In [ ]:
```python
#Q1. 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]

import matplotlib.pyplot as plt

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

# Create scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(x, y, marker='o', color='blue', alpha=0.7)

# Set title and labels
plt.title('Relationship between x and y')
plt.xlabel('x')
plt.ylabel('y')

# Display gridlines
plt.grid(True)

# Display trend line (optional)
import numpy as np
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
plt.plot(x, p(x), color='red', linestyle='--')

# Show plot
plt.show()
```

In [ ]:
```python
#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])

import matplotlib.pyplot as plt
import numpy as np

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

# Create line plot
plt.figure(figsize=(8, 6))
plt.plot(data, marker='o', linestyle='-', color='blue')

# Set title and labels
plt.title('Trend of Values')
plt.xlabel('Index')
plt.ylabel('Value')

# Display gridlines
plt.grid(True)

# Set x-axis ticks
plt.xticks(np.arange(len(data)), [f'Index {i}' for i in range(len(data))])

# Show plot
plt.show()
```

In [ ]:
```python
#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]

import matplotlib.pyplot as plt

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

# Create bar chart
plt.figure(figsize=(8, 6))
plt.bar(categories, values, color='skyblue')

# Set title and labels
plt.title('Frequency of Categories')
plt.xlabel('Category')
plt.ylabel('Frequency')

# Display values on top of bars
for i, value in enumerate(values):
    plt.text(i, value + 2, str(value), ha='center')

# Display gridlines
plt.grid(axis='y')

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

In [ ]:
```python
#Q4. Create a histogram to visualize the distribution of values in the array data.
data = np.random.normal(0, 1, 1000)

import matplotlib.pyplot as plt
import numpy as np

# Generate random data with normal distribution
data = np.random.normal(0, 1, 1000)

# Create histogram
plt.figure(figsize=(8, 6))
plt.hist(data, bins=30, density=True, color='skyblue', edgecolor='black')

# Set title and labels
plt.title('Distribution of Values')
plt.xlabel('Value')
plt.ylabel('Density')

# Display gridlines
plt.grid(True)

# Add a normal distribution curve (optional)
import scipy.stats as stats
x = np.linspace(-3, 3, 100)
y = stats.norm.pdf(x, 0, 1)
plt.plot(x, y, color='red', linestyle='--')

# Show plot
plt.show()
```

In [ ]:
```python
#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]

import matplotlib.pyplot as plt

# Given data
sections = ['Section A', 'Section B', 'Section C', 'Section D']
sizes = [25, 30, 15, 30]

# Create pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=sections, autopct='%1.1f%%',
        startangle=90, textprops={'size': 'x-large'})

# Set title
plt.title('Percentage Distribution of Sections')

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')

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

In [2]:
```python
# seaborn assignment.
```

In [ ]:
```python
#Q1. Create a scatter plot to visualize the relationship between two variables, by generating a synthetic dataset.

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

# Generate synthetic dataset
np.random.seed(0)
x = np.random.normal(0, 10, 100)
y = 2 * x + np.random.normal(0, 5, 100)

# Create DataFrame
df = pd.DataFrame({'x': x, 'y': y})

# Create scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df['x'], df['y'], marker='o', color='blue', alpha=0.7)

# Set title and labels
plt.title('Relationship between x and y')
plt.xlabel('x')
plt.ylabel('y')

# Display gridlines
plt.grid(True)

# Display trend line (optional)
import numpy.polynomial.polynomial as poly
trend = poly.Polynomial(poly.polyfit(df['x'], df['y'], 1))
xp = np.linspace(-20, 20, 100)
plt.plot(xp, trend(xp), color='red', linestyle='--')

# Show plot
plt.show()
```

In [ ]:
```python
#Q2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm # Importing statsmodels for qqplot

# Generate dataset of random numbers
np.random.seed(0)
data = np.random.normal(0, 1, 1000)

# Create DataFrame
df = pd.DataFrame({'Values': data})

# Histogram
plt.figure(figsize=(8, 6))
sns.histplot(df['Values'], bins=30, kde=True)
plt.title('Distribution of Values')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()

# Boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(df['Values'])
plt.title('Boxplot of Values')
plt.show()

# Q-Q Plot
# Using statsmodels.api.qqplot instead of sns.displot
plt.figure(figsize=(8, 6))
sm.qqplot(df['Values'], line='s')  # 'line' argument specifies the reference line type
plt.title('Q-Q Plot of Values')
plt.show()

# Density Plot
plt.figure(figsize=(8, 6))
sns.kdeplot(df['Values'], shade=True)
plt.title('Density Plot of Values')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```

In [ ]:
```python
#Q3. Create a dataset representing categories and their corresponding values. Compare different categories based on numerical values.

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

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

# Create DataFrame
df = pd.DataFrame({'Category': categories, 'Value': values})

# Bar Chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Category', y='Value', data=df)
plt.title('Comparison of Categories')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()

# Pie Chart
plt.figure(figsize=(8, 6))
plt.pie(df['Value'], labels=df['Category'], autopct='%1.1f%%')
plt.title('Distribution of Values')
plt.axis('equal')
plt.show()

# Horizontal Bar Chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Value', y='Category', data=df)
plt.title('Categories by Value')
plt.xlabel('Value')
plt.ylabel('Category')
plt.show()
```

In [ ]:
```python
#Q4. Generate a dataset with categories and numerical values. Visualize the distribution of a numerical variable across different categories.

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

# Generate dataset
np.random.seed(0)
categories = ['A', 'B', 'C', 'D']
values_A = np.random.normal(10, 2, 100)
values_B = np.random.normal(15, 3, 100)
values_C = np.random.normal(8, 1.5, 100)
values_D = np.random.normal(12, 2.5, 100)

# Create DataFrame
df = pd.DataFrame({
    'Category': np.repeat(categories, 100),
    'Value': np.concatenate([values_A, values_B, values_C, values_D])
})

# Boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='Category', y='Value', data=df)
plt.title('Distribution of Values by Category')
plt.show()

# Violin Plot
plt.figure(figsize=(8, 6))
sns.violinplot(x='Category', y='Value', data=df)
plt.title('Distribution of Values by Category')
plt.show()

# Swarm Plot
plt.figure(figsize=(8, 6))
sns.swarmplot(x='Category', y='Value', data=df)
plt.title('Distribution of Values by Category')
plt.show()

# Bar Chart with Error Bars
plt.figure(figsize=(8, 6))
sns.barplot(x='Category', y='Value', data=df, ci='sd')
plt.title('Mean Values by Category')
plt.show()
```

In [ ]:
```python
#Q5. Generate a synthetic dataset with correlated features. Visualize the correlation matrix of a dataset using a heatmap.

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Generate synthetic dataset with correlated features
np.random.seed(0)
n_samples = 100
x1 = np.random.normal(0, 1, n_samples)
x2 = x1 + np.random.normal(0, 0.5, n_samples)
x3 = -x1 + np.random.normal(0, 0.5, n_samples)
x4 = np.random.normal(0, 1, n_samples)

df = pd.DataFrame({'x1': x1, 'x2': x2, 'x3': x3, 'x4': x4})

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

In [ ]:
```python
# PLOTLY ASSIGNMENT
```

In [ ]:
```python
#Q1. Using the given dataset, to generate a 3D scatter plot to visualize the distribution of data points in a three-dimensional 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)

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

# Set random seed for reproducibility
np.random.seed(30)

# Generate dataset
data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}

# Create DataFrame
df = pd.DataFrame(data)

# Create 3D scatter plot
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['X'], df['Y'], df['Z'], marker='o', color='blue', alpha=0.7)

# Set title and labels
ax.set_title('3D Distribution of Data Points')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Display gridlines
ax.grid(True)

# Show plot
plt.show()
```

In [ ]:
```python
#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# Œ™ Using the sales data, generate a

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


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

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)

plt.figure(figsize=(8, 6))
sns.violinplot(x='Grade', y='Score', data=df)
plt.title('Distribution of Scores by Grade')
plt.xlabel('Grade')
plt.ylabel('Score')
plt.show()
```

In [ ]:
```python
#Q3. 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

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

# Sales data as provided
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)

# Pivot the data for the heatmap
sales_matrix = df.pivot_table(values='Sales', index='Month', columns='Day', aggfunc='mean')

# Reorder months (if needed)
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
sales_matrix = sales_matrix.reindex(index=month_order)

# Create the heatmap
plt.figure(figsize=(12, 6))  # Adjust figure size if needed
sns.heatmap(sales_matrix, annot=True, fmt=".0f", cmap="viridis", linewidths=.5)
plt.title('Sales Variation across Months and Days')
plt.xlabel('Day of the Month')
plt.ylabel('Month')
plt.show()
```

In [ ]:
```python
#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(x2 + y2)) data = { 'X': x.flatten(), 'Y': y.flatten(), 'Z': z.flatten() } df = pd.DataFr


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 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))

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

# Create the 3D plot
fig = plt.figure(figsize=(10, 8))  # Adjust figure size if needed
ax = fig.add_subplot(111, projection='3d')

# Plot the surface
ax.plot_surface(x, y, z, cmap='viridis')  # You can change the colormap

# Set labels and title
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D Surface Plot')

plt.show()
```

In [ ]:
```python
# Q5. Using the given dataset, create a bubble chart to represent each country's population (y-axis), GDP (x-axis), 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.DataFra


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

# Given data
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)

# Create the bubble chart
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
plt.scatter(df['GDP'], df['Population'], s=df['Population'], alpha=0.7)

# Add labels for each bubble (country names)
for i, row in df.iterrows():
    plt.text(row['GDP'], row['Population'], row['Country'], ha='center', va='center')

# Set labels and title
plt.xlabel('GDP')
plt.ylabel('Population')
plt.title('Bubble Chart of Population vs. GDP')

plt.show()
```

In [ ]:
```python
# BOKEH ASSIGNMENT.
```

In [ ]:
```python
# Q1. Create a Bokeh plot displaying a sine wave. Set x-values from 0 to 10 and y-values as the sine of x.

#!pip install bokeh
import bokeh.io
import bokeh.plotting
bokeh.io.output_notebook()


!pip install bokeh
from bokeh.plotting import figure, show
import numpy as np

# Generate x and y values
x = np.linspace(0, 10, 100)  # 100 points between 0 and 10
y = np.sin(x)

# Create a Bokeh plot
p = figure(title="Sine Wave", x_axis_label="x", y_axis_label="sin(x)")

# Add a line glyph
p.line(x, y, line_width=2, line_color="navy")  # Customize line properties

# Show the plot
show(p)
```

In [ ]:
```python
# Q2. 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.

#!pip install bokeh
import bokeh.io
import bokeh.plotting
bokeh.io.output_notebook()


!pip install bokeh
from bokeh.plotting import figure, show
import numpy as np
import pandas as pd

# Generate random data
np.random.seed(42)
num_points = 50
data = {
    'x': np.random.rand(num_points) * 10,
    'y': np.random.rand(num_points) * 10,
    'sizes': np.random.randint(5, 20, num_points),
    'colors': np.random.choice(['red', 'blue', 'green'], num_points)
}
df = pd.DataFrame(data)

# Create a Bokeh plot
p = figure(title="Scatter Plot with Random Data", x_axis_label="x", y_axis_label="y")

# Add scatter glyphs with varying size and color
p.scatter(df['x'], df['y'], size=df['sizes'], color=df['colors'], alpha=0.7)

# Show the plot
show(p)
```

In [ ]:
```python
# Q3. 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]


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

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

# Create Bokeh plot
p = figure(title="Fruit Counts", x_axis_label='Fruit', y_axis_label='Count',
           x_range=fruits)

# Add bar renderer to plot
p.vbar(x=fruits, top=counts, width=0.5, legend_label="Fruit Counts",
       line_color='white', fill_color="#CAB2D6")

# Show plot
show(p)
```

In [ ]:
```python
# Q4. 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)


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

# Generate random data
np.random.seed(0)
data_hist = np.random.randn(1000)

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

# Create Bokeh plot
p = figure(title="Histogram", x_axis_label='Value', y_axis_label='Frequency',
           x_range=[edges[0], edges[-1]])

# Add histogram renderer to plot
p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
       fill_color="#CAB2D6", line_color="white")

# Show plot
show(p)
```

In [ ]:
```python
# Q5. 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)


!pip install bokeh
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, LinearColorMapper, ColorBar
import numpy as np

# Provided data
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)

# Reshape data for Bokeh
data = {
    'x': xx.flatten(),
    'y': yy.flatten(),
    'values': data_heatmap.flatten()
}
source = ColumnDataSource(data)

# Create color mapper
color_mapper = LinearColorMapper(palette="Viridis256", low=data_heatmap.min(), high=data_heatmap.max())

# Create Bokeh plot
p = figure(title="Heatmap", x_axis_label="x", y_axis_label="y",
           x_range=(x.min(), x.max()), y_range=(y.min(), y.max()),
           toolbar_location=None, tools="")  # Remove toolbar and tools

# Add heatmap rectangles
p.rect(x="x", y="y", width=0.1, height=0.1, source=source,
       fill_color={'field': 'values', 'transform': color_mapper},
       line_color=None)

# Add color bar
color_bar = ColorBar(color_mapper=color_mapper, label_standoff=12, border_line_color=None, location=(0, 0))
p.add_layout(color_bar, 'right')

# Show the plot
show(p)
```

In [ ]: