Visual Representation of the Data

- ✓ **Visual representation helps in**
- ✓ **libraries for data visualization:**
    - **Matplotlib: Basic visualization.**
    - **Seaborn: Statistical plotting.**
    - **Pandas: Quick plots from dataframes.**
    - **Plotly: Interactive visualizations.**

- ✓ **Setting Up Libraries**
- ✓ **Visualizations in Python**
    - **Line Plot:**
    - **Bar Plot**
    - **Histogram**
    - **Box Plot**
    - **Scatter Plot**
    - **Pair Plot**
    - **Heatmap**
    - **Combining Multiple Plots**
    - **Choosing the Right Visualization**
    - **Real-World Example: Visualizing a Dataset**

Visual representation helps in:

- Understanding patterns, trends, and insights in data.
- Communicating findings effectively.
- Identifying outliers, anomalies, and relationships in the data.

Python provides robust libraries for data visualization, including:

- **Matplotlib**: Basic visualization.
- **Seaborn**: Statistical plotting.
- **Pandas**: Quick plots from dataframes.
- **Plotly**: Interactive visualizations.

Setting Up Libraries

First, ensure the required libraries are installed. If not, install them:

```
pip install matplotlib seaborn pandas
```

Import the libraries in Python:

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

Visualizations in Python

Line Plot:

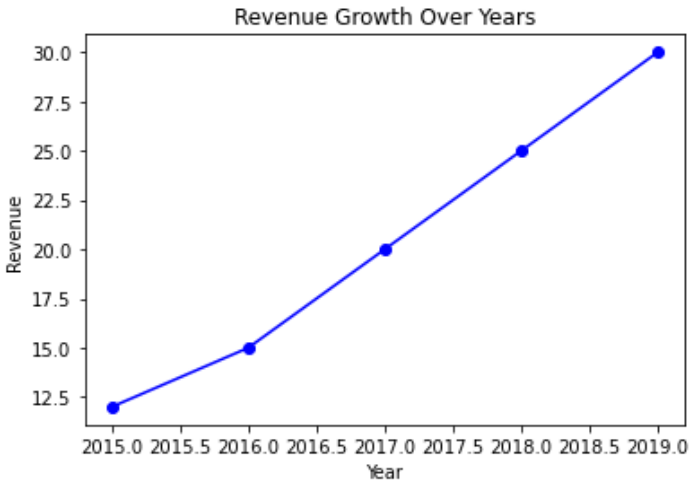
Line plots show trends over time or continuous data.

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

# Create sample data
data = {'Year': [2015, 2016, 2017, 2018, 2019],
        'Revenue': [12, 15, 20, 25, 30]}
df = pd.DataFrame(data)

# Line plot using Matplotlib
plt.plot(df['Year'], df['Revenue'], marker='o', linestyle='-', color='b')
plt.title('Revenue Growth Over Years')
plt.xlabel('Year')
plt.ylabel('Revenue')
plt.show()
```

Output: A line chart displaying revenue growth across years.

In [6]:

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

# Create sample data
data = {'Year': [2015, 2016, 2017, 2018, 2019],
        'Revenue': [12, 15, 20, 25, 30]}
df = pd.DataFrame(data)

# Line plot using Matplotlib
plt.plot(df['Year'], df['Revenue'], marker='o', linestyle='-', color='b')
plt.title('Revenue Growth Over Years')
plt.xlabel('Year')
plt.ylabel('Revenue')
plt.show()
```



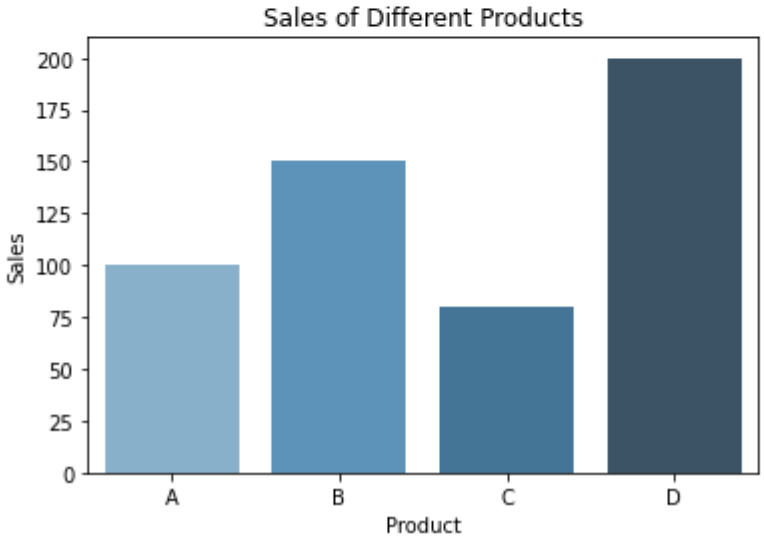
Bar Plot

Bar plots are used to compare values across categories.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Create sample data
data = {'Product': ['A', 'B', 'C', 'D'],
       'Sales': [100, 150, 80, 200]}
df = pd.DataFrame(data)

# Bar plot using Seaborn
sns.barplot(x='Product', y='Sales', data=df, palette='Blues_d')
plt.title('Sales of Different Products')
plt.show()
```

Output: A bar chart comparing sales for different products.

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Create sample data
data = {'Product': ['A', 'B', 'C', 'D'],
        'Sales': [100, 150, 80, 200]}
df = pd.DataFrame(data)

# Bar plot using Seaborn
sns.barplot(x='Product', y='Sales', data=df, palette='Blues_d')
plt.title('Sales of Different Products')
plt.show()
```



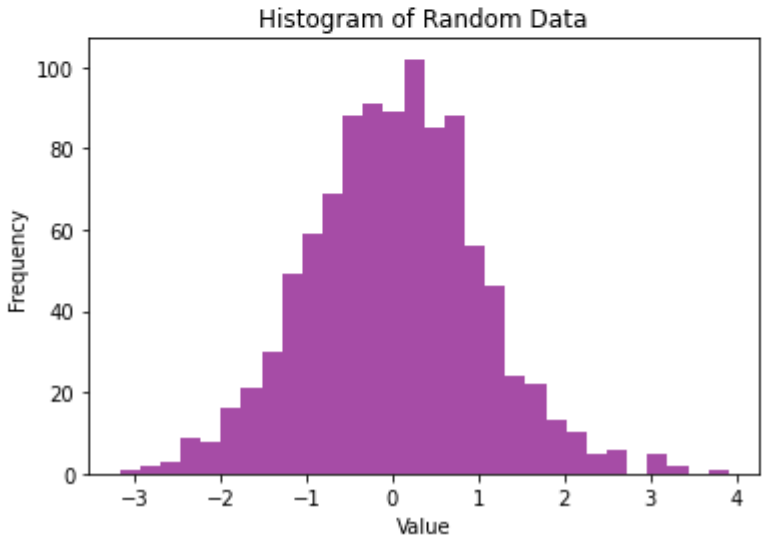
Histogram

Histograms display the frequency distribution of a dataset.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
import numpy as np
data = np.random.randn(1000)

# Histogram using Matplotlib
plt.hist(data, bins=30, color='purple', alpha=0.7)
plt.title('Histogram of Random Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

Output: A histogram showing how data is distributed.

In [9]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
import numpy as np
data = np.random.randn(1000)

# Histogram using Matplotlib
plt.hist(data, bins=30, color='purple', alpha=0.7)
plt.title('Histogram of Random Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



Box Plot

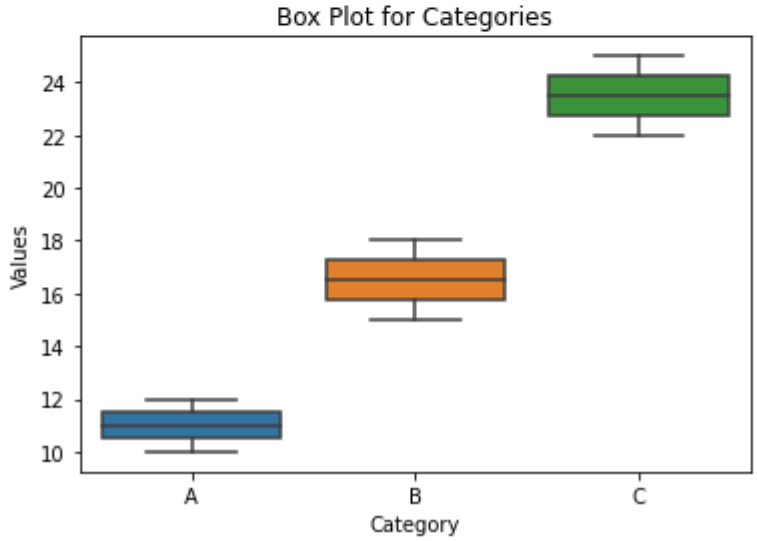
Box plots summarize data by showing quartiles, median, and outliers.

Example:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
data = {'Category': ['A', 'A', 'B', 'B', 'C', 'C'],
        'Values': [10, 12, 15, 18, 22, 25]}
df = pd.DataFrame(data)

# Box plot using Seaborn
sns.boxplot(x='Category', y='Values', data=df)
plt.title('Box Plot for Categories')
plt.show()
```

Output: A box plot that highlights the spread and outliers of each category.

In [12]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
data = {'Category': ['A', 'A', 'B', 'B', 'C', 'C'],
        'Values': [10, 12, 15, 18, 22, 25]}
df = pd.DataFrame(data)

# Box plot using Seaborn
sns.boxplot(x='Category', y='Values', data=df)
plt.title('Box Plot for Categories')
plt.show()
```



Scatter Plot

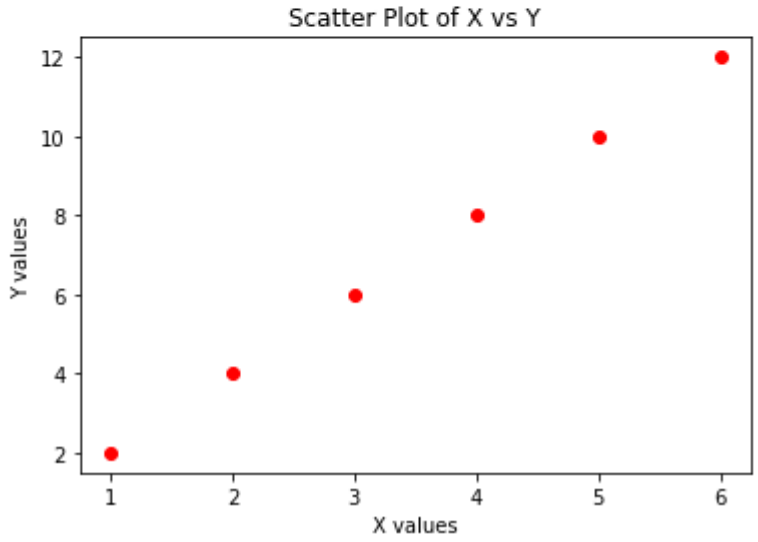
Scatter plots show relationships between two numerical variables.

Example:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
x = [1, 2, 3, 4, 5, 6]
y = [2, 4, 6, 8, 10, 12]

# Scatter plot
plt.scatter(x, y, color='red')
plt.title('Scatter Plot of X vs Y')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```

Output: A scatter plot showing a linear relationship between `X` and `Y`.

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample data
x = [1, 2, 3, 4, 5, 6]
y = [2, 4, 6, 8, 10, 12]

# Scatter plot
plt.scatter(x, y, color='red')
plt.title('Scatter Plot of X vs Y')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



Pair Plot

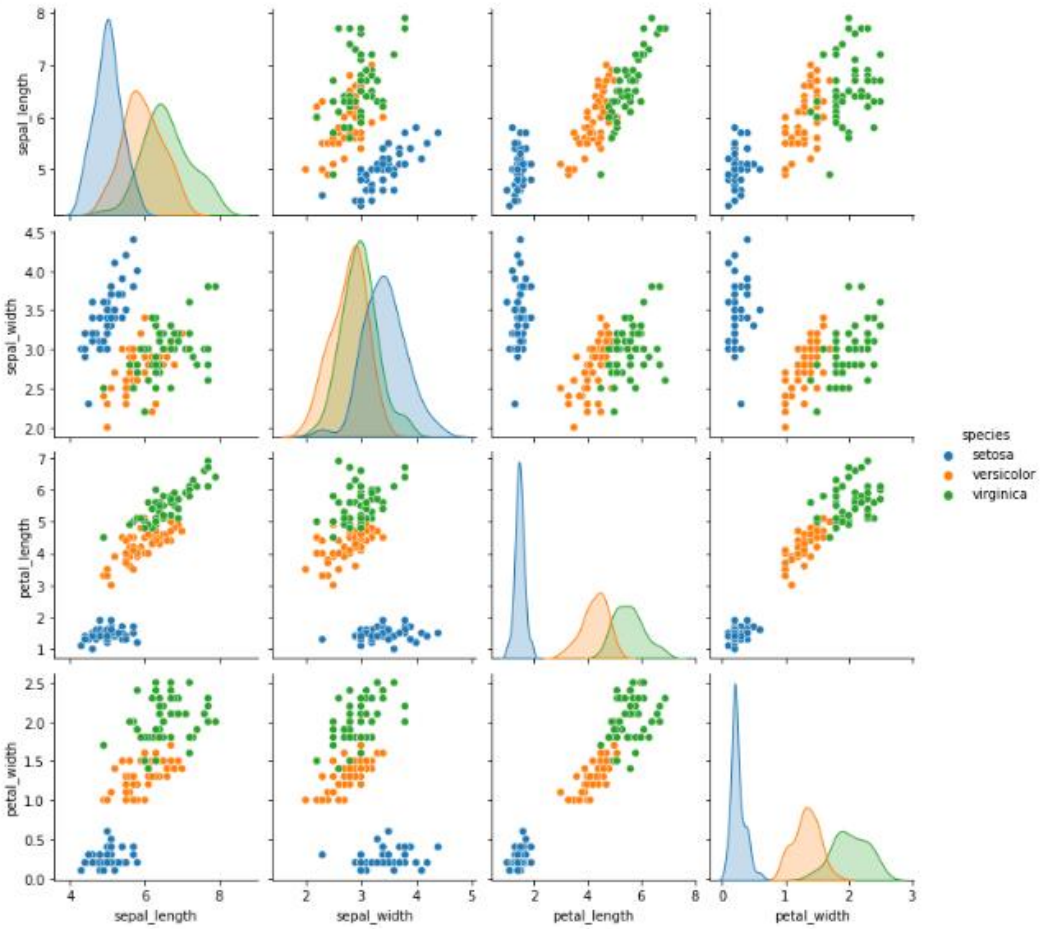
Pair plots visualize pairwise relationships between variables in a dataset.

Example:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample dataset
iris = sns.load_dataset('iris')

# Pair plot using Seaborn
sns.pairplot(iris, hue='species')
plt.show()
```

Output: Pair plots for the **Iris** dataset with relationships categorized by species.

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Sample dataset
iris = sns.load_dataset('iris')

# Pair plot using Seaborn
sns.pairplot(iris, hue='species')
plt.show()
```



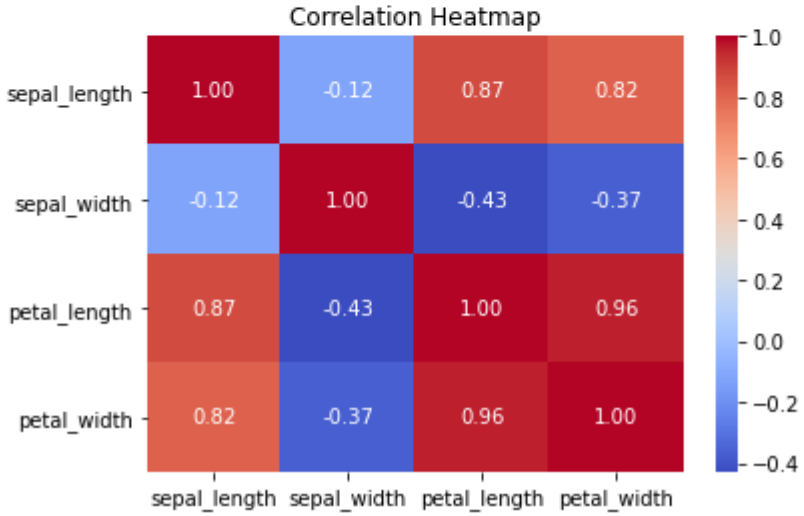
Heatmap: Heatmaps show correlations or relationships in a matrix form.

Example:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Correlation matrix
data = sns.load_dataset('iris')
correlation = data.corr()

# Heatmap using Seaborn
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

Output: A heatmap showing the correlation between numerical variables.

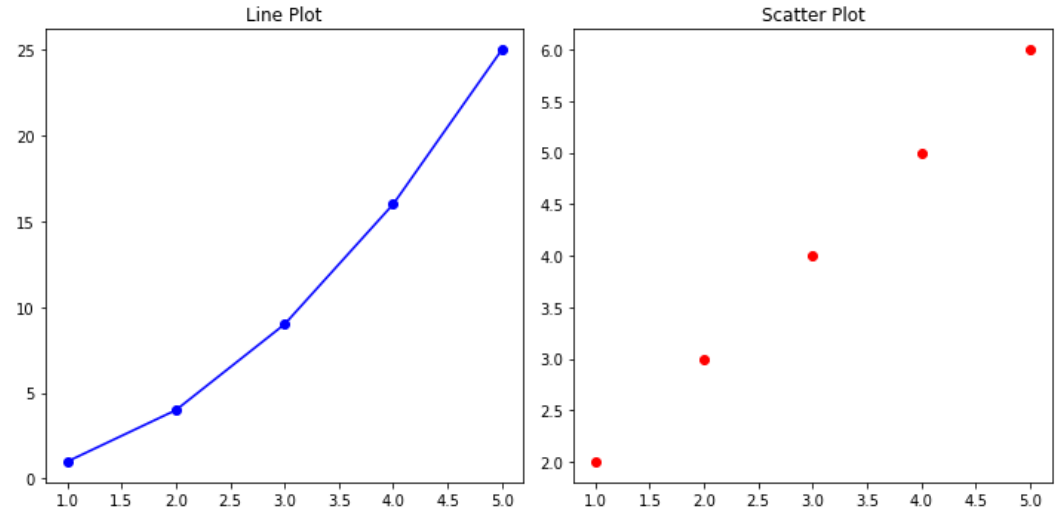
In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Correlation matrix
data = sns.load_dataset('iris')
correlation = data.corr()

# Heatmap using Seaborn
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



4. Combining Multiple Plots: You can combine multiple plots in a grid using subplot.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Create sample data
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 9, 16, 25]
y2 = [2, 3, 4, 5, 6]
# Subplots
plt.figure(figsize=(10, 5))
# Plot 1: Line plot
plt.subplot(1, 2, 1)
plt.plot(x, y1, marker='o', color='b')
plt.title('Line Plot')
# Plot 2: Scatter plot
plt.subplot(1, 2, 2)
plt.scatter(x, y2, color='r')
plt.title('Scatter Plot')
plt.tight_layout()
plt.show()
```

In [21]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Create sample data
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 9, 16, 25]
y2 = [2, 3, 4, 5, 6]

# Subplots
plt.figure(figsize=(10, 5))

# Plot 1: Line plot
plt.subplot(1, 2, 1)
plt.plot(x, y1, marker='o', color='b')
plt.title('Line Plot')

# Plot 2: Scatter plot
plt.subplot(1, 2, 2)
plt.scatter(x, y2, color='r')
plt.title('Scatter Plot')

plt.tight_layout()
plt.show()
```



5. Choosing the Right Visualization

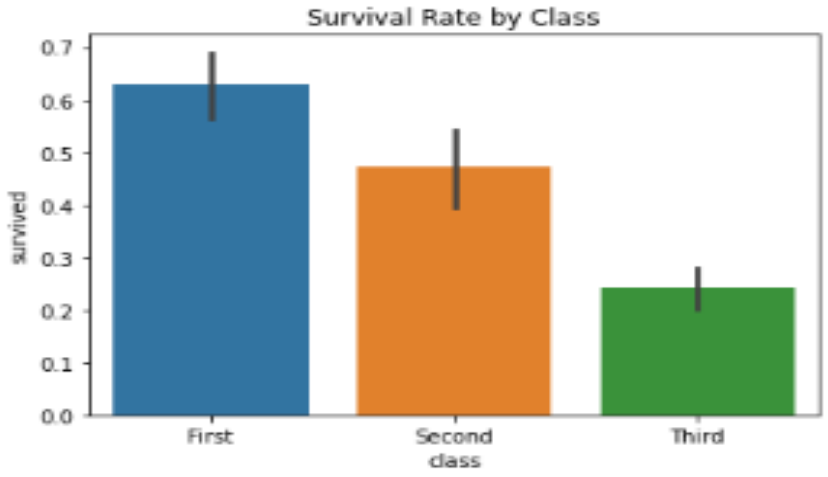
Visualization	Use Case
Line Plot	Trends over time or continuous data
Bar Plot	Comparison of categories
Histogram	Frequency distribution
Box Plot	Spread and outliers of numerical data
Scatter Plot	Relationship between two numerical variables
Heatmap	Correlation matrix visualization

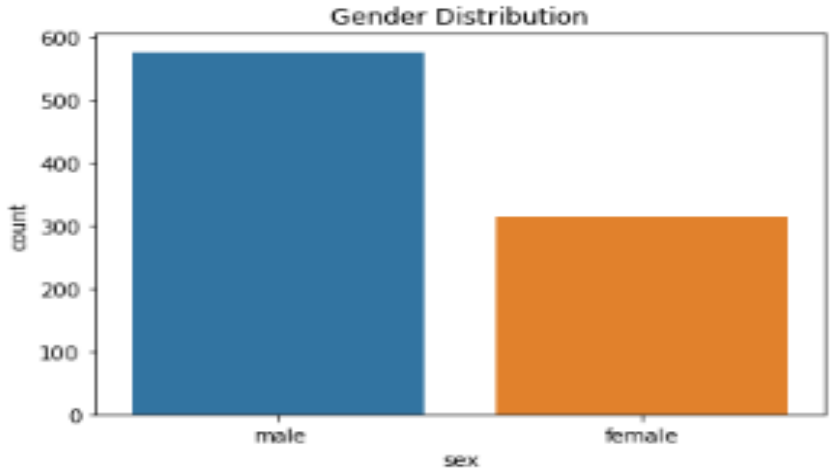
Real-World Example: Visualizing a Dataset

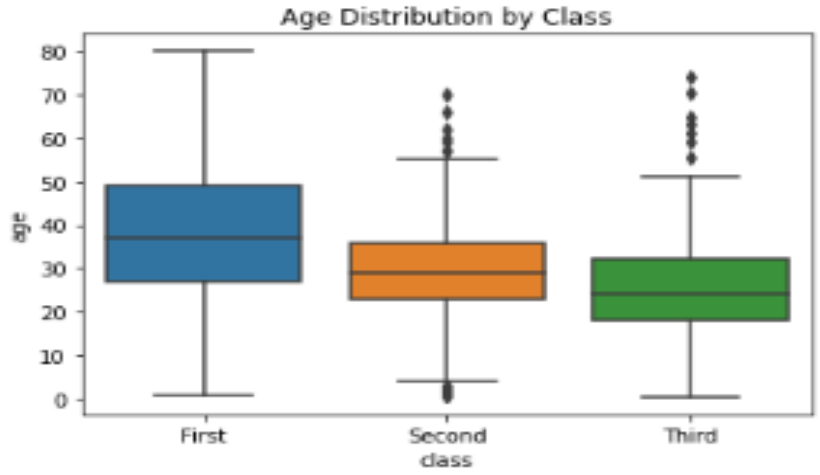
Let's visualize the Titanic dataset using multiple plots.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Load dataset
titanic = sns.load_dataset('titanic')

# Bar plot: Survival rate by class
sns.barplot(x='class', y='survived', data=titanic)
plt.title('Survival Rate by Class')
plt.show()

# Count plot: Gender distribution
sns.countplot(x='sex', data=titanic)
plt.title('Gender Distribution')
plt.show()

# Box plot: Age distribution by class
sns.boxplot(x='class', y='age', data=titanic)
plt.title('Age Distribution by Class')
plt.show()
```

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Load dataset
titanic = sns.load_dataset('titanic')

# Bar plot: Survival rate by class
sns.barplot(x='class', y='survived', data=titanic)
plt.title('Survival Rate by Class')
plt.show()

# Count plot: Gender distribution
sns.countplot(x='sex', data=titanic)
plt.title('Gender Distribution')
plt.show()

# Box plot: Age distribution by class
sns.boxplot(x='class', y='age', data=titanic)
plt.title('Age Distribution by Class')
plt.show()
```


Survival Rate by Class
survived
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
First
Second
Third
class


Gender Distribution
count
0
100
200
300
400
500
600
male
female
sex


Age Distribution by Class
age
0
10
20
30
40
50
60
70
80
First
Second
Third
class

Measures of Central Tendency & Dispersion

- **Central Tendency includes:**
    - Mean: The average of the data.
    - Median: The middle value when data is sorted.
    - Mode: The most frequent value
- **Measures of Dispersion**
    - Dispersion includes:
    - Range: Difference between maximum and minimum values.
    - Variance: The average of squared deviations from the mean.
    - Standard Deviation: Square root of the variance.
    - IQR (Interquartile Range): Spread of the middle 50% of the data.
    - Python Code for Dispersion
- **Visualizing Central Tendency and Dispersion**
- **Summary Table**

Measures of Central Tendency

Central Tendency includes:

- **Mean**: The average of the data.
- **Median**: The middle value when data is sorted.
- **Mode**: The most frequent value.

Python Code for Central Tendency

Here's an example using Python with the pandas and statistics libraries.

```
import pandas as pd
from statistics import mean, median, mode
from scipy import stats

# Sample data
data = [10, 20, 20, 40, 50, 50, 50, 60, 70, 80]

# Converting data to a pandas DataFrame
df = pd.DataFrame(data, columns=['Values'])

# Mean
mean_value = df['Values'].mean()
print("Mean:", mean_value)

# Median
median_value = df['Values'].median()
print("Median:", median_value)

# Mode
mode_value = df['Values'].mode()[0]  # Pandas returns a Series for mode
print("Mode:", mode_value)
```

Output:

```
Mean: 45.0
Median: 50.0
Mode: 50
```

In [24]:
```python
import pandas as pd
from statistics import mean, median, mode
from scipy import stats

# Sample data
data = [10, 20, 20, 40, 50, 50, 50, 60, 70, 80]

# Converting data to a pandas DataFrame
df = pd.DataFrame(data, columns=['Values'])

# Mean
mean_value = df['Values'].mean()
print("Mean:", mean_value)

# Median
median_value = df['Values'].median()
print("Median:", median_value)

# Mode
mode_value = df['Values'].mode()[0]  # Pandas returns a Series for mode
print("Mode:", mode_value)
```

```
Mean: 45.0
Median: 50.0
Mode: 50
```

Measures of Dispersion

Dispersion includes:

Range: Difference between maximum and minimum values.

Variance: The average of squared deviations from the mean.

Standard Deviation: Square root of the variance.

IQR (Interquartile Range): Spread of the middle 50% of the data.

Python Code for Dispersion

```
import numpy as np

# Range
range_value = df['Values'].max() - df['Values'].min()
print("Range:", range_value)

# Variance
variance_value = df['Values'].var()  # Sample variance
print("Variance:", variance_value)

# Standard Deviation
std_dev_value = df['Values'].std()  # Sample standard deviation
print("Standard Deviation:", std_dev_value)

# IQR (Interquartile Range)
Q1 = df['Values'].quantile(0.25)
Q3 = df['Values'].quantile(0.75)
IQR = Q3 - Q1
```

```
print("Interquartile Range (IQR):", IQR)
```

In [25]:
```python
import numpy as np

# Range
range_value = df['Values'].max() - df['Values'].min()
print("Range:", range_value)

# Variance
variance_value = df['Values'].var()  # Sample variance
print("Variance:", variance_value)

# Standard Deviation
std_dev_value = df['Values'].std()  # Sample standard deviation
print("Standard Deviation:", std_dev_value)

# IQR (Interquartile Range)
Q1 = df['Values'].quantile(0.25)
Q3 = df['Values'].quantile(0.75)
IQR = Q3 - Q1
print("Interquartile Range (IQR):", IQR)
```

```
Range: 70
Variance: 516.6666666666666
Standard Deviation: 22.73030282830976
Interquartile Range (IQR): 32.5
```

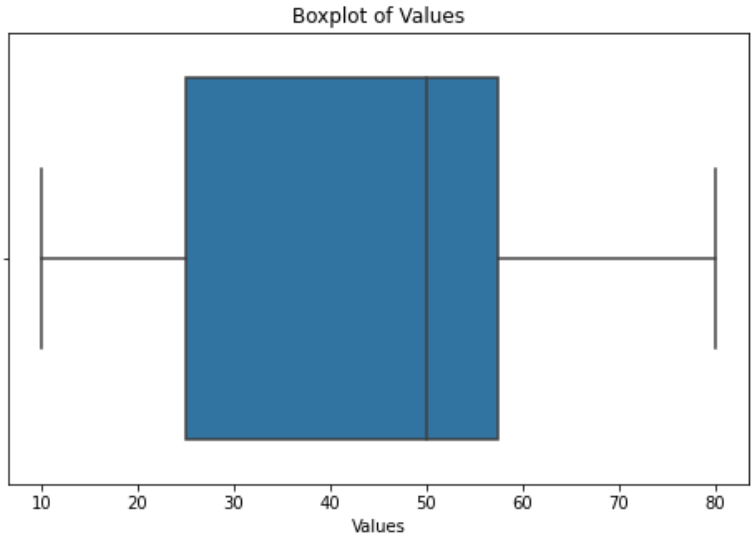
3. Visualizing Central Tendency and Dispersion

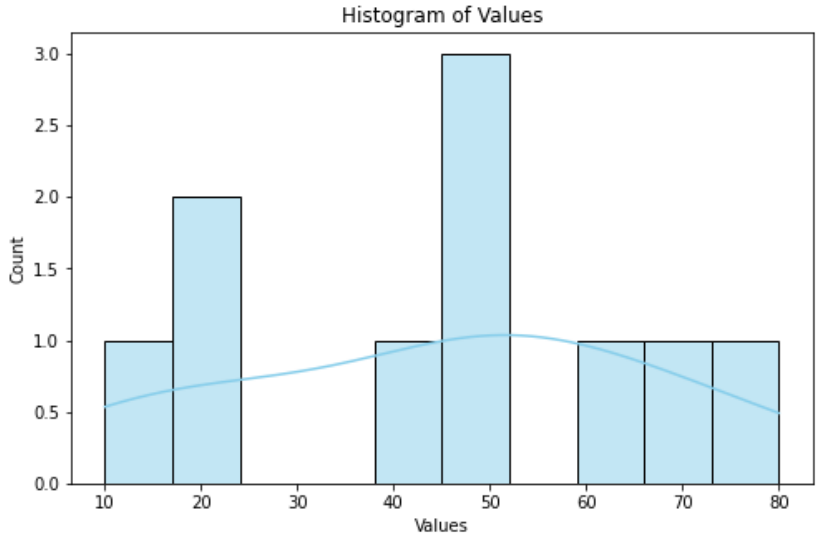
To better understand these measures, you can visualize the data:

Boxplot and Histogram

```
import matplotlib.pyplot as plt
import seaborn as sns
# Boxplot
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Values'])
plt.title('Boxplot of Values')
plt.show()

# Histogram
plt.figure(figsize=(8, 5))
sns.histplot(df['Values'], bins=10, kde=True, color='skyblue')
plt.title('Histogram of Values')
plt.xlabel('Values')
plt.show()
```

In [26]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Boxplot
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Values'])
plt.title('Boxplot of Values')
plt.show()

# Histogram
plt.figure(figsize=(8, 5))
sns.histplot(df['Values'], bins=10, kde=True, color='skyblue')
plt.title('Histogram of Values')
plt.xlabel('Values')
plt.show()
```





Explanation:Boxplot: Highlights the median, IQR, and outliers. & Histogram: Shows the distribution and central tendency of the data.

Summary Table:

Measure	Python Method
Mean	`df['Values'].mean()`
Median	`df['Values'].median()`
Mode	`df['Values'].mode()[0]`
Range	`max() - min()`
Variance	`df['Values'].var()`
Standard Deviation	`df['Values'].std()`
Interquartile Range	`Q3 - Q1` using `quantile()`