MATPLOTLIB ASSIGNMENT:

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]

ANSWER:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]
plt.scatter(x, y)
plt.title("Scatter Plot of x vs y")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.show()
```

Q2. Generate a line plot to visualize the trend of values for the given data.

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

ANSWER:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([3, 7, 9, 15, 22, 29, 35])
plt.plot(data, marker='o')
plt.title("Line Plot of Data Trend")
plt.xlabel("Index")
plt.ylabel("Value")
plt.grid(True)
```

```
plt.show()
```

Q3.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]

ANSWER:

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D', 'E']
values = [25, 40, 30, 35, 20]

plt.bar(categories, values)
plt.title("Bar Chart of Category Values")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.grid(axis='y')
plt.show()
```

Q4. Create a histogram to visualize the distribution of values in the array data.

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

ANSWER:

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

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

plt.hist(data, bins=30)
plt.title("Histogram of Data Distribution")
```

```
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```

Q5. 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]

ANSWER:

```
import matplotlib.pyplot as plt
sections = ['Section A', 'Section B', 'Section C', 'Section D']
sizes = [25, 30, 15, 30]
plt.pie(sizes, labels=sections, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart of Section Distribution")
plt.show()
```

SEABORN ASSIGNMENT:

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

ANSWER: # Seaborn Scatter Plot using Synthetic Dataset

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

sns.set(style="whitegrid")
```

```
# Generate synthetic dataset
np.random.seed(42)
x = np.random.randint(1, 100, 50)
y = x + np.random.randint(-10, 10, 50)   # relationship with noise

df = pd.DataFrame({"x": x, "y": y})

# Scatter plot
plt.figure(figsize=(6,4))
sns.scatterplot(data=df, x="x", y="y")
plt.title("Scatter Plot (Synthetic Dataset)")
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show()
```

Q2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

ANSWER:

```
# Seaborn Distribution Plot using Random Dataset

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

sns.set(style="whitegrid")

# Generate random dataset
np.random.seed(42)
data = np.random.randn(1000)  # 1000 random numbers (normal distribution)

df = pd.DataFrame({"values": data})
```

```
# Distribution plot (Histogram + KDE)
plt.figure(figsize=(6,4))
sns.histplot(df["values"], bins=30, kde=True)
plt.title("Distribution of Random Numbers")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```

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

ANSWER:

```
# Seaborn Bar Plot to Compare Categories

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")

# Create dataset
categories = ["A", "B", "C", "D", "E"]
values = [25, 40, 30, 35, 20]

df = pd.DataFrame({"Category": categories, "Value": values})

# Bar plot
plt.figure(figsize=(6,4))
sns.barplot(data=df, x="Category", y="Value")
plt.title("Category Comparison Using Bar Plot")
plt.xlabel("Category")
```

```
plt.ylabel("Value")

plt.show()
```

Q4. Generate a dataset with categories and numerical values. Visualize the distribution of a numerical variable across different categories.

ANSWER;

```
# Seaborn Box Plot to Show Distribution Across Categories


import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


sns.set(style="whitegrid")


# Generate dataset

np.random.seed(42)


categories = ["A", "B", "C", "D"]

values = np.random.randn(200) * 10 + 50   # random values


df = pd.DataFrame({

   "Category": np.random.choice(categories, 200),

   "Value": values

})


# Box plot (distribution across categories)

plt.figure(figsize=(6,4))

sns.boxplot(data=df, x="Category", y="Value")
```

```
plt.title("Distribution of Values Across Categories")
plt.xlabel("Category")
plt.ylabel("Value")
plt.show()
```

Q5. Generate a synthetic dataset with correlated features. Visualize the correlation matrix of a dataset using a heatmap.

ANSWER:

```
# Seaborn Heatmap for Correlation Matrix (Synthetic Correlated Dataset)

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

sns.set(style="whitegrid")
# Generate synthetic correlated dataset
np.random.seed(42)
x1 = np.random.randn(200)
x2 = x1 * 0.8 + np.random.randn(200) * 0.2   # correlated with x1
x3 = x1 * -0.6 + np.random.randn(200) * 0.4  # negatively correlated with x1
x4 = np.random.randn(200)                # independent
df = pd.DataFrame({
   "Feature1": x1,
   "Feature2": x2,
   "Feature3": x3,
   "Feature4": x4
})
# Correlation matrix
```

```
corr = df.corr()

# Heatmap

plt.figure(figsize=(6,4))

sns.heatmap(corr, annot=True, cmap="coolwarm")

plt.title("Correlation Matrix Heatmap")

plt.show()
```

PLOTLY ASSIGNMENT:

Q1. 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)
```

ANSWER:

```
# 3D Scatter Plot using Plotly (given dataset)


import numpy as np

import pandas as pd

import plotly.express as px


# Given dataset

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)

# 3D Scatter Plot
fig = px.scatter_3d(
    df, x='X', y='Y', z='Z',
    title="3D Scatter Plot (Plotly)"
)

fig.show()
```

Q2. 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)

ANSWER:

```
# Q2: Violin Plot (Student Grades) + Heatmap (Sales Data) using PLOTLY

import numpy as np
import pandas as pd
import plotly.express as px

# --------------------------- Q2 (A) ---------------------------
# Violin Plot: Student Grades

np.random.seed(15)
```

```
data1 = {
   'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
   'Score': np.random.randint(50, 100, 200)
}
df1 = pd.DataFrame(data1)

fig1 = px.violin(df1, x="Grade", y="Score", box=True, points="all",
            title="Violin Plot: Score Distribution Across Grades")
fig1.show()



# --------------------------- Q2 (B) ---------------------------
# Heatmap: Sales Variation Across Months and Days

np.random.seed(20)
data2 = {
   '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)
}
df2 = pd.DataFrame(data2)

# Pivot table for heatmap
heatmap_data = df2.pivot_table(values="Sales", index="Day", columns="Month",
aggfunc="mean")

fig2 = px.imshow(heatmap_data,
            title="Heatmap: Sales Variation by Month and Day",
```

```
        labels=dict(x="Month", y="Day", color="Sales"))
fig2.show()
```

Q 3. 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)

ANSWER:

```
# Q3: Heatmap using Plotly (Sales variation across months and days)

import numpy as np
import pandas as pd
import plotly.express as px

# Given sales dataset
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)

# Create pivot table for heatmap
heatmap_data = df.pivot_table(values="Sales", index="Day", columns="Month", aggfunc="mean")

# Plot heatmap
```

```
fig = px.imshow(
   heatmap_data,
   title="Heatmap: Sales Variation Across Months and Days",
   labels=dict(x="Month", y="Day", color="Sales")
)
fig.show()
```

Q4.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(x**2 + y**2)) data = { 'X': x.flatten(), 'Y': y.flatten(), 'Z': z.flatten() } df = pd.DataFrame(data)

ANSWER:

```
# Q4: 3D Surface Plot using Plotly
# z = sin(sqrt(x^2 + y^2))

import numpy as np
import pandas as pd
import plotly.graph_objects as go

# Given data
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))

# Plotly 3D Surface Plot
fig = go.Figure(data=[go.Surface(x=X, y=Y, z=Z)])
```

```
fig.update_layout(
   title="3D Surface Plot: z = sin(sqrt(x^2 + y^2))",
   scene=dict(
      xaxis_title="X",
      yaxis_title="Y",
      zaxis_title="Z"
   )
)
fig.show()
```

Q 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)

ANSWER:

```
# Q5: Bubble Chart using Plotly
# GDP on X-axis, Population on Y-axis, Bubble size proportional to Population

import numpy as np
import pandas as pd
import plotly.express as px

# Given dataset
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)

# Bubble Chart
fig = px.scatter(
    df,
    x="GDP",
    y="Population",
    size="Population",
    color="Country",
    hover_name="Country",
    title="Bubble Chart: GDP vs Population (Bubble size = Population)",
    labels={"GDP": "GDP", "Population": "Population"}
)
fig.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.

ANSWER:

```
# Bokeh Plot: Sine Wave (x: 0 to 10, y = sin(x))

import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook   # use output_file if running in .py
```

```
output_notebook()


# Data
x = np.linspace(0, 10, 200)
y = np.sin(x)


# Plot
p = figure(title="Sine Wave using Bokeh", x_axis_label="X", y_axis_label="sin(X)",
        width=700, height=400)


p.line(x, y, line_width=2)
show(p)
```

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.

```
# Bokeh Scatter Plot with random x, y and different marker sizes + colors


import numpy as np
import pandas as pd
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource


output_notebook()


# Generate random dataset
np.random.seed(10)
data = {
```

```
    "x": np.random.rand(50) * 10,
    "y": np.random.rand(50) * 10,
    "sizes": np.random.randint(5, 25, 50),
    "colors": np.random.choice(["red", "blue", "green", "orange", "purple"], 50)
}
df = pd.DataFrame(data)

source = ColumnDataSource(df)

# Plot
p = figure(title="Bokeh Scatter Plot with Sizes and Colors",
        x_axis_label="X", y_axis_label="Y",
        width=700, height=400)

p.scatter(x="x", y="y", size="sizes", color="colors", alpha=0.7, source=source)

show(p)
```

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]

ANSWER:

```
# Bokeh Bar Chart: Fruit Counts

from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource
```

```
output_notebook()

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

source = ColumnDataSource(data=dict(fruits=fruits, counts=counts))

# Plot
p = figure(x_range=fruits, title="Fruit Counts Bar Chart",
        x_axis_label="Fruits", y_axis_label="Counts",
        width=700, height=400)

p.vbar(x='fruits', top='counts', width=0.6, source=source)
show(p)
```

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)
```

ANSWER:

```
# Bokeh Histogram: Distribution of Data

import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

output_notebook()
```

```
# Data
np.random.seed(42)
data_hist = np.random.randn(1000)

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

# Plot
p = figure(title="Bokeh Histogram",
        x_axis_label="Value", y_axis_label="Frequency",
        width=700, height=400)
p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
     fill_alpha=0.7, line_color="white")
show(p)
```

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)

ANSWER:

```
# Bokeh Heatmap using random 10x10 dataset

import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import LinearColorMapper, ColorBar
from bokeh.palettes import Viridis256

output_notebook()
```

```
# Given dataset
np.random.seed(42)
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)

# Convert heatmap data into x,y,value format
x_coords = xx.flatten()
y_coords = yy.flatten()
values = data_heatmap.flatten()

# Plot
p = figure(title="Bokeh Heatmap", width=700, height=400,
        x_axis_label="X", y_axis_label="Y")

# Color mapper
mapper = LinearColorMapper(palette=Viridis256, low=values.min(), high=values.max())

p.rect(x=x_coords, y=y_coords, width=0.1, height=0.1,
     fill_color={'field': 'values', 'transform': mapper},
     line_color=None,
     source=dict(x=x_coords, y=y_coords, values=values))

# Add color bar
color_bar = ColorBar(color_mapper=mapper, label_standoff=12, location=(0,0))
```

```
p.add_layout(color_bar, 'right')


show(p)
```