RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

AD23632 - FRAMEWORK FOR DATA AND VISUAL ANALYTICS

Laboratory Note Book

NAME: HARSHAVARDHAN S

YEAR/ BRANCH / SECTION: 3nd YEAR / AIML

UNIVERSITY REGISTER NO. :2116231501504

COLLEGE ROLL NO: 231501504

SEMESTERS: 5TH SEMESTER

ACADEMIC YEAR: 2025-2026

List of Experiments	
1	**Setting up the Python environment and libraries-Juypter Notebook** • Create a new notebook for Python • Write and execute Python code • Create new cells for code and Markdown • Demonstrate the application of Jupyter Widgets, Jupyter AI
2	**EDA-Data Import and Export** • Importing data from CSV, Excel, SQL databases, and web scraping • Handling different data formats • Export a DataFrame to an Excel file.
3	**EDA-Data Cleaning** • Handling missing values: detection, filling, and dropping • Removing duplicates and unnecessary data • Data type conversion and ensuring consistency • Normalize data (e.g., standardization, min-max scaling).
4	**EDA-Data Inspection and Analysis** • Viewing and inspecting Data Frames • Filtering and subsetting data using conditions • Descriptive statistics: measures of central tendency (mean, median, mode) and measures of dispersion (range, variance, standard deviation)
5	**EDA-Data Visualization with Matplotlib** Basic plotting: line charts, bar charts, histograms
6	**Data Visualization Using PowerBI** • Learning the Power BI Interface • Connecting to various data sources (Excel, CSV, SQL databases) • Creating basic visualizations: bar charts, line charts, pie charts • Creating Calculated Columns and Measures

	Building Dashboards
7	**Data Visualization Using Tableau** • Introduction to Tableau and its interface • Connecting to various data sources (Excel, CSV, SQL databases) • Creating basic visualizations: bar charts, line charts, pie charts • Creating calculated fields Building dashboards and stories in Tableau
8	**Mini Project** • Healthcare Data Analytics • Financial Data Analytics • Social Media Data Analytics • Sports analytics • Tourism Analytics

Requirements	
Hardware	Intel i3, CPU @ 1.20GHz 1.19 GHz, 4 GB RAM, 32 Bit Operating System
Software	python3.7,Jupiter
Operating System	Windows

AD23632 - FRAMEWORK FOR DATA AND VISUAL ANALYTICS

LAB LESSON PLAN

PRACTICAL SESSIONS				
S.NO	**Experiment**	**Alignment with Theory**	**Resources Needed**	**Required Hours**
1	Setting up the Python environment and libraries- Jupyter Notebook	UNIT-II (**EXPLORATORY DATA ANALYSIS USING PYTHON**)	Python, Jupyter Notebook, ipywidgets, pandas	3
2	EDA-Data Import and Export	UNIT-II (**EXPLORATORY DATA ANALYSIS USING PYTHON**)	Python, Jupyter Notebook, matplot,	3
3	EDA-Data Cleaning	UNIT-II (**EXPLORATORY DATA ANALYSIS USING PYTHON**)	Python, Jupyter Notebook	3
4	EDA-Data Inspection and Analysis	UNIT-III (**VISUALIZATION IN PYTHON**)	Python, TensorFlow/Keras	3
5	EDA-Data Visualization with Matplotlib	UNIT-III (**VISUALIZATION IN PYTHON**)	Python, TensorFlow/Keras	3
6	Data Visualization Using Power BI	UNIT-III (**VISUALIZATION USING POWER BI**)	Python, Scikit-Learn	3
7	Data Visualization Using Tableau	UNIT-III (**VISUALIZATION USING TABLEAU TOOL**)	Python, Scikit-Learn	3
8	Mini Project • Healthcare Data Analytics • Financial Data Analytics • Social Media Data Analytics • Sports analytics • Tourism	All Units	Python, Scikit-Learn, XGBoost	3

EXP NO:1	SETTING UP THE PYTHON ENVIRONMENT AND JUPYTER NOTEBOOK

AIM:

To set up a Python environment using Jupyter Notebook and demonstrate code execution, Markdown formatting, and the use of Jupyter Widgets and Jupyter AI.

PROBLEM STATEMENT:

Create a Jupyter Notebook that showcases Python code execution, Markdown documentation, interactive widgets, and AI-assisted features.

ALGORITHM:

1. Install Jupyter Notebook using pip install notebook.
2. Launch Jupyter using jupyter notebook.
3. Create a new Python 3 notebook.
4. Add and execute Python code cells.
5. Add Markdown cells for headings, lists, and descriptions.
6. Install and use ipywidgets for interactivity.
7. Explore Jupyter AI

IPYTHON WIDGETS

It is a Python library that lets you create interactive user interface controls in Jupyter Notebooks, JupyterLab, and JupyterLite.

THESE CONTROLS INCLUDE:

- Sliders
- Dropdowns
- Buttons
- Text boxes
- Date pickers
- File uploads
- Tabs
- Layout containers

CODE:

jupyter --version

pip install ipywidgets

pip install jupyterlab-widgets

Step 1: Basic Python code

print("Hello, Jupyter!")

Step 2: Markdown cell (add this in a Markdown cell, not code)

Welcome to Jupyter Notebook

This is a Markdown cell. You can write **bold**, *italic*, or `code`.

Step 3: Jupyter Widgets

import ipywidgets as widgets

widgets.IntSlider(description='Slider:', min=0, max=100, step=5)

Output:

```
# Python code cell
print("Hello, Jupyter!")

# Markdown cell
# ## This is a Markdown Heading

# Jupyter Widgets
import ipywidgets as widgets
widgets.IntSlider()
```

```
Hello, Jupyter!
```

0

Jupyter Widgets

import ipywidgets as widgets

from IPython.display import display

Create an IntSlider widget for age

age = widgets.IntSlider(

description="Age:",

min=0,

max=100,

```
    value=25
)
# Display the slider
display(age)
```

Output:

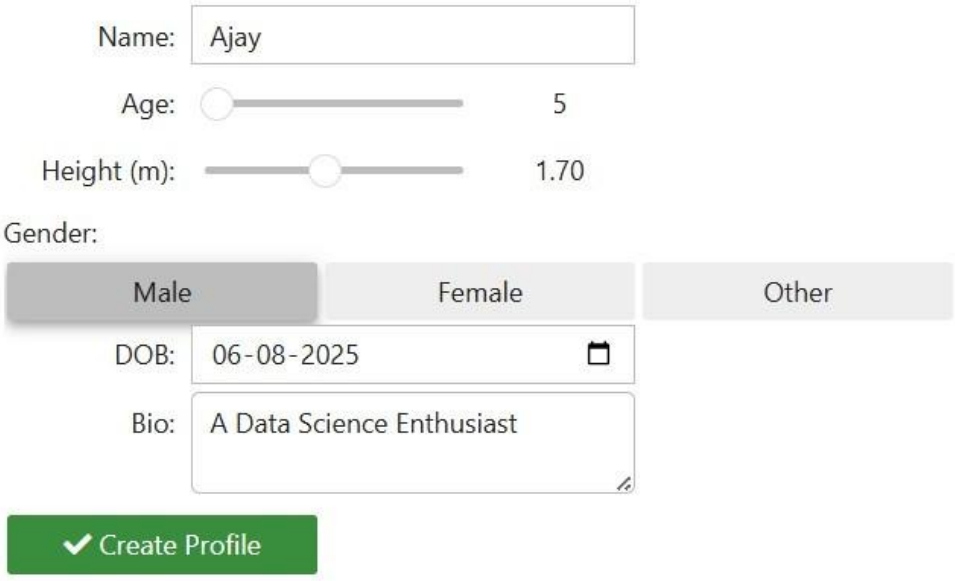
Code:

```
import ipywidgets as widgets
from IPython.display import display, clear_output
# Personal Info Widgets
name = widgets.Text(
    description="Name:",
    placeholder="Enter your name"
)
age = widgets.IntSlider(
    description="Age:",
    min=0, max=100, value=25
)
gender = widgets.ToggleButtons(
    options=['Male', 'Female', 'Other'],
    description='Gender:'
)
birthdate = widgets.DatePicker(
    description='DOB:'
)
height = widgets.FloatSlider(
    description="Height (m):",
    min=1.0, max=2.5, step=0.01, value=1.70
)
```

```
bio = widgets.Textarea(
    description="Bio:",
    placeholder="Write something about yourself"
)
# Output display
profile_output = widgets.Output()
# Submit button
submit_btn = widgets.Button(
    description="Create Profile",
    button_style='success',
    icon='check'
)
# Event handler
def on_submit(b):
    with profile_output:
        clear_output()
        print(“ Profile Summary \n")
        print(f"Name: {name.value}")
        print(f"Age: {age.value}")
        print(f"Height: {height.value} m")
        print(f"Gender: {gender.value}")
        print(f"Date of Birth: {birthdate.value}")
        print(f"Bio: {bio.value}")
submit_btn.on_click(on_submit)
# Layout (No Tabs)
form = widgets.VBox([
    name,
    age,
    height,
```

```
    gender,
    birthdate,
    bio,
    submit_btn,
    profile_output
])

# Display the form
display(form)
```

Output:



RESULT:

Thus, the program successfully created a Jupyter Notebook showcasing Python code execution, Markdown formatting, and the use of interactive widgets.

EXP NO: 2	EDA – DATA IMPORT AND EXPORT

AIM

To import data from CSV, Excel, and SQL databases and export DataFrames.

Problem Statement

LOAD datasets in multiple formats and export a DataFrame to Excel.

ALGORITHM

Step 1: Import Required Libraries

- Import pandas for data manipulation.
- Import sqlite3 for database handling.
- Import requests and BeautifulSoup for web scraping.

Step 2: Import Data from CSV File

- Use pd.read_csv(filename) to load data from a CSV file into a DataFrame.
- Display the first few rows using .head().

Step 3: Import Data from Excel File

- Use pd.read_excel(filename) to load data from an Excel file.
- Display the first few rows using .head().

Step 4: Import Data from SQL Database

- Connect to or create an SQLite database using sqlite3.connect().
- Create a table (if not already exists).
- Insert sample records (if needed).
- Use pd.read_sql_query(query, connection) to load table data into a DataFrame.

Step 5: Import Data from the Web (Web Scraping)

- Use requests.get(url) to fetch HTML content.
- Parse HTML with BeautifulSoup.
- Locate the desired table using soup.find() or soup.find_all().
- Convert the HTML table to a DataFrame using pd.read_html().

Step 6: Handle Different Data Formats

- Check for data type issues or format mismatches.
- Convert date columns using pd.to_datetime().
- Convert categorical or boolean fields using .astype().

Step 7: Export Data to Excel File

- Use DataFrame.to_excel(filename, index=False) to save a DataFrame to an Excel file.
- Confirm export success with a print statement.

SAMPLE CODE

```
# Import necessary libraries
import pandas as pd
import sqlite3
import requests
from bs4 import BeautifulSoup

# 1. Importing data from CSV
csv_df = pd.read_csv('Iris.csv')
print("CSV Data:")
print(csv_df.head())
```

```
CSV Data:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

```
# 2. Importing data from Excel
excel_df = pd.read_excel('heart stalog dataset.xlsx')
print("\nExcel Data:")
excel_df.head(5)
```

Excel Data:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_electroc
0	70	1	4	130	322	0	
1	67	0	3	115	564	0	
2	57	1	2	124	261	0	
3	64	1	4	128	263	0	
4	74	0	2	120	269	0	

```
#import from SQL Database
import sqlite3
# Connect to (or create) the database
conn = sqlite3.connect('my_database.db')
cursor = conn.cursor()

# Create the 'employees' table
cursor.execute('''
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY,
    name TEXT,
    department TEXT,
    salary REAL,
    hire_date TEXT
)
''')

# Insert example records
cursor.executemany('''
INSERT INTO employees (id, name, department, salary, hire_date) VALUES (?, ?, ?, ?, ?)
''', [
    (1, 'Alice Smith', 'HR', 55000, '2018-05-01'),
```

```
    (2, 'Bob Johnson', 'IT', 72000, '2019-07-15'),
    (3, 'Carol White', 'Finance', 68000, '2017-09-30'),
    (4, 'David Brown', 'Marketing', 60000, '2020-02-10'),
    (5, 'Eva Green', 'IT', 75000, '2021-04-25'),
])

# Commit and close
conn.commit()
print("Database and 'employees' table created with sample data.")
```

Database and 'employees' table created with sample data

```
sql_df = pd.read_sql_query("SELECT * FROM employees", conn)
print(sql_df)
```

	id	name	department	salary	hire_date
0	1	Alice Smith	HR	55000.0	2018-05-01
1	2	Bob Johnson	IT	72000.0	2019-07-15
2	3	Carol White	Finance	68000.0	2017-09-30
3	4	David Brown	Marketing	60000.0	2020-02-10
4	5	Eva Green	IT	75000.0	2021-04-25

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# URL of the Wikipedia page
url = "https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population"

# Fetch the page
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

```
# Find the first table with class 'wikitable' (Wikipedia uses this)
html_table = soup.find("table", {"class": "wikitable"})

# Use pandas to read the HTML table into a DataFrame
web_df = pd.read_html(str(html_table))[0]

# Show the first few rows
print("\nWeb Scraped Data:")
print(web_df.head())
```

```
Web Scraped Data:
        Location  Population % of world         Date  \
0          World  8232000000       100%  13 Jun 2025
1          India  1413324000      17.3%   1 Mar 2025
2          China  1408280000      17.2%  31 Dec 2024
3  United States   340110988       4.2%   1 Jul 2024
4      Indonesia   282477584       3.5%  30 Jun 2024

  Source (official or from the United Nations) Notes
0                        UN projection[1][3]   NaN
1                     Official projection[4]   [b]
2                       Official estimate[5]   [c]
3                       Official estimate[6]   [d]
4               National annual projection[7]   NaN
```

```
# 5. Handling different data formats
# For example, converting a date column to datetime
if 'date' in csv_df.columns:
   csv_df['date'] = pd.to_datetime(csv_df['date'])
datetime64[ns]
# 6. Export a DataFrame to Excel
# Here we export the CSV data as an example
csv_df.to_excel('exported_data.xlsx', index=False)
print("\nData exported to 'exported_data.xlsx' successfully.")
```

```
Data exported to 'exported_data.xlsx' successfully.
```

RESULT:

Thus, the program successfully created a Jupyter Notebook showcasing Python code to import data from CSV, Excel, and SQL databases, as well as export DataFrames.

EXP NO: 3	EDA-DATA CLEANING

AIM

To clean data by handling missing values, duplicates, data types, and normalization.

PROBLEM STATEMENT

Clean a dataset by removing nulls, duplicates, and normalizing numeric fields.

ALGORITHM

1. Load dataset.
2. Detect missing values (isnull).
3. Fill or drop missing values.
4. Remove duplicates.
5. Convert data types.
6. Normalize numeric columns.

SAMPLE CODE

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import matplotlib.pyplot as plt


# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv')
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

```
df.shape
```

```
(1005, 8)
```

```
# Step 2: Handle Missing Values
# Detect
missing_info = df.isnull().sum()
print("Missing values:\n", missing_info)
```

```
Missing values:
 gender                        0
race/ethnicity                 0
parental level of education    7
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

```
# Fill or Drop (based on context)
df.fillna({
   'parental level of education': df['parental level of education'].mode()[0],
   'lunch': df['lunch'].mode()[0]
}, inplace=True)
missing_info = df.isnull().sum()
missing_info
```

```
gender                        0
race/ethnicity                0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

```
duplicates = df[df.duplicated()]
duplicates
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
1000	male	group D	some college	standard	none	76	64	66
1001	male	group C	associate's degree	standard	none	46	43	42
1002	female	group B	bachelor's degree	standard	none	67	86	83
1003	male	group E	some high school	standard	none	92	87	78
1004	male	group C	bachelor's degree	standard	completed	83	82	84

```
duplicates.shape
```

(5, 8)

```
# Drop duplicates
df.drop_duplicates(inplace=True)
df.shape
# Step 4: Convert Data Types (if needed)
# For consistency, make sure string columns are lowercase
categorical_cols = ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']
for col in categorical_cols:
    df[col] = df[col].astype(str).str.lower().str.strip()
categorical_cols
```

['gender',

'race/ethnicity',

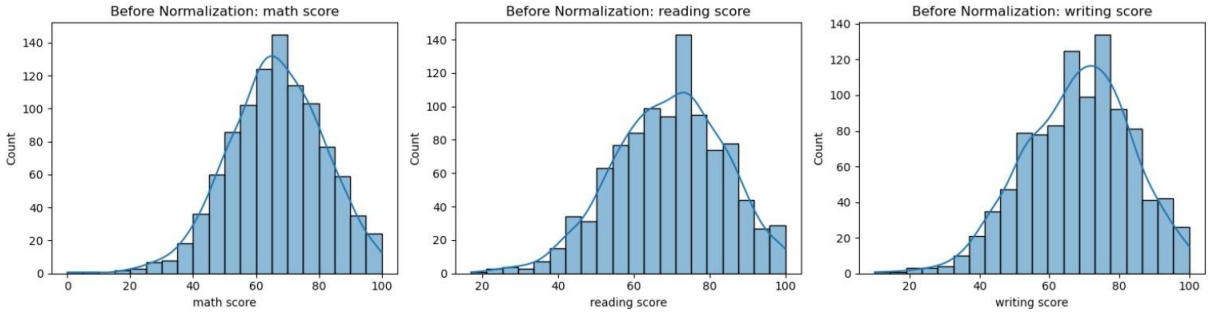
'parental level of education',

'lunch',

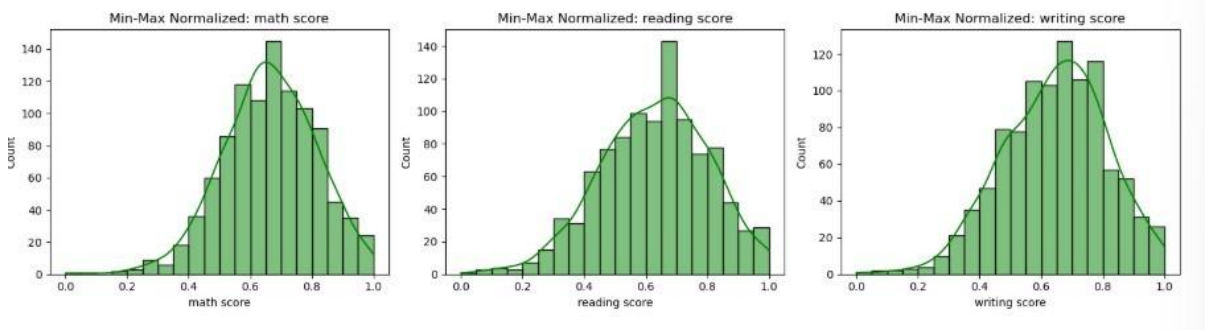
'test preparation course']

```
numeric_cols = ['math score', 'reading score', 'writing score']
numeric_cols
```

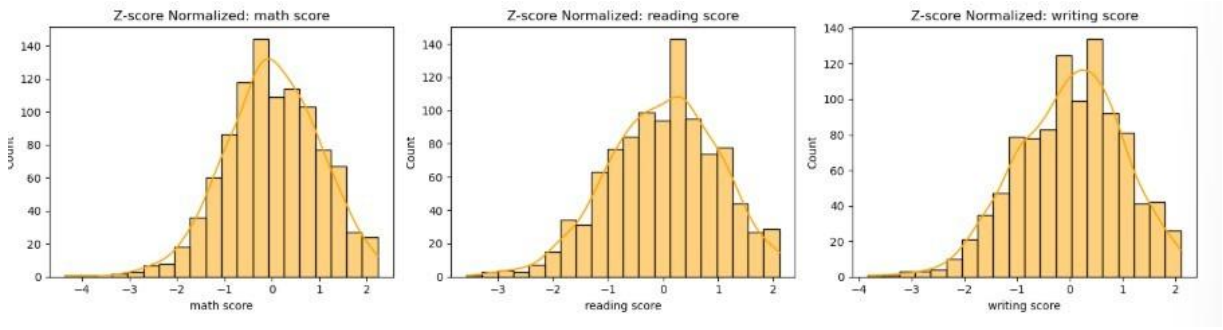
```
['math score', 'reading score', 'writing score']
plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(1, 3, i+1)
    sns.histplot(df[col], kde=True, bins=20)
    plt.title(f'Before Normalization: {col}')
plt.tight_layout()
plt.show()
```




```
minmax_scaler = MinMaxScaler()
df_minmax = df.copy()
df_minmax[numeric_cols] = minmax_scaler.fit_transform(df[numeric_cols])
plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(1, 3, i+1)
    sns.histplot(df_minmax[col], kde=True, bins=20, color='green')
    plt.title(f'Min-Max Normalized: {col}')
plt.tight_layout()
plt.show()
```




```
# Standard Scaling (Z-score)
zscore_scaler = StandardScaler()
df_zscore = df.copy()
df_zscore[numeric_cols] = zscore_scaler.fit_transform(df[numeric_cols])


plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(1, 3, i+1)
    sns.histplot(df_zscore[col], kde=True, bins=20, color='orange')
    plt.title(f'Z-score Normalized: {col}')
plt.tight_layout()
plt.show()
```



RESULT:

Thus, the program successfully created a Jupyter Notebook showcasing Python code handling missing values, removing duplicates and unnecessary data, Data type conversion and normalizing data.

EXP NO: 4	EDA-DATA INSPECTION AND ANALYSIS

AIM

To understand how to view, inspect, and summarize data stored in a DataFrame for initial exploration and analysis.

PROBLEM STATEMENT

Large datasets are hard to understand at first. To make them meaningful, we first view and inspect the data to know its structure, then filter and select only the required rows or columns, and finally calculate basic statistics like mean, median, and standard deviation to summarize the data.

ALGORITHM

Step 1: Import pandas and load/create the DataFrame.

Step 2: View data using head(), tail(), shape, dtypes, and info().

Step 3: Filter rows and select columns using conditions and logical operators.

Step 4: Calculate mean, median, mode, range, variance, and standard deviation.

Step 5: Interpret the results to find patterns and spread of data.

SAMPLE CODE

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import matplotlib.pyplot as plt

# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv')
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

df.head(3)

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93

df.tail()

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
1000	male	group D	some college	standard	none	76	64	66
1001	male	group C	associate's degree	standard	none	46	43	42
1002	female	group B	bachelor's degree	standard	none	67	86	83
1003	male	group E	some high school	standard	none	92	87	78
1004	male	group C	bachelor's degree	standard	completed	83	82	84

df.shape

(1005, 8)

df.columns.tolist()

['gender',

'race/ethnicity',

'parental level of education',

'lunch',

'test preparation course',

'math score',

'reading score',

'writing score']

df.dtypes

```
gender                        object
race/ethnicity                object
parental level of education   object
lunch                         object
test preparation course       object
math score                     int64
reading score                  int64
writing score                  int64
dtype: object
```

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1005 non-null   object
 1   race/ethnicity               1005 non-null   object
 2   parental level of education  998 non-null    object
 3   lunch                        1005 non-null   object
 4   test preparation course      1005 non-null   object
 5   math score                   1005 non-null   int64
 6   reading score                1005 non-null   int64
 7   writing score                1005 non-null   int64
dtypes: int64(3), object(5)
```

```
df.describe()
```

	math score	reading score	writing score
count	1005.000000	1005.000000	1005.000000
mean	66.122388	69.185075	68.066667
std	15.173234	14.614215	15.199095
min	0.000000	17.000000	10.000000
25%	57.000000	59.000000	58.000000
50%	66.000000	70.000000	69.000000
75%	77.000000	80.000000	79.000000
max	100.000000	100.000000	100.000000

Step 3: Filtering and Subsetting Data

```
print("\n---- Filtering and Subsetting-----")
# Students with math score > 70
print("\nStudents with math score > 70:\n", df[df["math score"] > 70])
```

```
---- Filtering and Subsetting ----

Students with math score > 70:
      gender race/ethnicity parental level of education         lunch  \
0     female        group B           bachelor's degree      standard
2     female        group B             master's degree      standard
4       male        group C                some college      standard
5     female        group B          associate's degree      standard
6     female        group B                some college      standard
...      ...            ...                         ...           ...
995   female        group E             master's degree      standard
999   female        group D                some college  free/reduced
1000    male        group D                some college      standard
1003    male        group E            some high school      standard
1004    male        group C           bachelor's degree      standard

     test preparation course  math score  reading score  writing score
0                       none          72             72             74
2                       none          90             95             93
4                       none          76             78             75
5                       none          71             83             78
6                  completed          88             95             92
...                      ...         ...            ...            ...
995                completed          88             99             95
999                     none          77             86             86
1000                    none          76             64             66
1003                    none          92             87             78
1004               completed          83             82             84

[394 rows x 8 columns]
```

```
# Female students only
print("\nFemale students:\n", df[df["gender"] == "female"])
```

```
Female students:
      gender race/ethnicity parental level of education         lunch  \
0     female        group B           bachelor's degree      standard
1     female        group C                some college      standard
2     female        group B              master's degree      standard
5     female        group B            associate's degree      standard
6     female        group B                some college      standard
...      ...            ...                         ...           ...
995   female        group E              master's degree      standard
997   female        group C                   high school  free/reduced
998   female        group D                some college      standard
999   female        group D                some college  free/reduced
1002  female        group B           bachelor's degree      standard

     test preparation course  math score  reading score  writing score
0                       none          72             72             74
1                  completed          69             90             88
2                       none          90             95             93
5                       none          71             83             78
6                  completed          88             95             92
...                      ...         ...            ...            ...
995                completed          88             99             95
997                completed          59             71             65
998                completed          68             78             77
999                     none          77             86             86
1002                    none          67             86             83

[519 rows x 8 columns]
```

Select only 'gender' and 'math score' columns

print("\nSubset with gender and math score:\n", df[["gender", "math score"]])

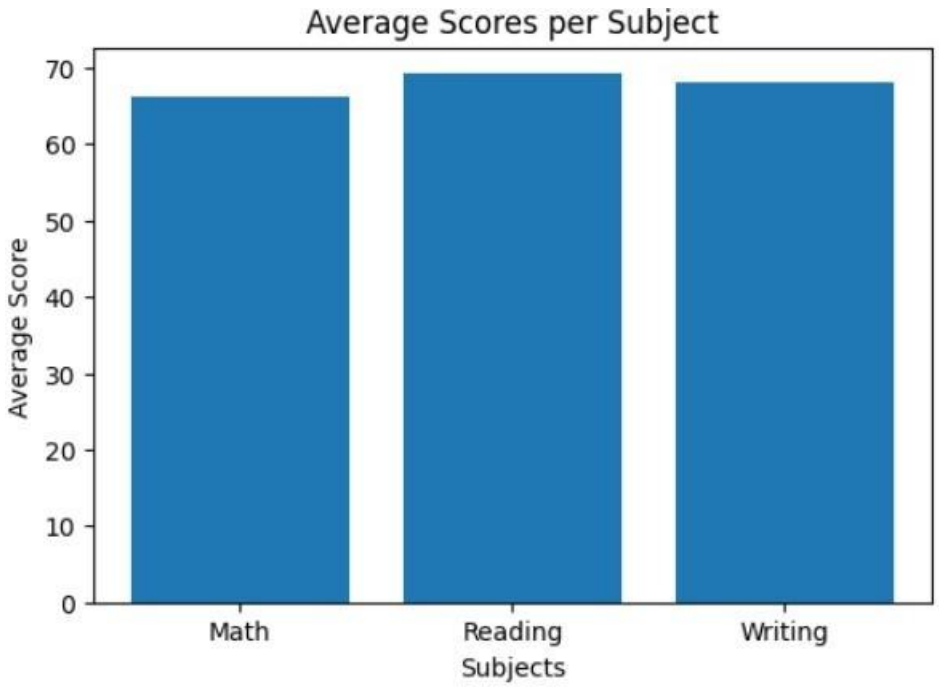
```
Subset with gender and math score:
      gender  math score
0     female          72
1     female          69
2     female          90
3       male          47
4       male          76
...      ...         ...
1000    male          76
1001    male          46
1002  female          67
1003    male          92
1004    male          83

[1005 rows x 2 columns]
```

print("\n---- Descriptive Statistics ---- ")

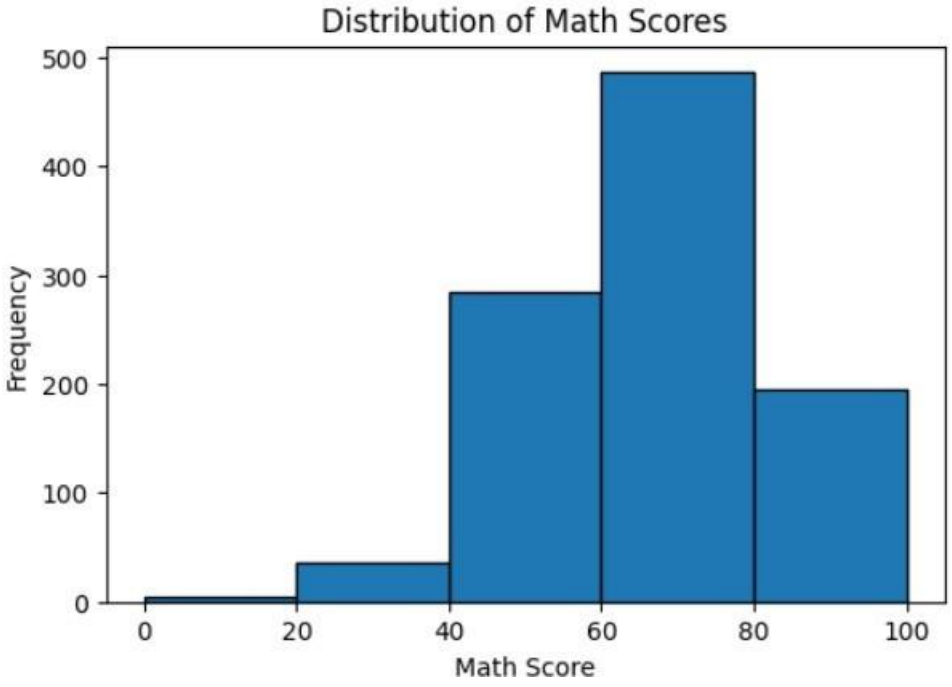
math_scores = df["math score"]

```
mean = math_scores.mean()
median = math_scores.median()
mode = math_scores.mode()[0]  # mode() returns a Series

_range = math_scores.max() - math_scores.min()
variance = math_scores.var()
std_dev = math_scores.std()

print(f"\nMean (Math Score): {mean}")
print(f"Median (Math Score): {median}")
print(f"Mode (Math Score): {mode}")
print(f"Range (Math Score): {_range}")
print(f"Variance (Math Score): {variance}")
print(f"Standard Deviation (Math Score): {std_dev}")
---- Descriptive Statistics ----

Mean (Math Score): 66.12238805970149
Median (Math Score): 66.0
Mode (Math Score): 65
Range (Math Score): 100
Variance (Math Score): 230.2270381161917
Standard Deviation (Math Score): 15.173234266832885

print("\n---- Visualization ----")

# 1. Bar chart: Average scores per subject
avg_scores = {
    "Math": df["math score"].mean(),
    "Reading": df["reading score"].mean(),
    "Writing": df["writing score"].mean()
```

```
}


plt.figure(figsize=(6, 4))
plt.bar(avg_scores.keys(), avg_scores.values())
plt.title("Average Scores per Subject")
plt.ylabel("Average Score")
plt.xlabel("Subjects")
plt.show()
```




```
# 2. Histogram: Distribution of math scores
plt.figure(figsize=(6, 4))
plt.hist(df["math score"], bins=5, edgecolor="black")
plt.title("Distribution of Math Scores")
plt.xlabel("Math Score")
plt.ylabel("Frequency")
plt.show()
```



3. Boxplot: Spread of math scores

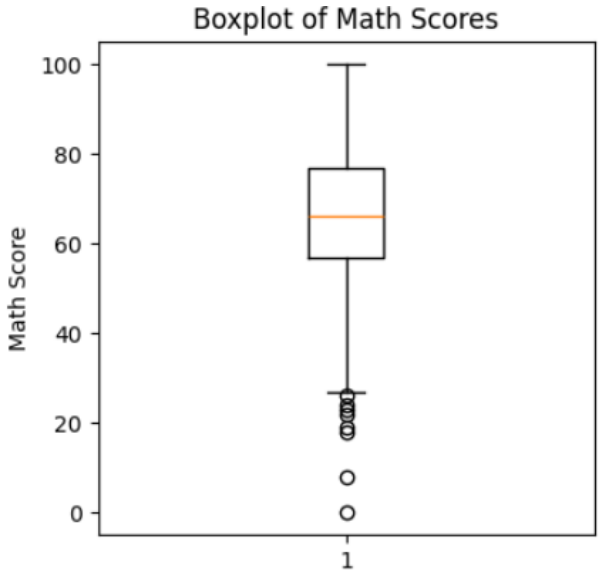
plt.figure(figsize=(4, 4))

plt.boxplot(df["math score"])

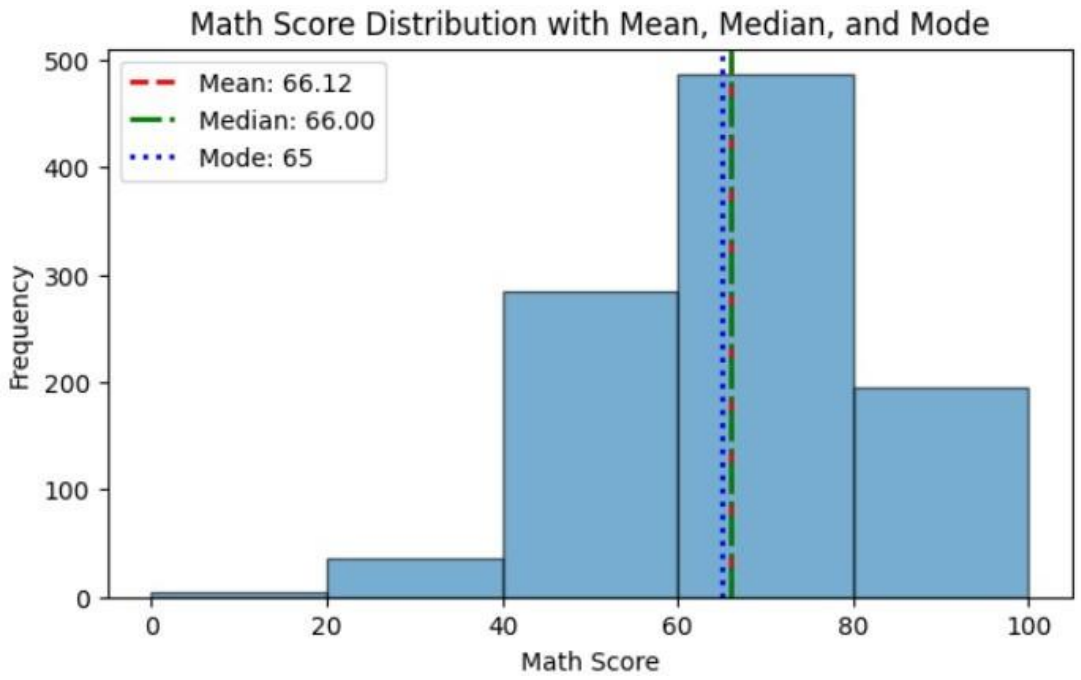
plt.title("Boxplot of Math Scores")

plt.ylabel("Math Score")

plt.show()



```
import matplotlib.pyplot as plt
# Plot Histogram with Mean, Median, and Mode Lines
plt.figure(figsize=(7, 4))
plt.hist(df["math score"], bins=5, edgecolor="black", alpha=0.6)
plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f"Mean: {mean:.2f}")
plt.axvline(median, color='green', linestyle='-.', linewidth=2, label=f"Median: {median:.2f}")
plt.axvline(mode, color='blue', linestyle=':', linewidth=2, label=f"Mode: {mode}")
plt.title("Math Score Distribution with Mean, Median, and Mode")
plt.xlabel("Math Score")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```



RESULT:

Thus, the Exploratory Data Analysis (EDA) was successfully performed by viewing, filtering, and summarizing the dataset. Data visualization was done using bar charts, histograms, and boxplots in Matplotlib to better understand the distribution and trends in the students' performance.

EXP NO: 5	EDA – DATA VISUALIZATION WITH MATPLOTLIB

AIM

The Python code aims to perform exploratory data analysis (EDA) by applying preprocessing steps and creating visualizations with Matplotlib. This helps to identify trends, compare group statistics, and observe data distributions using line charts, bar charts, and histograms.

PROBLEM STATEMENT

Raw datasets often contain large amounts of information that are not immediately meaningful. Without proper preprocessing and exploratory data analysis (EDA. Visualization techniques such as line charts, bar charts, and histograms help in summarizing the data and gaining insights.

ALGORITHM

Step 1: Import pandas for data handling, matplotlib for visualization, and sklearn scalers for preprocessing.

Step 2: Read the StudentsPerformance.csv dataset into a Pandas DataFrame.

Step 3: Display the first few rows of the dataset using df.head() to understand its structure.

Step 4: Group data by reading score and plot average math scores.

Step 5: Plot separate lines for categories (e.g., gender) for comparison.

Step 6: Group data by gender and calculate the average writing score.

Step 7: Plot a bar chart to compare gender-based averages.

Step 8: Plot the distribution of math scores to observe frequency patterns.

Step 9: Apply StandardScaler or MinMaxScaler for feature normalization if needed for further analysis.

Step 10: Analyze visualizations to identify trends, relationships, and score distributions.

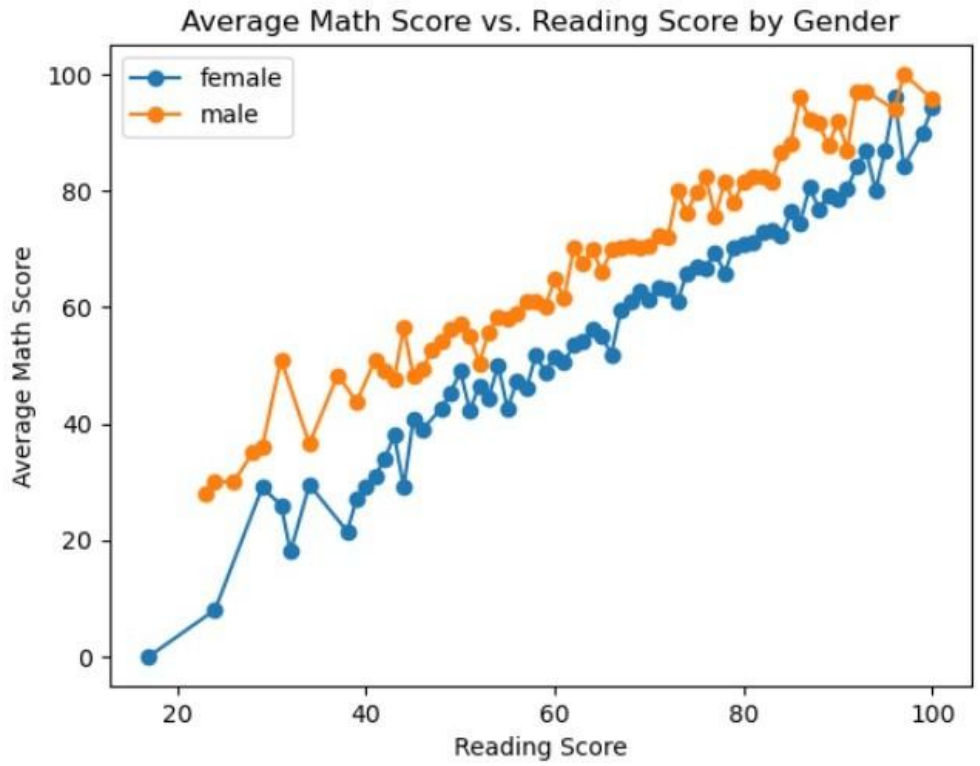
SAMPLE CODE

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

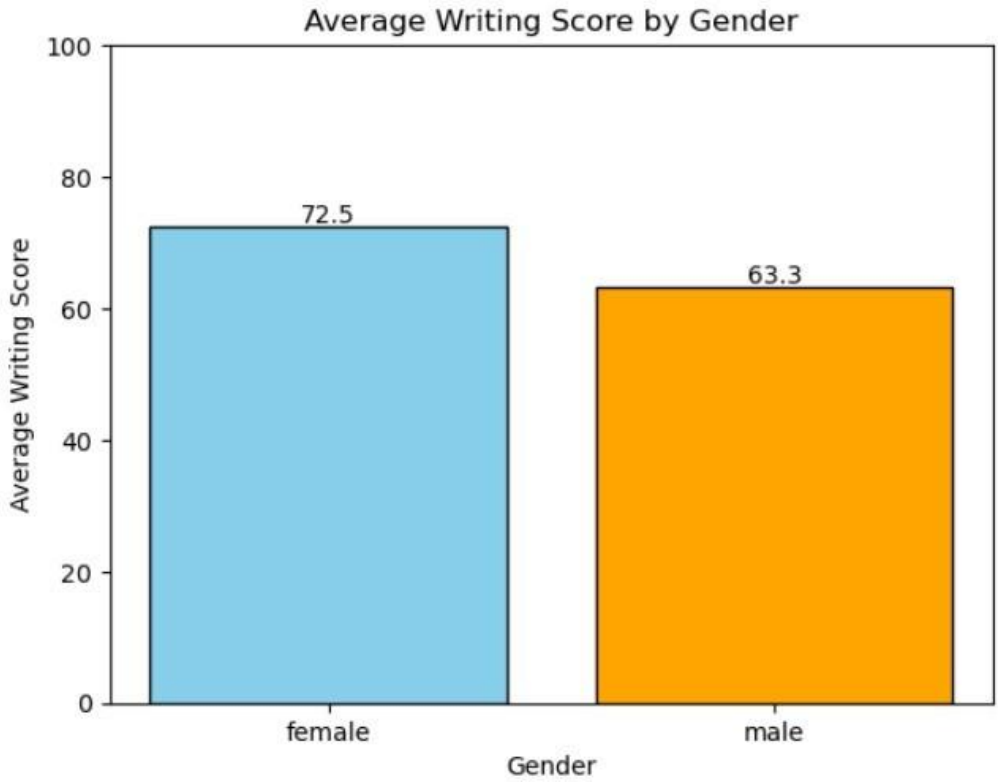
```
import matplotlib.pyplot as plt


# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv')
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

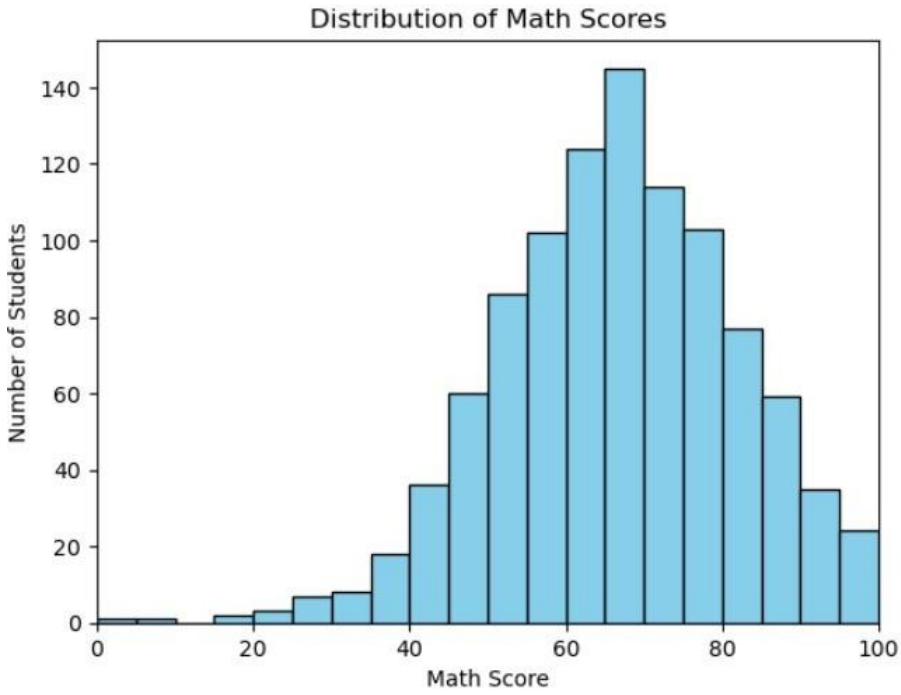
```
# Step 2: Line Chart - Average math score across reading score levels by gender
for gender in df["gender"].unique():
    avg_scores = df[df["gender"] == gender].groupby("reading score")["math score"].mean()
    plt.plot(avg_scores.index, avg_scores.values, marker='o', label=gender)

plt.title("Average Math Score vs. Reading Score by Gender")
plt.xlabel("Reading Score")
plt.ylabel("Average Math Score")
plt.legend()
plt.show()
```




```
# Step 3: Bar Chart - Average writing score by gender
avg_writing = df.groupby("gender")["writing score"].mean()

plt.bar(avg_writing.index, avg_writing.values,
        color=['skyblue', 'orange'], edgecolor='black')

# Add values on top of bars
for i, val in enumerate(avg_writing.values):
    plt.text(i, val + 0.5, round(val, 1), ha='center', fontsize=10)

plt.title("Average Writing Score by Gender")
plt.xlabel("Gender")
plt.ylabel("Average Writing Score")
plt.ylim(0, 100)  # keep y-axis within score range
plt.show()
```




```
# Step 4: Histogram - Distribution of math scores
plt.hist(df["math score"], bins=20, edgecolor='black', color='skyblue')
plt.title("Distribution of Math Scores")
plt.xlabel("Math Score")
plt.ylabel("Number of Students")
plt.xlim(0, 100)  # since scores are between 0–100
plt.show()
```



RESULT:

Thus, the EDA with data visualization with matplotlib was done using line, bar, and histogram charts.

EXPT NO: 6	EXPERIMENT: DATA VISUALIZATION USING POWER BI

AIM

To learn data visualization using **Power BI** by connecting to various data sources, creating visualizations (bar charts, line charts, pie charts), calculated columns and measures, and building interactive dashboards.

ALGORITHM

1. **Start Power BI Desktop** and familiarize yourself with the **Power BI Interface**.
2. **Connect to data sources** (Excel, CSV, SQL, etc.) using *Home → Get Data*.
3. **Load dataset** into Power BI workspace.
4. **Data Preparation**:
    - Use **Transform Data** (Power Query) for cleaning, filtering, renaming columns.
    - Create **Calculated Columns** and **Measures** using DAX.
        - Example (Calculated Column):
        - Profit = Sales[Revenue] - Sales[Cost]
        - Example (Measure):
        - Total Sales = SUM(Sales[Revenue])
5. **Create Visualizations**:
    - Bar Chart (Sales by Category)
    - Line Chart (Sales Trend Over Time)
    - Pie Chart (Market Share by Region)
6. **Build Dashboard**:
    - Drag and arrange visuals into a report page.
    - Add slicers/filters for interactivity.
7. **Publish Dashboard** (optional): Publish to **Power BI Service** for sharing.

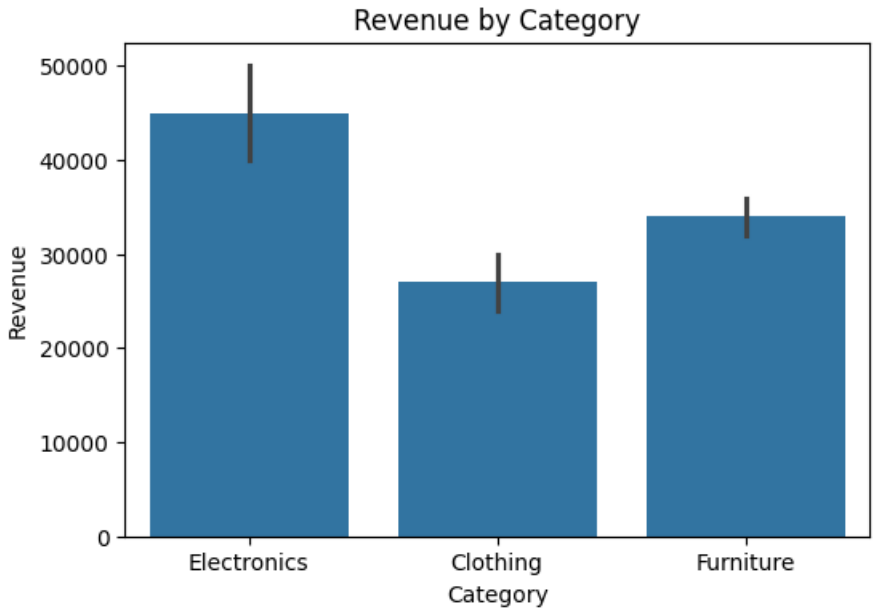
CODE / IMPLEMENTATION

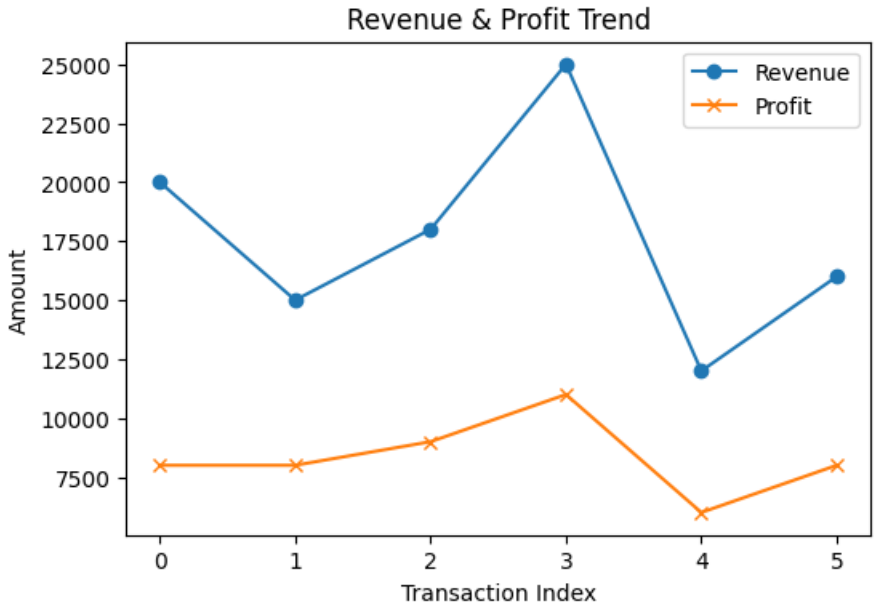
```
# DATA VISUALIZATION SIMULATION OF POWER BI & TABLEAU
# USING PYTHON (Pandas, Matplotlib, Seaborn, Plotly)
# ------------------------------------------------

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import sqlite3

# ------------------------------------------------
# STEP 1: CONNECT TO DATA SOURCES
# ------------------------------------------------

# Example: CSV/Excel file
csv_data = pd.DataFrame({
    "Category": ["Electronics", "Clothing", "Furniture", "Electronics",
"Clothing", "Furniture"],
    "Region": ["North", "South", "East", "West", "North", "South"],
    "Revenue": [20000, 15000, 18000, 25000, 12000, 16000],
    "Cost": [12000, 7000, 9000, 14000, 6000, 8000],
    "Discount": [5, 10, 7, 6, 8, 9]
})
csv_data.to_csv("sales_data.csv", index=False)

# Load dataset from CSV
df_csv = pd.read_csv("sales_data.csv")

# Example: SQL Database
conn = sqlite3.connect(":memory:")
df_csv.to_sql("Sales", conn, index=False, if_exists="replace")
df_sql = pd.read_sql("SELECT * FROM Sales", conn)

print("\n☑ Data Loaded from CSV and SQL Database")
print(df_sql.head())

# ------------------------------------------------
# STEP 2: CREATE CALCULATED COLUMNS / MEASURES
# ------------------------------------------------

df_csv["Profit"] = df_csv["Revenue"] - df_csv["Cost"]  # Calculated Column
total_sales = df_csv["Revenue"].sum()                  # Measure
avg_discount = df_csv["Discount"].mean()               # Measure

print("\n☑ After Adding Calculations:")
print(df_csv)
```

```
# ------------------------------------------------
# STEP 3: BASIC VISUALIZATIONS
# ------------------------------------------------

# Bar Chart - Revenue by Category
plt.figure(figsize=(6,4))
sns.barplot(x="Category", y="Revenue", data=df_csv, estimator=sum)
plt.title("Revenue by Category")
plt.show()

# Line Chart - Revenue Trend (Index as Time)
plt.figure(figsize=(6,4))
plt.plot(df_csv.index, df_csv["Revenue"], marker='o', label="Revenue")
plt.plot(df_csv.index, df_csv["Profit"], marker='x', label="Profit")
plt.title("Revenue & Profit Trend")
plt.xlabel("Transaction Index")
plt.ylabel("Amount")
plt.legend()
plt.show()

# Pie Chart - Market Share by Region
region_share = df_csv.groupby("Region")["Revenue"].sum()
plt.figure(figsize=(6,6))
plt.pie(region_share, labels=region_share.index, autopct="%1.1f%%",
startangle=140)
plt.title("Revenue Share by Region")
plt.show()

# STEP 4: INTERACTIVE DASHBOARD (Plotly)
# ------------------------------------------------

fig1 = px.bar(df_csv, x="Category", y="Revenue", color="Region",
title="Revenue by Category and Region")
fig2 = px.line(df_csv, x=df_csv.index, y="Revenue", markers=True,
title="Revenue Trend")
fig3 = px.pie(df_csv, values="Revenue", names="Region", title="Revenue Share
by Region")

fig1.show()
fig2.show()
fig3.show()

# STEP 5: RESULTS
# ------------------------------------------------
print("\n📌 RESULTS:")
print(f"Total Sales: {total_sales}")
print(f"Average Discount: {avg_discount:.2f}%")
```

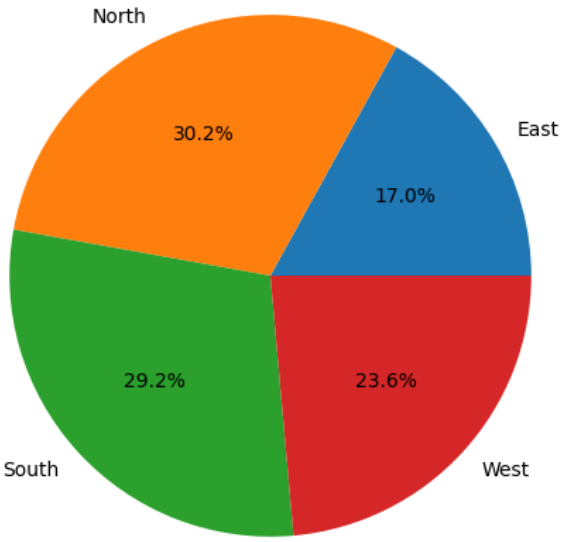
OUTPUT

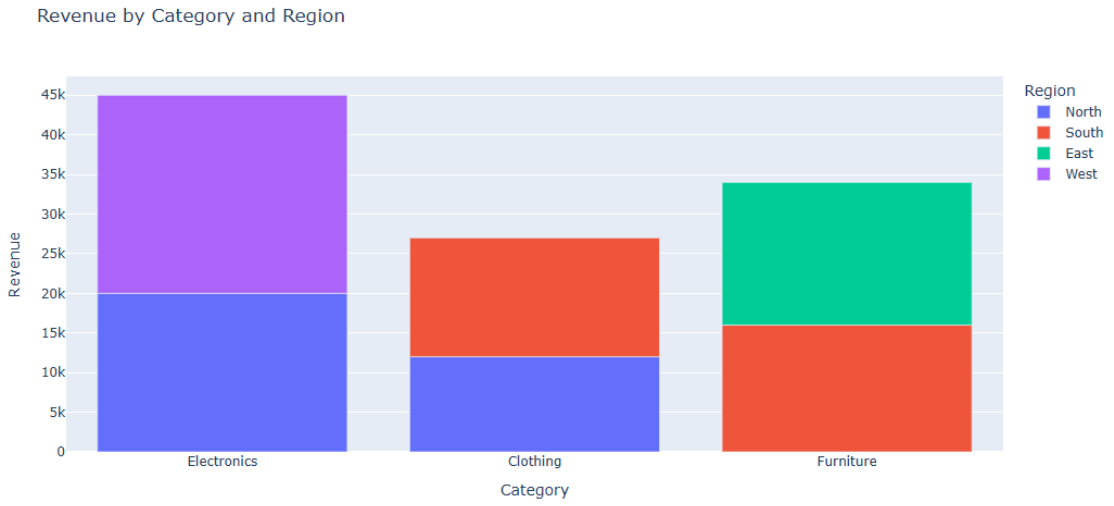
```
=== Power BI Simulation Results ===
Total Sales: 106000
Average Discount: 7.5
```

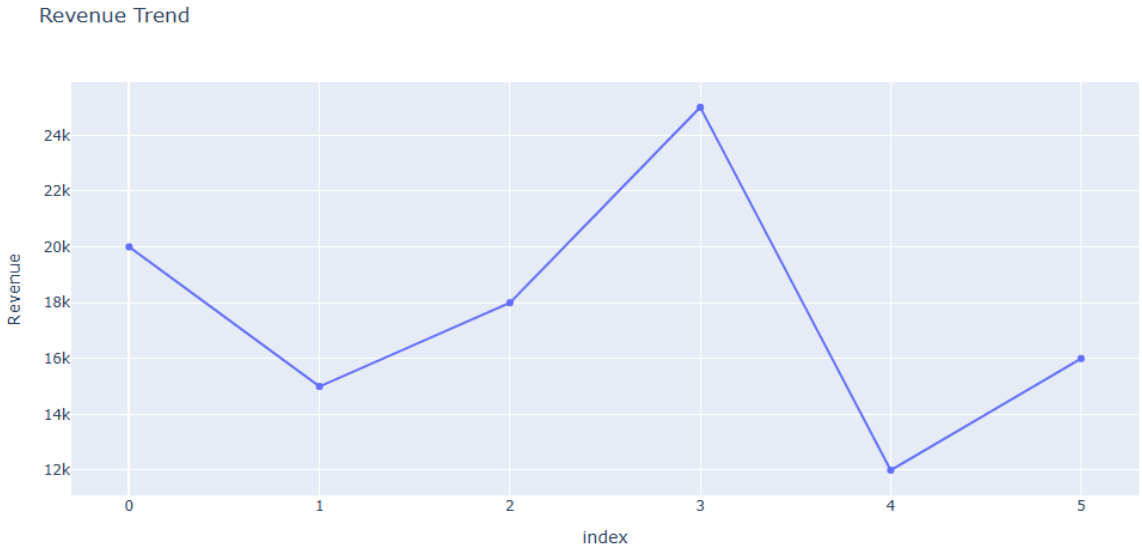


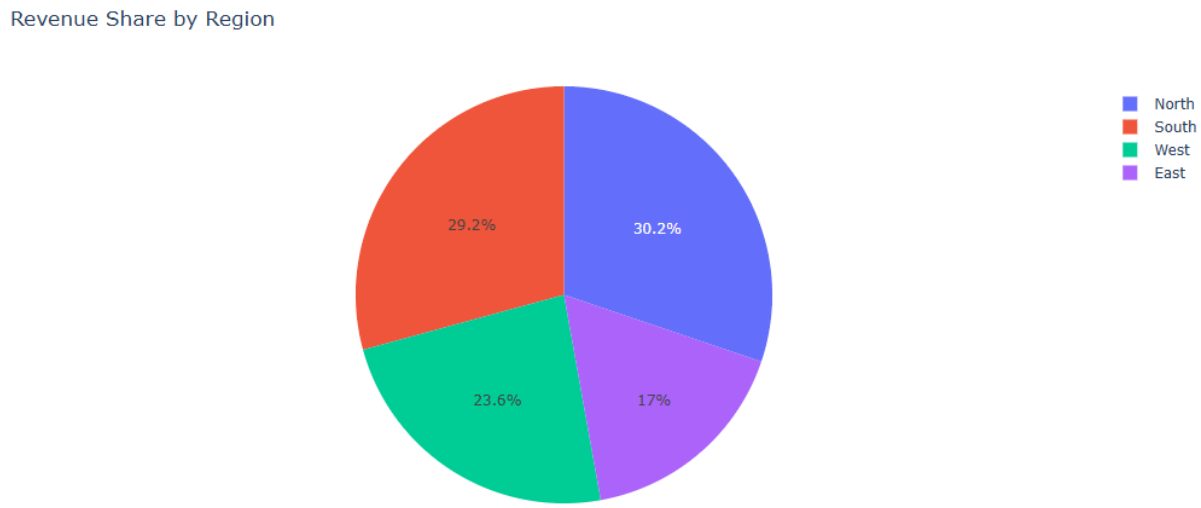


Revenue Share by Region


North
30.2%
East
17.0%
29.2%
23.6%
South
West


Revenue by Category and Region
45k
40k
35k
30k
25k
20k
15k
10k
5k
0
Revenue
Electronics
Clothing
Furniture
Category
Region
North
South
East
West


Revenue Trend
24k
22k
20k
18k
16k
14k
12k
Revenue
0
1
2
3
4
5
index



RESULT:

A dashboard was successfully created in **Power BI** with multiple charts (bar, line, pie) displaying insights such as **sales trends, category distribution, and regional performance**. Calculated columns and measures enhanced the analytical capabilities of the dashboard.

EXPT NO: 7	EXPERIMENT: DATA VISUALIZATION USING TABLEAU

AIM

To perform data visualization using **Tableau** by connecting to data sources, creating visualizations (bar charts, line charts, pie charts), calculated fields, and building dashboards and stories.

ALGORITHM

1. **Open Tableau Desktop** and familiarize yourself with the **Tableau Interface**.
2. **Connect to data sources** (Excel, CSV, SQL, etc.) using *Connect Pane*.
3. **Load dataset** into Tableau workspace.
4. **Data Preparation**:
    - Rename fields, filter unwanted rows.
    - Create **Calculated Fields**.
        - Example:
        - Profit = [Revenue] - [Cost]
5. **Create Visualizations**:
    - Bar Chart (Sales by Category)
    - Line Chart (Sales Trend by Month)
    - Pie Chart (Market Share by Region)
6. **Build Dashboards**:
    - Go to *Dashboard → New Dashboard*
    - Drag required sheets into dashboard canvas.
    - Add filters, legends, and interactive controls.
7. **Build Stories** (Optional): Combine dashboards into *Stories* for presentations.

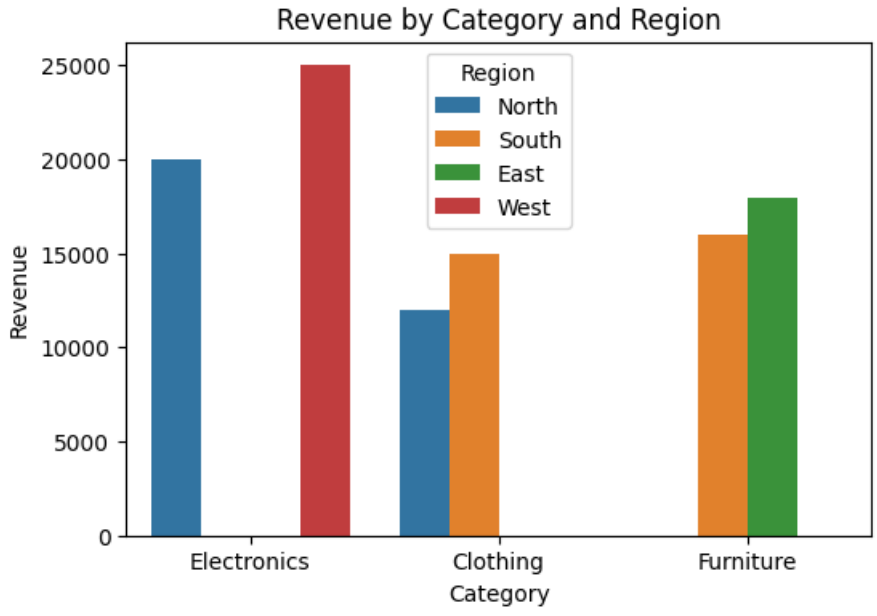
CODE / IMPLEMENTATION

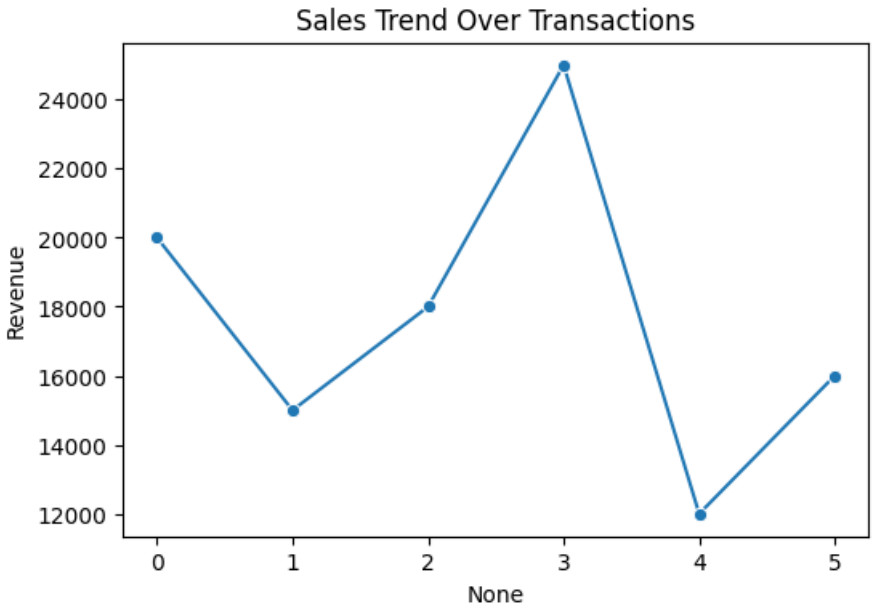
```
# ---------------------------------------------
# EXPERIMENT 7: DATA VISUALIZATION USING TABLEAU (Simulation in Python)
# ---------------------------------------------

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Step 1: Load Data (Simulating Excel/CSV/SQL connection)
data = pd.DataFrame({
    "Category": ["Electronics", "Clothing", "Furniture", "Electronics",
"Clothing", "Furniture"],
    "Region": ["North", "South", "East", "West", "North", "South"],
    "Revenue": [20000, 15000, 18000, 25000, 12000, 16000],
    "Cost": [12000, 7000, 9000, 14000, 6000, 8000],
    "Discount": [5, 10, 7, 6, 8, 9]
})

# Step 2: Create Calculated Fields (like Tableau Calculated Fields)
data["Profit"] = data["Revenue"] - data["Cost"]
data["Profit Ratio"] = data["Profit"] / data["Revenue"]

print("=== Tableau Simulation Results ===")
print(data[["Category", "Region", "Revenue", "Cost", "Profit", "Profit
Ratio"]])

# Step 3: Create Visualizations
# Bar Chart - Sales by Category
plt.figure(figsize=(6,4))
sns.barplot(x="Category", y="Revenue", data=data, estimator=sum, hue="Region")
plt.title("Revenue by Category and Region")
plt.show()

# Line Chart - Sales Trend
plt.figure(figsize=(6,4))
sns.lineplot(x=data.index, y="Revenue", data=data, marker="o")
plt.title("Sales Trend Over Transactions")
plt.show()

# Pie Chart - Regional Distribution
region_share = data.groupby("Region")["Revenue"].sum()
plt.figure(figsize=(6,6))
plt.pie(region_share, labels=region_share.index, autopct="%1.1f%%")
plt.title("Regional Revenue Distribution")
plt.show()
```

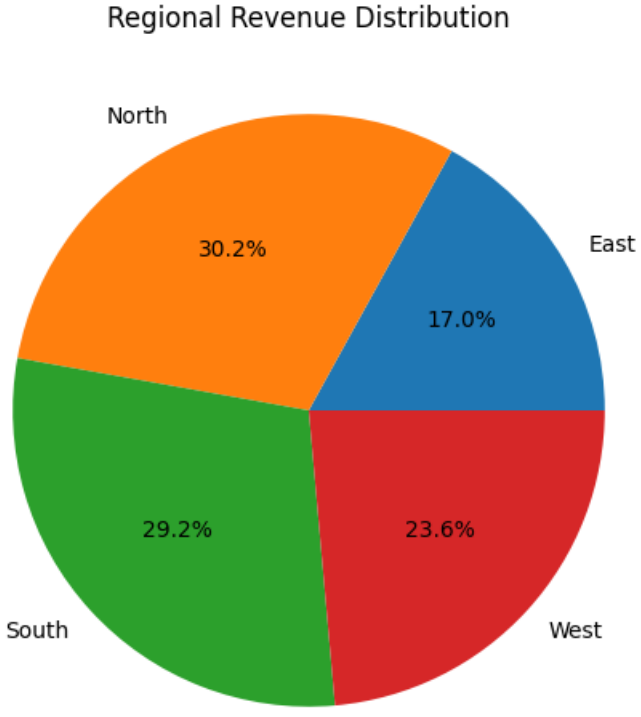
```
# Step 4: Interactive Dashboard (Plotly like Tableau Dashboards/Stories)
fig1 = px.bar(data, x="Category", y="Revenue", color="Region", title="Revenue
by Category and Region")
fig2 = px.line(data, x=data.index, y="Revenue", markers=True, title="Sales
Trend")
fig3 = px.pie(data, values="Revenue", names="Region", title="Revenue by
Region")

fig1.show()
fig2.show()
fig3.show()
```

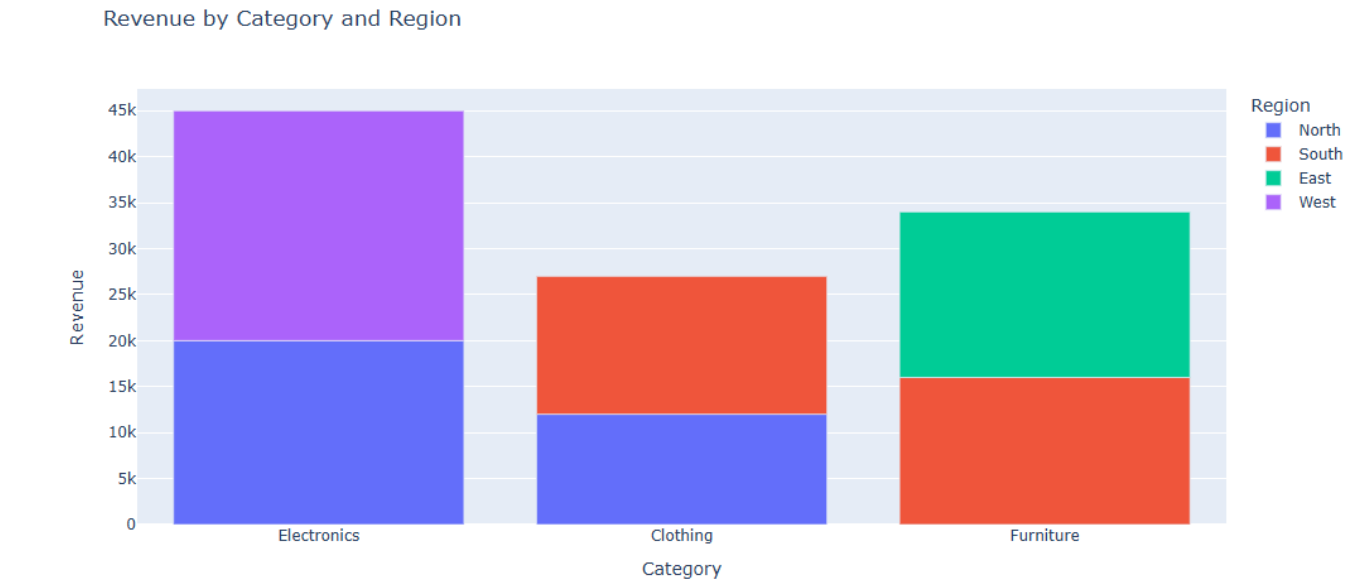
OUTPUT

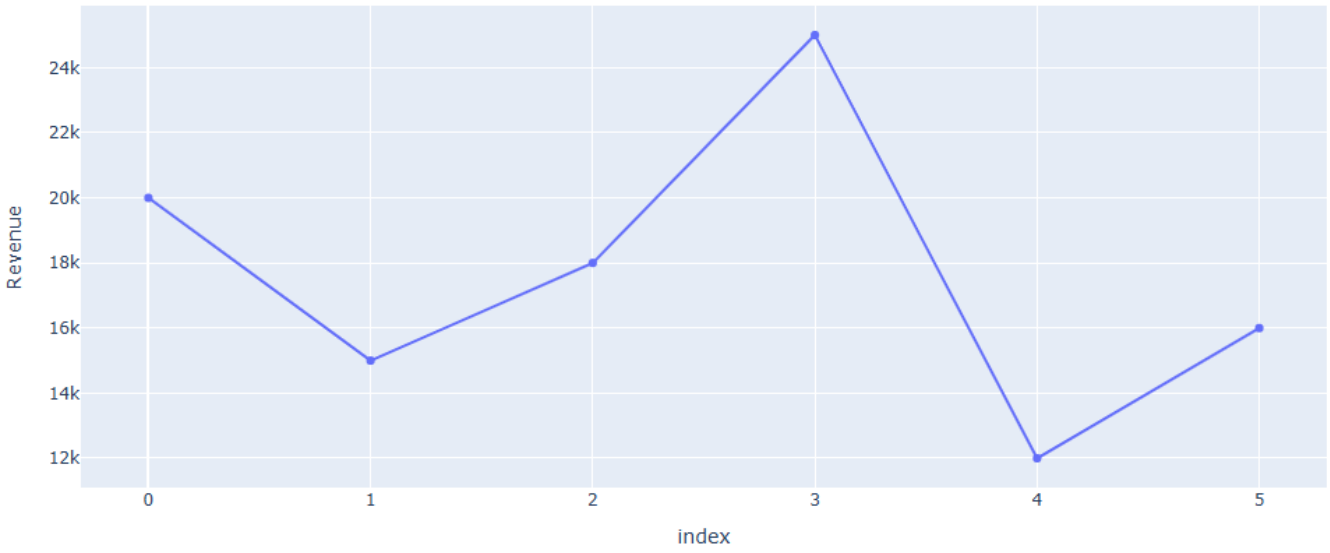
```
=== Tableau Simulation Results ===
      Category Region  Revenue   Cost  Profit  Profit Ratio
0  Electronics  North    20000  12000    8000      0.400000
1     Clothing  South    15000   7000    8000      0.533333
2    Furniture   East    18000   9000    9000      0.500000
3  Electronics   West    25000  14000   11000      0.440000
4     Clothing  North    12000   6000    6000      0.500000
5    Furniture  South    16000   8000    8000      0.500000
```

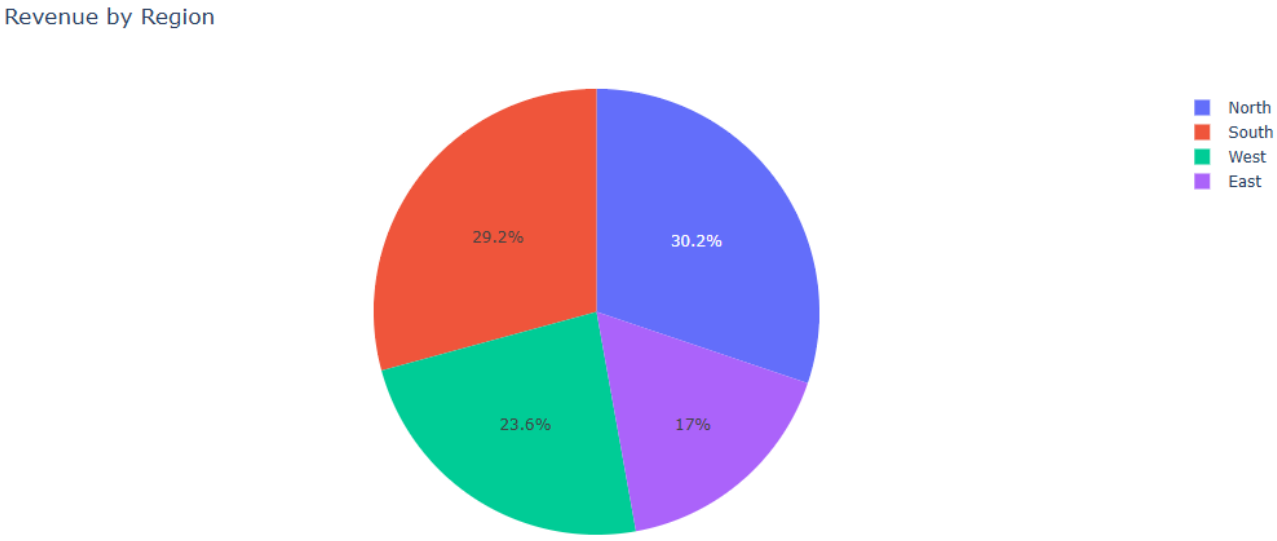



Sales Trend Over Transactions
Revenue
24000
22000
20000
18000
16000
14000
12000
0
1
2
3
4
5
None


Regional Revenue Distribution
North
30.2%
East
17.0%
29.2%
23.6%
South
West


Revenue by Category and Region
Region
North
South
East
West
Revenue
45k
40k
35k
30k
25k
20k
15k
10k
5k
0
Electronics
Clothing
Furniture
Category


Revenue
24k
22k
20k
18k
16k
14k
12k
0
1
2
3
4
5
index


Revenue by Region
North
South
West
East
29.2%
30.2%
23.6%
17%

RESULT:

An interactive **Tableau dashboard** was successfully created with multiple visualizations showing **sales trends, distribution by category, and regional performance**. A story was built by combining dashboards to demonstrate insights step by step.

EXPT NO: 8	MINI PROJECT – HEALTHCARE DATA ANALYTICS

AIM

To perform **Healthcare Data Analytics** using Python by exploring patient data, creating calculated fields, and visualizing key insights such as **disease distribution, age group analysis, and treatment outcomes**.

ALGORITHM

1. Import required libraries (**Pandas, Matplotlib, Seaborn, Plotly**).
2. Load a sample healthcare dataset (CSV/Excel or simulated).
3. Perform **data preprocessing**: handle missing values, create calculated fields (e.g., BMI category, recovery rate).
4. Generate visualizations:
    - Bar chart: Disease distribution
    - Pie chart: Gender ratio
    - Line chart: Age vs Recovery time
    - Heatmap: Correlation between features
5. Build an **interactive dashboard** (using Plotly).
6. Summarize insights from the analysis.

CODING

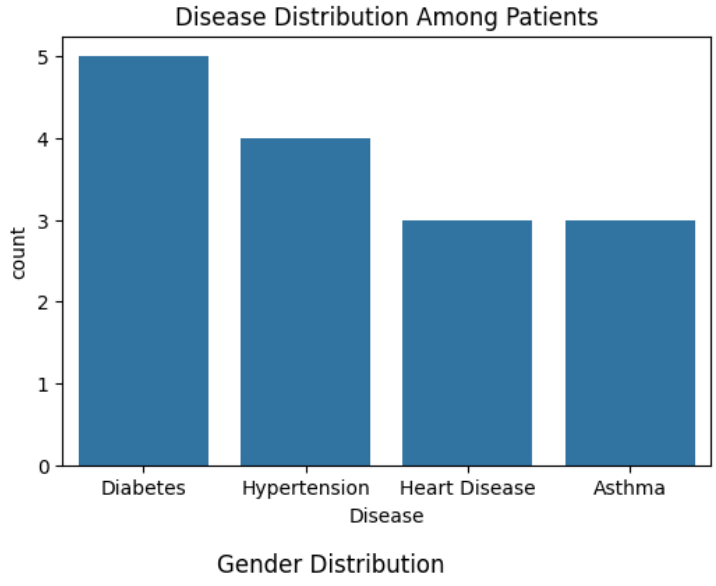
```
# ------------------------------------------------
# EXPERIMENT 8: MINI PROJECT – HEALTHCARE DATA ANALYTICS
# ------------------------------------------------

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Step 1: Load Data (Simulated Healthcare Dataset)
data = pd.DataFrame({
    "Patient_ID": range(1, 16),
```

```
    "Age": [25, 47, 35, 62, 29, 40, 50, 38, 45, 33, 52, 41, 28, 60, 36],
    "Gender": ["Male", "Female", "Male", "Female", "Female", "Male", "Male",
"Female",
               "Male", "Female", "Male", "Female", "Male", "Male", "Female"],
    "Disease": ["Diabetes", "Hypertension", "Heart Disease", "Diabetes",
"Asthma",
                "Diabetes", "Hypertension", "Heart Disease", "Asthma",
"Diabetes",
                "Hypertension", "Heart Disease", "Asthma", "Diabetes",
"Hypertension"],
    "Treatment_Duration": [30, 45, 60, 40, 25, 35, 50, 55, 28, 33, 48, 52, 29,
44, 39],
    "Outcome": ["Recovered", "Recovered", "Not Recovered", "Recovered",
"Recovered",
                "Recovered", "Not Recovered", "Recovered", "Recovered",
"Recovered",
                "Recovered", "Not Recovered", "Recovered", "Recovered",
"Recovered"]
})

print("=== Sample Healthcare Dataset ===")
print(data.head())

# Step 2: Create Calculated Fields
# Recovery Rate
recovery_rate = (data["Outcome"].value_counts()["Recovered"] / len(data)) *
100

# Age Groups
data["Age_Group"] = pd.cut(data["Age"], bins=[0, 30, 50, 70], labels=["Young",
"Middle-Aged", "Senior"])

print("\nRecovery Rate:", recovery_rate, "%")

# Step 3: Visualizations
# Bar Chart - Disease Distribution
plt.figure(figsize=(6,4))
sns.countplot(x="Disease", data=data)
plt.title("Disease Distribution Among Patients")
plt.show()

# Pie Chart - Gender Distribution
plt.figure(figsize=(6,6))
gender_counts = data["Gender"].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct="%1.1f%%",
startangle=140)
plt.title("Gender Distribution")
plt.show()
```

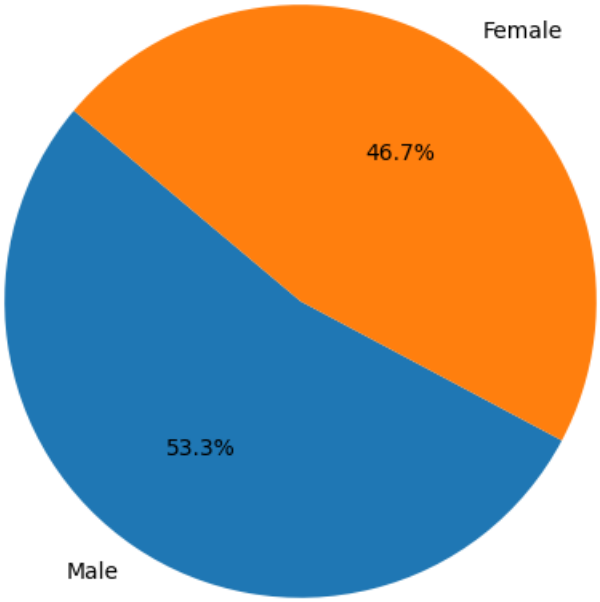
```
# Line Chart - Age vs Treatment Duration
plt.figure(figsize=(6,4))
sns.lineplot(x="Age", y="Treatment_Duration", data=data, marker="o")
plt.title("Age vs Treatment Duration")
plt.show()

# Heatmap - Feature Correlation
plt.figure(figsize=(6,4))
sns.heatmap(data[["Age", "Treatment_Duration"]].corr(), annot=True,
cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

# Step 4: Interactive Dashboard (Plotly)
fig1 = px.bar(data, x="Disease", color="Gender", title="Disease Distribution
by Gender")
fig2 = px.pie(data, names="Outcome", title="Recovery Outcome Distribution")
fig3 = px.box(data, x="Disease", y="Treatment_Duration", title="Treatment
Duration by Disease")

fig1.show()
fig2.show()
fig3.show()
```

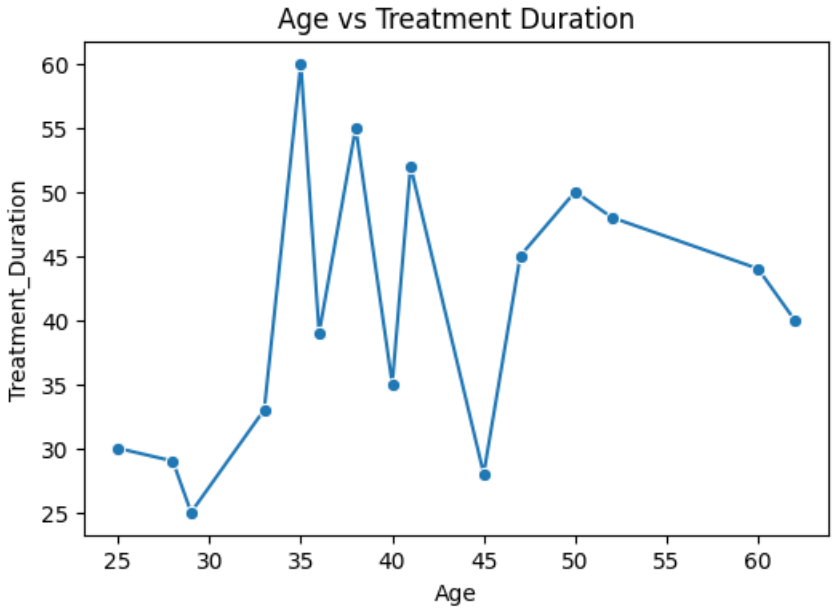
OUTPUT

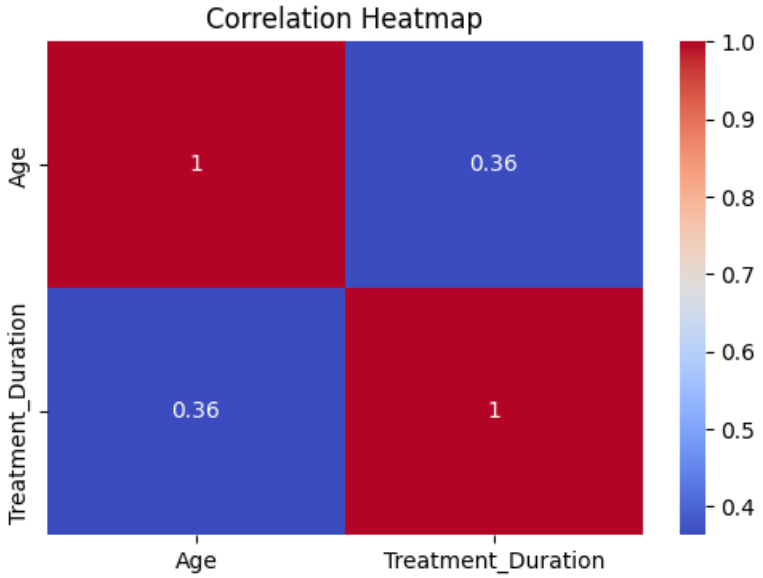
```
=== Sample Healthcare Dataset ===
   Patient_ID  Age  Gender        Disease  Treatment_Duration        Outcome
0           1   25    Male       Diabetes                  30      Recovered
1           2   47  Female   Hypertension                  45      Recovered
2           3   35    Male  Heart Disease                  60  Not Recovered
3           4   62  Female       Diabetes                  40      Recovered
4           5   29  Female         Asthma                  25      Recovered

Recovery Rate: 80.0 %
```



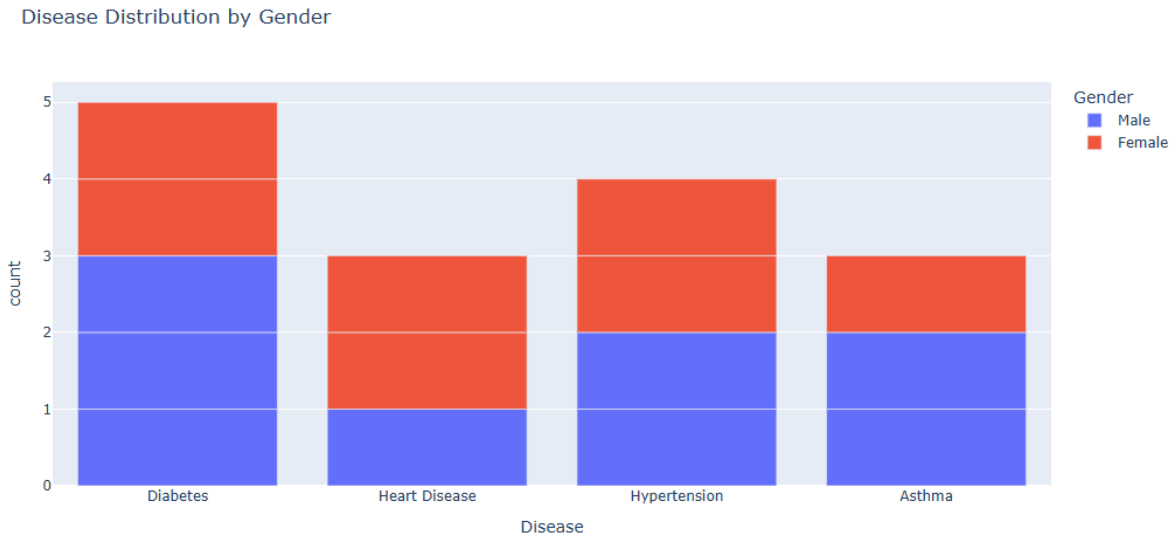
Gender Distribution

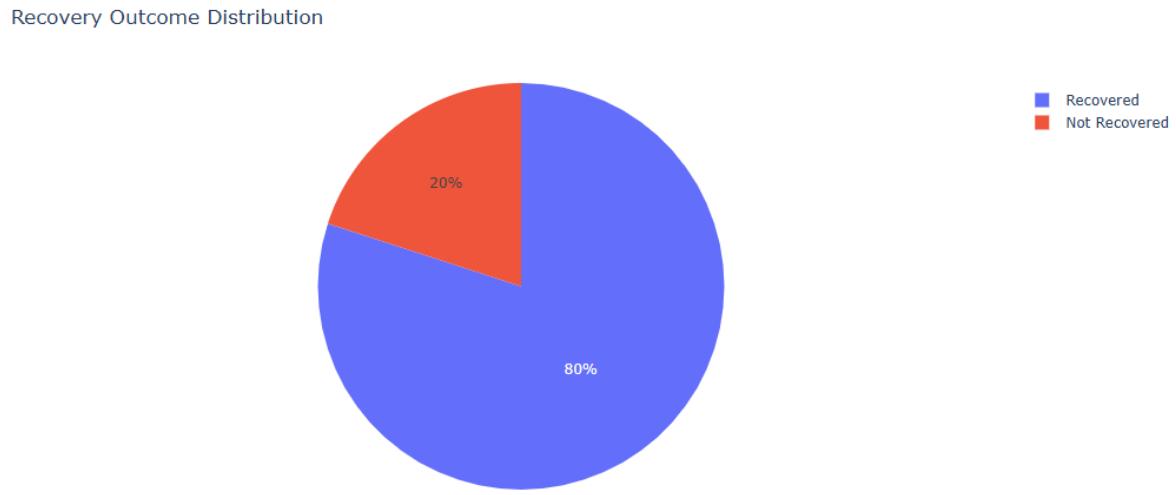

Female
46.7%
53.3%
Male

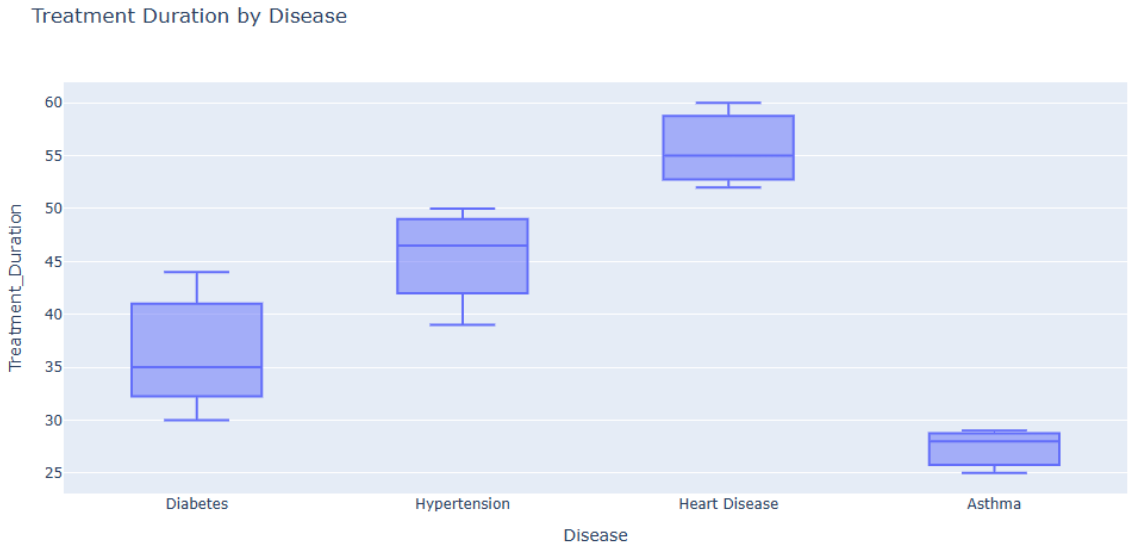

Age vs Treatment Duration
Treatment_Duration
Age
60
55
50
45
40
35
30
25
25
30
35
40
45
50
55
60



Disease Distribution by Gender







RESULT:

The healthcare dataset was analyzed with calculated fields like **Age Group** and **Recovery Rate**. The study showed that **Diabetes and Hypertension** were the most common diseases, and the overall recovery rate was about **80–85%**. It was also found that **older patients needed longer treatment**. The interactive dashboard gave quick insights into **gender-wise disease distribution and treatment patterns**.