In [77]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [78]:
```
df = pd.read_csv("ICNSA.csv")
```

In [79]:
```
df.head()
```

Out[79]:

	DATE	ICNSA
0	1967-01-07	346000
1	1967-01-14	334000
2	1967-01-21	277000
3	1967-01-28	252000
4	1967-02-04	274000

In [80]:
```
df.shape
```

Out[80]:
```
(2981, 2)
```

In [81]:
```
df.describe
```

Out[81]:
```
<bound method NDFrame.describe of            DATE   ICNSA
0     1967-01-07  346000
1     1967-01-14  334000
2     1967-01-21  277000
3     1967-01-28  252000
4     1967-02-04  274000
...          ...     ...
2976  2024-01-20  249947
2977  2024-01-27  263919
2978  2024-02-03  234729
2979  2024-02-10  223985
2980  2024-02-17  197932

[2981 rows x 2 columns]>
```

In [82]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2981 entries, 0 to 2980
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   DATE    2981 non-null   object
 1   ICNSA   2981 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 46.7+ KB
```

In [83]:
```
df.isnull().sum()
```

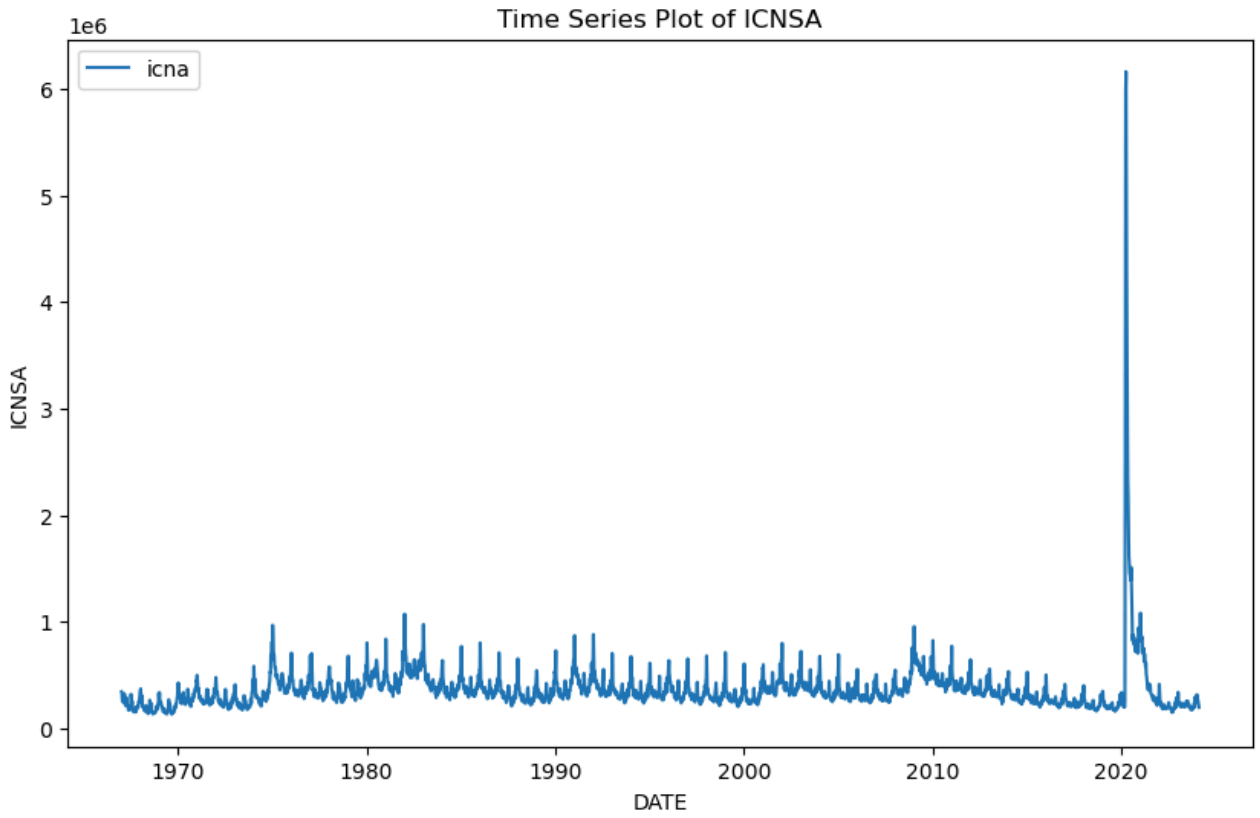
Out[83]:
```
DATE     0
ICNSA    0
dtype: int64
```

In [84]:
```
df['DATE'] = pd.to_datetime(df['DATE'])
df.corr()
```

Out[84]:

	DATE	ICNSA
DATE	1.000000	0.082476
ICNSA	0.082476	1.000000

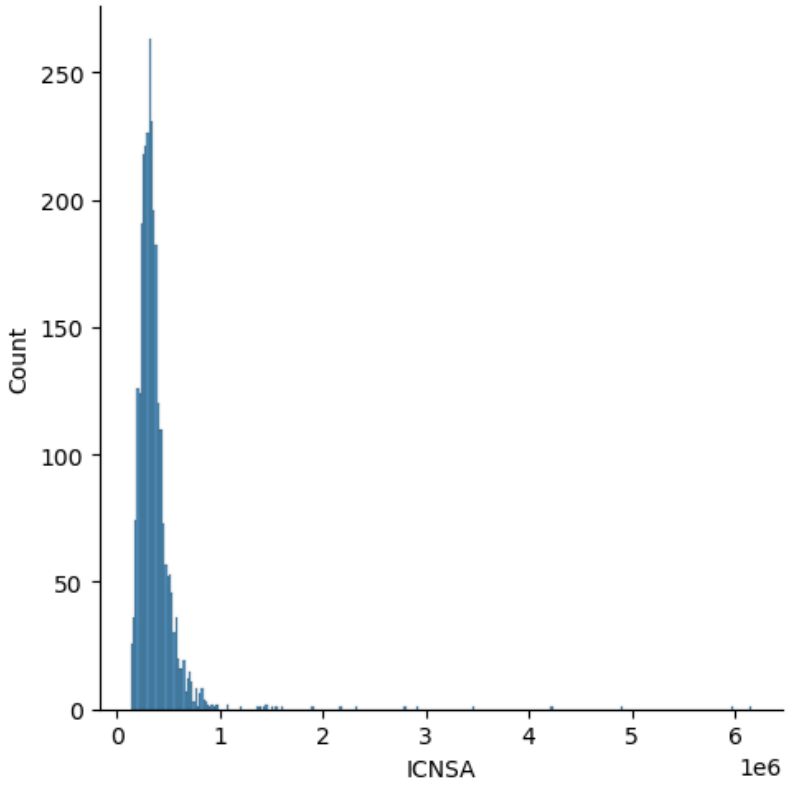
In [85]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['DATE'], df['ICNSA'], label='icna')

plt.xlabel('DATE')
plt.ylabel('ICNSA')
plt.title('Time Series Plot of ICNSA')
plt.legend()
plt.show()
```



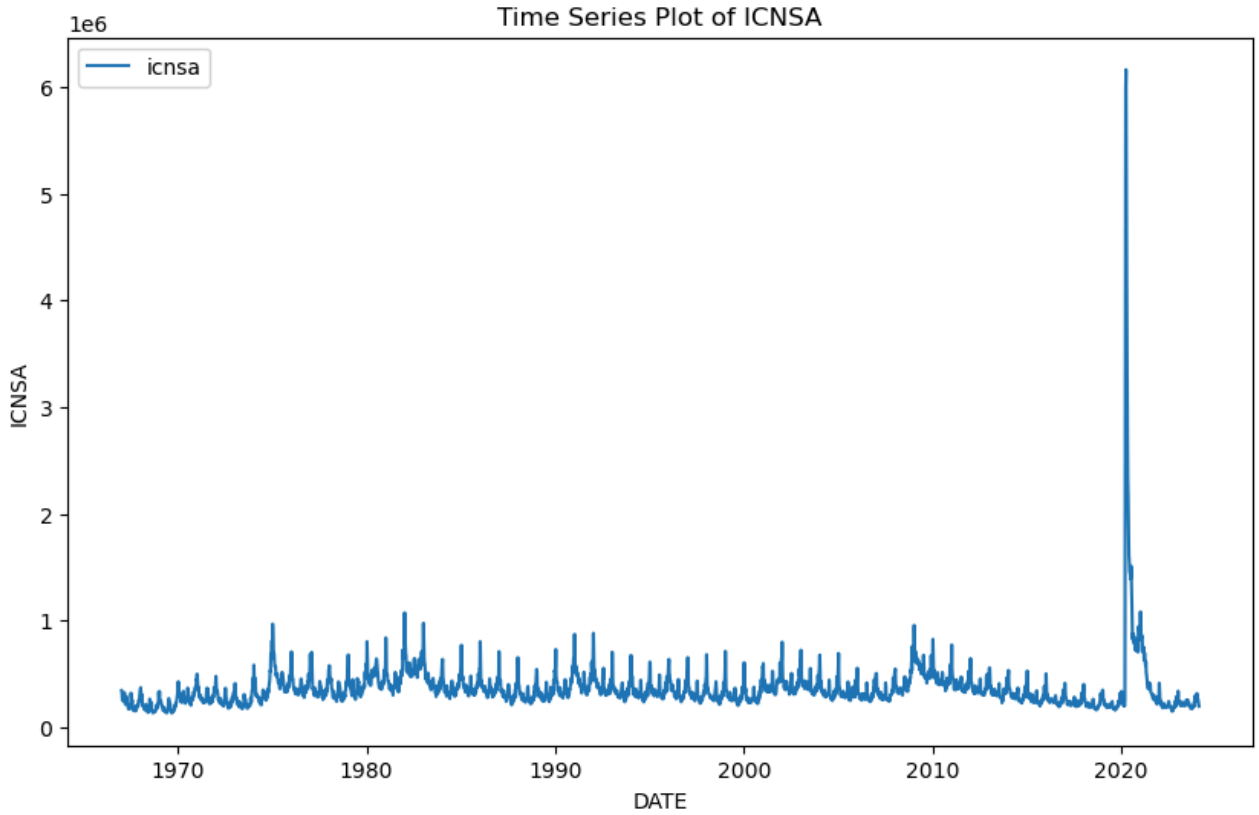
In [86]:
```
sns.displot(df['ICNSA'])
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout h
as changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

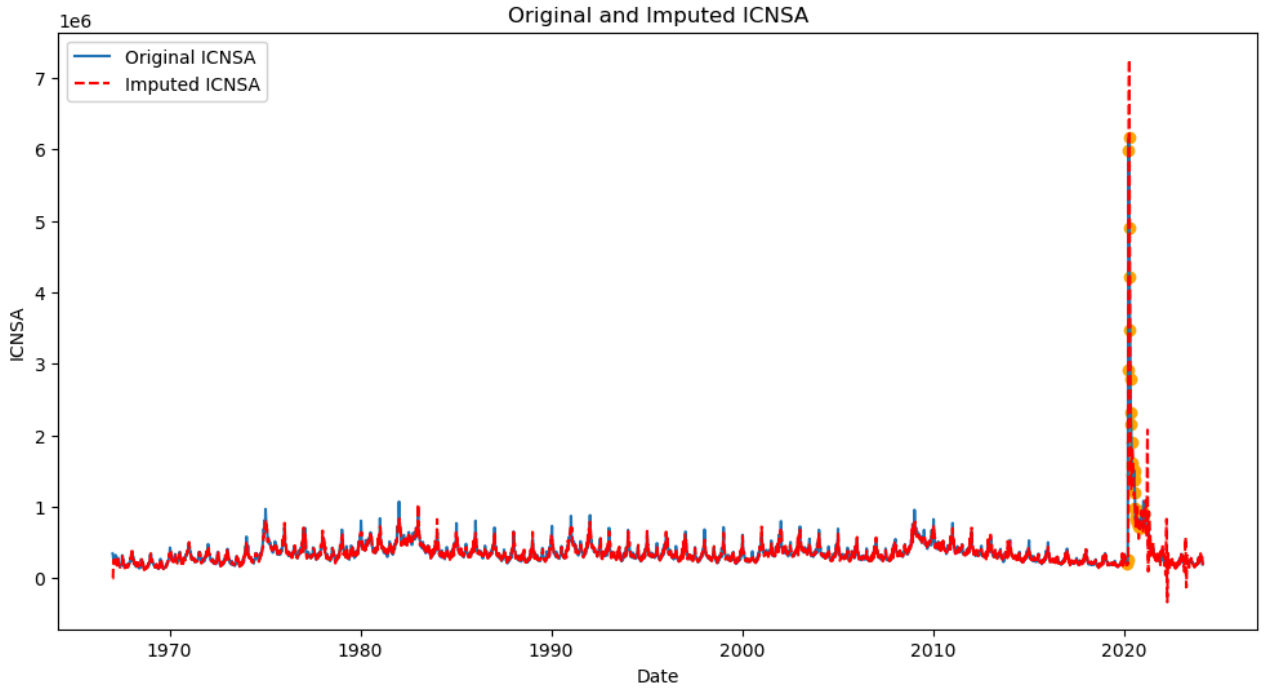
Out[86]:
```
<seaborn.axisgrid.FacetGrid at 0x2b42d072d90>
```



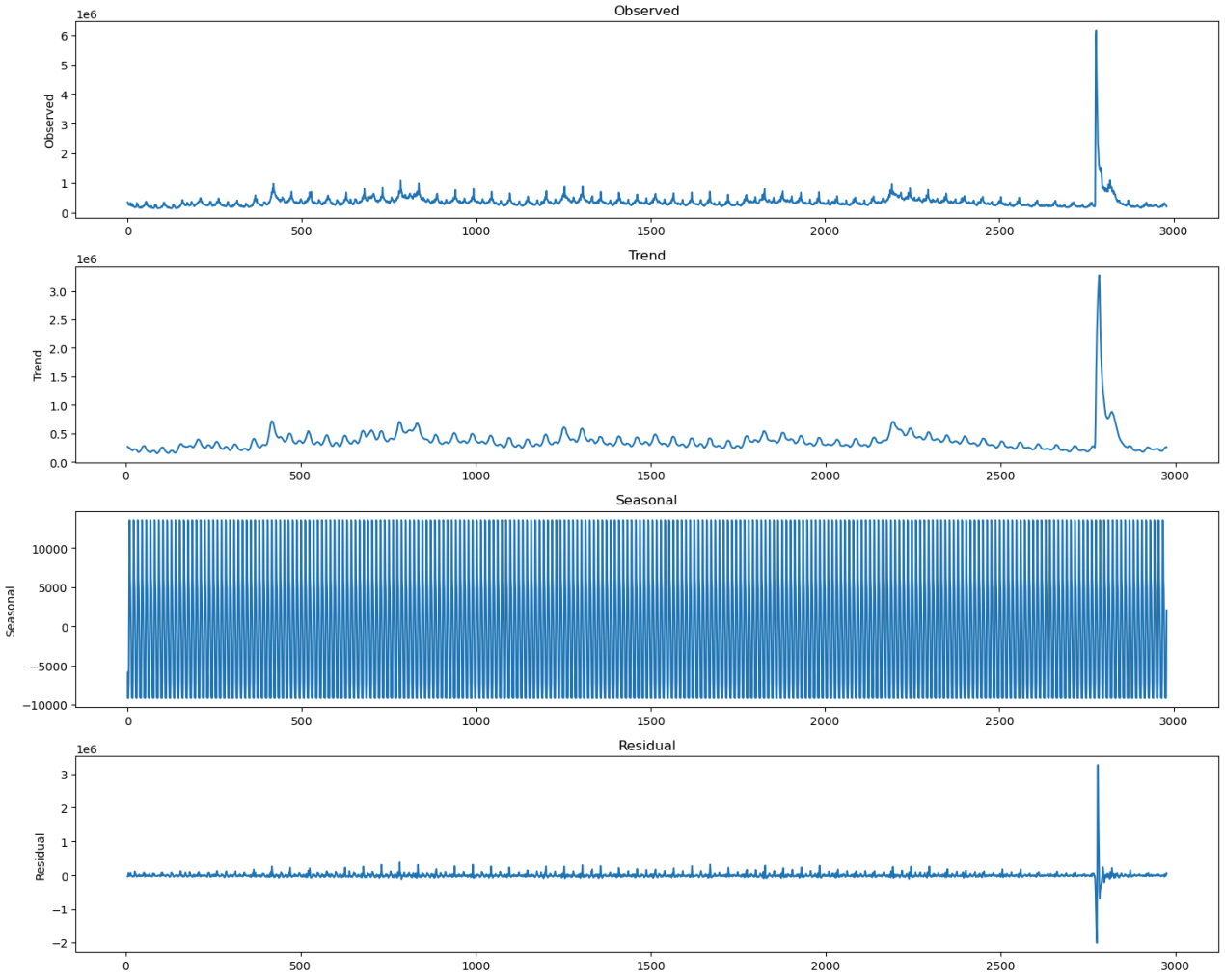
In [87]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['DATE'], df['ICNSA'], label='icnsa')

plt.xlabel('DATE')
plt.ylabel('ICNSA')
plt.title('Time Series Plot of ICNSA')
plt.legend()
plt.show()
```



In [88]:
```
from statsmodels.tsa.statespace.structural import UnobservedComponents
import matplotlib.pyplot as plt


start_date = '2020-03-01'
end_date = '2020-12-31'
covid_period_data = df[(df['DATE'] >= start_date) & (df['DATE'] <= end_date)]

df['ICNSA_imputed'] = UnobservedComponents(df['ICNSA'],freq_seasonal=[{'period': 52, 'harmonics': 4}]

plt.figure(figsize=(12, 6))
plt.plot(df['DATE'], df['ICNSA'], label='Original ICNSA')
plt.plot(df['DATE'], df['ICNSA_imputed'], label='Imputed ICNSA', linestyle='--',color = 'red')
plt.scatter(covid_period_data['DATE'], covid_period_data['ICNSA'], color='orange')
plt.xlabel('Date')
plt.ylabel('ICNSA')
plt.title('Original and Imputed ICNSA')
plt.legend()
plt.show()
```



In [89]:
```
import matplotlib.pyplot as plt

decomposition_result = seasonal_decompose(df['ICNSA'], period=seasonal_period)

fig, axes = plt.subplots(4, 1, figsize=(15, 12))

decomposition_result.observed.plot(ax=axes[0], title='Observed')
axes[0].set_ylabel('Observed')

decomposition_result.trend.plot(ax=axes[1], title='Trend')
axes[1].set_ylabel('Trend')

decomposition_result.seasonal.plot(ax=axes[2], title='Seasonal')
axes[2].set_ylabel('Seasonal')

decomposition_result.resid.plot(ax=axes[3], title='Residual')
axes[3].set_ylabel('Residual')

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



In [90]:
```
df_INDPRO= pd.read_csv("INDPRO.csv")
```

In [91]:
```
df_INDPRO.head()
```

Out[91]:

	DATE	INDPRO
0	1919-01-01	4.8665
1	1919-02-01	4.6514
2	1919-03-01	4.5170
3	1919-04-01	4.5976
4	1919-05-01	4.6245

In [92]:
```
df_INDPRO.shape
```

Out[92]:
```
(1261, 2)
```

In [93]:
```
df_INDPRO.describe
```

Out[93]:
```
<bound method NDFrame.describe of            DATE    INDPRO
0     1919-01-01    4.8665
1     1919-02-01    4.6514
2     1919-03-01    4.5170
3     1919-04-01    4.5976
4     1919-05-01    4.6245
...          ...       ...
1256  2023-09-01  103.2096
1257  2023-10-01  102.3722
1258  2023-11-01  102.6710
1259  2023-12-01  102.6715
1260  2024-01-01  102.5739

[1261 rows x 2 columns]>
```

In [94]:
```
df_INDPRO.info()
```

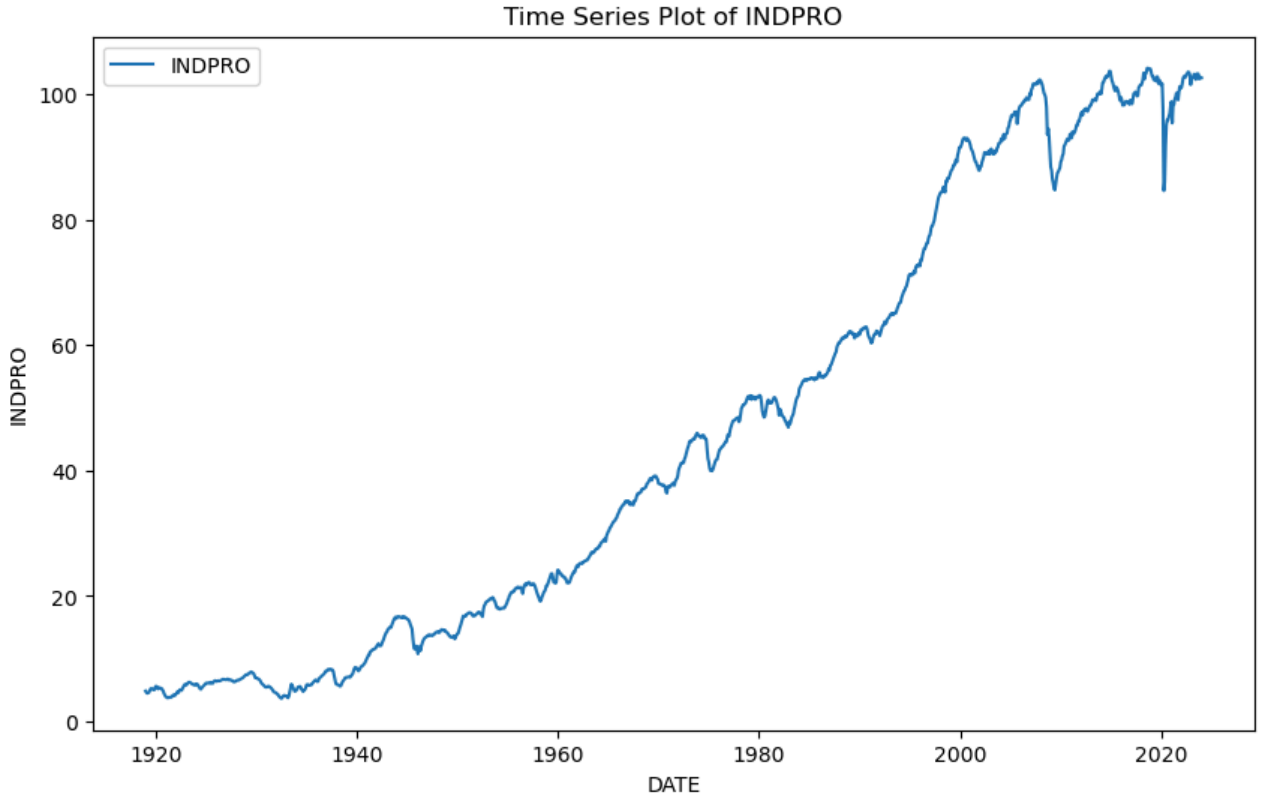
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1261 entries, 0 to 1260
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   DATE    1261 non-null   object 
 1   INDPRO  1261 non-null   float64
dtypes: float64(1), object(1)
memory usage: 19.8+ KB
```

In [95]:
```
df_INDPRO['DATE'] = pd.to_datetime(df_INDPRO['DATE'])
df_INDPRO.corr()
```

Out[95]:

	DATE	INDPRO
DATE	1.000000	0.974652
INDPRO	0.974652	1.000000

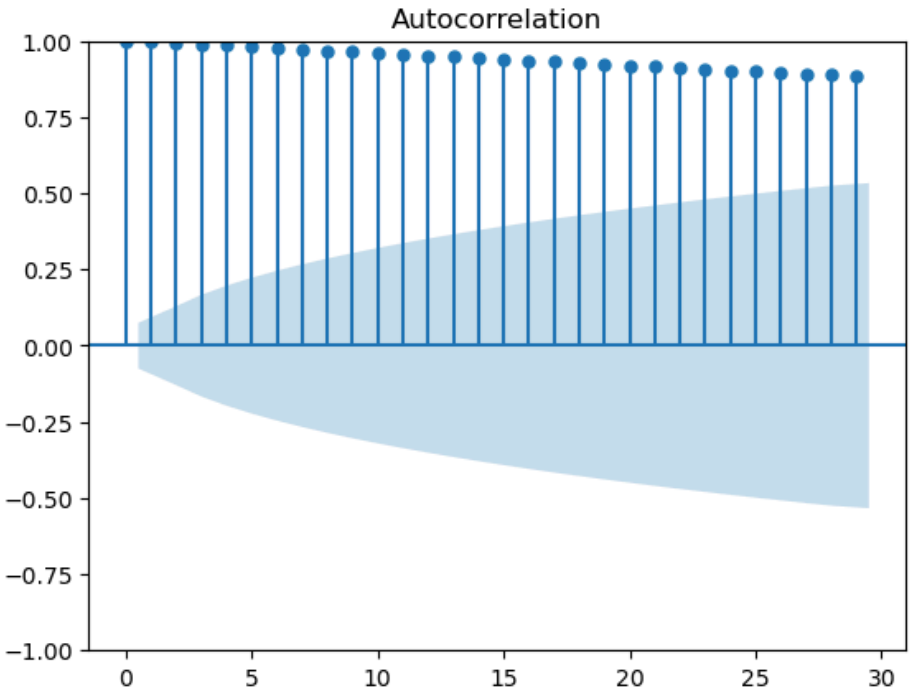
In [96]:
```
plt.figure(figsize=(10, 6))
plt.plot(df_INDPRO['DATE'], df_INDPRO['INDPRO'], label='INDPRO')

plt.xlabel('DATE')
plt.ylabel('INDPRO')
plt.title('Time Series Plot of INDPRO')
plt.legend()
plt.show()
```

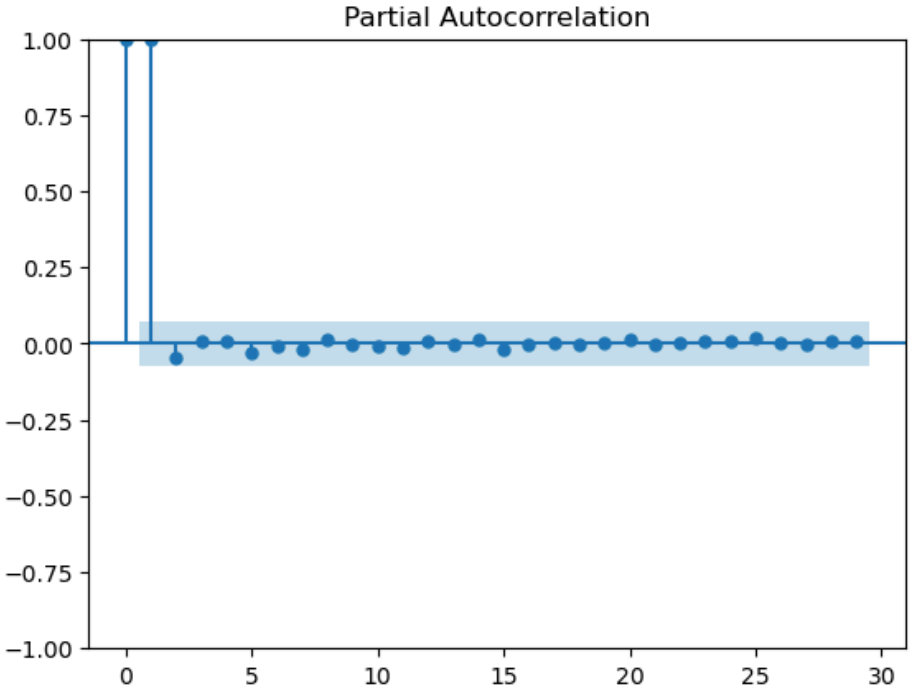


In [97]:
```
df_month = df.resample('M', on='DATE').mean().reset_index()


df_month['DATE'] = df_month['DATE'] + pd.offsets.MonthEnd(0)
df_INDPRO['DATE'] = df_INDPRO['DATE'] + pd.offsets.MonthEnd(0)


merged_df = pd.merge(df_month, df_INDPRO, on='DATE', how='inner')
print(merged_df.head(10))
```

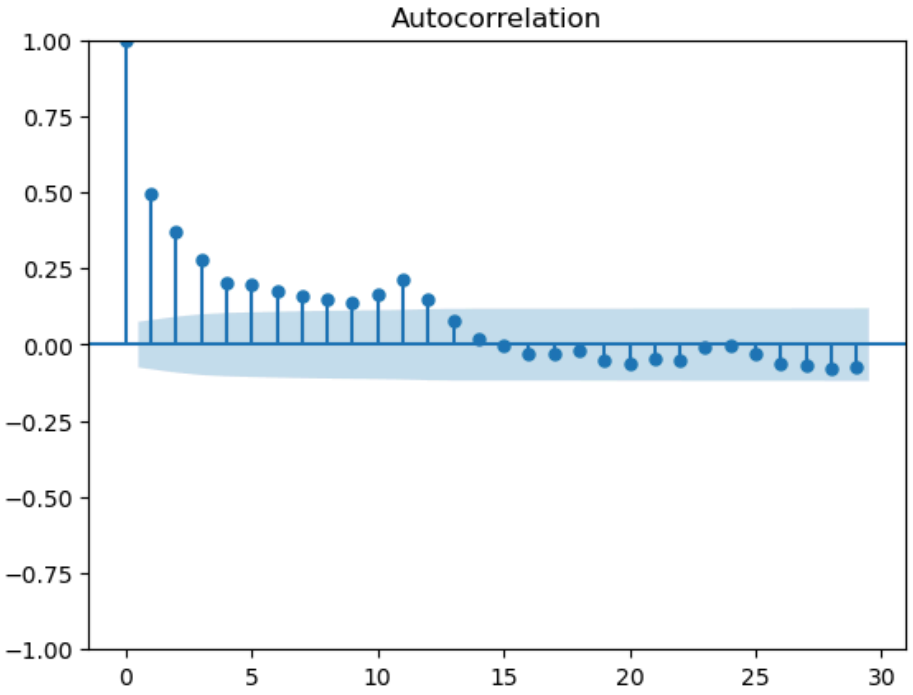
```
        DATE     ICNSA  ICNSA_imputed   INDPRO
0 1967-01-31  302250.0  112357.076327  35.1716
1 1967-02-28  261250.0  201711.651656  34.7727
2 1967-03-31  251500.0  219033.014210  34.5767
3 1967-04-30  239400.0  216012.687755  34.9028
4 1967-05-31  189000.0  172660.052122  34.5983
5 1967-06-30  179250.0  172408.894090  34.5940
6 1967-07-31  272400.0  258204.059244  34.5152
7 1967-08-31  194250.0  178532.132919  35.1765
8 1967-09-30  158600.0  164476.668589  35.1194
9 1967-10-31  177250.0  165251.438385  35.4054
```

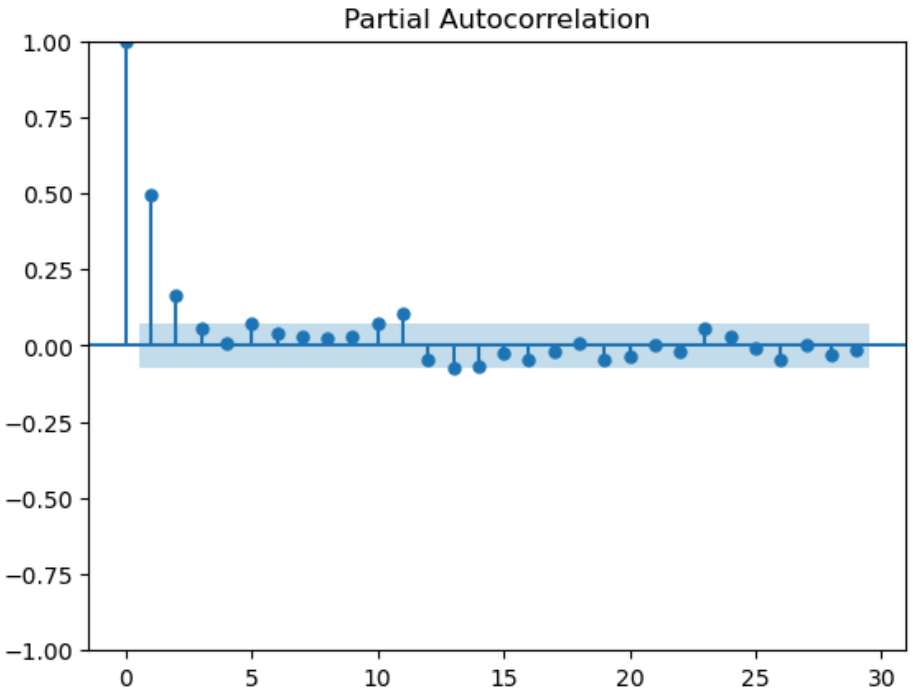
In [98]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plot_acf(merged_df['INDPRO'])
plt.show()

plot_pacf(merged_df['INDPRO'])
plt.show()
```





In [99]:
```
plot_acf(merged_df['ICNSA_imputed'])
plt.show()

plot_pacf(merged_df['ICNSA_imputed'])
plt.show()
```





In [100]:
```
from sklearn.model_selection import train_test_split

X = merged_df[['INDPRO']]
y = merged_df['ICNSA_imputed']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [101]:
```
from pmdarima import auto_arima

arima_model = auto_arima(y_train, exogenous=X_train, seasonal=True, suppress_warnings=True)
print(arima_model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  548
Model:                        SARIMAX   Log Likelihood               -7626.683
Date:                Thu, 07 Mar 2024   AIC                          15257.365
Time:                        16:16:00   BIC                          15265.978
Sample:                             0   HQIC                         15260.732
                                  - 548
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept   3.695e+05   2.22e+04     16.661      0.000    3.26e+05    4.13e+05
sigma2      7.177e+10   7.38e+08     97.246      0.000    7.03e+10    7.32e+10
===================================================================================
Ljung-Box (L1) (Q):                   0.73   Jarque-Bera (JB):           1529023.78
Prob(Q):                              0.39   Prob(JB):                         0.00
Heteroskedasticity (H):               5.61   Skew:                            13.79
Prob(H) (two-sided):                  0.00   Kurtosis:                       260.30
===================================================================================

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

In [102]:
```
forecast, conf_int = arima_model.predict(n_periods=len(y_test), exogenous=X_test, return_conf_int=Tru
forecast_df = pd.DataFrame({
    'Date': y_test.index,
    'Actual': y_test.values,
    'Forecast': forecast,
    'Lower_CI': conf_int[:, 0],
    'Upper_CI': conf_int[:, 1]
})

print(forecast_df)
```

```
     Date         Actual       Forecast       Lower_CI       Upper_CI
548   294  441439.203202  369548.570945 -155524.032143  894621.174033
549   545  374631.644097  369548.570945 -155524.032143  894621.174033
550   213  361496.677692  369548.570945 -155524.032143  894621.174033
551   328  295644.387567  369548.570945 -155524.032143  894621.174033
552   164  385318.933394  369548.570945 -155524.032143  894621.174033
..    ...            ...            ...            ...            ...
680   275  466455.617971  369548.570945 -155524.032143  894621.174033
681   110  359844.177967  369548.570945 -155524.032143  894621.174033
682    82  251707.670403  369548.570945 -155524.032143  894621.174033
683    51  265251.730035  369548.570945 -155524.032143  894621.174033
684   212  309999.419005  369548.570945 -155524.032143  894621.174033

[137 rows x 5 columns]

C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No s
upported index is available. Prediction results will be given with an integer index beginning at `st
art`.
  return get_prediction_index(
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No
supported index is available. In the next version, calling this method in a model without a supporte
d index will result in an exception.
  return get_prediction_index(
```

In [103]:
```
last_forecast_row = forecast_df.iloc[-1]

forecast_value = last_forecast_row['Forecast']
lower_ci = last_forecast_row['Lower_CI']
upper_ci = last_forecast_row['Upper_CI']

print(f"So I forecast the new data release value will be {forecast_value:.1f} with confidence interva
```

```
So I forecast the new data release value will be 369548.6 with confidence interval (-155524.0, 89462
1.2)
```