Experiment – 5.1

AIM: To Create a series of plots to analyze a given dataset.

DESCRIPTION:

1) Import necessary libraries such as pandas, matplotlib, and seaborn for data handling and visualization.

2)Load the dataset into a pandas DataFrame using functions like read_csv() or read_excel().

3) Perform initial data exploration to understand the structure and summary of the dataset.

4) Create univariate plots such as histograms or boxplots to analyze individual features.

5) Create bivariate or multivariate plots such as scatter plots, pair plots, or heatmaps to study relationships between variables.

6) Customize the plots by adding titles, labels, legends, and adjusting color schemes for better clarity.

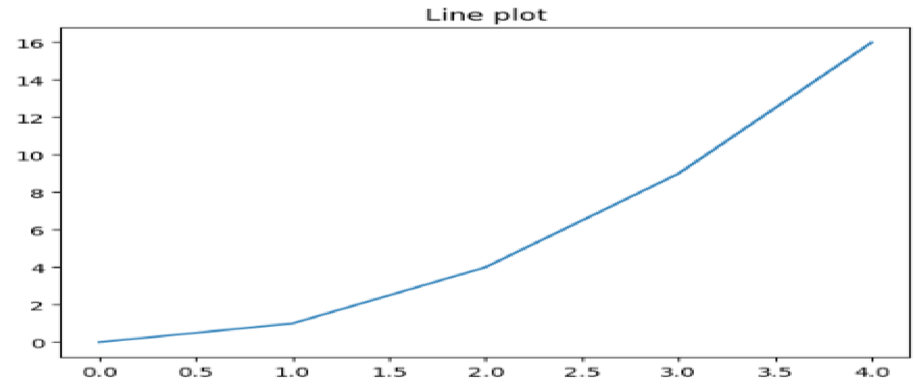
7) Display the plots using the show function or save them as image files for documentation and reporting.

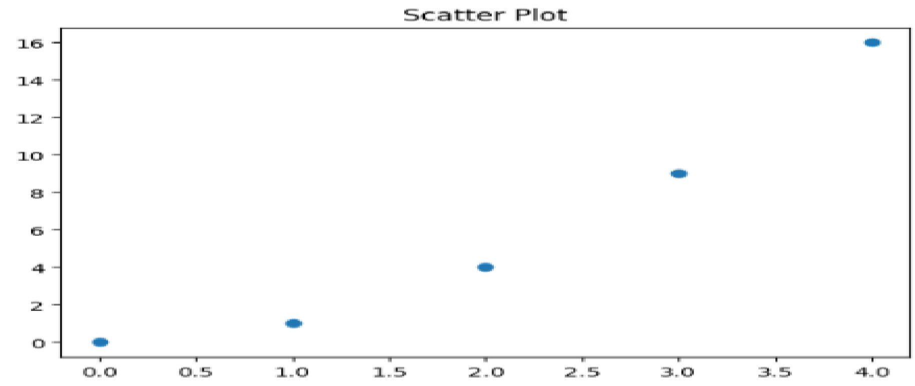
PROGRAM:

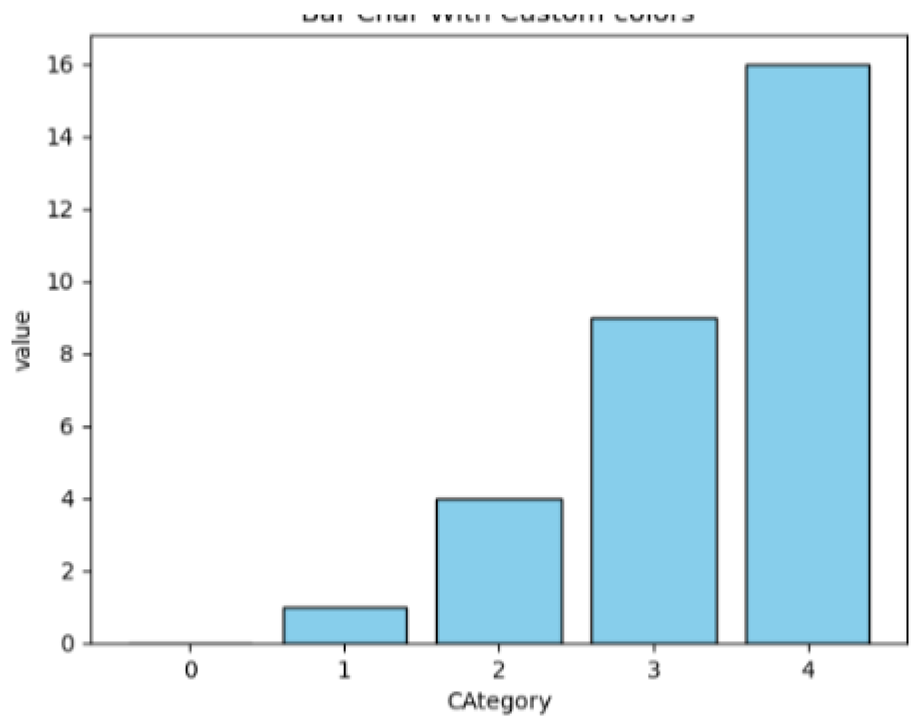
```
import matplotlib.pyplot as plt

import numpy as np

#Sample Datset

x = [0,1,2,3,4]

y = [0,1,4,9,16]

  #line plot(trend)

plt.plot(x,y)

plt.title("Line plot")

plt.show()

  #scateer plot(relation)
```

```
plt.scatter(x,y)

plt.title("Scatter Plot")

plt.show()

  #Bar Plot

plt.bar(x,y,color = 'skyblue',
edgecolor = 'black')

plt.title("Bar Char With Custom
colors")

plt.xlabel("Category")

plt.ylabel("value")

plt.show()
```

OUTPUT:







RESULT:

Hence the program To Create a series of plots to analyze a given dataset is executed and it’s output is verified successfully

Experiment – 5.2

AIM: To Generate a subplot layout with various plot types (scatter, line, bar).

DESCRIPTION:

1) Import required libraries such as matplotlib and pandas for data visualization and manipulation.

2) Load or create the dataset containing numerical values for plotting different graphs.

3) Use the plt.subplots() function to create multiple subplots within a single figure.

4) In the first subplot, create a **scatter plot** to show the relationship between two variables.

5) In the second subplot, create **a line plot** to display trends or changes of a variable over time.

6) In the third subplot, create a **bar plot** to compare categorical data or quantities.

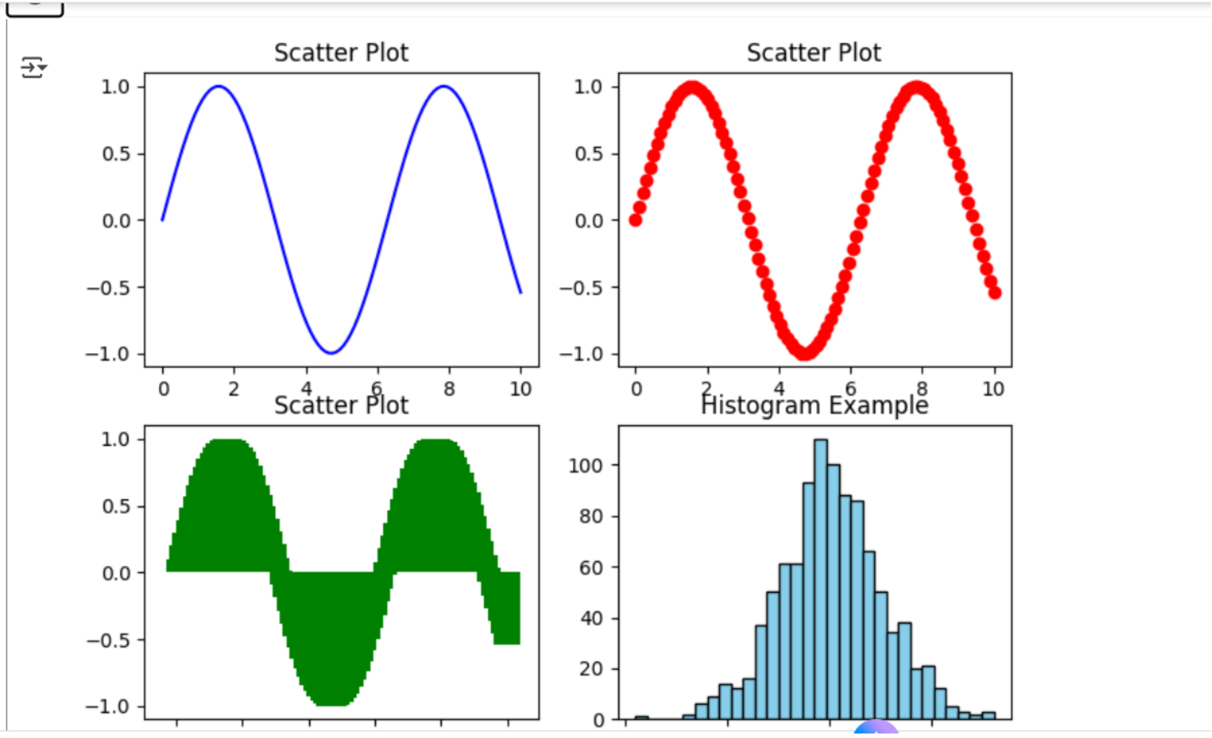
7) Add titles, axis labels, and adjust layout spacing using plt.tight_layout() before displaying all plots together with plt.show().

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
#DAta
x = np.linspace(0,10,100)
y = np.sin(x)
 #Create subplots (2 rows,2 columns)
fig,axs = plt.subplots(2,2,figsize = (8,6))

 #Line Plot
axs[0,0].plot(x,y,color = 'blue')
axs[0,0].set_title("Scatter Plot")

 #Scatter Plot
axs[0,1].scatter(x,y,color = 'red')
axs[0,1].set_title("Scatter Plot")

#bar plot
axs[1,0].bar(x,y,color = 'green')
axs[1,0].set_title("Scatter Plot")
```

```
 #Create histogram
data = np.random.randn(1000)  #1000 randoom numbers (normal distribution)
axs[1,1].hist(data , bins = 30, color = 'skyblue', edgecolor = 'black')
axs[1,1].set_title('Histogram Example')
 #Display the plot
plt.show()
#adjust space
plt.tight_layout()
plt.show()
```

OUTPUT:



RESULT:

Hence the program To Generate a subplot layout with various plot types (scatter, line, bar) is executed and it’s output is verified successfully

Experiment – 5.3

AIM: To Visualize time-series data and customize axis labels and date formats.

DESCRIPTION:

1)import necessary libraries such as pandas and matplotlib for handling and plotting time-series data.

2) Load the dataset containing date or time information and convert the date column to datetime format using pd.to_datetime().

3) Set the date column as the index of the DataFrame for easy plotting and time-based operations.

4) Use the plot() function to create a line graph representing trends or patterns over time.

5) Customize the x-axis labels to display readable date formats using plt.gcf().autofmt_xdate() or DateFormatter.

6) Add appropriate axis labels, titles, and legends to make the visualization clear and informative.

7) Adjust the date intervals and format (e.g., daily, monthly, yearly) using matplotlib.dates functions for better readability.

8) Highlight specific time ranges or events using vertical lines or shaded regions with axvline() or axvspan().

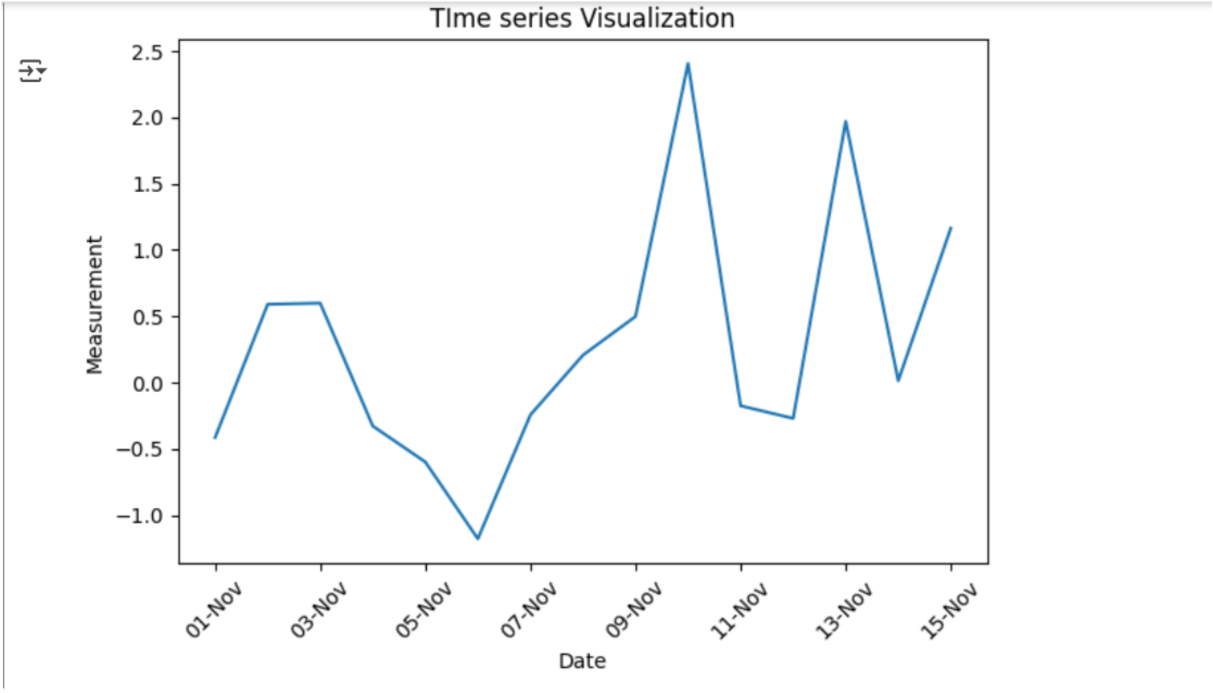
9) Save the final time-series plot using plt.savefig() for reporting or presentation purposes.

PROGRAM:

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import matplotlib.dates as mdates

#simulate time-series data

date_rng = pd.date_range(start = '2025-10-01', end = '2025-10-10', freq = 'D')

data = np.random.randn(len(date_rng))

fig,ax = plt.subplots()

ax.plot(date_rng, data)

 #Label axis and use custom date format

ax.set_xlabel("Date")

ax.set_ylabel("Measurement")

ax.set_title("TIme series Visualization")

ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
```

```
plt.xticks(rotation = 45)

plt.tight_layout()

plt.show()
```

OUTPUT:



RESULT:

Hence the program To Visualize time-series data and customize axis labels and date formats is executed and it’s output is verified successfully.

Experiment – 5.4

AIM: To Create a 3D plot.

DESCRIPTION:

1) Import necessary libraries such as matplotlib and numpy for 3D plotting and data generation.

2) Import the 3D plotting toolkit using from mpl_toolkits.mplot3d import Axes3D.

3) Prepare or generate the data for the x, y, and z axes.

4) Create a figure object using plt.figure() and add a 3D subplot with fig.add_subplot(111, projection='3d').

5) Use plotting functions such as scatter(), plot_surface(), or plot_wireframe() to visualize data in 3D.

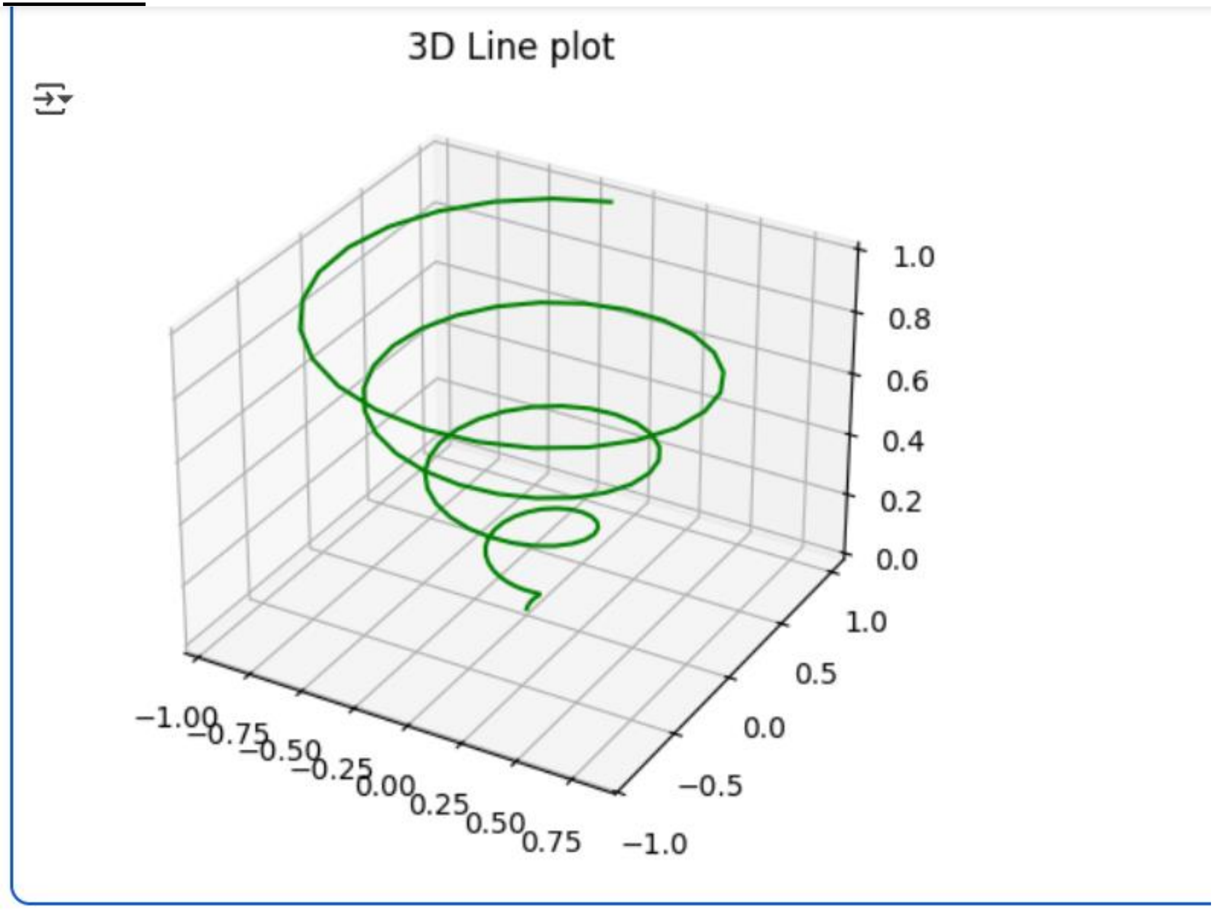
6) Customize the plot by adding axis labels, a title, and optionally a color map for better clarity.

7) Display the 3D plot using plt.show() to view the interactive 3D visualization.

PROGRAM:

```
from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

fig = plt.figure()

ax = fig.add_subplot(111,projection = '3d')

z = np.linspace(0,1,100)

x = z * np.sin(25 * z)

y = z * np.cos(25 * z)

ax.plot3D(x,y,z,color = 'green')

ax.set_title("3D Line plot")

plt.show()
```

OUTPUT:



RESULT:

Hence the program To Create a 3D plot is executed and it’s output is verified successfully.