INTRODUCTION

The increasing frequency and severity of extreme weather events, including variations in rainfall patterns, have significant implications for various sectors such as agriculture, water resource management, and disaster preparedness. To address these challenges, this project focuses on harnessing the power of big data analytics to improve rainfall prediction and analysis. Through the integration of extensive datasets and advanced analytics techniques, our research aims to enhance our understanding of rainfall patterns, enabling more accurate predictions and proactive decision-making in response to changing weather conditions.

Rainfall data is a valuable resource for understanding and managing water resources, agriculture, and disaster risk. However, rainfall data is often complex and difficult to analyze, due to its high volume, velocity, and variety. Big data analytics can be used to overcome these challenges and extract valuable insights from rainfall data.

ABSTRACT

Big data analytics is a powerful tool for extracting insights from large and complex datasets. This project will use big data analytics to analyze rainfall data from a variety of sources, including ground-based rain gauges, satellite observations, and radar data. The goal is to develop new insights into rainfall patterns and variability, and to improve our ability to predict rainfall events.

The project will use a variety of big data analytics techniques, including machine learning, data mining, and statistical analysis. The project will also develop new algorithms for pattern analysis.

The project is expected to have a significant impact on our understanding of rainfall and our ability to manage water resources, agriculture, and disaster risk. The project will also contribute to the development of new big data analytics techniques for other applications.

DATA PREPROCESSING

1. Data Collection:

- The dataset used for this project is obtained from Kaggle. “Rainfall in India (1901-2015)” dataset is used for the Rainfall Analysis.

2. Data Loading:

- In the first step, we import the necessary libraries, including numpy, pandas, matplotlib.pyplot, seaborn & plotly.graph_objects.

- Using pd.read_csv(), we load the dataset from the file "rainfall in india 1901-2015.csv" into a Pandas DataFrame (data). The encoding='unicode_escape' parameter is specified to handle any potential encoding issues.

```python
#Importing the Dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go

```

```python
#Loading the csv data
data = pd.read_csv("rainfall in india 1901-2015.csv", encoding='unicode_escape')
```

3. Data Overview:

- data.head() displays the first few rows of the dataset, allowing you to inspect the structure and content of the data.

```python
data.head()
```

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7

- data.info() provides a concise summary of the dataset, including the data types of each column and the presence of missing values.

```
1  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SUBDIVISION  4116 non-null   object 
 1   YEAR         4116 non-null   int64  
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

- data.describe() gives statistical information about numerical columns, such as count, mean, std (standard deviation), min, and max.

```
1  data.describe()
```

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.000000	4109.000000	4112.000000	4110.000000
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	269.539667	188.770477	135.408345
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	175.600000	155.975000	100.525000
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	284.800000	259.400000	173.900000
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	418.400000	377.800000	265.800000
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000

- data.shape returns the total number of rows and columns in the dataset, which is printed to provide an overview.

```
#Number of rows and columns in the dataset
data.shape
```

```
(4116, 19)
```

- data.tail() displays the last few rows of the dataset.

```
data.tail()
```

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9

4. Handling Missing Values:

- data.isnull().sum() is used to identify the number of missing values in each column. This step helps you understand the extent of missing data. We can observe that there are many missing values from each column. This needs to be taken care of for better analysis.

```
#checking for missing values
data.isnull().sum()
```

```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

- data.dropna(inplace=True) is employed to remove rows with missing values. This decision is made to ensure a clean dataset, as an alternative to imputation strategies. From the below snippet of code we can see that all the rows containing missing values has been removed.

```
data.dropna(inplace=True)
data.isnull().sum()
```

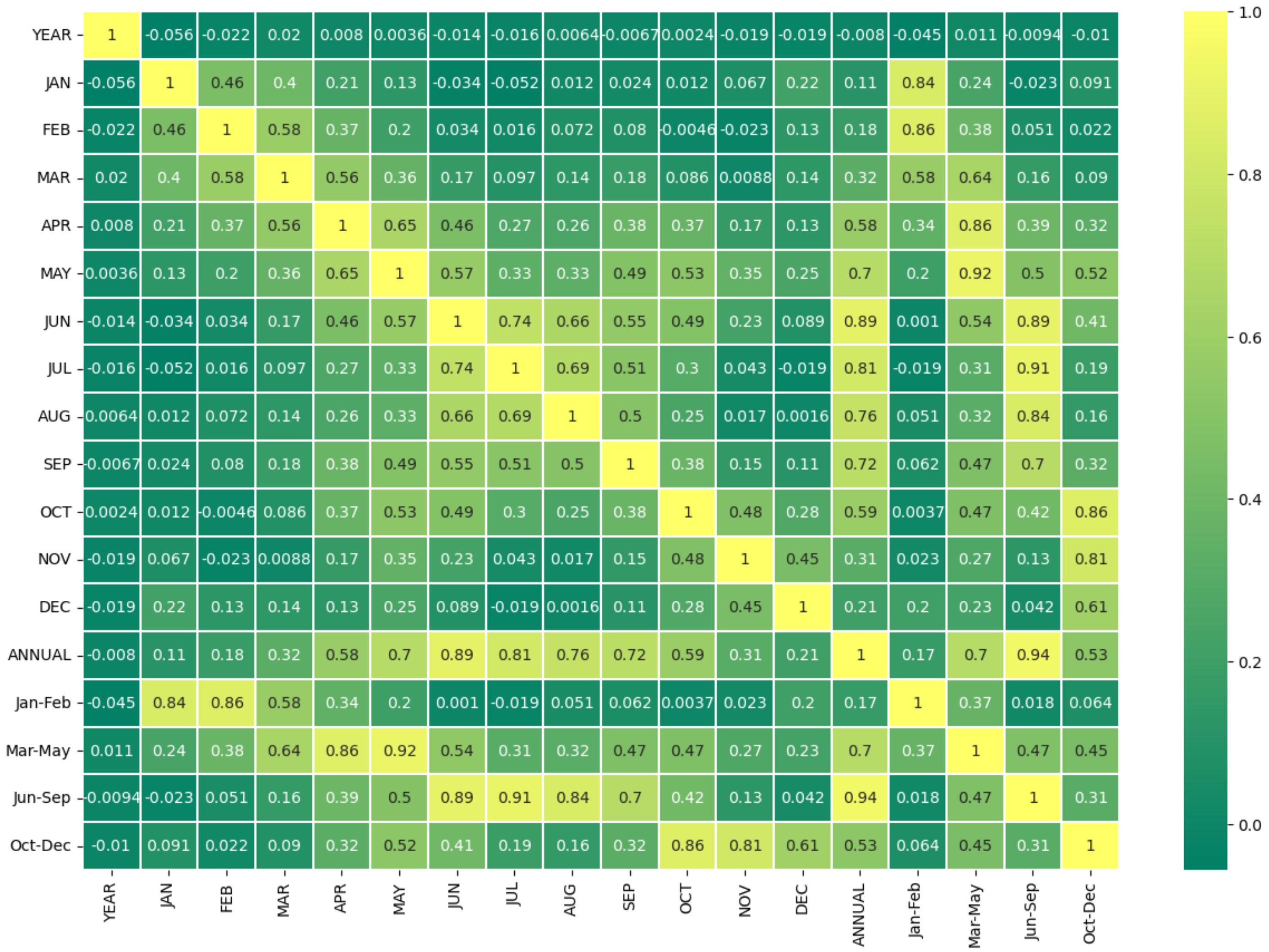
```
SUBDIVISION    0
YEAR           0
JAN            0
FEB            0
MAR            0
APR            0
MAY            0
JUN            0
JUL            0
AUG            0
SEP            0
OCT            0
NOV            0
DEC            0
ANNUAL         0
Jan-Feb        0
Mar-May        0
Jun-Sep        0
Oct-Dec        0
dtype: int64
```

DATA EXPLORATION & VISUALIZATION

1. Correlation Analysis

Correlation Analysis is statistical method that is used to discover if there is a relationship between two variables/datasets, and how strong that relationship may be. Essentially, correlation analysis is used for spotting patterns within datasets. A positive correlation result means that both variables increase in relation to each other, while a negative correlation means that as one variable decreases, the other increases.

```
plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),linewidth=.01,annot=True,cmap="summer")
plt.show()
```

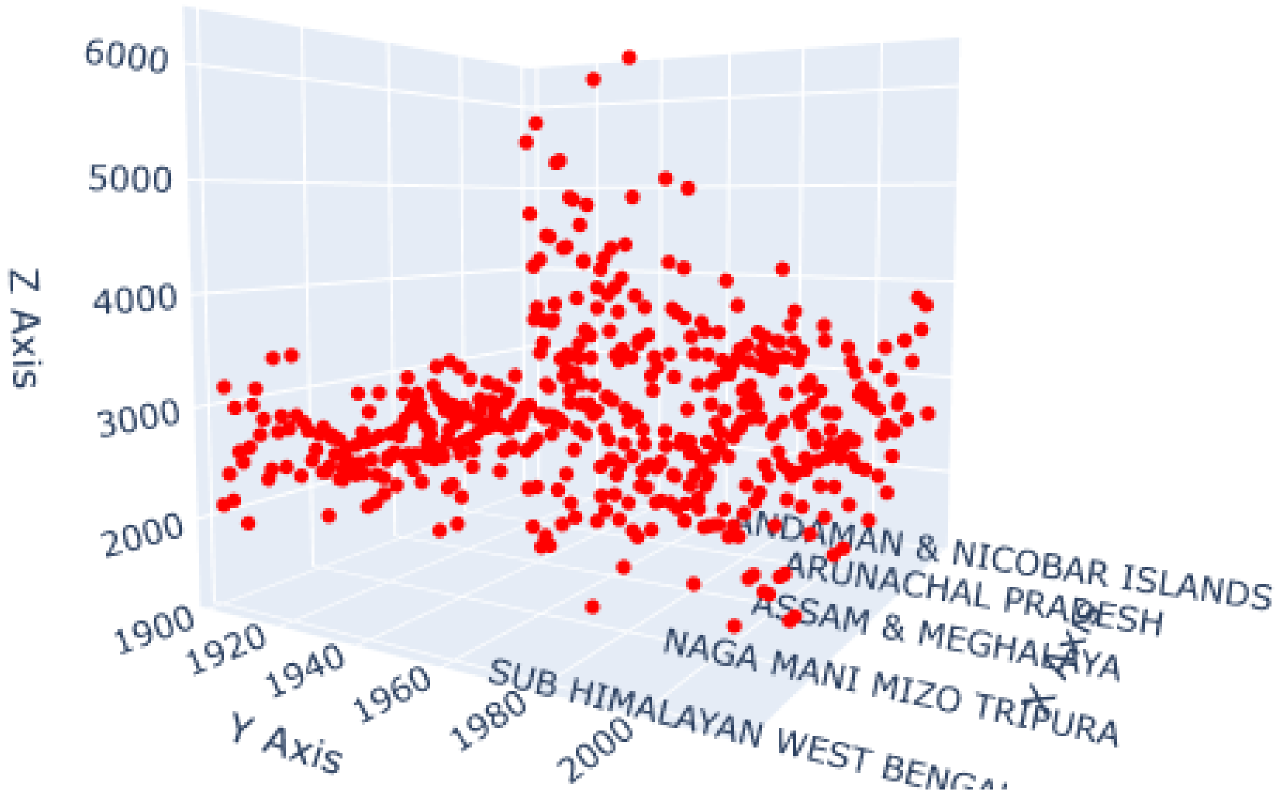


The above heat map visualizes the correlation between different features like year, different months, annual and seasons. It helps identify relationships between variables and assess multicollinearity.

2. 3D Scatter Plot

A 3D Scatter Plot is a mathematical diagram, the most basic version of three-dimensional plotting used to display the properties of data as three variables of a dataset using the Cartesian coordinates.

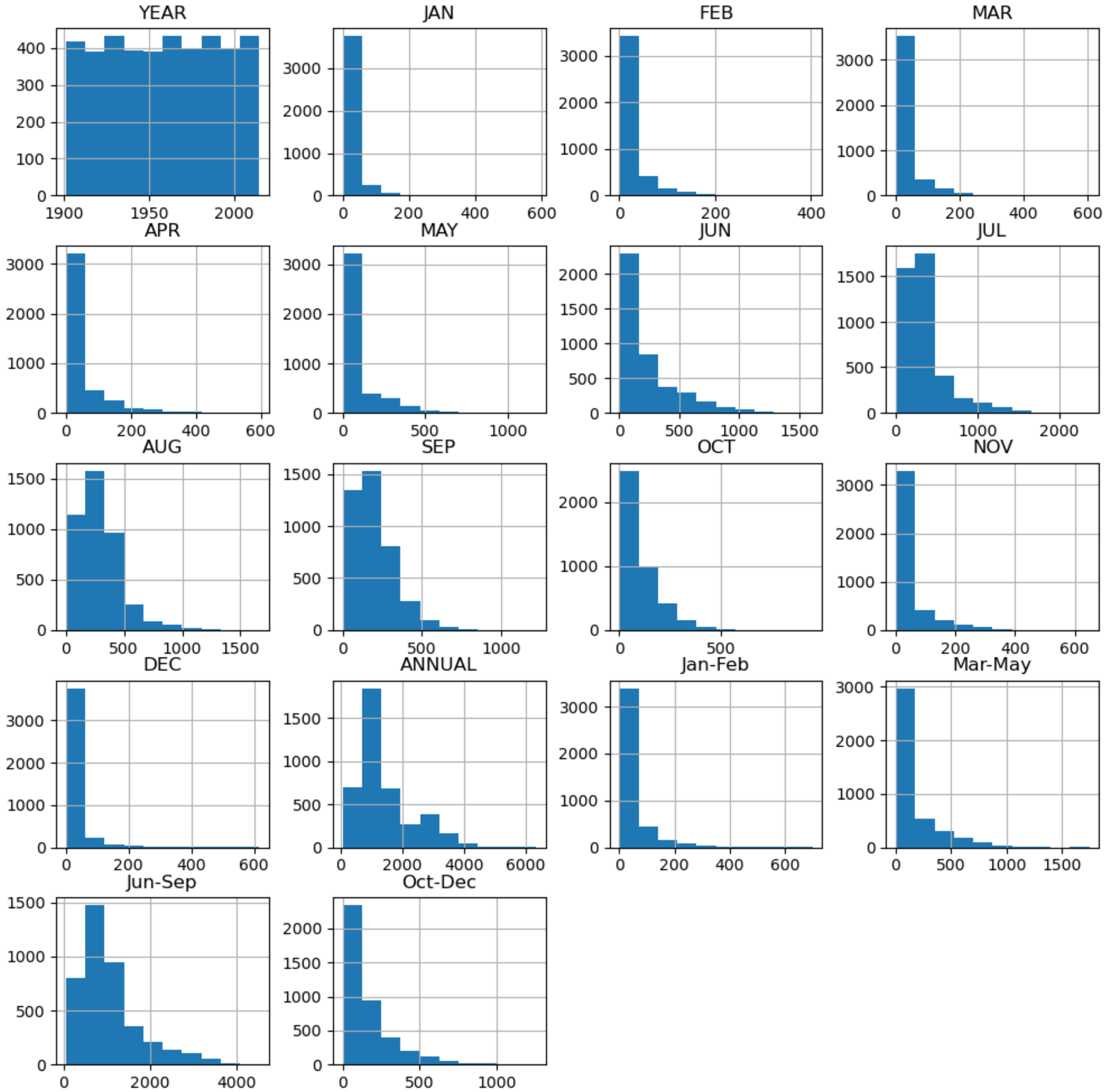
```
# Create a 3D scatter plot
fig = go.Figure(data=[go.Scatter3d(x=data["SUBDIVISION"], y=data["YEAR"],
                                   z=data["ANNUAL"], mode='markers',
                                   marker=dict(size=8, color='red'))])

# Set axis labels
fig.update_layout(scene=dict(xaxis_title='X Axis', yaxis_title='Y Axis',
                             zaxis_title='Z Axis'))

# Show the plot
fig.show()

```



Plotly is used to create an interactive 3D scatter plot, visualizing the relationship between subdivision, year, and annual rainfall. The plot can help to identify the patterns, trends, and outliers in the data. For example, one can see that some subdivisions have higher or lower rainfall than others, or that some years have more or less rainfall than the average.

3. Histogram:

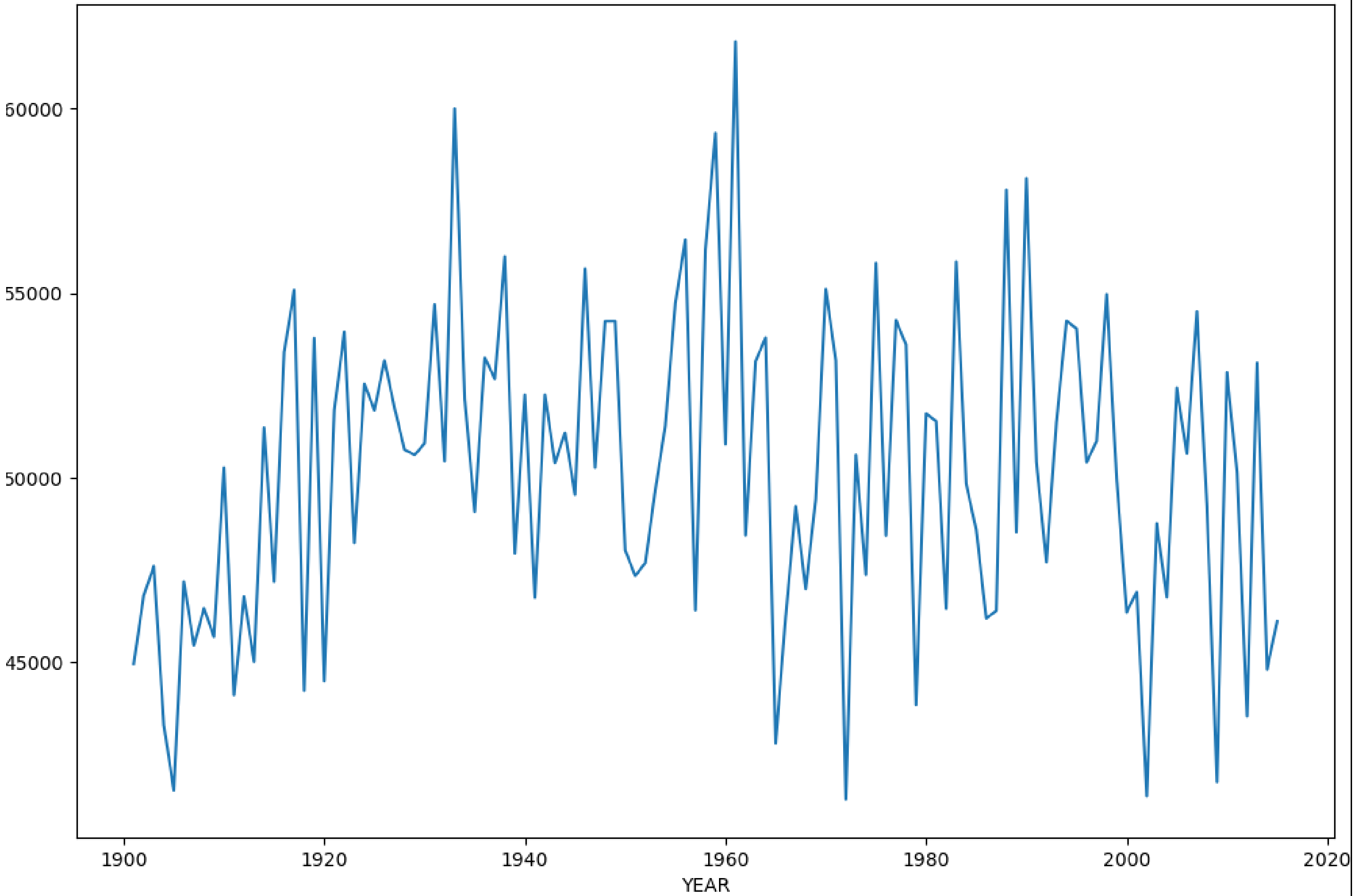
```
#Histogram Representation
data.hist(figsize=(12,12));
```



A histogram graph is a bar graph representation of data. It is a representation of a range of outcomes into columns formation along the x-axis. In the same histogram, the number count or multiple occurrences in the data for each column is represented by the y-axis. The x-axis represents the range of annual rainfall values, and the y-axis would represent the frequency or count of years falling into specific ranges. By examining the histograms, we can identify potential outliers or unusual patterns in the data. The shape of the histograms can provide insights into the central tendency of the data.

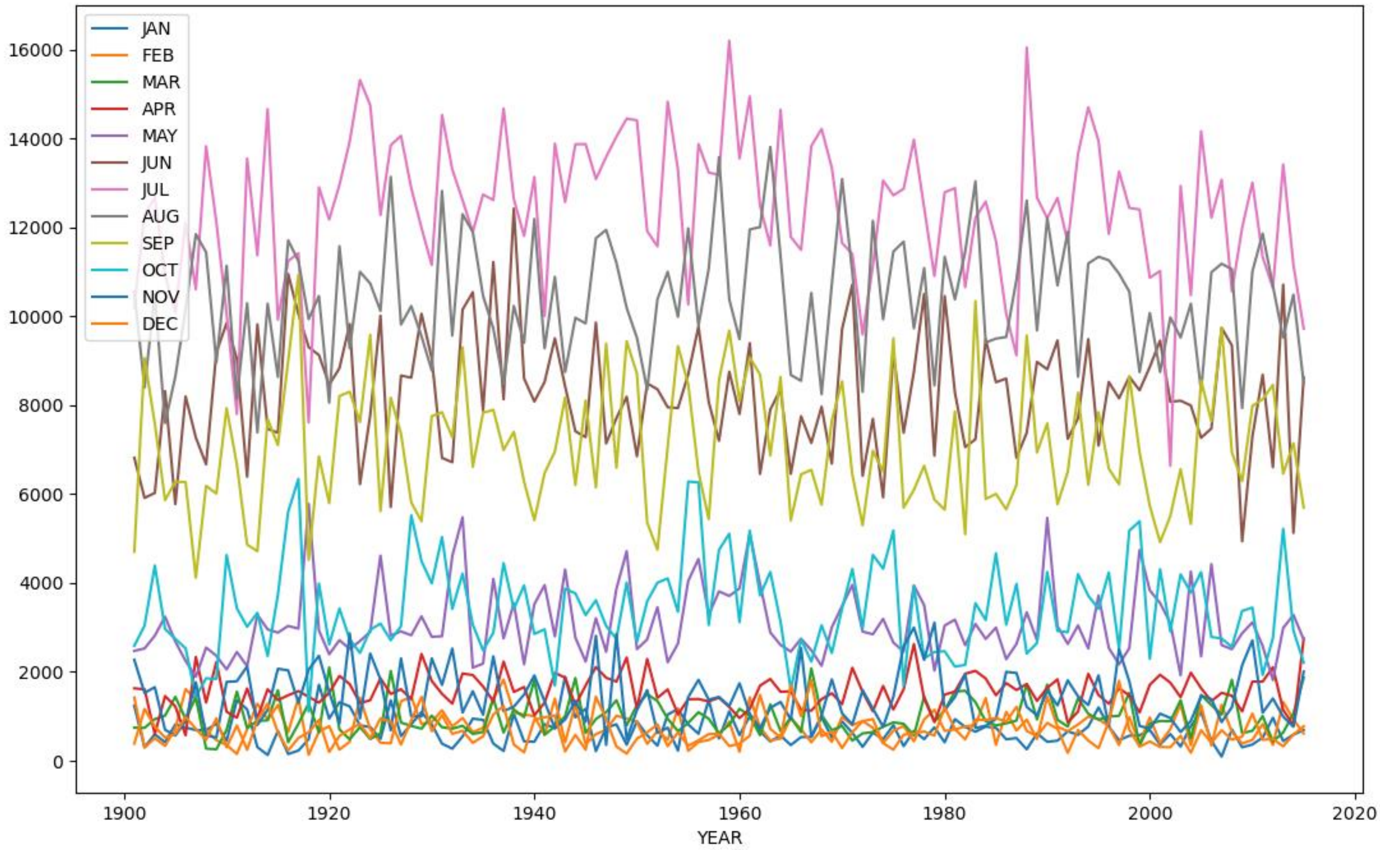
4. Time Series Plot

```
data.groupby("YEAR").sum()['ANNUAL'].plot(figsize=(12,8));
```



A time series plot is created to visualize the overall trend of annual rainfall over the years. A time-series plot, also known as a time plot, is a type of graph that displays data points collected in a time sequence. The x-axis represents the year and the y-axis represents the rainfall in mm. The highest annual rainfall was recorded in 1960 with 6331.1 mm, and the lowest was in 2002 with 62.3 mm.

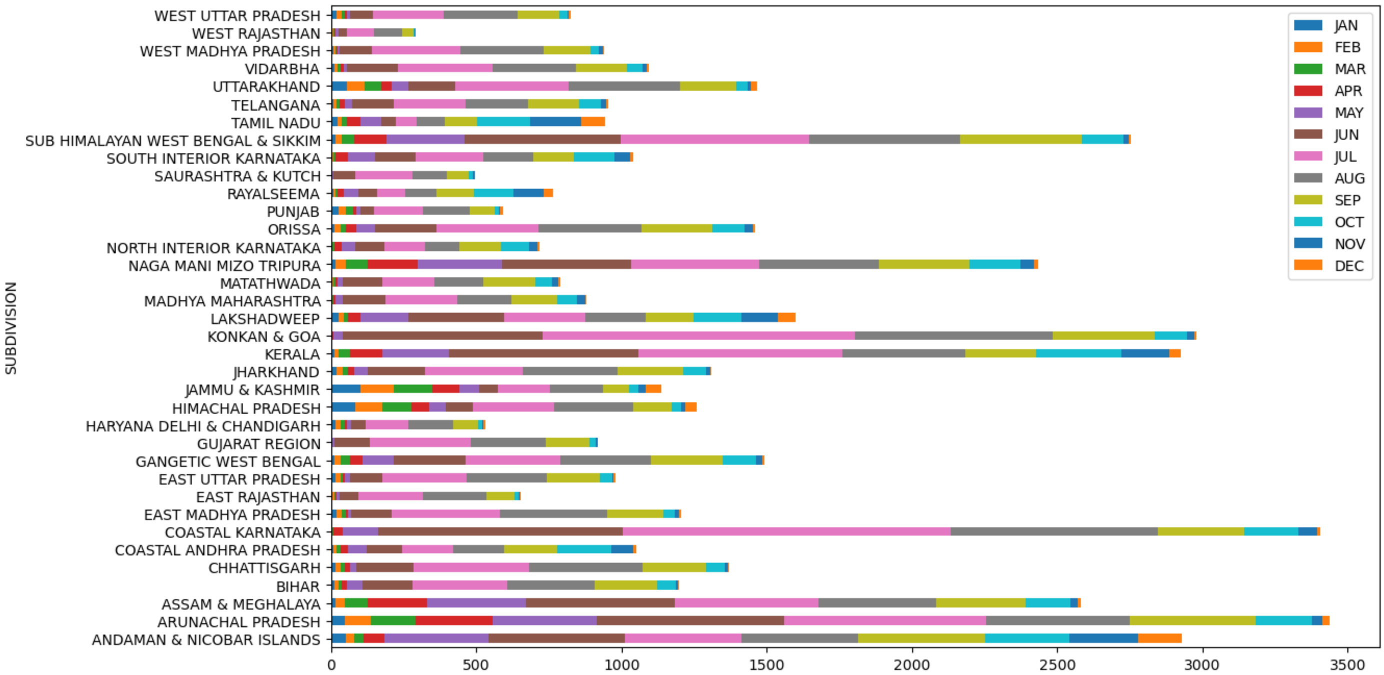
```python
data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
        'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(figsize=(13,8));
```



The above graph shows the line plot of the total annual rainfall for each year, grouped by the 12 months. The x-axis represents the year and the y-axis represents the rainfall in mm. There are 12 line plots which shows 12 months of a year. We can observe that rainfall varies significantly across the months, with the highest peaks occurring in July and the lowest values in February.

5. Horizontal bar graph

```
data[['SUBDIVISION', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("SUBDIVISION").mean().plot.barh(stacked=True,figsize=(13,8));
```



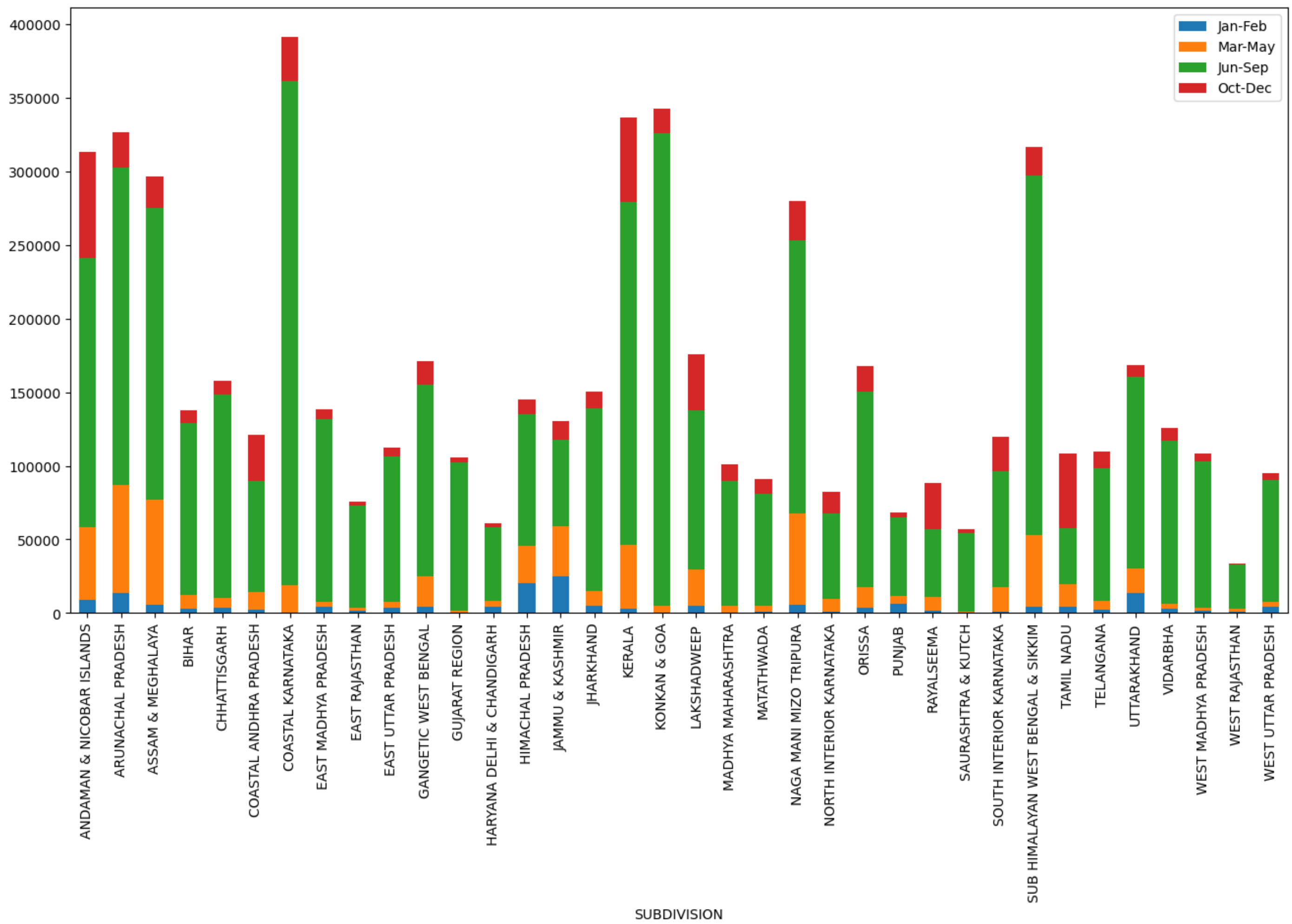
The above graph shows a horizontal bar plot of the mean monthly rainfall for each subdivision in India. The plot is stacked, meaning that the length of each bar represents the total annual rainfall, and the different colors represent the contribution of each month. The subdivisions with the highest annual rainfall are Coastal Karnataka, Konkan & Goa, and Andaman & Nicobar Islands. The subdivisions with the lowest annual rainfall are West Rajasthan, Saurashtra & Kutch, and Haryana Delhi & Chandigarh.

The months with the highest rainfall are July and August, which correspond to the peak of the monsoon season in India. The months with the lowest rainfall are January, February, and December, which correspond to the winter season in India.

The subdivisions with the most variation in monthly rainfall are Arunachal Pradesh, Nagaland, Manipur, Mizoram & Tripura, and Sub Himalayan West Bengal & Sikkim. The subdivisions with the least variation in monthly rainfall are West Rajasthan, Saurashtra & Kutch, and Tamil Nadu.

6. Vertical Bar Graph

```
data[['SUBDIVISION', 'Jan-Feb', 'Mar-May', 'Jun-Sep', 'Oct-Dec']].
groupby("SUBDIVISION").sum().plot.bar(stacked=True,figsize=(16,8));
```



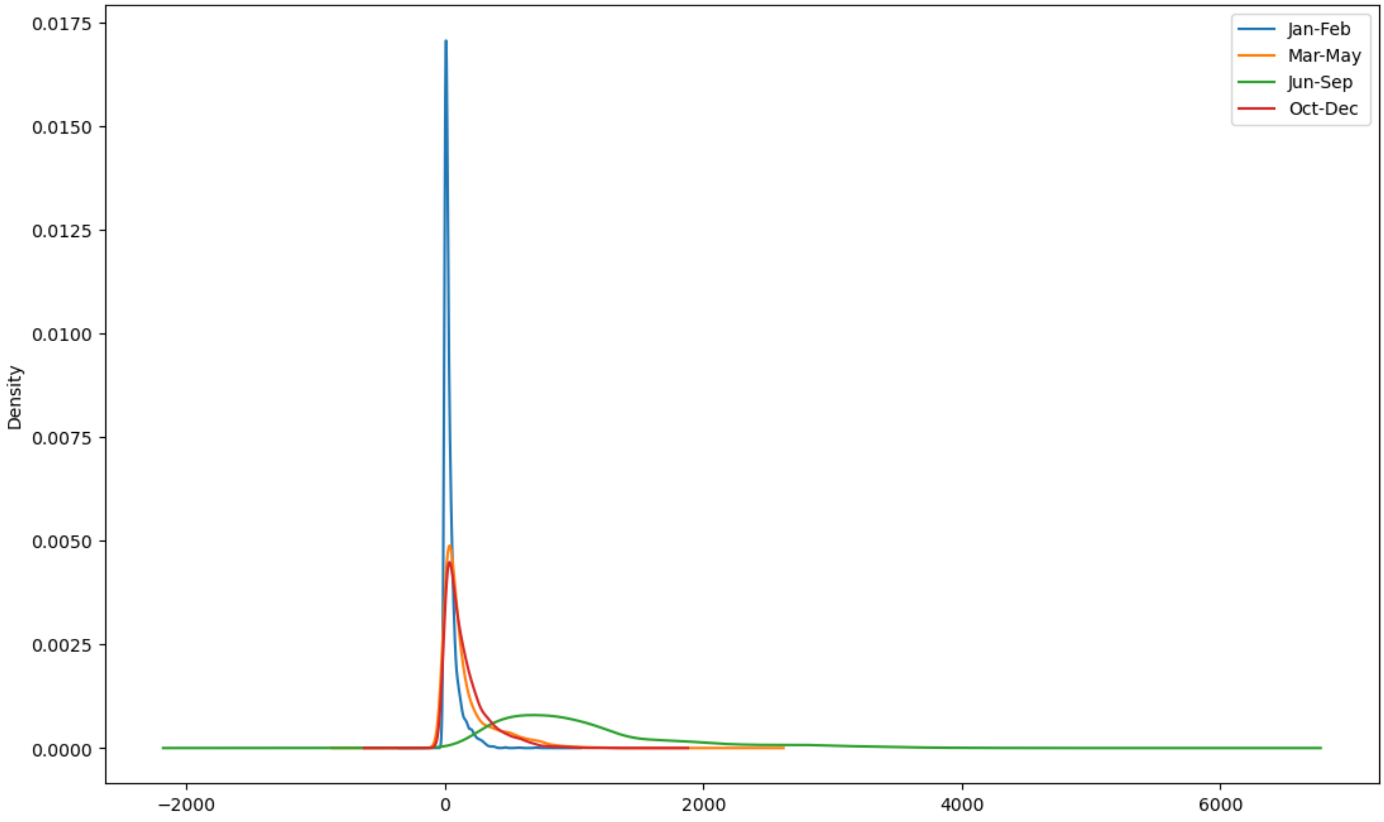
The above graph shows a stacked bar plot of the total rainfall in each subdivision of India for four seasons: Jan-Feb, Mar-May, Jun-Sep, and Oct-Dec. The plot shows the variation of rainfall across different regions and seasons.

Jun-Sep is the rainiest season for most of the subdivisions, except for Tamil Nadu and Coastal Andhra Pradesh, where Oct-Dec is the rainiest season.

Jan-Feb is the driest season for almost all the subdivisions, except for Jammu & Kashmir and Himachal Pradesh, where Dec-Jan is the driest season.

7. Density Graph

```
data[['Jan-Feb', 'Mar-May',
      'Jun-Sep', 'Oct-Dec']].plot(kind="kde",figsize=(13,8));
```



The above figure is a kernel density estimation (KDE) plot for the four seasonal rainfall columns in the data. A KDE plot is a smoothed version of a histogram that shows the probability density of the data at different values.

Jun-Sep is the peak season for rainfall in India, as it corresponds to the monsoon period. The KDE curve for this season is right-skewed, meaning most of the rainfall values are concentrated on the lower end. The mean rainfall for this season is around 1000 mm. Oct-Dec is the post-monsoon season, which has a moderate amount of rainfall. The KDE curve for this season is symmetric, meaning the rainfall values are evenly distributed around the mean. The mean rainfall for this season is around 150 mm. Jan-Feb is the winter season, which has the least amount of rainfall. The KDE curve for this season is left-skewed, meaning most of the rainfall values are close to zero. The mean rainfall for this season is around 20 mm. Mar-May is the pre-monsoon season, which has a variable amount of rainfall. The KDE curve for this season is bimodal, meaning there are two peaks in the distribution. One peak is around 50 mm, which represents the regions with low rainfall, and the other peak is around 200 mm, which represents the regions with high rainfall.

SOURCE CODE

```
#Importing the Dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go

#Loading the csv data
data = pd.read_csv("rainfall in india 1901-2015.csv")

# Data preprocessing
data.info()
data.head()
data.tail()
data.describe()

#Number of rows and columns in the dataset
data.shape

#checking for missing values
data.isnull().sum()
```

```
data.dropna(inplace=True)

data.isnull().sum()


#Correlation matrix

plt.figure(figsize=(15,6))

sns.heatmap(data.corr(),linewidth=.01,annot=True,cmap="summer")

plt.show()


# Create a 3D scatter plot

fig = go.Figure(data=[go.Scatter3d(x=data[1:500]["SUBDIVISION"], y=data["YEAR"],
z=data["ANNUAL"], mode='markers', marker=dict(size=8, color='red'))])


# Set axis labels

fig.update_layout(scene=dict(xaxis_title='X Axis', yaxis_title='Y Axis', zaxis_title='Z Axis'))


# Show the plot

fig.show()


#Histogram Representation

data.hist(figsize=(12,12));


data.groupby("YEAR").sum()['ANNUAL'].plot(figsize=(12,8));
```

```
data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(figsize=(13,8));


data[['SUBDIVISION', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
'AUG', 'SEP', 'OCT', 'NOV',
'DEC']].groupby("SUBDIVISION").mean().plot.barh(stacked=True,figsize=(13,8));


data[['SUBDIVISION', 'Jan-Feb', 'Mar-May',
'Jun-Sep', 'Oct-
Dec']].groupby("SUBDIVISION").sum().plot.bar(stacked=True,figsize=(16,8));


data[['Jan-Feb', 'Mar-May',
'Jun-Sep', 'Oct-Dec']].plot(kind="kde",figsize=(13,8));
```

CONCLUSION

In conclusion this project provides a comprehensive approach to exploring historical rainfall data in India. The methodology involved loading the data, checking for missing values, performing correlation analysis, and visualizing the data through various plots and charts.

The combination of spatial and temporal analyses, correlation studies, and visualizations has uncovered valuable information. This project serves as a foundation for more in-depth analyses, including advanced statistical modeling and machine learning approaches, to further unravel complex patterns and contribute to informed decision-making in the context of water resource management and climate resilience.

Policymakers can leverage these insights to formulate data-driven policies for water resource allocation, disaster preparedness, and sustainable development.

BIBLIOGRAPHY

- https://www.kaggle.com/datasets/kkhandekar/statewise-rainfall-1901-to-2015
- https://www.kaggle.com/code/kkhandekar/indian-sub-division-rainfall-starter-notebook
- https://rmets.onlinelibrary.wiley.com/doi/full/10.1002/asl.932
- https://chat.openai.com/c/62542427-4dc3-4774-9811-894d1b4c4b60
- https://www.bing.com/search?q=Bing+AI&showconv=1&FORM=undexpand
- https://www.nature.com/articles/s41598-020-67228-7
- https://www.kaggle.com/code/darsh79/starter-rainfall-in-india-99bfc809-4