K.T.S.P. MANDAL'S

K.M.C. COLLEGE, KHOPOLI

DEPARTMENT OF COMPUTER SCIENCE

KHOPOLI – 410203

A

PROJECT REPORT

ON

"***CYCLONE AND EARTHQUAKE ANALYSIS***"

UNDER THE GUIDANCE OF

Mrs.Tanuja Patil

SUBMITTED TO

UNIVERSITY OF MUMBAI

BY

Mrs. Neha Narendra Ghonge

Msc(cs)-pt-2

(COMPUTER SCIENCE) 2020-2021

ACKNOWLEDGEMENT

It gives me great pleasure to present my project on, "***Cyclone And Earthquake Analysis***".

This is my first milestone in MSC. Computer Science. I would like to thank our ***Prof.Mrs.Dhanashree Pawar*** (HOD of ComputerScience),who helped throughout the project.

I would like to express my sincere gratitude to all the professors who helped me in project. I would also like to acknowledge the help and guidance of ***Prof.Miss.Tanuja Patil*** for acknowledging the help & guidance provided by them for project in all the places during the presentation of the project .I would also extend my to our principle ***Dr.Mr.Pratap Patil*** Sir for his support &facilities provided to us for the same.

Onward my project work ,I am also grateful to the staff member of computer department for their moral support & application shown towards my project.

Yours Sincerely,

Mrs. Neha Narendra Ghonge

MSC (COMPUTER SCIENCE)

INDEX

INTRODUCTION

Cyclone and earthquake analysis play a crucial role in understanding and assessing the impact of these natural disasters on human lives, infrastructure, and the environment. Cyclones and earthquakes are two distinct but significant geophysical hazards that pose a considerable threat to countries like India. Therefore, studying and analyzing the patterns, characteristics, and impacts of cyclones and earthquakes is essential for disaster preparedness, mitigation, and response efforts.

- **Background**

  Cyclone Disaster Management encompasses mitigation and preparedness measures for cyclones. India has a long history with cyclones. The location of India in the north Indian Ocean makes it vulnerable to the tropical cyclone. In 2019-20, India witnessed multiple cyclones including Amphan, Nisarga, Nivar, etc. Hence, it is important for the IAS Exam aspirants to look into the issue from a holistic perspective.

  On the other hand, earthquakes result from the sudden release of energy in the Earth's crust, leading to ground shaking. India lies in a seismically active zone due to the collision of the Indian and Eurasian tectonic plates. The Himalayan region and the north eastern states are particularly prone to earthquakes, with varying magnitudes and frequencies. Earthquakes can cause immense destruction, including building collapses, landslides, and loss of lives.

➢ Aim

The aim of cyclone and earthquake analysis is to investigate and understand the characteristics, occurrences, and impacts of these natural hazards in India. This analysis involves studying historical data, monitoring current events, and applying statistical and geospatial techniques to identify patterns, trends, and potential risk areas.

By conducting thorough analysis, researchers, policymakers, and disaster management authorities can achieve several objectives:

1.Risk assessment
2.Early warning systems
3.Disaster preparedness and mitigation
4. Post-disaster assessment and recovery

Overall, cyclone and earthquake analysis aims to provide valuable insights into these natural hazards, enabling informed decision-making, risk reduction, and effective disaster management strategies. By understanding the patterns and impacts of these events, India can enhance its resilience and mitigate the adverse effects of cyclones and earthquakes on its population and infrastructure.

SOFTWARE REQUIREMENTS

- **Operating System: -** Windows 10

- **Hardware Requirements**

- *Processor* :- AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz.
- *Ram* :- 8 Giga Bytes.
- *Hard Disk* :- 40 Giga Bytes.
- *Type Of System* :-Single User.
- *System Type*:-64-bit operating system, x64-based processor

- **Software Requirements**

- Jupyter Notebook (Installed via command prompt)

PRELIMINARY INVESTIGATION

ABSTRACT

Cyclone and earthquake analysis plays a crucial role in understanding the patterns, impacts, and risks associated with these natural disasters. This study aims to investigate the characteristics and occurrences of cyclones and earthquakes in India. By analyzing historical data, utilizing statistical and geospatial techniques, and exploring the relationships between variables, valuable insights are gained for disaster preparedness, mitigation, and response efforts.

The findings contribute to risk assessment, early warning systems, and informed decision-making in terms of infrastructure resilience and community preparedness.

Ultimately, this analysis enhances the understanding of cyclone and earthquake dynamics, enabling effective strategies to minimize their impact on human lives and infrastructure.

- **Analysis Methodology :**

A. *Data Collection And Visualization*:

Historical data on cyclones and earthquakes, including their locations, dates, magnitudes, and other relevant parameters, is collected from reliable sources such as meteorological departments, geological surveys, or disaster management agencies .And stored in CSV which can be read by using pandas.

Data visualization is a powerful tool for representing data in a visual format, enabling users to gain insights and understand patterns or relationships within the data. Here are some common techniques used in data visualization:

Line Charts: Line charts are effective for visualizing trends over time. In earthquake analysis, they can be used to show the frequency or magnitude of earthquakes over a specific period.

***Scatter Plots*:** Scatter plots are useful for displaying the relationship between two variables. In earthquake analysis, they can illustrate the relationship between earthquake magnitude and depth, or the location of earthquakes on a map.

Heatmaps: Heatmaps represent data using color gradients, providing a visual depiction of intensity or density. In earthquake analysis, a heatmap can show the spatial distribution of earthquake occurrences or the intensity of seismic activity in different regions.

Geographic Maps: Geographic maps are particularly relevant for earthquake analysis, as they allow the visualization of earthquake locations on a map. Different symbols or colors can represent earthquake magnitudes or depths, providing insights into spatial patterns.

Histograms: Histograms are useful for illustrating the frequency distribution of a dataset. In earthquake analysis, they can show the distribution of earthquake magnitudes or depths.

B. *Data Pre-processing*:
The collected data is cleaned, validated, and preprocessed to ensure its quality and consistency. This may involve removing duplicates, handling missing values, and standardizing the data format.

C. *Exploratory Data Analysis*: Statistical techniques and visualization methods are applied to explore and understand the data. This step helps identify patterns, trends, and potential relationships between different variables.

To begin this exploratory analysis, first import libraries and define functions for plotting the data using matplotlib. Depending on the data, not all plots will be made.

D. Time Series Analysis:
It focuses on studying the patterns, trends, and dependencies within a time-ordered sequence of observations. Time series analysis can be applied to various domains, including finance, economics, weather forecasting, stock market analysis, and more.

***Key steps involved in time series analysis*:**

1. *Data Collection*
2. *Data Preprocessing*

3. *Time Series Visualization*
4. *Stationarity Analysis*
5. *Trend and Seasonality Analysis*
6. *Model Selection*
7. *Model Fitting*
8. *Model Evaluation.*
9. *Forecasting*
10. *Model Validation.*
11. *Model Refinement*

E. **Spatial Analysis**: Geospatial analysis tools and techniques are employed to study the spatial distribution of cyclones and earthquakes. This may include mapping the occurrences, identifying high-risk zones, and analyzing proximity to vulnerable areas such as coastal regions or fault lines.

F. **Machine Learning Algorithms**: Machine learning algorithms can be applied to develop predictive models for cyclones and earthquakes. These algorithms can learn patterns from historical data and make predictions about future events. Techniques like classification, clustering, or regression algorithms may be employed for this purpose.

G. **Validation and Evaluation**: The developed models or analysis results are validated and evaluated using appropriate metrics and validation techniques. This ensures the reliability and accuracy of the predictions or findings.

H. **Interpretation and Conclusion:** The results of the analysis are interpreted in the context of cyclone and earthquake dynamics, risk assessment, or disaster management. The conclusions drawn from the analysis help in informing decision-making processes and developing strategies for disaster preparedness, mitigation, and response.

➢ Key Findings

Summarize the main findings of the analysis. This could include identifying high-risk areas for cyclones or earthquakes, uncovering temporal or spatial patterns, evaluating prediction accuracy, or understanding the triggering mechanisms of seismic activity.

Feasibility Study

- **Technical feasibility**:-

Technical feasibility raises the questions like

a)Is it possible that the work can be done with current equipments, software technology and person?

b)Is new technology required, what is the possibilities that it can be developed?

In case of our project, the Analysis which we have built up fully support current windows OS but it lacks the support of other environment OS. It is not depended on the large number of user. So, it can handle a very large number of user's environment.The support for the hardware:

It has full support for new hardware. So no hardware compatibility issues arise as it requires minimum configuration.

- **Economic feasibility**:-

It deals with economical impact of the system on the environment it is used i.e. benefit in creating the systems. And the project is economical feasible.

ANALYSIS AND DESIGN

➢ *Data Mining*

In the data mining stage of the project, various tasks were performed.Using the ‘normalised’ data set, the creation of a correlation matrix (Pearson parametric correlation test) was completed. The matrix will be used to explore the relationship/dependency between the variables. The results section below contains the output table of correlation coefficients

➢ *Dimensionality reduction*

Dimensionality reduction is the process of reducing the number of features (or dimensions) in a dataset while retaining as much information as possible. This can be done for a variety of reasons, such as to reduce the complexity of a model, to improve the performance of a learning algorithm, or to make it easier to visualize the data. There are several techniques for dimensionality reduction, including principal component analysis (PCA), singular value decomposition (SVD), and linear discriminant analysis (LDA). Each technique uses a different method to project the data onto a lower-dimensional space while preserving important information.

Dimensionality reduction is a technique used to reduce the number of features in a dataset while retaining as much of the important information as possible. In other words, it is a process of transforming high-dimensional data into a lower-dimensional space that still preserves the essence of the original data.

There are two main approaches to dimensionality reduction: feature selection and feature extraction.

Feature Selection:
Feature selection involves selecting a subset of the original features that are most relevant to the problem at hand. The goal is to reduce the dimensionality of the dataset while retaining the most important features. There are several methods for feature selection, including filter methods, wrapper methods, and embedded methods. Filter methods rank the features based on their relevance to the target variable, wrapper methods use the model performance as the criteria for selecting features, and embedded methods combine feature selection with the model training process.

Feature Extraction:
Feature extraction involves creating new features by combining or transforming the original features. The goal is to create a set of features that captures the essence of the original data in a lower-dimensional space. There are several methods for feature

extraction, including principal component analysis (PCA), linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE). PCA is a popular technique that projects the original features onto a lower-dimensional space while preserving as much of the variance as possible.

➢ *Clustering*

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.
We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm; an unsupervised learning algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

➢ *Model Evaluation*

Once you have crafted your model you need to evaluate the model thoroughly. In this stage you have to determine if your model is working properly, did you get the desired outcome also if it meets the business requirements. Always ensure that data is properly handled and interpreted. There are two methods of evaluating models in data analysis, Hold Out and Cross-Validation. They help to find the best model.
Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance.

The main purpose of cross validation is to prevent overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data.

➢ *Deployment and Visualization*

This is the final and the most crucial step of completing your data analytics project. After setting a model that performs well you can deploy the model for different applications and in the business market. This phase examines how well the model can withstand in the external environment. To explain your findings to the client you can use different interactive visualization tools. *Data Visualization i*s a graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide a quick and effective way to communicate and illustrate your conclusions.

DataSets on which the analysis is performed:-

- 4.5_month.csv ➔ CSV file contains data of all the earthquakes with a magnitude of 4.5 or higher.
- 1.0_month.csv ➔CSV file containing data of earthquakes of magnitude 1.0+.
- cyclones - Sheet1.csv ➔ It contains information about cyclones, including their names, lowest pressure (in millibars), and the corresponding years they occurred.Here are some key details from the dataset:
  Column Names:
  - Name: Name of the cyclone.
  - Lowest Pressure (mbar): The lowest recorded pressure of the cyclone.
  - Year: The year in which the cyclone occurred.

  Cyclones:

The dataset includes information about several cyclones, along with their lowest pressure and year of occurrence. Some of the cyclones mentioned are:

- BOB 03 (occurred in multiple years)
- BOB 05
- 03B
- Yemyin ,etc .

- Earthquakes - Sheet1.csv ➔It contains information about earthquakes, including the date, time, location, latitude, longitude, deaths, comments, and magnitude (M).Here are some key details from the dataset:
  Column Names:
  - Date: The date of the earthquake.
  - Time: The time of the earthquake.
  - Location: The location where the earthquake occurred.
  - Lat: The latitude of the earthquake's epicenter.

- Long: The longitude of the earthquake's epicenter.
- Deaths: The number of deaths caused by the earthquake.
- Comments: Additional comments or information about the earthquake.
- M: The magnitude of the earthquake.

Earthquakes:

- The dataset includes information about several earthquakes, including their dates, times, locations, latitude, longitude, deaths, comments, and magnitudes.

- Earthquake.csv ➔ file contains earthquake data with 22 columns. Here is a breakdown of the columns:

1. time: Date and Time of Earthquake Occurrence
2. latitude: Latitude of the earthquake location
3. longitude: Longitude of the earthquake location
4. depth: Depth of the earthquake's center
5. mag: Magnitude of the earthquake
6. magType: Type of magnitude measurement
7. nst: Number of seismic stations used for calculating the magnitude
8. gap: Gap between stations (in degrees) for magnitude calculation
9. dmin: Minimum distance to the earthquake (in degrees)
10. rms: Root mean square of the earthquake's residual

- Indian_earthquake_data.csv ➔This dataset includes a record of the date, time, location, depth, and magnitude of every earthquake since 1st August 2019. The magnitude refers to the amplitude or size of the seismic waves generated by an earthquake source
- Natural_Disasters_in_India .csv ➔ The dataset has been acquired from Wikipedia. The text is extracted from the Wikipedia articles and then the text is cleaned, processed, and sorted according to the date.

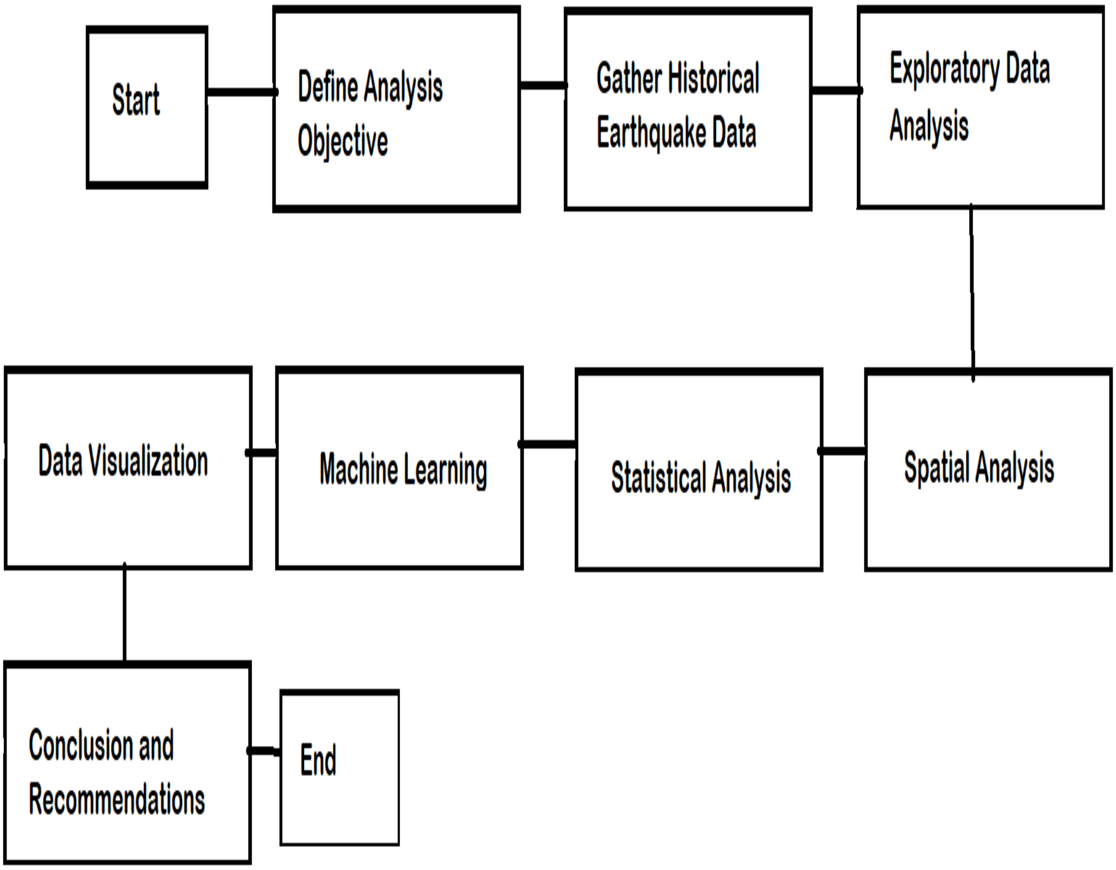
The dataset contains the following columns:

- Duration: includes day and month as we as intervals for some disasters that lasted more than a day
- Year: year of the disaster
- Disaster_Info: Information about the disaster (contains the long and short text describing the disaster)
- Date: Date in the specific format (for some disasters that lasted more than a day we have added the first day of disaster)

More specific information can be extracted from the text using natural language processing techniques.

- Significant_Earthquakes.csv ➔This dataset provides comprehensive information on significant earthquakes that have occurred around the world since 1900 with a magnitude of 5 or above. The data includes essential details such as location, date and time, magnitude, depth, and other relevant information about each earthquake.The dataset is updated weekly and sourced from the United States Geological Survey (USGS), which maintains a global catalog of earthquake information. The dataset includes earthquakes from all regions of the world, from the most seismically active regions like the Pacific Ring of Fire to less active regions like Europe and Africa.
  Earthquakes are natural disasters that can cause severe damage to property, loss of life, and environmental damage. The dataset can be used for various research purposes, including studying earthquake patterns and trends over time, examining the impact of earthquakes on human populations and infrastructure, and developing models to predict future earthquake activity.
  Researchers can use the dataset to explore the characteristics of earthquakes such as their frequency, magnitude, and location. By analyzing this data, researchers can identify earthquake patterns and trends and use the information to develop better models to predict future earthquakes. This dataset is a valuable resource for researchers and scientists who study earthquakes and their effects on the environment and human life.

FLOW CHART


Start
Define Analysis Objective
Gather Historical Earthquake Data
Exploratory Data Analysis
Spatial Analysis
Statistical Analysis
Machine Learning
Data Visualization
Conclusion and Recommendations
End

IMPLEMENTATION

The following analysis is performed on jupyter notebook And the language used is python .

JUPYTER NOTEBOOK

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Jupyter Notebook is probably the first Python IDE we used for data science. Its simplicity makes it great for beginners

PYTHON

Python has been around since 1991. It is one of the best programming languages widely used in data analytics. It is easy to use, fast, and manipulates data seamlessly. It supports various data analytics activities such as data collection, analysis, modelling, and visualisation.

MODULES/LIBRARIES

The Modules and Libraries used in this project are as

follows :

1.Matplotlib:-

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.

- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.

Install :- pip install matplotlib.

2.skitlearn :- (sklearn)

scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

Install:- pip install -U scikit-learn

3.pandas :-

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.

Install:- pip install pandas

4.Numpy:-

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Install:- pip install numpy

5.Plotly:-

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

Install:- pip install plotly

6. Seaborn: -

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Install :- pip install seaborn

RESULTS

1. CorrelationMatrix :
   The correlation matrix is a table that displays the correlation coefficients between variables in a DataFrame. The correlation coefficient measures the strength and direction of the linear relationship between two variables. It can range from -1 to 1, where -1 indicates a strong negative correlation, 0 indicates no correlation, and 1 indicates a strong positive correlation.
2. Scatter matrix:

The scatter matrix, also known as a pair plot or scatter plot matrix, is a grid of scatter plots that visualizes the pairwise relationships between variables in a DataFrame. It allows us to examine how variables are related to each other by plotting their values against each other.

3. Basemap object :-
   Basemap allows you to create different map projections, draw coastlines, countries, continents, and gridlines, and plot data on the map. It provides a high level of customization for creating various types of maps
   To use Basemap, you need to install the basemap package. You can install it using the following command:-
   pip install basemap

4.Piecharts And Barcharts:-

1. *Creating a Pie Chart of the Distribution of Disaster Types:*

- The code starts by grouping the DataFrame **df** by the "Title" column and calculating the count of each unique disaster type using the **groupby()** and **size()** functions. The result is stored in the **disaster_types** DataFrame.
- Next, the **px.pie()** function from the Plotly Express library (**px**) is used to create a pie chart. It takes the **disaster_types** DataFrame as input and specifies the values to be plotted as "Count" and the names of the pie slices as "Title". The title of the chart is set as "Distribution of Disaster Types".
- Finally, the **fig.show()** function is called to display the pie chart.

2. *Creating a Bar Chart of the Number of Disasters by Year:*

- The code groups the DataFrame **df** by the "Year" column and calculates the count of disasters for each year using the **groupby()** and **size()** functions. The result is stored in the **disasters_by_year** DataFrame.
- The **px.bar()** function is then used to create a bar chart. It takes the **disasters_by_year** DataFrame as input and specifies the "Year" column as the x-axis and the "Count" column as the y-axis. The title of the chart is set as "Number of Disasters by Year".
- Finally, the **fig.show()** function is called to display the bar chart.

3. *Creating a Pie Chart of Earthquake Magnitudes*:

- The code directly uses the **px.pie()** function to create a pie chart of earthquake magnitudes.
- The **df** DataFrame is used as input, and the "magType" column is specified as the names of the pie slices.
- The **fig.show()** function is called to display the pie chart.

Overall, the code snippet utilizes Plotly Express (**px**) to create interactive and visually appealing pie charts and bar charts. Each chart is generated based on specific data manipulations and column selections from the original DataFrame **df**. The resulting charts provide insights into the distribution of disaster types, the number of disasters by year, and the magnitudes of earthquakes.

5 . Histogram of Earthquake Magnitudes:-

1. *Creating a Histogram of Earthquake Magnitudes*:
    - The code uses the **px.histogram()** function from the Plotly Express library (**px**) to create a histogram.
    - The DataFrame **df** is passed as input, and the "mag" column is specified as the data to be plotted on the x-axis of the histogram.
    - The parameter **nbins=20** sets the number of bins (or bars) in the histogram to 20, indicating the desired level of granularity in the magnitude range.
    - The resulting histogram object is stored in the **fig** variable.
2. *Displaying the Histogram*:
    - The **fig.show()** function is called to display the histogram.
    - This will open a new window or notebook output cell (depending on the environment) showing the interactive histogram plot.

6.3D scatter plot:-

3D scatter plot using the Plotly library to visualize earthquake data. The plot represents the relationship between earthquake magnitudes, depths, and gaps

7.Heatmap:-

Heatmap using the Plotly Express library to visualize the number of earthquakes based on their occurrence year and magnitude.

The px.density_heatmap() function is used to create the heatmap.

CODING AND OUTPUT

IN:-

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

IN:-

```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]]
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) // nGraphPerRow # corrected the calculation for nGraphRow
    plt.figure(figsize=(6 * nGraphPerRow, 8 * nGraphRow))
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if not np.issubdtype(type(columnDf.iloc[0]), np.number):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
```

```
        plt.ylabel('Counts')
        plt.xticks(rotation=90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad=1.0, w_pad=1.0, h_pad=1.0)
    plt.show()
```

IN:-

```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where 
there are more than 1 unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant 
columns ({df.shape[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, 
facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

IN:-

```
# Scatter and density plots
```

```
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

IN:-

```
nRowsRead = 1000 # specify 'None' if want to read whole file
df1 = pd.read_csv('C:\\New folder\\cyclones - Sheet1.csv', delimiter=',', nrows = nRowsRead)
df1.dataframeName = 'cyclones - Sheet1.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')
```

OUT:-

There are 45 rows and 3 columns

IN:-

```
df1.head(5)
```

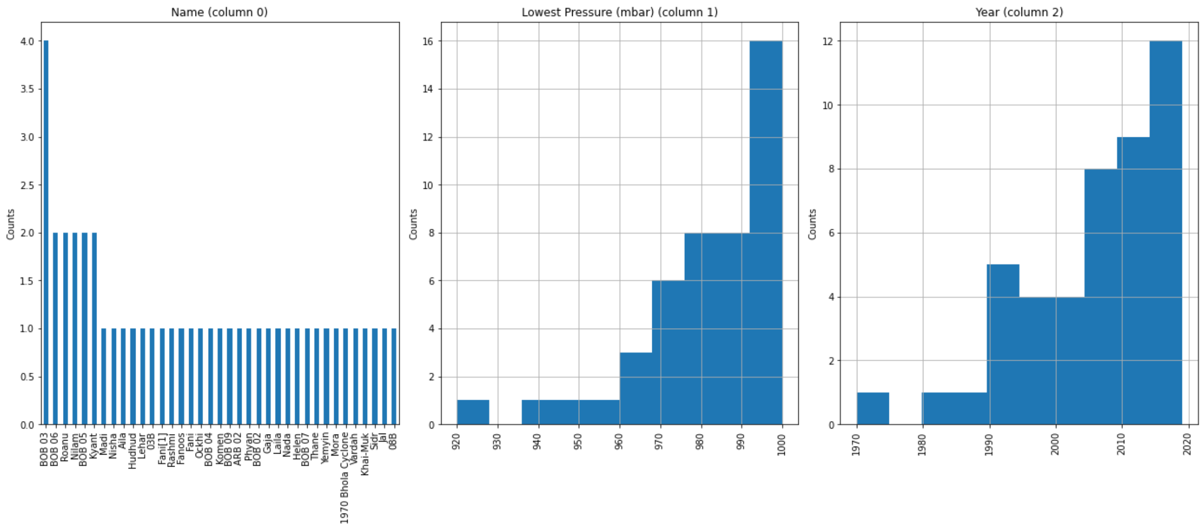
OUT:-

	Name	Lowest Pressure (mbar)	Year
0	BOB 02	920	1990
1	BOB 05	982	1998
2	03B	992	2003
3	Yemyin	986	2007
4	Khai-Muk	996	2008

IN:-

```
plotPerColumnDistribution(df1, 10, 5)
```

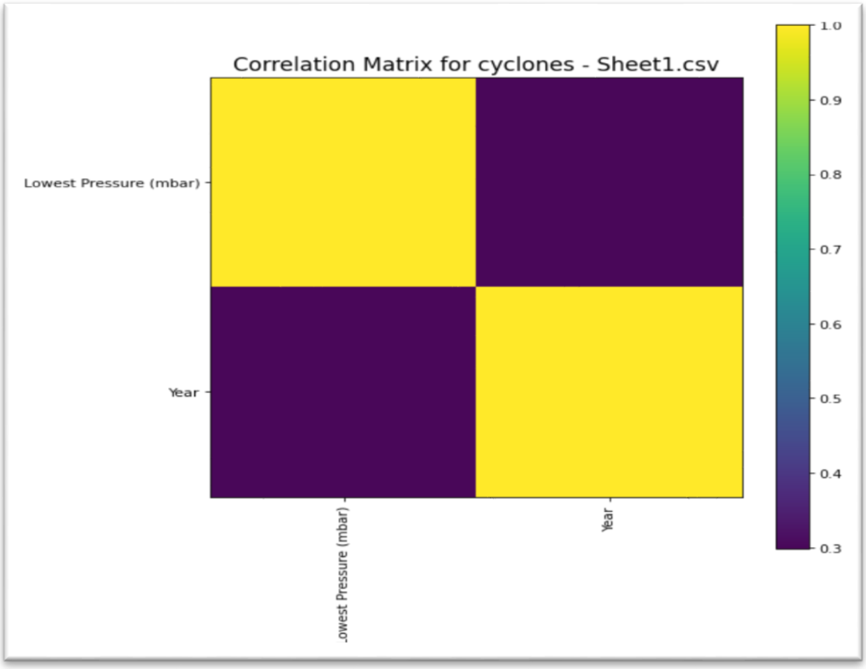
OUT:-



IN:-

```
plotCorrelationMatrix(df1, 8)
```

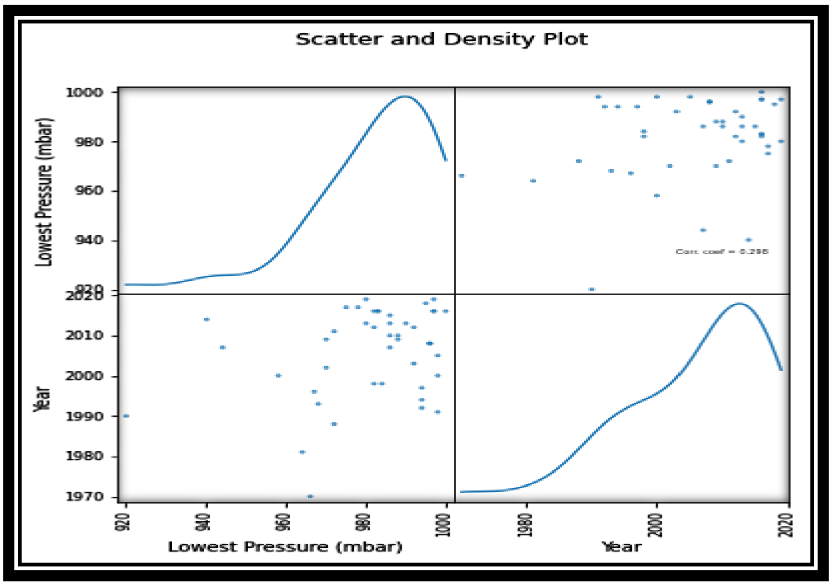
OUT:-



IN:-

plotScatterMatrix(df1, 6, 5)

OUT:-



IN:-

nRowsRead = 1000 # specify 'None' if want to read whole file

df2 = pd.read_csv('C:\\New folder\\earthquakes - Sheet1.csv', delimiter=',', nrows = nRowsRead)

df2.dataframeName = 'earthquakes - Sheet1.csv'

nRow, nCol = df2.shape

print(f'There are {nRow} rows and {nCol} columns')

OUT:-

There are 25 rows and 8 columns

IN:-

df2.head(5)

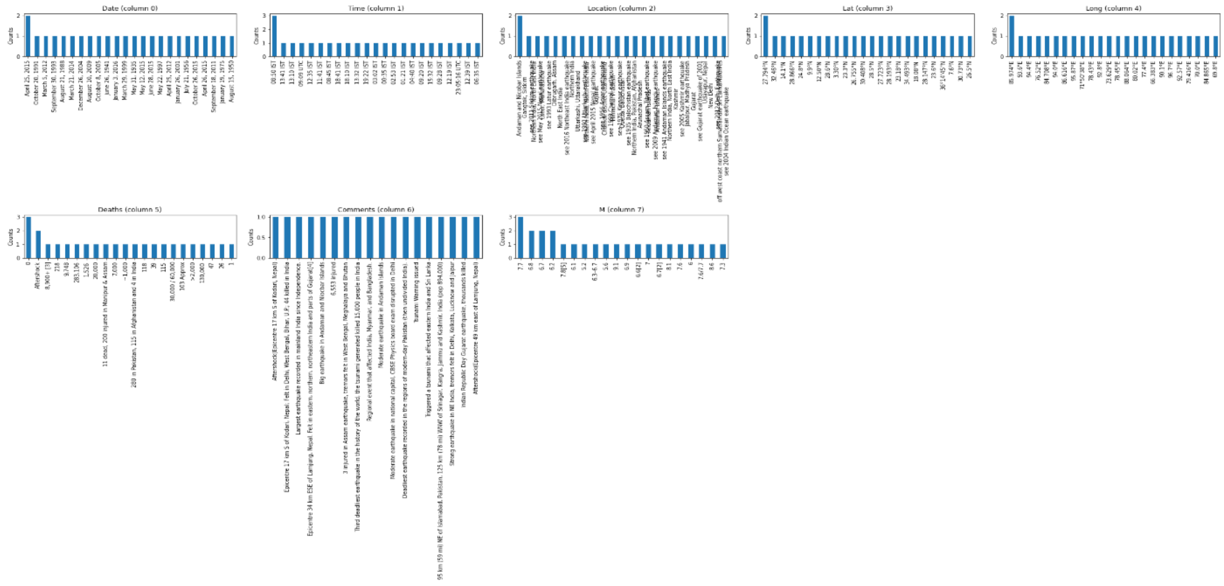
OUT:-

	Date	Time	Location	Lat	Long	Deaths	Comments	M
0	January 3, 2016	23:05:16 UTC	North East India\nsee 2016 Northeast India ear...	24.8°N	93.6°E	11 dead, 200 injured in Manipur & Assam	Regional event that affected India, Myanmar, a...	6.7
1	October 26, 2015	09:09 UTC	Northern India, Pakistan, Afghanistan	36°14′45″N	71°50′38″E	280 in Pakistan, 115 in Afghanistan and 4 in I...	NaN	7.7
2	June 28, 2015	06:35 IST	Dibrugarh, Assam	26.5°N	90.1°E	0	3 injured in Assam earthquake, tremors felt in...	5.6
3	May 12, 2015	12:35 IST	Northern India, North East India\nsee May 2015...	27.794°N	85.974°E	218	Epicentre 17 km S of Kodari, Nepal; Felt in De...	7.3
4	April 26, 2015	12:39 IST	Northern India, North East India	27.794°N	85.974°E	Aftershock	Aftershock(Epicentre 17 km S of Kodari, Nepal)	6.7[2]

IN:-

plotPerColumnDistribution(df2, 16, 5)

OUT:-



IN:-

```
#Importing necessary libaries
import pandas as pd
from sklearn.cluster import DBSCAN
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
import math
import sys
```

```
if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

IN :-

```
# Reading data
df = pd.read_csv('C:\\Indian_earthquake_data.csv')
df.head()
```

OUT:-

	Origin Time	Latitude	Longitude	Depth	Magnitude	Location
0	2021-07-31 09:43:23 IST	29.06	77.42	5.0	2.5	53km NNE of New Delhi, India
1	2021-07-30 23:04:57 IST	19.93	72.92	5.0	2.4	91km W of Nashik, Maharashtra, India
2	2021-07-30 21:31:10 IST	31.50	74.37	33.0	3.4	49km WSW of Amritsar, Punjab, India
3	2021-07-30 13:56:31 IST	28.34	76.23	5.0	3.1	50km SW of Jhajjar, Haryana
4	2021-07-30 07:19:38 IST	27.09	89.97	10.0	2.1	53km SE of Thimphu, Bhutan

IN:-

```
#PREPROCESSING
df['Origin Time'] = pd.to_datetime(df['Origin Time'])
```

IN:-

```
#FIND NULL VALUES
df.isna().sum()
```

OUT:-

```
Origin Time    0
Latitude       0
Longitude      0
Depth          0
Magnitude      0
Location       0
dtype: int64
```

IN:-

```
# mean ,and Quantile of Magnitude
```

```
q1 = df["Magnitude"].quantile(.25)
q2 = df["Magnitude"].quantile(.5)
q3 = df["Magnitude"].quantile(.75)
q90 = df["Magnitude"].quantile(.9)
q95 = df["Magnitude"].quantile(.95)
q99 = df["Magnitude"].quantile(.99)
mean =df["Magnitude"].mean()
print("mean = ",mean,"\nFirst Quartile = ",q1)
print("Second Quartile = ",q2,"\nThird Quartile = ",q3)
print("99th Quantile = ",q99)
```

OUT:-

```
mean =  3.7721956601691793
First Quartile =  3.2
Second Quartile =  3.9
Third Quartile =  4.3
99th Quantile =  5.5
```

IN:-

```
def cluster_distance(lat,lon):
    # custom distance funciton for clustering
    # returns distance betowin points on the earth surface in km
    lat1,lon1= lat
    lat2,lon2= lon
    R = 6371e3; # earth
    f1 = lat1 * math.pi/180; # phi lambda in radians
    f2 = lat2 * math.pi/180;
    delf = (lat2-lat1) * math.pi/180;
    dellambda = (lon2-lon1) * math.pi/180;
    a = math.sin(delf/2) * math.sin(delf/2) +  math.cos(f1) * math.cos(f2) *
math.sin(dellambda/2) * math.sin(dellambda/2);
```

```
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a));
    d = R * c; # in metres
    return d/1000 # in km
X = df[['Latitude', 'Longitude']].values
aa = DBSCAN( eps=5,min_samples=1,metric=cluster_distance)
aa = aa.fit(X)
```

IN:-

```
# setting lables for clusters
df["cluster_label"] = aa.labels_
```

IN:-

```
print("NUMBER OF CLUSTERS ",aa.labels_.max())
```

OUT:-

```
NUMBER OF CLUSTERS  2008
```

IN:-

```
earth_quake_with_sequence = 0
earth_quake_without_sequence = 0
def cluster_distance_time(time1,time2):
    diff = pd.to_datetime(time1)-pd.to_datetime(time2)
    diff = float(abs(diff.days.values[0]))
    return diff
X = df[['Origin Time']].values
from sklearn.cluster import DBSCAN
import numpy as np
from scipy.cluster.hierarchy import ward, fcluster
from scipy.spatial.distance import pdist
maxcluster = aa.labels_.max() + 1
mincluster = aa.labels_.min()
```

```
counttotal = 0
for i in range(mincluster , maxcluster):
    dftemp = df[df['cluster_label']==i]
    if i < 0:
        if len(dftemp) > 0:
            earth_quake_without_sequence += len(dftemp)
        continue
    if len(dftemp) <=1:
        if len(dftemp) == 1:
            earth_quake_without_sequence += 1
        continue
    X = dftemp[['Origin Time']].values
    aa2 = DBSCAN( eps=7,min_samples=1,metric=cluster_distance_time)
    aa2 = aa2.fit(X)
    dftemp[ 'time_cluster'] = aa2.labels_
    tempmax = aa2.labels_.max()+1
    tempmin = aa2.labels_.min()
    for l  in range (tempmin, tempmax):
        dftemplen = len(dftemp[dftemp[ 'time_cluster'] == l])
        counttotal += dftemplen
        if l < 0:
            if dftemplen > 0:
                earth_quake_without_sequence += dftemplen
            continue
        if dftemplen > 1:
            earth_quake_with_sequence += dftemplen
        elif dftemplen == 1:
```

```
        earth_quake_without_sequence +=1
```

IN:-

```
print("EARTHQUAKE WITHOUT SEQUENCE  = ", earth_quake_without_sequence)
print("EARTHQUAKE WITH SEQUENCE  = ", earth_quake_with_sequence)
# earth_quake_with_sequence + earth_quake_without_sequence
```

OUT:-

```
EARTHQUAKE WITHOUT SEQUENCE  =  2403
EARTHQUAKE WITH SEQUENCE  =  316
```

IN:-

```
print("TOTAL EARTHQUAKE COUNT CHECK \n[earth_quake_with_sequence + earth_quake_without_sequence = len(df)] ? = ",(earth_quake_with_sequence + earth_quake_without_sequence) == len(df))
```

OUT:-

```
TOTAL EARTHQUAKE COUNT CHECK
[earth_quake_with_sequence + earth_quake_without_sequence = len(df)] ? =  True
```

IN:-

```
print("PROBABILITY OF SEQUENCE EARTHQUAKE = ",(earth_quake_with_sequence/len(df))*100 ,"%")
```

OUT:-

```
PROBABILITY OF SEQUENCE EARTHQUAKE =  11.62191982346451 %
```

IN:-

```
df.info()
```

OUT:-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2719 entries, 0 to 2718
Data columns (total 7 columns):
```

```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Origin Time    2719 non-null   datetime64[ns]
 1   Latitude       2719 non-null   float64
 2   Longitude      2719 non-null   float64
 3   Depth          2719 non-null   float64
 4   Magnitude      2719 non-null   float64
 5   Location       2719 non-null   object
 6   cluster_label  2719 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 148.8+ KB
```

IN:-

```
df[['time']] = df['Origin Time'].apply(lambda x: str(x)[0:11])
df
```

OUT:-

	Origin Time	Latitude	Longitude	Depth	Magnitude	Location	cluster_label	time
0	2021-07-31 09:43:23	29.06	77.42	5.0	2.5	53km NNE of New Delhi, India	0	2021-07-31
1	2021-07-30 23:04:57	19.93	72.92	5.0	2.4	91km W of Nashik, Maharashtra, India	1	2021-07-30
2	2021-07-30 21:31:10	31.50	74.37	33.0	3.4	49km WSW of Amritsar, Punjab, India	2	2021-07-30
3	2021-07-30 13:56:31	28.34	76.23	5.0	3.1	50km SW of Jhajjar, Haryana	3	2021-07-30
4	2021-07-30 07:19:38	27.09	89.97	10.0	2.1	53km SE of Thimphu, Bhutan	4	2021-07-30
...	...	...	...	...	...	...	...	...
2714	2019-08-04 06:56:19	12.30	94.80	10.0	4.8	224km ESE of Diglipur, Andaman and Nicobar isl...	2006	2019-08-04
2715	2019-08-04 05:40:33	24.70	94.30	40.0	4.1	31km SW of Ukhrul, Manipur, India	1609	2019-08-04
2716	2019-08-03 16:29:37	22.50	88.10	10.0	3.6	28km WSW of Kolkata, India	2007	2019-08-03
2717	2019-08-03 01:59:11	24.60	94.20	54.0	3.5	35km SE of Imphal, Manipur, India	1866	2019-08-03
2718	2019-08-01 06:13:21	14.50	92.90	10.0	4.6	137km N of Diglipur, Andaman and Nicobar islan...	2008	2019-08-01

2719 rows × 8 columns

IN:-

```
df = df[['time','Depth','Magnitude']]
df
```

OUT:-

	time	Depth	Magnitude
0	2021-07-31	5.0	2.5
1	2021-07-30	5.0	2.4
2	2021-07-30	33.0	3.4
3	2021-07-30	5.0	3.1
4	2021-07-30	10.0	2.1
...	...	...	...
2714	2019-08-04	10.0	4.8
2715	2019-08-04	40.0	4.1
2716	2019-08-03	10.0	3.6
2717	2019-08-03	54.0	3.5
2718	2019-08-01	10.0	4.6

2719 rows × 3 columns

IN:-

```
df['time'] = pd.to_datetime(df['time'])
df.info()
```

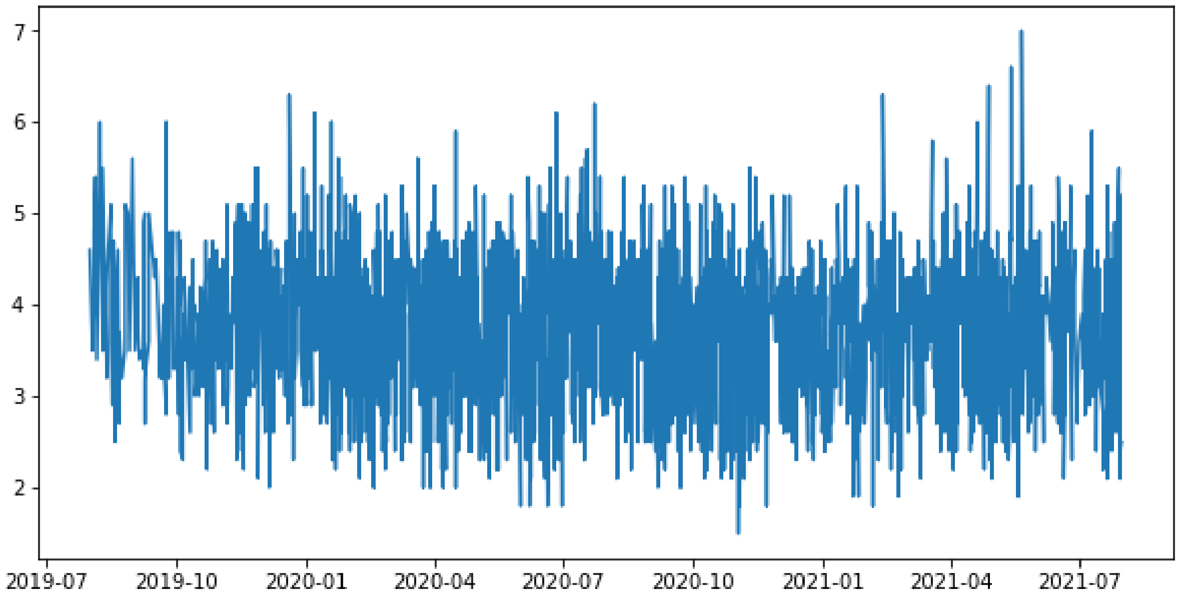
OUT:-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2719 entries, 0 to 2718
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   time       2719 non-null   datetime64[ns]
 1   Depth      2719 non-null   float64
 2   Magnitude  2719 non-null   float64
dtypes: datetime64[ns](1), float64(2)
memory usage: 63.9 KB
```

IN:-

```
plt.figure(figsize=(10,5))
plt.plot(df.time,df.Magnitude)
plt.show()
```

OUT:-



IN:-

```
grouped_df = df.groupby(by='time').mean()
grouped_df
```

OUT:-

time	Depth	Magnitude
2019-08-01	10.000000	4.600
2019-08-03	32.000000	3.550
2019-08-04	25.000000	4.450
2019-08-05	190.000000	5.400
2019-08-06	33.000000	3.400
...	...	...
2021-07-26	22.250000	4.375
2021-07-27	85.000000	3.450
2021-07-29	19.000000	4.300
2021-07-30	29.666667	3.200
2021-07-31	5.000000	2.500

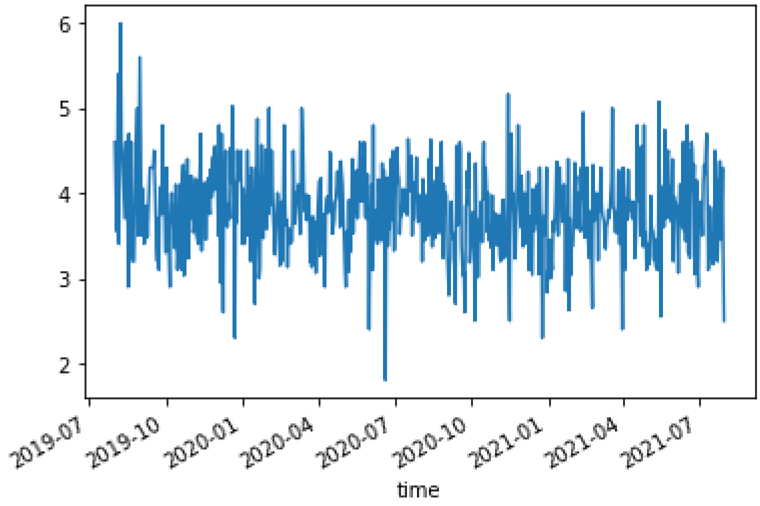
679 rows × 2 columns

IN:-

```
grouped_df.Magnitude.plot()
```

OUT:-

<AxesSubplot:xlabel='time'>



IN:-

```
model_df = grouped_df.drop(columns=['Depth'])
model_df
```

OUT:-

time	Magnitude
2019-08-01	4.600
2019-08-03	3.550
2019-08-04	4.450
2019-08-05	5.400
2019-08-06	3.400
...	...
2021-07-26	4.375
2021-07-27	3.450
2021-07-29	4.300
2021-07-30	3.200
2021-07-31	2.500

679 rows × 1 columns

IN:-

```
model_df = model_df.resample('D').sum()
model_df
```

OUT:-

time	Magnitude
2019-08-01	4.60
2019-08-02	0.00
2019-08-03	3.55
2019-08-04	4.45
2019-08-05	5.40
...	...
2021-07-27	3.45
2021-07-28	0.00
2021-07-29	4.30
2021-07-30	3.20
2021-07-31	2.50

731 rows × 1 columns

IN:-

```
model_df.reset_index(inplace=True)
```

IN:-

```
model_df.columns = ['ds','y']
```

IN:-

```
model_df
```

OUT:-

	ds	y
0	2019-08-01	4.60
1	2019-08-02	0.00
2	2019-08-03	3.55
3	2019-08-04	4.45
4	2019-08-05	5.40
...	...	...
726	2021-07-27	3.45
727	2021-07-28	0.00
728	2021-07-29	4.30
729	2021-07-30	3.20
730	2021-07-31	2.50

731 rows × 2 columns

IN:-

```
train_df = model_df[:-30]
test_df = model_df[-30:]
```

IN:-

```
train_df
```

OUT:-

	ds	y
0	2019-08-01	4.60
1	2019-08-02	0.00
2	2019-08-03	3.55
3	2019-08-04	4.45
4	2019-08-05	5.40
...	...	...
696	2021-06-27	0.00
697	2021-06-28	4.15
698	2021-06-29	3.10
699	2021-06-30	2.90
700	2021-07-01	3.60

701 rows × 2 columns

IN:-

test_df

OUT:-

	ds	y
701	2021-07-02	0.000000
702	2021-07-03	3.900000
703	2021-07-04	3.733333
704	2021-07-05	3.500000
705	2021-07-06	4.100000
706	2021-07-07	4.325000
707	2021-07-08	4.225000
708	2021-07-09	4.466667
709	2021-07-10	4.700000

710	2021-07-11	3.850000
711	2021-07-12	3.100000
712	2021-07-13	3.300000
713	2021-07-14	3.840000
714	2021-07-15	3.800000
715	2021-07-16	3.300000
716	2021-07-17	0.000000
717	2021-07-18	3.166667
718	2021-07-19	3.300000
719	2021-07-20	4.500000
720	2021-07-21	3.400000
721	2021-07-22	4.150000
722	2021-07-23	3.200000
723	2021-07-24	3.433333
724	2021-07-25	4.100000
725	2021-07-26	4.375000
726	2021-07-27	3.450000
727	2021-07-28	0.000000
728	2021-07-29	4.300000
729	2021-07-30	3.200000
730	2021-07-31	2.500000

IN:-

```
# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load in
```

```
import pandas as pd

import matplotlib

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

get_ipython().run_line_magic('matplotlib', 'inline')

# Input data files are available in the "../input/" directory.

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, _, filenames in os.walk('C:\\New folder\\'):

    for filename in filenames:

        print(os.path.join(dirname, filename))
```

OUT:-

```
C:\New folder\cyclones - Sheet1.csv
C:\New folder\Earthquake.csv
C:\New folder\earthquakes - Sheet1.csv
```

IN:-

```
df=pd.read_csv('C:\\New folder\\Earthquake.csv',engine='python')

df.head()
```

OUT:-

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type	horizontalError	depthErr
0	2009-12-31T09:57:29.720Z	27.319	91.510	10.0	5.5	mb	205.0	37.4	NaN	0.89	...	2017-04-26T18:09:55.932Z	Bhutan	earthquake	NaN	Na
1	2009-12-29T13:33:22.870Z	35.017	73.005	63.8	4.0	mb	40.0	95.8	NaN	0.94	...	2014-11-07T01:40:19.294Z	northwestern Kashmir	earthquake	NaN	8
2	2009-12-29T09:01:55.310Z	24.357	94.807	124.8	5.6	mwb	206.0	17.3	NaN	0.77	...	2016-11-10T02:22:03.905Z	Myanmar-India border region	earthquake	NaN	Na
3	2009-12-28T02:15:04.870Z	30.686	83.769	10.0	4.4	mb	50.0	40.6	NaN	1.08	...	2014-11-07T01:40:19.031Z	western Xizang	earthquake	NaN	Na
4	2009-12-26T00:23:38.570Z	14.001	92.862	42.6	5.0	mb	117.0	68.1	NaN	0.82	...	2014-11-07T01:40:18.641Z	Andaman Islands, India region	earthquake	NaN	5

5 rows × 22 columns

IN:-

```
len(df.index)
```

OUT:-

```
14698
```

IN:-

```
df.nunique()
```

OUT:-

```
time               14698
latitude           10713
longitude          10140
depth               3554
mag                   48
magType               10
nst                  427
gap                 2253
dmin                1954
rms                  171
net                    2
id                 14698
updated            13594
place               2419
type                   1
horizontalError      145
depthError           463
magError             269
magNst               193
status                 2
locationSource         5
magSource              8
dtype: int64
```

IN:-

```
df.drop_duplicates(subset ="time",keep = False, inplace = True)
```

IN:-

```
len(df.index)
```

OUT:-

14698
IN:-

df.fillna('')

#df.isnull().sum().sum()

OUT:-

time latitude longitude depth mag magType nst gap dmin rms ... updated place type horizontalError depthError magError magNst status locationSource magSource

0 2009-12-31T09:57:29.720Z 27.319 91.510 10.0 5.5 mb 205 37.4 0.89 ... 2017-04-26T18:09:55.932Z Bhutan earthquake 118 reviewed us us

1 2009-12-29T13:33:22.870Z 35.017 73.005 63.8 4.0 mb 40 95.8 0.94 ... 2014-11-07T01:40:19.294Z northwestern Kashmir earthquake 8.6 7 reviewed us us

2 2009-12-29T09:01:55.310Z 24.357 94.807 124.8 5.6 mwb 206 17.3 0.77 ... 2016-11-10T02:22:03.905Z Myanmar-India border region earthquake reviewed us us

3 2009-12-28T02:15:04.870Z 30.686 83.769 10.0 4.4 mb 50 40.6 1.08 ... 2014-11-07T01:40:19.031Z western Xizang earthquake 9 reviewed us us

4 2009-12-26T00:23:38.570Z 14.001 92.862 42.6 5.0 mb 117 68.1 0.82 ... 2014-11-07T01:40:18.641Z Andaman Islands, India region earthquake 5.7 56 reviewed us us

... ...

14693 2010-01-11T16:07:13.800Z 36.041 70.893 104.7 4.6 mb 41 93.7 1.11 ... 2014-11-07T01:40:26.476Z Hindu Kush region, Afghanistan earthquake 6 reviewed us us

14694 2010-01-11T05:15:16.160Z 29.744 80.557 35.0 4.7 mb 27 84.6 1.09 ... 2014-11-07T01:40:26.418Z Nepal-India border region earthquake 0 4 reviewed us us

14695 2010-01-11T03:38:43.300Z 14.243 93.505 66.4 4.0 mb 18 181.4 0.85 ... 2014-11-07T01:40:26.412Z Andaman Islands, India region earthquake 22.2 2 reviewed us us

14696 2010-01-05T14:28:18.340Z 32.384 85.273 54.1 4.9 mb 58 108.2 0.83 ... 2014-11-07T01:40:25.340Z western Xizang earthquake 11.4 15 reviewed us us

14697 2010-01-01T02:22:23.820Z 30.646 83.791 10.0 5.2 mwc 84 78.6 1.09 ... 2016-11-10T02:22:11.026Z western Xizang earthquake

IN:-

```
df.columns
```

OUT:-

```
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',
       'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place', 'type',
       'horizontalError', 'depthError', 'magError', 'magNst', 'status',
       'locationSource', 'magSource'],
      dtype='object')
```

IN:-

```
df['time'] = pd.to_datetime(df['time'])
df.head()
```

OUT:-

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type	horizontalError	dept
0	2009-12-31 09:57:29.720000+00:00	27.319	91.510	10.0	5.5	mb	205.0	37.4	NaN	0.89	...	2017-04-26T18:09:55.932Z	Bhutan	earthquake	NaN	
1	2009-12-29 13:33:22.870000+00:00	35.017	73.005	63.8	4.0	mb	40.0	95.8	NaN	0.94	...	2014-11-07T01:40:19.294Z	northwestern Kashmir	earthquake	NaN	
2	2009-12-29 09:01:55.310000+00:00	24.357	94.807	124.8	5.6	mwb	206.0	17.3	NaN	0.77	...	2016-11-10T02:22:03.905Z	Myanmar-India border region	earthquake	NaN	
3	2009-12-28 02:15:04.870000+00:00	30.686	83.769	10.0	4.4	mb	50.0	40.6	NaN	1.08	...	2014-11-07T01:40:19.031Z	western Xizang	earthquake	NaN	
4	2009-12-26 00:23:38.570000+00:00	14.001	92.862	42.6	5.0	mb	117.0	68.1	NaN	0.82	...	2014-11-07T01:40:18.641Z	Andaman Islands, India region	earthquake	NaN	

5 rows × 22 columns

IN:-

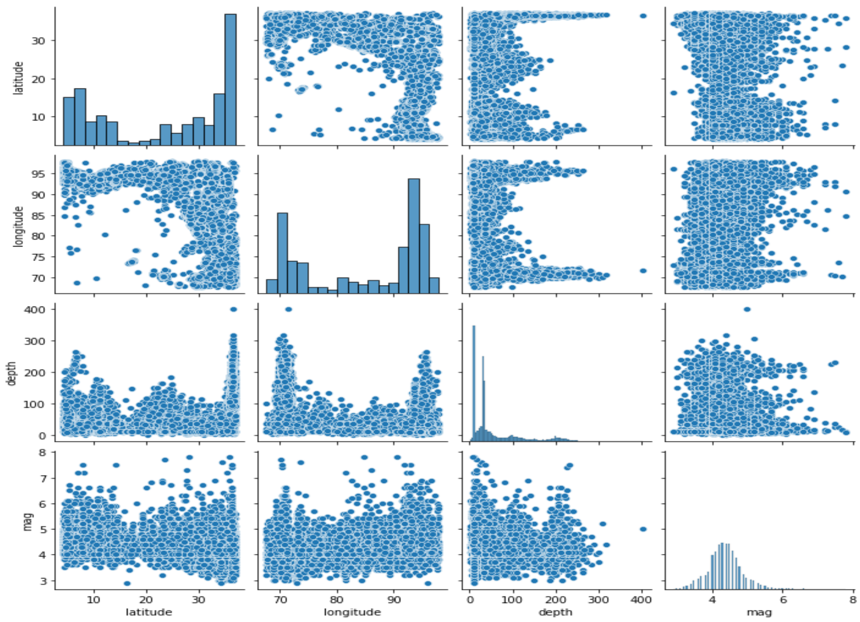
```
df = df.drop(columns = ['magType', 'nst','gap', 'dmin', 'rms', 'net', 'id', 'updated', 'type',
       'horizontalError', 'depthError', 'magError', 'magNst', 'status',
       'locationSource', 'magSource'])
```

IN:-

```
#sns.pairplot(df,vars=df.columns[1:],)
sns.pairplot(df)
```

OUT:-

<seaborn.axisgrid.PairGrid at 0x2636e4626a0>

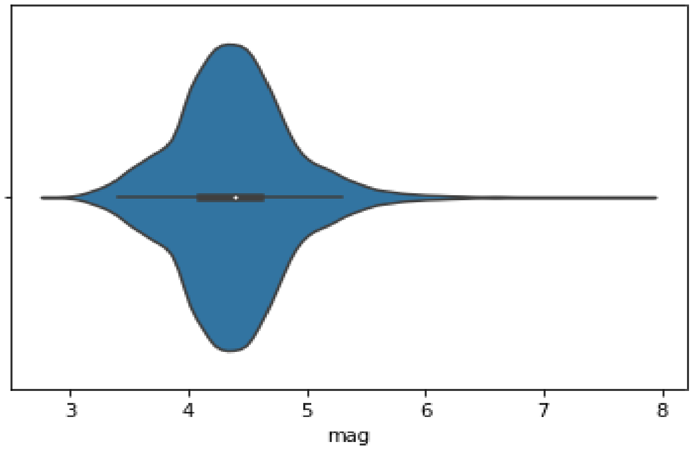


IN:-

sns.violinplot(df['mag'],orient='v')

OUT:-

<AxesSubplot:xlabel='mag'>



IN:-

df[df['mag']==df['mag'].max()]

OUT:-

	time	latitude	longitude	depth	mag	place
9586	2001-11-14 09:26:10.010000+00:00	35.9460	90.5410	10.00	7.8	southern Qinghai, China
12139	2015-04-25 06:11:25.950000+00:00	28.2305	84.7314	8.22	7.8	36km E of Khudi, Nepal

IN:-

(df['mag']>=3.5).value_counts()

OUT:-

True   14344
False   354
Name: mag, dtype: int64

IN:-

new_df=df.loc[df['mag'] >= 3.5].reset_index().drop(columns=['index']) len(new_df.index)

OUT:-

14344
IN:-

new_df.head()

OUT:-

	time	latitude	longitude	depth	mag	place
0	2009-12-31 09:57:29.720000+00:00	27.319	91.510	10.0	5.5	Bhutan
1	2009-12-29 13:33:22.870000+00:00	35.017	73.005	63.8	4.0	northwestern Kashmir
2	2009-12-29 09:01:55.310000+00:00	24.357	94.807	124.8	5.6	Myanmar-India border region
3	2009-12-28 02:15:04.870000+00:00	30.686	83.769	10.0	4.4	western Xizang
4	2009-12-26 00:23:38.570000+00:00	14.001	92.862	42.6	5.0	Andaman Islands, India region

IN:-

diction={}

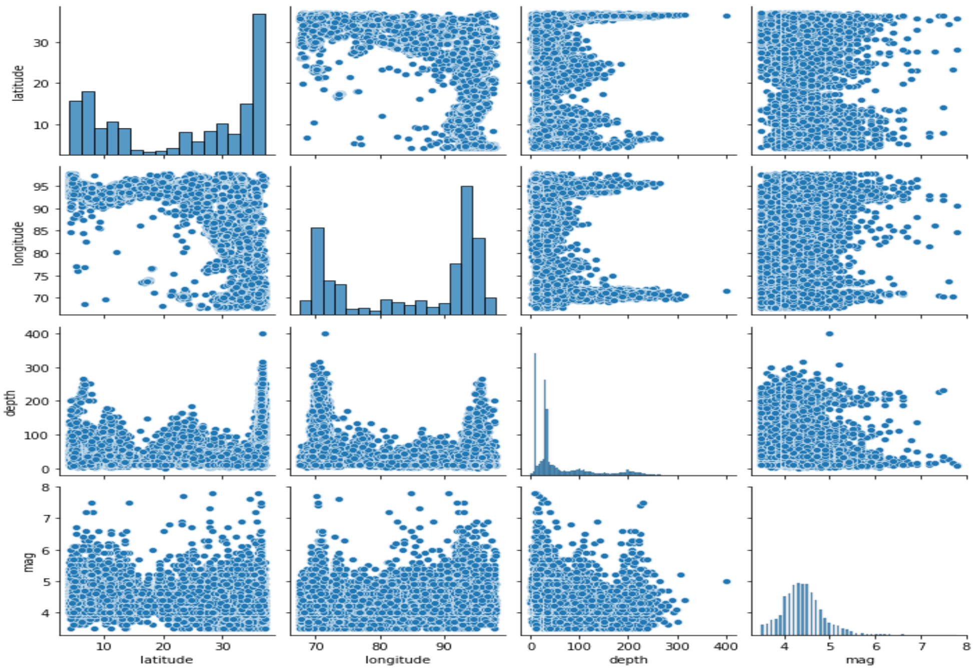
IN:-

new_df.drop(['place'], axis =1 , inplace = True)

IN:-

sns.pairplot(new_df,vars=new_df.columns[1:])

OUT:- <seaborn.axisgrid.PairGrid at 0x2636339bd90>

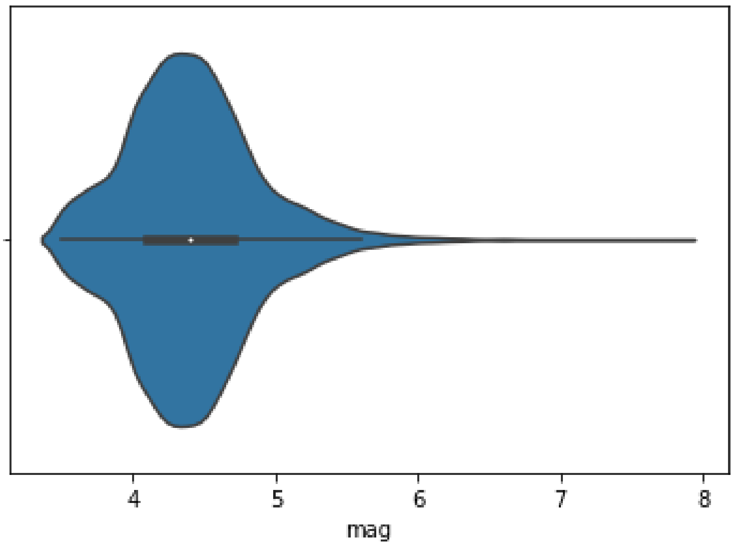


IN:-

```
sns.violinplot(new_df['mag'],orient='v')
```

OUT:-

```
<AxesSubplot:xlabel='mag'>
```

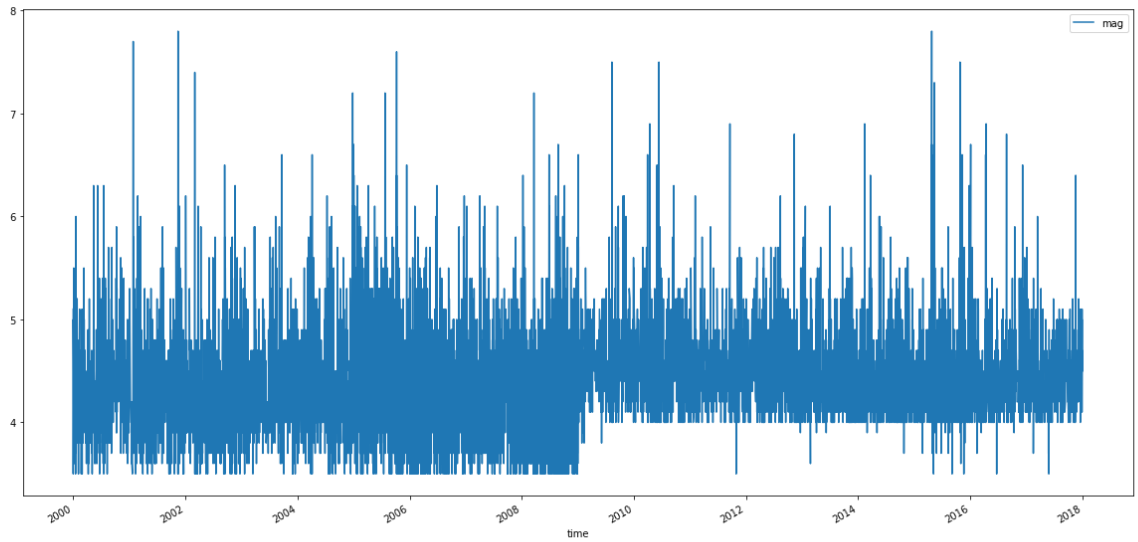


IN:-

```
new_df.plot(x='time',y='mag',figsize=(20,10))
```

- OUT:-

<AxesSubplot:xlabel='time'>



IN:-

```
for i in range(2000,2020,5):
    mask = (new_df['time'] > str(i+1)+'-1-1') & (new_df['time'] <= str(i+5)+'-12-31')
    diction.update({(str(i+1)+'-'+str(i+5)):new_df.loc[mask].reset_index().drop(columns=['index'])})
```

IN:-

```
diction['2001-2005'].head()
```

OUT:-

	time	latitude	longitude	depth	mag
0	2005-12-30 23:56:04.650000+00:00	5.508	94.546	49.5	4.6
1	2005-12-30 19:47:31.340000+00:00	11.524	93.553	30.0	3.8
2	2005-12-30 19:31:59.800000+00:00	36.250	71.262	186.4	3.8
3	2005-12-30 15:18:14.030000+00:00	36.660	71.108	224.2	4.5
4	2005-12-30 11:13:43.510000+00:00	36.536	71.063	192.6	4.5

IN:-

```
mean_mag=[]
time=[]
interval=[]
sqrt_dE=[]
b=[]
a=[]
niu=[]
delta_M=[]
max_mag=[]
for key in diction:
    interval+=[key]
    time+=[diction[key].time.max()-diction[key].time.min()]
    mean_mag+=[diction[key].mag.mean()]
    sqrt_dE+=[sum((10**(11.8+1.5*diction[key].mag))**0.5)]
    n=len(diction[key])
```

```
    Ni=[]
    for i in range(0,len(diction[key]),1):
        Ni+=[(diction[key].mag[i]<=diction[key].mag).sum()]
    sum_1=0
    for i in range(0,len(diction[key]),1):
        sum_1+=diction[key].mag[i]*np.log10(Ni[i])
    sum_mi=sum(diction[key].mag)
    sum_ni=0
    for i in range(0,len(diction[key]),1):
        sum_ni+=np.log10(Ni[i])
    sum_mi2=sum(diction[key].mag**2)
    b_temp=0
    b_temp=(n*sum_1-sum_mi*sum_ni)/(sum_mi**2-n*sum_mi2)
    b+=[b_temp]
    a_temp=0
    for i in range(0,len(diction[key]),1):
        a_temp+=(np.log(Ni[i])+b_temp*diction[key].mag[i])/n
    a+=[a_temp]
    niu_temp=0
    for i in range(0,len(diction[key]),1):
        niu_temp+=((np.log(Ni[i])-(a_temp-b_temp*diction[key].mag[i]))**2/(n-
1))
    niu+=[niu_temp]
    delta_M+=[abs(diction[key].mag.max()-a_temp/b_temp)/10]
    max_mag+=[diction[key].mag.max()]
time=pd.to_timedelta(time, errors='coerce').days
sqrt_dE = [i / j for i, j in zip(sqrt_dE , time)]
df=pd.DataFrame({'period':interval,
```

'T':time,

'mean_mag':mean_mag,

'Speed':sqrt_dE,

'b':b,

'niu':niu,

'delta_M':delta_M,

'max_mag': max_mag,

#'Ptest':[0]*24,

#'Ytest':[0]*24,

})

df.head()

OUT:-

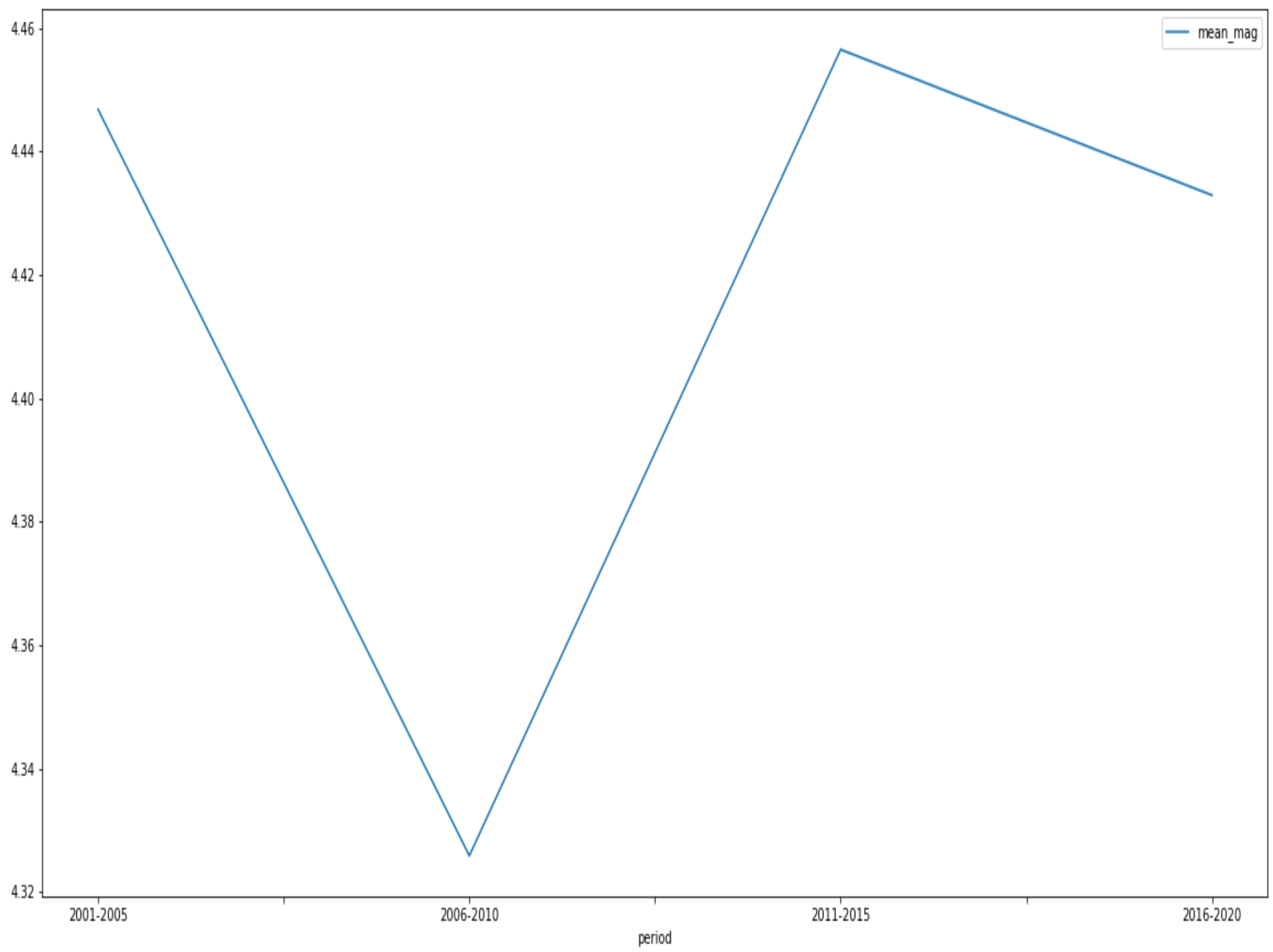
	period	T	mean_mag	Speed	b	niu	delta_M	max_mag
0	2001-2005	1823	4.446800	9.769447e+09	0.814042	0.358947	0.621844	7.8
1	2006-2010	1823	4.325855	5.429726e+09	0.807539	0.351617	0.600057	7.5
2	2011-2015	1822	4.456482	4.528406e+09	1.042164	0.319705	0.344146	7.8
3	2016-2020	730	4.432896	3.130998e+09	1.102924	0.320471	0.291965	6.9

IN:-

df.plot(x='period',y='mean_mag',figsize=(20,10))

OUT:-

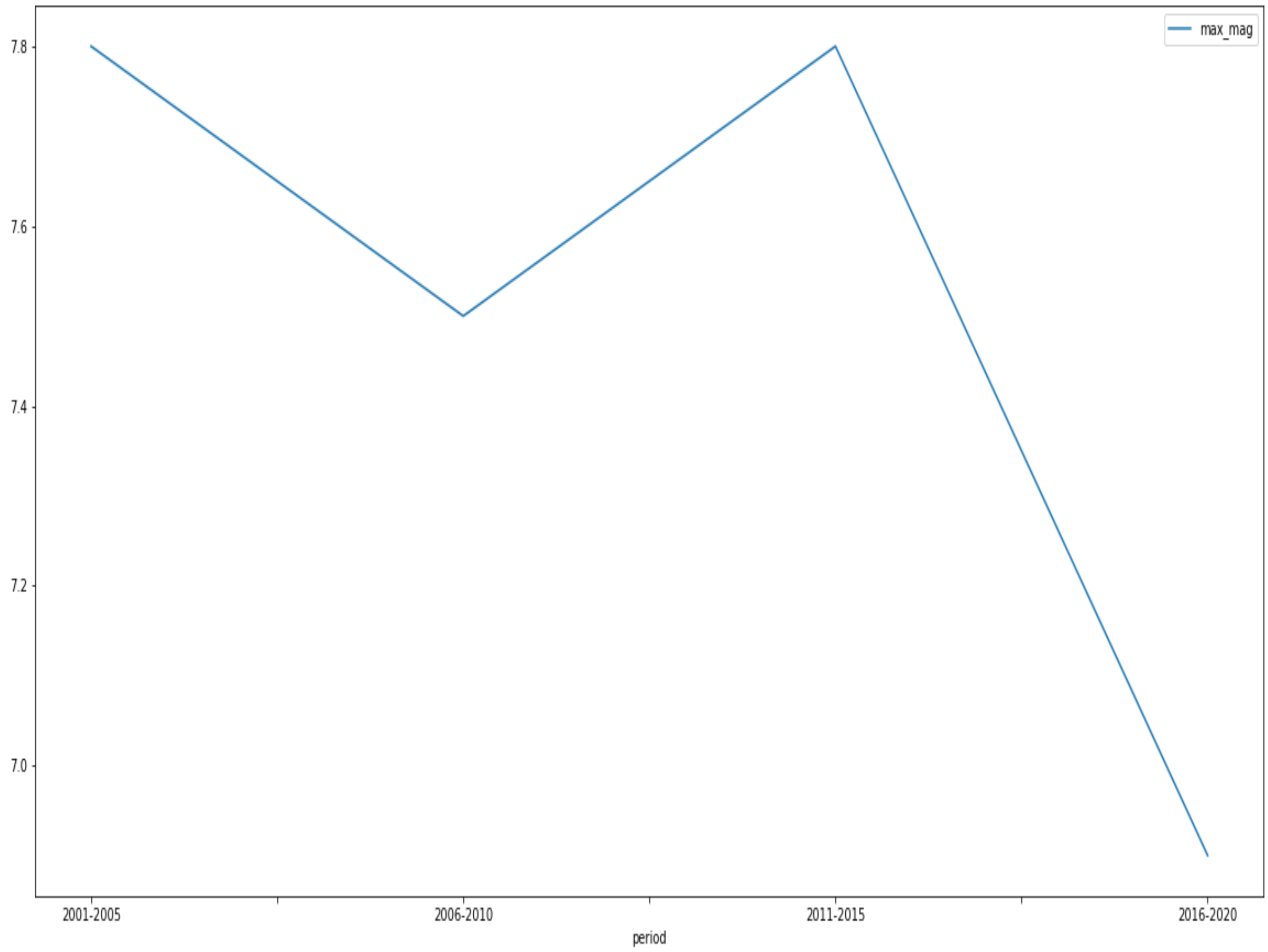
<AxesSubplot:xlabel='period'>



IN:-

```
df.plot(x='period',y='max_mag',figsize=(20,10))
```

OUT:-



IN:-

```
df.to_csv(r'quakes_toTrain.csv', index = None, header=True)
```

IN:-

```
import numpy as np
import pandas as pd
from pandas import read_csv
from matplotlib import pyplot
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
from statsmodels.tsa.vector_ar.vecm import coint_johansen
```

from statsmodels.tsa.vector_ar.var_model import VAR

from sklearn.metrics import mean_squared_error

get_ipython().run_line_magic('matplotlib', 'inline')

IN:-

df1 = pd.read_csv('C:\\New folder\\quakes_toTrain.csv')

df1.dtypes

OUT:-

```
period       object
T            int64
mean_mag    float64
Speed       float64
b           float64
niu         float64
delta_M     float64
max_mag     float64
dtype: object
```

IN:-

```
temp=[]
for i in range(len(df1.period)):
    temp += [pd.to_datetime(df1.period[i][:4] , format = '%Y')]
df1['period'] = temp;
data = df1.drop(['period'], axis=1)
data.index = df1.period
data.head()
```

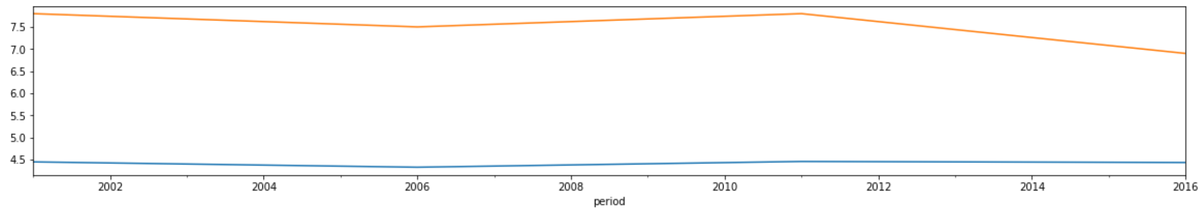
OUT:-

	T	mean_mag	Speed	b	niu	delta_M	max_mag
period							
2001-01-01	1823	4.446800	9.769447e+09	0.814042	0.358947	0.621844	7.8
2006-01-01	1823	4.325855	5.429726e+09	0.807539	0.351617	0.600057	7.5
2011-01-01	1822	4.456482	4.528406e+09	1.042164	0.319705	0.344146	7.8
2016-01-01	730	4.432896	3.130998e+09	1.102924	0.320471	0.291965	6.9

IN:-

```
plt.rcParams["figure.figsize"] = (20,3)
data.mean_mag.plot()
data.max_mag.plot()
pyplot.show()
```

OUT:-

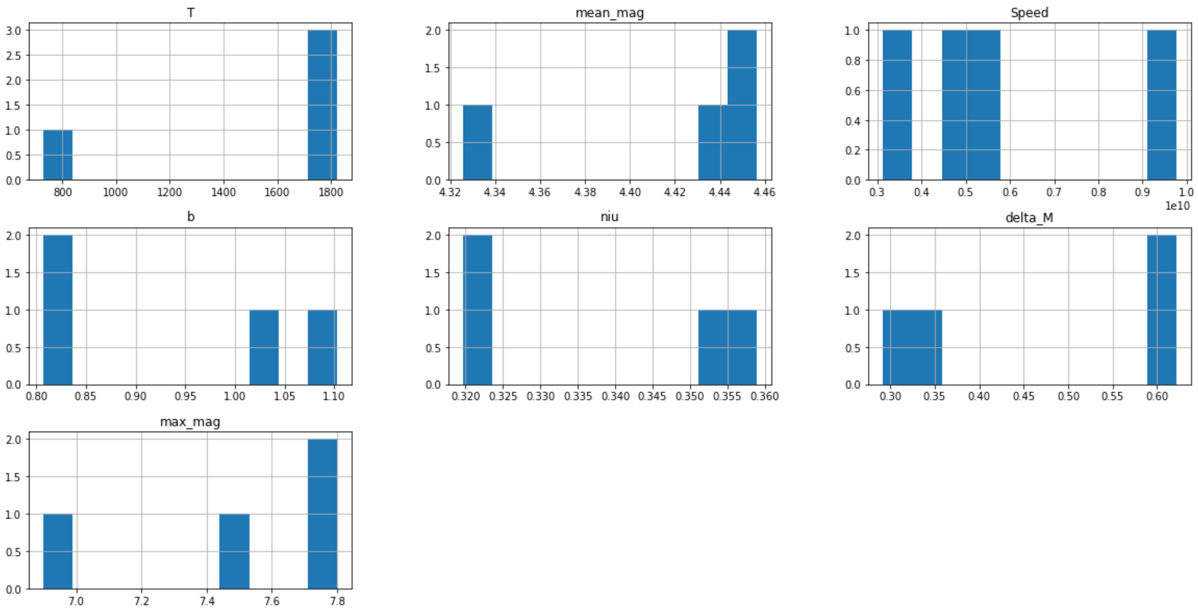


IN:-

```
df1.hist(figsize=(20,10))
```

OUT:-

```
array([[<AxesSubplot:title={'center':'T'}>,
        <AxesSubplot:title={'center':'mean_mag'}>,
        <AxesSubplot:title={'center':'Speed'}>],
       [<AxesSubplot:title={'center':'b'}>,
        <AxesSubplot:title={'center':'niu'}>,
        <AxesSubplot:title={'center':'delta_M'}>],
       [<AxesSubplot:title={'center':'max_mag'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



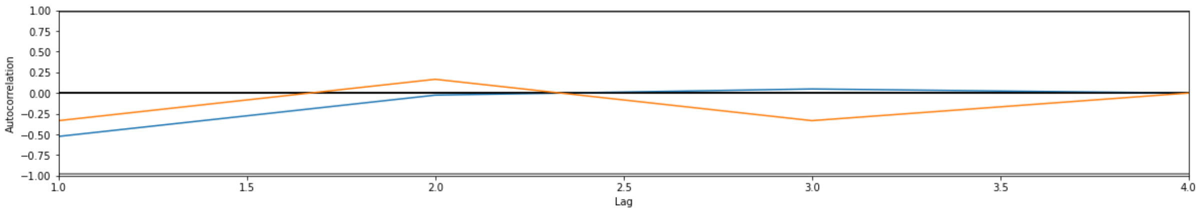
IN:-

autocorrelation_plot(data.mean_mag)

autocorrelation_plot(data.max_mag)

pyplot.show()

OUT:-



IN:-

data.corr()

OUT:-

	T	mean_mag	Speed	b	niu	delta_M	max_mag
T	1.000000	-0.191919	0.601874	-0.702484	0.559397	0.674593	0.942665
mean_mag	-0.191919	1.000000	0.113221	0.537452	-0.434119	-0.489042	0.145944
Speed	0.601874	0.113221	1.000000	-0.771395	0.838900	0.810124	0.661371
b	-0.702484	0.537452	-0.771395	1.000000	-0.971015	-0.996891	-0.538043
niu	0.559397	-0.434119	0.838900	-0.971015	1.000000	0.985815	0.432716
delta_M	0.674593	-0.489042	0.810124	-0.996891	0.985815	1.000000	0.527636
max_mag	0.942665	0.145944	0.661371	-0.538043	0.432716	0.527636	1.000000

IN:-

```
import requests
import csv
from csv import DictReader
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sb
# below lines are important when you get KeyError: 'PROJ_LIB'
import os
```

IN:-

```
address = 'C:\\New folder\\4.5_month.csv'
eq = pd.read_csv(address)
eq.head()
```

OUT:-

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type	horizontalError	depthE
0	2023-02-26T05:52:43.397Z	1.7814	125.8377	8.842	4.8	mb	71.0	73.0	1.828	0.66	...	2023-02-26T07:30:41.819Z	101 km ENE of Laikit, Laikit II (Dimembe), Ind...	earthquake	4.75	3.
1	2023-02-26T05:13:13.523Z	-11.3973	165.7114	10.000	4.8	mb	36.0	139.0	4.279	0.63	...	2023-02-26T05:32:45.040Z	74 km S of Lata, Solomon Islands	earthquake	12.87	1.
2	2023-02-26T03:53:02.135Z	5.5632	126.4766	10.000	4.8	mb	22.0	121.0	1.742	0.70	...	2023-02-26T04:11:03.040Z	94 km SSE of Pondaguitan, Philippines	earthquake	6.45	1.
3	2023-02-26T03:21:19.743Z	-11.2993	165.7764	10.000	4.9	mb	24.0	117.0	4.350	0.57	...	2023-02-26T03:39:31.040Z	Santa Cruz Islands	earthquake	10.65	1.
4	2023-02-25T23:32:47.291Z	-0.0602	124.6411	11.109	4.9	mb	57.0	47.0	2.108	0.84	...	2023-02-26T00:02:25.040Z	Molucca Sea	earthquake	5.59	4.

5 rows × 22 columns

IN:-

```
len(eq)
```

OUT:-

571

IN:-

```
freq = eq['mag'].value_counts()
freq
```

OUT:-

```
4.50    127
4.60    110
4.70     69
4.90     56
5.00     53
4.80     53
5.10     22
5.30     17
5.20     14
5.40     11
5.50      9
5.70      5
5.60      5
6.00      4
6.10      3
5.90      3
6.30      2
5.80      2
6.20      1
```

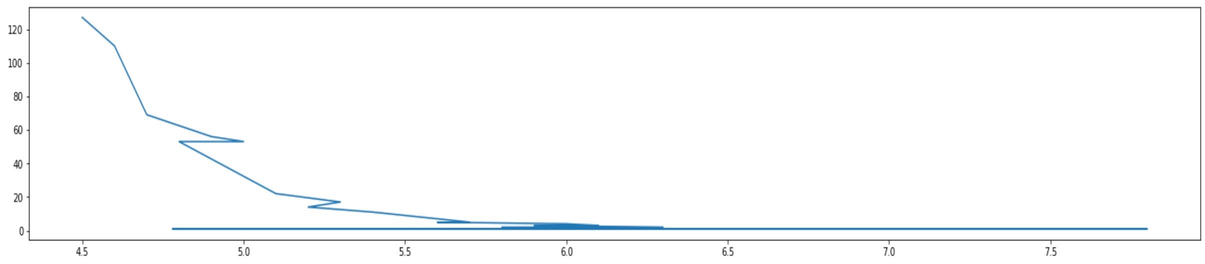
7.80   1
7.50   1
4.78   1
6.70   1
6.80   1
Name: mag, dtype: int64

IN:-

```
fig = plt.figure()

ax = fig.add_axes([.1, .1, 1, 1])

ax.plot(freq)
```

OUT:-

[<matplotlib.lines.Line2D at 0x2636976f8b0>]

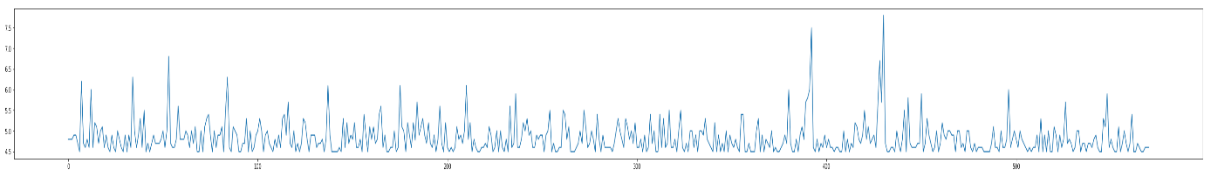


IN:-

```
fig = plt.figure()

ax = fig.add_axes([.1, .1, 2, 1])

ax.plot(eq['mag'])
```

OUT:-

[<matplotlib.lines.Line2D at 0x263659b0f70>]



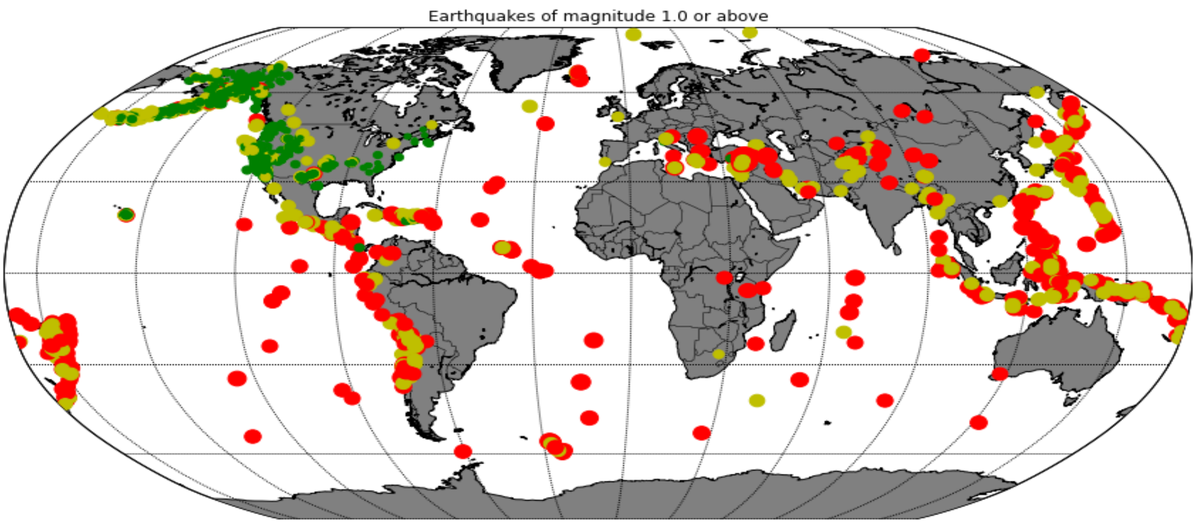
IN:-

```
import pandas as pd
```

```
from mpl_toolkits.basemap import Basemap
eq_lat, eq_lon = [], []
magnitudes = []
eq_ts = []
reader = 'C:\\New folder\\1.0_month.csv'
eq1 = pd.read_csv(reader)
eq1.head()
def mk_color(magnitude):
    # red color for significant earthquakes, yellow for earthquakes below 4.5 and
above 3.0
    #  and green for earthquakes below 3.0
    if magnitude < 3.0:
        return ('go')
    elif magnitude < 4.5:
        return ('yo')
    else:
        return ('ro')
plt.figure(figsize=(15,11))
my_map = Basemap(projection='robin', resolution = 'l', area_thresh = 1000.0,
lat_0=0, lon_0=-10)
my_map.drawcoastlines()
my_map.drawcountries()
my_map.fillcontinents(color = 'grey')
my_map.drawmapboundary()
my_map.drawmeridians(np.arange(0, 360, 30))
my_map.drawparallels(np.arange(-90, 90, 30))
mk_size = 2.4
eq_lat = eq1["latitude"].tolist()
```

```
eq_lon = eq1["longitude"].tolist()
magnitudes = eq1["mag"].tolist()
print("Latitude values:", eq_lat)
print("Longitude values:", eq_lon)
print("Magnitude values:", magnitudes)

for lon, lat, mag in zip(eq_lon, eq_lat, magnitudes):
    x,y = my_map(lon, lat)
    msize = mag * mk_size
    marker_string = mk_color(mag)
    my_map.plot(x, y, marker_string, markersize=msize)
plt.title('Earthquakes of magnitude 1.0 or above')
# we can save the image as png file locally to the directory we are working in
plt.savefig('eq_data.png')
plt.show()
```

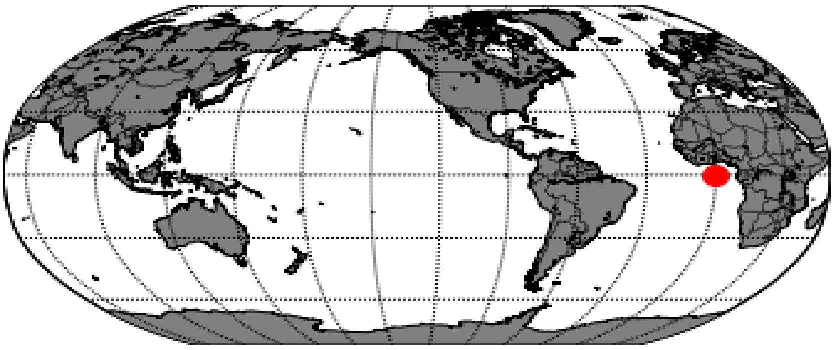
OUT:-



IN:-

```
import csv
# Open the earthquake data file.
filename = r'C:\\New folder\\2.5_month.csv'
# Create empty lists for the latitudes and longitudes.
lats, lons = [0], [0]



# --- Build Map ---
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
eq_map = Basemap(projection='robin', resolution = 'l', area_thresh = 1000.0,
            lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'grey')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
eq_map.drawparallels(np.arange(-90, 90, 30))
x,y = eq_map(lons, lats)
eq_map.plot(x, y, 'ro', markersize=10)
plt.show()
```

OUT:-

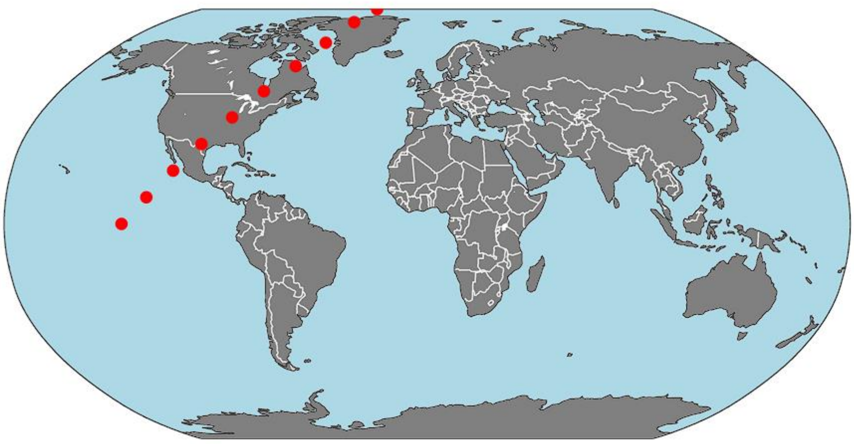


IN:-

```
#create a map of the world centered on the Pacific Ocean and plot red circles at the specified latitude and longitude coordinates

import plotly.graph_objs as go

# Create lists of latitude and longitude values.

lats = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

lons = [-130, -120, -110, -100, -90, -80, -70, -60, -50, -40]

# Create a scattergeo trace for the latitude and longitude data.

trace = go.Scattergeo(

    lat=lats,

    lon=lons,

    mode='markers',

    marker=dict(

        size=10,

        color='red',

        symbol='circle'

    )

)

# Create a layout for the map.

layout = go.Layout(
```

```
    geo=dict(
        projection=dict(type='robinson'),
        showland=True,
        showcountries=True,
        landcolor='grey',
        countrycolor='white',
        showocean=True,
        oceancolor='lightblue'
    )
)
# Create a figure with the trace and layout, and display the map.
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```

OUT:-

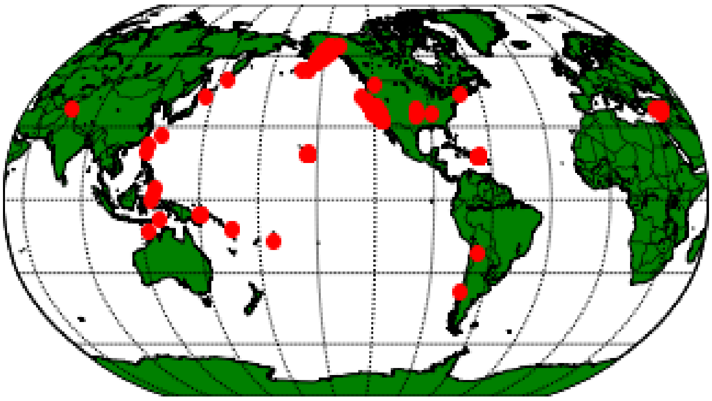


IN:-

```
import csv
# Open the earthquake data file.
filename = 'C:\\New folder\\all_day.csv'
# Create empty lists for the latitudes and longitudes.
lats, lons = [], []
# Read through the entire file, skip the first line,
#  and pull out just the lats and lons.
with open(filename, encoding="utf8") as f:
    # Create a csv reader object.
    reader = csv.reader(f)
    # Ignore the header row.
    next(reader)
    # Store the latitudes and longitudes in the appropriate lists.
    for row in reader:
        lats.append(float(row[1]))
        lons.append(float(row[2]))
# --- Build Map ---
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
eq_map = Basemap(projection='robin', resolution = 'l', area_thresh = 1000.0,
              lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'green')
eq_map.drawmapboundary()
```

```
eq_map.drawmeridians(np.arange(0, 360, 30))
eq_map.drawparallels(np.arange(-90, 90, 30))
x,y = eq_map(lons, lats)
eq_map.plot(x, y, 'ro', markersize=6)
plt.show()
```

OUT:-



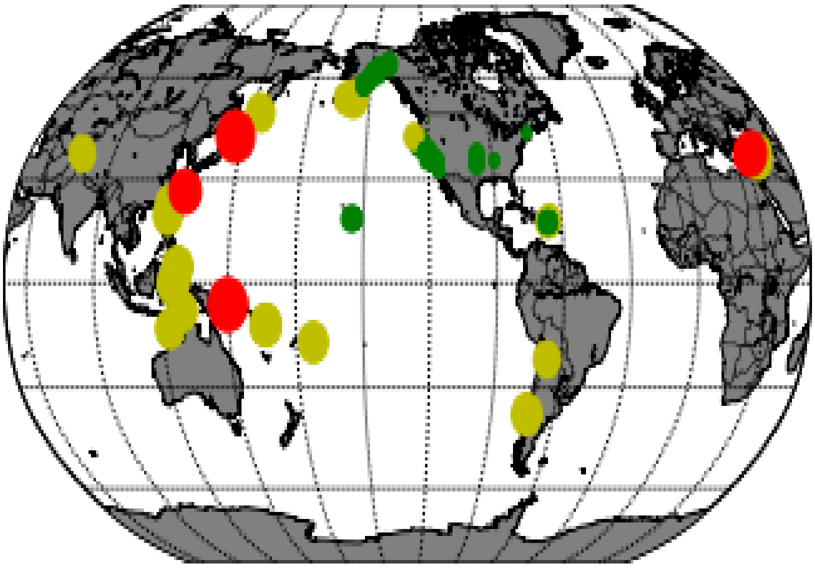
IN:-

```
import csv
# Open the earthquake data file.
filename = 'C:\\New folder\\all_day.csv'
# Create empty lists for the data we are interested in.
lats, lons = [], []
magnitudes = []
# Read through the entire file, skip the first line,
#  and pull out just the lats and lons.
with open(filename,encoding="utf8") as f:
    # Create a csv reader object.
    reader = csv.reader(f)
```

```
    # Ignore the header row.
    next(reader)
    # Store the latitudes and longitudes in the appropriate lists.
    for row in reader:
        lats.append(float(row[1]))
        lons.append(float(row[2]))
        magnitudes.append(float(row[4]))
# --- Build Map ---
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
def get_marker_color(magnitude):
    # Returns green for small earthquakes, yellow for moderate
    #  earthquakes, and red for significant earthquakes.
    if magnitude < 3.0:
        return ('go')
    elif magnitude < 5.0:
        return ('yo')
    else:
        return ('ro')
eq_map = Basemap(projection='robin', resolution = 'l', area_thresh = 1000.0,
              lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'gray')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
```

```
eq_map.drawparallels(np.arange(-90, 90, 30))
min_marker_size = 2.5
for lon, lat, mag in zip(lons, lats, magnitudes):
    x,y = eq_map(lon, lat)
    msize = mag * min_marker_size
    marker_string = get_marker_color(mag)
    eq_map.plot(x, y, marker_string, markersize=msize)
plt.show()
```

OUT:-

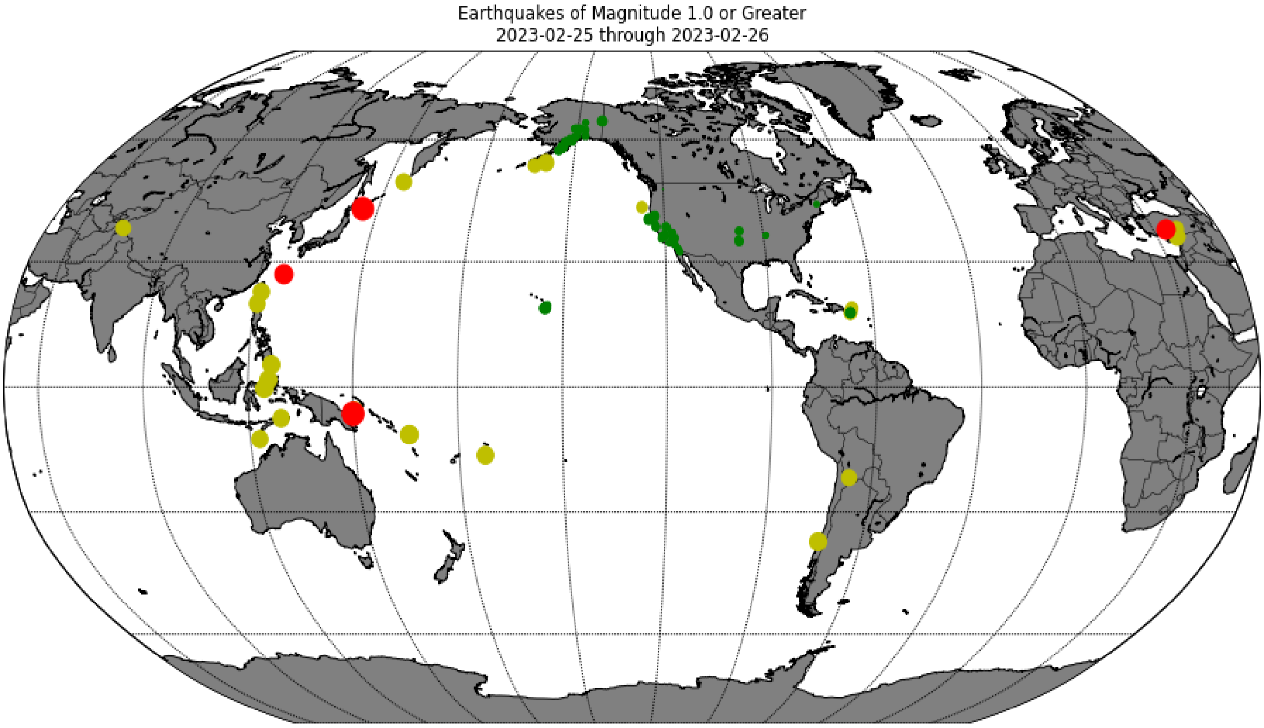


IN:-

```
import csv
# Open the earthquake data file.
filename = 'C:\\New folder\\all_day.csv'
# Create empty lists for the data we are interested in.
lats, lons = [], []
magnitudes = []
timestrings = []
# Read through the entire file, skip the first line,
#  and pull out just the lats and lons.
with open(filename,encoding="utf-8") as f:
    # Create a csv reader object.
    reader = csv.reader(f)
    # Ignore the header row.
    next(reader)
    # Store the latitudes and longitudes in the appropriate lists.
    for row in reader:
        lats.append(float(row[1]))
        lons.append(float(row[2]))
        magnitudes.append(float(row[4]))
        timestrings.append(row[0])
# --- Build Map ---
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
def get_marker_color(magnitude):
    # Returns green for small earthquakes, yellow for moderate
```

```
    #  earthquakes, and red for significant earthquakes.
    if magnitude < 3.0:
        return ('go')
    elif magnitude < 5.0:
        return ('yo')
    else:
        return ('ro')
# Make this plot larger.
plt.figure(figsize=(16,12))
eq_map = Basemap(projection='robin', resolution = 'l', area_thresh = 1000.0,
              lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'gray')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
eq_map.drawparallels(np.arange(-90, 90, 30))
min_marker_size = 2.5
for lon, lat, mag in zip(lons, lats, magnitudes):
    x,y = eq_map(lon, lat)
    msize = mag * min_marker_size
    marker_string = get_marker_color(mag)
    eq_map.plot(x, y, marker_string, markersize=msize)
title_string = "Earthquakes of Magnitude 1.0 or Greater\n"
title_string += "%s through %s" % (timestrings[-1][:10], timestrings[0][:10])
plt.title(title_string)
plt.show()
```

OUT:-



IN:-

import pandas as pd

import matplotlib.pyplot as plt

IN:-

Load the dataset

df = pd.read_csv('C:\\New folder\\Disaster\\Natural_Disasters_in_India .csv')

IN:-

Display the first few rows of the dataset

print(df.head(5))

OUT:-

```
Unnamed: 0                      Title     Duration  Year  \
0          0  1990 Andhra Pradesh cyclone        4 May  1990
1          1   Indian Airlines Flight 605  14 February  1990
2          3   1991 Uttarkashi earthquake   20 October  1991
```

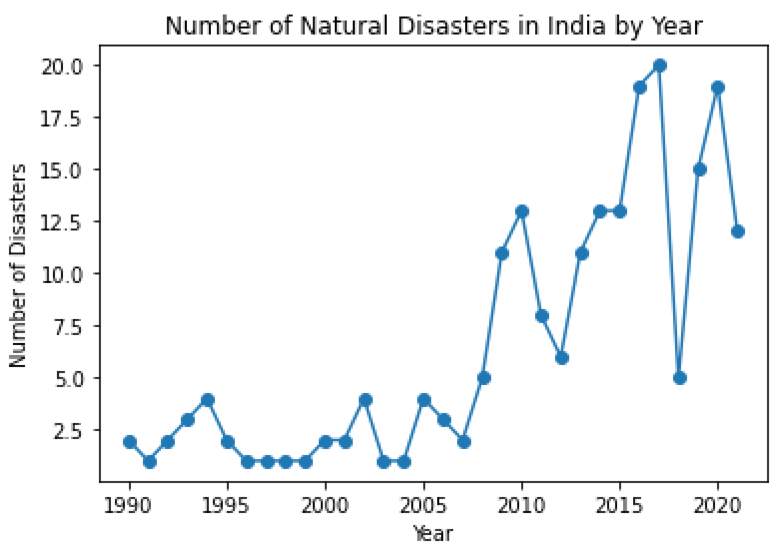
3 4 1992 India–Pakistan floods 7 September 1992
4 5 Mahamaham stampede 18 February 1992

	Disaster_Info	Date
0	the andhra pradesh cyclone or the machilipat...	1990-05-04
1	indian airlines flight was a scheduled domest...	1990-02-14
2	the uttarkashi earthquake also known as the g...	1991-10-20
3	the india–pakistan floods was a deadliest flo...	1992-09-07
4	mahamaham stampede was a disaster that occurre...	1992-02-18

IN:-

```
%matplotlib inline

import matplotlib.pyplot as plt

# Plot the number of disasters by year

disasters_by_year = df.groupby('Year').size()

plt.plot(disasters_by_year.index, disasters_by_year.values, marker='o')

plt.xlabel('Year')

plt.ylabel('Number of Disasters')

plt.title('Number of Natural Disasters in India by Year')

plt. show()
```

OUT:-



IN:-

```
import pandas as pd
import plotly.express as px
```

IN:-

```
# Show the first five rows of the dataset
print(df.head(5))


# Show the basic statistics of the dataset
print(df.describe())
# Show the number of unique values in each column of the dataset
print(df.nunique())
```

OUT:-

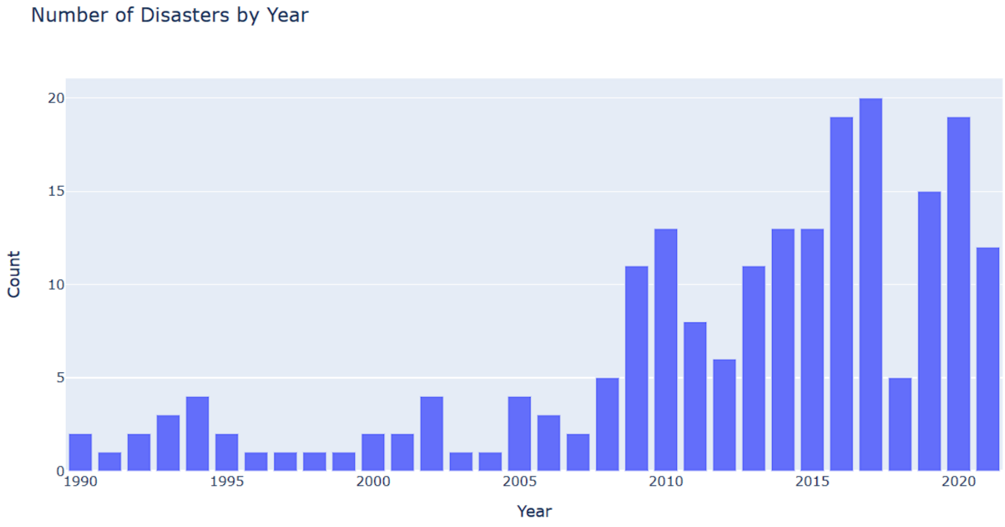
```
   Unnamed: 0                        Title     Duration  Year  \
0           0  1990 Andhra Pradesh cyclone        4 May  1990
1           1   Indian Airlines Flight 605  14 February  1990
2           3   1991 Uttarkashi earthquake   20 October  1991
3           4   1992 India–Pakistan floods  7 September  1992
4           5            Mahamaham stampede  18 February  1992

                                        Disaster_Info        Date
0  the  andhra pradesh cyclone or the  machilipat...  1990-05-04
1  indian airlines flight  was a scheduled domest...  1990-02-14
2  the  uttarkashi earthquake also known as the g...  1991-10-20
3  the  india–pakistan floods was a deadliest flo...  1992-09-07
4  mahamaham stampede was a disaster that occurre...  1992-02-18
       Unnamed: 0         Year
count  207.000000   207.000000
mean   108.661836  2012.362319
std     61.442069     7.549136
min      0.000000  1990.000000
25%     56.500000  2009.000000
50%    109.000000  2014.000000
75%    161.500000  2017.000000
max    213.000000  2021.000000
Unnamed: 0       207
Title            195
```

```
Duration        174
Year             32
Disaster_Info   205
Date            197
dtype: int64
```

IN:-

```
# Create a bar chart of the number of disasters by year

disasters_by_year = df.groupby("Year").size().reset_index(name="Count")

fig = px.bar(disasters_by_year, x="Year", y="Count", title="Number of
Disasters by Year")

fig.show()
```

OUT:-

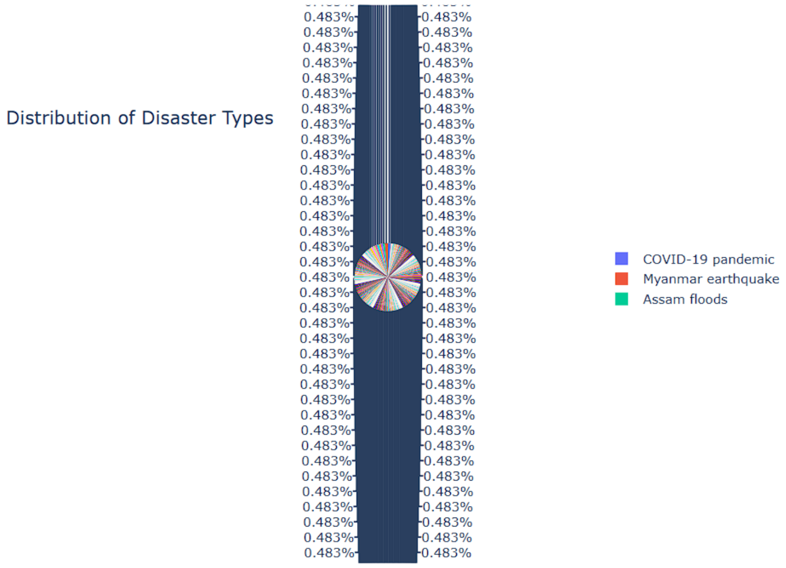


IN:-

```
# Create a pie chart of the distribution of disaster types
```

```
disaster_types = df.groupby("Title").size().reset_index(name="Count")

fig = px.pie(disaster_types, values="Count", names="Title", title="Distribution of Disaster Types")

fig.show()
```

OUT:-



IN:-

```
import pandas as pd

# read CSV file into a Pandas dataframe

df = pd.read_csv('C:\\New folder\\significant\\Significant_Earthquakes.csv')

# print first five rows of the dataframe

print(df.head())
```

OUT:-

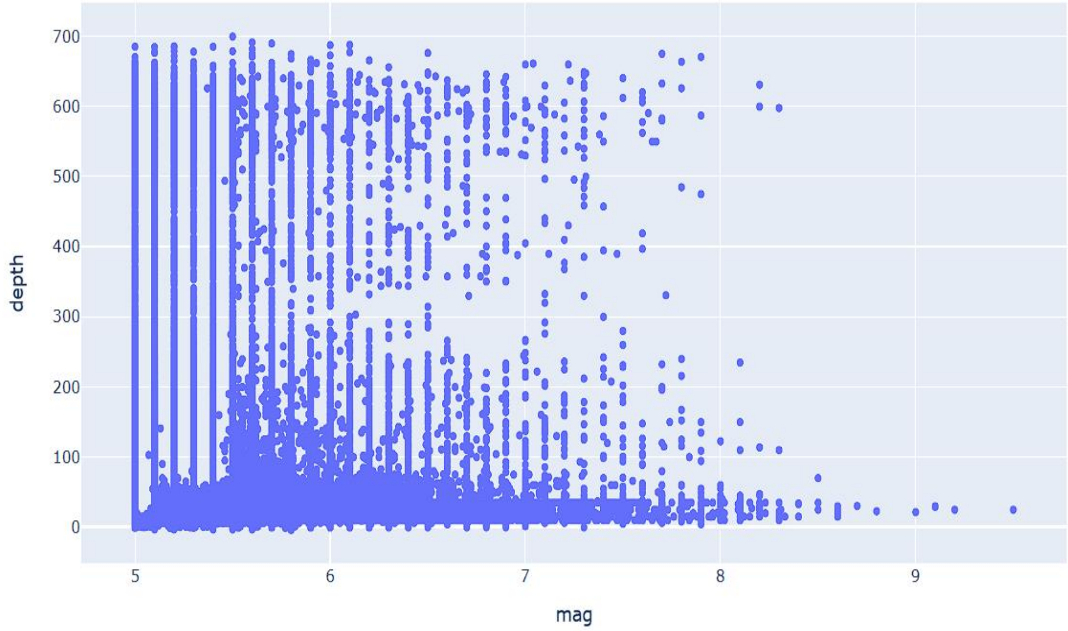
```
   Unnamed: 0                      time  latitude  longitude  depth   mag  \
0           0  1900-10-09T12:25:00.000Z     57.09    -153.48    NaN  7.86
1           1  1901-03-03T07:45:00.000Z     36.00    -120.50    NaN  6.40
2           2  1901-07-26T22:20:00.000Z     40.80    -115.70    NaN  5.00
```

```
3          3  1901-12-30T22:34:00.000Z     52.00    -160.00    NaN  7.00
4          4  1902-01-01T05:20:30.000Z     52.38    -167.45    NaN  7.00

  magType  nst  gap  dmin  ...                  updated  \
0      mw  NaN  NaN   NaN  ...  2022-05-09T14:44:17.838Z
1      ms  NaN  NaN   NaN  ...  2018-06-04T20:43:44.000Z
2      fa  NaN  NaN   NaN  ...  2018-06-04T20:43:44.000Z
3      ms  NaN  NaN   NaN  ...  2018-06-04T20:43:44.000Z
4      ms  NaN  NaN   NaN  ...  2018-06-04T20:43:44.000Z

                                place        type horizontalError depthError  \
0      16 km SW of Old Harbor, Alaska  earthquake             NaN        NaN
1  12 km NNW of Parkfield, California  earthquake             NaN        NaN
2            6 km SE of Elko, Nevada  earthquake             NaN        NaN
3                    south of Alaska  earthquake             NaN        NaN
4      113 km ESE of Nikolski, Alaska  earthquake             NaN        NaN

  magError  magNst    status  locationSource  magSource
0      NaN     NaN  reviewed           ushis         pt
1      NaN     NaN  reviewed           ushis        ell
2      NaN     NaN  reviewed           ushis        sjg
3      NaN     NaN  reviewed           ushis        abe
4      NaN     NaN  reviewed           ushis        abe

[5 rows x 23 columns]
```

IN:-

```
import pandas as pd

import plotly.express as px

# create a scatter plot of magnitude vs depth

fig = px.scatter(df, x='mag', y='depth', hover_name='place')

# show the plot

fig.show()
```

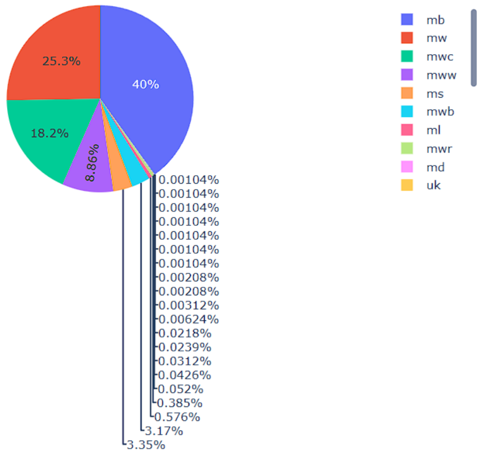
OUT:-



IN:-

```
# create a pie chart of earthquake magnitudes
fig = px.pie(df, names='magType')
# show the pie chart
fig.show()
```

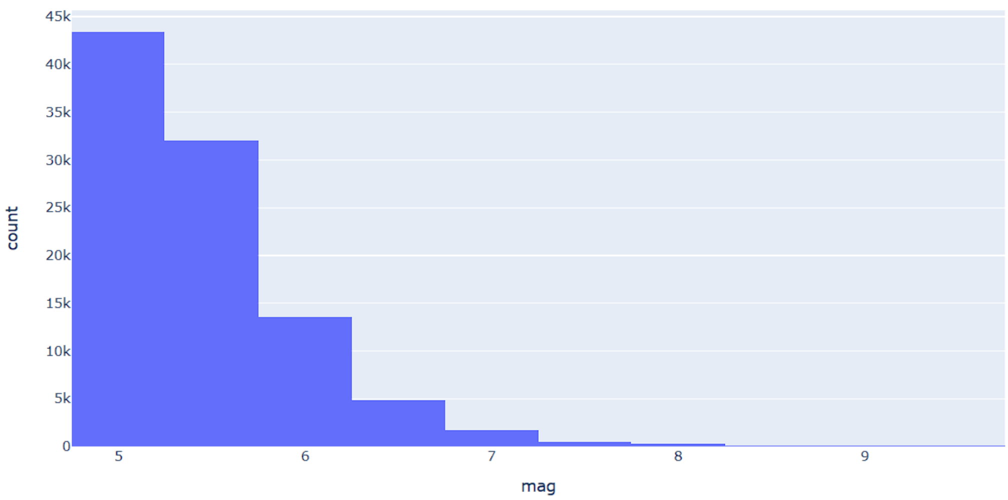
OUT:-



IN:-

```
# create a histogram of earthquake magnitudes
fig = px.histogram(df, x='mag', nbins=20)
# show the histogram
fig.show()
```

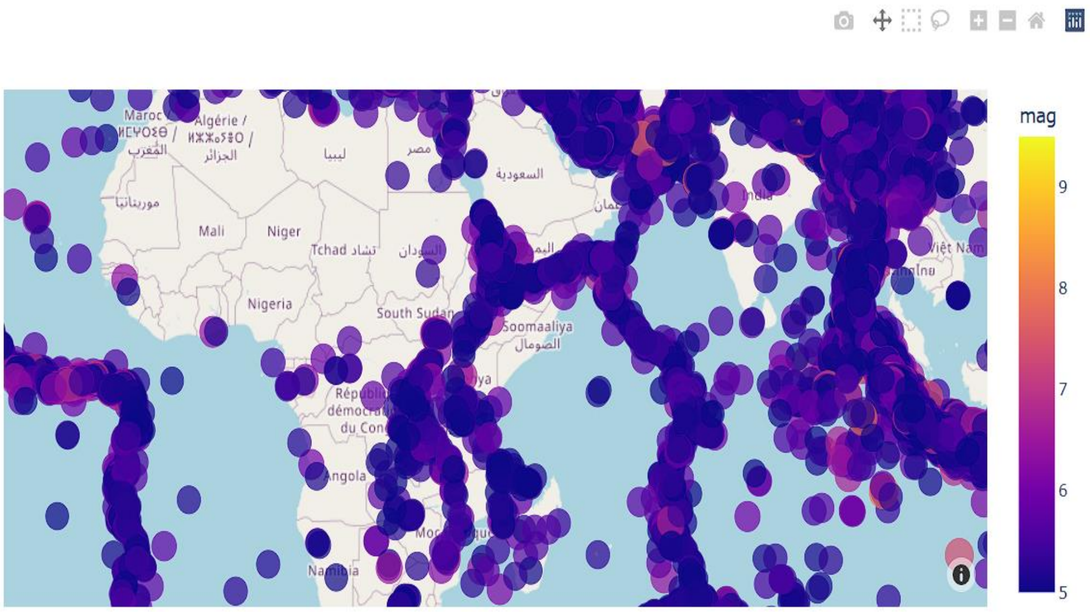
OUT:-



IN:-

```
# create a scatter mapbox plot of earthquake locations
fig = px.scatter_mapbox(df, lat='latitude', lon='longitude', color='mag',
size='mag', hover_name='place', zoom=2, height=500)
fig.update_layout(mapbox_style='open-street-map')
fig.show()
```

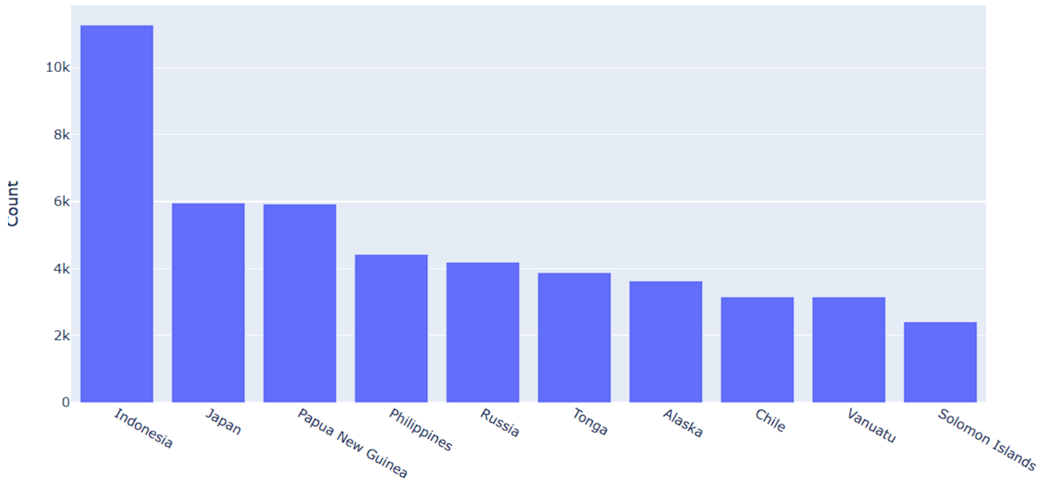
OUT:-



IN:-

```
# create a bar chart of earthquake counts by country
country_counts = df['place'].str.split(', ').str[-1].value_counts().reset_index()
country_counts.columns = ['Country', 'Count']
fig = px.bar(country_counts.head(10), x='Country', y='Count')
fig.show()
```

OUT:-



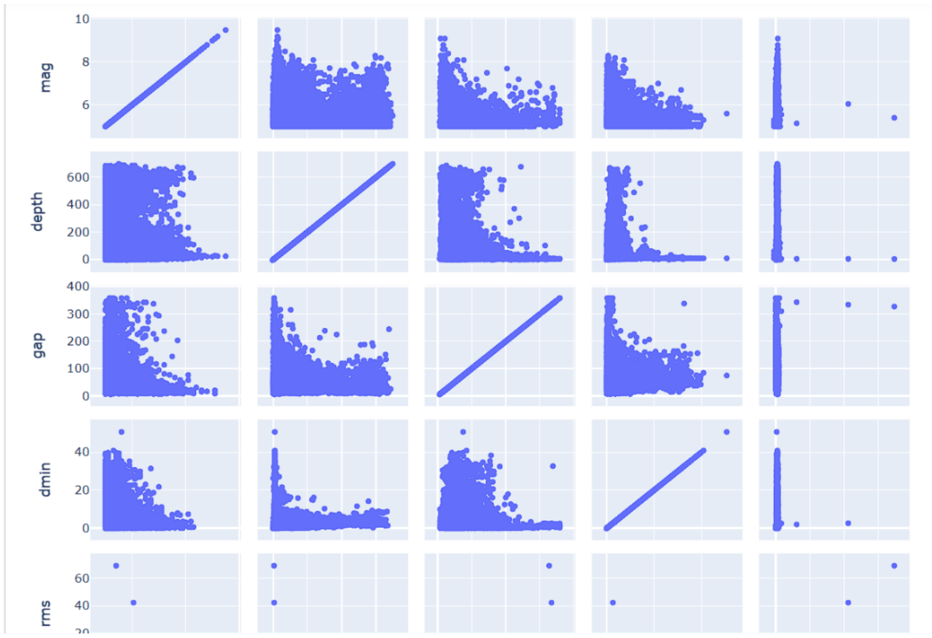
IN:-

create a scatter matrix of earthquake features

fig = px.scatter_matrix(df[['mag', 'depth', 'gap', 'dmin', 'rms']], height=800)

fig.show()

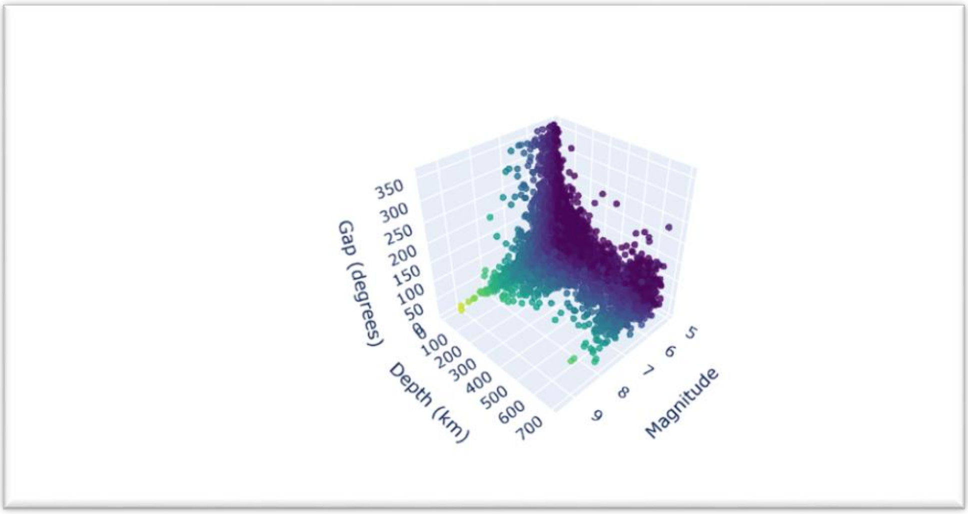
OUT:-



IN:-

```
# create a 3D scatter plot of earthquake magnitudes, depths, and gaps
fig = go.Figure(data=[go.Scatter3d(
    x=df['mag'],
    y=df['depth'],
    z=df['gap'],
    mode='markers',
    marker=dict(
        size=3,
        color=df['mag'],
        colorscale='Viridis',
        opacity=0.8
    ),
    text=df['place'],
    hovertemplate='Magnitude: %{x:.2f}<br>Depth: %{y:.2f} km<br>Gap:
%{z:.2f} degrees<extra>%{text}</extra>'
)])
fig.update_layout(scene=dict(
    xaxis_title='Magnitude',
    yaxis_title='Depth (km)',
    zaxis_title='Gap (degrees)',
    aspectmode='cube'
))
fig.show()
```

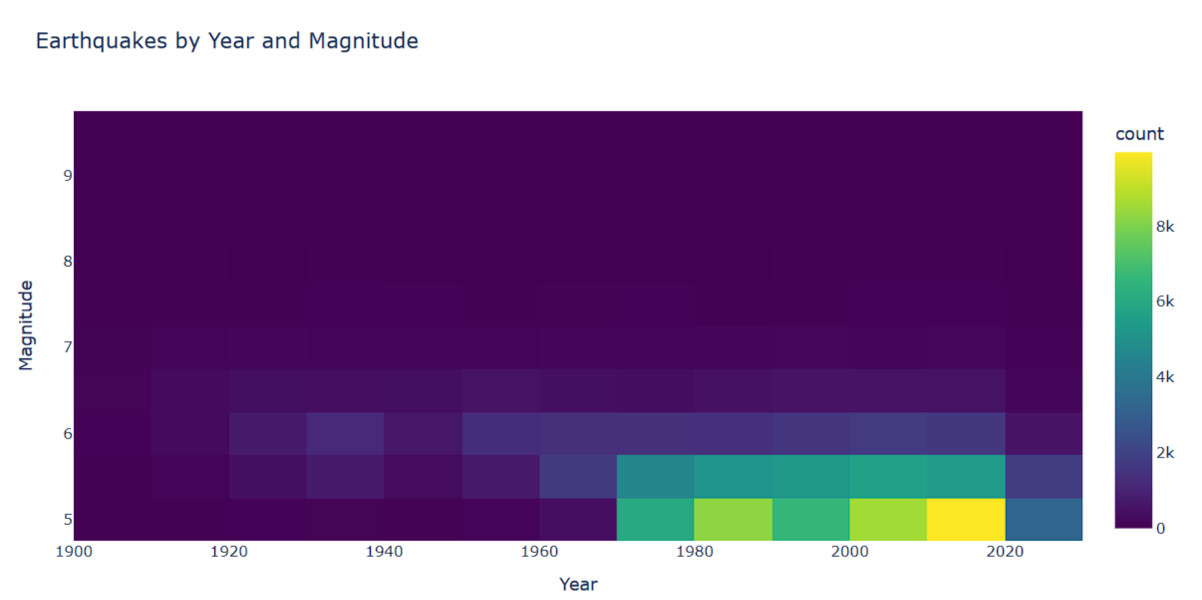
OUT:-



IN:-

```
# create a heatmap of the number of earthquakes by year and magnitude
fig = px.density_heatmap(df, x='time', y='mag', nbinsx=20, nbinsy=20,
                         color_continuous_scale='viridis', title='Earthquakes by Year and Magnitude')
fig.update_layout(xaxis_title='Year', yaxis_title='Magnitude')
fig.show()
```

OUT:-



IN:-

```
import plotly.graph_objs as go
import pandas as pd
# Load the earthquake data
data = pd.read_csv('C:\\New folder\\all_month.csv')
# Create a scatter plot on a world map
fig = go.Figure(data=go.Scattergeo(
    lon=data['longitude'],
    lat=data['latitude'],
    marker=dict(size=3),
))

# Update the layout to show the whole circle as Earth
fig.update_geos(
```

```
    projection_type="orthographic",
    landcolor="white",
    oceancolor="MidnightBlue",
    showocean=True,
    lakecolor="LightBlue",
    showlakes=True,
    showland=True,
    bgcolor='black'
)
# Set the title of the plot
fig.update_layout(title_text='Earthquakes in the last month')
# Show the plot
fig.show()
```

OUT:-

CONCLUSION

In conclusion, the analysis of cyclones and earthquakes using graphs and visualizations provides valuable insights into the patterns and characteristics of these natural phenomena. By visually representing data, trends, and relationships, graphs help in understanding and communicating complex information effectively.In cyclone analysis, graphical representations such as time series plots, bar charts, and maps can depict the frequency and intensity of cyclones over time and in different regions. These visualizations help identify cyclone-prone areas, track their movement, and assess the impact on coastal regions. Graphical representations also aid in comparing cyclone events, identifying trends, and detecting anomalies.For earthquake analysis, graphs such as magnitude-time plots, frequency-magnitude distributions, and seismicity maps are commonly used. These visuals provide a comprehensive overview of earthquake activity, including the frequency, magnitude, and spatial distribution of seismic events. Graphs can reveal patterns of seismic activity, identify clusters of earthquakes, and assess the likelihood of future events.Graphs and visualizations play a vital role in communicating the analysis results to stakeholders, policymakers, and the general public. They help in raising awareness about the risks associated with cyclones and earthquakes, supporting decision-making for disaster preparedness, and facilitating effective communication of information to affected communities. By presenting data in a clear and intuitive manner, graphs enhance understanding, facilitate interpretation, and promote informed actions to mitigate the impact of cyclones and earthquakes.

Overall, the use of graphs and visualizations in cyclone and earthquake analysis enhances the effectiveness and accessibility of the findings, contributing to improved preparedness, response, and resilience in the face of these natural disasters.

FUTURE ENHANCEMENT

Future enhancements in cyclone and earthquake analysis can further improve our understanding and predictive capabilities of these natural phenomena. Here are some potential areas for future development:

1. ***Advanced Data Integration***: Integrating multiple data sources, such as satellite imagery, remote sensing data, and social media feeds, can provide a more comprehensive and real-time view of cyclones and earthquakes. Incorporating data from diverse sources can enhance the accuracy and timeliness of analysis, enabling more effective early warning systems and response strategies.
2. ***Machine Learning and Artificial Intelligence***: Applying machine learning algorithms and AI techniques can help identify patterns, correlations, and precursors in cyclone and earthquake data. By leveraging these technologies, we can develop more accurate prediction models and improve our understanding of the complex dynamics associated with these natural disasters.
3. ***Ensemble Modeling***: Utilizing ensemble modeling techniques, which involve combining multiple forecasting models, can enhance the reliability and robustness of cyclone and earthquake predictions. Ensemble models consider different algorithms, data sources, and approaches, providing a more comprehensive and reliable forecast.
4. ***High-resolution and 3D Mapping***: Improving the resolution and detail of mapping techniques can enhance our understanding of the geographical and geological factors contributing to cyclones and earthquakes. High-resolution mapping, including 3D mapping technologies, can help identify vulnerable areas, fault lines, and other geological features critical to predicting and mitigating the impact of these events.
5. ***Climate Change Analysis***: Considering the influence of climate change on cyclones and earthquakes is crucial for future analysis. Studying the correlation between climate change indicators, such as sea surface temperatures, atmospheric conditions, and seismic activity, can help us understand the evolving patterns and impacts of these natural disasters in a changing climate.
6. ***Risk Assessment and Vulnerability Mapping***.

REFERENCE

1. https://www.kaggle.com/code/udayanguha/indian-subcontinent-earthquakes/notebook
2. https://www.kaggle.com/code/kerneler/starter-earthquakes-and-cyclones-in-70a05863-4
3. https://www.kaggle.com/code/rohithmahadevan/predicting-india-s-seismic-activity-using-ts
4. https://www.kaggle.com/code/mehedeehassan/do-small-earthquake-leads-to-bigger-earthquake
5. https://www.codespeedy.com/analyze-and-visualize-earthquake-data-in-python-with-matplotlib/

Thank You