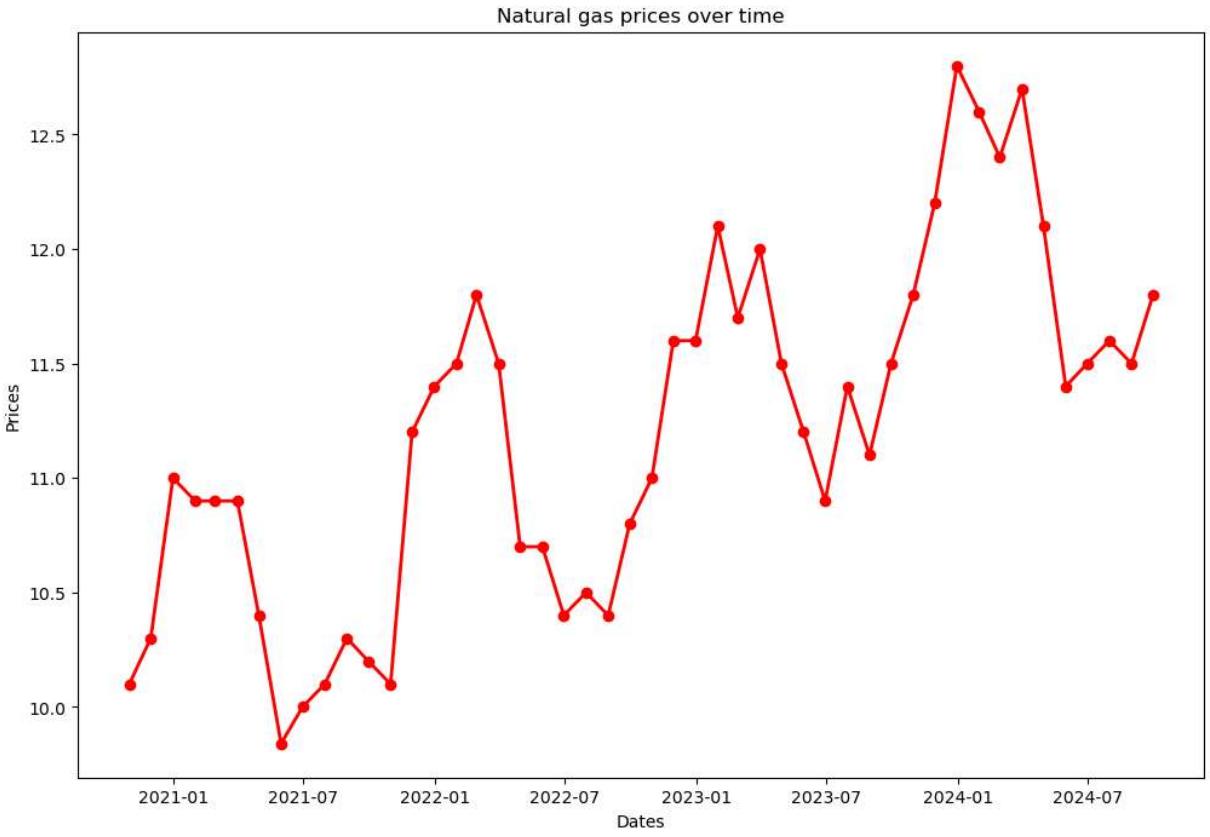
```
C:\Users\hp\AppData\Local\Temp\ipykernel_2508\2958409106.py:1: UserWarning: Could no
t infer format, so each element will be parsed individually, falling back to `dateut
il`. To ensure parsing is consistent and as-expected, please specify a format.
  df['Dates']=pd.to_datetime(df['Dates'])
```

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

Out[4]:

| | Dates | Prices |
|---|---|---|
| **0** | 2020-10-31 | 10.1 |
| **1** | 2020-11-30 | 10.3 |
| **2** | 2020-12-31 | 11.0 |
| **3** | 2021-01-31 | 10.9 |
| **4** | 2021-02-28 | 10.9 |

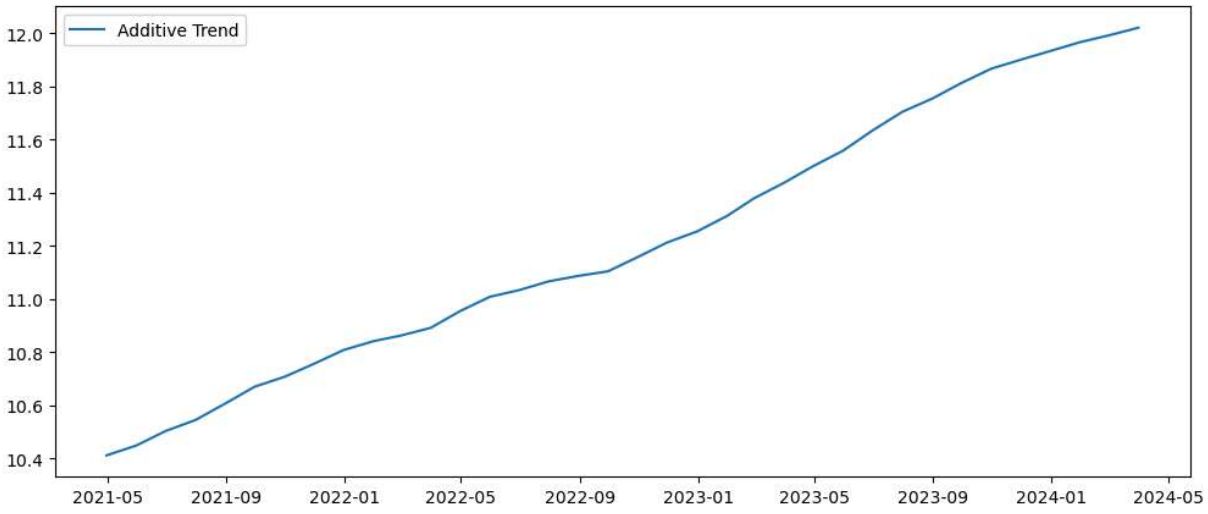
In [5]:
```
plt.figure(figsize=(12,8))
plt.plot(df['Dates'],df['Prices'],marker='o',color='red',linewidth=2)
plt.title("Natural gas prices over time")
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.show()
```



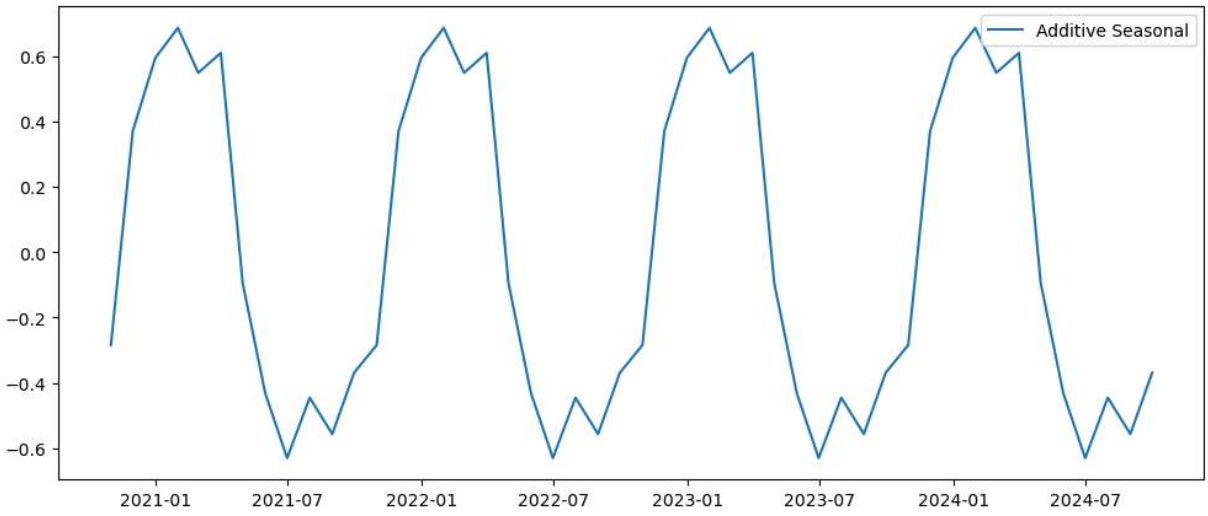

In [6]:
```
df=df.set_index('Dates')
```

In [7]:
```
# additive decomposition
result_add = seasonal_decompose(df, model='additive')
```

In [8]:
```
plt.figure(figsize=(12, 5))
plt.plot(result_add.trend, label='Additive Trend')
plt.legend()
plt.show()
```




In [9]:
```
plt.figure(figsize=(12, 5))

plt.plot(result_add.seasonal, label='Additive Seasonal')
plt.legend(loc='upper right')
plt.show()
```




In [10]:
```
# calculating 4 months simple moving average (sma)
sma_window = 4
sma = df.rolling(window=sma_window).mean()
sma
```

Out[10]:

| Dates | Prices |
|---|---|
| **2020-10-31** | NaN |
| **2020-11-30** | NaN |
| **2020-12-31** | NaN |
| **2021-01-31** | 10.575 |
| **2021-02-28** | 10.775 |
| **2021-03-31** | 10.925 |
| **2021-04-30** | 10.775 |
| **2021-05-31** | 10.510 |
| **2021-06-30** | 10.285 |
| **2021-07-31** | 10.085 |
| **2021-08-31** | 10.060 |
| **2021-09-30** | 10.150 |
| **2021-10-31** | 10.175 |
| **2021-11-30** | 10.450 |
| **2021-12-31** | 10.725 |
| **2022-01-31** | 11.050 |
| **2022-02-28** | 11.475 |
| **2022-03-31** | 11.550 |
| **2022-04-30** | 11.375 |
| **2022-05-31** | 11.175 |
| **2022-06-30** | 10.825 |
| **2022-07-31** | 10.575 |
| **2022-08-31** | 10.500 |
| **2022-09-30** | 10.525 |
| **2022-10-31** | 10.675 |
| **2022-11-30** | 10.950 |
| **2022-12-31** | 11.250 |
| **2023-01-31** | 11.575 |
| **2023-02-28** | 11.750 |

| Dates | Prices |
|---|---|
| **2023-03-31** | 11.850 |
| **2023-04-30** | 11.825 |
| **2023-05-31** | 11.600 |
| **2023-06-30** | 11.400 |
| **2023-07-31** | 11.250 |
| **2023-08-31** | 11.150 |
| **2023-09-30** | 11.225 |
| **2023-10-31** | 11.450 |
| **2023-11-30** | 11.650 |
| **2023-12-31** | 12.075 |
| **2024-01-31** | 12.350 |
| **2024-02-29** | 12.500 |
| **2024-03-31** | 12.625 |
| **2024-04-30** | 12.450 |
| **2024-05-31** | 12.150 |
| **2024-06-30** | 11.925 |
| **2024-07-31** | 11.650 |
| **2024-08-31** | 11.500 |
| **2024-09-30** | 11.600 |

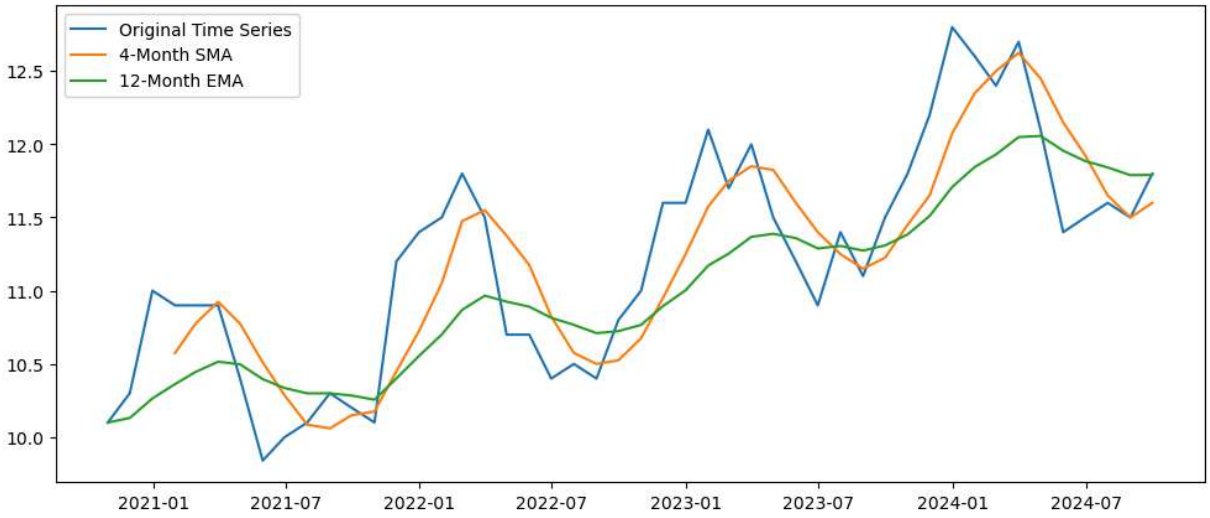
In [11]:
```
# 12 months exponentially moving average (ema)
ema_window = 12  # 12-months moving average
ema = df.ewm(span=ema_window, adjust=False).mean()
ema
```

Out[11]:

| Dates | Prices |
|---|---|
| **2020-10-31** | 10.100000 |
| **2020-11-30** | 10.130769 |
| **2020-12-31** | 10.264497 |
| **2021-01-31** | 10.362267 |
| **2021-02-28** | 10.444995 |
| **2021-03-31** | 10.514996 |
| **2021-04-30** | 10.497304 |
| **2021-05-31** | 10.396180 |
| **2021-06-30** | 10.335230 |
| **2021-07-31** | 10.299040 |
| **2021-08-31** | 10.299188 |
| **2021-09-30** | 10.283928 |
| **2021-10-31** | 10.255632 |
| **2021-11-30** | 10.400919 |
| **2021-12-31** | 10.554624 |
| **2022-01-31** | 10.700066 |
| **2022-02-28** | 10.869287 |
| **2022-03-31** | 10.966320 |
| **2022-04-30** | 10.925347 |
| **2022-05-31** | 10.890679 |
| **2022-06-30** | 10.815190 |
| **2022-07-31** | 10.766699 |
| **2022-08-31** | 10.710284 |
| **2022-09-30** | 10.724086 |
| **2022-10-31** | 10.766534 |
| **2022-11-30** | 10.894760 |
| **2022-12-31** | 11.003258 |
| **2023-01-31** | 11.171988 |
| **2023-02-28** | 11.253221 |

| Dates | Prices |
|---|---|
| **2023-03-31** | 11.368110 |
| **2023-04-30** | 11.388400 |
| **2023-05-31** | 11.359416 |
| **2023-06-30** | 11.288736 |
| **2023-07-31** | 11.305854 |
| **2023-08-31** | 11.274184 |
| **2023-09-30** | 11.308925 |
| **2023-10-31** | 11.384475 |
| **2023-11-30** | 11.509940 |
| **2023-12-31** | 11.708411 |
| **2024-01-31** | 11.845579 |
| **2024-02-29** | 11.930874 |
| **2024-03-31** | 12.049201 |
| **2024-04-30** | 12.057016 |
| **2024-05-31** | 11.955937 |
| **2024-06-30** | 11.885793 |
| **2024-07-31** | 11.841825 |
| **2024-08-31** | 11.789236 |
| **2024-09-30** | 11.790892 |

In [12]:
```
# plotting the moving averages
plt.figure(figsize=(12,5))
plt.plot(df, label='Original Time Series')
plt.plot(sma, label=f"{sma_window}-Month SMA")
plt.plot(ema, label=f"{ema_window}-Month EMA")
plt.legend()
plt.show()
```




In [13]:
```
# Augmented Dickey Fuller test for confirmation of non-stationarity
from statsmodels.tsa.stattools import adfuller
values=df.values
res=adfuller(values)
print('Augmented Dickeyfuller statistic: %f' % res[0])
print('p-value: %f' %res[1])
print('Critical values at different levels: ')
for k,v in res[4].items():
    print('\t%s: %.3f' %(k,v))
```

```
Augmented Dickeyfuller statistic: 0.218077
p-value: 0.973257
Critical values at different levels:
        1%: -3.621
        5%: -2.944
        10%: -2.610
```

In [14]:
```
# same test for log values
values=np.log(df.values)
res=adfuller(values)
print('Augmented Dickeyfuller statistic: %f' % res[0])
print('p-value: %f' %res[1])
print('Critical values at different levels: ')
for k,v in res[4].items():
    print('\t%s: %.3f' %(k,v))
```

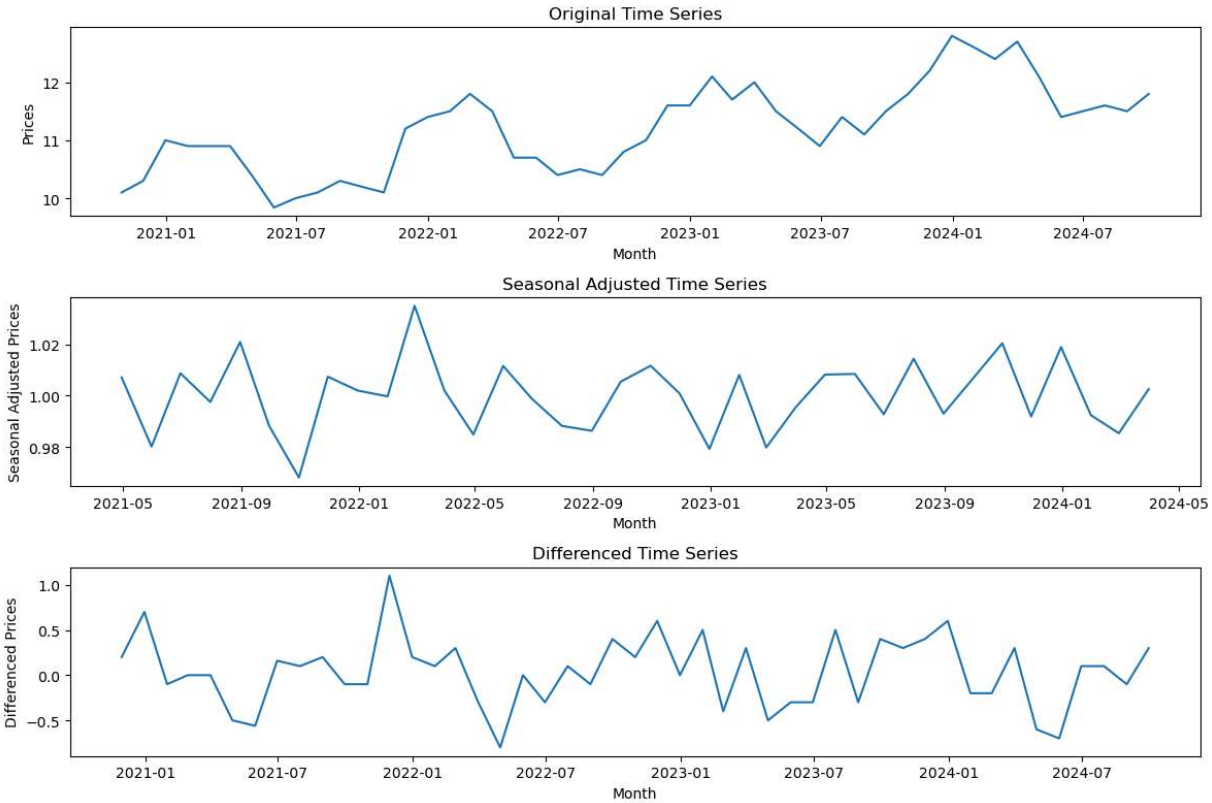
```
Augmented Dickeyfuller statistic: -0.049701
p-value: 0.954224
Critical values at different levels:
        1%: -3.621
        5%: -2.944
        10%: -2.610
```

In [15]:
```
import statsmodels.api as sm
# Seasonal Adjustment (to remove seasonal patterns)
decomposed_result=sm.tsa.seasonal_decompose(df['Prices'],model='multiplicative', pe
df['Seasonal_Adjusted']=decomposed_result.resid
df['Seasonal_Adjusted']
```

Out[15]:
```
Dates
2020-10-31         NaN
2020-11-30         NaN
2020-12-31         NaN
2021-01-31         NaN
2021-02-28         NaN
2021-03-31         NaN
2021-04-30    1.007101
2021-05-31    0.980079
2021-06-30    1.008743
2021-07-31    0.997554
2021-08-31    1.020974
2021-09-30    0.988254
2021-10-31    0.968016
2021-11-30    1.007416
2021-12-31    1.001997
2022-01-31    0.999713
2022-02-28    1.035111
2022-03-31    1.001963
2022-04-30    0.984837
2022-05-31    1.011603
2022-06-30    0.998698
2022-07-31    0.988176
2022-08-31    0.986258
2022-09-30    1.005473
2022-10-31    1.011678
2022-11-30    1.000899
2022-12-31    0.979186
2023-01-31    1.008092
2023-02-28    0.979739
2023-03-31    0.995631
2023-04-30    1.008231
2023-05-31    1.008488
2023-06-30    0.992728
2023-07-31    1.014439
2023-08-31    0.992937
2023-09-30    1.006442
2023-10-31    1.020475
2023-11-30    0.991854
2023-12-31    1.018987
2024-01-31    0.992364
2024-02-29    0.985319
2024-03-31    1.002576
2024-04-30         NaN
2024-05-31         NaN
2024-06-30         NaN
2024-07-31         NaN
2024-08-31         NaN
2024-09-30         NaN
Name: Seasonal_Adjusted, dtype: float64
```

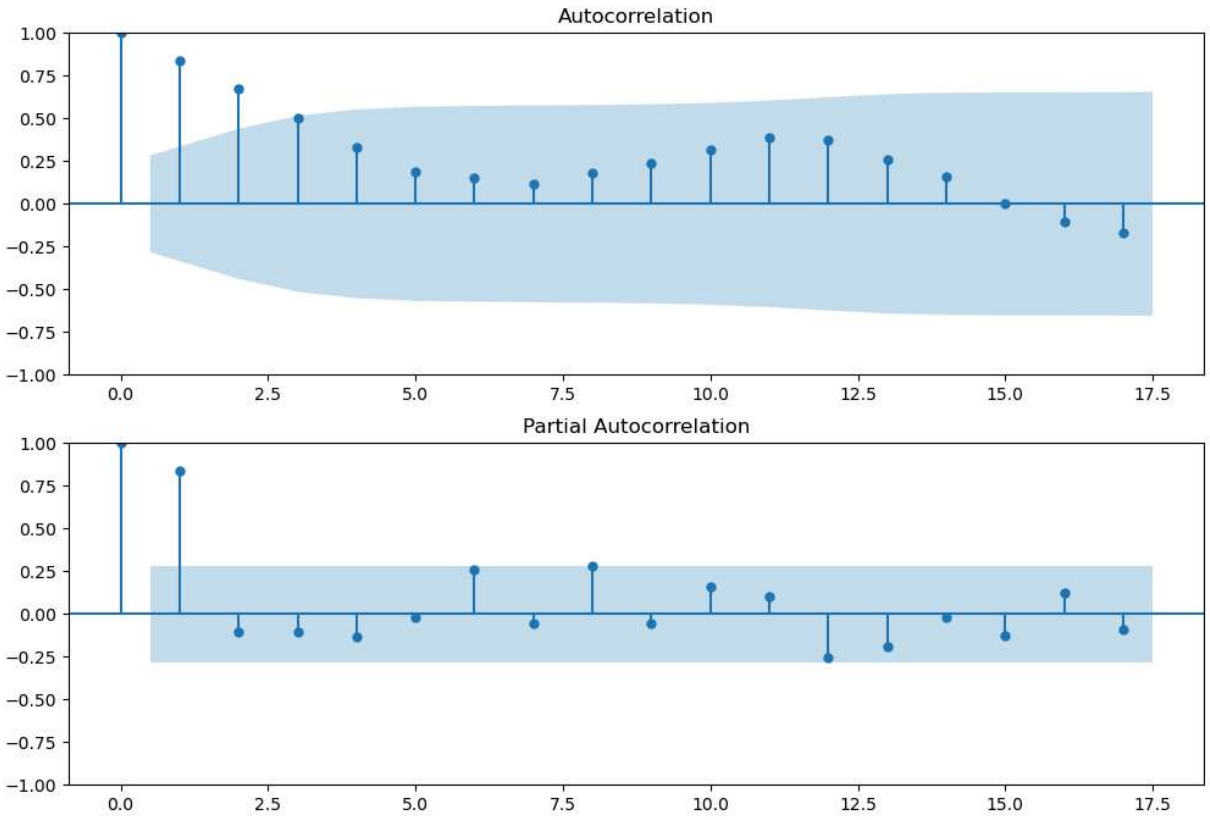
In [16]:
```
# Differencing (to remove trends)
df['Differenced']=df['Prices'].diff()
```

In [17]:
```
# Visualize the results
plt.figure(figsize=(12, 8))

# Original time series
plt.subplot(3, 1, 1)
plt.plot(df['Prices'])
plt.title('Original Time Series')
plt.xlabel('Month')
plt.ylabel('Prices')

# Seasonal Adjusted
plt.subplot(3, 1, 2)
plt.plot(df['Seasonal_Adjusted'])
plt.title('Seasonal Adjusted Time Series')
plt.xlabel('Month')
plt.ylabel('Seasonal Adjusted Prices')

# Differenced
plt.subplot(3, 1, 3)
plt.plot(df['Differenced'])
plt.title('Differenced Time Series')
plt.xlabel('Month')
plt.ylabel('Differenced Prices')

plt.tight_layout()
plt.show()
```




In [18]:
```
# Function to plot ACF and PACF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
def plot_acf_pacf(timeseries):
```

```
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
    plot_acf(timeseries, ax=ax1)
    plot_pacf(timeseries, ax=ax2)
    plt.show()


# Plotting ACF and PACF of the closing value time series
plot_acf_pacf(df['Prices'])
```




In [19]:
```
df
```

Out[19]:

| Dates | Prices | Seasonal_Adjusted | Differenced |
|---|---|---|---|
| **2020-10-31** | 10.10 | NaN | NaN |
| **2020-11-30** | 10.30 | NaN | 0.20 |
| **2020-12-31** | 11.00 | NaN | 0.70 |
| **2021-01-31** | 10.90 | NaN | -0.10 |
| **2021-02-28** | 10.90 | NaN | 0.00 |
| **2021-03-31** | 10.90 | NaN | 0.00 |
| **2021-04-30** | 10.40 | 1.007101 | -0.50 |
| **2021-05-31** | 9.84 | 0.980079 | -0.56 |
| **2021-06-30** | 10.00 | 1.008743 | 0.16 |
| **2021-07-31** | 10.10 | 0.997554 | 0.10 |
| **2021-08-31** | 10.30 | 1.020974 | 0.20 |
| **2021-09-30** | 10.20 | 0.988254 | -0.10 |
| **2021-10-31** | 10.10 | 0.968016 | -0.10 |
| **2021-11-30** | 11.20 | 1.007416 | 1.10 |
| **2021-12-31** | 11.40 | 1.001997 | 0.20 |
| **2022-01-31** | 11.50 | 0.999713 | 0.10 |
| **2022-02-28** | 11.80 | 1.035111 | 0.30 |
| **2022-03-31** | 11.50 | 1.001963 | -0.30 |
| **2022-04-30** | 10.70 | 0.984837 | -0.80 |
| **2022-05-31** | 10.70 | 1.011603 | 0.00 |
| **2022-06-30** | 10.40 | 0.998698 | -0.30 |
| **2022-07-31** | 10.50 | 0.988176 | 0.10 |
| **2022-08-31** | 10.40 | 0.986258 | -0.10 |
| **2022-09-30** | 10.80 | 1.005473 | 0.40 |
| **2022-10-31** | 11.00 | 1.011678 | 0.20 |
| **2022-11-30** | 11.60 | 1.000899 | 0.60 |
| **2022-12-31** | 11.60 | 0.979186 | 0.00 |
| **2023-01-31** | 12.10 | 1.008092 | 0.50 |
| **2023-02-28** | 11.70 | 0.979739 | -0.40 |

| Dates | Prices | Seasonal_Adjusted | Differenced |
|---|---|---|---|
| **2023-03-31** | 12.00 | 0.995631 | 0.30 |
| **2023-04-30** | 11.50 | 1.008231 | -0.50 |
| **2023-05-31** | 11.20 | 1.008488 | -0.30 |
| **2023-06-30** | 10.90 | 0.992728 | -0.30 |
| **2023-07-31** | 11.40 | 1.014439 | 0.50 |
| **2023-08-31** | 11.10 | 0.992937 | -0.30 |
| **2023-09-30** | 11.50 | 1.006442 | 0.40 |
| **2023-10-31** | 11.80 | 1.020475 | 0.30 |
| **2023-11-30** | 12.20 | 0.991854 | 0.40 |
| **2023-12-31** | 12.80 | 1.018987 | 0.60 |
| **2024-01-31** | 12.60 | 0.992364 | -0.20 |
| **2024-02-29** | 12.40 | 0.985319 | -0.20 |
| **2024-03-31** | 12.70 | 1.002576 | 0.30 |
| **2024-04-30** | 12.10 | NaN | -0.60 |
| **2024-05-31** | 11.40 | NaN | -0.70 |
| **2024-06-30** | 11.50 | NaN | 0.10 |
| **2024-07-31** | 11.60 | NaN | 0.10 |
| **2024-08-31** | 11.50 | NaN | -0.10 |
| **2024-09-30** | 11.80 | NaN | 0.30 |

In [20]:
```
df['Differenced'].fillna(method='backfill', inplace=True)
df['Seasonal_Adjusted'].interpolate(method='time', inplace=True)
df
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_2508\2830060183.py:1: FutureWarning: A valu
e is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Differenced'].fillna(method='backfill', inplace=True)
C:\Users\hp\AppData\Local\Temp\ipykernel_2508\2830060183.py:1: FutureWarning: Serie
s.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffi
ll() or obj.bfill() instead.
  df['Differenced'].fillna(method='backfill', inplace=True)
```

Out[20]:

| Dates | Prices | Seasonal_Adjusted | Differenced |
|---|---|---|---|
| **2020-10-31** | 10.10 | NaN | 0.20 |
| **2020-11-30** | 10.30 | NaN | 0.20 |
| **2020-12-31** | 11.00 | NaN | 0.70 |
| **2021-01-31** | 10.90 | NaN | -0.10 |
| **2021-02-28** | 10.90 | NaN | 0.00 |
| **2021-03-31** | 10.90 | NaN | 0.00 |
| **2021-04-30** | 10.40 | 1.007101 | -0.50 |
| **2021-05-31** | 9.84 | 0.980079 | -0.56 |
| **2021-06-30** | 10.00 | 1.008743 | 0.16 |
| **2021-07-31** | 10.10 | 0.997554 | 0.10 |
| **2021-08-31** | 10.30 | 1.020974 | 0.20 |
| **2021-09-30** | 10.20 | 0.988254 | -0.10 |
| **2021-10-31** | 10.10 | 0.968016 | -0.10 |
| **2021-11-30** | 11.20 | 1.007416 | 1.10 |
| **2021-12-31** | 11.40 | 1.001997 | 0.20 |
| **2022-01-31** | 11.50 | 0.999713 | 0.10 |
| **2022-02-28** | 11.80 | 1.035111 | 0.30 |
| **2022-03-31** | 11.50 | 1.001963 | -0.30 |
| **2022-04-30** | 10.70 | 0.984837 | -0.80 |
| **2022-05-31** | 10.70 | 1.011603 | 0.00 |
| **2022-06-30** | 10.40 | 0.998698 | -0.30 |
| **2022-07-31** | 10.50 | 0.988176 | 0.10 |
| **2022-08-31** | 10.40 | 0.986258 | -0.10 |
| **2022-09-30** | 10.80 | 1.005473 | 0.40 |
| **2022-10-31** | 11.00 | 1.011678 | 0.20 |
| **2022-11-30** | 11.60 | 1.000899 | 0.60 |
| **2022-12-31** | 11.60 | 0.979186 | 0.00 |
| **2023-01-31** | 12.10 | 1.008092 | 0.50 |
| **2023-02-28** | 11.70 | 0.979739 | -0.40 |

| Dates | Prices | Seasonal_Adjusted | Differenced |
|---|---|---|---|
| **2023-03-31** | 12.00 | 0.995631 | 0.30 |
| **2023-04-30** | 11.50 | 1.008231 | -0.50 |
| **2023-05-31** | 11.20 | 1.008488 | -0.30 |
| **2023-06-30** | 10.90 | 0.992728 | -0.30 |
| **2023-07-31** | 11.40 | 1.014439 | 0.50 |
| **2023-08-31** | 11.10 | 0.992937 | -0.30 |
| **2023-09-30** | 11.50 | 1.006442 | 0.40 |
| **2023-10-31** | 11.80 | 1.020475 | 0.30 |
| **2023-11-30** | 12.20 | 0.991854 | 0.40 |
| **2023-12-31** | 12.80 | 1.018987 | 0.60 |
| **2024-01-31** | 12.60 | 0.992364 | -0.20 |
| **2024-02-29** | 12.40 | 0.985319 | -0.20 |
| **2024-03-31** | 12.70 | 1.002576 | 0.30 |
| **2024-04-30** | 12.10 | 1.002576 | -0.60 |
| **2024-05-31** | 11.40 | 1.002576 | -0.70 |
| **2024-06-30** | 11.50 | 1.002576 | 0.10 |
| **2024-07-31** | 11.60 | 1.002576 | 0.10 |
| **2024-08-31** | 11.50 | 1.002576 | -0.10 |
| **2024-09-30** | 11.80 | 1.002576 | 0.30 |

In [21]:
```
df['month_index'] = df.index.month
df
```

Out[21]:

| Dates | Prices | Seasonal_Adjusted | Differenced | month_index |
|---|---|---|---|---|
| **2020-10-31** | 10.10 | NaN | 0.20 | 10 |
| **2020-11-30** | 10.30 | NaN | 0.20 | 11 |
| **2020-12-31** | 11.00 | NaN | 0.70 | 12 |
| **2021-01-31** | 10.90 | NaN | -0.10 | 1 |
| **2021-02-28** | 10.90 | NaN | 0.00 | 2 |
| **2021-03-31** | 10.90 | NaN | 0.00 | 3 |
| **2021-04-30** | 10.40 | 1.007101 | -0.50 | 4 |
| **2021-05-31** | 9.84 | 0.980079 | -0.56 | 5 |
| **2021-06-30** | 10.00 | 1.008743 | 0.16 | 6 |
| **2021-07-31** | 10.10 | 0.997554 | 0.10 | 7 |
| **2021-08-31** | 10.30 | 1.020974 | 0.20 | 8 |
| **2021-09-30** | 10.20 | 0.988254 | -0.10 | 9 |
| **2021-10-31** | 10.10 | 0.968016 | -0.10 | 10 |
| **2021-11-30** | 11.20 | 1.007416 | 1.10 | 11 |
| **2021-12-31** | 11.40 | 1.001997 | 0.20 | 12 |
| **2022-01-31** | 11.50 | 0.999713 | 0.10 | 1 |
| **2022-02-28** | 11.80 | 1.035111 | 0.30 | 2 |
| **2022-03-31** | 11.50 | 1.001963 | -0.30 | 3 |
| **2022-04-30** | 10.70 | 0.984837 | -0.80 | 4 |
| **2022-05-31** | 10.70 | 1.011603 | 0.00 | 5 |
| **2022-06-30** | 10.40 | 0.998698 | -0.30 | 6 |
| **2022-07-31** | 10.50 | 0.988176 | 0.10 | 7 |
| **2022-08-31** | 10.40 | 0.986258 | -0.10 | 8 |
| **2022-09-30** | 10.80 | 1.005473 | 0.40 | 9 |
| **2022-10-31** | 11.00 | 1.011678 | 0.20 | 10 |
| **2022-11-30** | 11.60 | 1.000899 | 0.60 | 11 |
| **2022-12-31** | 11.60 | 0.979186 | 0.00 | 12 |
| **2023-01-31** | 12.10 | 1.008092 | 0.50 | 1 |
| **2023-02-28** | 11.70 | 0.979739 | -0.40 | 2 |

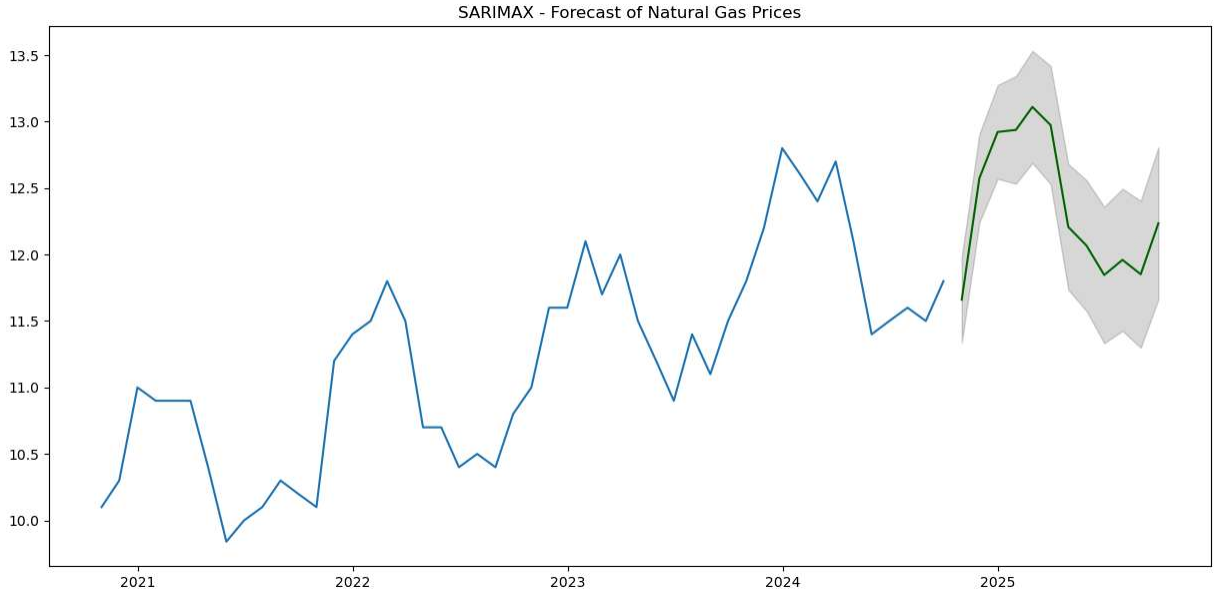
| Dates | Prices | Seasonal_Adjusted | Differenced | month_index |
|---|---|---|---|---|
| **2023-03-31** | 12.00 | 0.995631 | 0.30 | 3 |
| **2023-04-30** | 11.50 | 1.008231 | -0.50 | 4 |
| **2023-05-31** | 11.20 | 1.008488 | -0.30 | 5 |
| **2023-06-30** | 10.90 | 0.992728 | -0.30 | 6 |
| **2023-07-31** | 11.40 | 1.014439 | 0.50 | 7 |
| **2023-08-31** | 11.10 | 0.992937 | -0.30 | 8 |
| **2023-09-30** | 11.50 | 1.006442 | 0.40 | 9 |
| **2023-10-31** | 11.80 | 1.020475 | 0.30 | 10 |
| **2023-11-30** | 12.20 | 0.991854 | 0.40 | 11 |
| **2023-12-31** | 12.80 | 1.018987 | 0.60 | 12 |
| **2024-01-31** | 12.60 | 0.992364 | -0.20 | 1 |
| **2024-02-29** | 12.40 | 0.985319 | -0.20 | 2 |
| **2024-03-31** | 12.70 | 1.002576 | 0.30 | 3 |
| **2024-04-30** | 12.10 | 1.002576 | -0.60 | 4 |
| **2024-05-31** | 11.40 | 1.002576 | -0.70 | 5 |
| **2024-06-30** | 11.50 | 1.002576 | 0.10 | 6 |
| **2024-07-31** | 11.60 | 1.002576 | 0.10 | 7 |
| **2024-08-31** | 11.50 | 1.002576 | -0.10 | 8 |
| **2024-09-30** | 11.80 | 1.002576 | 0.30 | 9 |

In [22]:
```
import pmdarima as pm
SARIMAX_model = pm.auto_arima(df[['Prices']], exogenous=df[['month_index']],
                              start_p=1, start_q=1,
                              test='adf',
                              max_p=3, max_q=3, m=12,
                              start_P=0, seasonal=True,
                              d=None, D=1,
                              trace=False,
                              error_action='ignore',
                              suppress_warnings=True,
                              stepwise=True)
```

In [38]:
```
def sarimax_forecast(SARIMAX_model, periods=24):
    # Forecast
    n_periods = periods

    forecast_df = pd.DataFrame({"month_index": pd.date_range(df.index[-1], periods=
```

```
                                index=pd.date_range(df.index[-1] + pd.DateOffset(mon

    fitted, confint = SARIMAX_model.predict(n_periods=n_periods,
                                            return_conf_int=True,
                                            exogenous=forecast_df[['month_index']])
    index_of_fc = pd.date_range(df.index[-1] + pd.DateOffset(months=1), periods=n_p

    # make series for plotting purpose
    fitted_series = pd.Series(fitted, index=index_of_fc)
    lower_series = pd.Series(confint[:, 0], index=index_of_fc) # lower confidence i
    upper_series = pd.Series(confint[:, 1], index=index_of_fc) # upper confidence i

    # Plot
    plt.figure(figsize=(15, 7))
    plt.plot(df["Prices"], color='#1f76b4')
    plt.plot(fitted_series, color='darkgreen')
    plt.fill_between(lower_series.index,
                     lower_series,
                     upper_series,
                     color='k', alpha=.15)

    plt.title("SARIMAX - Forecast of Natural Gas Prices")
    plt.show()
    print(forecast_df)
    # merged_df=pd.merge(df, forecast_df)
    # merged_df.set_index(df['Dates'])
    # print(merged_df)
```

In [39]:
```
sarimax_forecast(SARIMAX_model, periods=12)
```



| | month_index |
|---|---|
| 2024-10-31 | 9 |
| 2024-11-30 | 10 |
| 2024-12-31 | 11 |
| 2025-01-31 | 12 |
| 2025-02-28 | 1 |
| 2025-03-31 | 2 |
| 2025-04-30 | 3 |
| 2025-05-31 | 4 |
| 2025-06-30 | 5 |
| 2025-07-31 | 6 |
| 2025-08-31 | 7 |
| 2025-09-30 | 8 |

In [ ]: