**EX.NO:1** **INSTALLATION OF DATA ANALYSIS**

**AND**

**VISUALIZATION TOOL**

**AIM :**

**INSTALLATION PROCEDURE:**

**Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric Python packages.**Pandas**is one of those packages, and makes importing and analyzing data much easier**

**Step 1: To ensure that the system is updated and the necessary packages are installed, open a terminal window and type the following commands:**

**Sudo apt update**

**Step 2:**

**Sudo apt install python3**

**Step 3:**Verify the installation by checking the installed version:

**$ python3 –version**

Step 4 : Install pandas and numpy

**$ pip install pandas**

**$ pip install numpy**

Step 5: Install visualization tool

**$ pip install matplotlib**

**EX.NO:2** **EXPLORATORY DATA ANALYSIS**

**AIM:**

To Perform exploratory data analysis (EDA) on with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data.

ALGORITHM:

Step 1 : **Import necessary libraries**

Step 2 : **Create a simple example dataset**

Step 3 : **Display the dataset**

Step 4 : **Check for missing values**

Step 5 : **Visualize the distribution of email lengths,count and common words**

Step 6 : Stop

PROGRAM:

**# Import necessary libraries**

import pandas as pd

import matplotlib.pyplot as plt

from wordcloud

import WordCloud

**# Create a simple example dataset (replace this with your actual dataset**)

data = {'sender': ['sender1', 'sender2', 'sender1', 'sender3'], 'content': ['Hello, how are you?', 'Meeting tomorrow?','Important update', 'Check this out!']}

email\_data = pd.DataFrame(data)

**# Display the dataset**

print(email\_data)

**# Check for missing values** print(email\_data.isnull().sum())

**# Visualize the distribution of email lengths**

email\_data['email\_length'] = email\_data['content'].apply(len)

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

plt.hist(email\_data['email\_length'], bins=30, edgecolor='black')

plt.title('Distribution of Email Lengths')

plt.xlabel('Email Length') plt.ylabel('Frequency')

plt.show()

**# Visualize the count of emails per sender** plt.figure(figsize=(8, 5))

email\_data['sender'].value\_counts().plot(kind='bar', color='skyblue')

plt.title('Count of Emails per Sender')

plt.xlabel('Sender')

plt.ylabel('Count') plt.show()

**# Visualize the most common words in emails**

all\_emails = ' '.join(email\_data['content'])

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(all\_emails)

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

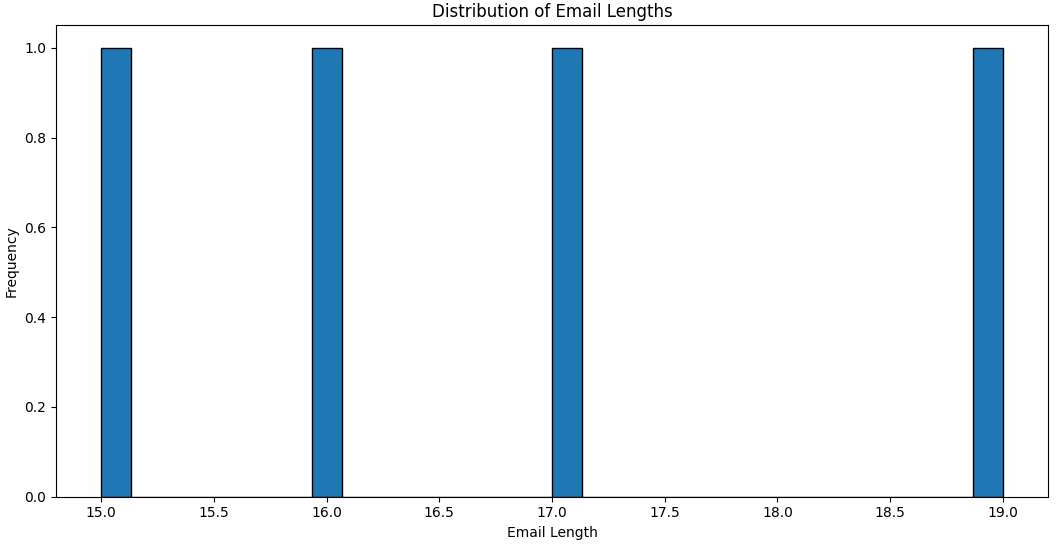
plt.imshow(wordcloud, interpolation='bilinear')

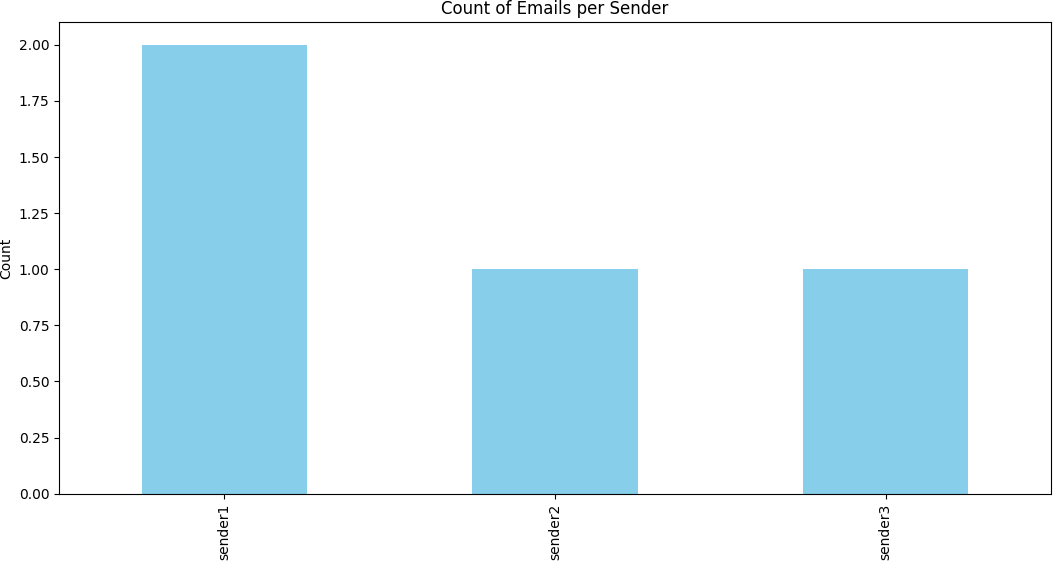
plt.axis('off')

plt.title('Word Cloud of Email Content')

plt.show()

**OUTPUT:**





**RESULT:**

**EX.NO:3**.1 **NUMPY ARRAYS**

**AIM:**

To Working with NumPy arrays, Pandas data frames, and creating basic plots using Matplotlib

**PROGRAM**

import numpy as np

**# Creating NumPy arrays**

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])

**# Basic operations on arrays**

sum\_array = arr1 + arr2

product\_array = arr1 \* arr2 mean\_value = np.mean(arr1)

**# Displaying array contents and results** print("Array 1:", arr1)

print("Array 2:", arr2)

print("Sum of arrays:", sum\_array) print("Product of arrays:", product\_array) print("Mean of Array 1:", mean\_value)

**OUTPUT**

**Array** **1:** **[1** **2** **3** **4** **5]**

**Array** **2:** **[** **6** **7** **8** **9** **10]**

**Sum** **of** **arrays:** **[** **7** **9** **11** **13** **15]**

**Product** **of** **arrays:** **[** **6** **14** **24** **36** **50]**

**Mean** **of** **Array** **1:** **3.0**

**Ex.No:3.2** **PANDAS** **DATA** **FRAMES**

**AIM:**

**ALGORITHM:**

**Step 1 : Creating a Pandas DataFrame from a dictionary**

**Step 2 : Displaying the DataFrame**

**Step 3: Basic DataFrame operations**

**Step 4 : Displaying results of operations**

**PROGRAM**

import pandas as pd

**# Creating a Pandas DataFrame from a dictionary** data = {

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'], 'Age': [25, 30, 22, 35, 28],

'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Boston']

}

df = pd.DataFrame(data)

**# Displaying the DataFrame** print("Original DataFrame:") print(df)

**# Basic DataFrame operations** average\_age = df['Age'].mean()

youngest\_person = df[df['Age'] == df['Age'].min()]

**# Displaying results of operations**

print("\nAverage Age:", average\_age) print("\nYoungest Person:") print(youngest\_person)

**OUTPUT:**

Original DataFrame:

**Name Age City**

1. Alice 25 New York
2. Bob 30 San Francisco
3. Charlie 22 Los Angeles
4. David 35 Chicago
5. Emily 28 Boston

**Average** **Age:** **28.0**

**Youngest** **Person:** **Name** **Age City**

**2** **Charlie** **22**  **Los** **Angeles**

**EX.NO:3.3** **BASIC** **PLOTS** **USING** **MATPLOTLIB**

**PROGRAM**

import matplotlib.pyplot as plt

**# Example data**

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

**# Line plot** plt.figure(figsize=(8, 6))

plt.plot(x, y, label='Line Plot', marker='o', linestyle='-', color='blue') plt.title('Simple Line Plot')

plt.xlabel('X-axis') plt.ylabel('Y-axis') plt.legend() plt.grid(True) plt.show()

**# Scatter plot** plt.figure(figsize=(8, 6))

plt.scatter(x, y, label='Scatter Plot', color='red', marker='x') plt.title('Simple Scatter Plot')

plt.xlabel('X-axis') plt.ylabel('Y-axis')

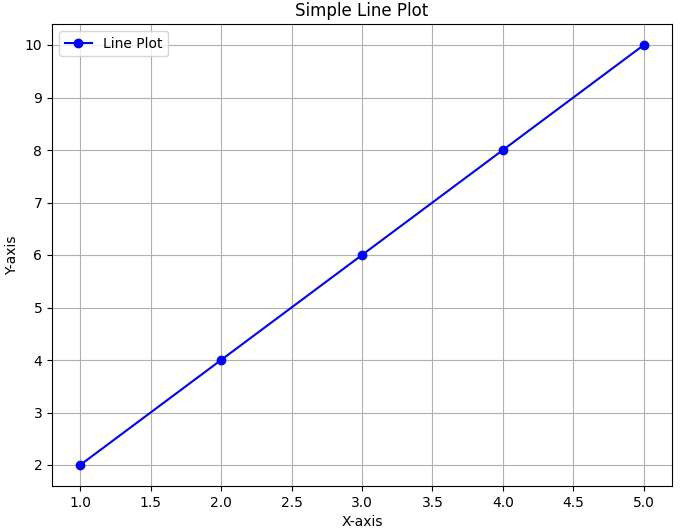
plt.legend()

plt.grid(True)

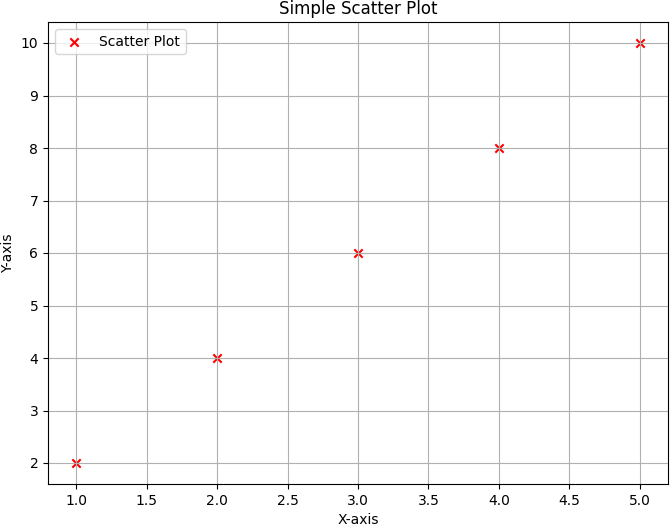
plt.show()

**OUTPUT:**

1. **Line** **Plot:**



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



**Ex.No:3.4** **CUSTOMIZING PLOT**

**Program**

import numpy as np

import matplotlib.pyplot as plt

**# Generate data**

x = np.linspace(0, 10, 100) y1 = np.sin(x)

y2 = np.cos(x)

**# Create a figure and axis**

fig, ax = plt.subplots(figsize=(8, 6))

**# Plot the data with custom styles**

ax.plot(x, y1, label='Sine Wave', color='blue', linestyle='--', linewidth=2) ax.plot(x, y2, label='Cosine Wave', color='red', linestyle='-', linewidth=2)

**# Customize axes labels and title** ax.set\_xlabel('X-axis') ax.set\_ylabel('Y-axis')

ax.set\_title('Customized Sine and Cosine Waves')

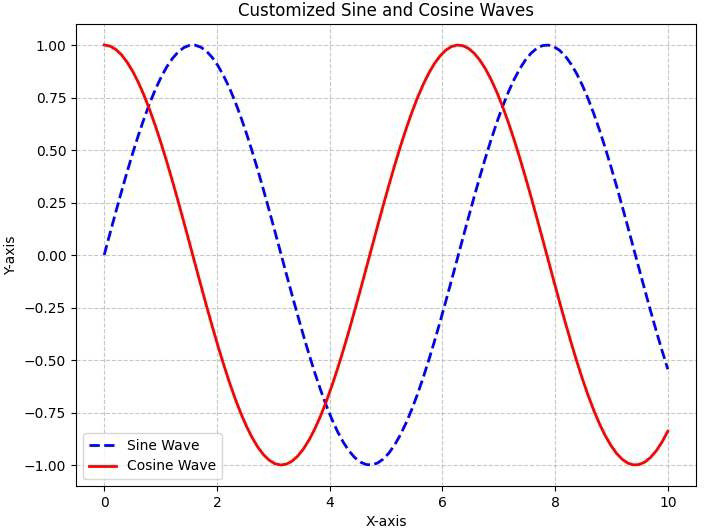
**# Add a legend** ax.legend()

**# Add grid**

ax.grid(True, linestyle='--', alpha=0.7)

**# Show the plot** plt.show()

**OUTPUT:**



**RESULT:**

**EX.NO:4** **DATA CLEANING USING R**

**AIM:**

To explore various variable and row filtering techniques in R for cleaning data, and then apply different plot features on a sample dataset.

**# Exploring and Cleaning Data in R:**

1. \*\*Loading Sample Data:\*\* **DEFINITION:**

Loading sample data refers to importing pre-existing datasets for analysis or experimentation. This process allows users to explore and manipulate data without creating it manually. Sample data often represents realistic scenarios, aiding in testing algorithms, visualizations, or statistical analyses in various domains

**PROGRAM**

import pandas as pd

**# Load a sample dataset (e.g., Iris dataset)**

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data" column\_names = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width", "class"]

df = pd.read\_csv(url, header=None, names=column\_names)

**# Display the first few rows of the dataset** print("Sample Dataset:")

print(df.head())

**OUTPUT**:

**Sample Dataset:**

sepal\_length sepal\_width petal\_length petal\_width class

| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| --- | --- | --- | --- | --- | --- |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

**EXPLORING** **DATA:**

**DEFINITION:**

Exploring data involves analyzing and understanding the characteristics, patterns, and relationships within a dataset. It encompasses tasks such as visualizations, summary statistics, and hypothesis testing to gain insights, identify trends, and inform subsequent analyses in various fields, including statistics, machine learning, and data science.

**PROGRAM**

import pandas as pd

**# Load a sample dataset (e.g., Iris dataset)**

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data" column\_names = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width", "class"] df = pd.read\_csv(url, header=None, names=column\_names)

**# Display basic information about the dataset** print("Dataset Information:")

print(df.info())

**#** **Display** **summary** **statistics** print("\nSummary Statistics:") print(df.describe())

**# Display the first few rows of the dataset** print("\nFirst Few Rows of the Dataset:") print(df.head())

**# Display unique classes in the 'class' column** print("\nUnique Classes:") print(df['class'].unique())

**OUTPUT:**

Dataset Information:

<class 'pandas.core.frame.DataFrame'> RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

* 1. sepal\_length 150 non-null float64
  2. sepal\_width 150 non-null float64
  3. petal\_length 150 non-null float64
  4. petal\_width 150 non-null float64
  5. class 150 non-null object dtypes: float64(4), object(1) memory usage: 6.0+ KB

None

**Summary** **Statistics:**

sepal\_length sepal\_width petal\_length petal\_width count 150.000000 150.000000 150.000000 150.000000

| mean | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| --- | --- | --- | --- | --- |
| std | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

**First** **Few** **Rows** **of** **the** **Dataset:**

**sepal\_length** **sepal\_width** **petal\_length** **petal\_width** **class**

| **0** | **5.1** | **3.5** | **1.4** | **0.2** **Iris-setosa** |
| --- | --- | --- | --- | --- |
| **1** | **4.9** | **3.0** | **1.4** | **0.2** **Iris-setosa** |
| **2** | **4.7** | **3.2** | **1.3** | **0.2** **Iris-setosa** |
| **3** | **4.6** | **3.1** | **1.5** | **0.2** **Iris-setosa** |
| **4** | **5.0** | **3.6** | **1.4** | **0.2** **Iris-setosa** |

**Unique** **Classes:**

**['Iris-setosa'** **'Iris-versicolor'** **'Iris-virginica']**

**EX.NO:4.2 VARIABLE FILTERS**

**DEFINITION:**

Variable filters refer to mechanisms that selectively process, manipulate, or exclude specific variables or features in a dataset, aiding in data analysis, pattern recognition, and model training.

**PROGRAM**

import pandas as pd

**# Create a sample DataFrame**

data = {

'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 22],

'City': ['New York', 'San Francisco', 'Los Angeles']

}

df = pd.DataFrame(data)

**# Display the original DataFrame** print("Original DataFrame:")

print(df)

**# Filter specific variables (columns**) selected\_columns = ['Name', 'City'] filtered\_df = df[selected\_columns]

**# Display the filtered DataFrame** print("\nFiltered DataFrame:") print(filtered\_df)

**OUTPUT:**

**Original DataFrame:**

| Name | Age | City |
| --- | --- | --- |
| 0 Alice | 25 | New York |
| 1 Bob | 30 | San Francisco |
| 2 Charlie | 22 | Los Angeles |

**Filtered** DataFrame: Name City

* 1. Alice New York
  2. Bob San Francisco
  3. Charlie Los Angeles

**EX.NO:4.3 ROW FILTERS**

**DEFINITION**:

Row filters in Pandas involve selecting specific rows from a DataFrame based on certain conditions. This enables data manipulation by focusing on rows that meet specific criteria, enhancing analysis and exploration.

**PROGRAM**

import pandas as pd

**# Sample data**

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [20, 22, 21, 23],

'Grade': [85, 92, 78, 95]}

df = pd.DataFrame(data) print("Original DataFrame:") print(df)

**OUTPUT:**

Original DataFrame:

| Name | Age | Grade |
| --- | --- | --- |
| 0 Alice | 20 | 85 |
| 1 Bob | 22 | 92 |
| 2 Charlie | 21 | 78 |
| 3 David | 23 | 95 |

**EX.NO:4.3 DATA CLEANING**

**DEFINITIO**N

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in datasets. It involves handling missing values, removing duplicates, and ensuring data is accurate and reliable for analysis.

**PROGRAM**

import pandas as pd

**# Sample data with missing values and duplicates**

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Alice'],

'Age': [20, None, 21, 23, 22],

'Grade': [85, 92, 78, 95, 92]}

df = pd.DataFrame(data)

print("Original DataFrame:") print(df)

**OUTPUT:**

Original DataFrame:

| Name | Age | Grade |
| --- | --- | --- |
| 0 Alice | 20.0 | 85 |
| 1 Bob | NaN | 92 |
| 2 Charlie | 21.0 | 78 |
| 3 David | 23.0 | 95 |
| 4 Alice | 22.0 | 92 |

In this example, we have a DataFrame with missing values in the 'Age' column and a duplicate row for the name 'Alice.'

**RESULT:**

**EX.NO: 5 TIME SERIES**

**AIM**:

**To Perform Time Series Analysis and apply the various visualization techniques**

**DEFINITION**

Time series analysis involves analyzing and modeling data collected over time to identify patterns, trends, and make predictions. Here, I'll demonstrate time series analysis using Python with the pandas, matplotlib, and seaborn libraries. I'll use a simple example with a synthetic time series dataset.

Firstly, ensure you have the required libraries installed:

bash

**pip install pandas matplotlib seaborn**

Now, let's create a synthetic time series dataset and perform basic time series analysis:

**PROGRAM**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

from datetime import datetime, timedelta

**# Generate a synthetic time series dataset** np.random.seed(42)

date\_today = datetime.now()

days = pd.date\_range(date\_today, date\_today + timedelta(9), freq='D')

data = {'Date': days, 'Value': np.random.randint(10, 100,size=(len(days)))}

df = pd.DataFrame(data)

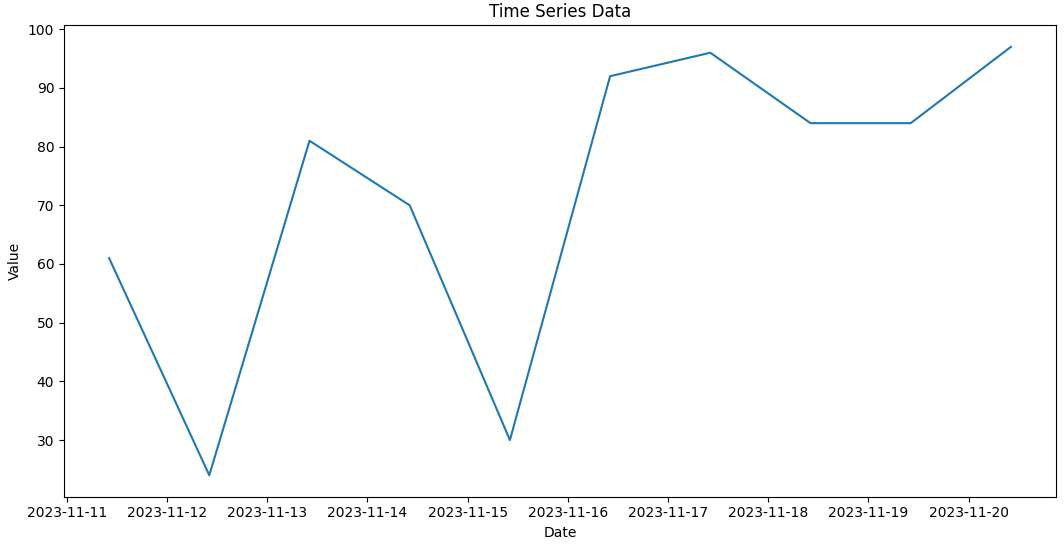
**# Display the dataset** print("Time Series Dataset:")

print(df.head())

**# Visualize the time series data** plt.figure(figsize=(10, 6)) sns.lineplot(x='Date', y='Value', data=df) plt.title('Time Series Data') plt.xlabel('Date')

plt.ylabel('Value') plt.show()

**OUTPUT:**



This will generate a synthetic time series dataset and plot it using a line chart.

Now, let's perform some basic time series analysis and visualization techniques:

**EX.NO:5.1 TREND ANALYSIS**

**DEFINITION:**

Trend analysis examines data over time to identify long-term patterns, helping discern upward, downward, or stable directional changes.

**PROGRAM**

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

**# Generate a synthetic time series dataset** np.random.seed(42)

date\_today = pd.to\_datetime('2023-01-01')

days = pd.date\_range(date\_today, date\_today + pd.to\_timedelta(365, unit='D'), freq='D') values = np.cumsum(np.random.randn(len(days)))

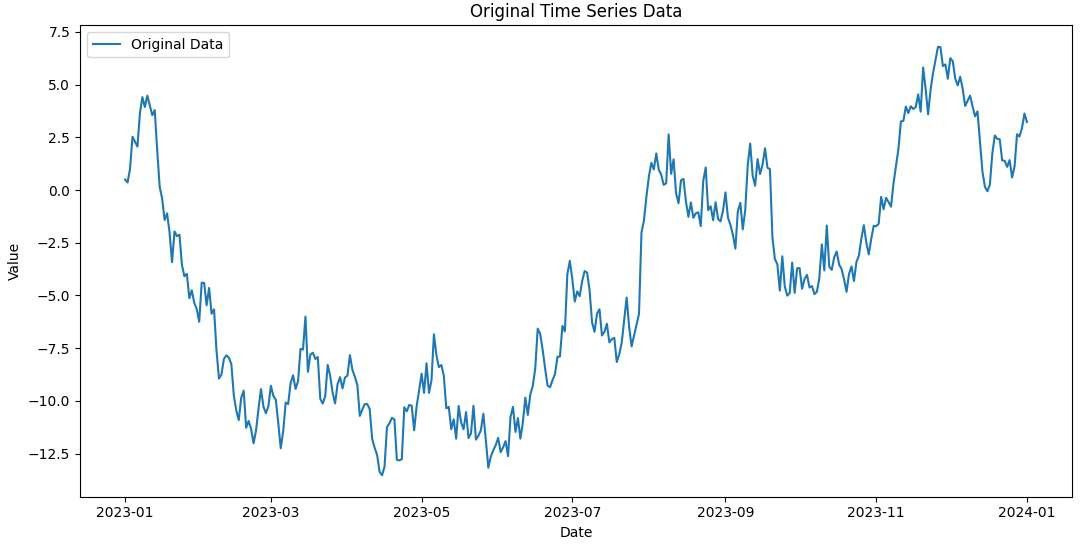
df = pd.DataFrame({'Date': days, 'Value': values})

**# Plot the original time series data** plt.figure(figsize=(10, 6))

plt.plot(df['Date'], df['Value'], label='Original Data') plt.title('Original Time Series Data') plt.xlabel('Date')

plt.ylabel('Value') plt.legend() plt.show()

**OUTPUT**:



**EX.NO : 5.2 SEASONAL DECOMPOSITION**

**DEFINITION:**

Seasonal decomposition separates a time series into components: trend, seasonal, and residual, aiding analysis by isolating underlying patterns and fluctuations.

**PROGRAM**

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

**# Generate a synthetic time series dataset with a clear seasonal component**

np.random.seed(42)

date\_today = pd.to\_datetime('2023-01-01')

days = pd.date\_range(date\_today, date\_today + pd.to\_timedelta(365, unit='D'), freq='D')

**# Creating a seasonal component**

seasonal\_component = np.sin(2 \* np.pi \* np.arange(len(days)) / 365 \* 7)

**# Generating synthetic data with trend and seasonal components**

trend\_component = np.cumsum(np.random.randn(len(days))) values = trend\_component + 10 \* seasonal\_component

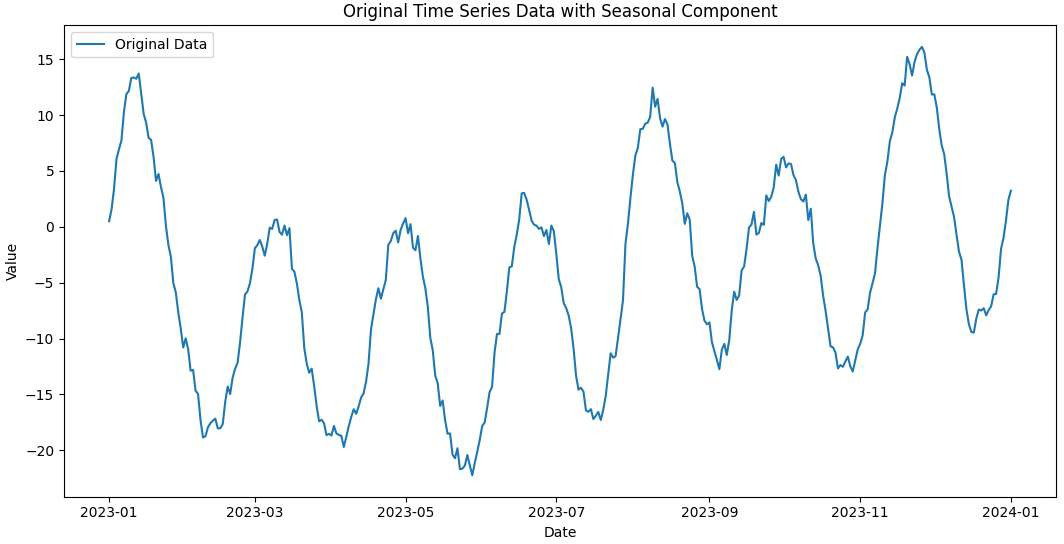
df = pd.DataFrame({'Date': days, 'Value': values})

**# Plot the original time series data** plt.figure(figsize=(10, 6))

plt.plot(df['Date'], df['Value'], label='Original Data') plt.title('Original Time Series Data with Seasonal Component') plt.xlabel('Date')

plt.ylabel('Value') plt.legend() plt.show()

**OUTPUT**:



**EX.NO : 6 INTERACTIVE MAP VISUALIZATION**

**AIM:**

**To Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction, etc..**

**DEFINITION**:

Performing data analysis and representation on a map involves visualizing geographic data in a way that provides insights into spatial patterns. A popular tool for this is the `folium` library in Python, which allows you to create interactive maps. Below is a basic example using `folium` with mouse rollover effect and user interaction.

**PROGRAM**

import folium

**# Load a GeoJSON dataset (you can use your own GeoJSON file or find one online)**

geojson\_data = "https://raw.githubusercontent.com/nvkelso/natural-earth-

vector/master/geojson/ne\_110m\_admin\_0\_countries.geojson"

**# Create a folium map centered around the mean latitude and longitude of the dataset** m = folium.Map(location=[0, 0], zoom\_start=2)

**# Add GeoJSON data with mouse rollover effect** folium.GeoJson(

geojson\_data, name='geojson',

style\_function=lambda x: {'fillColor': 'green', 'color': 'black'}, highlight\_function=lambda x: {'fillColor': 'yellow', 'color': 'blue'}, tooltip=folium.features.GeoJsonTooltip(fields=['name'], labels=False, sticky=False)

).add\_to(m)

**# Add layer control for user interaction** folium.LayerControl().add\_to(m)

**# Display the map** m.save("interactive\_map\_with\_interaction.html")

**OUTPUT:**

When you run the script, it will create an HTML file (in this case, "interactive\_map\_with\_tooltip.html") in the same directory where you run the script.

Open this HTML file in a web browser to visualize the map with mouse rollover effects.

**RESULT:**

**EX.NO : 7 CARTOGRAPHIC VISUALIZATION**

**AIM:**

**To build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India etc.**

**DEFINITION**:

Creating cartographic visualizations for multiple datasets involving various countries, states, or districts often involves combining data with geographical boundaries. Here's a Python script that utilizes `geopandas` and `folium` to create visualizations for both world countries and states in India, along with fictional data for illustration

**PROGRAM**

import folium

**# Create a folium map centered around a specific location**

m = folium.Map(location=[20.5937, 78.9629], zoom\_start=5)

**# Add a marker for a few world countries**

folium.Marker([37.7749, -122.4194], popup='USA').add\_to(m)

folium.Marker([35.8617, 104.1954], popup='China').add\_to(m)

folium.Marker([20.5937, 78.9629], popup='India').add\_to(m)

**# Add a marker for a few Indian states**

folium.Marker([19.7515, 75.7139], popup='Maharashtra').add\_to(m)

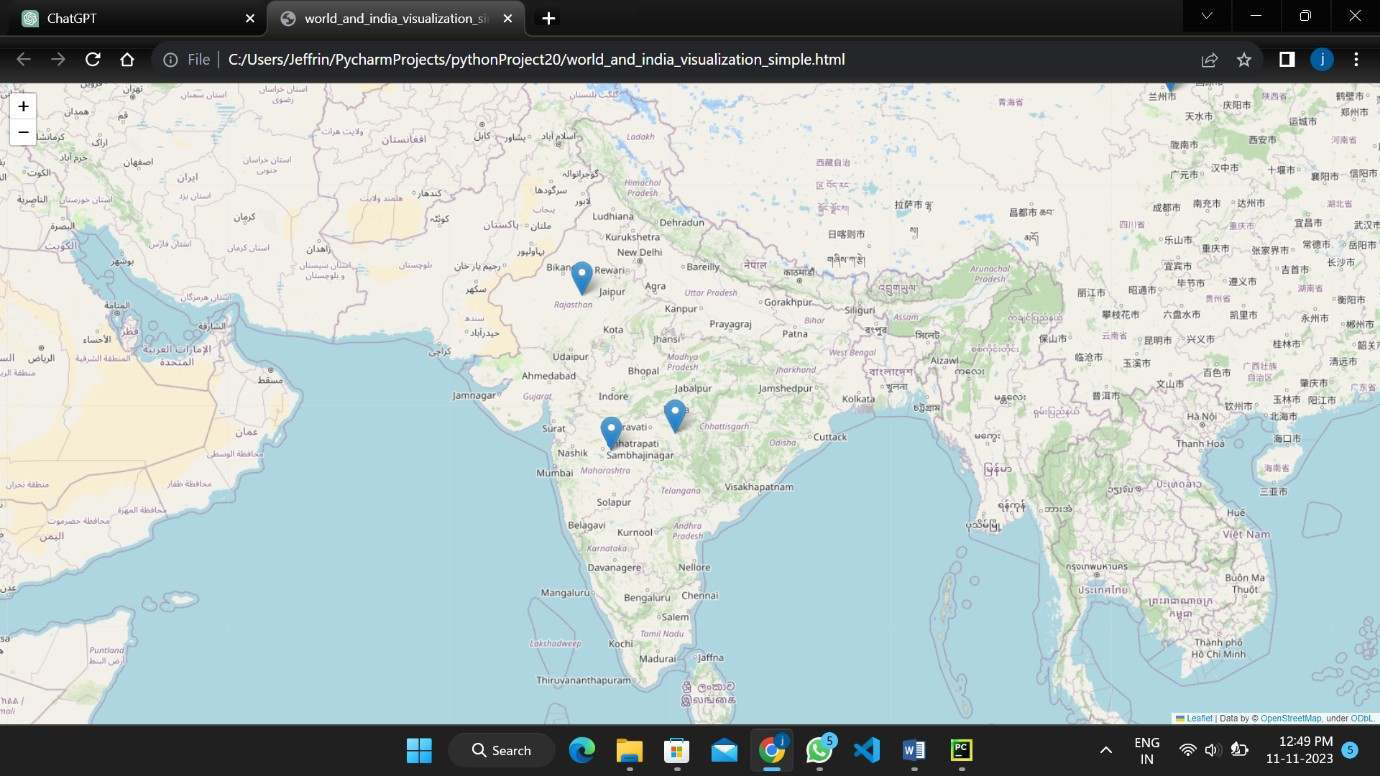
folium.Marker([27.0238, 74.2179], popup='Rajasthan').add\_to(m)

**# Save the map** m.save("world\_and\_india\_visualization\_simple.html")

**PROCEDURE**:

You need to run the provided code on your local machine to see the output. When you run the script, it will generate an HTML file named "world\_and\_india\_visualization\_simple.html" in the same directory where you saved the script.

OUTPUT:



**RESULT:**

**EX.NO : 8 EDA ON WINE QUALITY DATA SET**

**AIM:**

**To Perform EDA on Wine Quality Data Set**.

**DEFINITION**:

Exploratory Data Analysis (EDA) is a crucial step in understanding the characteristics of a dataset. Let's perform EDA on a wine quality dataset. For this example, I'll use the Wine Quality dataset available in the UCI Machine Learning Repository.

**PROGRAM**:

import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

**# Load the Wine Quality dataset**

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality- white.csv"

wine\_data = pd.read\_csv(url, sep=';')

**# Display the first few rows of the dataset** print(wine\_data.head())

**# Summary statistics** print(wine\_data.describe())

**# Distribution of Wine Quality** sns.countplot(x='quality', data=wine\_data) plt.title('Distribution of Wine Quality') plt.show()

**# Correlation heatmap** correlation\_matrix = wine\_data.corr() plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap') plt.show()

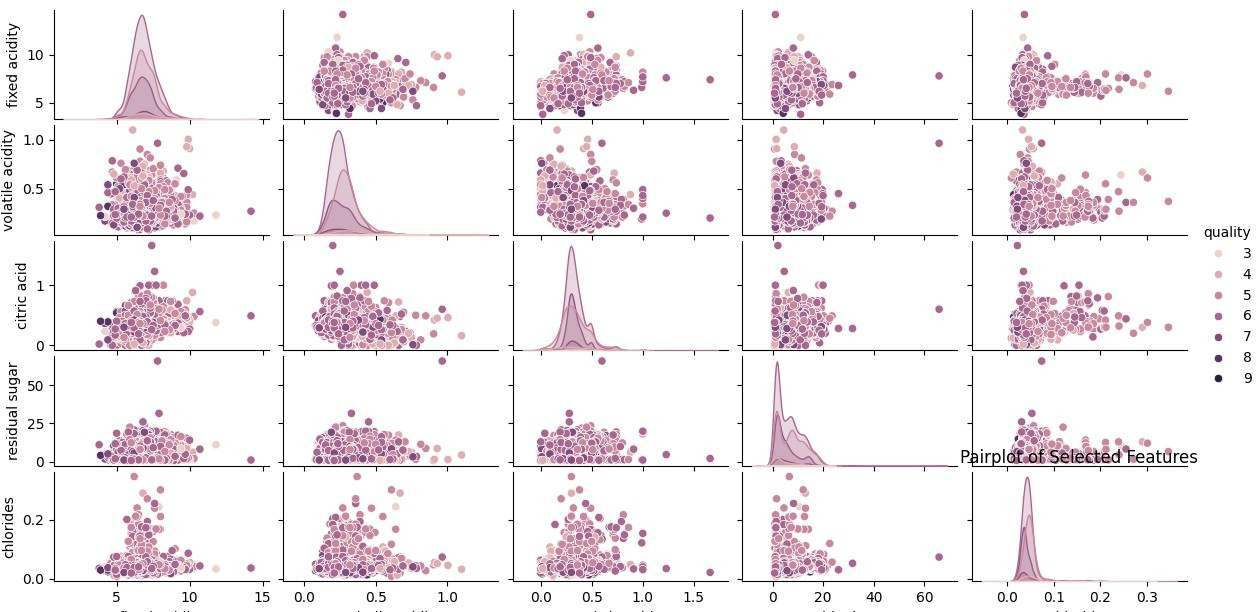
**# Pairplot for selected features**

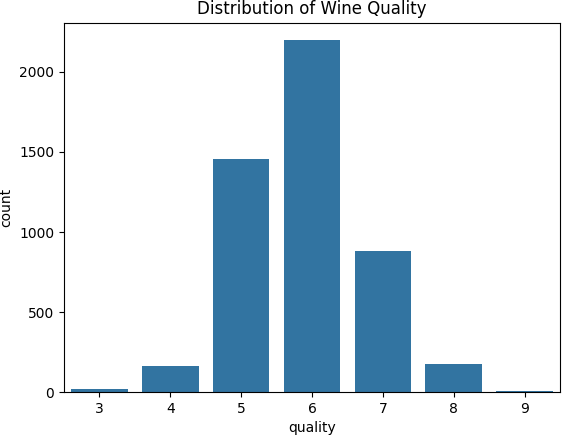
selected\_features = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'quality']

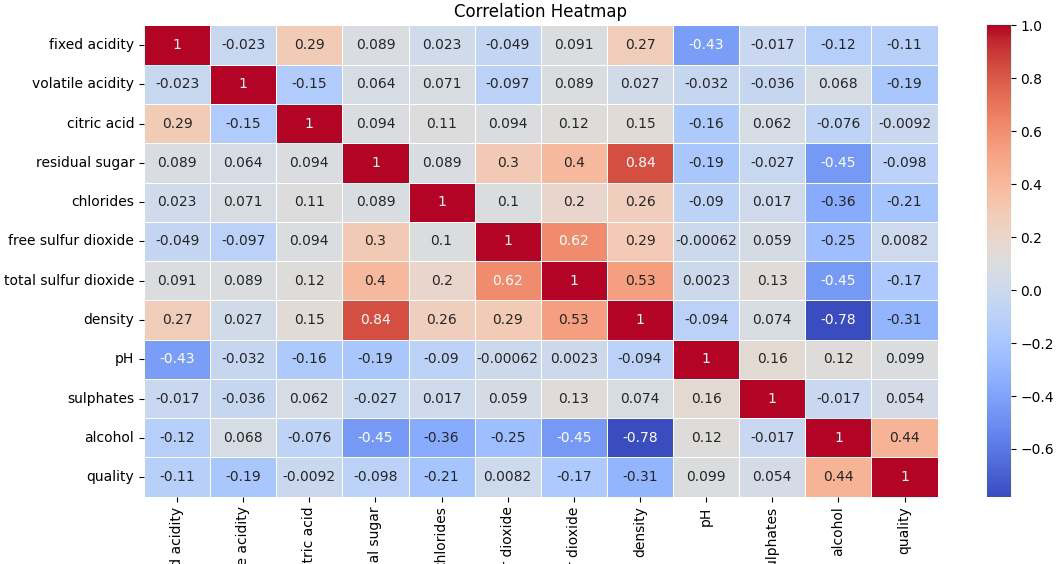
sns.pairplot(wine\_data[selected\_features], hue='quality', markers='o')

plt.title('Pairplot of Selected Features')

plt.show()

**OUTPUT:**





**RESULT:**

**EX.NO : 9 CASE STUDY ON A DATA SET TO PRESENT AN**

**ANALYSIS REPORT**

**AIM:**

**To Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.**

**DEFINITION**:

**PROGRAM**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**# Load the dataset (replace 'college\_students.csv' with your actual file path)**

df = pd.read\_csv('college\_students.csv')

**# Display the first few rows of the dataset**

print(df.head())

**# Check for missing values** print(df.isnull().sum())

**# Summary statistics**

print(df.describe())

**# Univariate Analysis**

# Histograms for numerical variables

df.plot(kind='hist', subplots=True, layout=(2, 2), figsize=(12, 10), bins=20, title='Histograms')

plt.show()

**# Bar plot for categorical variables (e.g., Gender)**

df['Gender'].value\_counts().plot(kind='bar', color=['skyblue', 'pink'])

plt.title('Distribution of Gender')

plt.xlabel('Gender') plt.ylabel('Count')

plt.show()

**# Bivariate Analysis**

**# Pair plot for numerical variables**

sns.pairplot(df, hue='Gender', markers=['o', 's'], height=3) plt.suptitle('Pair Plot of Numerical Variables by Gender', y=1.02)

plt.show()

**# Correlation heatmap**

correlation\_matrix = df.corr()

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()

**# Insights and Analysis**

**# You can print and analyze key insights based on the EDA performed**

**# Save the visualizations or export the analysis report as needed**

Csv file : Age,Gender,GPA,StudyHours,Grade 24,Female,2.76,24,A

21,Male,2.53,25,D

22,Male,3.24,14,D

24,Female,2.77,12,F

20,Female,3.05,21,B

22,Female,3.62,29,F

22,Female,3.58,13,B

24,Female,2.96,25,F

19,Female,3.31,16,C

20,Female,3.26,22,C

24,Female,3.45,19,C

20,Male,2.88,16,C

20,Female,3.38,23,A

22,Female,3.97,14,C

21,Male,3.23,12,D

20,Female,3.86,20,D

23,Male,3.15,20,A

22,Male,3.03,27,C

19,Female,3.47,24,C

21,Male,3.5,21,C

23,Male,3.8,18,F

23,Male,2.85,19,B

19,Male,3.25,21,F

21,Female,3.36,26,B

22,Male,3.65,15,C

18,Female,2.57,16,C

21,Male,3.99,23,F

19,Male,3.2,22,F

23,Male,2.92,17,B

22,Male,3.83,19,D

21,Female,3.62,18,B

18,Female,3.93,27,F

18,Male,3.0,11,F

20,Male,3.33,14,A

20,Female,3.36,14,F

24,Male,3.97,15,A

19,Male,2.61,28,D

21,Male,2.96,17,B

21,Female,2.79,25,B

24,Female,2.9,22,A

23,Female,3.23,10,B

23,Male,3.06,29,F

24,Male,3.09,26,C

23,Female,3.77,16,A

20,Female,3.9,22,B

21,Female,2.61,13,A

24,Female,2.81,13,A

21,Male,3.51,15,C

18,Female,3.04,28,F

20,Male,2.88,21,A

22,Female,2.94,16,B

20,Male,2.98,19,D

24,Female,3.77,28,A

22,Female,2.7,16,A

18,Female,3.56,12,C

24,Female,3.33,22,F

19,Male,2.94,22,D

21,Female,3.13,27,B

18,Male,2.88,29,D

21,Male,3.42,17,B

23,Male,2.62,18,F

19,Male,2.51,16,B

19,Female,3.44,10,C

18,Male,2.79,12,C

19,Male,2.61,22,C

22,Male,3.1,26,C

19,Female,2.58,10,D

21,Female,3.83,15,F

21,Female,2.54,15,B

24,Female,3.37,21,B

21,Male,3.16,22,C

24,Male,3.51,22,C

21,Female,2.99,24,A

22,Male,2.73,25,F

24,Male,3.97,20,D

20,Male,3.76,14,B

23,Female,3.79,13,A

18,Female,2.88,12,A

21,Male,2.56,28,B

19,Female,2.95,29,D

21,Female,3.31,27,A

19,Female,2.99,24,A

23,Female,3.74,18,F

23,Female,2.91,26,D

23,Male,3.95,23,A

19,Female,3.19,24,D

21,Male,3.76,10,B

23,Male,2.79,12,C

22,Female,3.12,25,A

24,Male,3.55,20,F

19,Male,2.71,21,B

19,Male,2.7,19,D

21,Female,3.95,25,B

19,Female,3.57,17,A

19,Male,2.56,15,D

23,Female,3.1,21,C

21,Female,3.15,17,B

23,Female,3.62,13,A

24,Female,2.88,17,F

24,Male,2.78,27,D

23,Male,2.62,14,B

24,Female,3.14,18,B

21,Female,3.53,13,C

18,Male,2.59,26,C

23,Male,3.87,18,F

22,Female,3.16,10,F

22,Male,2.86,29,A

19,Female,2.64,22,A

24,Female,2.77,25,F

22,Male,3.9,22,F

19,Male,3.46,23,D

18,Female,3.28,12,C

21,Female,3.49,15,A

21,Male,3.15,27,C

21,Female,3.6,28,C

22,Male,2.57,14,F

18,Female,3.35,24,D

22,Male,2.74,11,B

24,Male,2.68,19,D

22,Male,3.01,27,D

18,Female,2.64,22,C

18,Female,2.64,14,D

24,Male,2.97,10,A

18,Female,3.97,10,C

18,Male,2.76,27,A

21,Male,2.53,24,B

24,Female,3.65,26,C

20,Female,3.71,20,B

20,Male,3.02,26,C

18,Female,3.2,22,F

20,Female,3.47,10,D

20,Male,2.57,11,F

18,Male,3.92,18,B

20,Female,3.83,12,D

22,Male,2.89,10,C

19,Male,2.52,25,D

24,Female,3.9,15,A

19,Male,3.25,26,D

18,Male,3.31,14,A

21,Female,3.53,14,D

24,Female,3.42,15,A

18,Male,3.92,12,B

21,Male,3.92,14,F

19,Female,3.8,14,C

18,Female,3.45,19,D

24,Female,3.7,19,F

24,Female,3.52,28,C

23,Male,3.36,26,C

22,Female,2.69,23,A

20,Female,3.72,18,B

21,Male,3.73,23,B

23,Female,3.44,10,F

20,Male,3.73,28,B

20,Female,3.48,22,D

18,Female,2.81,22,B

20,Female,2.91,13,F

22,Male,2.82,10,B

24,Male,3.07,26,D

23,Female,2.56,17,A

20,Male,3.43,11,F

18,Male,3.0,17,A

22,Male,3.48,16,A

19,Male,3.08,11,A

24,Male,3.52,12,C

24,Male,3.01,27,C

23,Female,2.89,21,A

24,Female,3.24,10,F

20,Male,3.54,21,D

18,Female,3.02,14,D

24,Female,3.9,26,B

24,Female,2.56,25,F

19,Female,3.13,24,C

19,Male,3.95,24,A

21,Female,3.32,14,B

22,Female,3.14,23,D

20,Male,3.35,11,C

24,Male,3.36,20,C

24,Female,3.6,28,A

18,Female,2.69,16,D

21,Female,2.88,15,F

22,Female,3.37,11,C

21,Female,3.8,15,A

23,Female,3.34,27,F

22,Male,2.86,11,D

24,Female,3.52,27,C

24,Female,3.61,24,F

22,Male,2.86,28,F

24,Male,3.07,11,F

20,Male,3.3,29,C

22,Male,3.24,15,C

21,Female,3.08,10,B

22,Female,2.95,24,D

24,Female,2.65,19,A

20,Female,2.58,28,F

20,Female,3.94,26,B

23,Female,3.77,14,A

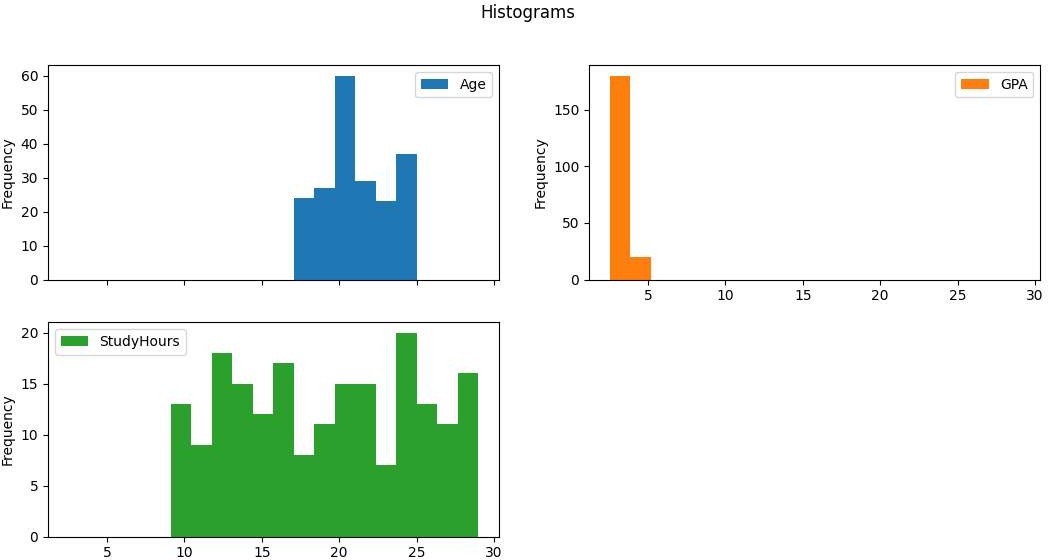
21,Male,3.03,13,B

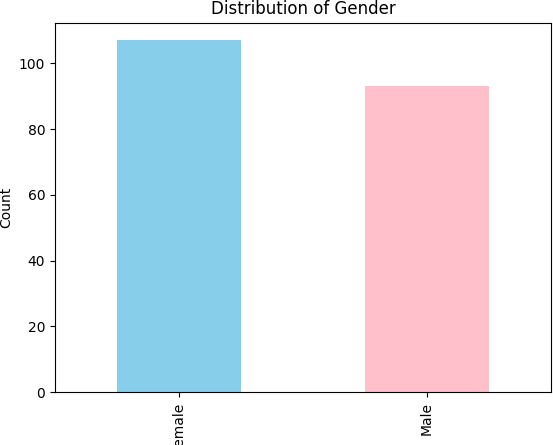
19,Female,3.94,19,C

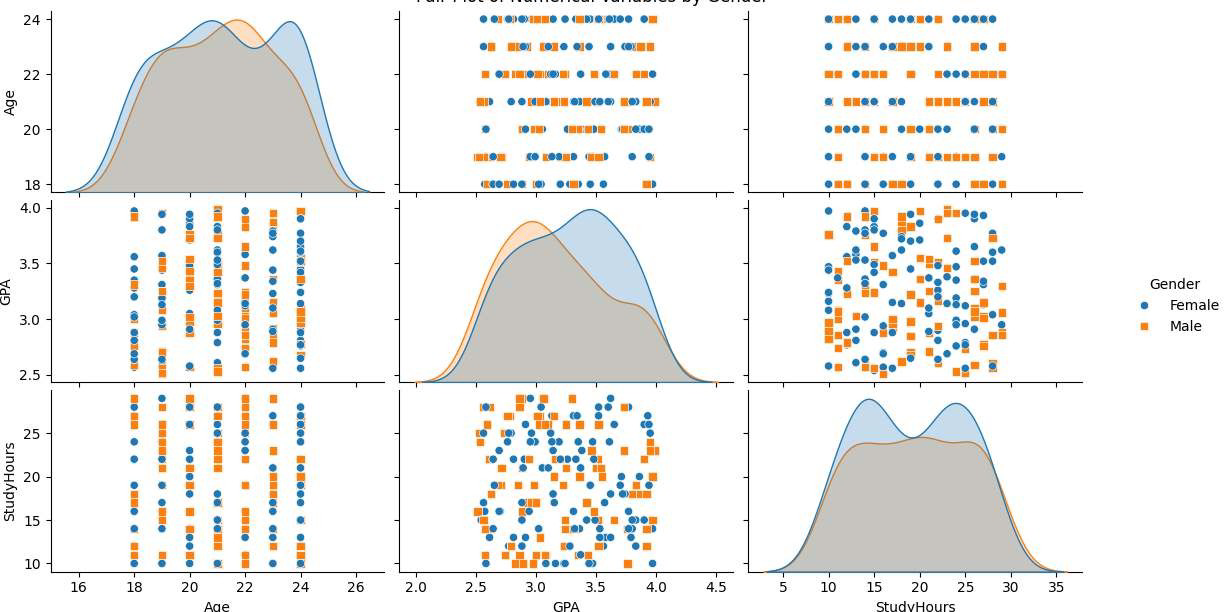
19,Male,3.52,26,F

22,Male,3.22,19,A

**OUTPUT**:







**RESULT:**