A logo for a company

Description automatically generatedA logo for a university

Description automatically generated**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):

    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)

    ],

    '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)

    ]

}

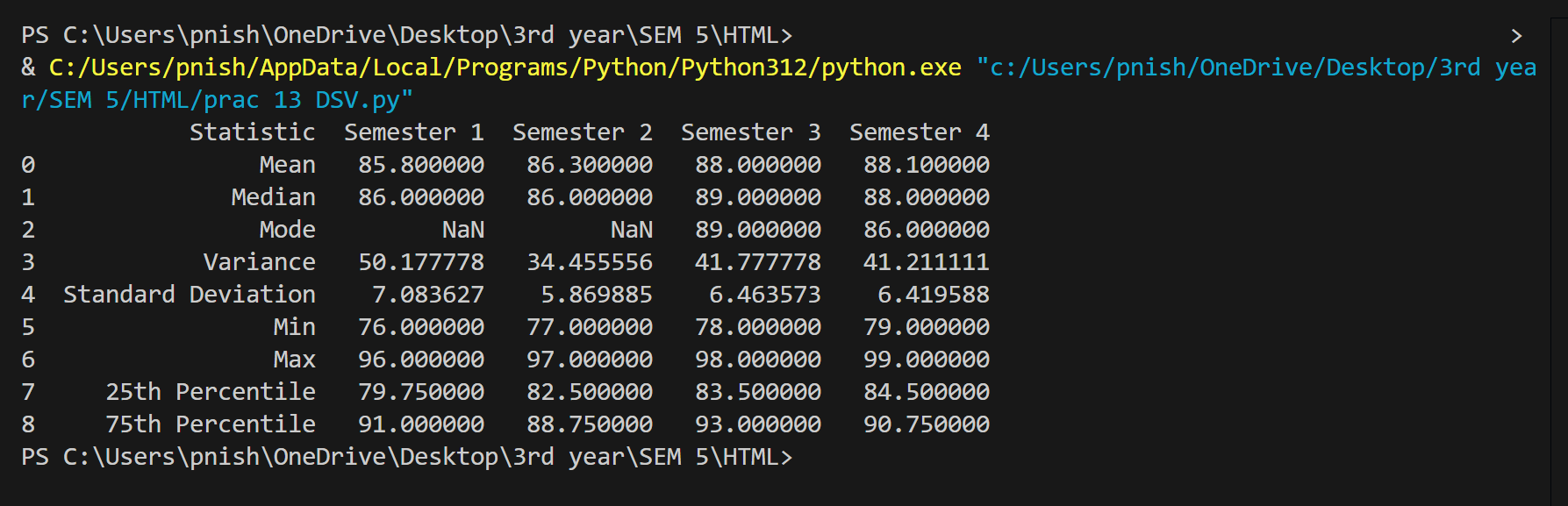
# Converting detailed stats to a DataFrame

detailed\_stats\_df = pd.DataFrame(detailed\_stats)

# Showing detailed statistics

print(detailed\_stats\_df)

**Output:**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated**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)

# 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")

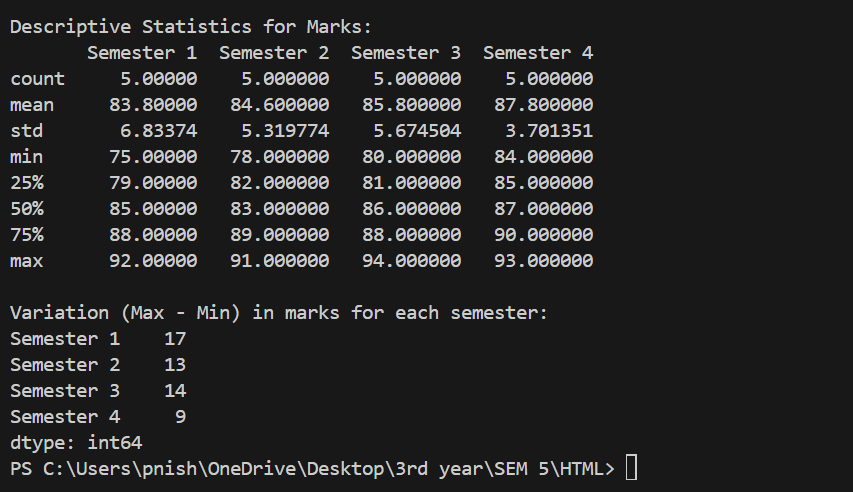
plt.legend(title="Semesters")

plt.grid(True)

plt.show()

**Output: A screenshot of a computer

Description automatically generated**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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:

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

plt.title("Student Marks in Each Semester")

plt.xlabel("Students")

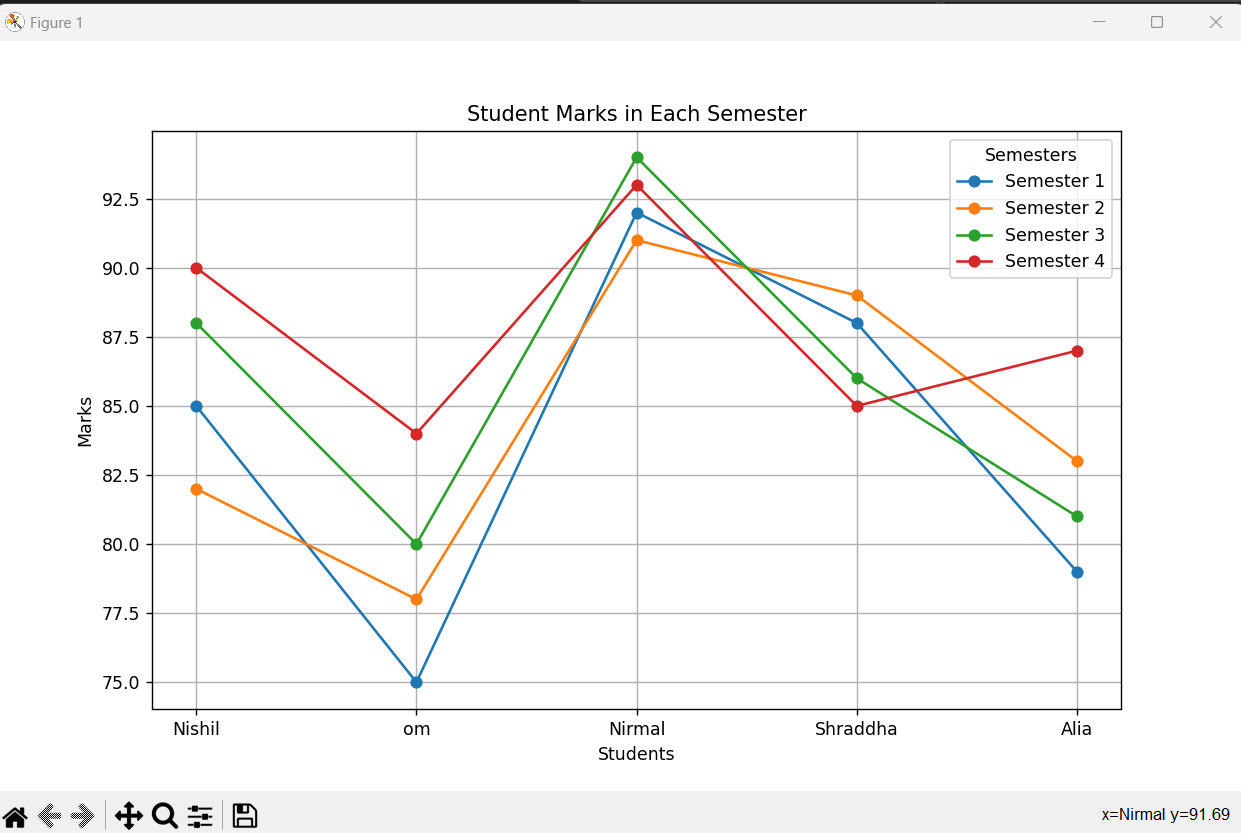
plt.ylabel("Marks")

plt.legend(title="Semesters")

plt.grid(True)

plt.show()

**output:**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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

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()

# 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)

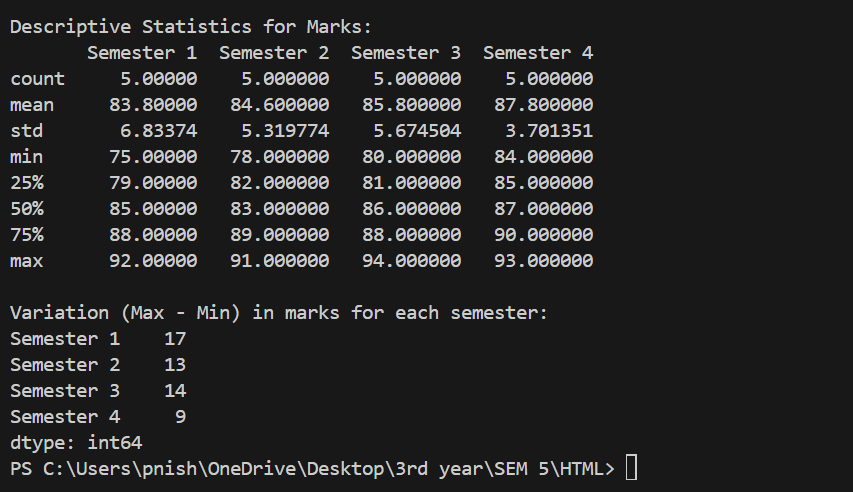
# Check the dataset after handling missing values

print("\nDataset after handling missing values:")

print(df)

**Output:** **A screenshot of a computer

Description automatically generated**



A graph with lines and points

Description automatically generatedA graph with blue bars

Description automatically generated with medium confidenceA graph with different colored squares

Description automatically generated

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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()

**Output:**

**A screenshot of a graph

Description automatically generated**

**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')

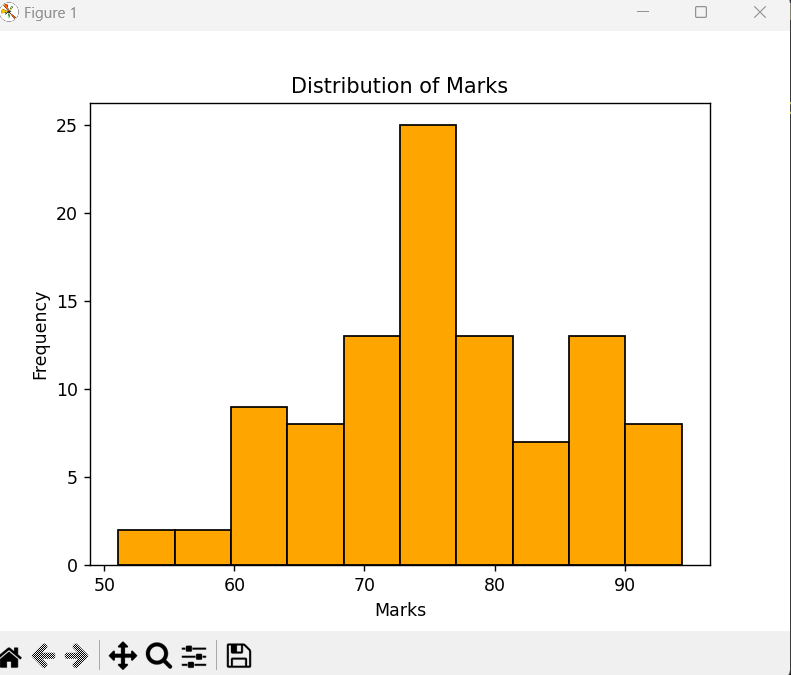
plt.xlabel('Marks')

plt.ylabel('Frequency')

plt.title('Distribution of Marks')

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')

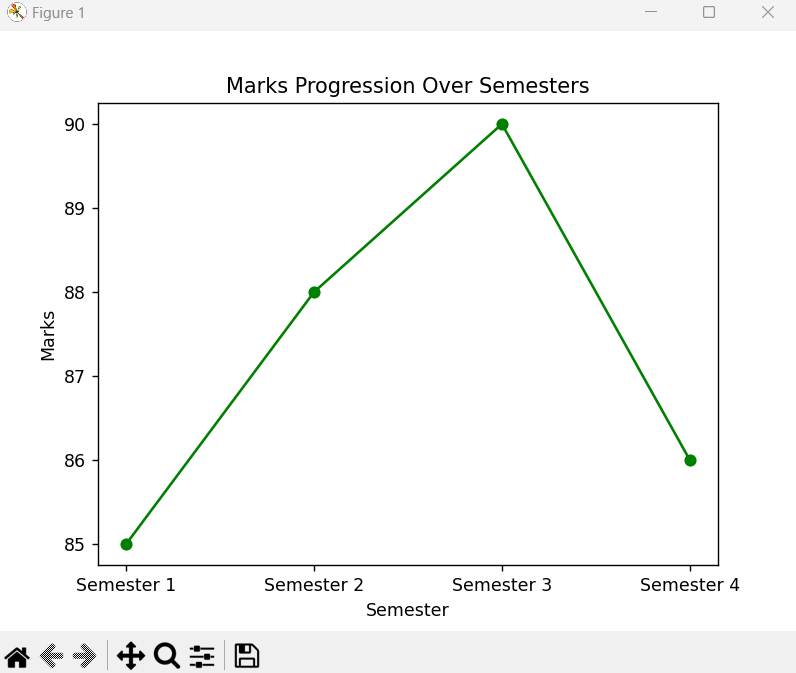
plt.xlabel('Semester')

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')

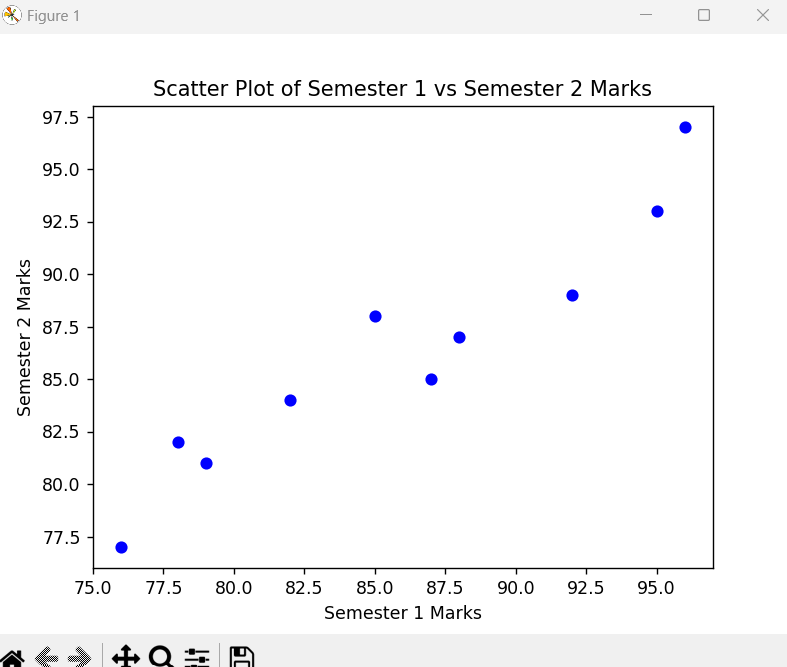
plt.xlabel('Semester 1 Marks')

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],

    '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']])

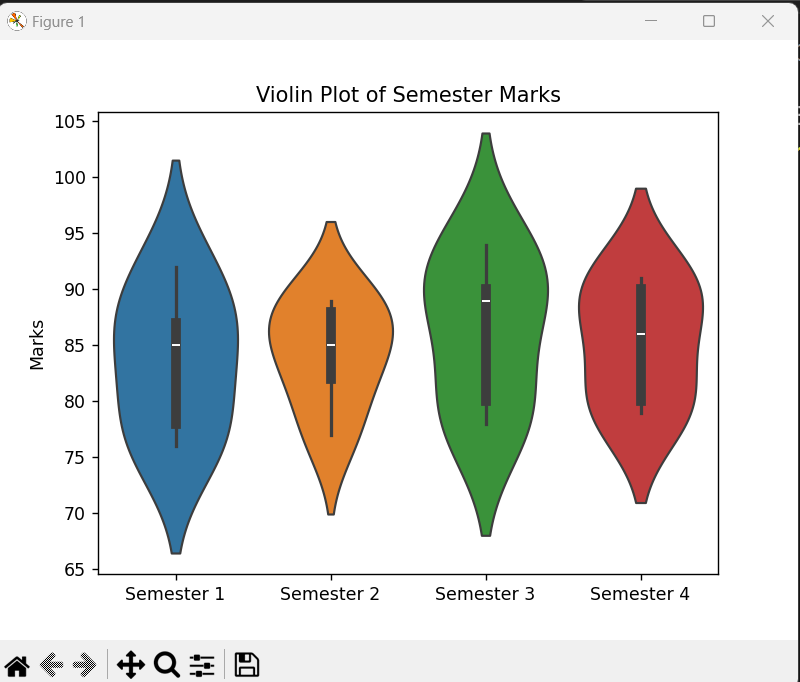
plt.xticks([0, 1, 2, 3], ['Semester 1', 'Semester 2', 'Semester 3', 'Semester 4'])

plt.ylabel('Marks')

plt.title('Violin Plot of Semester Marks')

plt.show()

**Output:**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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'])

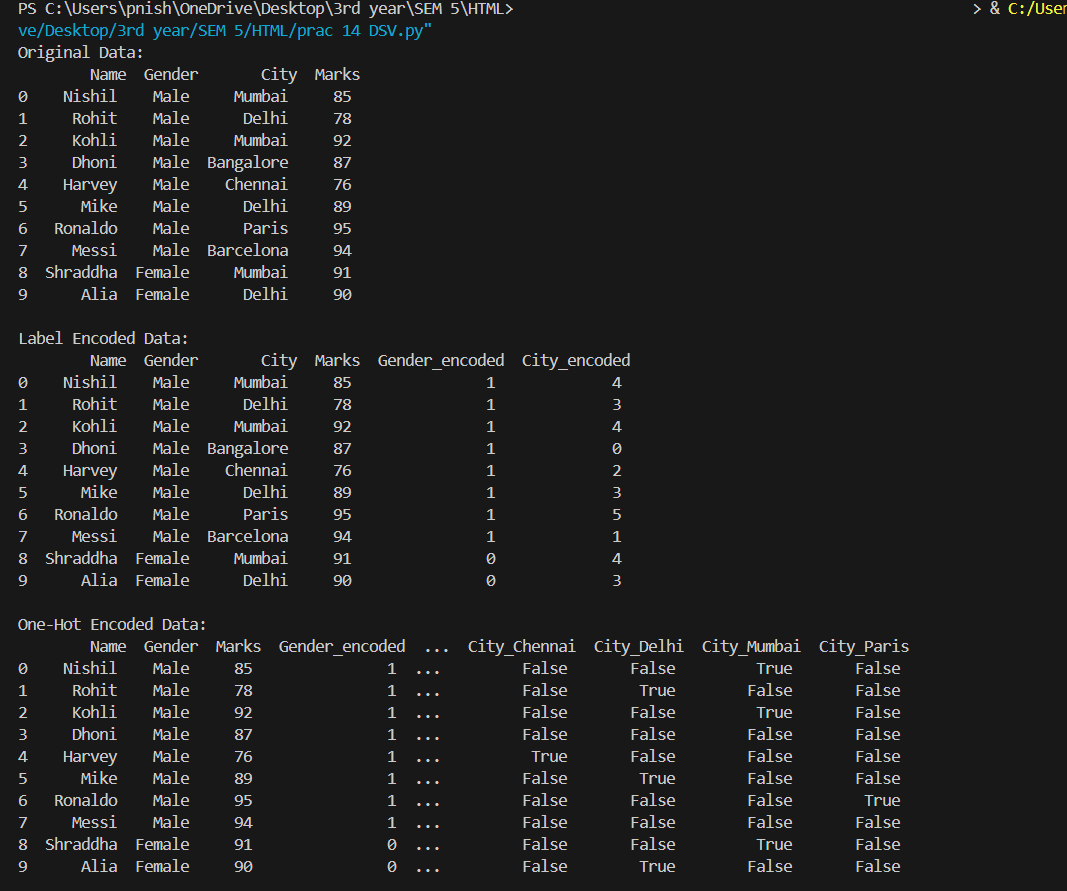
print("\nLabel Encoded Data:\n", df)

# One-Hot Encoding for City

df\_one\_hot = pd.get\_dummies(df, columns=['City'], drop\_first=True)

print("\nOne-Hot Encoded Data:\n", df\_one\_hot)

**Output:**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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.
* 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](https://www.tableau.com/).
* 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](https://powerbi.microsoft.com/).
* 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

<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

* 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.

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **A.D PATEL INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**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';

            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;

            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:**

A group of colorful shapes

Description automatically generated

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated**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"/>

        <!-- 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 -->

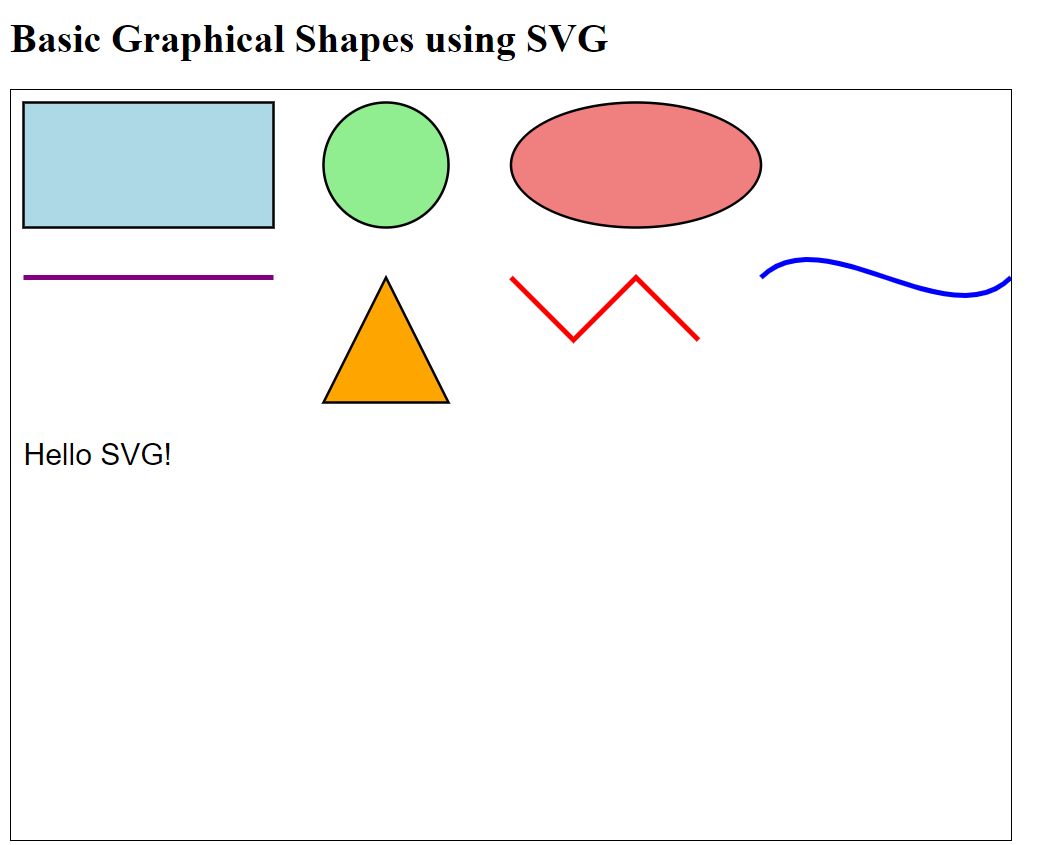
        <text x="10" y="300" font-family="Arial" font-size="24" fill="black">Hello SVG!</text>

    </svg>

</body>

</html>

**Output:**

****

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **A.D PATEL INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**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

        // 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();

        }

        // Call the function to draw the chart

        drawBarChart();

    </script>

</body>

</html>

**Output:**

**A graph with blue bars

Description automatically generated**

A logo for a company

Description automatically generatedA logo for a university

Description automatically generated **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.

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)

# 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']:

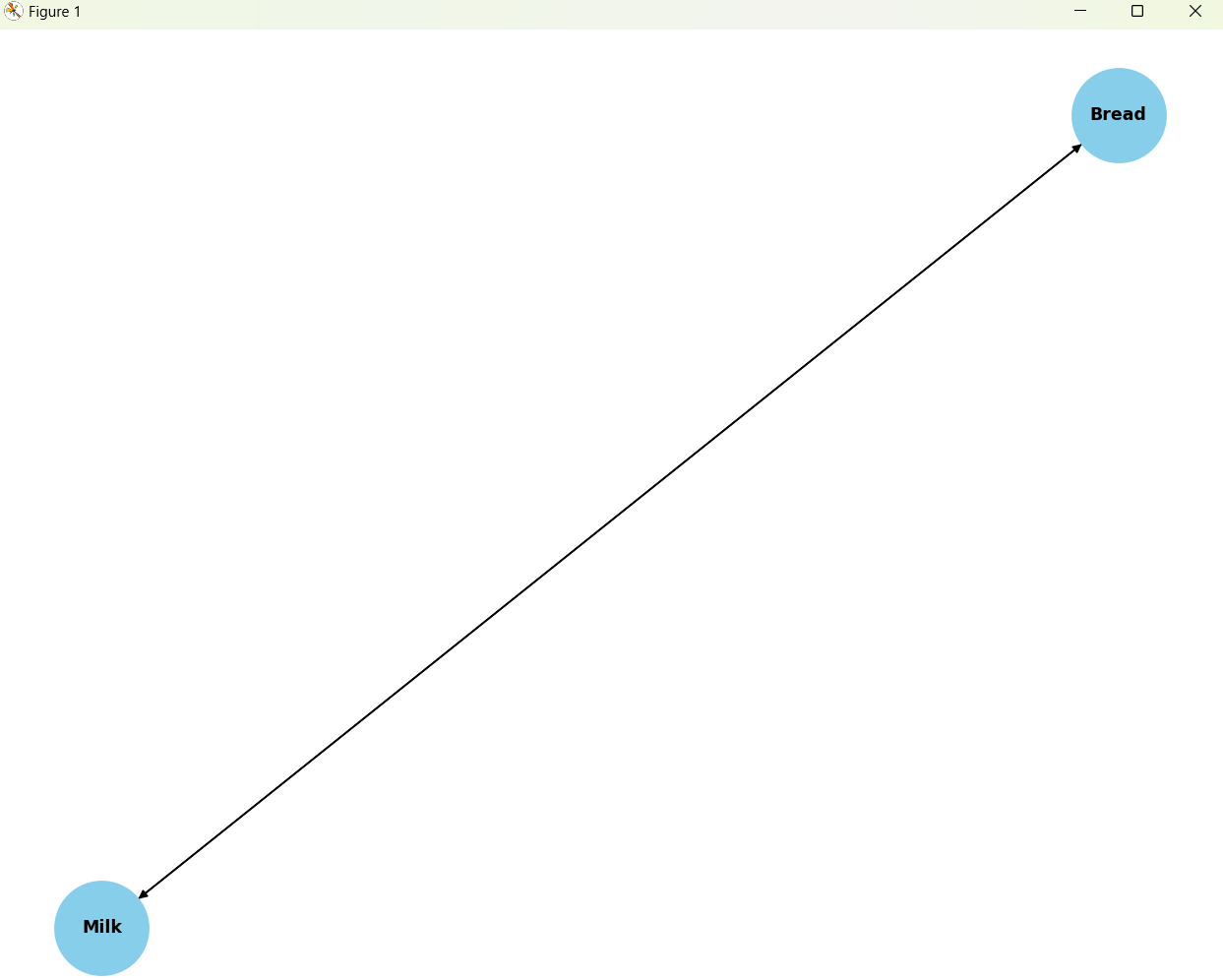
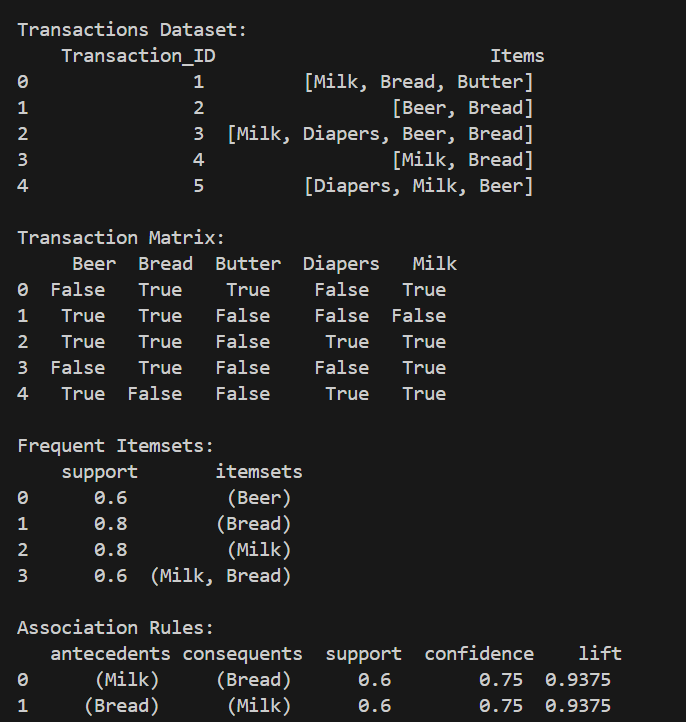
G.add\_edge(antecedent, consequent, weight=rule['lift'])

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()

  **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.

**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.