A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:1

1] Perform descriptive statistics on given dataset.

Ans:

```
import pandas as pd
import numpy as np
from scipy import stats

# Creating a dataset with the given tuples
data = {
    'Student Name': ['Nishil', 'Rohit', 'Kohli', 'Dhoni', 'Harvey', 'Mike', 'Ronaldo', 'Messi', 'Shraddha', 'Alia'],
   'Gender': ['Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Female'],
   'Enrollment No': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],
    'Mobile Number': [9876543210, 9876543211, 9876543212, 9876543213, 9876543214, 9876543215, 9876543216, 9876543217, 9876543218, 9876543219],
    'City': ['Mumbai', 'Delhi', 'Bangalore', 'Ranchi', 'New York', 'Los Angeles', 'Lisbon', 'Barcelona', 'Mumbai', 'Mumbai'],
   'Semester 1 Marks': [85, 78, 92, 87, 76, 79, 95, 96, 82, 88],
   'Semester 2 Marks': [88, 82, 89, 85, 77, 81, 93, 97, 84, 87],
   'Semester 3 Marks': [90, 80, 94, 89, 78, 83, 94, 98, 85, 89],
   'Semester 4 Marks': [86, 79, 90, 91, 80, 84, 96, 99, 86, 90]
}

# Creating a DataFrame
df = pd.DataFrame(data)

# Function to safely calculate mode
def calculate_mode(series):
```


Name: Dhvanil Patel
En number:12102080601029

```
    mode_result = stats.mode(series, keepdims=True)  # Avoids warnings from deprecated 'keepdims' setting
    if mode_result.count[0] > 1:  # If there's a clear mode
        return mode_result.mode[0]
    else:
        return np.nan  # No mode (or all values occur equally often)


# Calculating the mean, median, mode, variance, and standard deviation for each semester's marks
detailed_stats = {
    'Statistic': ['Mean', 'Median', 'Mode', 'Variance', 'Standard Deviation', 'Min', 'Max', '25th Percentile', '75th Percentile'],
    'Semester 1': [
        df['Semester 1 Marks'].mean(),
        df['Semester 1 Marks'].median(),
        calculate_mode(df['Semester 1 Marks']),
        df['Semester 1 Marks'].var(),
        df['Semester 1 Marks'].std(),
        df['Semester 1 Marks'].min(),
        df['Semester 1 Marks'].max(),
        np.percentile(df['Semester 1 Marks'], 25),
        np.percentile(df['Semester 1 Marks'], 75)
    ],
    'Semester 2': [
        df['Semester 2 Marks'].mean(),
        df['Semester 2 Marks'].median(),
        calculate_mode(df['Semester 2 Marks']),
        df['Semester 2 Marks'].var(),
        df['Semester 2 Marks'].std(),
        df['Semester 2 Marks'].min(),
        df['Semester 2 Marks'].max(),
        np.percentile(df['Semester 2 Marks'], 25),
        np.percentile(df['Semester 2 Marks'], 75)
    ],
```


Name: Dhvanil Patel
En number:12102080601029

```
    'Semester 3': [
        df['Semester 3 Marks'].mean(),
        df['Semester 3 Marks'].median(),
        calculate_mode(df['Semester 3 Marks']),
        df['Semester 3 Marks'].var(),
        df['Semester 3 Marks'].std(),
        df['Semester 3 Marks'].min(),
        df['Semester 3 Marks'].max(),
        np.percentile(df['Semester 3 Marks'], 25),
        np.percentile(df['Semester 3 Marks'], 75)
    ],
    'Semester 4': [
        df['Semester 4 Marks'].mean(),
        df['Semester 4 Marks'].median(),
        calculate_mode(df['Semester 4 Marks']),
        df['Semester 4 Marks'].var(),
        df['Semester 4 Marks'].std(),
        df['Semester 4 Marks'].min(),
        df['Semester 4 Marks'].max(),
        np.percentile(df['Semester 4 Marks'], 25),
        np.percentile(df['Semester 4 Marks'], 75)
    ]
}

# Converting detailed stats to a DataFrame
detailed_stats_df = pd.DataFrame(detailed_stats)

# Showing detailed statistics
print(detailed_stats_df)
```


Name: Dhvanil Patel
En number:12102080601029

Output:

```
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>                                                    >
& C:/Users/pnish/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/pnish/OneDrive/Desktop/3rd yea
r/SEM 5/HTML/prac 13 DSV.py"
            Statistic  Semester 1  Semester 2  Semester 3  Semester 4
0                Mean   85.800000   86.300000   88.000000   88.100000
1              Median   86.000000   86.000000   89.000000   88.000000
2                Mode         NaN         NaN   89.000000   86.000000
3            Variance   50.177778   34.455556   41.777778   41.211111
4  Standard Deviation    7.083627    5.869885    6.463573    6.419588
5                 Min   76.000000   77.000000   78.000000   79.000000
6                 Max   96.000000   97.000000   98.000000   99.000000
7     25th Percentile   79.750000   82.500000   83.500000   84.500000
8     75th Percentile   91.000000   88.750000   93.000000   90.750000
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>
```


Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:2

2] Consider dataset with student name, gender, Enrollmentno, 4-semester result with marks of each subject, his mobile number, city. Implement the following in Python (For Practical 1,2) Perform descriptive analysis and identify the data type and implement a method to find out variation in data. For example, the difference between the highest and lowest marks in each subject semester-wise.

Ans:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. Creating a sample dataset
data = {
    'Student Name': ['Nishil', 'om', 'Nirmal', 'Shraddha', 'Alia'],
    'Gender': ['M', 'M', 'M', 'F', 'F'],
    'Enrollment No': [101, 102, 103, 104, 105],
    'Semester 1': [85, 75, 92, 88, 79],
    'Semester 2': [82, 78, 91, 89, 83],
    'Semester 3': [88, 80, 94, 86, 81],
    'Semester 4': [90, 84, 93, 85, 87],
   'Mobile Number': ['1234567890', '0987654321', '1234509876', '1234098765', '9876543210'],
    'City': ['CityA', 'CityB', 'CityA', 'CityC', 'CityB']
}

# Creating a DataFrame
df = pd.DataFrame(data)
```


Name: Dhvanil Patel
En number:12102080601029

```
# 2. Descriptive analysis
# Display the first few rows of the dataset
print("Dataset:")
print(df)

# Identifying data types
print("\nData Types:")
print(df.dtypes)

# Summary of numerical data (marks)
print("\nDescriptive Statistics for Marks:")
print(df[['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']].describe())

# 3. Calculating the variation (max - min) in marks for each subject semester-wise
variation = df[['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']].apply(lambda x: x.max() - x.min())
print("\nVariation (Max - Min) in marks for each semester:")
print(variation)

# 4. Plotting the results of students in each semester
plt.figure(figsize=(10, 6))
semesters = ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']

for semester in semesters:
    plt.plot(df['Student Name'], df[semester], marker='o', label=semester)

plt.title("Student Marks in Each Semester")
plt.xlabel("Students")
plt.ylabel("Marks")
plt.legend(title="Semesters")
plt.grid(True)
plt.show()
```


Name: Dhvanil Patel
En number:12102080601029

Output:

```
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>
s/pnish/OneDrive/Desktop/3rd year/SEM 5/HTML/prac 1 DSV.py"
Dataset:
  Student Name Gender  Enrollment No  ...  Semester 4  Mobile Number   City
0       Nishil      M            101  ...          90     1234567890  CityA
1           om      M            102  ...          84     0987654321  CityB
2       Nirmal      M            103  ...          93     1234509876  CityA
3     Shraddha      F            104  ...          85     1234098765  CityC
4         Alia      F            105  ...          87     9876543210  CityB

[5 rows x 9 columns]

Data Types:
Student Name     object
Gender           object
Enrollment No     int64
Semester 1        int64
Semester 2        int64
Semester 3        int64
Semester 4        int64
Mobile Number    object
City             object
dtype: object
```

```
Descriptive Statistics for Marks:
       Semester 1  Semester 2  Semester 3  Semester 4
count     5.00000    5.000000    5.000000    5.000000
mean     83.80000   84.600000   85.800000   87.800000
std       6.83374    5.319774    5.674504    3.701351
min      75.00000   78.000000   80.000000   84.000000
25%      79.00000   82.000000   81.000000   85.000000
50%      85.00000   83.000000   86.000000   87.000000
75%      88.00000   89.000000   88.000000   90.000000
max      92.00000   91.000000   94.000000   93.000000

Variation (Max - Min) in marks for each semester:
Semester 1    17
Semester 2    13
Semester 3    14
Semester 4     9
dtype: int64
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>
```


Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:3

3] Plot the graph showing the results of students in each semester.

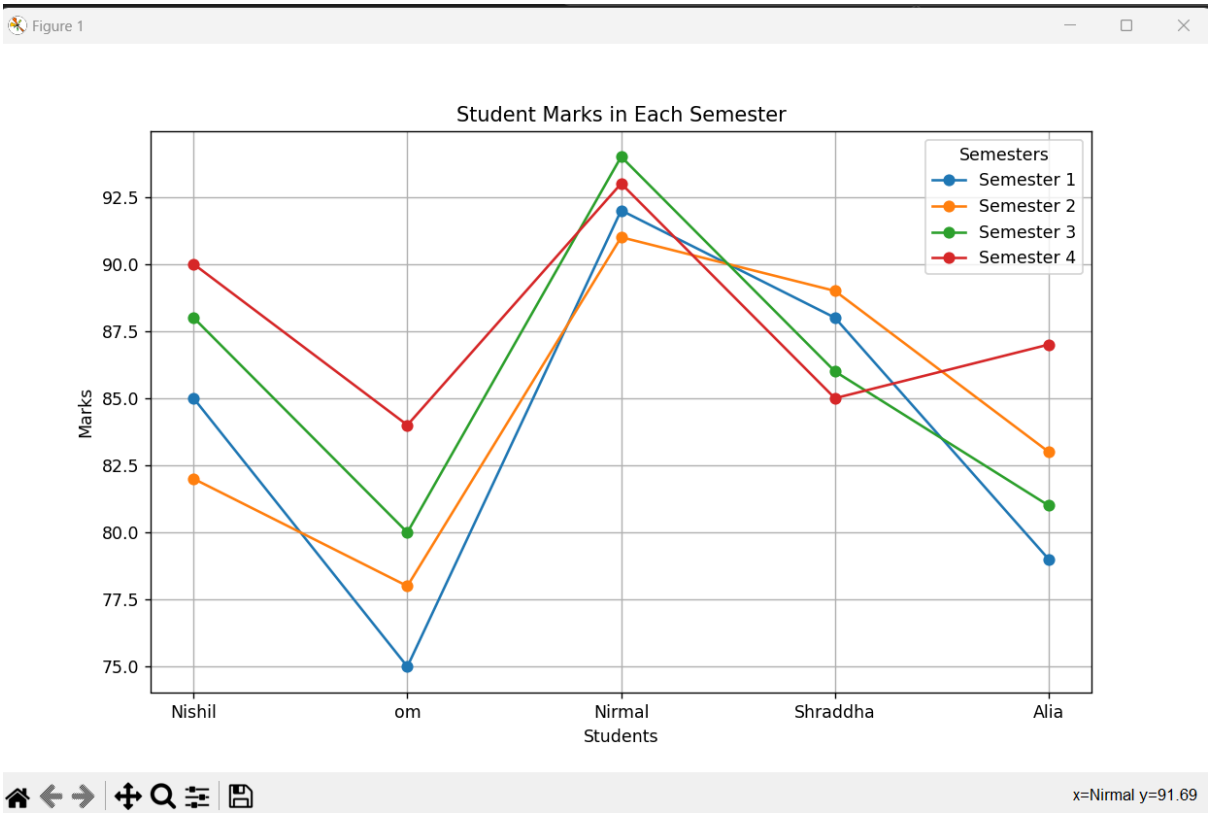
Ans:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. Creating a sample dataset
data = {
    'Student Name': ['Nishil', 'om', 'Nirmal', 'Shraddha', 'Alia'],
    'Gender': ['M', 'M', 'M', 'F', 'F'],
    'Enrollment No': [101, 102, 103, 104, 105],
    'Semester 1': [85, 75, 92, 88, 79],
    'Semester 2': [82, 78, 91, 89, 83],
    'Semester 3': [88, 80, 94, 86, 81],
    'Semester 4': [90, 84, 93, 85, 87],
   'Mobile Number': ['1234567890', '0987654321', '1234509876', '1234098765', '9876543210'],
    'City': ['CityA', 'CityB', 'CityA', 'CityC', 'CityB']
}

# Creating a DataFrame
df = pd.DataFrame(data)

plt.figure(figsize=(10, 6))
semesters = ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']

for semester in semesters:
```


Name: Dhvanil Patel
En number:12102080601029

```
plt.plot(df['Student Name'], df[semester], marker='o', label=semester)

plt.title("Student Marks in Each Semester")
plt.xlabel("Students")
plt.ylabel("Marks")
plt.legend(title="Semesters")
plt.grid(True)
plt.show()
```

output:



Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:4

4] Plot the graph showing the geographical location of students, also plot the graph showing number of male and female students and implement a method to treat missing values for gender and missing value for marks.

Ans:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. Creating a sample dataset
data = {
    'Student Name': ['Nishil', 'Om', 'Nirmal', 'Shraddha', 'Alia'],
    'Gender': ['M', 'M', 'M', 'F', 'F'],
    'Enrollment No': [101, 102, 103, 104, 105],
    'Semester 1': [85, 75, 92, 88, 79],
    'Semester 2': [82, 78, 91, 89, 83],
    'Semester 3': [88, 80, 94, 86, 81],
    'Semester 4': [90, 84, 93, 85, 87],
   'Mobile Number': ['1234567890', '0987654321', '1234509876', '1234098765', '9876543210'],
    'City': ['CityA', 'CityB', 'CityA', 'CityC', 'CityB']
}

# Introducing missing values for demonstration
data['Gender'][1] = np.nan  # Missing gender for Om
data['Semester 2'][3] = np.nan  # Missing Semester 2 mark for Shraddha

# Creating a DataFrame
```


Name: Dhvanil Patel
En number:12102080601029

```
df = pd.DataFrame(data)

# 2. Descriptive analysis
# Display the first few rows of the dataset
print("Dataset:")
print(df)

# Identifying data types
print("\nData Types:")
print(df.dtypes)

# Summary of numerical data (marks)
print("\nDescriptive Statistics for Marks:")
print(df[['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']].describe())

# 3. Calculating the variation (max - min) in marks for each subject semester-wise
variation = df[['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']].apply(lambda x: x.max() - x.min())
print("\nVariation (Max - Min) in marks for each semester:")
print(variation)

# 4. Plotting the results of students in each semester
plt.figure(figsize=(10, 6))
semesters = ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']

for semester in semesters:
    plt.plot(df['Student Name'], df[semester], marker='o', label=semester)

plt.title("Student Marks in Each Semester")
plt.xlabel("Students")
plt.ylabel("Marks")
plt.legend(title="Semesters")
plt.grid(True)
plt.show()
```


Name: Dhvanil Patel
En number:12102080601029

```
# 5. Plotting the geographical location (City) of students
plt.figure(figsize=(8, 5))
city_count = df['City'].value_counts()
city_count.plot(kind='bar', color='skyblue')
plt.title("Number of Students by City")
plt.xlabel("City")
plt.ylabel("Number of Students")
plt.grid(True)
plt.show()

# 6. Plotting the number of male and female students
# Filling missing values for gender with a placeholder
df['Gender'].fillna('Unknown', inplace=True)
gender_count = df['Gender'].value_counts()
plt.figure(figsize=(8, 5))
gender_count.plot(kind='bar', color=['blue', 'pink', 'gray'])
plt.title("Number of Male and Female Students")
plt.xlabel("Gender")
plt.ylabel("Number of Students")
plt.grid(True)
plt.show()

# 7. Handling missing values for marks (imputation)
# For demonstration, filling missing marks with the mean of the respective semester
df['Semester 2'].fillna(df['Semester 2'].mean(), inplace=True)

# Check the dataset after handling missing values
print("\nDataset after handling missing values:")
print(df)
```

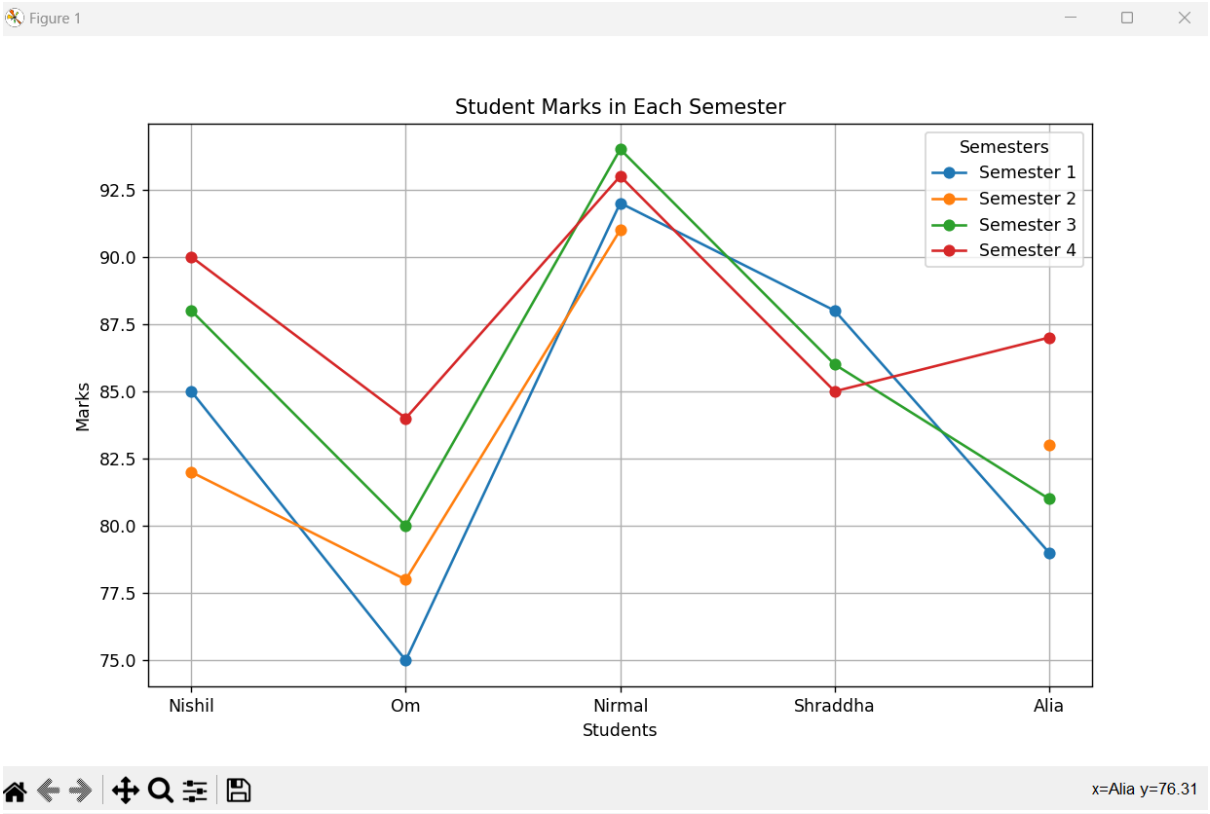

Name: Dhvanil Patel
En number:12102080601029

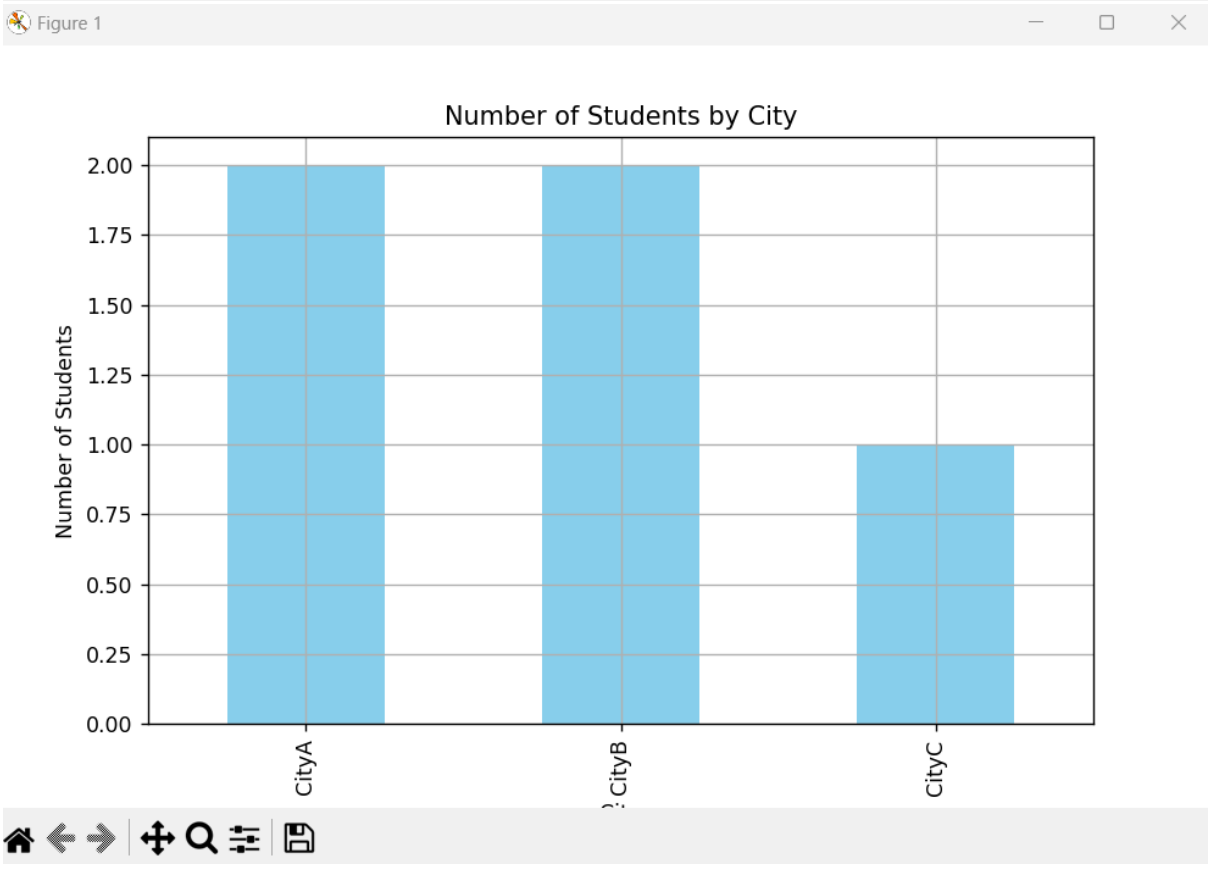
Output:

```
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>                                                    > &
python.exe "c:/Users/pnish/OneDrive/Desktop/3rd year/SEM 5/HTML/prac 3 DSV.py"
Dataset:
  Student Name Gender  Enrollment No  Semester 1  Semester 2  Semester 3  Semester 4 Mobile Number   City
0       Nishil      M           101          85        82.0          88          90    1234567890  CityA
1           Om    NaN           102          75        78.0          80          84    0987654321  CityB
2       Nirmal      M           103          92        91.0          94          93    1234509876  CityA
3     Shraddha      F           104          88         NaN          86          85    1234098765  CityC
4         Alia      F           105          79        83.0          81          87    9876543210  CityB

Data Types:
Student Name      object
Gender            object
Enrollment No      int64
Semester 1         int64
Semester 2       float64
Semester 3         int64
Semester 4         int64
Mobile Number     object
City              object
dtype: object
```

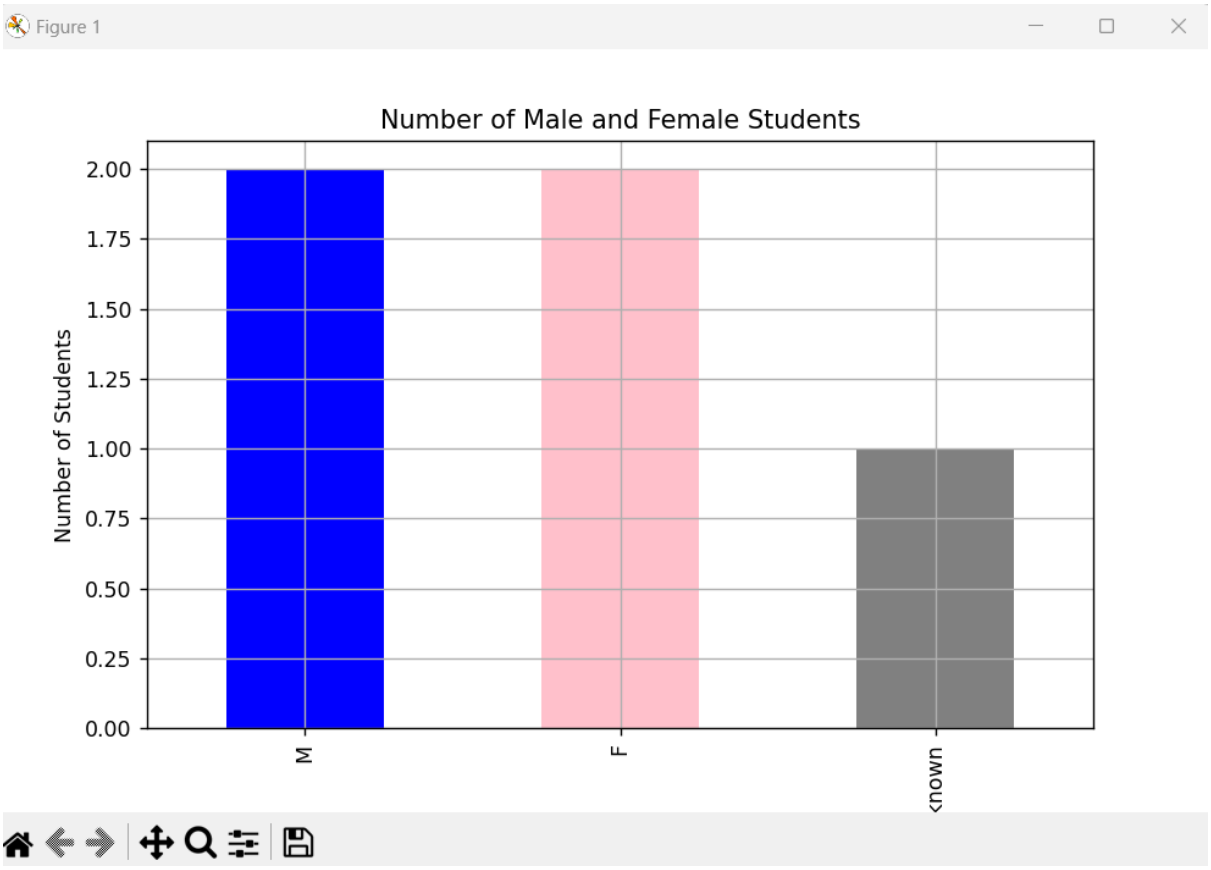
```
Descriptive Statistics for Marks:
       Semester 1  Semester 2  Semester 3  Semester 4
count     5.00000    5.000000    5.000000    5.000000
mean     83.80000   84.600000   85.800000   87.800000
std       6.83374    5.319774    5.674504    3.701351
min      75.00000   78.000000   80.000000   84.000000
25%      79.00000   82.000000   81.000000   85.000000
50%      85.00000   83.000000   86.000000   87.000000
75%      88.00000   89.000000   88.000000   90.000000
max      92.00000   91.000000   94.000000   93.000000

Variation (Max - Min) in marks for each semester:
Semester 1    17
Semester 2    13
Semester 3    14
Semester 4     9
dtype: int64
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>
```


Name: Dhvanil Patel
En number:12102080601029





Name: Dhvanil Patel
En number:12102080601029



Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:5

5] Study the various graph using visualization library.

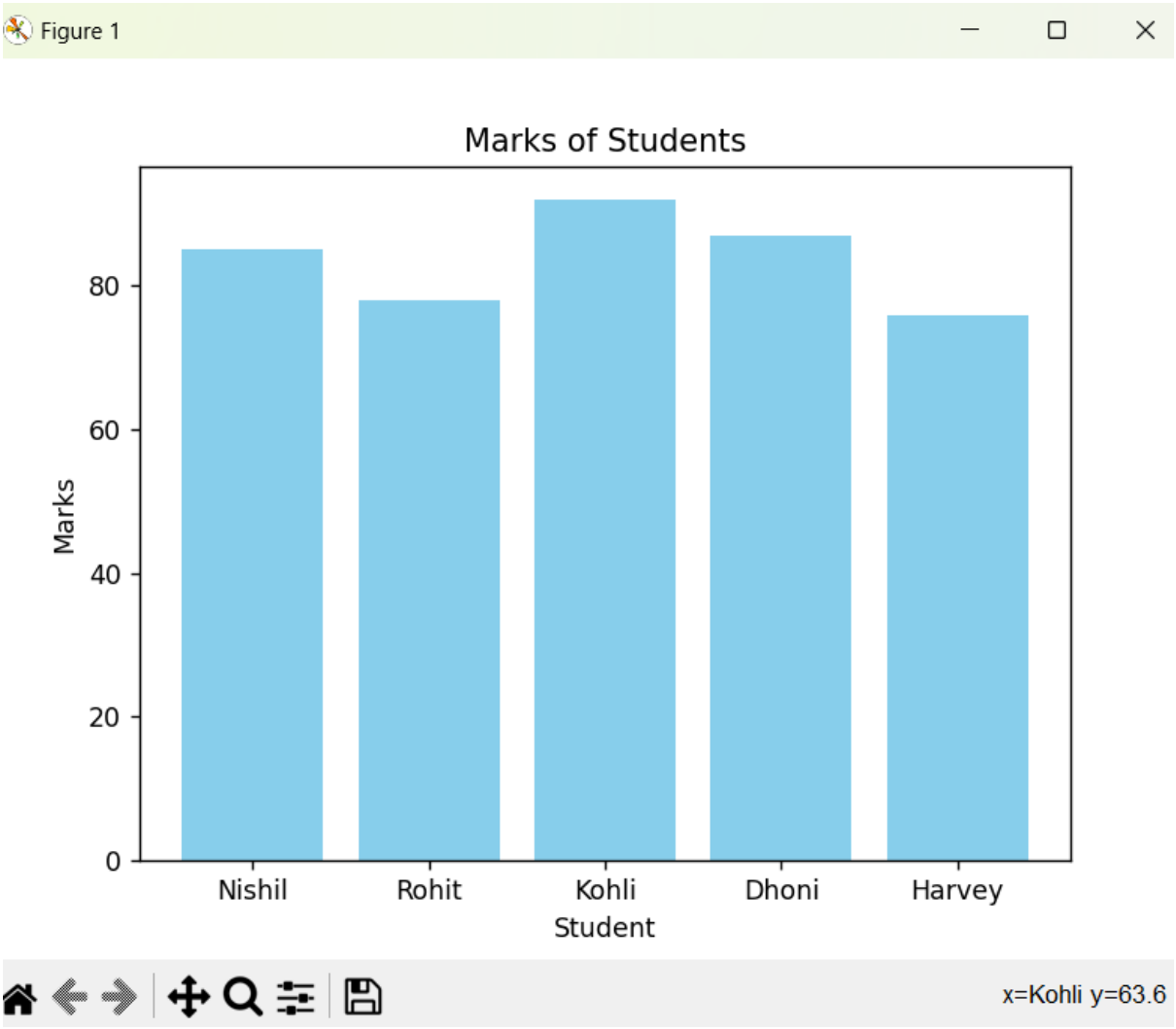
Ans:

1)Bar plot:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Sample Data
data = {'Student': ['Nishil', 'Rohit', 'Kohli', 'Dhoni', 'Harvey'],
        'Marks': [85, 78, 92, 87, 76]}

# Bar Plot
plt.bar(data['Student'], data['Marks'], color='skyblue')
plt.xlabel('Student')
plt.ylabel('Marks')
plt.title('Marks of Students')
plt.show()
```


Name: Dhvanil Patel
En number:12102080601029

Output:



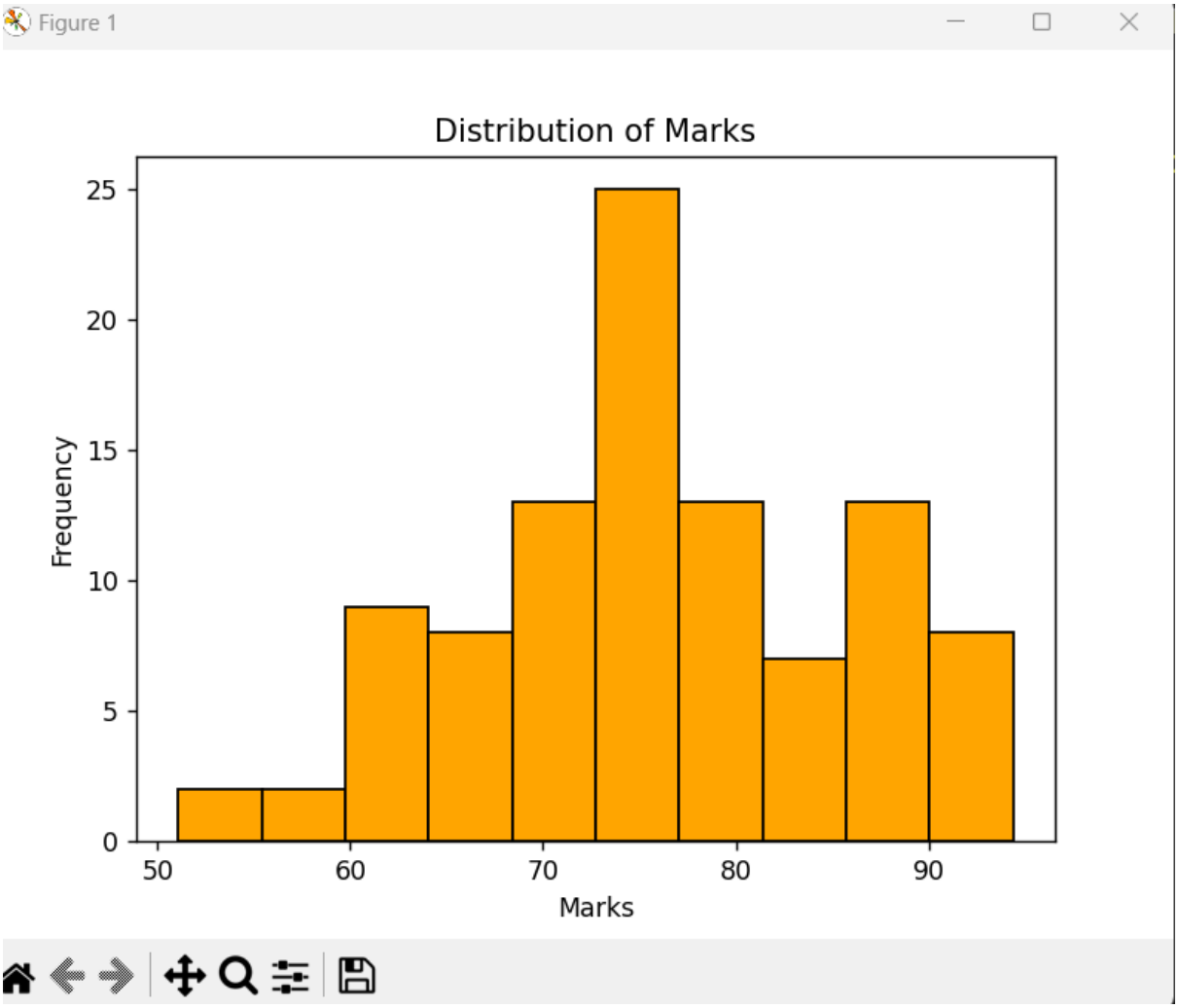
2)Histogram:

```
import numpy as np
import matplotlib.pyplot as plt

# Generate random data
marks = np.random.normal(75, 10, 100)

# Histogram
plt.hist(marks, bins=10, color='orange', edgecolor='black')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.title('Distribution of Marks')
```


Name: Dhvanil Patel
En number:12102080601029

plt.show()

Output:



3)Line plot:

import numpy as np

import matplotlib.pyplot as plt

Line Plot

semesters = ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4']

marks = [85, 88, 90, 86]

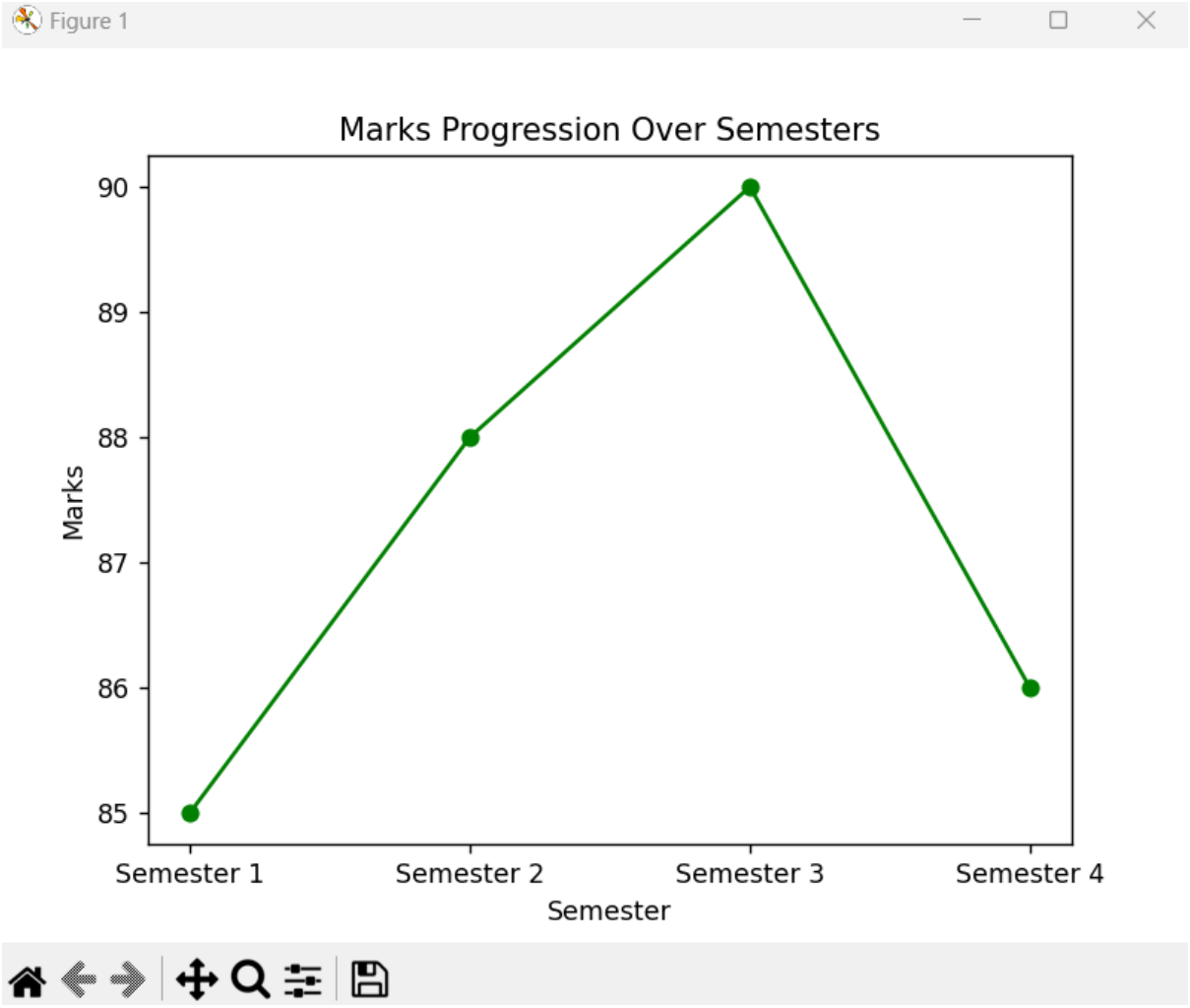
plt.plot(semesters, marks, marker='o', linestyle='-', color='green')

plt.xlabel('Semester')

Name: Dhvanil Patel
En number:12102080601029

```
plt.ylabel('Marks')
plt.title('Marks Progression Over Semesters')
plt.show()
```

Output:



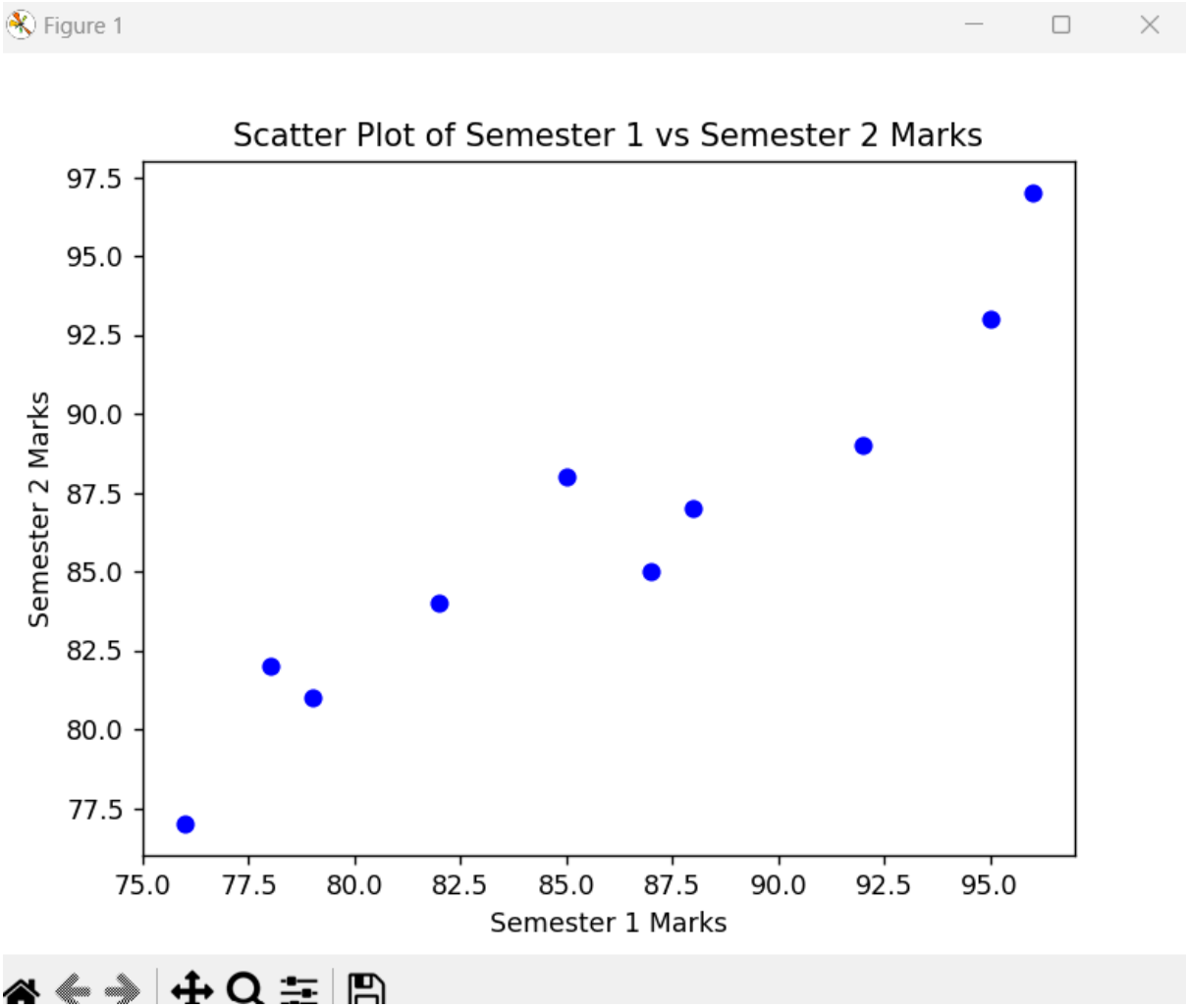
4)Scatter plot:

```
import numpy as np
import matplotlib.pyplot as plt

# Scatter Plot
semester1_marks = [85, 78, 92, 87, 76, 79, 95, 96, 82, 88]
semester2_marks = [88, 82, 89, 85, 77, 81, 93, 97, 84, 87]

plt.scatter(semester1_marks, semester2_marks, color='blue')
plt.xlabel('Semester 1 Marks')
```


Name: Dhvanil Patel
En number:12102080601029

```
plt.ylabel('Semester 2 Marks')
plt.title('Scatter Plot of Semester 1 vs Semester 2 Marks')
plt.show()
```

Output:



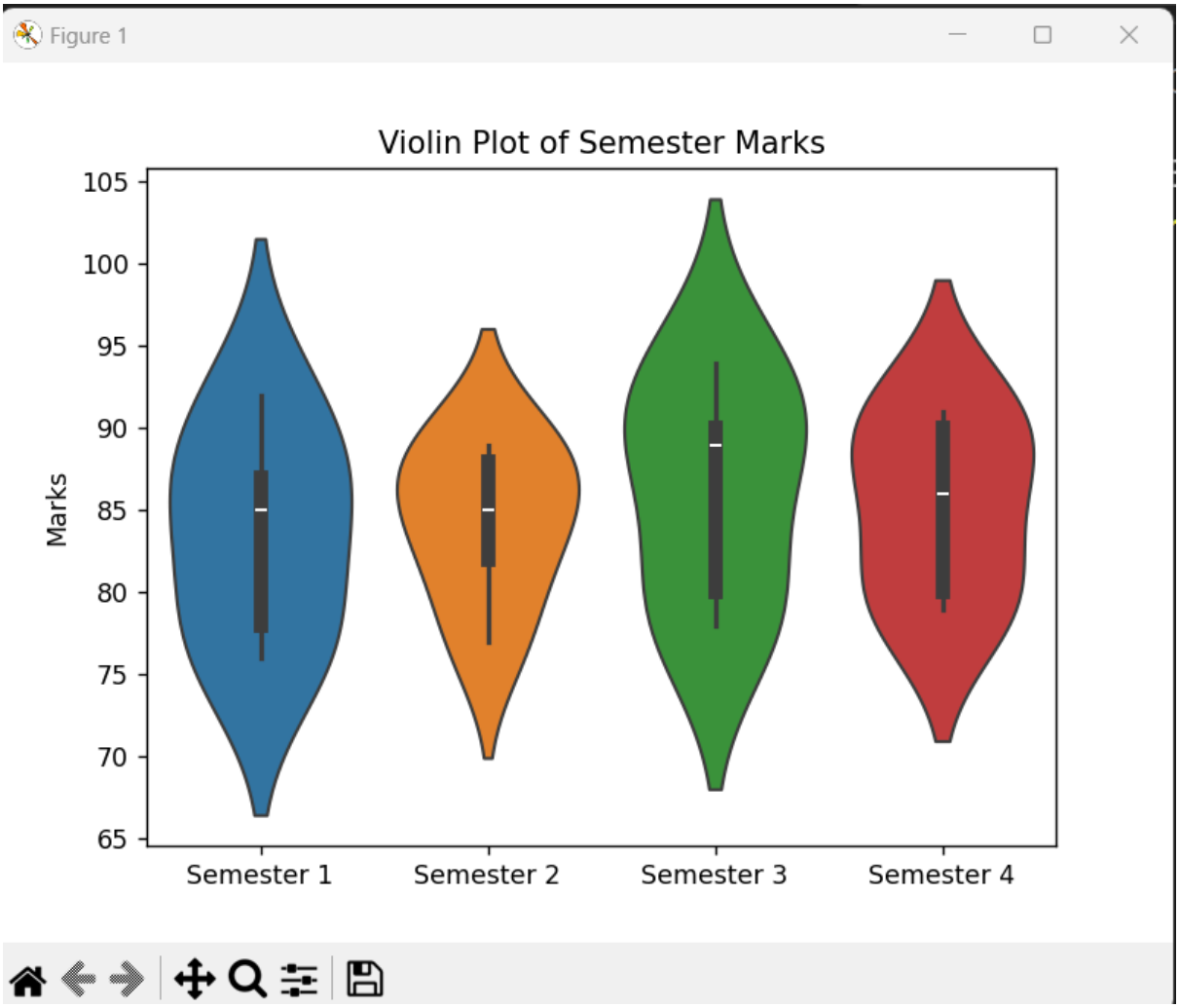
5)Violin plot:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Sample DataFrame
data = {
    'Semester 1': [85, 78, 92, 87, 76],
    'Semester 2': [88, 82, 89, 85, 77],
```


Name: Dhvanil Patel
En number:12102080601029

```
    'Semester 3': [90, 80, 94, 89, 78],
    'Semester 4': [86, 79, 90, 91, 80]
}

df = pd.DataFrame(data)


# Violin Plot
sns.violinplot(data=[df['Semester 1'], df['Semester 2'], df['Semester 3'], df['Semester 4']])
plt.xticks([0, 1, 2, 3], ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4'])
plt.ylabel('Marks')
plt.title('Violin Plot of Semester Marks')
plt.show()
```

Output:



Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:6

6] Perform encoding of categorical variable in given dataset.

Ans:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Sample dataset
data = {
    'Name': ['Nishil', 'Rohit', 'Kohli', 'Dhoni', 'Harvey', 'Mike', 'Ronaldo', 'Messi', 'Shraddha', 'Alia'],
   'Gender': ['Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Female'],
    'City': ['Mumbai', 'Delhi', 'Mumbai', 'Bangalore', 'Chennai', 'Delhi', 'Paris', 'Barcelona', 'Mumbai', 'Delhi'],
   'Marks': [85, 78, 92, 87, 76, 89, 95, 94, 91, 90]
}

df = pd.DataFrame(data)
print("Original Data:\n", df)

# Label Encoding for Gender and City
le = LabelEncoder()
df['Gender_encoded'] = le.fit_transform(df['Gender'])
df['City_encoded'] = le.fit_transform(df['City'])

print("\nLabel Encoded Data:\n", df)

# One-Hot Encoding for City
```


Name: Dhvanil Patel
En number:12102080601029

```
df_one_hot = pd.get_dummies(df, columns=['City'], drop_first=True)
print("\nOne-Hot Encoded Data:\n", df_one_hot)
```

Output:

```
PS C:\Users\pnish\OneDrive\Desktop\3rd year\SEM 5\HTML>                                                  > & C:/User
ve/Desktop/3rd year/SEM 5/HTML/prac 14 DSV.py"
Original Data:
       Name  Gender       City  Marks
0    Nishil    Male     Mumbai     85
1     Rohit    Male      Delhi     78
2     Kohli    Male     Mumbai     92
3     Dhoni    Male  Bangalore     87
4    Harvey    Male    Chennai     76
5      Mike    Male      Delhi     89
6   Ronaldo    Male      Paris     95
7     Messi    Male  Barcelona     94
8  Shraddha  Female     Mumbai     91
9      Alia  Female      Delhi     90

Label Encoded Data:
       Name  Gender       City  Marks  Gender_encoded  City_encoded
0    Nishil    Male     Mumbai     85               1             4
1     Rohit    Male      Delhi     78               1             3
2     Kohli    Male     Mumbai     92               1             4
3     Dhoni    Male  Bangalore     87               1             0
4    Harvey    Male    Chennai     76               1             2
5      Mike    Male      Delhi     89               1             3
6   Ronaldo    Male      Paris     95               1             5
7     Messi    Male  Barcelona     94               1             1
8  Shraddha  Female     Mumbai     91               0             4
9      Alia  Female      Delhi     90               0             3

One-Hot Encoded Data:
       Name  Gender  Marks  Gender_encoded  ...  City_Chennai  City_Delhi  City_Mumbai  City_Paris
0    Nishil    Male     85               1  ...         False       False         True       False
1     Rohit    Male     78               1  ...         False        True        False       False
2     Kohli    Male     92               1  ...         False       False         True       False
3     Dhoni    Male     87               1  ...         False       False        False       False
4    Harvey    Male     76               1  ...          True       False        False       False
5      Mike    Male     89               1  ...         False        True        False       False
6   Ronaldo    Male     95               1  ...         False       False        False        True
7     Messi    Male     94               1  ...         False       False        False       False
8  Shraddha  Female     91               0  ...         False       False         True       False
9      Alia  Female     90               0  ...         False        True        False       False
```


Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:7

7]Study and Introduction to data visualization setup tools.

Ans:

Data visualization setup tools are software libraries, platforms, or tools that help transform raw data into graphical formats like charts, graphs, and dashboards, making it easier to understand patterns, trends, and insights. Here's an introduction to some widely used tools for setting up and creating data visualizations.

1. Matplotlib

- Overview: A popular Python 2D plotting library used for basic static, interactive, and animated plots.
- Key Features:
  - Simple line plots, bar charts, histograms, and scatter plots.
  - Fine control over plot elements (axes, labels, legends).
  - Works well with Python's scientific libraries like NumPy and Pandas.
- Setup:

pip install matplotlib

- Example:

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [10, 20, 25, 30])

plt.show()
```

2. Seaborn

- Overview: Built on top of Matplotlib, Seaborn provides a high-level interface for creating attractive statistical graphics.
- Key Features:
  - Easier to create complex visualizations like heatmaps, pair plots, and violin plots.
  - Works seamlessly with Pandas DataFrames.
  - Built-in themes for aesthetic improvements.

Name: Dhvanil Patel
En number:12102080601029

- Setup:

pip install seaborn

- Example:

```
import seaborn as sns
sns.set(style="darkgrid")
tips = sns.load_dataset("tips")
sns.relplot(x="total_bill", y="tip", hue="day", data=tips)
```

3. Plotly

- Overview: An interactive graphing library for Python that enables the creation of highly customizable, interactive plots.
- Key Features:
  - Supports interactive plots (zoom, hover, pan).
  - Can generate 3D charts, maps, and even dashboards.
  - Integrates well with web applications like Flask and Django.
- Setup:

pip install plotly

- Example:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```

4. Bokeh

- Overview: An interactive visualization library for creating complex plots and dashboards for modern web browsers.
- Key Features:
  - Interactive plots that support web embedding.
  - Ability to build interactive dashboards with widgets.
  - Seamless integration with Jupyter notebooks.
- Setup:

pip install bokeh

- Example:

```
from bokeh.plotting import figure, output_file, show
p = figure(title="Line example", x_axis_label='x', y_axis_label='y')
p.line([1, 2, 3, 4], [4, 7, 2, 5], legend_label="Temp", line_width=2)
```

show(p)

5. Tableau

- Overview: Tableau is a leading platform for business intelligence and data visualization, allowing users to create interactive, shareable dashboards.
- Key Features:
  - Drag-and-drop interface for non-programmers.
  - Can connect to multiple data sources (Excel, SQL, cloud databases).
  - Allows the creation of interactive dashboards and storyboards.
- Setup: Tableau is a standalone software and does not require programming.
  - Download from Tableau.
- Example: You can import your dataset into Tableau and use its GUI to create charts, graphs, and dashboards without needing any code.

6. Power BI

- Overview: A business analytics service by Microsoft that allows users to visualize data and share insights across their organization.
- Key Features:
  - Drag-and-drop interface with built-in data connectors.
  - Supports both real-time and batch data visualizations.
  - Integrated with Microsoft services (Excel, SQL Server, etc.).
- Setup: Download and install Power BI Desktop from the Microsoft website.
- Example: Like Tableau, users can load their dataset and create visualizations using the drag-and-drop interface.

7. D3.js (Data-Driven Documents)

- Overview: A JavaScript library for creating interactive and dynamic data visualizations directly in web browsers.
- Key Features:
  - Very powerful for creating custom, interactive visualizations.
  - Uses web standards like SVG, HTML5, and CSS for rendering.
  - Highly customizable but requires more coding knowledge.
- Setup:
  - Include the D3.js library in your HTML file:

html

```
<script src="https://d3js.org/d3.v6.min.js"></script>
```

- Example:

html

Name: Dhvanil Patel
En number:12102080601029

```
<script>
var data = [30, 80, 45, 60, 20, 90, 35];
var width = 500, height = 200;
var svg = d3.select("body").append("svg")
   .attr("width", width)
   .attr("height", height);
svg.selectAll("rect")
   .data(data)
   .enter().append("rect")
   .attr("width", 40)
   .attr("height", function(d) { return d; })
   .attr("x", function(d, i) { return i * 50; })
   .attr("y", function(d) { return height - d; });
</script>
```

8. ggplot2 (R Programming)

- Overview: A popular data visualization package in R, based on the grammar of graphics.
- Key Features:
    - Creates complex multi-layered graphics using minimal code.
    - Highly customizable and supports advanced statistical visualizations.
- Setup: Install the package in R:

```
install.packages("ggplot2")
```

- Example:

```
library(ggplot2)
ggplot(mpg, aes(x=displ, y=hwy, color=class)) + geom_point()
```

9. Altair

- Overview: A declarative statistical visualization library for Python based on Vega and Vega-Lite.
- Key Features:
    - Declarative syntax (you specify the what and Altair handles the how).
    - Works seamlessly with Pandas.
    - Supports interactive visualizations.
- Setup:

```
pip install altair
```


Name: Dhvanil Patel
En number:12102080601029

- Example:

```
import altair as alt
import pandas as pd
df = pd.DataFrame({'x': range(10), 'y': range(10)})
chart = alt.Chart(df).mark_line().encode(x='x', y='y')
chart.show()
```

Conclusion

Each tool has its strengths depending on your project's requirements:

- Matplotlib, Seaborn, Plotly: Best for Python users.
- Tableau, Power BI: Suitable for business users who prefer GUI-based tools.
- D3.js: For web developers needing highly customized interactive visualizations.
- Altair, ggplot2: For users who prefer simple yet powerful syntax for statistical plots.

Selecting the appropriate tool depends on your programming skill, type of data, and whether you need static or interactive visuals.

Name: Dhvanil Patel
En number:12102080601029

PRACTICAL:8

8] Develop the different basic Graphical Shapes using HTML5 CANVAS .
Ans:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Shapes</title>
    <style>
        canvas {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <canvas id="myCanvas" width="400" height="400"></canvas>

    <script>
        // Get the canvas element
        const canvas = document.getElementById('myCanvas');
        const ctx = canvas.getContext('2d');

        // Function to draw shapes
        function drawShapes() {
            // 1. Draw a Rectangle
            ctx.fillStyle = 'blue';
```

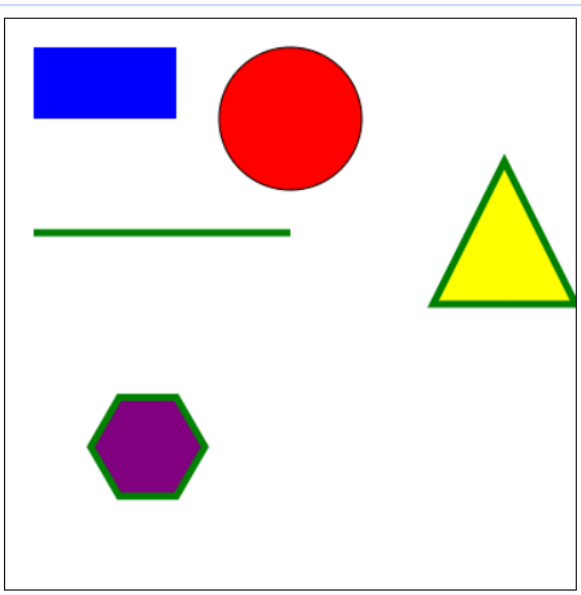

Name: Dhvanil Patel
En number:12102080601029

```
ctx.fillRect(20, 20, 100, 50);  // (x, y, width, height)

// 2. Draw a Circle (Arc)
ctx.beginPath();
ctx.arc(200, 70, 50, 0, 2 * Math.PI); // (x, y, radius, startAngle, endAngle)
ctx.fillStyle = 'red';
ctx.fill();
ctx.stroke(); // Adds the outline to the circle

// 3. Draw a Line
ctx.beginPath();
ctx.moveTo(20, 150);  // Starting point (x1, y1)
ctx.lineTo(200, 150); // Ending point (x2, y2)
ctx.strokeStyle = 'green';
ctx.lineWidth = 5;
ctx.stroke();

// 4. Draw a Triangle
ctx.beginPath();
ctx.moveTo(300, 200);   // First vertex
ctx.lineTo(350, 100);   // Second vertex
ctx.lineTo(400, 200);   // Third vertex
ctx.closePath();        // Close the path to form a triangle
ctx.fillStyle = 'yellow';
ctx.fill();
ctx.stroke();

// 5. Draw a Polygon (Hexagon)
ctx.beginPath();
const radius = 40;
const centerX = 100;
const centerY = 300;
const sides = 6;
```


Name: Dhvanil Patel
En number:12102080601029

```
        for (let i = 0; i < sides; i++) {
            const angle = (i / sides) * 2 * Math.PI;
            const x = centerX + radius * Math.cos(angle);
            const y = centerY + radius * Math.sin(angle);
            if (i === 0) {
                ctx.moveTo(x, y); // First vertex
            } else {
                ctx.lineTo(x, y); // Draw lines between vertices
            }
        }
        ctx.closePath(); // Close the shape
        ctx.fillStyle = 'purple';
        ctx.fill();
        ctx.stroke();
    }

    // Call the function to draw shapes on the canvas
    drawShapes();
  </script>
</body>
</html>
```

Output:



Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY
Department of Information Technology

PRACTICAL:9

9] Develop the different basic Graphical Shapes using SVG TAG.

Ans:

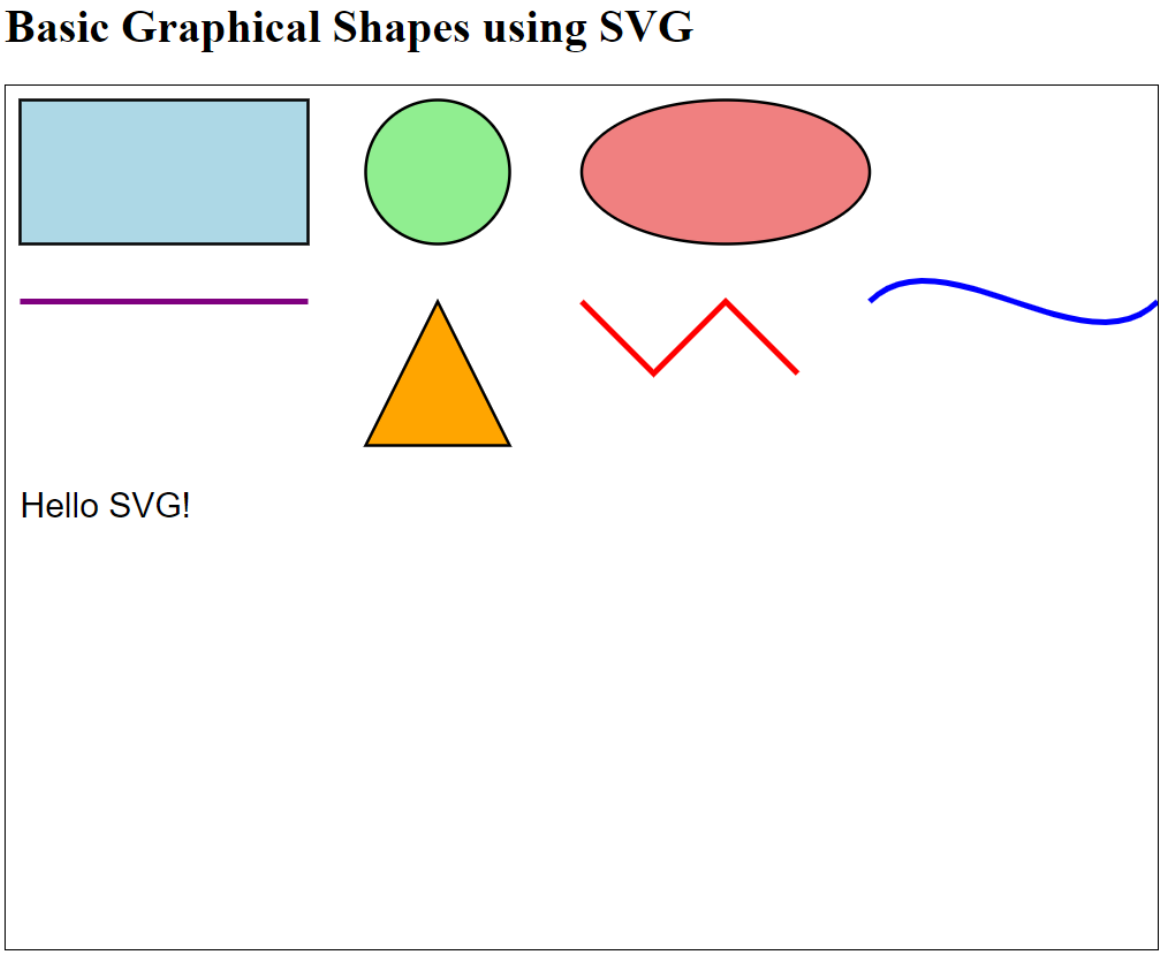
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SVG Basic Shapes</title>
    <style>
        svg {
            border: 1px solid #000; /* SVG border for visibility */
            width: 800px; /* Set width for SVG */
            height: 600px; /* Set height for SVG */
        }
    </style>
</head>
<body>
    <h1>Basic Graphical Shapes using SVG</h1>

    <svg>
        <!-- 1. Rectangle -->
        <rect x="10" y="10" width="200" height="100" fill="lightblue" stroke="black" stroke-width="2"/>

        <!-- 2. Circle -->
        <circle cx="300" cy="60" r="50" fill="lightgreen" stroke="black" stroke-width="2"/>
```


Name: Dhvanil Patel
En number:12102080601029

```
    <!-- 3. Ellipse -->
     <ellipse cx="500" cy="60" rx="100" ry="50" fill="lightcoral" stroke="black" stroke-width="2"/>

    <!-- 4. Line -->
    <line x1="10" y1="150" x2="210" y2="150" stroke="purple" stroke-width="4"/>

    <!-- 5. Polygon (Triangle) -->
      <polygon points="300,150 250,250 350,250" fill="orange" stroke="black" stroke-width="2"/>

    <!-- 6. Polyline -->
     <polyline points="400,150 450,200 500,150 550,200" fill="none" stroke="red" stroke-width="4"/>

    <!-- 7. Path -->
     <path d="M 600 150 C 650 100, 750 200, 800 150" fill="none" stroke="blue" stroke-width="4"/>

    <!-- 8. Text -->
        <text x="10" y="300" font-family="Arial" font-size="24" fill="black">Hello SVG!</text>
  </svg>
</body>
</html>
```


Name: Dhvanil Patel
En number:12102080601029

Output:



Name: Dhvanil Patel
En number:12102080601029

PRACTICAL:10

10] Develop the simple bar chart using HTML5 CANVAS.

Ans:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Bar Chart with Canvas</title>
    <style>
        canvas {
            border: 1px solid #000; /* Add a border to the canvas */
        }
    </style>
</head>
<body>
    <h1>Simple Bar Chart using HTML5 Canvas</h1>
    <canvas id="myCanvas" width="800" height="400"></canvas>

    <script>
        // Get the canvas element and its context
        const canvas = document.getElementById('myCanvas');
        const ctx = canvas.getContext('2d');

        // Sample data for the bar chart
        const names = ['Nishil', 'Virat', 'Dhoni', 'Rohit', 'Ronaldo', 'Harvey', 'Mike'];
        const scores = [85, 92, 78, 88, 90, 95, 80]; // Example scores for each name
```

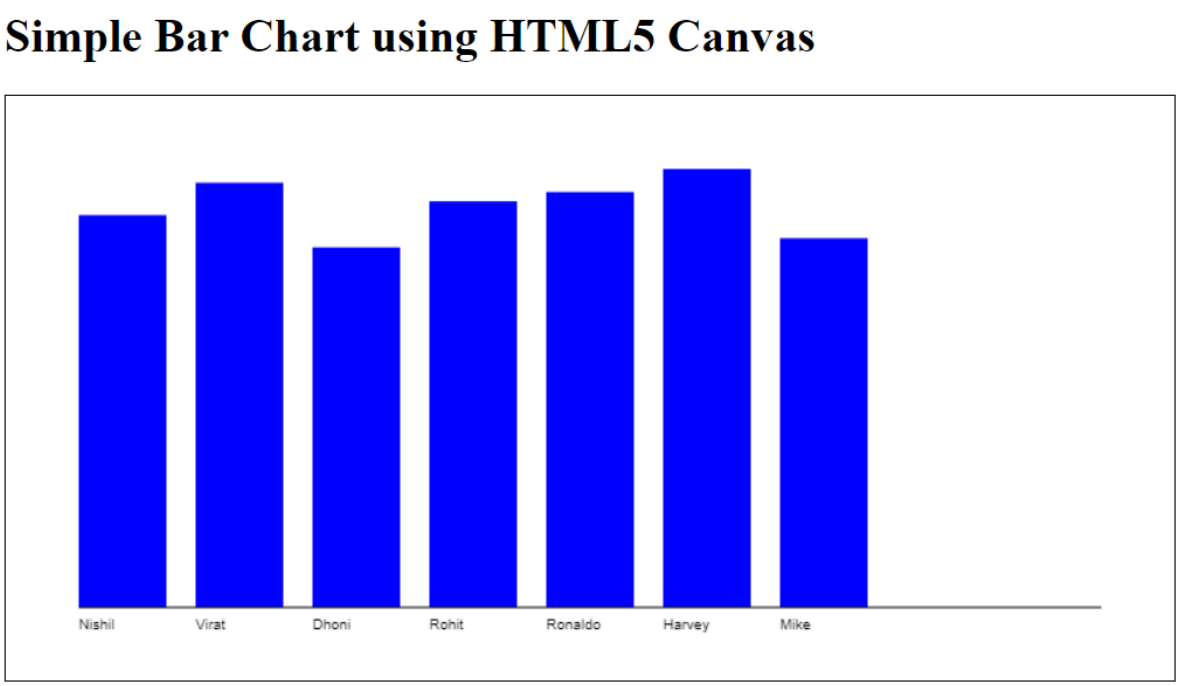

Name: Dhvanil Patel
En number:12102080601029

```
// Chart settings
const barWidth = 60; // Width of each bar
const barSpacing = 20; // Space between bars
const maxBarHeight = 300; // Max height for bars
const xOffset = 50; // X offset for bars
const yOffset = 350; // Y offset for the baseline

// Function to draw the bar chart
function drawBarChart() {
    // Clear the canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Draw the bars
    for (let i = 0; i < scores.length; i++) {
        const barHeight = (scores[i] / Math.max(...scores)) * maxBarHeight; // Calculate height
        const x = xOffset + (i * (barWidth + barSpacing)); // Calculate x position
        const y = yOffset - barHeight; // Calculate y position

        // Draw the bar
        ctx.fillStyle = 'blue'; // Bar color
        ctx.fillRect(x, y, barWidth, barHeight);

        // Draw the label
        ctx.fillStyle = 'black'; // Text color
        ctx.fillText(names[i], x, yOffset + 15); // Draw name below the bar
    }

    // Draw the baseline
    ctx.beginPath();
    ctx.moveTo(xOffset, yOffset);
    ctx.lineTo(canvas.width - xOffset, yOffset);
    ctx.stroke();
```


Name: Dhvanil Patel
En number:12102080601029

```
        }

        // Call the function to draw the chart
        drawBarChart();
    </script>
</body>
</html>
```

Output:



Name: Dhvanil Patel
En number:12102080601029

A.D PATEL INSTITUTE OF TECHNOLOGY

Department of Information Technology

PRACTICAL:11

11] Case study:i.e. market basket analysis or other.

Ans:

Case Study: Market Basket Analysis Using Association Rule Learning

Market Basket Analysis (MBA) is a data mining technique used by retailers to understand the purchase behavior of customers. It uses association rule learning to find patterns, such as the relationships between items bought together in a single transaction. One of the key algorithms for this analysis is the Apriori algorithm, which helps find frequent itemsets and association rules.

Problem Statement:

A grocery store wants to analyze customer purchasing habits to determine which products are frequently bought together. They want to use the insights to optimize store layout, create product bundles, and design promotions.

Objective:

- Perform Market Basket Analysis using a dataset of customer transactions.
- Discover frequent itemsets and association rules.
- Make recommendations based on the insights to increase sales.

Dataset:

We'll use a dataset containing transactions. Each transaction consists of a list of items purchased by a customer.

1. Data Preprocessing

We need to convert the transactions data into a format that the Apriori algorithm can process. We'll use transaction encoding to create a boolean matrix.

2. Apriori Algorithm

We use the Apriori algorithm to find frequent itemsets. The algorithm takes a support threshold, which defines the minimum proportion of transactions that should include an itemset for it to be considered frequent.

3. Association Rule Mining

Once we have the frequent itemsets, we use the association_rules function to derive rules based on metrics like confidence and lift.

Name: Dhvanil Patel
En number:12102080601029

4. Visualizing the Results

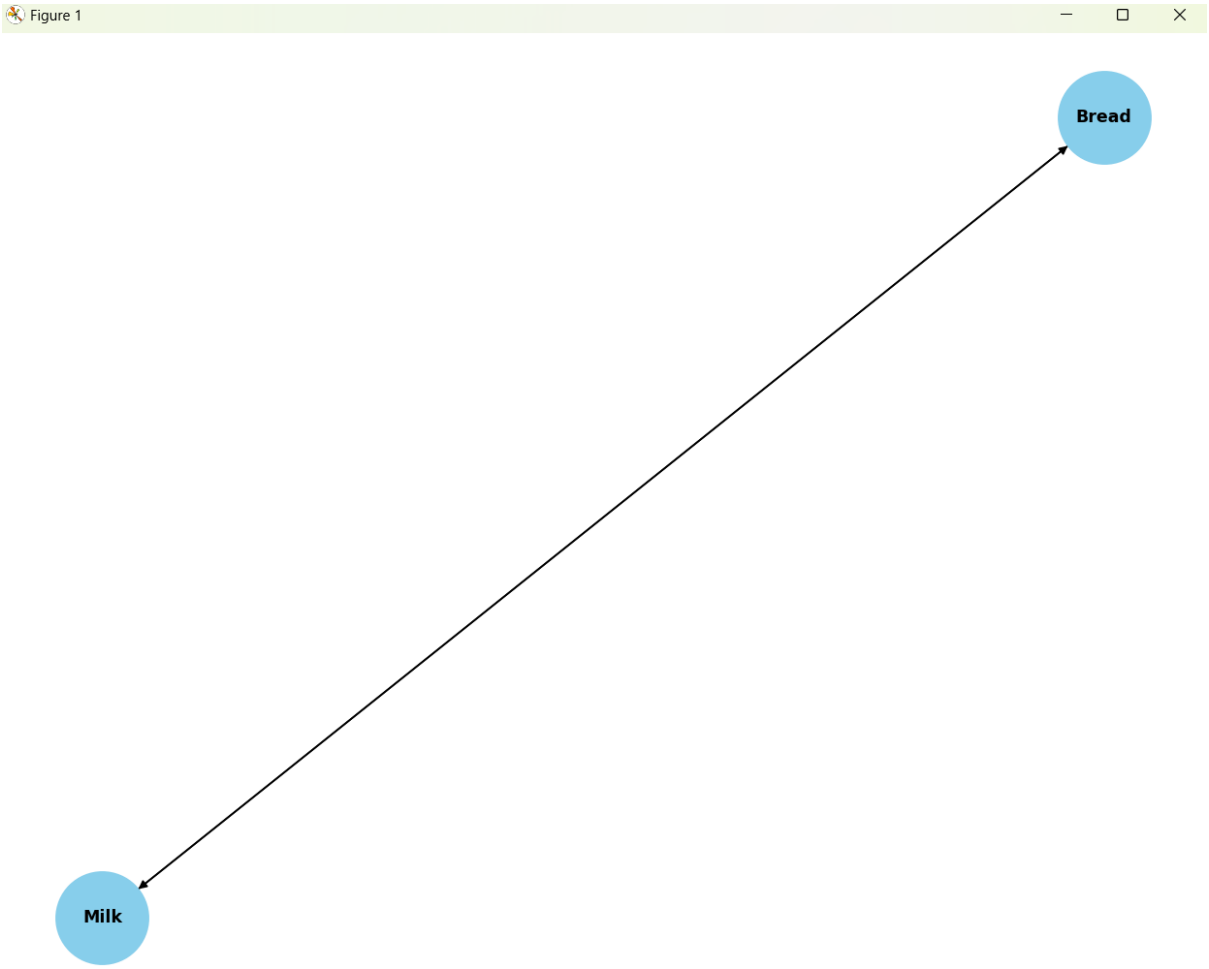
Visualizing the association rules can help better understand the relationships between items.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt
import networkx as nx

# Sample dataset (transactions)
data = {
    'Transaction_ID': [1, 2, 3, 4, 5],
    'Items': [
        ['Milk', 'Bread', 'Butter'],
        ['Beer', 'Bread'],
        ['Milk', 'Diapers', 'Beer', 'Bread'],
        ['Milk', 'Bread'],
        ['Diapers', 'Milk', 'Beer']
    ]
}

df = pd.DataFrame(data)
print("Transactions Dataset:\n", df)

# Convert the list of transactions into a format suitable for apriori
te = TransactionEncoder()
te_ary = te.fit(df['Items']).transform(df['Items'])
df_trans = pd.DataFrame(te_ary, columns=te.columns_)

print("\nTransaction Matrix:\n", df_trans)

# Apply apriori algorithm to find frequent itemsets with a minimum support of 0.6
frequent_itemsets = apriori(df_trans, min_support=0.6, use_colnames=True)
print("\nFrequent Itemsets:\n", frequent_itemsets)
```


Name: Dhvanil Patel
En number:12102080601029

```
# Find association rules with minimum confidence of 0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
print("\nAssociation Rules:\n", rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Visualize the association rules as a network graph
plt.figure(figsize=(10, 8))
G = nx.DiGraph()

for i, rule in rules.iterrows():
    for antecedent in rule['antecedents']:
        for consequent in rule['consequents']:
            G.add_edge(antecedent, consequent, weight=rule['lift'])

pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='skyblue', font_size=10, font_weight='bold')
plt.title('Association Rule Network')
plt.show()
```


Name: Dhvanil Patel
En number:12102080601029



Name: Dhvanil Patel
En number:12102080601029

```
Transactions Dataset:
    Transaction_ID                      Items
0               1      [Milk, Bread, Butter]
1               2             [Beer, Bread]
2               3  [Milk, Diapers, Beer, Bread]
3               4             [Milk, Bread]
4               5      [Diapers, Milk, Beer]

Transaction Matrix:
     Beer  Bread  Butter  Diapers   Milk
0  False   True    True    False   True
1   True   True   False    False  False
2   True   True   False     True   True
3  False   True   False    False   True
4   True  False   False     True   True

Frequent Itemsets:
    support       itemsets
0       0.6         (Beer)
1       0.8        (Bread)
2       0.8         (Milk)
3       0.6  (Milk, Bread)

Association Rules:
   antecedents consequents  support  confidence    lift
0      (Milk)     (Bread)      0.6        0.75  0.9375
1     (Bread)      (Milk)      0.6        0.75  0.9375
```

Interpretation of Results:

- **Frequent Itemsets**: These are the sets of items that frequently appear together in transactions, based on the minimum support value. For example, if Milk and Bread appear together in 60% of transactions, they form a frequent itemset.
- **Association Rules**: These are the if-then rules derived from the frequent itemsets. For instance, a rule like If someone buys Milk, they are likely to buy Bread with high **confidence** indicates a strong relationship between the two items.

Recommendations for the Grocery Store:

1. **Product Bundling**: Bundle frequently bought items (e.g., Milk and Bread) to encourage customers to purchase more items.
2. **Store Layout**: Place related items like Milk and Bread near each other to increase the likelihood of joint purchases.
3. **Promotions**: Offer discounts on associated products when one is bought, e.g., a discount on Bread when customers buy Milk.

Name: Dhvanil Patel
En number:12102080601029

Conclusion:

Market Basket Analysis can provide valuable insights into customer purchasing patterns, allowing businesses to improve product placement, create bundles, and design effective promotions, ultimately increasing sales.

Name: Dhvanil Patel
En number:12102080601029