Name - Gaurang A Raorane Div - D15A Roll no - 49 Batch - C

**Experiment - 2**

**Aim -** Data Visualization/ Exploratory data Analysis using Matplotlib and Seaborn.

Theory :-

Exploratory Data Analysis (EDA) is a critical step in the data analysis process, where the primary goal is to summarize the main characteristics of the dataset, often with the help of visualizations, to uncover patterns, relationships, and potential outliers.

### **Matplotlib:**

Matplotlib is a comprehensive 2D plotting library for creating static, animated, and interactive visualizations in Python. It provides a wide range of high-quality charts and plots, making it a versatile tool for data visualization.

#### Key Features:

* Basic Plots:
  + Matplotlib offers a variety of basic plots, including line plots, scatter plots, bar plots, histograms, and more.
* Customization:
  + Users have extensive control over plot customization, allowing adjustments to colors, markers, labels, and titles.
* Subplots:
  + Matplotlib supports creating multiple plots within the same figure, facilitating side-by-side visualizations for better comparison.
* Annotations and Text:
  + Annotations, text, and arrows can be added to enhance plot interpretation and communication of insights.
* Export and Integration:
  + Plots can be saved in various formats (PNG, PDF, SVG), ensuring compatibility with different platforms. Integration with LaTeX is also supported.
* Versatility:
  + Matplotlib is suitable for a range of applications, from simple exploratory data analysis to complex scientific visualizations.

### **Seaborn:**

Seaborn is built on top of Matplotlib and provides a high-level interface for creating informative and attractive statistical graphics. It simplifies the process of creating complex visualizations, particularly for statistical analysis.

#### Key Features:

* Statistical Plots:
  + Seaborn includes specialized functions for statistical visualizations like scatter plots with regression lines, box plots, and violin plots.
* Color Palettes:
  + Seaborn provides visually appealing color palettes to enhance the aesthetics of plots.
* Grids and Subplots:
  + Grids and subplots are easily customizable, allowing for a more organized presentation of data.
* Integration with Pandas:
  + Seaborn seamlessly integrates with Pandas DataFrames, simplifying the visualization of structured data.
* Facet Grids:
  + Facet grids enable the creation of multiple subplots based on different variables, providing a deeper analysis of the data.

### **Matplotlib:**

#### 1. **Scatter Plot:**

* Purpose: Visualize the relationship between two numerical variables.

#### 2. **Bar Graph:**

* Purpose: Compare categories or show the distribution of a categorical variable.

#### 3. **Histogram:**

* Purpose: Display the distribution of a numerical variable.

#### 4. **Box Plot:**

* Purpose: Summarize the distribution, identify outliers.

### **Seaborn:**

#### 1. **Pair Plot:**

* Purpose: Visualize pairwise relationships between numerical variables.

#### 2. **Correlation Heatmap:**

* Purpose: Visualize the correlation matrix.

#### 3. **Violin Plot:**

* Purpose: Display the distribution of a numerical variable across different categories.

#### 4. **Facet Grid:**

* Purpose: Create a grid of subplots for detailed comparisons.

Plots Implementation(using Matplotlib):-

import seaborn as sns

import matplotlib.pyplot as plt

numerical\_columns = df.select\_dtypes(include=['float64']).columns

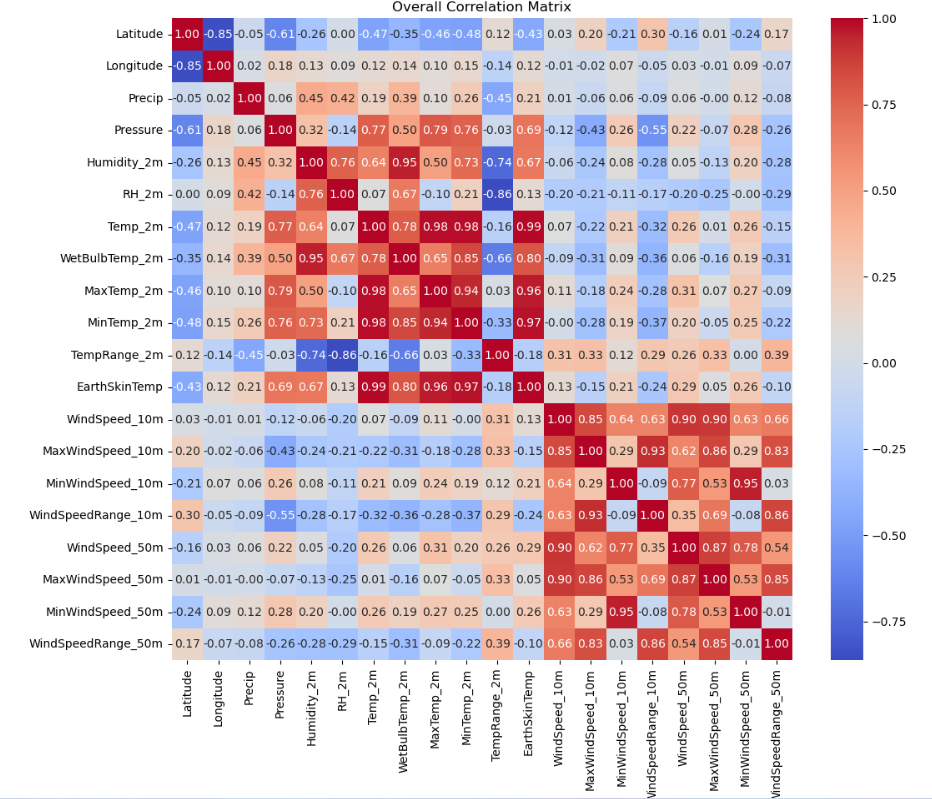
overall\_corr\_matrix = df[numerical\_columns].corr()

plt.figure(figsize=(12, 10))

sns.heatmap(overall\_corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Overall Correlation Matrix')

plt.show()



# Scatter plot for Temperature vs Precipitation

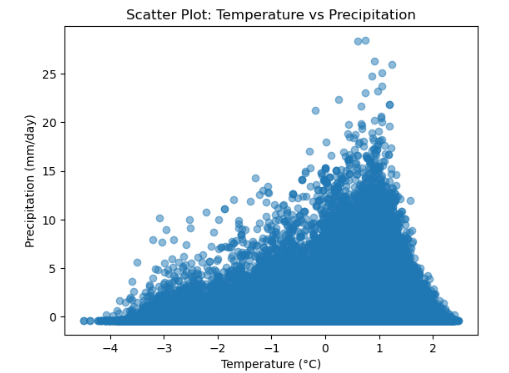
plt.scatter(df['Temp\_2m'], df['Precip'], alpha=0.5)

plt.title('Scatter Plot: Temperature vs Precipitation')

plt.xlabel('Temperature (°C)')

plt.ylabel('Precipitation (mm/day)')

plt.show()



district\_mean\_temp = df.groupby('District')['Temp\_2m'].mean().sort\_values()

plt.bar(district\_mean\_temp.index, district\_mean\_temp)

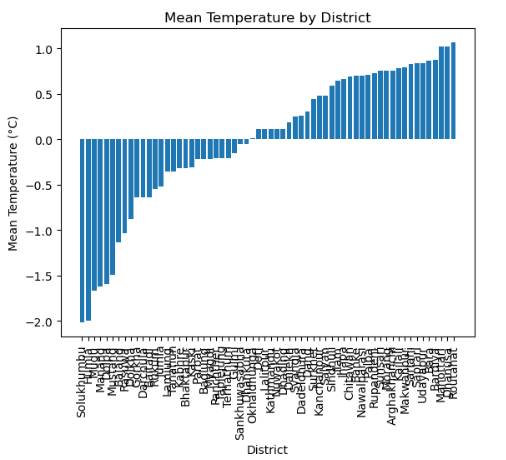
plt.title('Mean Temperature by District')

plt.xlabel('District')

plt.ylabel('Mean Temperature (°C)')

plt.xticks(rotation=90) # Rotate x-axis labels for better visibility

plt.show()



# Example: Histogram for Temperature distribution

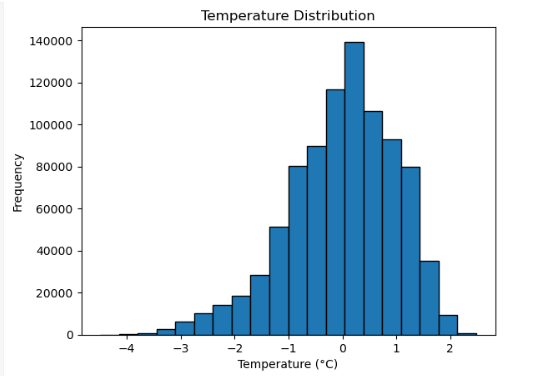
plt.hist(df['Temp\_2m'], bins=20, edgecolor='black') # Adjust the number of bins as needed

plt.title('Temperature Distribution')

plt.xlabel('Temperature (°C)')

plt.ylabel('Frequency')

plt.show()



# Example: Box plots for Temperature, Humidity, and Precipitation

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))

# Box plot for Temperature

axes[0].boxplot(df['Temp\_2m'])

axes[0].set\_title('Temperature Distribution')

axes[0].set\_ylabel('Temperature (°C)')

# Box plot for Humidity

axes[1].boxplot(df['Humidity\_2m'])

axes[1].set\_title('Humidity Distribution')

axes[1].set\_ylabel('Humidity')

# Box plot for Precipitation

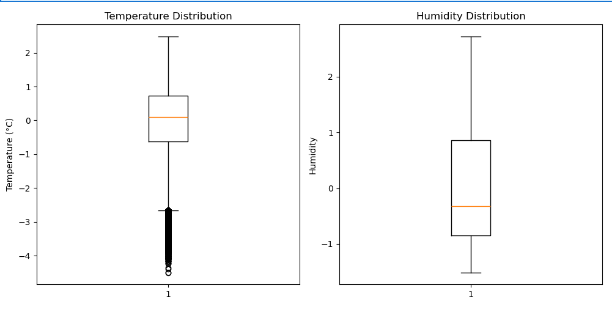
axes[2].boxplot(df['Precip'])

axes[2].set\_title('Precipitation Distribution')

axes[2].set\_ylabel('Precipitation (mm/day)')

plt.tight\_layout()

plt.show()



# Example: Pie chart for the distribution of districts

district\_counts = df['District'].value\_counts()

labels = district\_counts.index

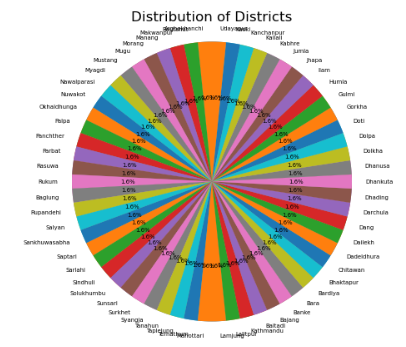
sizes = district\_counts.values

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, wedgeprops=dict(width=2.0), textprops={'fontsize': 5})

plt.title('Distribution of Districts')

plt.axis('equal') # Equal aspect ratio ensures that the pie chart is circular.

plt.show()

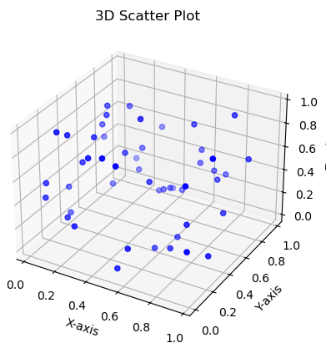


import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import numpy as np

# Sample data

np.random.seed(42)

x = np.random.rand(50)

y = np.random.rand(50)

z = np.random.rand(50)

# Create a 3D scatter plot

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(x, y, z, c='blue', marker='o')

# Set labels and title

ax.set\_xlabel('X-axis')

ax.set\_ylabel('Y-axis')

ax.set\_zlabel('Z-axis')

ax.set\_title('3D Scatter Plot')

plt.show()

Implementations of various plots(using Seaborn):-

import seaborn as sns

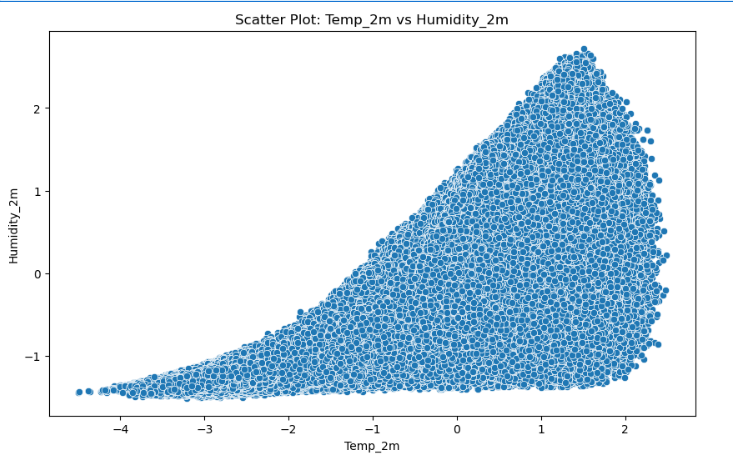
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Temp\_2m', y='Humidity\_2m', data=df)

plt.title('Scatter Plot: Temp\_2m vs Humidity\_2m')

plt.show()



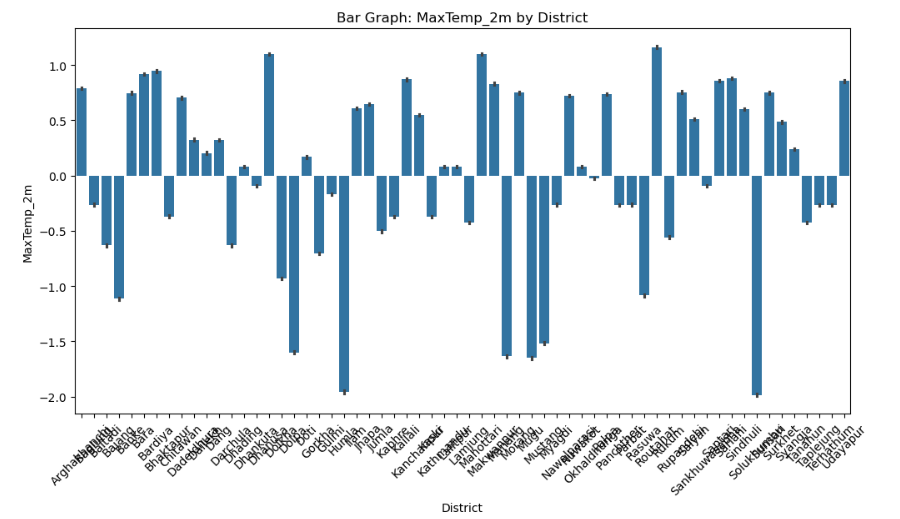
plt.figure(figsize=(12, 6))

sns.barplot(x='District', y='MaxTemp\_2m', data=df)

plt.title('Bar Graph: MaxTemp\_2m by District')

plt.xticks(rotation=45)

plt.show()



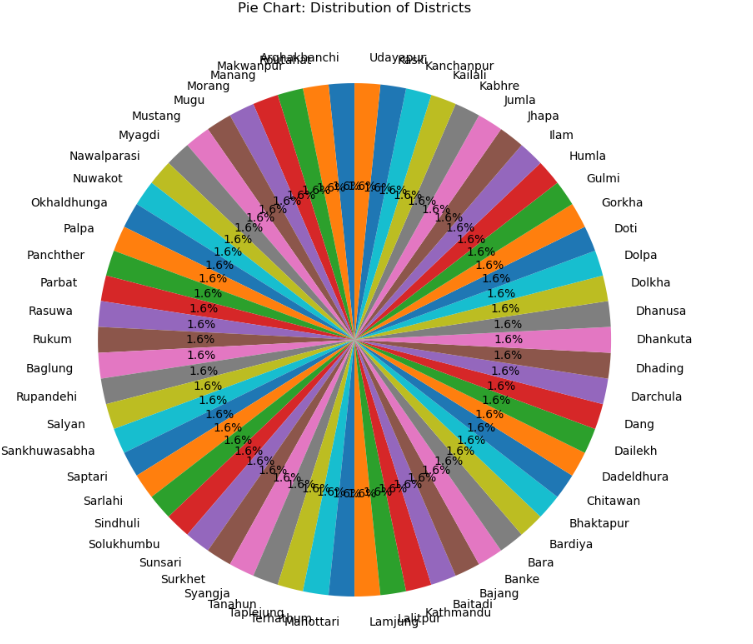
district\_counts = df['District'].value\_counts()

plt.figure(figsize=(10, 10))

plt.pie(district\_counts, labels=district\_counts.index, autopct='%1.1f%%', startangle=90)

plt.title('Pie Chart: Distribution of Districts')

plt.show()



# KDE plot for Temp\_2m

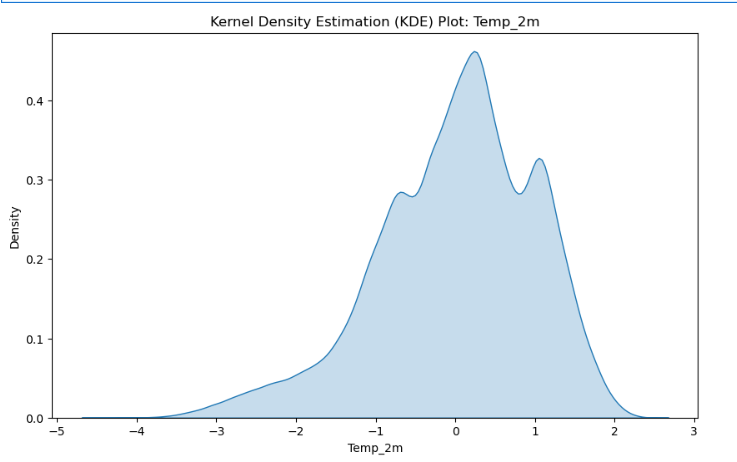
plt.figure(figsize=(10, 6))

sns.kdeplot(data=df, x='Temp\_2m', fill=True)

plt.title('Kernel Density Estimation (KDE) Plot: Temp\_2m')

plt.xlabel('Temp\_2m')

plt.show()



# Create a FacetGrid using Seaborn

g = sns.FacetGrid(df, col='District', col\_wrap=4, height=4) # Adjust col\_wrap and height as needed

# Scatter plot on each subplot

g.map(sns.scatterplot, 'Temp\_2m', 'Humidity\_2m')

# Customize labels and title

g.set\_axis\_labels('Temperature at 2m (°C)', 'Humidity at 2m (%)')

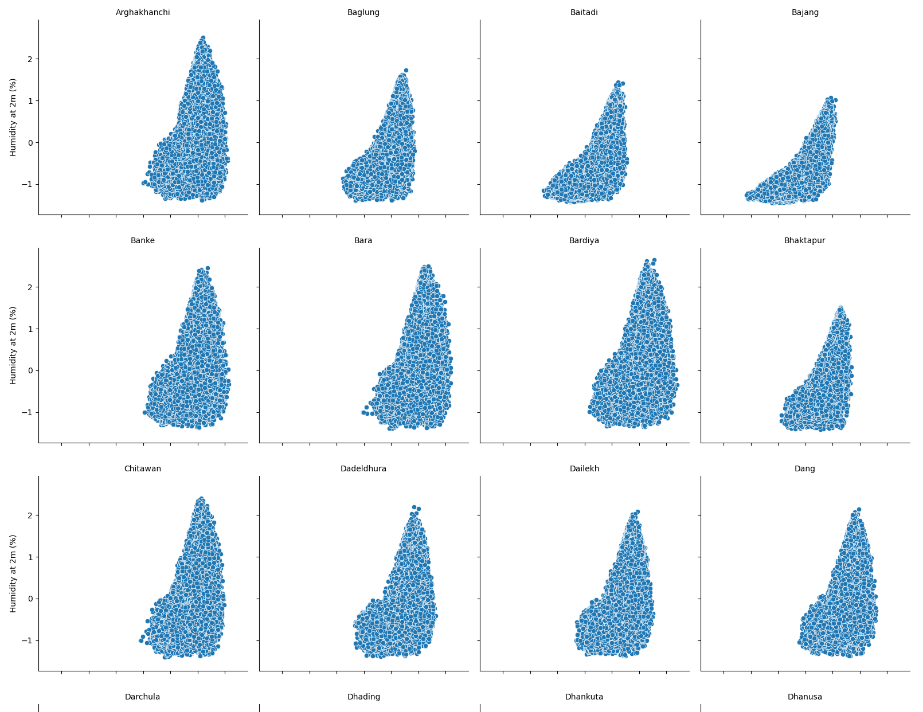
g.set\_titles(col\_template="{col\_name}")

# Adjust layout

plt.tight\_layout()

# Show the plot

plt.show()



Conclusion :- In conclusion, the exploratory data analysis (EDA) performed using Seaborn and Matplotlib on the given dataset has provided valuable insights into the various parameters. Through a range of visualizations, including scatter plots, bar graphs, histograms, pie charts, and facet grids, we've gained a comprehensive understanding of the relationships and distributions within the data. These visualizations have enhanced our ability to identify patterns, correlations, and potential outliers, laying the foundation for more in-depth analysis.