In [166]:
```
from pmdarima import auto_arima
import pandas as pd
import matplotlib.pyplot as plt
import requests
import io
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose




# URL of the dataset
url = "https://fred.stlouisfed.org/graph/fredgraph.csv?id=ICSA"

# Fetching the data from the URL
response = requests.get(url)

# Reading the data into a DataFrame
df = pd.read_csv(io.StringIO(response.text))

print(df.head(10))  # Displays the first 10 rows
```

```
         DATE    ICSA
0  1967-01-07  208000
1  1967-01-14  207000
2  1967-01-21  217000
3  1967-01-28  204000
4  1967-02-04  216000
5  1967-02-11  229000
6  1967-02-18  229000
7  1967-02-25  242000
8  1967-03-04  310000
9  1967-03-11  241000
```

In [167]:
```
df.shape
```

Out[167]: (2983, 2)

In [168]:
```
# Convert 'DATE' column to datetime
print('DATE data type before conversion:')
print(df['DATE'].dtypes)
df['DATE'] = pd.to_datetime(df['DATE'])
print('DATE data type after conversion:')
print(df['DATE'].dtypes)
```

```
DATE data type before conversion:
object
DATE data type after conversion:
datetime64[ns]
```

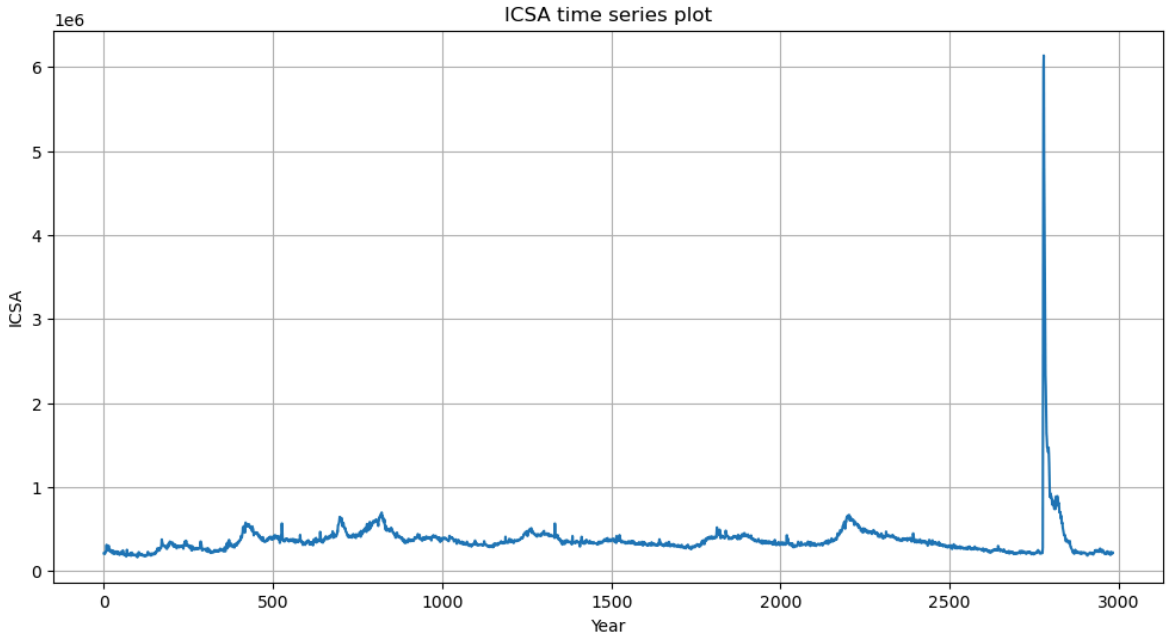
In [169]:
```
# Check for missing values
missing_values = df.isnull().sum()
# Check for duplicates
duplicates = df.duplicated().sum()

# Handling missing values (if any, here we just print them)
print(f"Missing values:\n{missing_values}")
print(f"Duplicate values:\n{duplicates}")
```

```
Missing values:
DATE    0
ICSA    0
dtype: int64
Duplicate values:
0
```

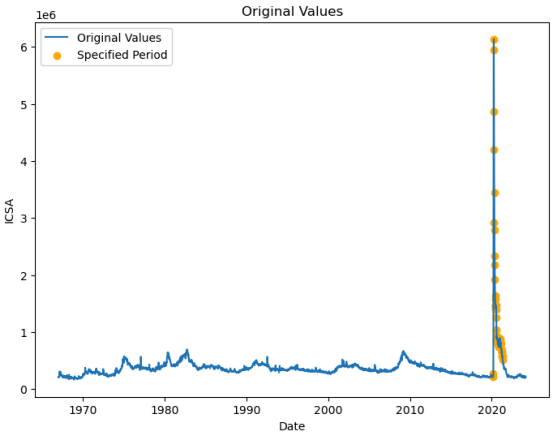
In [170]:
```
print(f"Statistical Summary:")
print(df.describe())
```

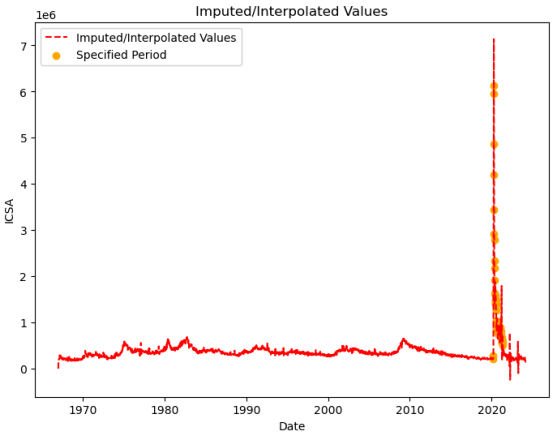
```
Statistical Summary:
                                DATE          ICSA
count                           2983  2.983000e+03
mean   1995-08-05 00:00:00.000000128  3.652685e+05
min              1967-01-07 00:00:00  1.620000e+05
25%              1981-04-21 12:00:00  2.910000e+05
50%              1995-08-05 00:00:00  3.420000e+05
75%              2009-11-17 12:00:00  3.990000e+05
max              2024-03-02 00:00:00  6.137000e+06
std                              NaN  2.417745e+05
```

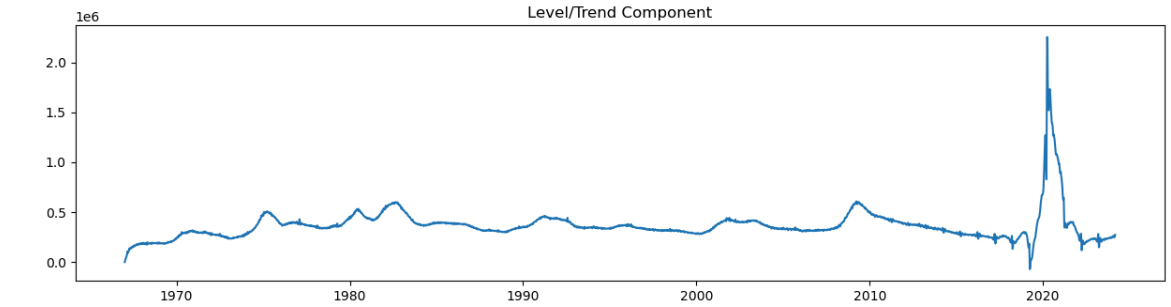
In [171]:
```
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['ICSA'])
plt.title('ICSA time series plot')
plt.xlabel('Year')
plt.ylabel('ICSA')
plt.grid(True)
plt.show()
```

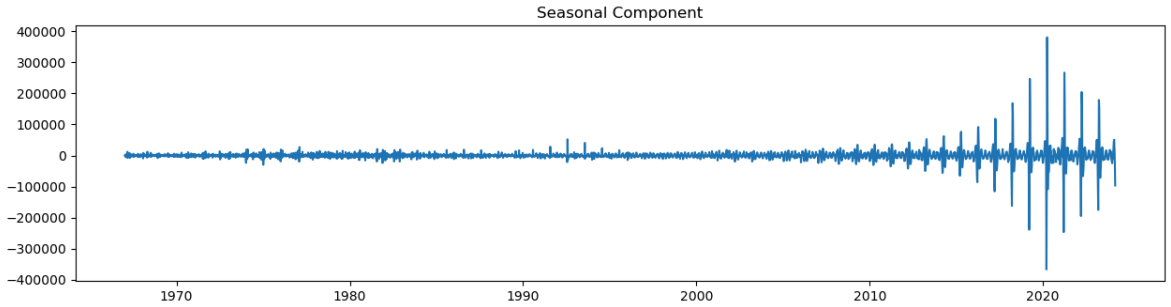


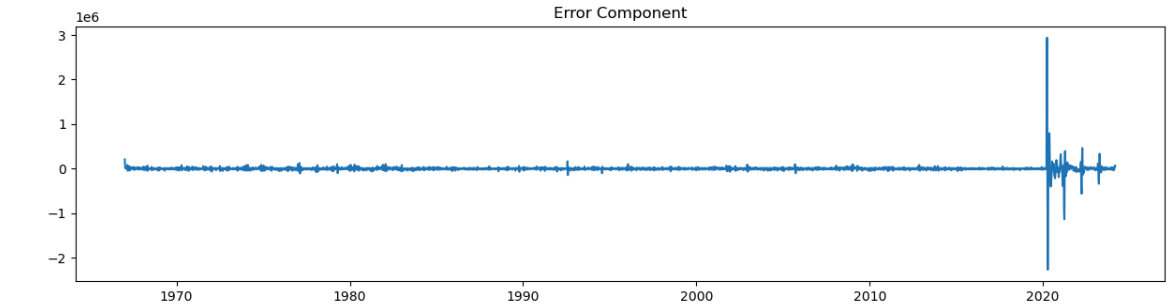
In [172]:
```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.structural import UnobservedComponents
import matplotlib.pyplot as plt


start_date = '2020-03-01'
end_date = '2021-05-01'
period_data = df[(df['DATE'] >= start_date) & (df['DATE'] <= end_date)]

results = UnobservedComponents(df['ICSA'],level='llevel',seasonal=52,freq_seas

fig, axs = plt.subplots(1, 2, figsize=(18, 6))

df['ICSA_imputed'] = results.predict()

axs[0].plot(df['DATE'], df['ICSA'], label='Original Values')
axs[0].scatter(period_data['DATE'], period_data['ICSA'], color='orange', label
axs[0].set_xlabel('Date')
axs[0].set_ylabel('ICSA')
axs[0].set_title('Original Values')
axs[0].legend()


axs[1].plot(df['DATE'], df['ICSA_imputed'], label='Imputed/Interpolated Values
axs[1].scatter(period_data['DATE'], period_data['ICSA'], color='orange', label
axs[1].set_xlabel('Date')
axs[1].set_ylabel('ICSA')
axs[1].set_title('Imputed/Interpolated Values')
axs[1].legend()

plt.show()

fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(15, 12))

axes[0].plot(df['DATE'], results.level.smoothed, label='Level')
axes[0].set_title('Level Component')


axes[1].plot(df['DATE'], results.seasonal.smoothed, label='Seasonal')
axes[1].set_title('Seasonal Component')


axes[2].plot(df['DATE'], results.resid, label='Error')
axes[2].set_title('Error Component')

plt.show()
```


Original Values
1e6
Original Values
Specified Period
ICSA
Date


Imputed/Interpolated Values
1e6
Imputed/Interpolated Values
Specified Period
ICSA
Date


Level/Trend Component
1e6


Seasonal Component


Error Component
1e6

In [173]:
```
url1 = "https://fred.stlouisfed.org/graph/fredgraph.csv?id=CCSA"

# Fetching the data from the URL
response = requests.get(url1)

# Reading the data into a DataFrame
df1 = pd.read_csv(io.StringIO(response.text))
print(df1.head(10))
```

```
         DATE     CCSA
0  1967-01-07  1134000
1  1967-01-14  1119000
2  1967-01-21  1119000
3  1967-01-28  1103000
4  1967-02-04  1131000
5  1967-02-11  1153000
6  1967-02-18  1167000
7  1967-02-25  1199000
8  1967-03-04  1235000
9  1967-03-11  1234000
```

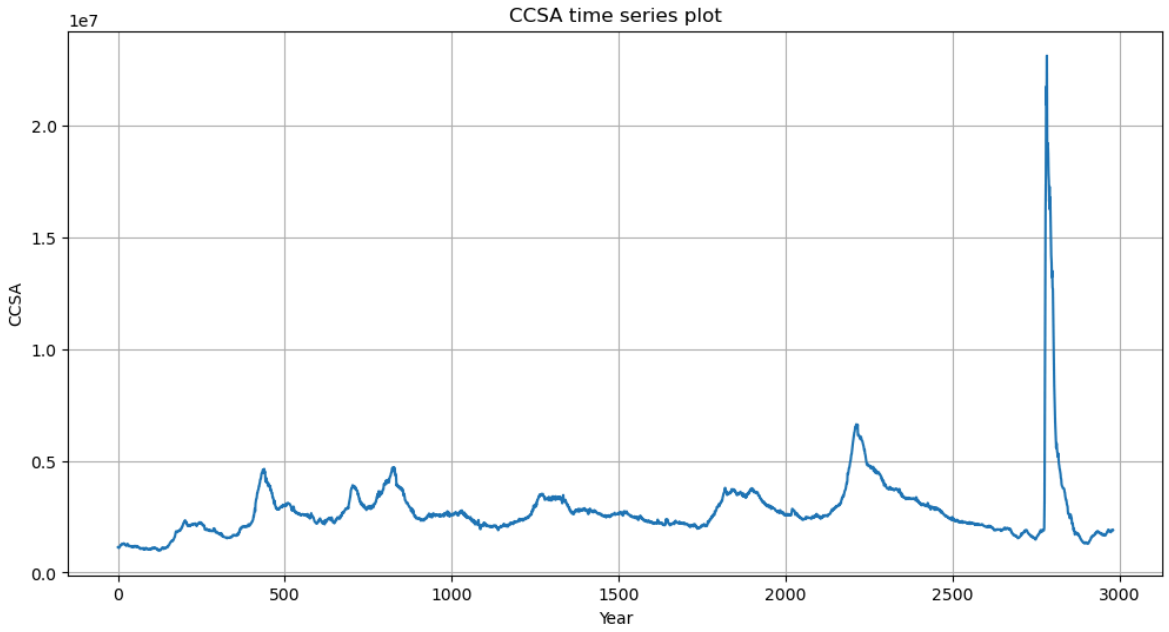
In [174]:
```
# Convert 'DATE' column to datetime
print('DATE data type before conversion:')
print(df1['DATE'].dtypes)
df1['DATE'] = pd.to_datetime(df1['DATE'])
print('DATE data type after conversion:')
print(df1['DATE'].dtypes)
```

```
DATE data type before conversion:
object
DATE data type after conversion:
datetime64[ns]
```

In [175]:
```
print(f"Statistical Summary:")
print(df1.describe())
```

```
Statistical Summary:
                                DATE          CCSA
count                           2982  2.982000e+03
mean   1995-08-01 11:59:59.999999872  2.762956e+06
min              1967-01-07 00:00:00  9.880000e+05
25%              1981-04-19 18:00:00  2.109000e+06
50%              1995-08-01 12:00:00  2.529500e+06
75%              2009-11-12 06:00:00  3.072000e+06
max              2024-02-24 00:00:00  2.313000e+07
std                              NaN  1.594004e+06
```

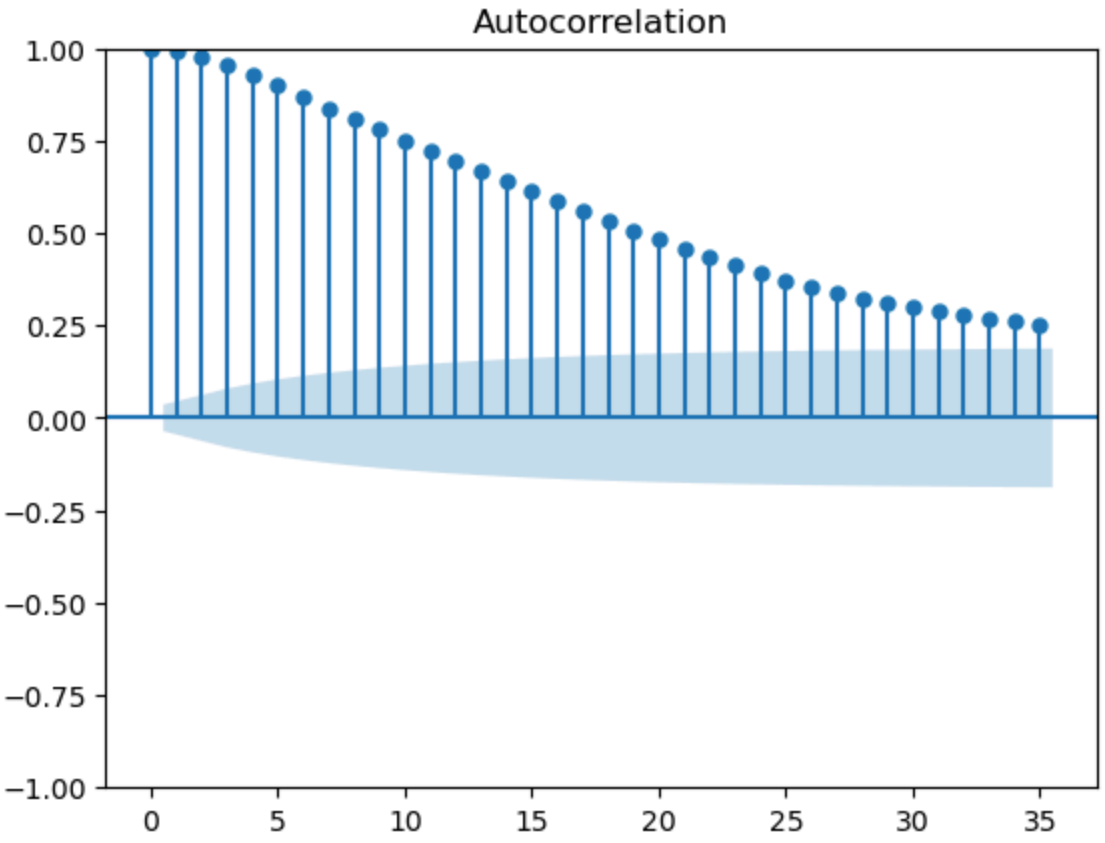
In [176]:
```
plt.figure(figsize=(12, 6))
plt.plot(df1.index, df1['CCSA'])
plt.title('CCSA time series plot')
plt.xlabel('Year')
plt.ylabel('CCSA')
plt.grid(True)
plt.show()
```

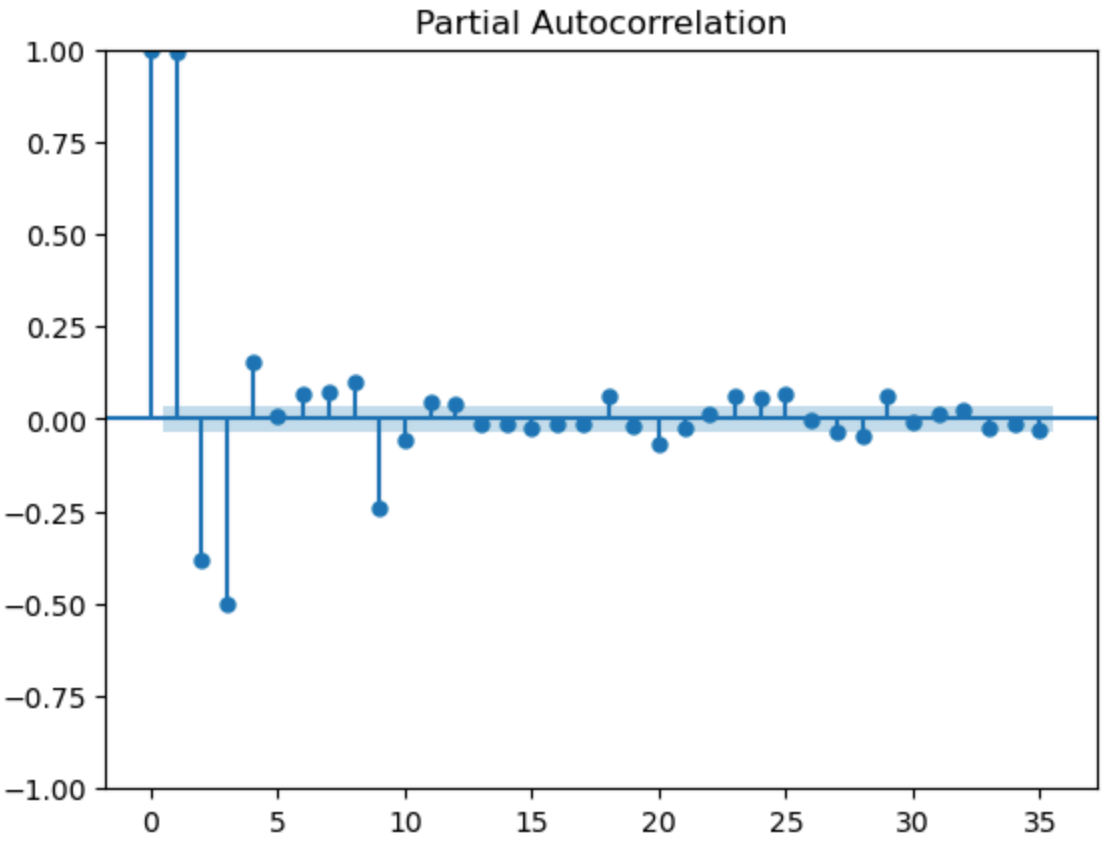


In [177]:
```
monthly_df = df.resample('M', on='DATE').mean().reset_index()
monthly_df['DATE'] = monthly_df['DATE'] + pd.offsets.MonthEnd(0)
df1['DATE'] = df1['DATE'] + pd.offsets.MonthEnd(0)
merged_df = pd.merge(monthly_df, df1, on='DATE', how='inner')
print(merged_df.head(10))
```

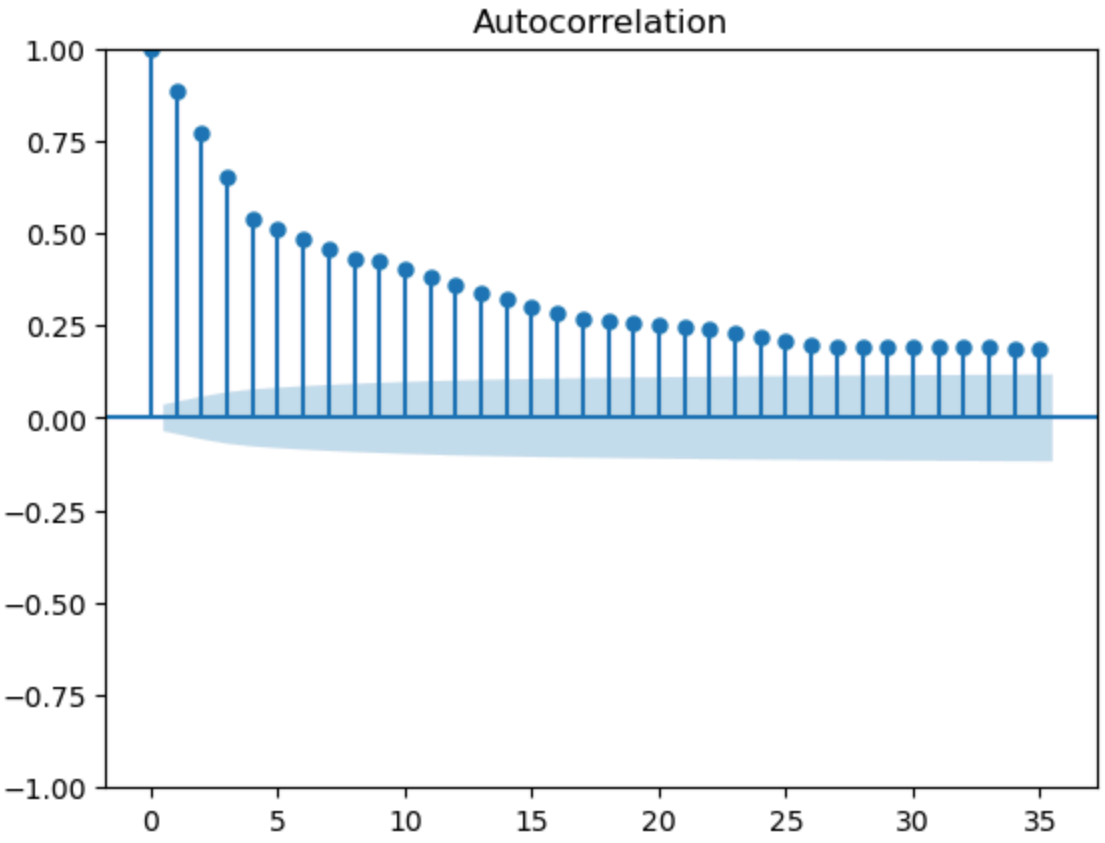
```
        DATE      ICSA   ICSA_imputed     CCSA
0 1967-01-31  209000.0  144938.820843  1134000
1 1967-01-31  209000.0  144938.820843  1119000
2 1967-01-31  209000.0  144938.820843  1119000
3 1967-01-31  209000.0  144938.820843  1103000
4 1967-02-28  229000.0  206952.631413  1131000
5 1967-02-28  229000.0  206952.631413  1153000
6 1967-02-28  229000.0  206952.631413  1167000
7 1967-02-28  229000.0  206952.631413  1199000
8 1967-03-31  260750.0  245153.974462  1235000
9 1967-03-31  260750.0  245153.974462  1234000
```

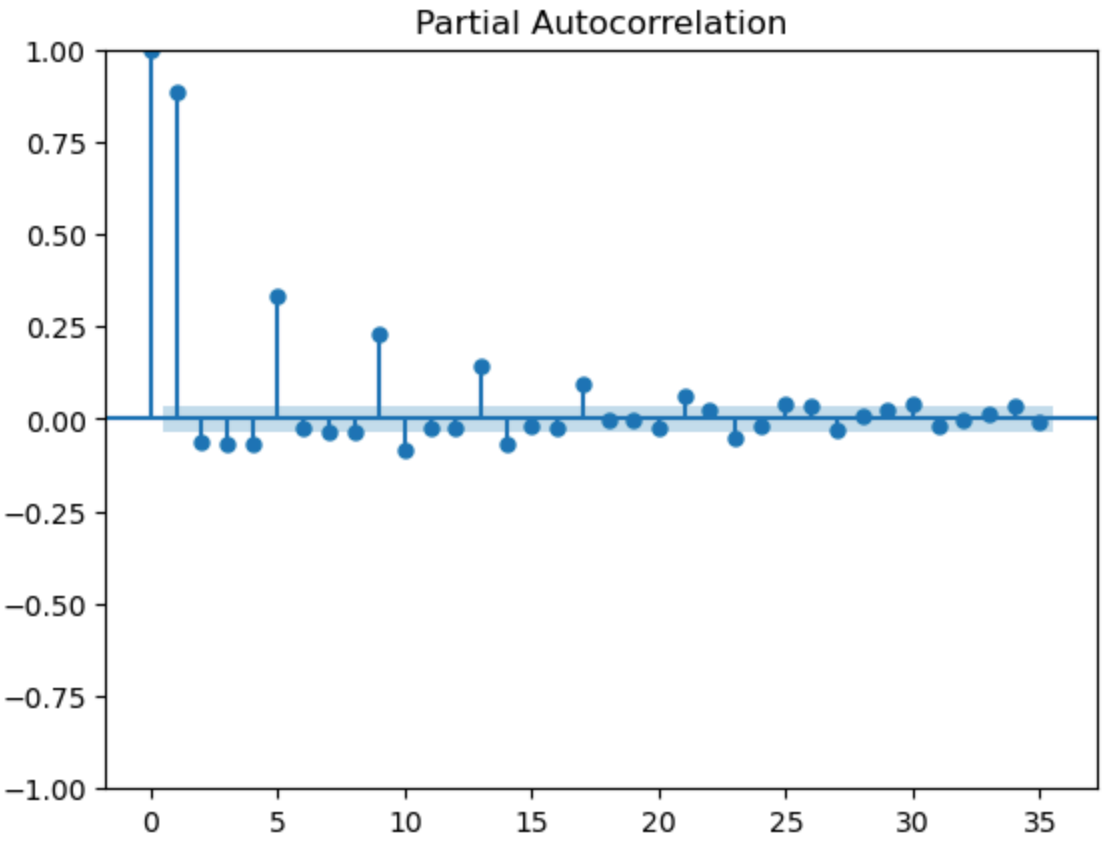
In [178]:
```
plot_acf(merged_df['CCSA'])
plot_pacf(merged_df['CCSA'])
plt.show()
```





In [179]:
```
plot_acf(merged_df['ICSA_imputed'])
plot_pacf(merged_df['ICSA_imputed'])
plt.show()
```





In [180]:
```
from sklearn.model_selection import train_test_split
X = merged_df[['CCSA']]
Y = merged_df['ICSA_imputed']
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, Y, test_size=0.2, random_s
```

In [181]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

model = SARIMAX(Ytrain, exog=Xtrain, order=(2, 0, 2))
result = model.fit()
print(result.summary())
```

```
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:473: ValueWarning: An unsupported index was provided and will be ignored w
hen e.g. forecasting.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:473: ValueWarning: An unsupported index was provided and will be ignored w
hen e.g. forecasting.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results
==========================================================================
==
Dep. Variable:           ICSA_imputed   No. Observations:                 23
85
Model:               SARIMAX(2, 0, 2)   Log Likelihood             -31832.8
83
Date:                Thu, 07 Mar 2024   AIC                        63677.7
66
Time:                        13:48:57   BIC                        63712.4
28
Sample:                             0   HQIC                       63690.3
80
                                - 2385
Covariance Type:                  opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
CCSA           0.1271      0.000    445.056      0.000       0.127       0.1
28
ar.L1         -0.3556      5.041     -0.071      0.944     -10.236       9.5
25
ar.L2         -0.1294      6.004     -0.022      0.983     -11.898      11.6
39
ma.L1          0.3643      5.039      0.072      0.942      -9.512      10.2
40
ma.L2          0.1367      6.004      0.023      0.982     -11.630      11.9
03
sigma2      2.301e+10   3.26e-07   7.06e+16      0.000     2.3e+10     2.3e+
10
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):          1433
1757.06
Prob(Q):                              0.80   Prob(JB):
0.00
Heteroskedasticity (H):               2.25   Skew:
14.81
Prob(H) (two-sided):                  0.00   Kurtosis:
381.61
==========================================================================
=======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (compl
ex-step).
[2] Covariance matrix is singular or near-singular, with condition number 9.
88e+30. Standard errors may be unstable.
```

In [182]:
```
f = result.get_forecast(steps=len(Ytest), exog =Xtest)
f_v = f.predicted_mean
c_i = f.conf_int()
print("Forecast values:")
print(f_v)
print("\nConfidence intervals:")
print(c_i)
```

```
Forecast values:
2385    144451.391298
2386    170138.683527
2387    318644.337043
2388    371079.906423
2389    233645.750670
            ...
2977    329253.476732
2978    237850.677593
2979    225011.063249
2980    214078.322323
2981    571426.400738
Name: predicted_mean, Length: 597, dtype: float64

Confidence intervals:
      lower ICSA_imputed  upper ICSA_imputed
2385       -152828.788700       441731.571295
2386       -127152.741248       467430.108301
2387         21350.314650       615938.359435
2388         73784.869959       668374.942887
2389        -63649.308160       530940.809500
...                   ...                 ...
2977         31958.409704       626548.543760
2978        -59444.389435       535145.744621
2979        -72284.003778       522306.130277
2980        -83216.744705       511373.389351
2981        274131.333710       868721.467765

[597 rows x 2 columns]

C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:836: ValueWarning: No supported index is available. Prediction results wil
l be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:836: FutureWarning: No supported index is available. In the next version,
calling this method in a model without a supported index will result in an e
xception.
  return get_prediction_index(
```

In [184]:
```
print("Forecasted value is:")
print(f_v.iloc[0])
```

```
Forecasted value is:
144451.39129792954
```

In [ ]: