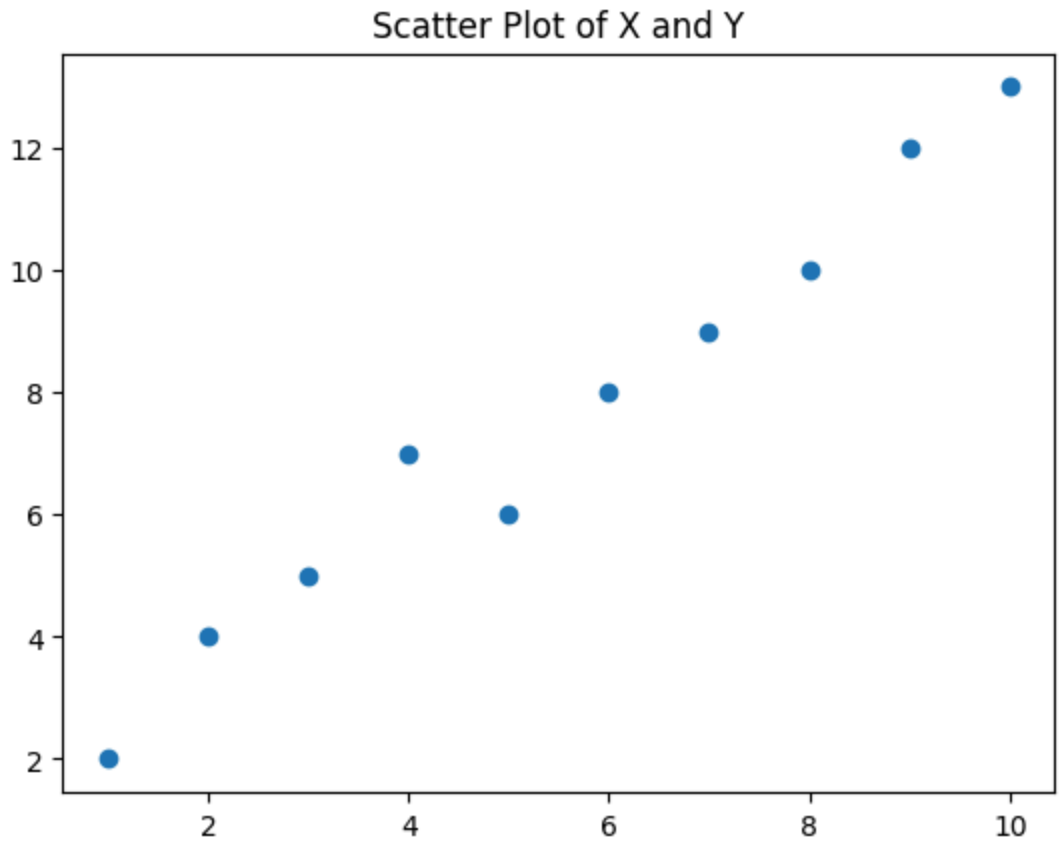
1. Create a scatter plot using Matplotlib to visualize the relationship between two arrays, x and y for the given
data.
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]

Ans

```
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]

# Create the scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot of X and Y')

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



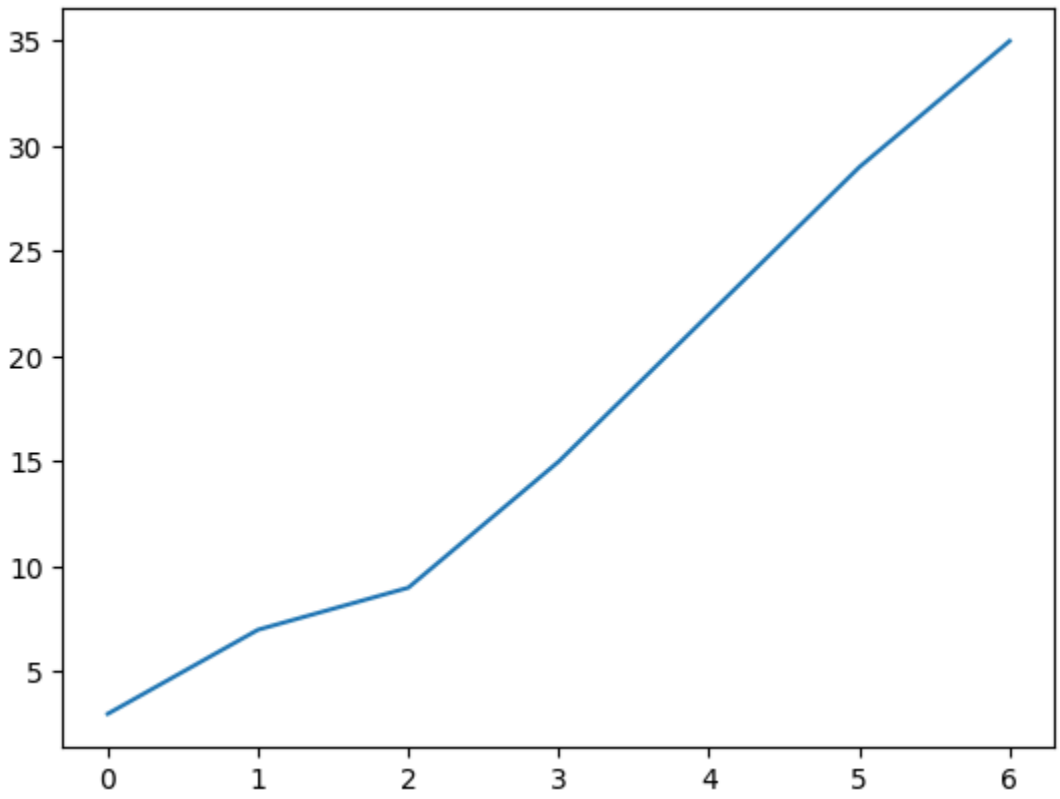
2. Generate a line plot to visualize the trend of values for the given data.

data = np.array([3, 7, 9, 15, 22, 29, 35])

Ans

```
import matplotlib.pyplot as plt

import numpy as np


# Data

data = np.array([3, 7, 9, 15, 22, 29, 35])
```

```
# Create the line plot

plt.plot(data)


# Add labels and title

plt.xlabel('Index')

plt.ylabel('Value')

plt.title('Line Plot of Data')


# Show the plot

plt.show()
```



3. Display a bar chart to represent the frequency of each item in the given array categories.

categories = ['A', 'B', 'C', 'D', 'E']

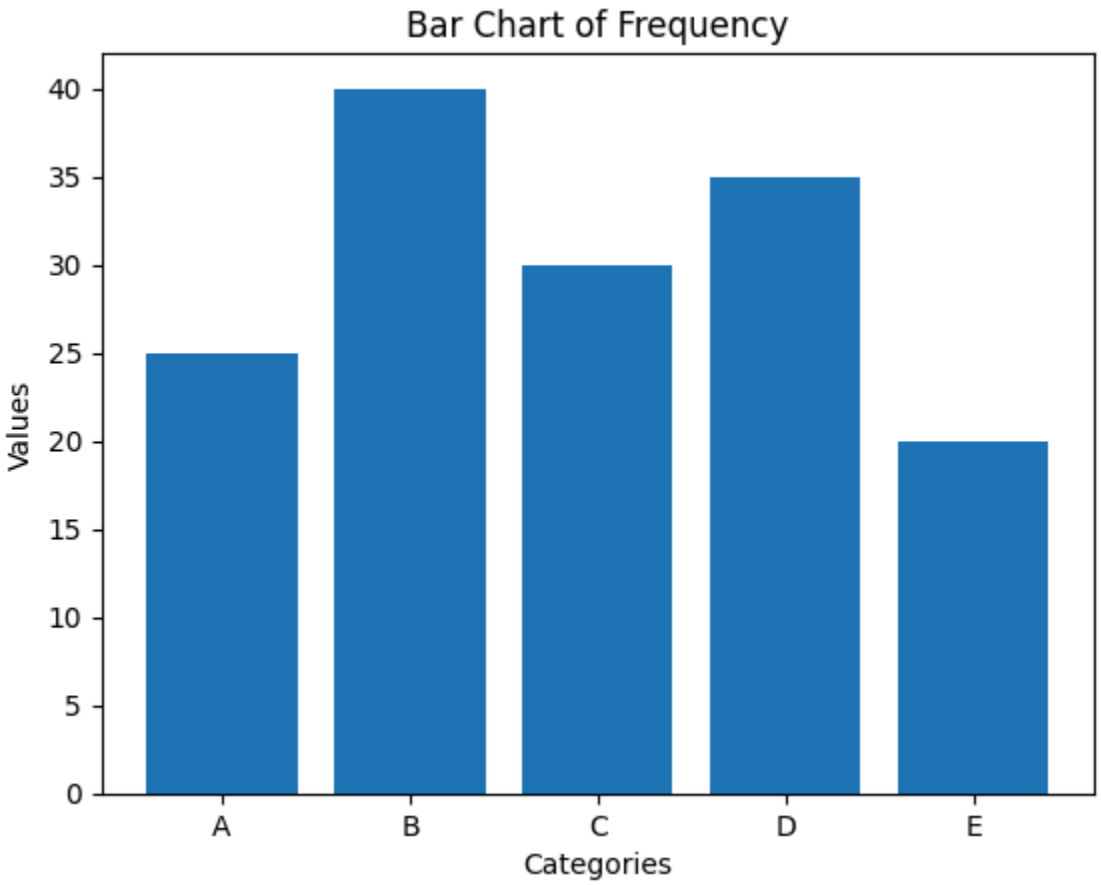
values = [25, 40, 30, 35, 20]

Ans

```
import matplotlib.pyplot as plt


# Data

categories = ['A', 'B', 'C', 'D', 'E']

values = [25, 40, 30, 35, 20]


# Create the bar chart

plt.bar(categories, values)


# Add labels and title

plt.xlabel('Categories')

plt.ylabel('Values')

plt.title('Bar Chart of Frequency')
```

Show the plot

plt.show()



4. Create a histogram to visualize the distribution of values in the array data.

data = np.random.normal(0, 1, 1000)

Ans

data = np.random.normal(0, 1, 1000)

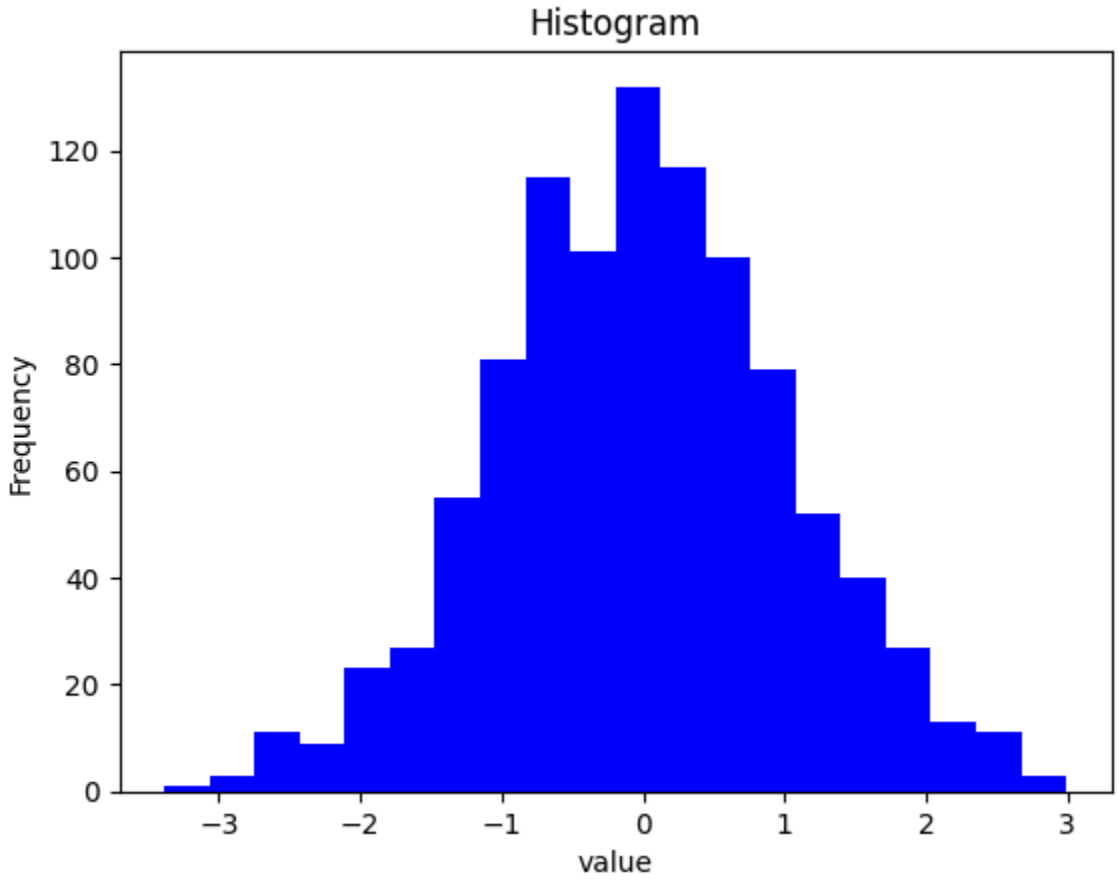
plt.hist(data, bins=20 , color= 'b')

plt.title("Histogram")

plt.xlabel("value")

plt.ylabel("Frequency")

plt.show()



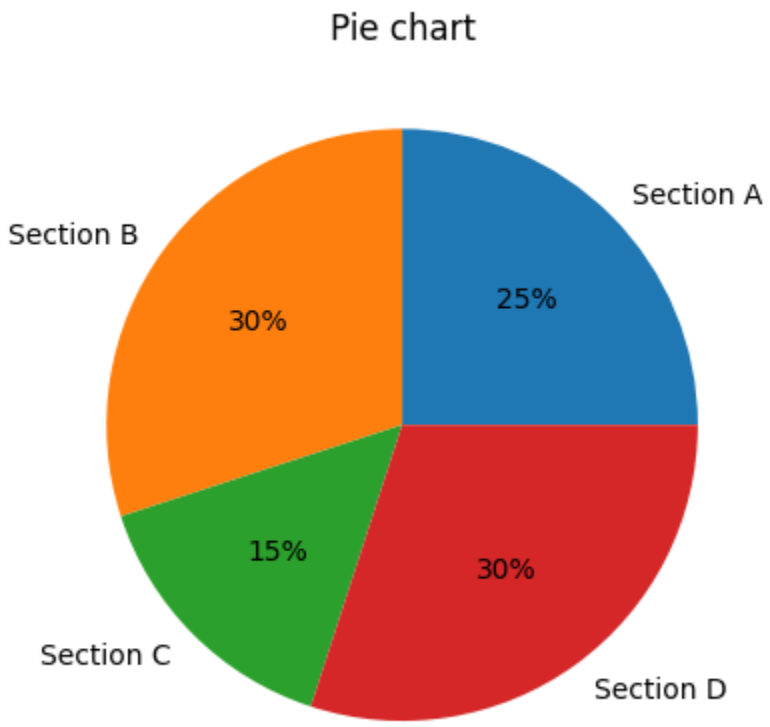
5. Show a pie chart to represent the percentage distribution of different sections in the array `sections`.

sections = ['Section A', 'Section B', 'Section C', 'Section D']

sizes = [25, 30, 15, 30]

Ans

```
sections = ['Section A', 'Section B', 'Section C', 'Section D']

sizes = [25, 30, 15, 30]


plt.pie(sizes, labels=sections, autopct='%1.f%%')

plt.title("Pie chart")

plt.show()
```

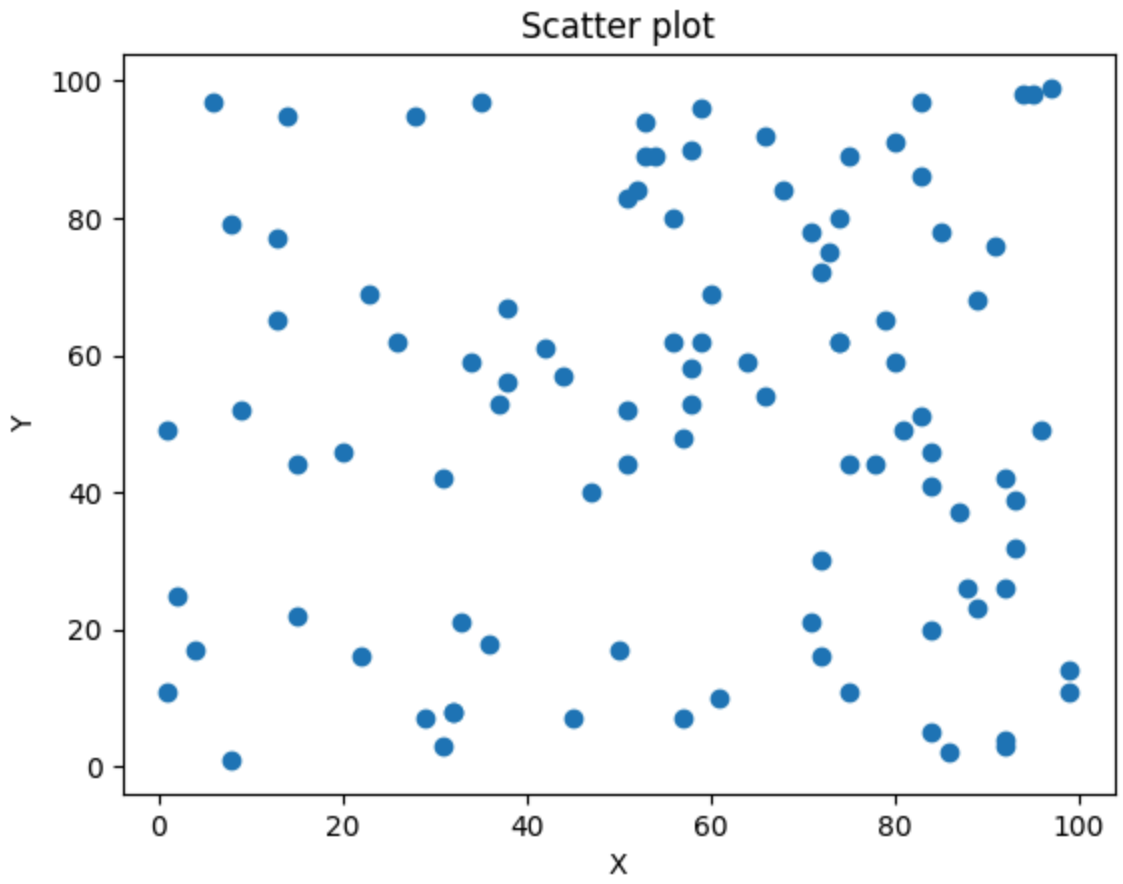


SEABORN ASSIGNMENT:

1. Create a scatter plot to visualize the relationship between two variables, by generating a synthetic dataset.

Ans

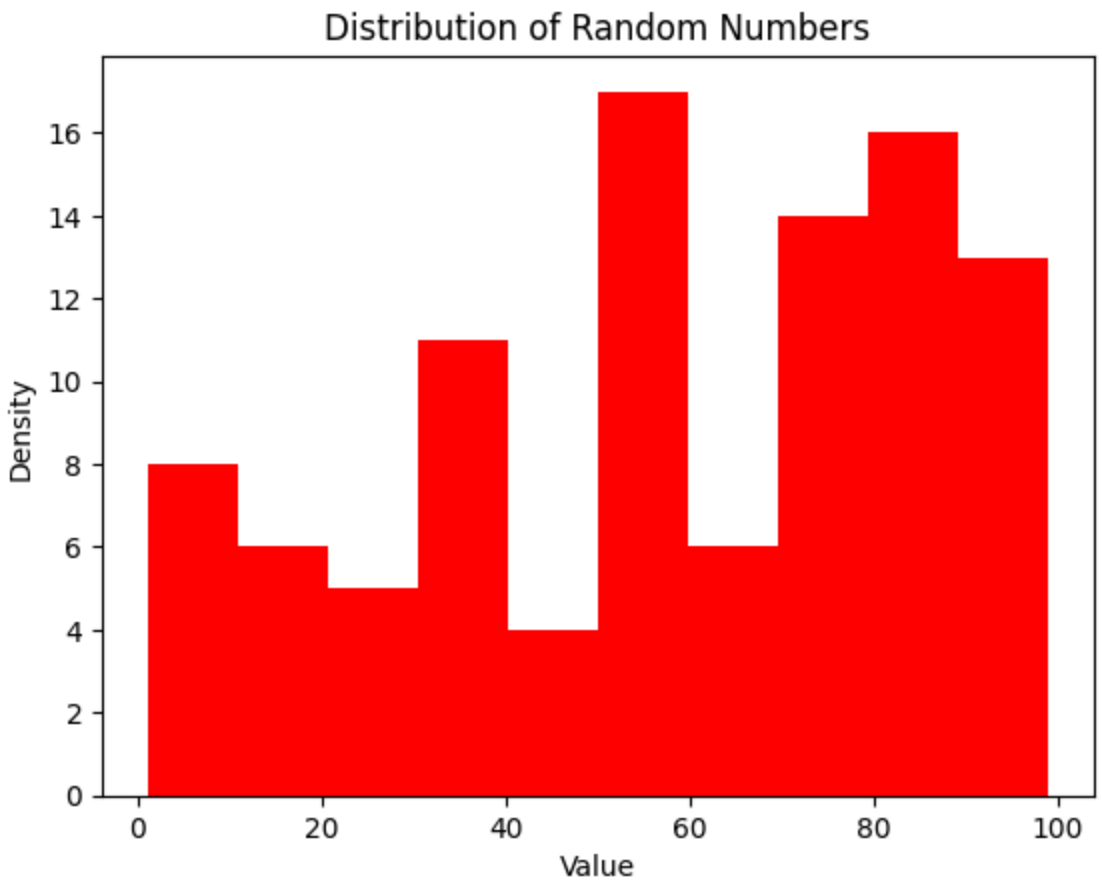
```
np.random.seed(40)

x = np.random.randint(1,100,100)

y = np.random.randint(1,100,100)

plt.scatter(x,y)

plt.title("Scatter plot")

plt.xlabel("X")

plt.ylabel("Y")

plt.show()
```



2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

Ans:-

```
plt.hist(x,bins=10,color="r")

plt.title("Distribution of Random Numbers")

plt.xlabel("Value")

plt.ylabel("Density")

plt.show()
```



3. Create a dataset representing categories and their corresponding values. Compare different categories based on numerical values.

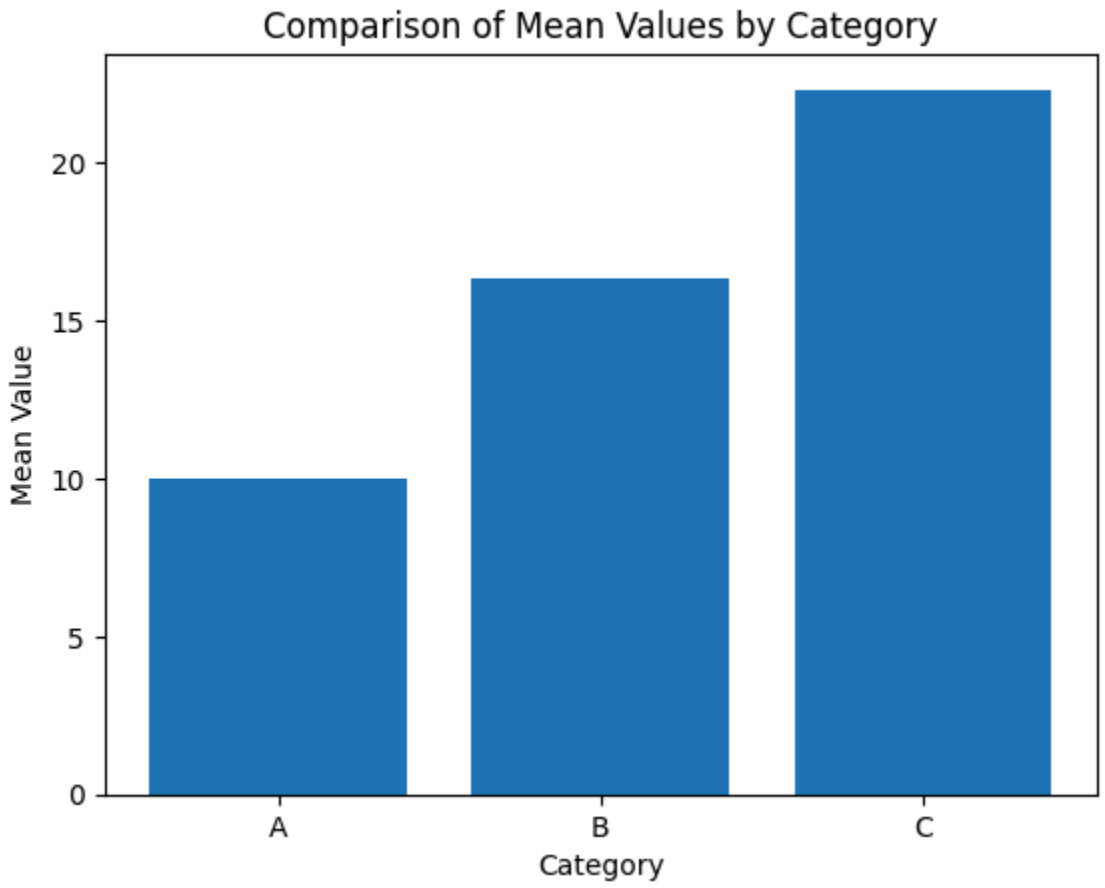
Ans:-

```
data = {'Category': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
        'Value': [10, 15, 20, 12, 18, 22, 8, 16, 25]}
df = pd.DataFrame(data)
grouped_data = df.groupby('Category')['Value'].mean()
plt.bar(grouped_data.index, grouped_data.values)
plt.xlabel('Category')
```

plt.ylabel('Mean Value')

plt.title('Comparison of Mean Values by Category')

plt.show()



4. Generate a dataset with categories and numerical values. Visualize the distribution of a numer

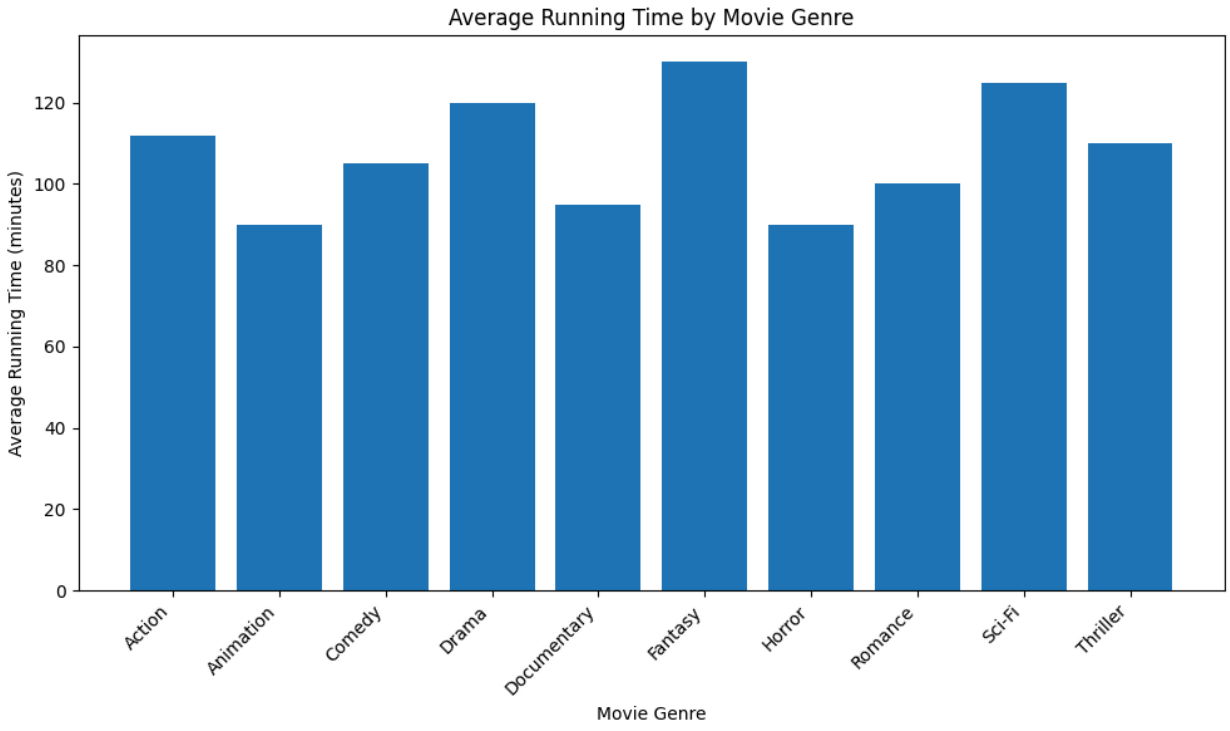
Ans:-

movie_data = {

"Category": ["Action", "Animation", "Comedy", "Drama", "Documentary", "Fantasy", "Horror", "Romance", "Sci-Fi", "Thriller"],

"Running Time (minutes)": [112, 90, 105, 120, 95, 130, 90, 100, 125, 110]

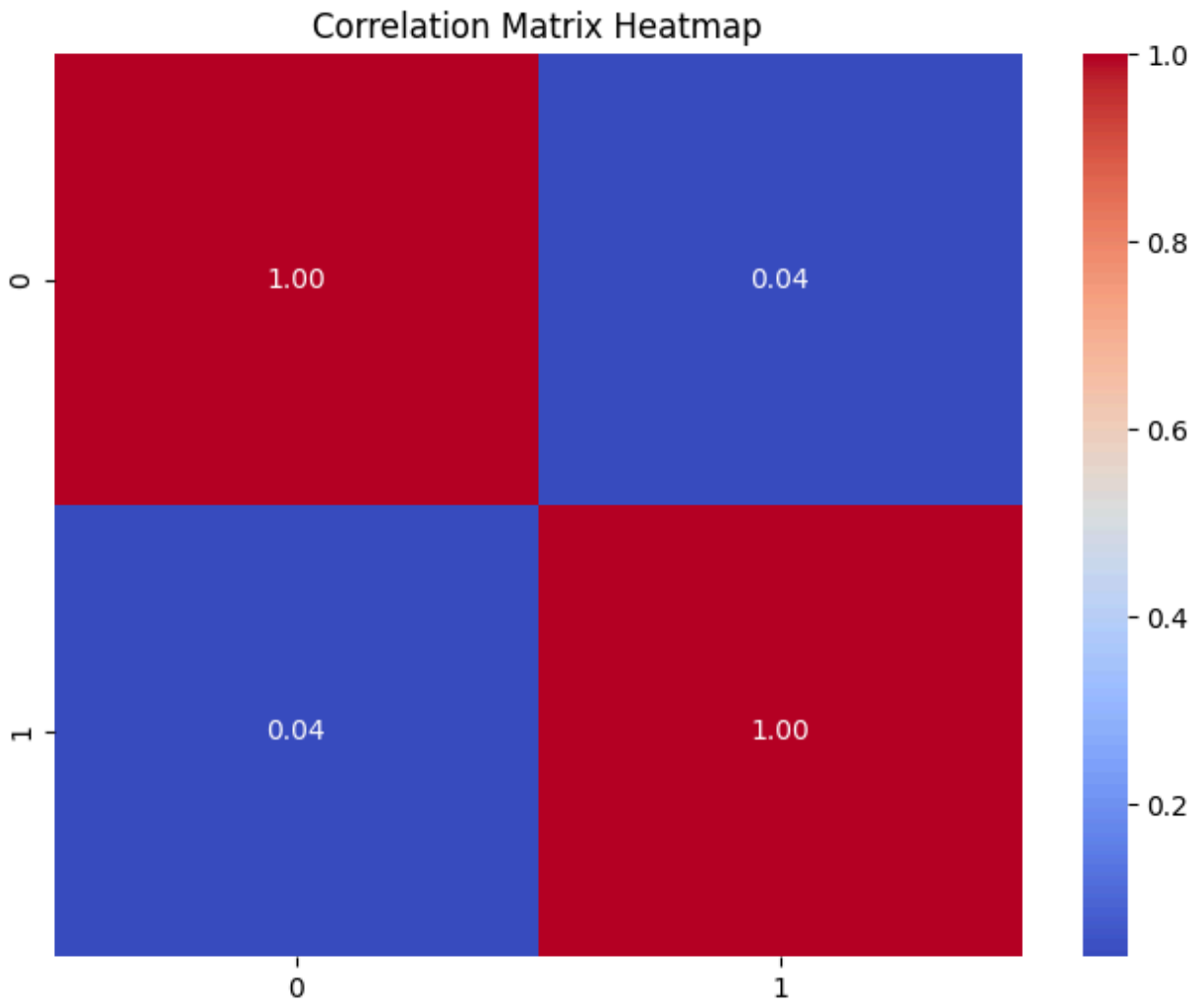
```
}


# Create a DataFrame using movie_data

df = pd.DataFrame(movie_data)


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

plt.bar(df["Category"], df["Running Time (minutes)"])

plt.xlabel("Movie Genre")

plt.ylabel("Average Running Time (minutes)")

plt.title("Average Running Time by Movie Genre")

plt.xticks(rotation=45, ha="right")

plt.tight_layout()

plt.show()ical variable across different categories.
```



5. Generate a synthetic dataset with correlated features. Visualize the correlation matrix of a dataset using a heatmap.

Ans:-

```
np.random.seed(40)

x = np.random.randint(1,100,100)

y = np.random.randint(1,100,100)

data = np.array([x, y])

corr_matrix = np.corrcoef(data)

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

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix Heatmap')

plt.show()
```



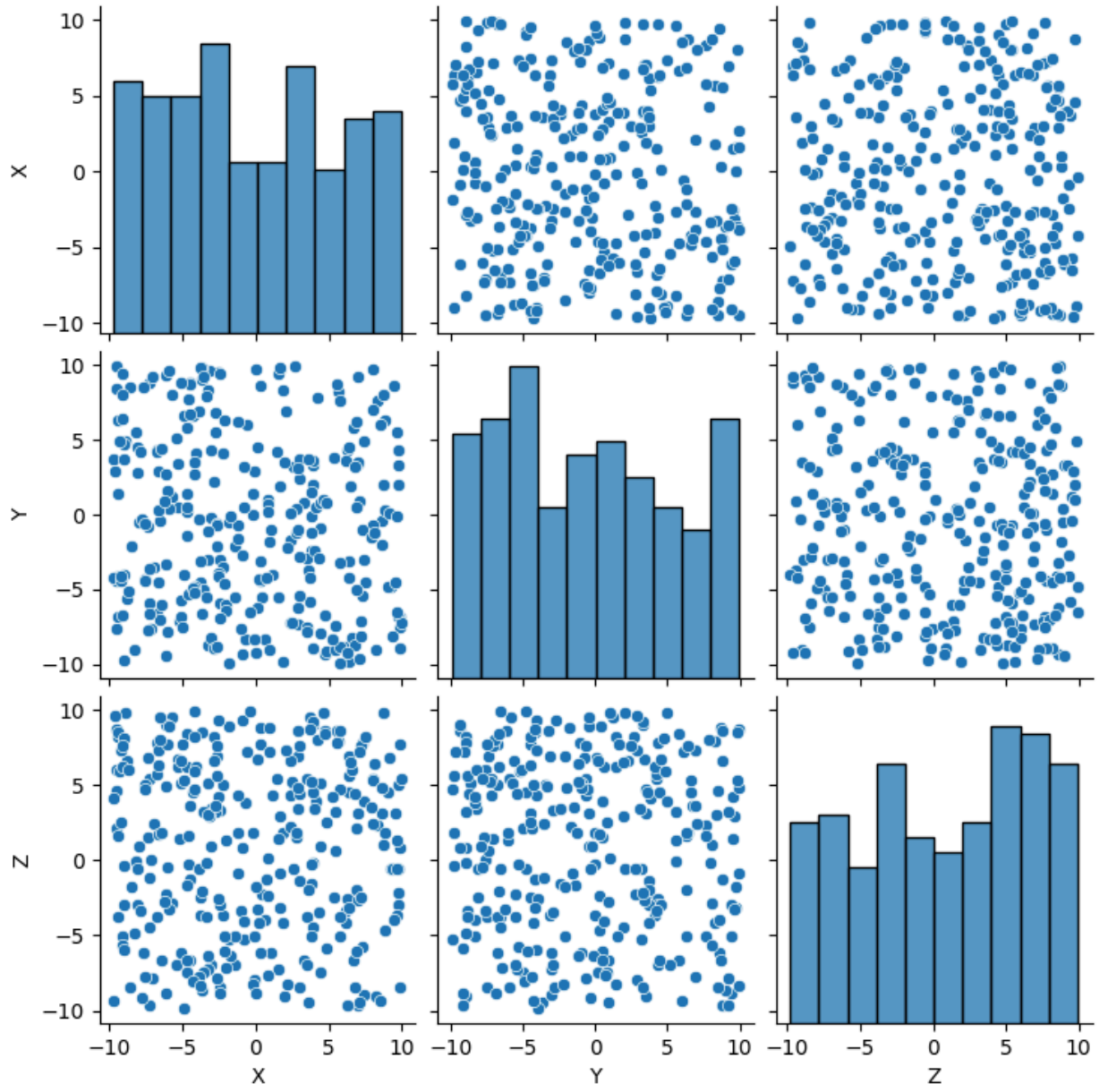
PLOTLY ASSIGNMENT:

1. Using the given dataset, to generate a 3D scatter plot to visualize the distribution of data points in a threedimensional space.

np.random.seed(30) data = { 'X': np.random.uniform(-10, 10, 300), 'Y': np.random.uniform(-10, 10, 300), 'Z': np.random.uniform(-10, 10, 300) } df = pd.DataFrame(data)

Ans:-

```
np.random.seed(30)
data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}
df = pd.DataFrame(data)
sns.pairplot(df)
plt.show()
```



2. Using the Student Grades, create a violin plot to display the distribution of scores across different grade categories.

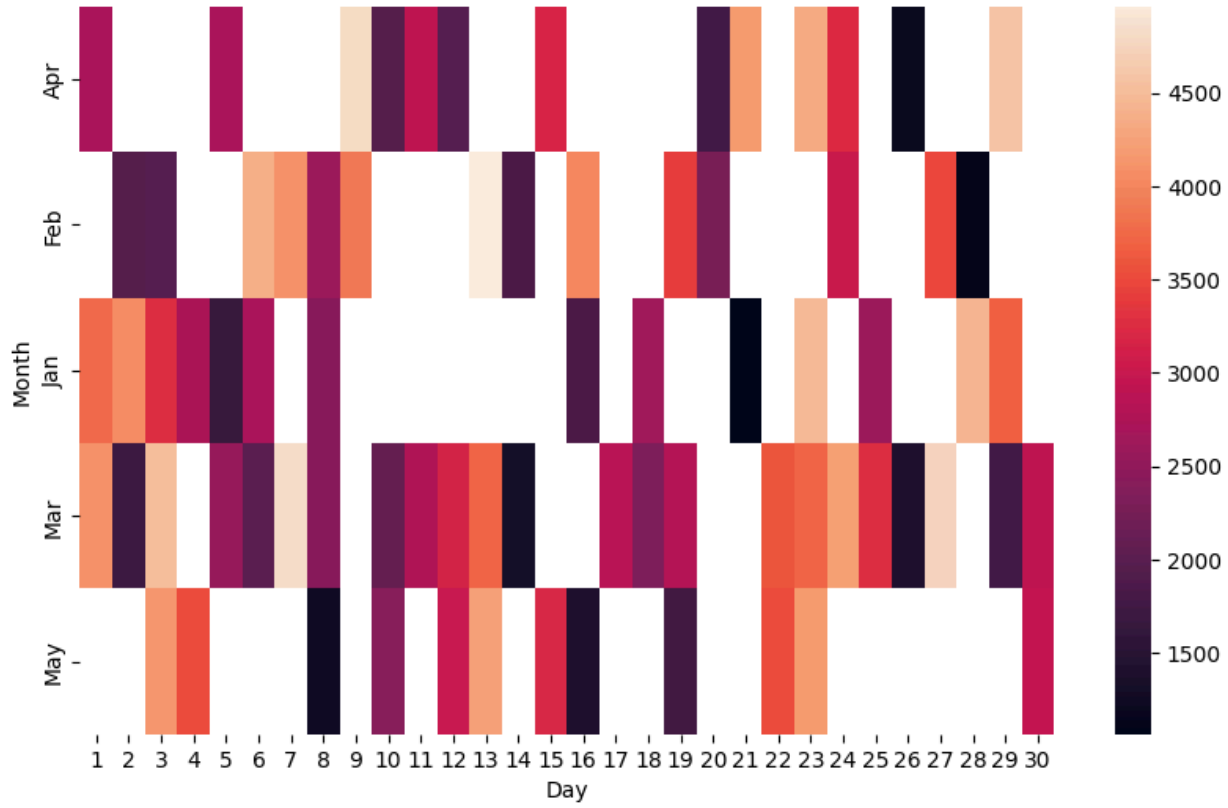
np.random.seed(15) data = { 'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200), 'Score': np.random.randint(50, 100, 200) } df = pd.DataFrame(data‡ €™

Using the sales data, generate a heatmap to visualize the variation in sales across different months and days. np.random.seed(20) data = { 'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100), 'Day':

np.random.choice(range(1, 31), 100), 'Sales': np.random.randint(1000, 5000, 100) } df = pd.DataFrame(data)

Ans:-

```
np.random.seed(15)
data = {
    'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
    'Score': np.random.randint(50, 100, 200)
}
df = pd.DataFrame(data)
# Using the sales data, generate a heatmap to visualize the variation in sales across
# different months and days.

np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)
plt.figure(figsize=(10 , 6))
sns.heatmap(df.pivot_table(index='Month', columns='Day', values='Sales'))
```



3. Using the sales data, generate a heatmap to visualize the variation in sales across diffenp.random.seed(20)

data = {

'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),

'Day': np.random.choice(range(1, 31), 100),

'Sales': np.random.randint(1000, 5000, 100)

}

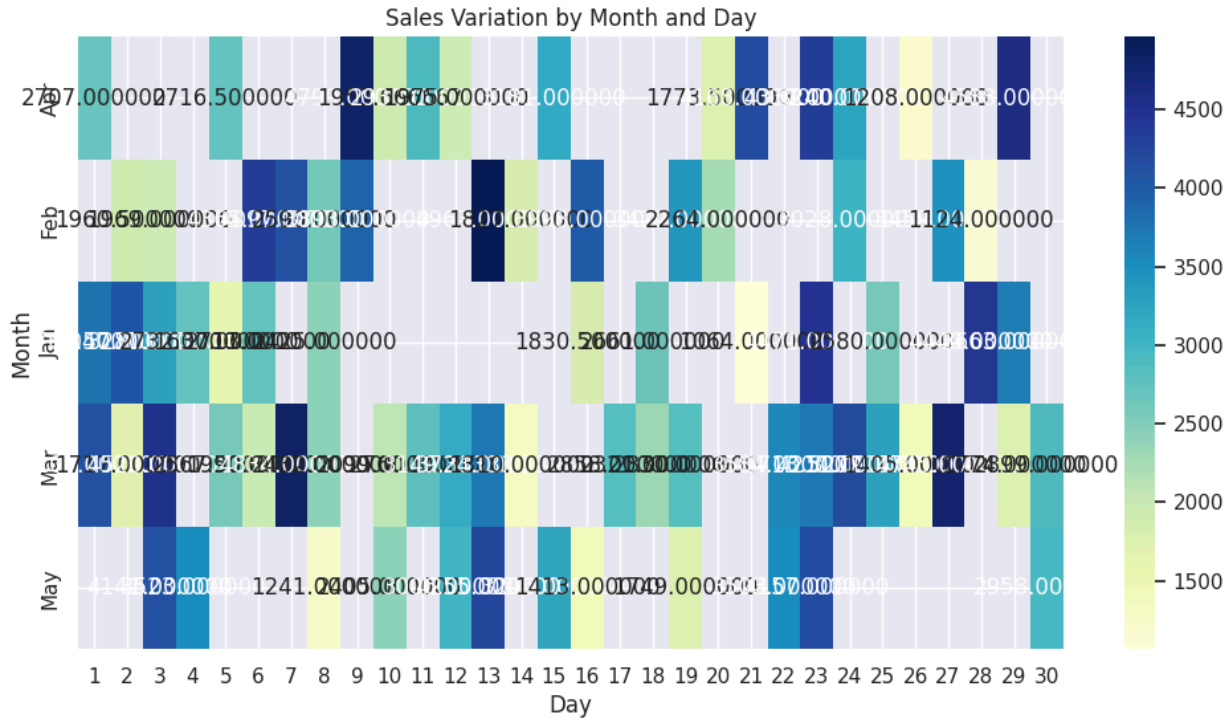
df = pd.DataFrame(data)rent months and days.

Ans:-

Pivot the DataFrame to create a matrix of sales values

pivot_df = df.pivot_table(index='Month', columns='Day', values='Sales', aggfunc='mean')

```
# Create the heatmap
sns.set(rc={'figure.figsize': (12, 6)})
sns.heatmap(pivot_df, annot=True, cmap='YlGnBu', fmt='0f')
plt.title('Sales Variation by Month and Day')
plt.show()
```



4. Using the given x and y data, generate a 3D surface plot to visualize the function

x = np.linspace(-5, 5, 100) y = np.linspace(-5, 5, 100) x, y = np.meshgrid(x, y) z = np.sin(np.sqrt(x2 + y**2)) data = { 'X': x.flatten(), 'Y': y.flatten(), 'Z': z.flatten() } df = pd.DataFrame(data)**

Ans:-

x = np.linspace(-5, 5, 100)

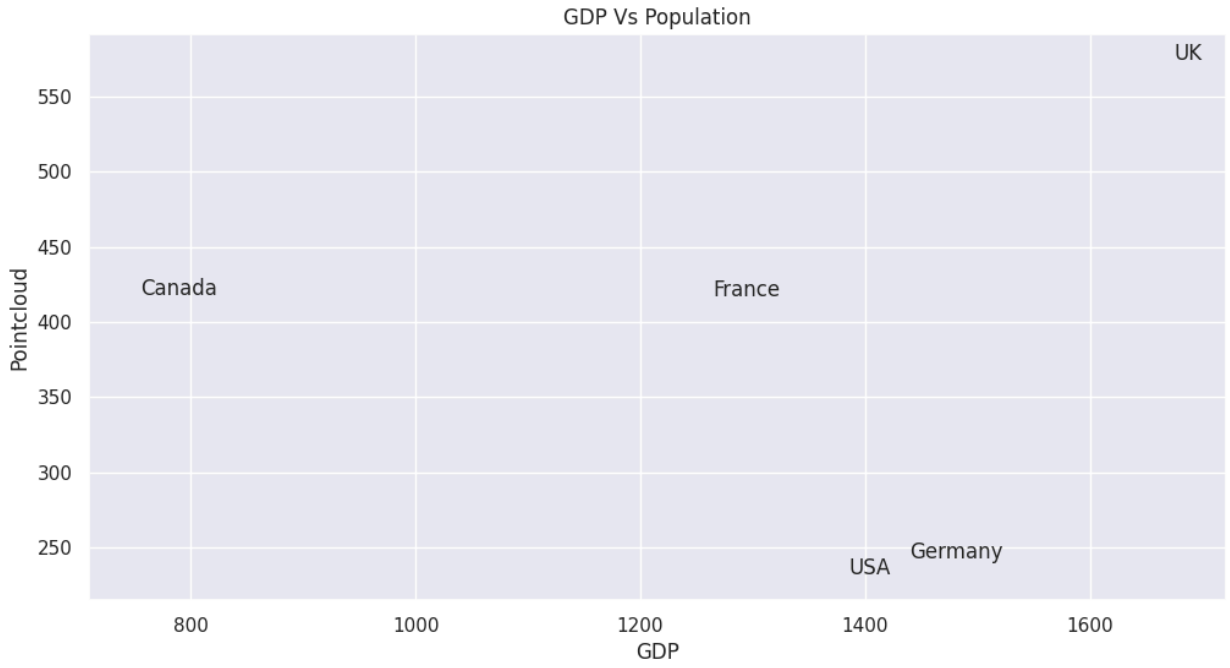
```
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)
fig = px.scatter_3d(df,x = "X" ,y= 'Y',z= 'Z' )
fig.show()
```

5. Using the given dataset, create a bubble chart to represent each country's population (y-axis), GDP (xaxis), and bubble size proportional to the population.

np.random.seed(25) data = { 'Country': ['USA', 'Canada', 'UK', 'Germany', 'France'], 'Population': np.random.randint(100, 1000, 5), 'GDP': np.random.randint(500, 2000, 5) } df = pd.DataFrame(data)

Ans:-np.random.seed(25)

```
data = {
    'Country': ['USA', 'Canada', 'UK',
'Germany', 'France'],
    'Population':
np.random.randint(100, 1000, 5),
    'GDP': np.random.randint(500, 2000,
5)
}
df = pd.DataFrame(data)
plt.scatter(data['GDP'], data['Population'], s=data['Population'] / 1000000, alpha=0.7)
plt.xlabel('GDP')
plt.ylabel('Pointcloud')
plt.title('GDP Vs Population')
for i, row in df.iterrows():
    plt.annotate(row['Country'], (row['GDP'], row['Population']))
plt.show()
```

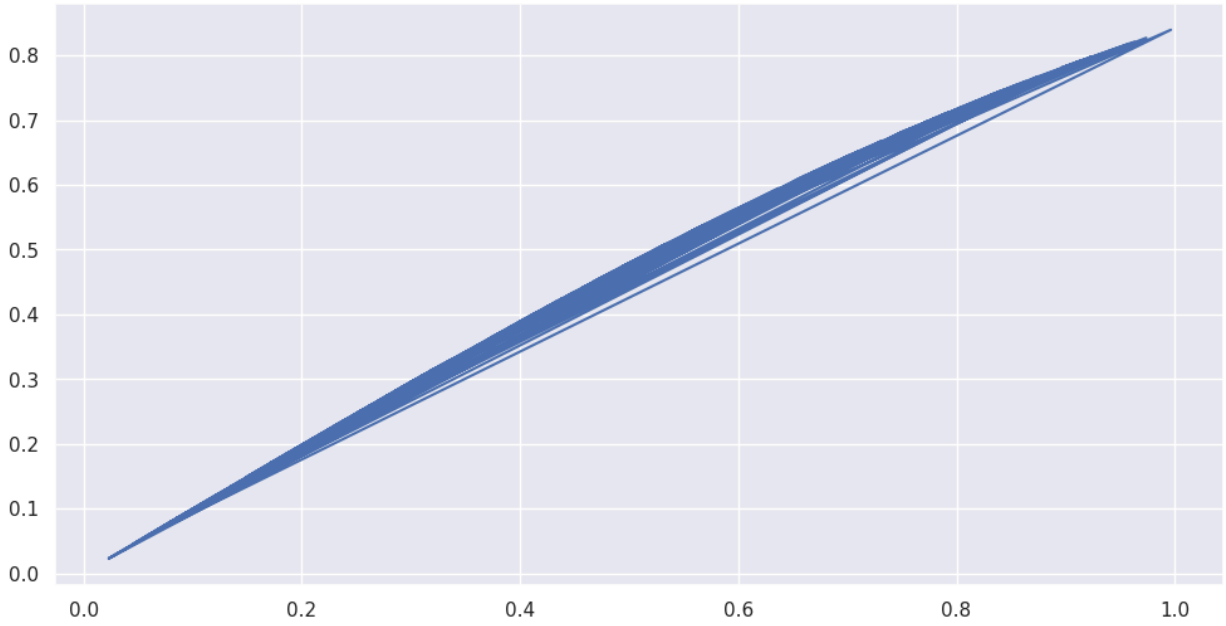


BOKEH ASSIGNMENT:

1.Create a Bokeh plot displaying a sine wave. Set x-values from 0 to 10 and y-values as the sine of x.

Ans:-

```
x = np.random.rand(100)

y = np.sin(x)

plt.plot(x ,y)

plt.show()
```



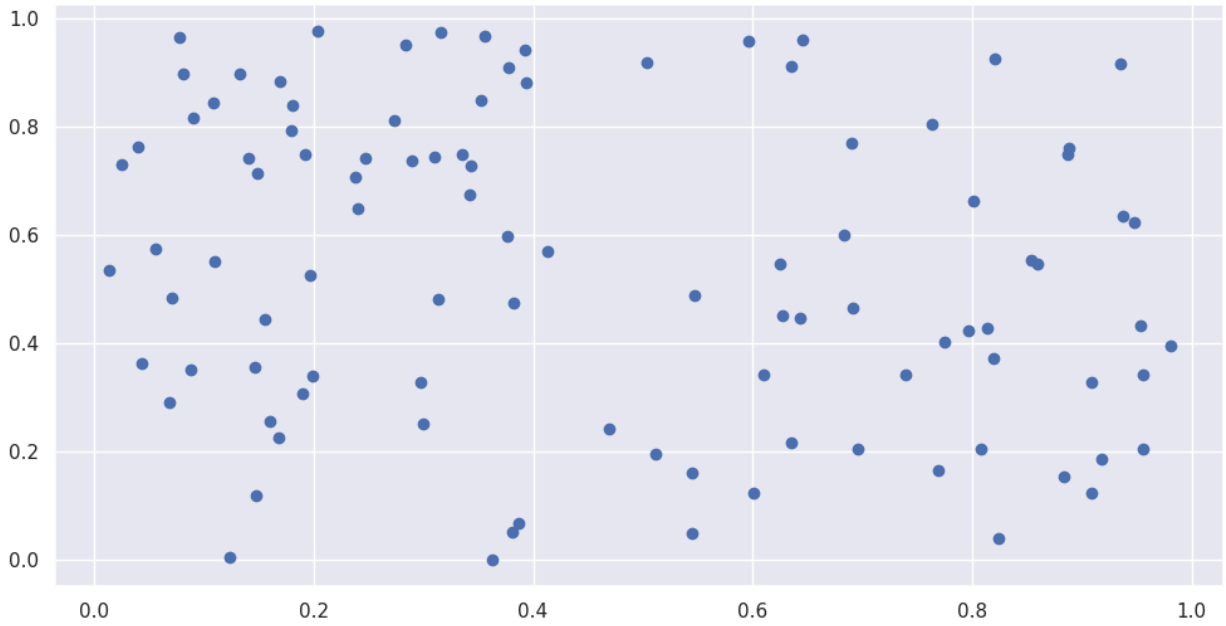
2.Create a Bokeh scatter plot using randomly generated x and y values. Use different sizes and colors for the markers based on the 'sizes' and 'colors' columns.

And:-

```
N = 100
x = np.random.rand(N)
y = np.random.rand(N)
```

```
# Generate random sizes and colors
sizes = np.random.randint(10, 50, N)
colors = np.random.choice(['red', 'blue', 'green'], N)

plt.scatter(x, y )
```

plt.show()



3. Generate a Bokeh bar chart representing the counts of different fruits using the following dataset.

fruits = ['Apples', 'Oranges', 'Bananas', 'Pears'] counts = [20, 25, 30, 35]

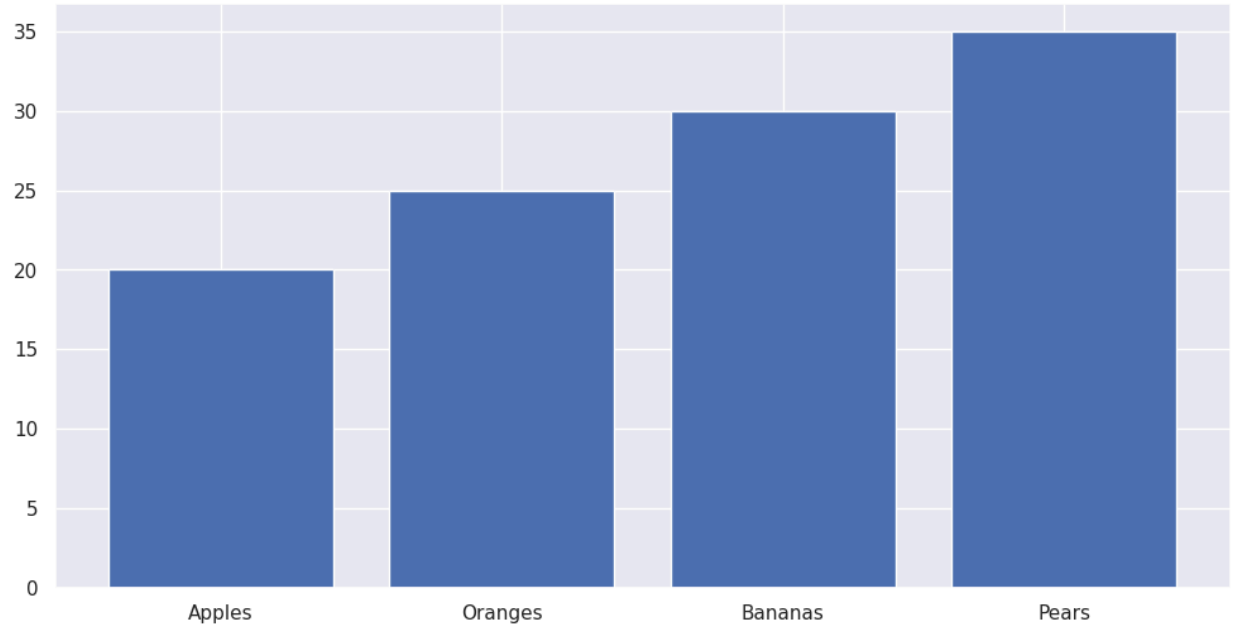
Ans:-

fruits = ['Apples', 'Oranges', 'Bananas', 'Pears']

counts = [20, 25, 30, 35]

plt.bar(fruits , counts)

plt.show()

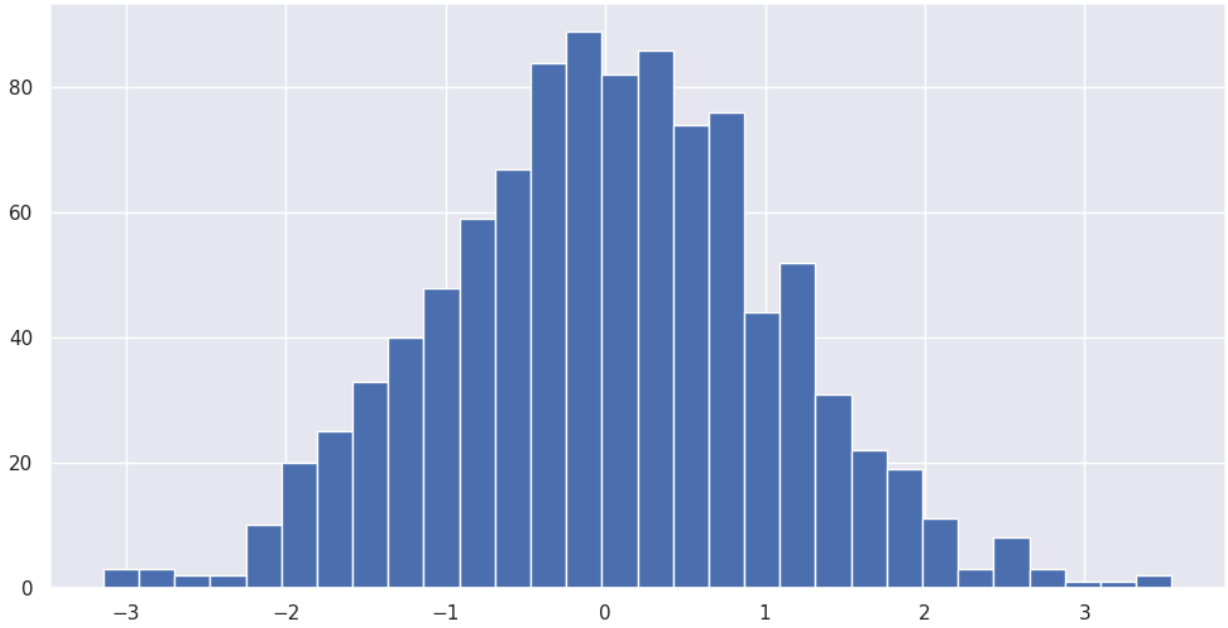


4. Create a Bokeh histogram to visualize the distribution of the given data.

data_hist = np.random.randn(1000) hist, edges = np.histogram(data_hist, bins=30)

Ans:-

```
data_hist = np.random.randn(1000)

hist, edges = np.histogram(data_hist, bins=30)

#BOKEH

plt.figure()

plt.hist(data_hist, bins=30)

plt.show()
```



5. Create a Bokeh heatmap using the provided dataset.

data_heatmap = np.random.rand(10, 10) x = np.linspace(0, 1, 10) y = np.linspace(0, 1, 10) xx, yy = np.meshgrid(x, y)

Ans:-

prompt: data_heatmap = np.random.rand(10, 10)

x = np.linspace(0, 1, 10)

y = np.linspace(0, 1, 10)

xx, yy = np.meshgrid(x, y)

heatmap

data_heatmap = np.random.rand(10, 10)

x = np.linspace(0, 1, 10)

```
y = np.linspace(0, 1, 10)

xx, yy = np.meshgrid(x, y)


plt.imshow(data_heatmap, cmap='viridis')

plt.colorbar()

plt.show()
```