# Experiment 1

## Objective:

To study about numpy, pandas and matplotlib libraries in python.

## Theory

In Python programming, libraries such as NumPy, Pandas, and Matplotlib are fundamental for data manipulation, analysis, and visualization. These libraries streamline complex operations, enabling data scientists, analysts, and developers to handle large datasets, perform numerical computations, and visualize data effectively.

**NumPy (Numerical Python)** is a powerful library primarily used for numerical and matrix computations. It introduces the `ndarray` object, an N-dimensional array for efficiently storing and manipulating large arrays of homogeneous data. NumPy arrays are faster and more memory-efficient than traditional Python lists due to their fixed size and storage of elements in contiguous memory. This efficiency allows for rapid mathematical computations and operations across entire arrays without the need for explicit loops. NumPy also provides a range of mathematical functions, including linear algebra, Fourier transforms, and random number generation. The broadcasting feature in NumPy enables arithmetic operations on arrays of different shapes, making it flexible for various data manipulation tasks. NumPy is often used in fields like machine learning, scientific computing, and engineering, where heavy numerical computation is required.

**Pandas** is another essential library in Python, designed specifically for data manipulation and analysis. It provides two primary data structures: `Series` (1D) and `DataFrame` (2D), which are built on top of NumPy arrays. A DataFrame can hold heterogeneous data types across columns, making it ideal for handling and analyzing structured data. Pandas offers a variety of functions for data cleaning, filtering, grouping, merging, and aggregation. It supports handling missing data, which is a common issue in real-world datasets. Pandas also has tools for time series analysis, making it valuable for financial and temporal data. The ability to handle large datasets in-memory, apply various operations on data frames, and reshape data makes Pandas a powerful tool for any data-driven task. Through Pandas, data can be imported from numerous file formats, such as CSV, Excel, and SQL databases, facilitating easy data integration and analysis.

**Matplotlib** is a popular Python library for data visualization. Its primary goal is to provide an easy way to generate visual representations of data, making it easier to understand complex patterns and relationships. Matplotlib offers an extensive range of plotting options, including line plots, scatter plots, bar charts, histograms, and more. The library's core, `pyplot`, provides a MATLAB-like interface for creating interactive and customizable plots. It allows users to control various aspects of a plot, such as labels, colors, styles, and legends, enabling high levels of customization. Matplotlib is often combined with Pandas, as Pandas' built-in plotting functions use Matplotlib as the backend, simplifying the visualization of DataFrames directly. Visualizations are essential in data analysis, as they provide insights and highlight trends that may not be evident through raw numbers alone.

Together, NumPy, Pandas, and Matplotlib form the foundation of Python's data science stack. NumPy handles numerical computations, Pandas manages data manipulation, and Matplotlib provides visualization tools, making it easier for data scientists to analyze and interpret data.

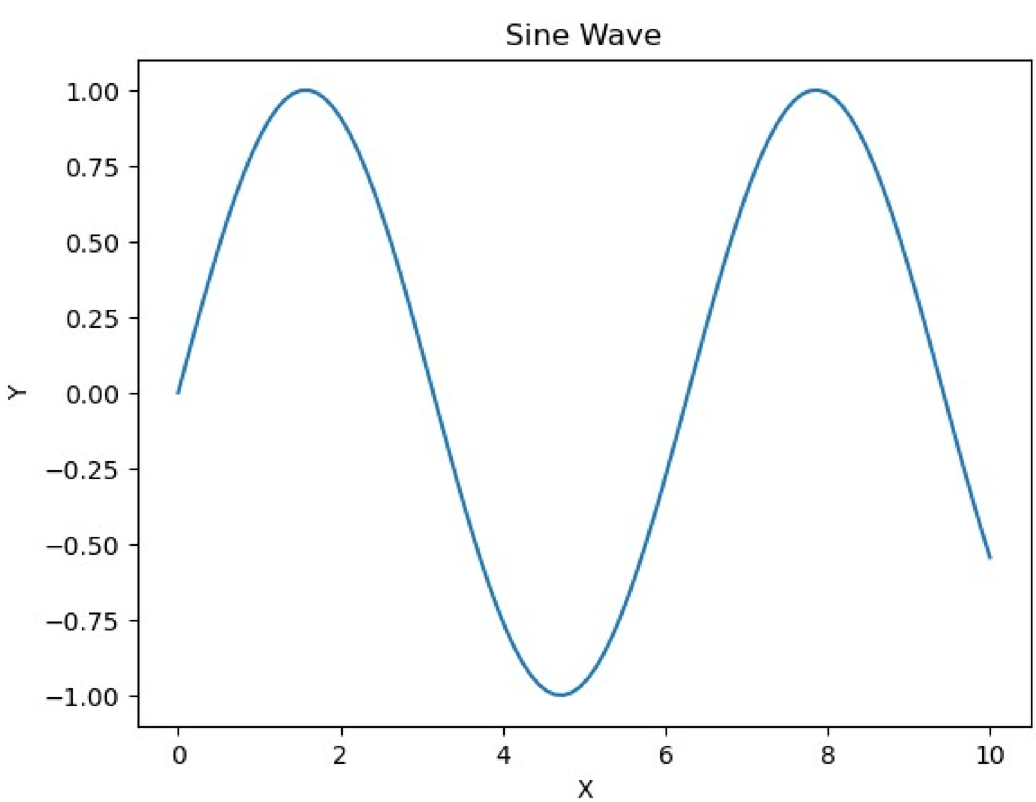
## Code & OUTPUT

In [1]:
```
print("Experiment No 01 : To study about numpy, pandas and matplotlib libraries in pytho
```

```
Experiment No 01 : To study about numpy, pandas and matplotlib libraries in python.
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
array = np.array([1,2,3,4,5])
print("OUTPUT:\n\n")
print("Numpy Array :",array)
data={'Name':['John','Anna','Peter','Linda'],
     'Age':[28,24,35,32]}
df = pd.DataFrame(data)
print("Pandas DataFrame:\n",df)
x = np.linspace(0,10,100)
y=np.sin(x)
plt.plot(x,y)
plt.title("Sine Wave")
plt.xlabel("X")
plt.ylabel("Y")
```

```
OUTPUT:


Numpy Array : [1 2 3 4 5]
Pandas DataFrame:
     Name  Age
0   John   28
1   Anna   24
2  Peter   35
3  Linda   32
```



## Result

As a result of this Experiment, we successfully wrote and executed the program to study about numpy, pandas and matplotlib libraries in python.

## Learning Outcomes

Understand and utilize Python libraries NumPy, Pandas, and Matplotlib for numerical operations, data manipulation, and data visualization in data science tasks.

# Experiment 2

## Objective:

To perform data preprocessing and data summarization on iris dataset.

## Theory

Data preprocessing and summarization are critical steps in the data analysis pipeline, particularly when working with machine learning models. These processes ensure that data is clean, consistent, and ready for analysis. The Iris dataset, one of the most well-known datasets in data science, serves as an excellent example for demonstrating these techniques. This dataset includes 150 samples of iris flowers, each described by four features: sepal length, sepal width, petal length, and petal width. Additionally, each sample is labelled as belonging to one of three species of iris: Iris-setosa, Iris-versicolor, and Iris-virginica.

**Data Preprocessing** is a series of steps used to prepare raw data for analysis or modelling. It involves cleaning and transforming the data, handling missing values, and ensuring consistency in data formats. In the case of the Iris dataset, data preprocessing might involve verifying that each feature is numeric and consistent in scale. Since the dataset does not contain missing values, a typical first step is to check for any outliers or data inconsistencies, though the Iris dataset is known for its clean structure. However, in more complex datasets, preprocessing could include filling missing values using techniques like mean imputation, median imputation, or even more sophisticated methods like k-nearest neighbours.

Another preprocessing step is **data normalization or standardization**, especially when working with distance-based machine learning models such as k-nearest neighbours. Normalization scales features to a range, typically [0,1], while standardization scales them to have a mean of 0 and a standard deviation of 1. For the Iris dataset, where all four features are continuous and on different scales, these transformations can ensure that each feature contributes equally to model performance.

**Data Summarization** follows preprocessing and is used to understand the characteristics of the dataset. Summary statistics provide insights into the distribution and spread of the data, aiding in pattern identification. Descriptive statistics like mean, median, standard deviation, minimum, and maximum values are calculated for each feature. For instance, the mean sepal length and standard deviation help provide a quick understanding of the central tendency and spread of this feature. Summarization also includes visual techniques such as histograms, box plots, and pair plots. For example, a pair plot can illustrate relationships between sepal length, sepal width, petal length, and petal width across the different species in the Iris dataset. Box plots can reveal the distribution of each feature, highlighting any outliers or variability between species.

Furthermore, **data visualization** is a part of data summarization that provides a graphical representation of statistical summaries. In the case of the Iris dataset, scatter plots of petal length versus petal width, colored by species, can reveal clusters of species and help in visualizing decision boundaries. These visualizations are crucial when interpreting data patterns before any modeling phase.

## Code & OUTPUT

In [1]:
```python
print("Experiment No 02 : To perform data preprocessing and data summarization on iris dataset.")
```

```
Experiment No 02 : To perform data preprocessing and data summarization on iris dataset.
```

In [2]:
```python
# Load libraries
import pandas as pd
from sklearn.datasets import load_iris
print("OUTPUT:\n\n")
# Load iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target
# Summarization
print(df.describe())
print(df.info())
# Checking for missing values
print(df.isnull().sum())
# Data Preprocessing (Normalizing the data)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df.iloc[:, :-1]), columns=iris.feature_names)
```

```
OUTPUT:


       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)     species  
count        150.000000  150.000000  
mean           1.199333    1.000000  
std            0.762238    0.819232  
min            0.100000    0.000000  
25%            0.300000    0.000000  
50%            1.300000    1.000000  
75%            1.800000    2.000000  
max            2.500000    2.000000  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
None
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
Scaled Data:
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0          -0.900681          1.019004          -1.340227         -1.315444
1          -1.143017         -0.131979          -1.340227         -1.315444
2          -1.385353          0.328414          -1.397064         -1.315444
3          -1.506521          0.098217          -1.283389         -1.315444
4          -1.021849          1.249201          -1.340227         -1.315444
```

## Result

As a result of this Experiment, we successfully wrote and executed the program to perform data preprocessing and data summarization on iris dataset.

.

## Learning Outcomes

Understand and apply data preprocessing and summarization techniques to clean, normalize, and analyse datasets, gaining insights into feature distributions and relationships.

# Experiment 3

## Objective:

To perform data preprocessing and data visualization on iris dataset.

## Theory

Data preprocessing and visualization are foundational steps in data analysis, particularly in machine learning and data science. They prepare data for deeper analysis and help reveal underlying patterns. Using the Iris dataset—a widely studied dataset in data science that includes 150 observations of iris flowers—allows us to practice these techniques effectively. This dataset contains four numerical features: sepal length, sepal width, petal length, and petal width, as well as a categorical target label representing the species of the iris flower: Iris-setosa, Iris-versicolor, and Iris-virginica.

**Data Preprocessing** involves cleaning and transforming data to improve its quality and suitability for analysis. Preprocessing can include multiple steps, such as handling missing values, scaling features, encoding categorical data, and removing duplicates. In the Iris dataset, although no missing values are present, it is still useful to check the dataset for any irregularities. Scaling or normalizing the features is also essential, especially if the data will be used in models sensitive to feature scales, such as k-nearest neighbors. Two popular scaling methods are normalization, which transforms features to a [0, 1] range, and standardization, which scales features to have a mean of 0 and a standard deviation of 1.

**Data Visualization** provides a visual interpretation of data and allows us to identify patterns, trends, and relationships within the dataset. Visualizations make it easier to compare features, observe correlations, and understand data distribution. Several plotting techniques can be applied to the Iris dataset:

1. **Scatter Plots**: By plotting feature pairs such as petal length vs. petal width and color-coding points by species, scatter plots reveal natural clusters. For example, petal length and petal width are particularly effective in distinguishing species clusters in the Iris dataset, as Iris-setosa tends to form a distinct group from Iris-versicolor and Iris-virginica.
2. **Box Plots**: Box plots show the distribution of each feature across the three iris species. They reveal the range, quartiles, and potential outliers within each feature, highlighting differences in feature distributions across species.
3. **Histograms**: Histograms help visualize the frequency distribution of each feature. For instance, a histogram of sepal length can show whether the values are normally distributed and if any values stand out as outliers.
4. **Pair Plots (or Scatterplot Matrix)**: A pair plot displays scatter plots for each pair of features, with color coding by species. This approach provides a comprehensive view of feature relationships and how they may relate to the species classification.
5. **Violin Plots**: These combine the features of box plots and histograms to show the distribution of each feature across species, providing insights into both the range and density of values within each species.

## Code & OUTPUT

In [1]:
```
print("Experiment No 03 : To perform data preprocessing and data visualization on iris dataset.")
```

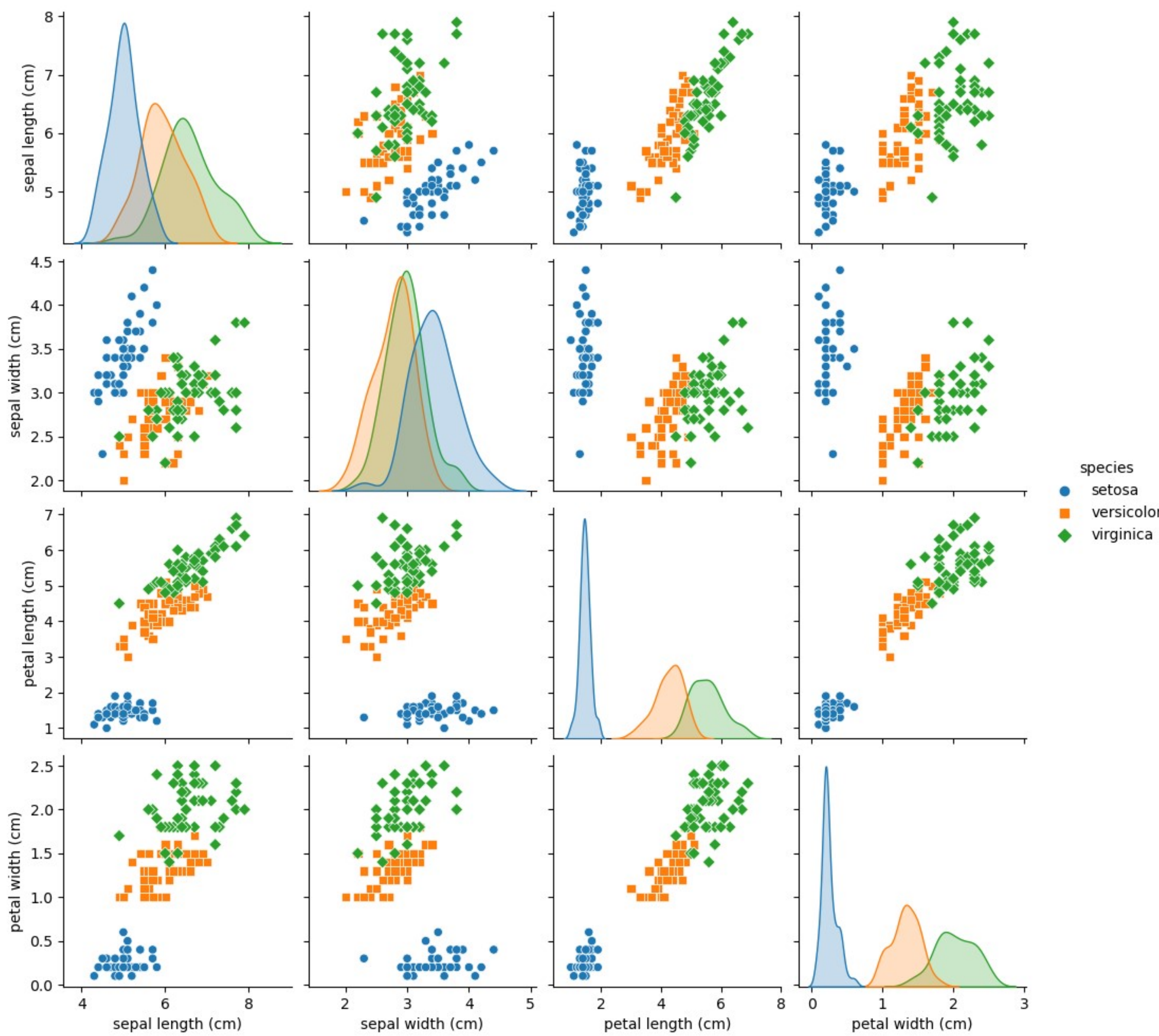
Experiment No 03 : To perform data preprocessing and data visualization on iris dataset.

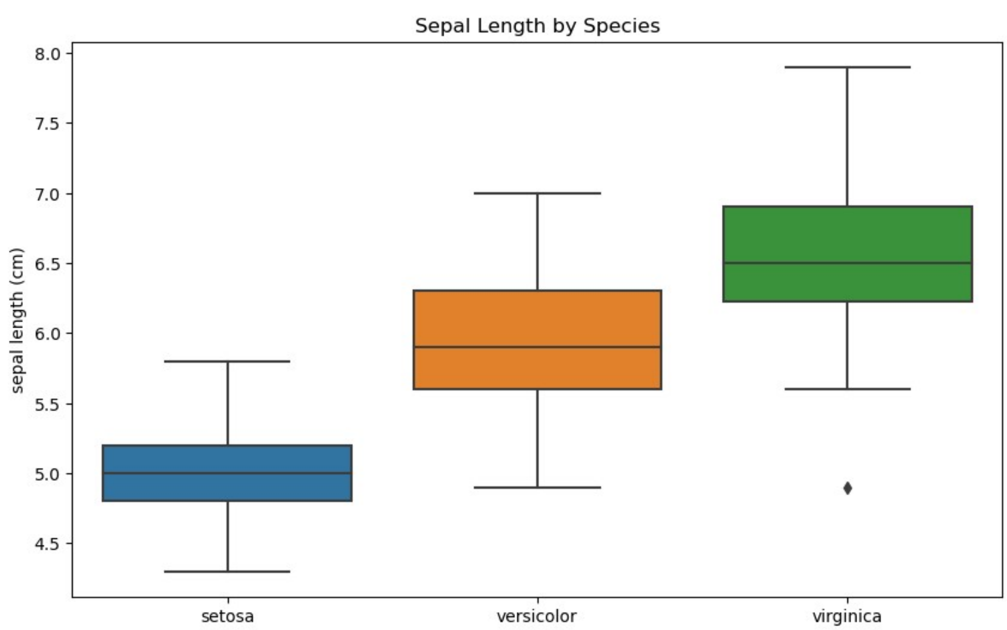
In [4]:
```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
print("OUTPUT:\n\n")

# Load the iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].apply(lambda x: iris.target_names[x])

# Display basic information about the dataset
print("Dataset preview:")
print(df.head())
print("\nDataset summary:")
print(df.describe())
print("\nClass distribution:")
print(df['species'].value_counts())

# Data Preprocessing
# Separate features and target
X = df.iloc[:, :-1]  # Features (sepal and petal measurements)
y = df['species']    # Target (species)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("\nData preprocessing complete.")

# Data Visualization

# Pair Plot
sns.pairplot(df, hue='species', markers=["o", "s", "D"])
plt.suptitle("Pairplot of Iris Dataset", y=1.02)
plt.show()

# Box Plot for Sepal Length and Petal Length by Species
plt.figure(figsize=(10, 6))
sns.boxplot(x='species', y='sepal length (cm)', data=df)
plt.title("Sepal Length by Species")
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='species', y='petal length (cm)', data=df)
plt.title("Petal Length by Species")
plt.show()

# Violin Plot for Sepal Width and Petal Width by Species
plt.figure(figsize=(10, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=df)
plt.title("Sepal Width by Species")
plt.show()

plt.figure(figsize=(10, 6))
sns.violinplot(x='species', y='petal width (cm)', data=df)
plt.title("Petal Width by Species")
plt.show()

# Heatmap of Correlation Matrix (excluding species column)
plt.figure(figsize=(8, 6))
sns.heatmap(df.iloc[:, :-1].corr(), annot=True, cmap='coolwarm', square=True)
plt.title("Correlation Heatmap of Iris Dataset")
plt.show()

# Histograms of Each Feature
df.iloc[:, :-1].hist(edgecolor='black', linewidth=1.2, figsize=(10, 8))
plt.suptitle("Feature Distributions")
plt.show()
```

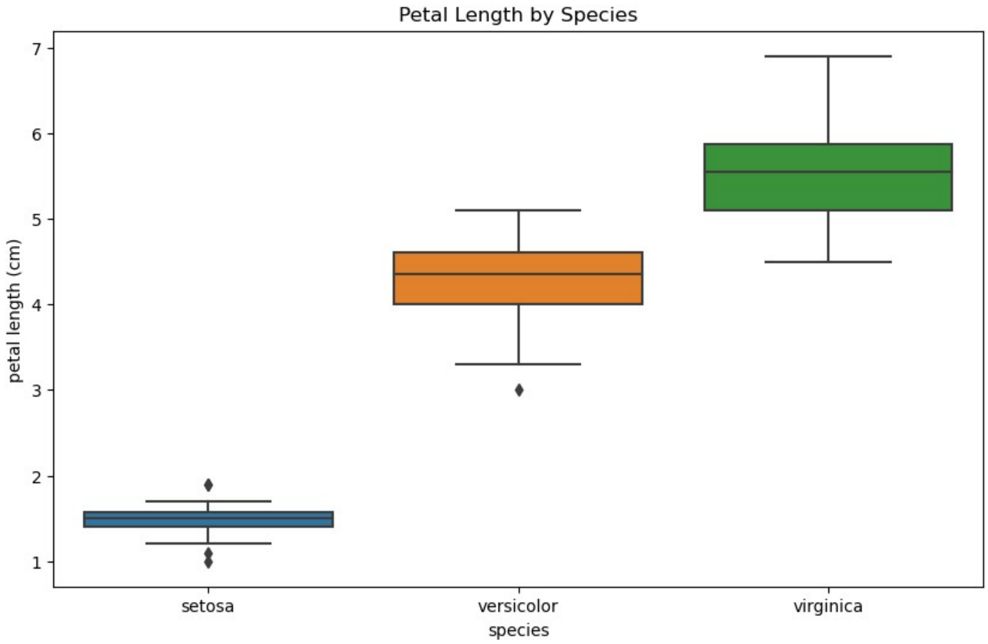
OUTPUT:

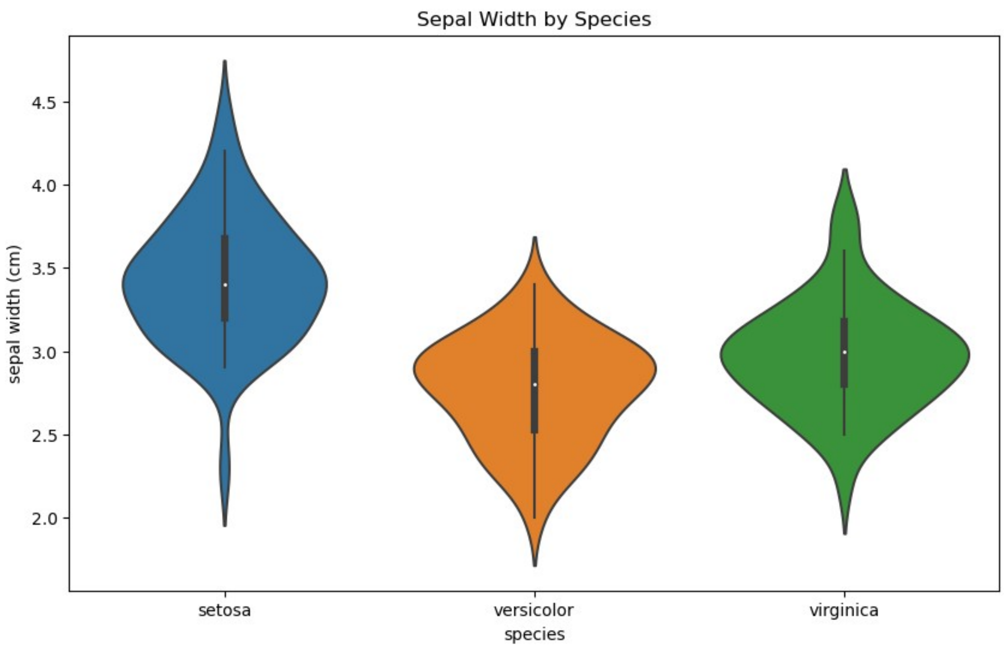
```
Dataset preview:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa

Dataset summary:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000

Class distribution:
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64

Data preprocessing complete.
c:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

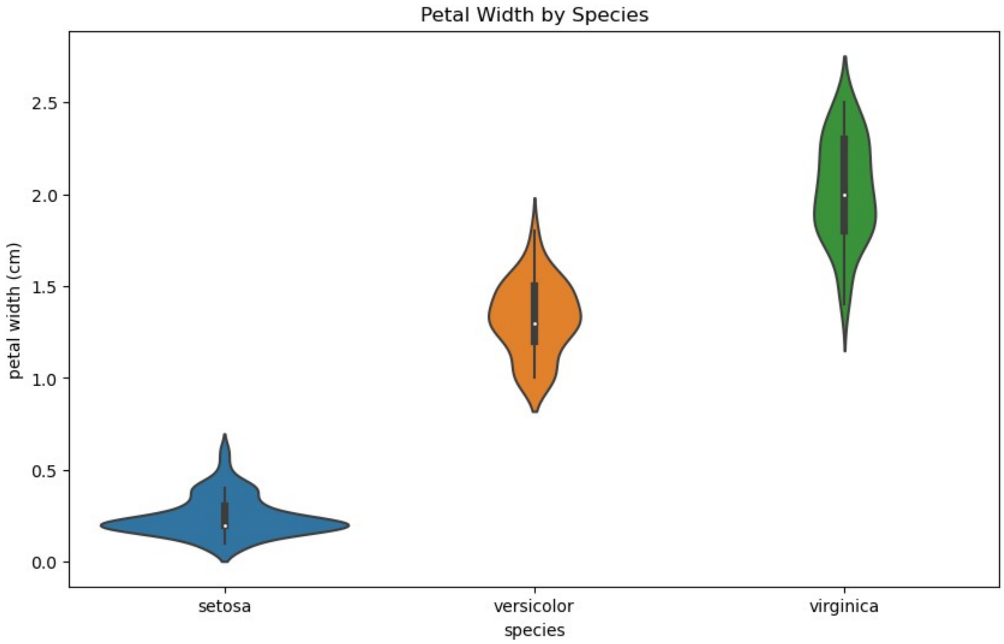
Pairplot of Iris Dataset

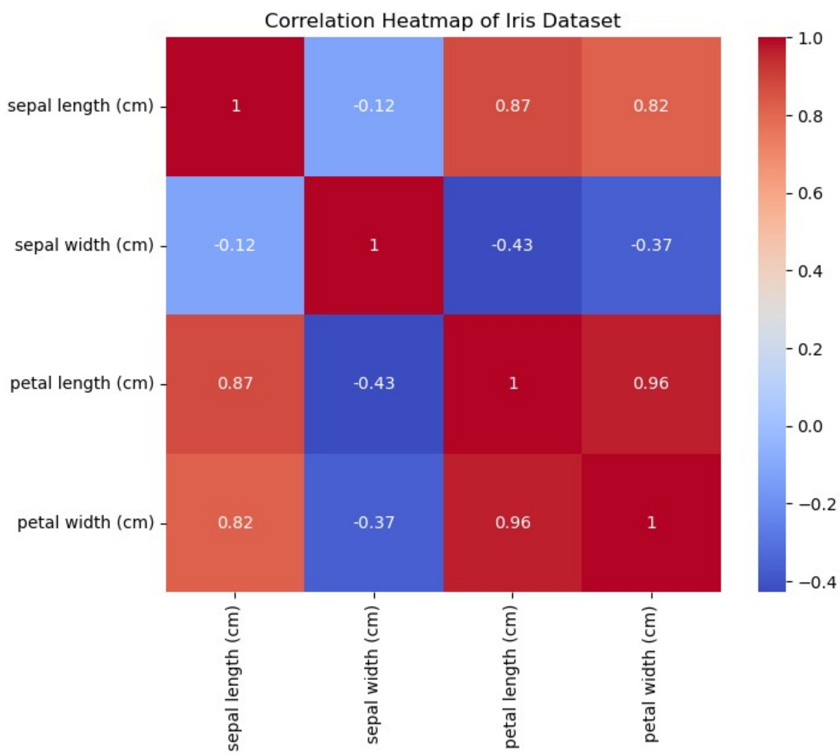


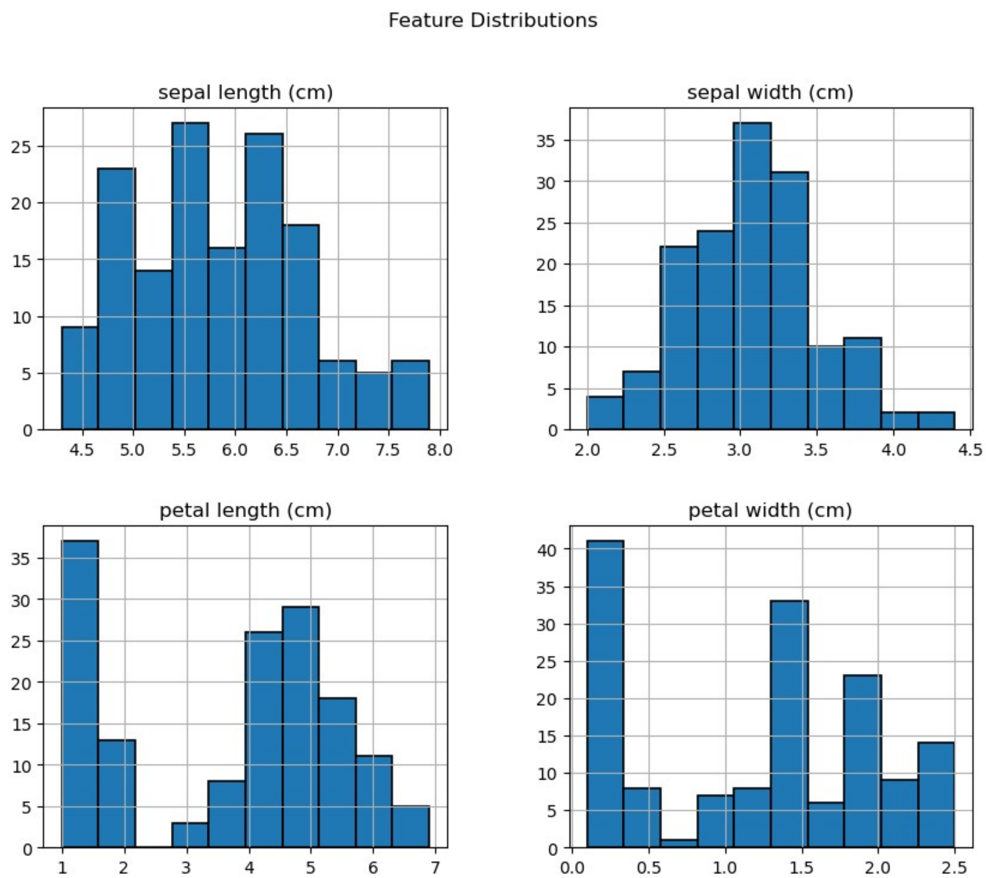




Petal Length by Species
petal length (cm)
1
2
3
4
5
6
7
setosa
versicolor
virginica
species


Sepal Width by Species
sepal width (cm)
2.0
2.5
3.0
3.5
4.0
4.5
setosa
versicolor
virginica
species


Petal Width by Species
petal width (cm)
0.0
0.5
1.0
1.5
2.0
2.5
setosa
versicolor
virginica
species


Correlation Heatmap of Iris Dataset
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
1
-0.12
0.87
0.82
-0.12
1
-0.43
-0.37
0.87
-0.43
1
0.96
0.82
-0.37
0.96
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Feature Distributions
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)

## Result
As a result of this Experiment, we successfully wrote and executed the program to perform data preprocessing and data visualization on iris dataset.

## Learning Outcomes
Understand and apply data preprocessing techniques and various visualization methods to clean, explore, and interpret patterns in the Iris dataset effectively.