1.TIME SERIES DATA CLEANING,LOADING AND HANDLING TIMES SERIES DATA AND PRE -PROCESSING

AIM:

To implement programs for time series data cleaning, loading and handling times series data and pre- processing techniques.

PROCEDURE AND CODE:

1. **Load the Dataset**: Import the data into a Pandas DataFrame.
2. **Inspect the Dataset**: Check for missing values, data types, and basic statistics.
3. **Handle Missing Data**: Fill or interpolate missing values.
4. **Convert to Time Series**: Ensure the data is in a proper time-series format.
5. **Resample and Aggregate**: If necessary, resample to a desired frequency.
6. **Normalize/Scale Data**: Standardize or normalize the data for modeling.
7. **Visualize Trends**: Plot the data to inspect trends and seasonality.

CODE:

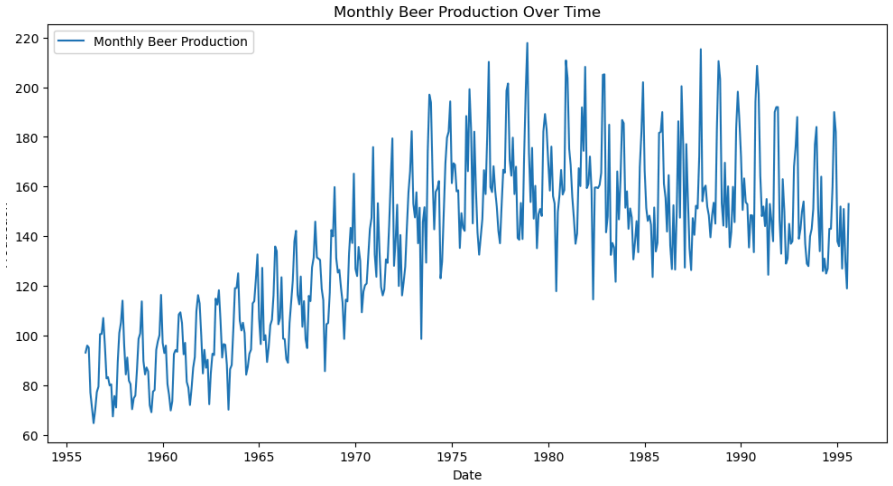
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Load dataset
file_path = r"c:\Users\HDC0422279\Downloads\monthly-beer-production.csv"  # Replace with your file path
data = pd.read_csv(file_path)

# Display the first few rows
print(data.head())
# Check for missing values and data types
print(data.info())
print(data.describe())

# Check for duplicates
print(f"Number of duplicate rows: {data.duplicated().sum()}")
# Handle missing values (e.g., interpolate or forward-fill)
```

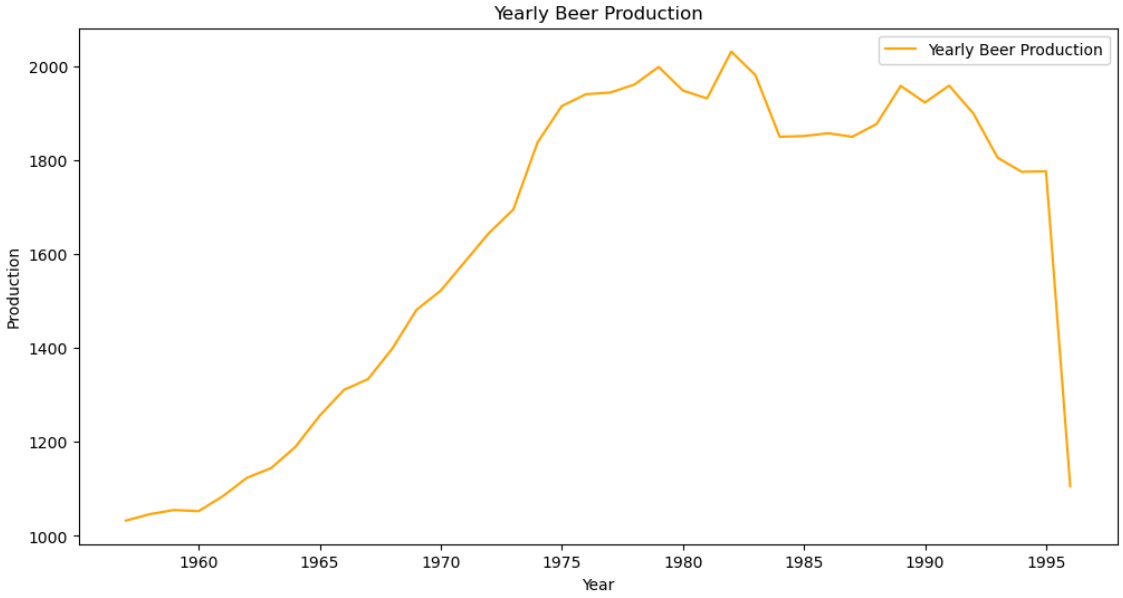
```
# Assuming your production column is named 'Monthly beer production'
# Replace 'Monthly beer production' with the actual name if different
data['Monthly beer production'] = data['Monthly beer production'].interpolate(method='linear')

# Verify no missing values remain
print(data.isnull().sum())
# Convert the 'Month' column to datetime
data['Month'] = pd.to_datetime(data['Month'])

# Set 'Month' as the index
data.set_index('Month', inplace=True)

# Sort index to ensure proper time series order
data = data.sort_index()

# Check the dataset
print(data.head())
# Plot the time series
plt.figure(figsize=(12, 6))
# Replace 'Monthly beer production' with actual column name
plt.plot(data['Monthly beer production'], label='Monthly Beer Production')
plt.title('Monthly Beer Production Over Time')
plt.xlabel('Date')
plt.ylabel('Production')
plt.legend()
plt.show()
# Resample to yearly frequency (if needed)
# Replace 'Monthly beer production' with actual column name
yearly_data = data['Monthly beer production'].resample('Y').sum()

# Plot yearly data
plt.figure(figsize=(12, 6))
plt.plot(yearly_data, label='Yearly Beer Production', color='orange')
plt.title('Yearly Beer Production')
plt.xlabel('Year')
plt.ylabel('Production')
plt.legend()
```

```
plt.show()
# Normalize the production data (Min-Max Scaling)
# Replace 'Monthly beer production' with actual column name
data['Normalized_Production'] = (data['Monthly beer production'] - data['Monthly beer
production'].min()) / \
                                (data['Monthly beer production'].max() - data['Monthly beer
production'].min())

# Check normalized data
print(data.head())
# Rolling statistics for outlier detection
# Replace 'Monthly beer production' with actual column name
data['Rolling_Mean'] = data['Monthly beer production'].rolling(window=12).mean()

# Plot rolling mean
plt.figure(figsize=(12, 6))
# Replace 'Monthly beer production' with actual column name
plt.plot(data['Monthly beer production'], label='Original')
plt.plot(data['Rolling_Mean'], label='Rolling Mean', color='red')
plt.title('Rolling Mean - Outlier Detection')
plt.legend()
plt.show()
```

OUTPUT:

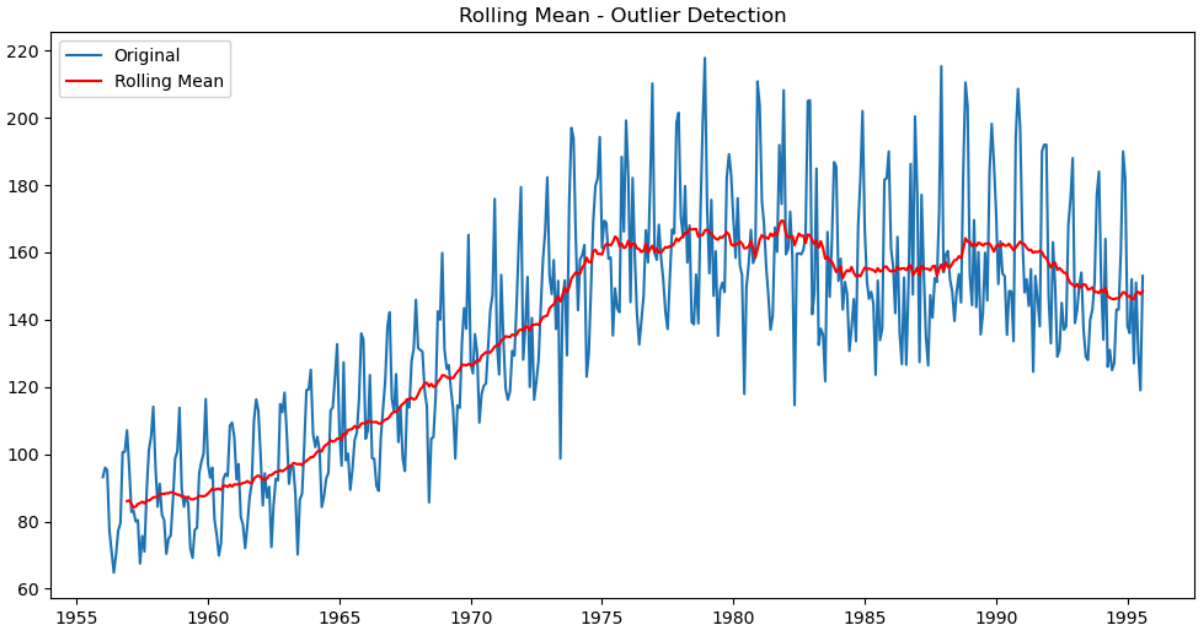
```
     Month  Monthly beer production
0  1956-01                     93.2
1  1956-02                     96.0
2  1956-03                     95.2
3  1956-04                     77.1
4  1956-05                     70.9
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 476 entries, 0 to 475
Data columns (total 2 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Month                   476 non-null    object
 1   Monthly beer production  476 non-null    float64
dtypes: float64(1), object(1)
memory usage: 7.6+ KB
None
       Monthly beer production
count               476.000000
mean                136.395378
std                  33.738725
min                  64.800000
25%                 112.900000
50%                 139.150000
75%                 158.825000
max                 217.800000
Number of duplicate rows: 0
Month                       0
Monthly beer production     0
dtype: int64
            Monthly beer production
Month
1956-01-01                     93.2
1956-02-01                     96.0
1956-03-01                     95.2
1956-04-01                     77.1
1956-05-01                     70.9
```





Month	Monthly beer production	Normalized_Production
1956-01-01	93.2	0.185621
1956-02-01	96.0	0.203922
1956-03-01	95.2	0.198693
1956-04-01	77.1	0.080392
1956-05-01	70.9	0.039869

\



RESULT:

The above program has been successfully written and executed.