In [154]:
```
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 [155]:
```
df.shape
```

Out[155]: (2981, 2)

In [156]:
```
# 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 [157]:
```
# 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}")
```

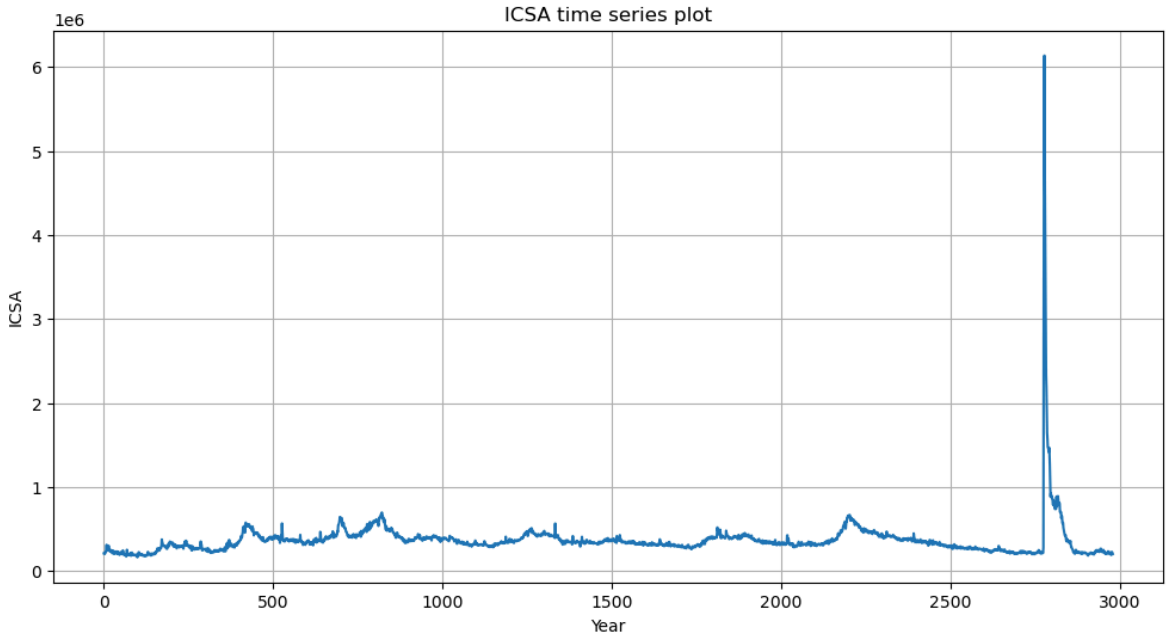
```
Missing values:
DATE    0
ICSA    0
dtype: int64
Duplicate values:
0
```

Here there are no missing or duplicate values

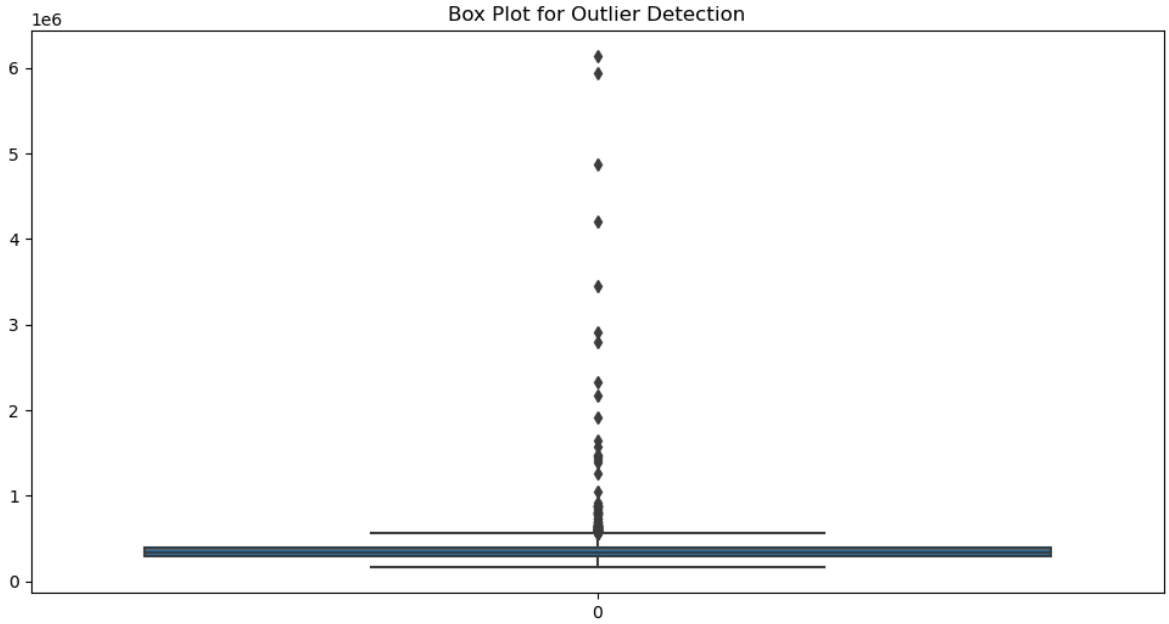
In [158]:
```
print(f"Statistical Summary:")
print(df.describe())
```

```
Statistical Summary:
                                DATE          ICSA
count                           2981  2.981000e+03
mean   1995-07-28 23:59:59.999999872  3.653677e+05
min              1967-01-07 00:00:00  1.620000e+05
25%              1981-04-18 00:00:00  2.910000e+05
50%              1995-07-29 00:00:00  3.420000e+05
75%              2009-11-07 00:00:00  3.990000e+05
max              2024-02-17 00:00:00  6.137000e+06
std                              NaN  2.418253e+05
```

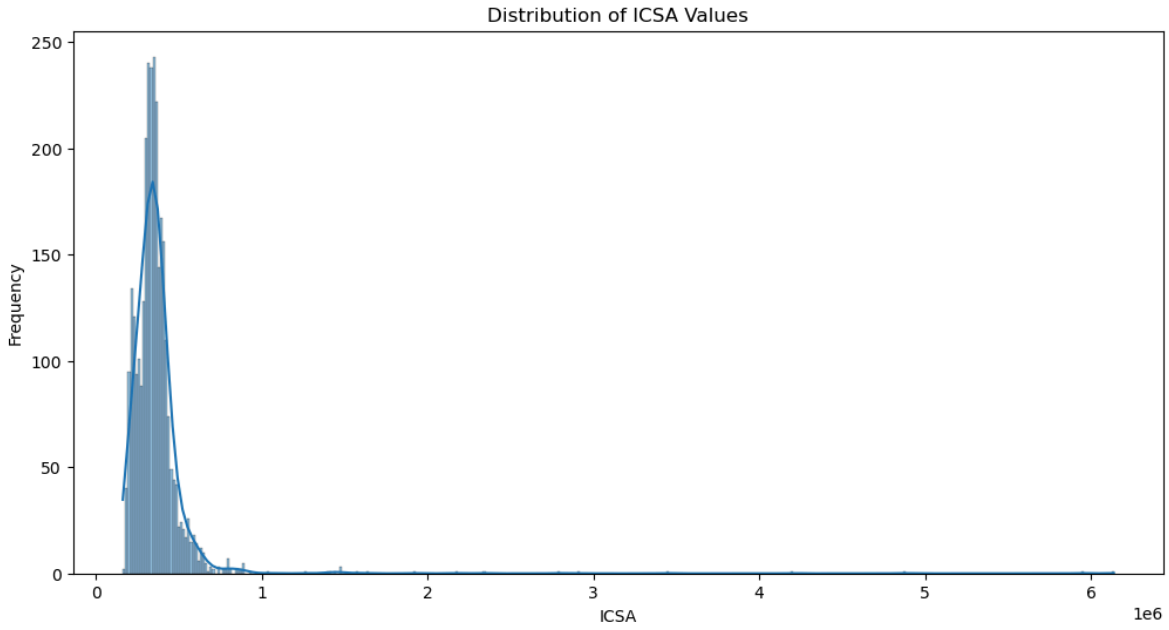
In [159]:
```
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['ICSA'])
plt.title('ICSA time series plot')
plt.xlabel('Year')
plt.ylabel('ICSA')
plt.grid(True)
plt.show()
```



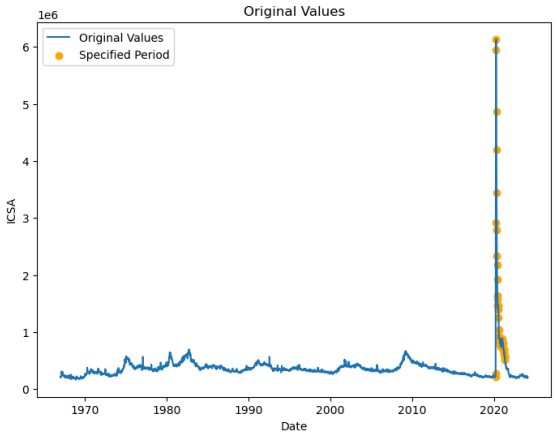
In [160]:
```
# Outlier Detection
plt.figure(figsize=(12, 6))
sns.boxplot(df['ICSA'])
plt.title('Box Plot for Outlier Detection')
plt.show()
```

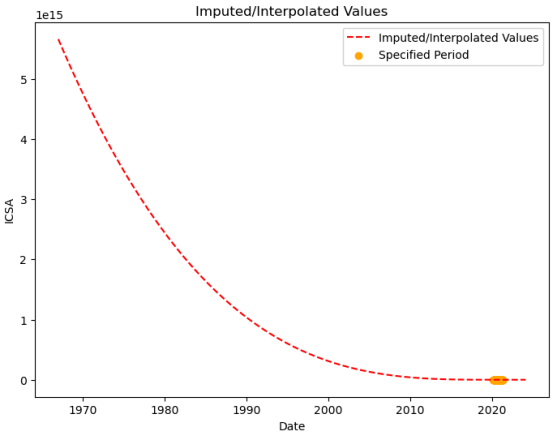


In [161]:
```
# Distribution of ICSA values
plt.figure(figsize=(12, 6))
sns.histplot(df['ICSA'], kde=True)
plt.title('Distribution of ICSA Values')
plt.xlabel('ICSA')
plt.ylabel('Frequency')
plt.show()
```



In [162]:
```
import pandas as pd
import numpy as np
from scipy.interpolate import CubicSpline
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)]


period_indices = df.index[(df['DATE'] >= start_date) & (df['DATE'] <= end_dat

cubic_spline = CubicSpline(period_indices, period_data['ICSA'])

imputation_indices = np.arange(df.index.min(), df.index.max() + 1)

imputed_values = cubic_spline(imputation_indices)

# Update the original dataframe with imputed or interpolated values
df.loc[imputation_indices, 'ICSA_imputed'] = imputed_values


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


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()
```


Original Values
1e6
Original Values
Specified Period
ICSA
Date
0
1
2
3
4
5
6
1970
1980
1990
2000
2010
2020


Imputed/Interpolated Values
1e15
Imputed/Interpolated Values
Specified Period
ICSA
Date
0
1
2
3
4
5
1970
1980
1990
2000
2010
2020

In [167]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing

df.index = pd.to_datetime(df.index)
size = int(len(df) * 0.8)
Xtrain, Xtest = df.iloc[:size]['ICSA'], df.iloc[size:]['ICSA']

model_multi = ExponentialSmoothing(train, trend='mul', seasonal='mul', seasona
forecast_multi = model_multi.forecast(steps=len(Xtest))
model_add = ExponentialSmoothing(train, trend='add', seasonal='add', seasonal_
forecast_add = model_add.forecast(steps=len(Xtest))
print(forecast_multi)
print(forecast_add)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:473: ValueWarning: No frequency information was provided, so inferred freq
uency N will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\holtwinters\mode
l.py:83: RuntimeWarning: overflow encountered in matmul
  return err.T @ err
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\holtwinters\mode
l.py:917: ConvergenceWarning: Optimization failed to converge. Check mle_ret
vals.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:473: ValueWarning: No frequency information was provided, so inferred freq
uency N will be used.
  self._init_dates(dates, freq)
```

```
1970-01-01 00:00:00.000002384    381707.454397
1970-01-01 00:00:00.000002385    380659.434903
1970-01-01 00:00:00.000002386    379030.605881
1970-01-01 00:00:00.000002387    379328.190128
1970-01-01 00:00:00.000002388    382446.048836
                                      ...
1970-01-01 00:00:00.000002976    338593.104774
1970-01-01 00:00:00.000002977    342003.420992
1970-01-01 00:00:00.000002978    342163.948836
1970-01-01 00:00:00.000002979    345011.154928
1970-01-01 00:00:00.000002980    343007.083610
Freq: N, Length: 597, dtype: float64
1970-01-01 00:00:00.000002384    384018.016148
1970-01-01 00:00:00.000002385    383453.564607
1970-01-01 00:00:00.000002386    382414.404511
1970-01-01 00:00:00.000002387    382362.326657
1970-01-01 00:00:00.000002388    386211.686758
                                      ...
1970-01-01 00:00:00.000002976    303835.104726
1970-01-01 00:00:00.000002977    306358.297952
1970-01-01 00:00:00.000002978    305296.519269
1970-01-01 00:00:00.000002979    305725.788561
1970-01-01 00:00:00.000002980    302394.957176
Freq: N, Length: 597, dtype: float64
```

```
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\tsa\holtwinters\mode
l.py:917: ConvergenceWarning: Optimization failed to converge. Check mle_ret
vals.
  warnings.warn(
```

In [169]:
```
print(forecast_add.iloc[-1])
```

```
302394.9571756705
```

In [168]:
```
print(forecast_multi.iloc[-1])
```

```
343007.0836100745
```

Hence the multiplicative Holt-Winters Last Forecast Value is 343007.08 and additive Holt-Winters Last Forecast Value is 302394.95