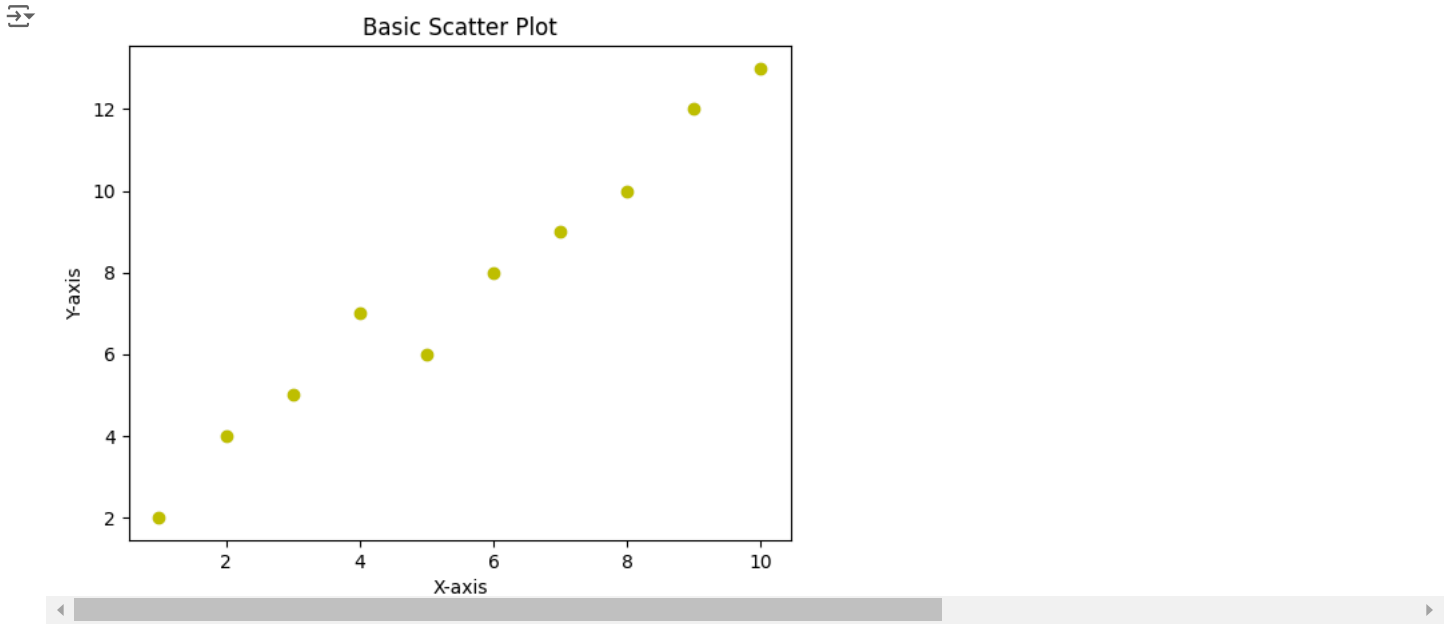
MATPLOTLIB ASSIGNMENT

ques 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]

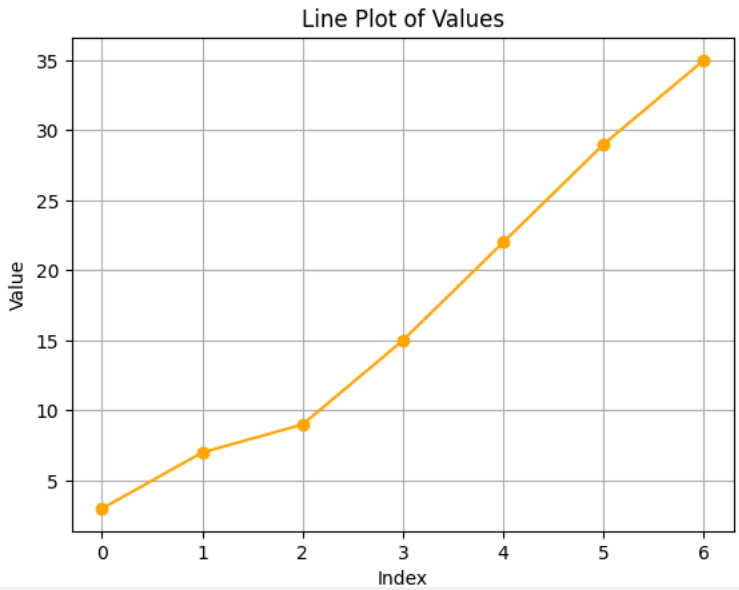
```
import matplotlib.pyplot as plt
x= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]
plt.scatter(x, y, color='y', marker='o')
# Add labels and title
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Basic Scatter Plot")
plt.show()
```



ques 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])

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

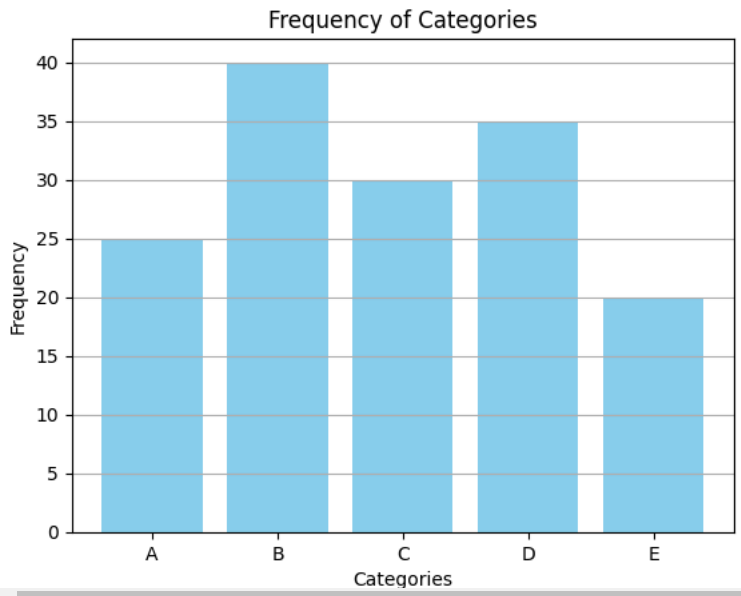


ques 3.. Display a bar chart to represent the frequency of each item in the given array categories.

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

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

```
import matplotlib.pyplot as plt
categories = ['A', 'B', 'C', 'D', 'E']
values = [25, 40, 30, 35, 20]
plt.bar(categories, values, color='skyblue')
plt.title('Frequency of Categories')
plt.xlabel('Categories')
plt.ylabel('Frequency')
plt.grid(axis='y') #or simply plt.grid(True)
plt.show()
```

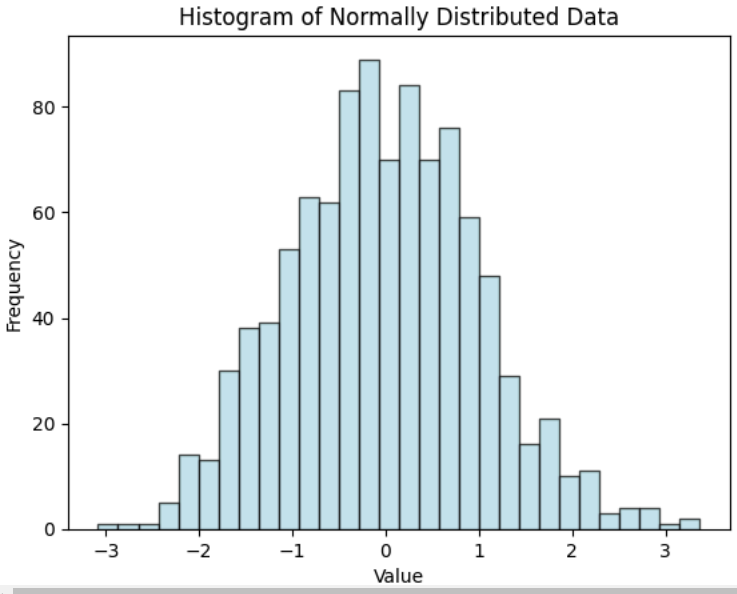


ques 4. Create a histogram to visualize the distribution of values in the array data.

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

```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.normal(0, 1, 1000)
```

```
plt.hist(data, bins=30, color='lightblue', edgecolor='black', alpha=0.7)
plt.title('Histogram of Normally Distributed Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

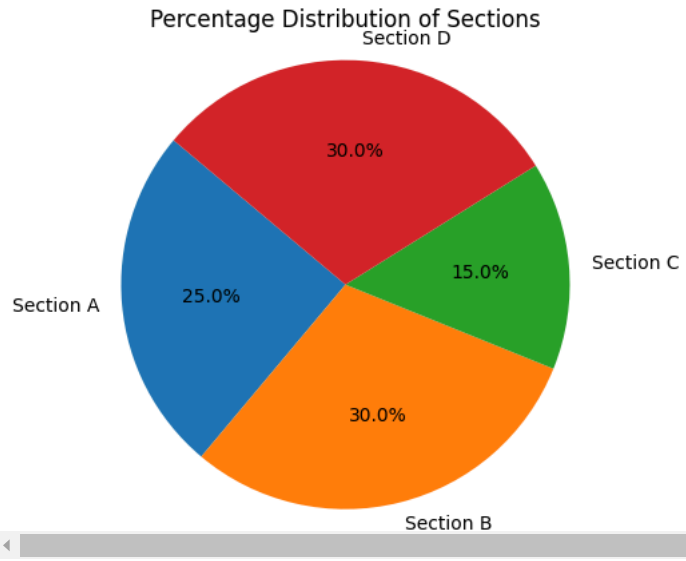


ques 5. Show a pie chart to represent the percentage distribution of different sections in the array `sections`.

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

sizes = [25, 30, 15, 30]

```
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=140)
plt.title('Percentage Distribution of Sections')
# Equal aspect ratio ensures that pie chart is a circle.
plt.axis('equal')
plt.show()
```

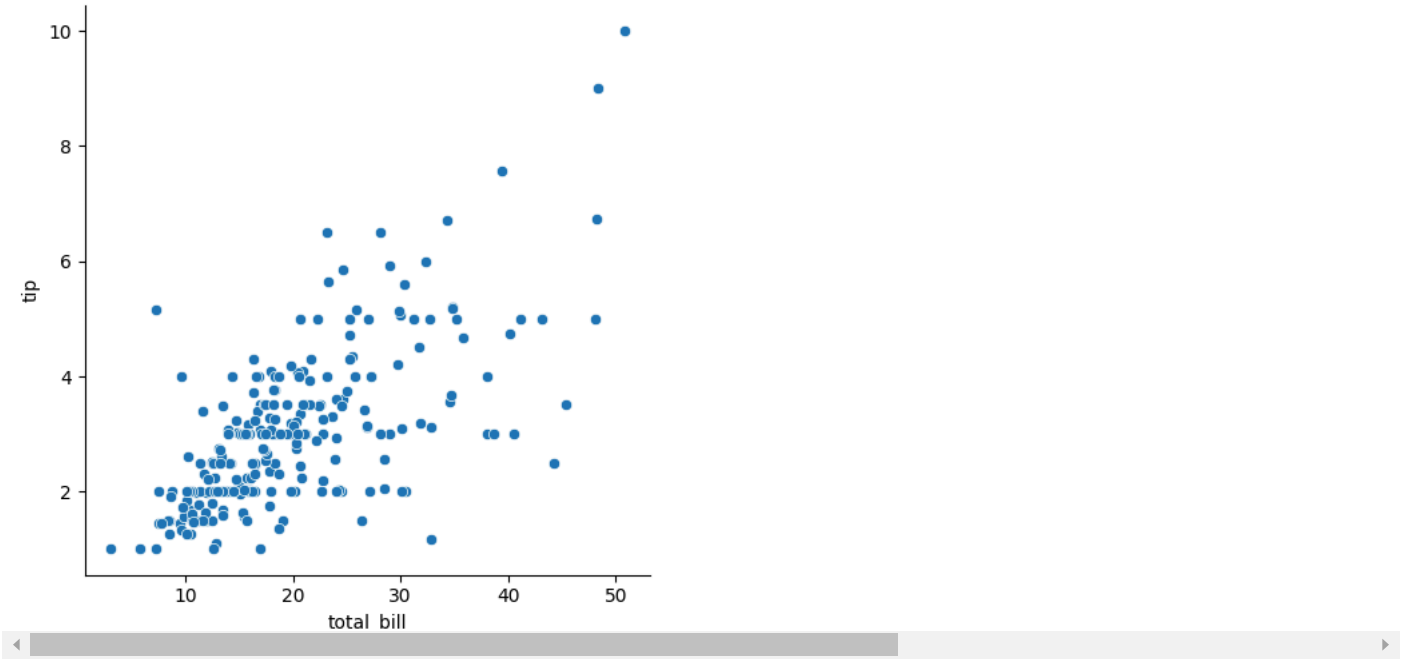


SEABORN ASSIGNMENT

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

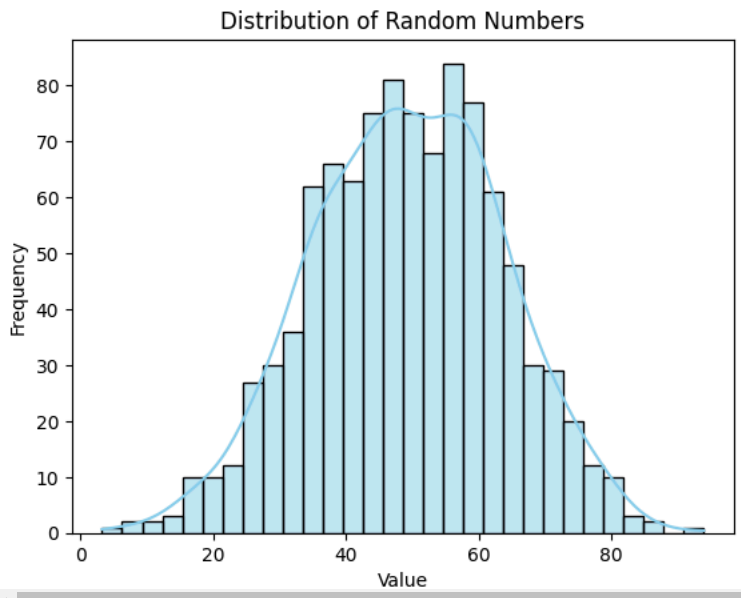
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
sns.get_dataset_names()
df = sns.load_dataset('tips')
sns.relplot(x = 'total_bill', y='tip', data = df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7c8311dcb880>
```



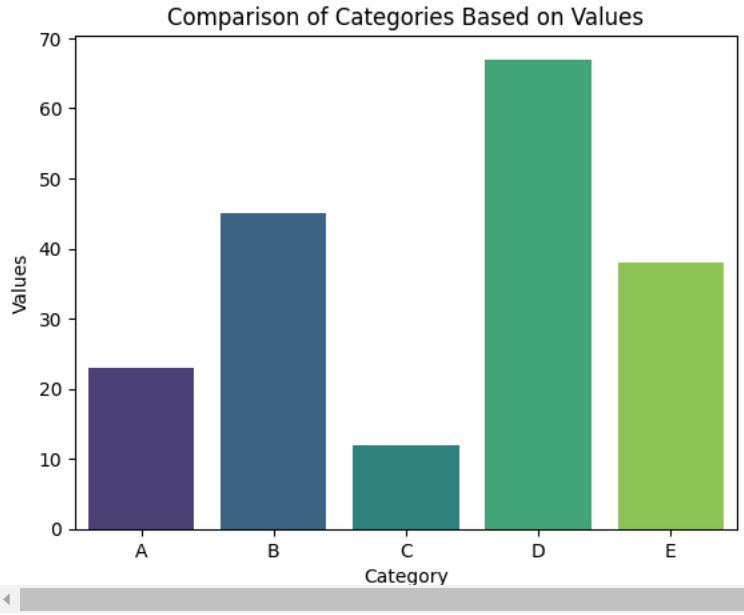
ques 2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
data = np.random.normal(loc=50, scale=15, size=1000)  # 1000 random numbers from a normal distribution
sns.histplot(data, bins=30, kde=True, color='skyblue', edgecolor='black')
plt.title('Distribution of Random Numbers')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



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

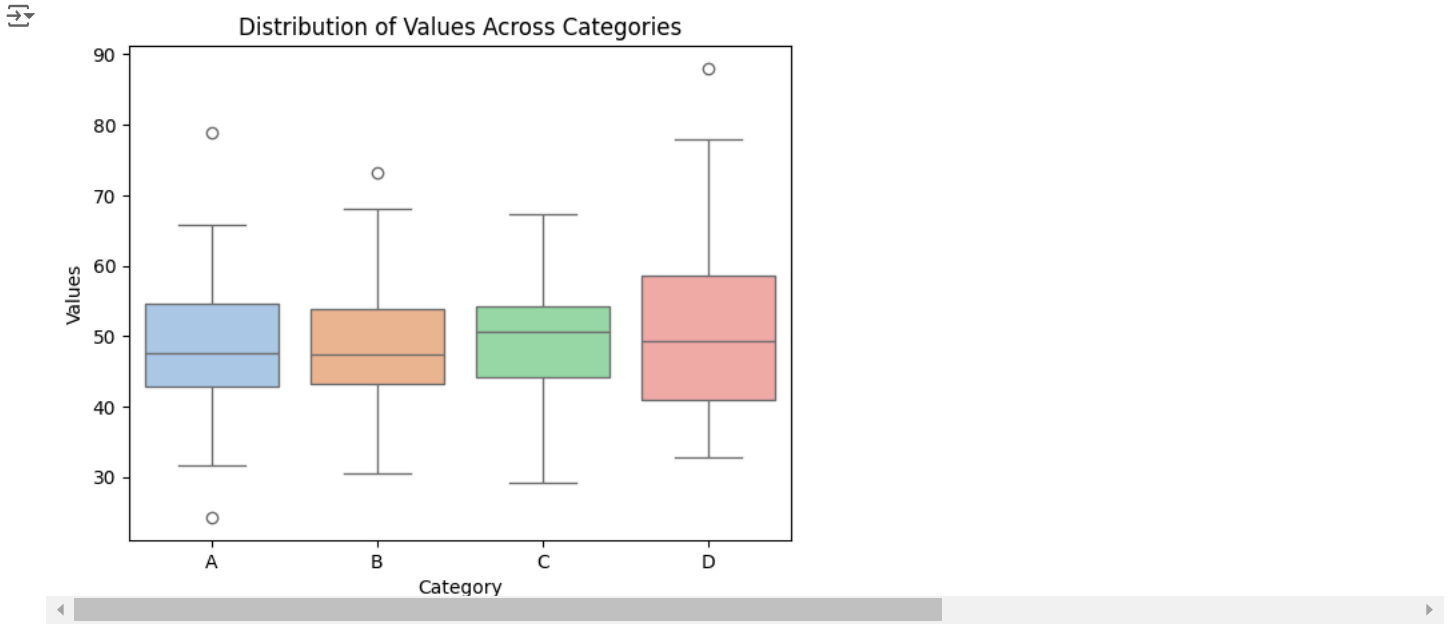
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = {
    'Category': ['A', 'B', 'C', 'D', 'E'],
    'Values': [23, 45, 12, 67, 38]
}
# Convert to DataFrame
df = pd.DataFrame(data)
sns.barplot(x='Category', y='Values', data=df, palette='viridis')
plt.title('Comparison of Categories Based on Values')
plt.xlabel('Category')
plt.ylabel('Values')
plt.show()
```



ques 4. Generate a dataset with categories and numerical values. Visualize the distribution of a numerical variable across different categories.

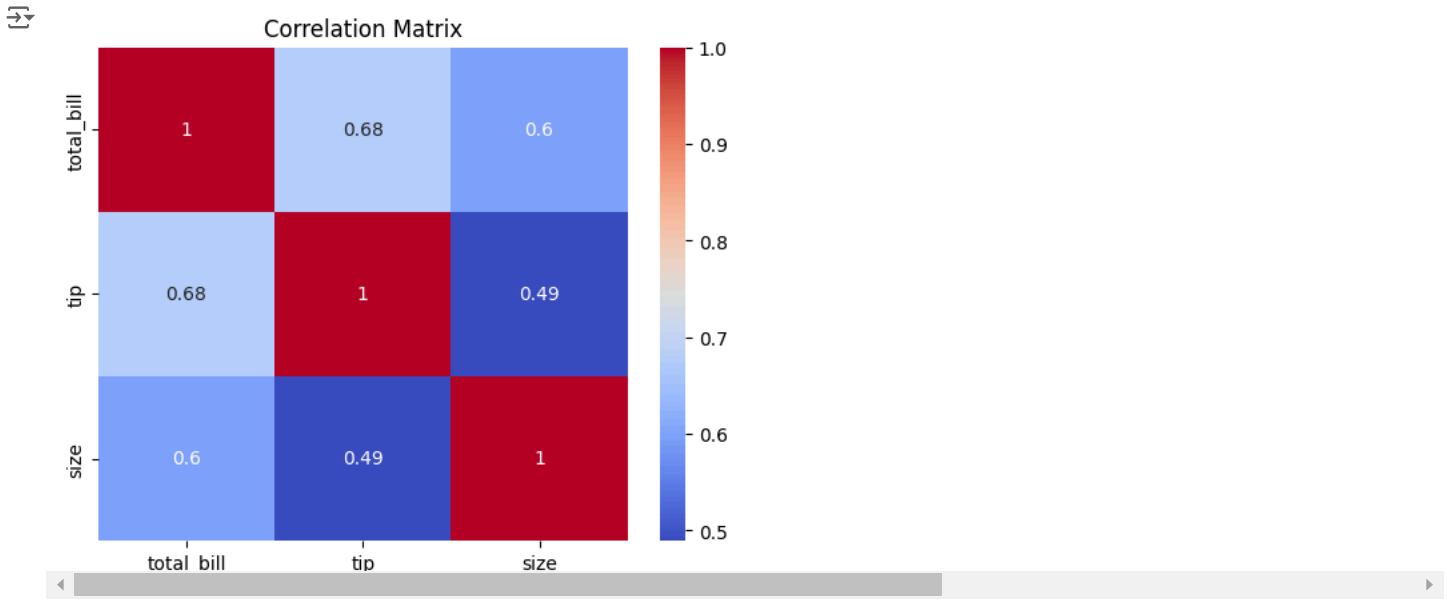
```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
categories = ['A', 'B', 'C', 'D']
data = {
    'Category': np.random.choice(categories, size=200),
    'Values': np.random.normal(loc=50, scale=10, size=200)
}
df = pd.DataFrame(data)
# Create a box plot to visualize the distribution of numerical values across categories
sns.boxplot(x='Category', y='Values', data=df, palette='pastel')
plt.title('Distribution of Values Across Categories')
plt.xlabel('Category')
plt.ylabel('Values')
plt.show()
```



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

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = sns.load_dataset("tips")
df
df1 = df[['total_bill', 'tip', 'size']]
df1.corr()
sns.heatmap(df1.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



PLOTLY ASSIGNMENT

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

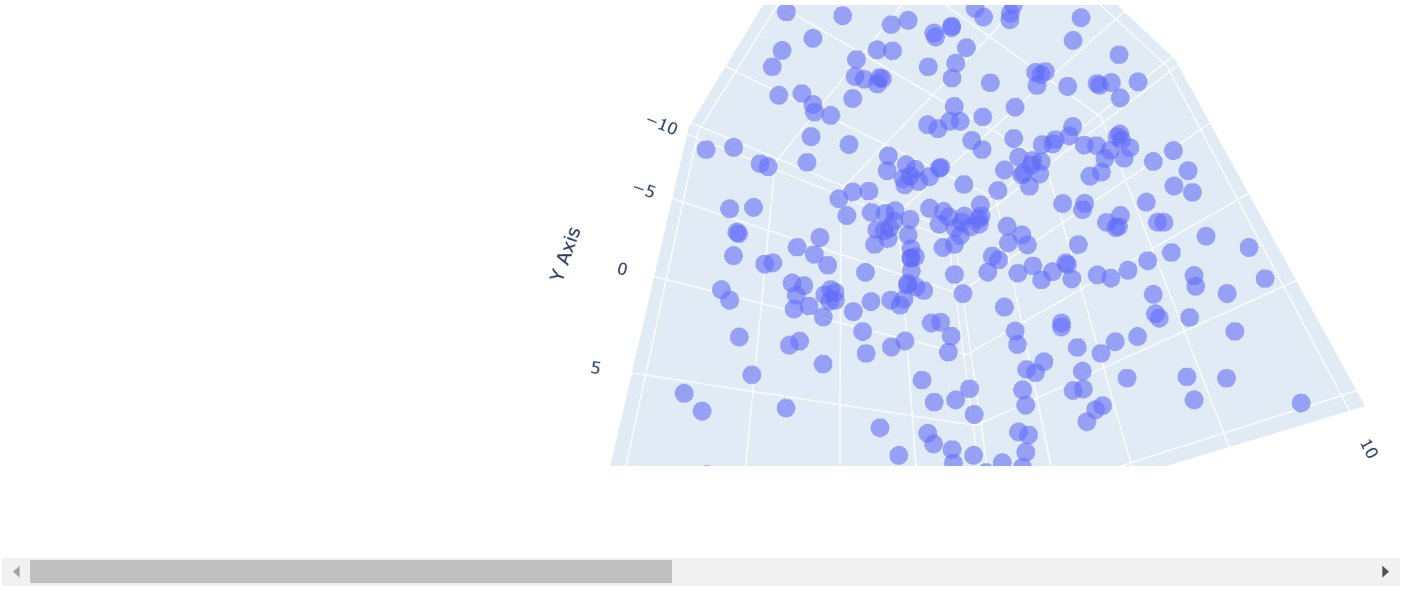
np.random.seed(30)

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)

```
import numpy as np
import pandas as pd
import plotly.express as px
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)
fig = px.scatter_3d(df, x='X', y='Y', z='Z', title='3D Scatter Plot of Random Data',
                    labels={'X': 'X Axis', 'Y': 'Y Axis', 'Z': 'Z Axis'},
                    opacity=0.6)
fig.show()
```



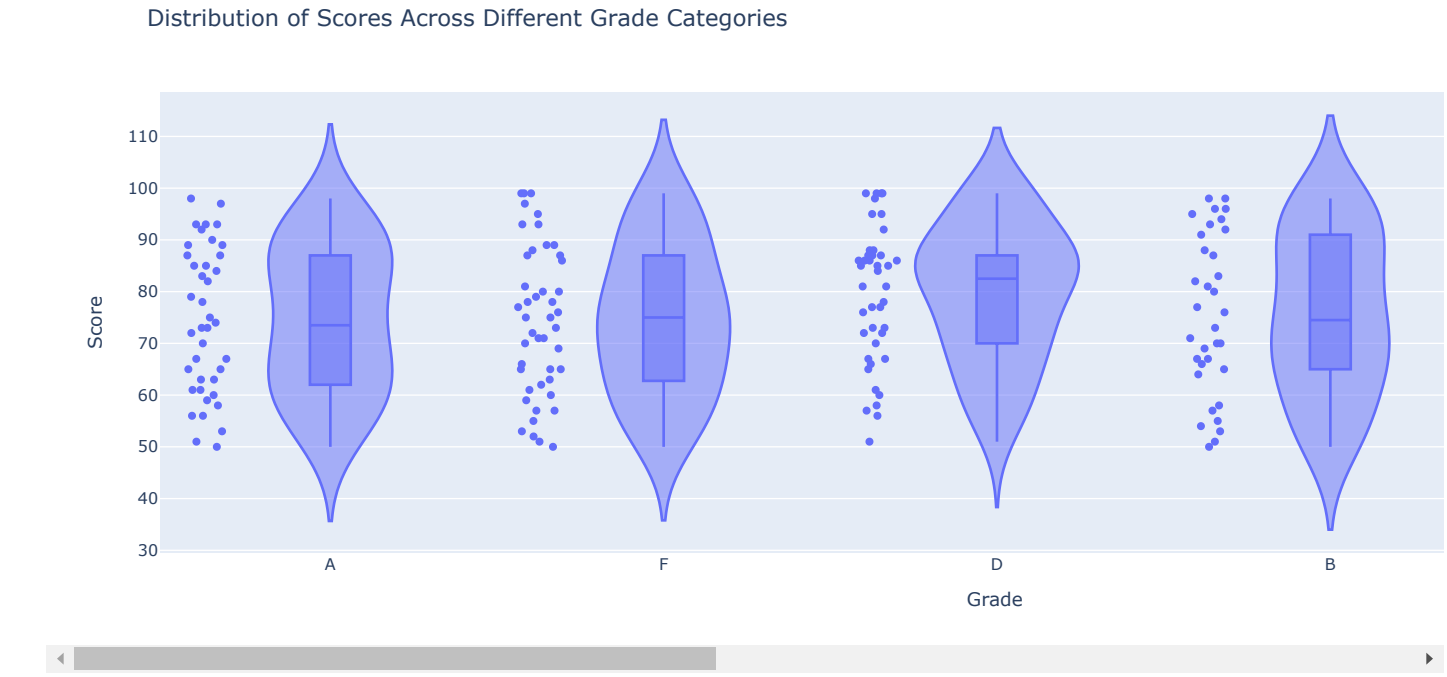
ques 2. Using the Student Grades, create a violin plot to display the distribution of scores across different grade categories.

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)

```
import numpy as np
import pandas as pd
import plotly.express as px
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)
fig = px.violin(df, x='Grade', y='Score', box=True, points='all',
                title='Distribution of Scores Across Different Grade Categories',
                labels={'Grade': 'Grade', 'Score': 'Score'})
fig.show()
```



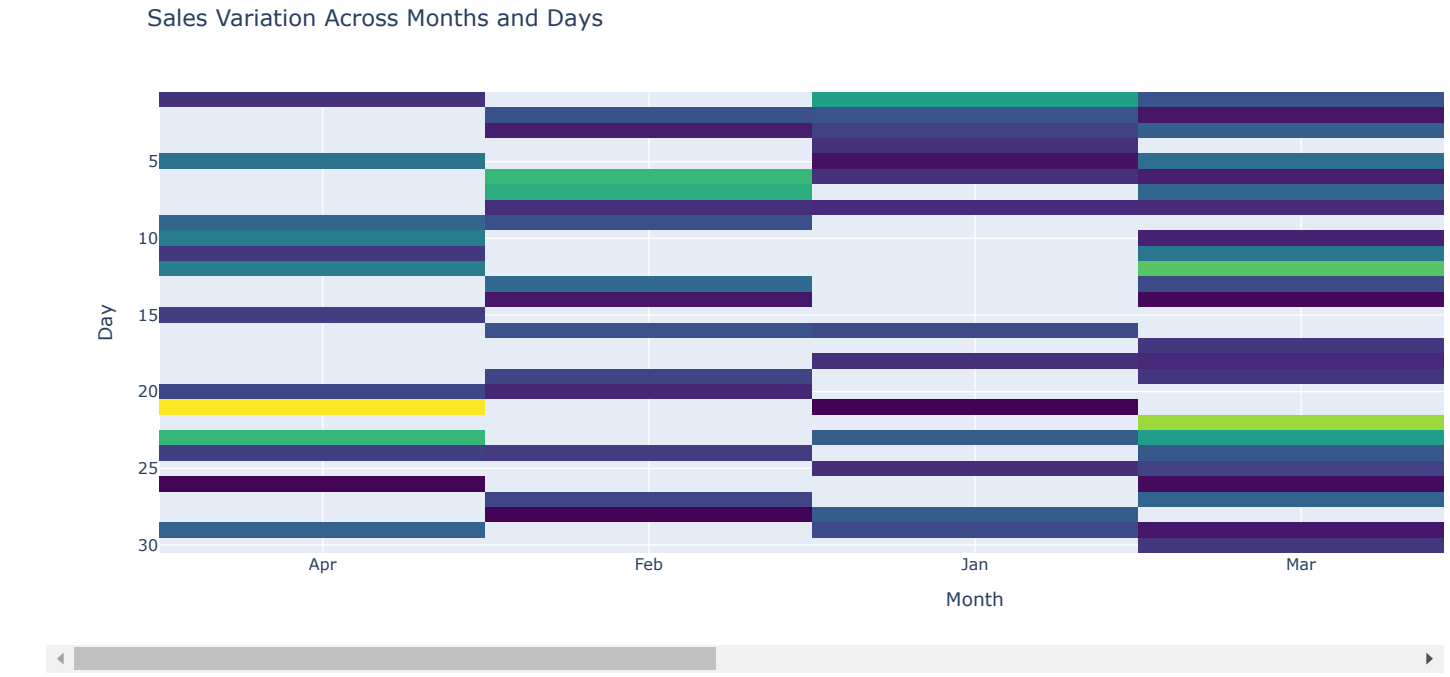
Using the sales data, generate a heatmap to visualize the variation in sales across different months and days.

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)

```
import numpy as np
import pandas as pd
import plotly.express as px
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_table = df.pivot_table(values='Sales', index='Day', columns='Month', aggfunc='sum')
fig = px.imshow(pivot_table,
                color_continuous_scale='Viridis',
                labels={'x': 'Month', 'y': 'Day', 'color': 'Sales'},
                title='Sales Variation Across Months and Days')
fig.show()
```



ques 4.4. Using the given x and y data, generate a 3D surface plot to visualize the function z =sin(root of x sq + y sq)

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

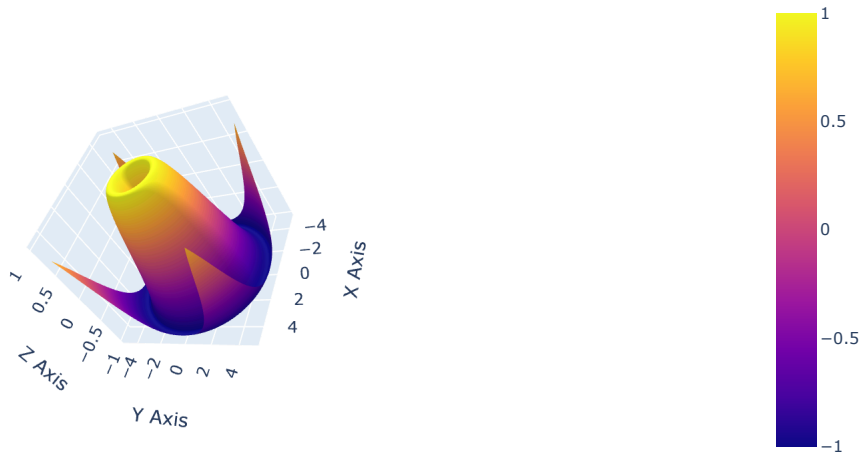
y = np.linspace(-5, 5, 100) x, y = np.meshgrid(x, y)

z = np.sin(np.sqrt(x**2 + y**2))

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

```
  df = pd.DataFrame(data)
```

```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)

# Calculate z values
z = np.sin(np.sqrt(x**2 + y**2))
# Create a DataFrame
data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)
fig = go.Figure(data=[go.Surface(z=df['Z'].values.reshape(100, 100),
                                 x=df['X'].values.reshape(100, 100),
                                 y=df['Y'].values.reshape(100, 100))])

# Update layout
fig.update_layout(title='3D Surface Plot of z = sin(sqrt(x^2 + y^2))',
                  scene=dict(
                      xaxis_title='X Axis',
                      yaxis_title='Y Axis',
                      zaxis_title='Z Axis'
                  ))
fig.show()
```



ques 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'],

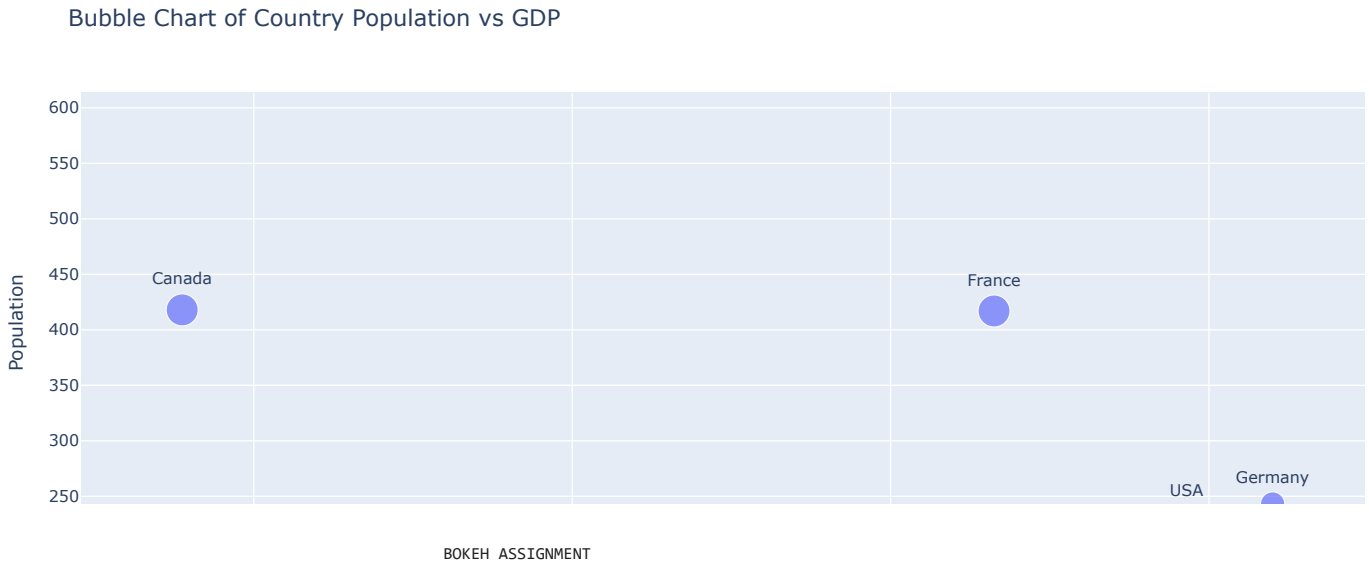
'Population':

np.random.randint(100, 1000, 5),

'GDP': np.random.randint(500, 2000, 5) }

df = pd.DataFrame(data)

```
import numpy as np
import pandas as pd
import plotly.express as px
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 a bubble chart using Plotly
fig = px.scatter(df,
                 x='GDP',
                 y='Population',
                 size='Population',
                 text='Country',
                 title='Bubble Chart of Country Population vs GDP',
                 labels={'GDP': 'GDP (in billions)', 'Population': 'Population (in millions)'})
# Update layout to enhance visualization
fig.update_traces(textposition='top center')
fig.update_layout(xaxis_title='GDP (in billions)',
                  yaxis_title='Population',
                  showlegend=False)
fig.show()
```



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

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

# Prepare the output to be displayed in a Jupyter notebook (or use output_file for HTML)
output_notebook()

# Generate data
x = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y = np.sin(x)  # Sine of x
```