

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

EX.N0 : 1	Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques.
<u>DATE : 25/01/2025</u>	

AIM:

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

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
import datetime
```

```
# Function to load dataset def
```

```
load_dataset(file_path):
    """
```

```
    Load a time series disaster dataset.
```

```
    Args:
```

```
        file_path (str): Path to the dataset file.
```

```
    Returns:
```

```
        DataFrame: Loaded dataset.
```

```
    """
```

```
    try:
```

```
        data = pd.read_csv(file_path)
```

```
        print("Dataset loaded successfully.")
```

```
        return data    except Exception as e:
```

```
            print(f"Error loading dataset: {e}")
```

```
        return None
```

```
# Function to clean dataset def
```

```
clean_dataset(data):
```

```
    """
```

```
    Clean the disaster dataset by handling missing values and duplicates.
```

Args:

data (DataFrame): Input dataset.

Returns:

DataFrame: Cleaned dataset.

```
"""    print("Cleaning
dataset...")
# Combine 'year', 'month', and 'day' into a single 'date' column    data['month'] =
data['month'].apply(lambda x: datetime.strptime(x, '%B').month if isinstance(x, str)
else x)    data['date'] = pd.to_datetime(data[['year', 'month', 'day']])
data.drop(columns=['year', 'month', 'day'], inplace=True)

# Handle missing values
data['area'].fillna('Unknown', inplace=True)
data['region'].fillna('Unknown', inplace=True)
data['deaths'].fillna(data['deaths'].median(), inplace=True)

# Remove duplicates    data
= data.drop_duplicates()

# Sort by date
data.sort_values(by='date', inplace=True)

    print("Dataset cleaned.")
return data
```

Function to preprocess time series data def

preprocess_timeseries(data):

```
"""
```

Preprocess the disaster dataset by resampling and feature extraction.

Args:

data (DataFrame): Input dataset.

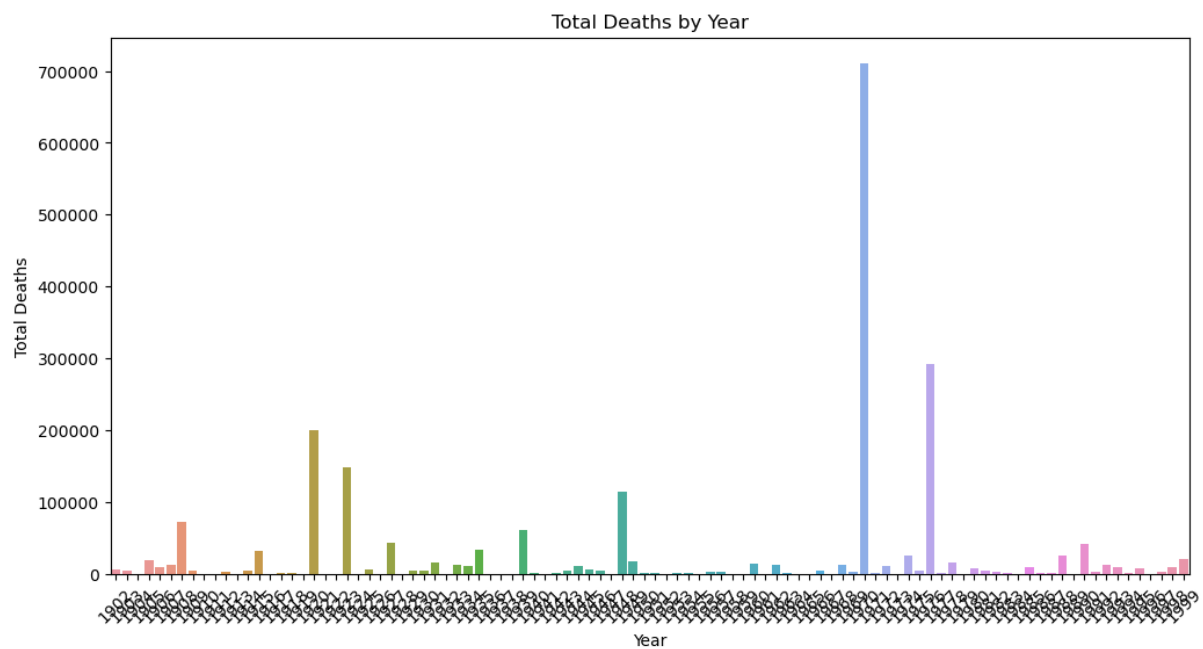
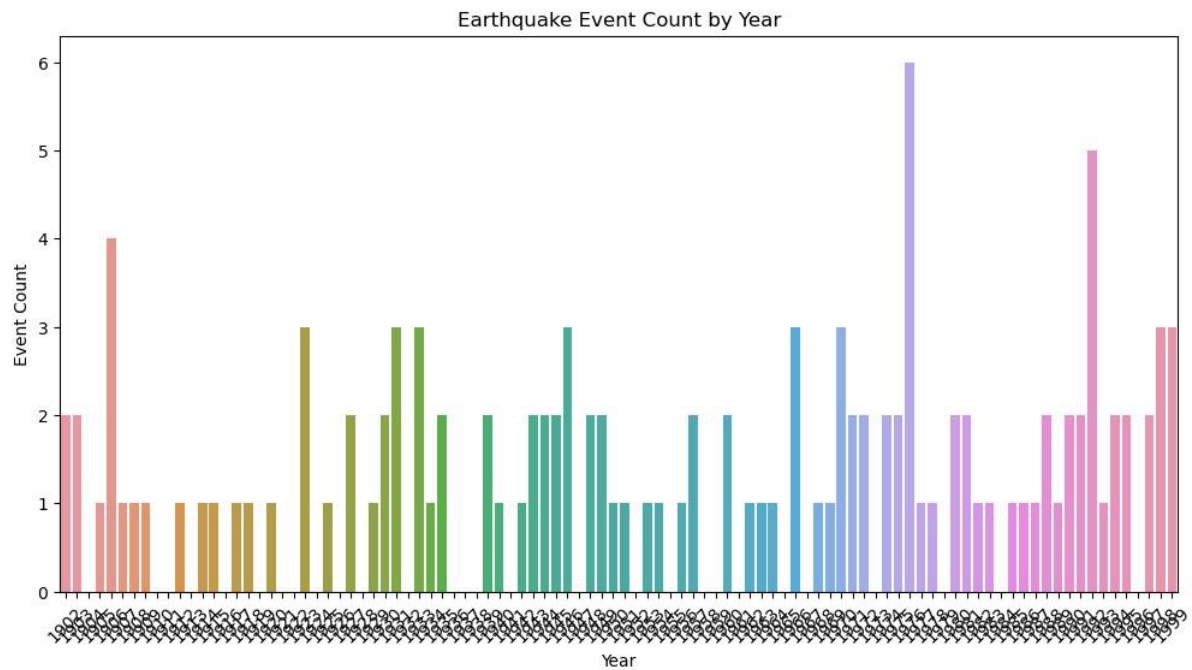
Returns:

DataFrame: Preprocessed dataset.

```
"""    print("Preprocessing
dataset...")

# Resample to yearly data    data_resampled =
data.set_index('date').resample('Y').agg({
    'richter': 'mean',    # Average magnitude per year
    'deaths': 'sum',    # Total deaths per year
    'area': 'count',    # Count of events per year (proxy for frequency)
}).rename(columns={'area': 'event_count'})
```

OUTPUT:



```
Clustering completed.
      richter  deaths  event_count  cluster
date
1902-12-31    6.950   6700.0         2       0
1903-12-31    6.400   4500.0         2       0
1905-12-31    7.500  19000.0         1       0
1906-12-31    7.975   9132.0         4       0
1907-12-31    8.100  12000.0         1       0
Clustered dataset saved to 'clustered_earthquake_data.csv'.
```

RESULT:

Thus, the program for Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques is executed successfully.