CO-TECH INTERNSHIP – TASK 1 EXPLORATION EXPLORATORY DATA ANALYSIS (EDA)

TASK - 1 [Start with a dataset of your choice and perform EDA using libraries like pandas, numpy, and matplotlib or seaborn. Explore the data's characteristics, distributions, correlations, and outliers. Visualize your findings with histograms, scatter plots, and heatmaps to gain insights into the data]

1. PROJECT OVERVIEW

- This project performs Exploratory Data Analysis (EDA) on the Iris Dataset to gain insights into the relationships between different features of the iris flowers. The primary objective is to explore the dataset's characteristics, visualize distributions, correlations, and outliers, and analyze the relationships between numerical and categorical variables. The Iris Dataset contains measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers. This analysis helps in understanding the dataset before applying machine learning algorithms.

2. Dataset Description

- The **Iris Dataset** contains 150 observations of iris flowers, with 50 samples from each of the three species:
- Iris setosa, Iris versicolor, and Iris virginica. The dataset includes the following attributes: • *Sepal Length (cm)*: Continuous variable representing the length of the sepal. • *Sepal Width* (cm): Continuous variable representing the width of the sepal. • *Petal Length* (cm): Continuous variable representing the length of the petal. • *Petal Width* (cm): Continuous variable representing the width of the petal. • *Species*: Categorical variable representing the species of the iris flower, with values: 'Iris setosa', 'Iris versicolor', and 'Iris virginica'.

STEP 1 - SET UP THE ENVIROMENT

Libraries Used

```
1. pandas: For data manipulation and analysis (loading, cleaning,
transforming data).

2. numpy: For numerical operations and handling arrays.

3. matplotlib: For creating static, animated, and interactive
visualizations.

4. seaborn: For statistical data visualization, including plots
like heatmaps, pair plots, and boxplots.
```

to import all the libraries required

In [9]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
```

In [11]:
```
import pandas as pd  # Import pandas
```

1. Data Loading:

Loaded the Iris dataset into a pandas DataFrame from a CSV file.

In [13]:
```
df = pd.read_csv('iris_dataset.csv')
```

Step - 2 : Data Exploration:

a) df.info - Provides information on the dataset, such as the number of entries, column names, data types, and null values.

In [12]:
```
df.info()
```

```
<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    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [16]:
```
iris_df = pd.read_csv('C:/Users/User/Downloads/iris_dataset.csv')
```

In [20]:
```
iris_df = pd.read_csv('iris_dataset.csv')
```

b) df.dtypes - Provides Lists the data types for each column

In [17]:
```
iris_df.dtypes
```

Out[17]:
```
sepal length (cm)    float64
sepal width (cm)     float64
petal length (cm)    float64
petal width (cm)     float64
species               object
dtype: object
```

c) df.head - To read the head the and the first row of the dataset

The `data.head()` method allows you to quickly see the structure of the dataset. For this task, we're assuming `petal length` as the feature and `petal width` as the target variable.

In [18]:
```
print(iris_df.head())
```

```
   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
```

d) df.columns - It shows the names of all columns

In [9]:
```
iris_df.columns
```

Out[9]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species'],
      dtype='object')
```

e) df.describe - Generates descriptive statistics for each numerical column, including mean, standard deviation, and percentiles.

In [22]:
```
iris_df.describe()
```

Out[22]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Step - 3 : Visual Analysis

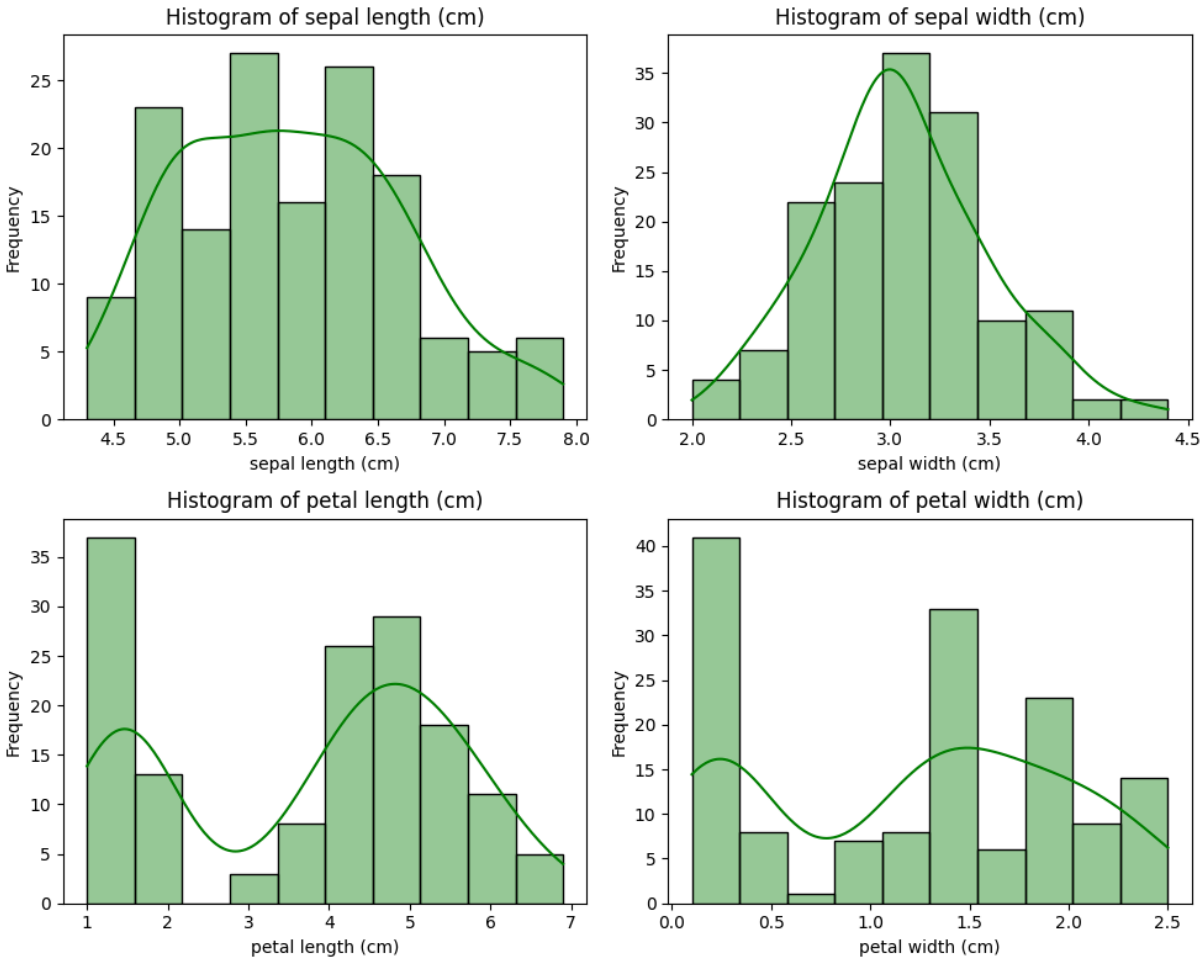
The visual analysis phase involves examining the distributions, relationships, and patterns in the dataset through various types of plots:

a) Histograms for Feature Distributions:

1. Purpose: To understand the distribution of each numerical feature (sepal length, sepal width, petal length, and petal width).

3. Insight: Histograms help reveal whether features are normally distributed, skewed, or bimodal. They also provide a quick overview of ranges and potential variations within each feature. For instance, petal width and petal length typically show more variation across species than sepal measurements.

In [46]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virg

# Set the figure size
plt.figure(figsize=(10, 8))

# Create histograms for each feature
for i, feature in enumerate(iris_df.columns[:-1]):
    plt.subplot(2, 2, i + 1)
    sns.histplot(iris_df[feature], kde=True, bins=10, color='green', alpha=0.4)
    plt.title(f'Histogram of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```

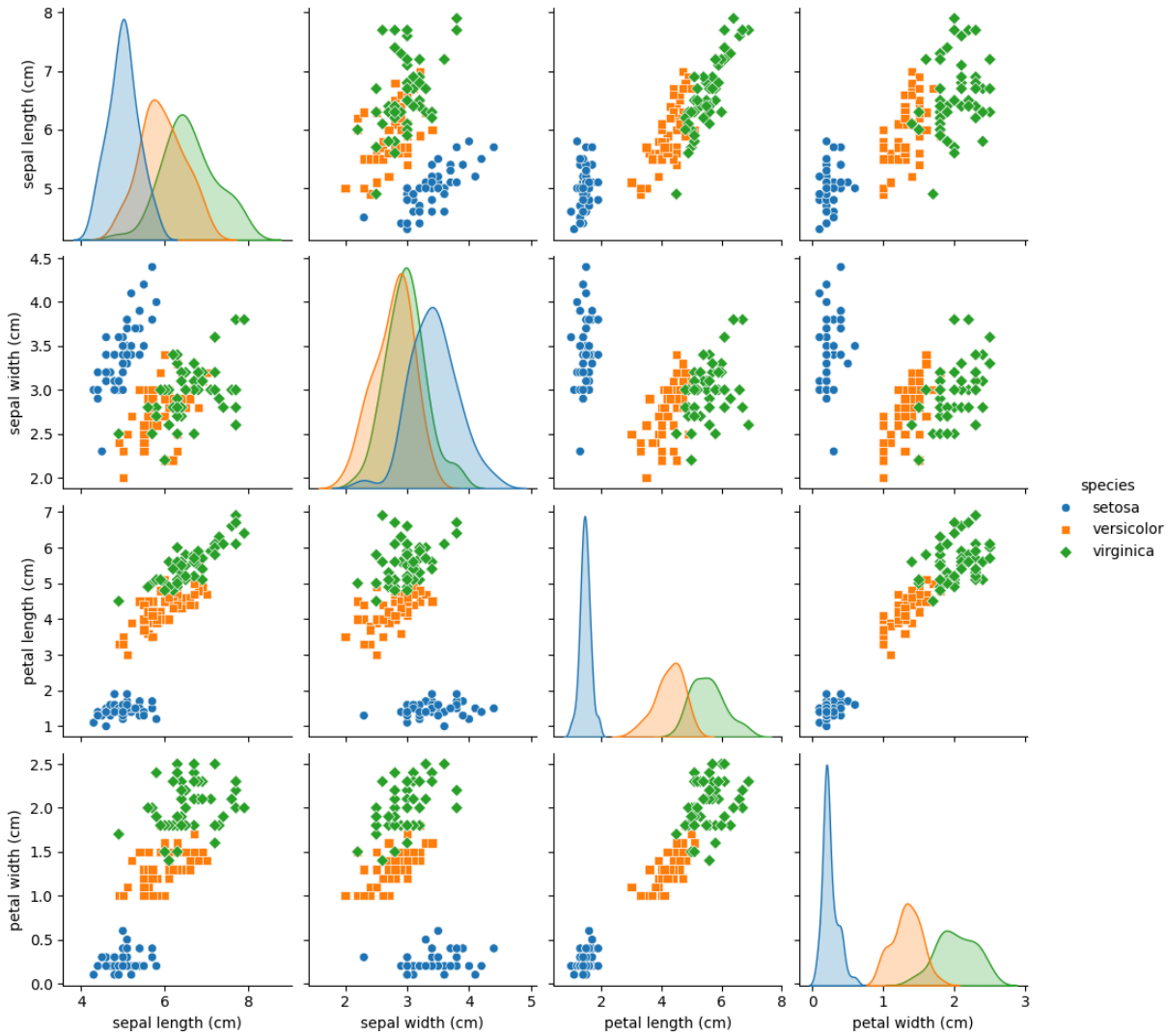


b) Pair Plot :

Purpose: To visualize distributions and relationships between all pairs of features simultaneously.

Insight: Pair plots provide a grid of scatter plots and histograms for quick, comprehensive insights. In the Iris dataset, pair plots can reveal natural groupings by species and help distinguish which features are the most informative for species classification.

In [15]:
```
sns.pairplot(iris_df, hue='species', markers=["o", "s", "D"])#marker is defined the
plt.show()
```



The below code identifies outliers in the Iris dataset using the Interquartile Range (IQR) method.

First, a DataFrame iris_df is created with the feature data. The first quartile (Q1, or 25th percentile) and third quartile (Q3, or 75th percentile) are calculated for each feature, and the IQR is derived by subtracting Q1 from Q3.

The lower and upper bounds for identifying outliers are then determined by extending 1.5 times the IQR below Q1 and above Q3, respectively.

Finally, rows in the dataset with values outside these bounds are flagged as outliers and displayed.

In [16]:
```
# redefine the loaded data
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
# Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
Q1 = iris_df.quantile(0.25)
Q3 = iris_df.quantile(0.75)

# Calculate IQR (Interquartile Range)
IQR = Q3 - Q1
```

```
# Determine the lower and upper bounds for outliers
lower_bound = Q1 - (1.5 * IQR)
upper_bound = Q3 + (1.5 * IQR)

# Find outliers
outliers = ((iris_df < lower_bound) | (iris_df > upper_bound))

# Display the outliers
outlier_values = iris_df[outliers.any(axis=1)]
print(outlier_values)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
15	5.7	4.4	1.5	0.4
32	5.2	4.1	1.5	0.1
33	5.5	4.2	1.4	0.2
60	5.0	2.0	3.5	1.0

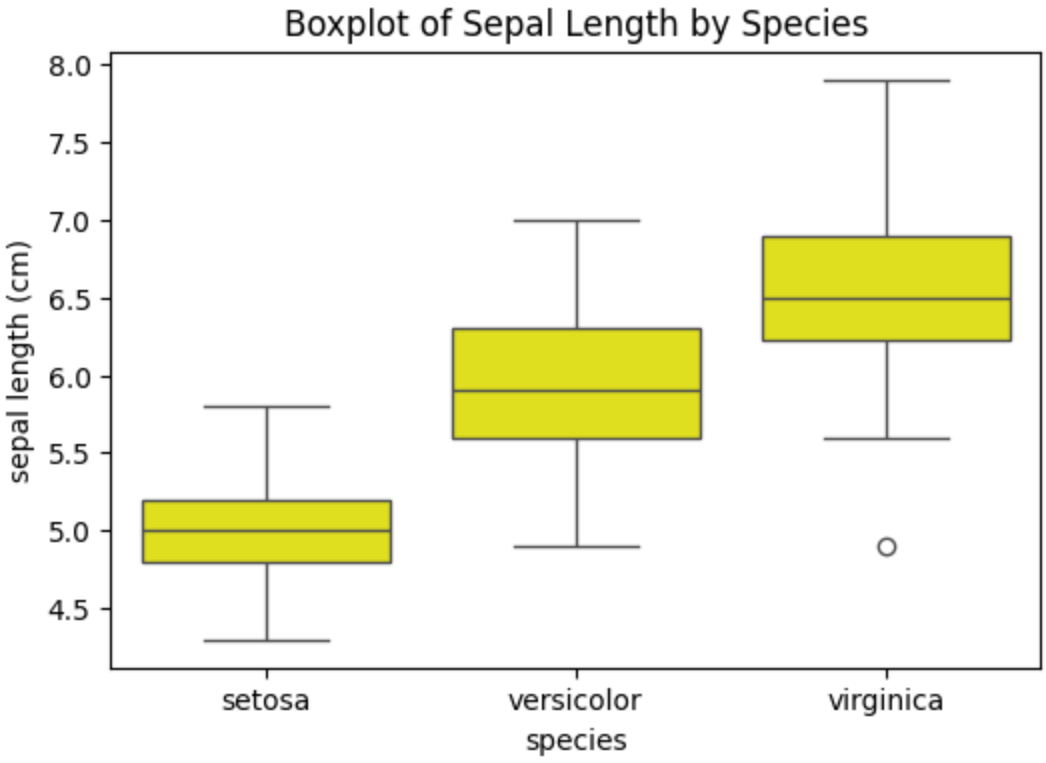
c) Boxplots for Outlier Detection:

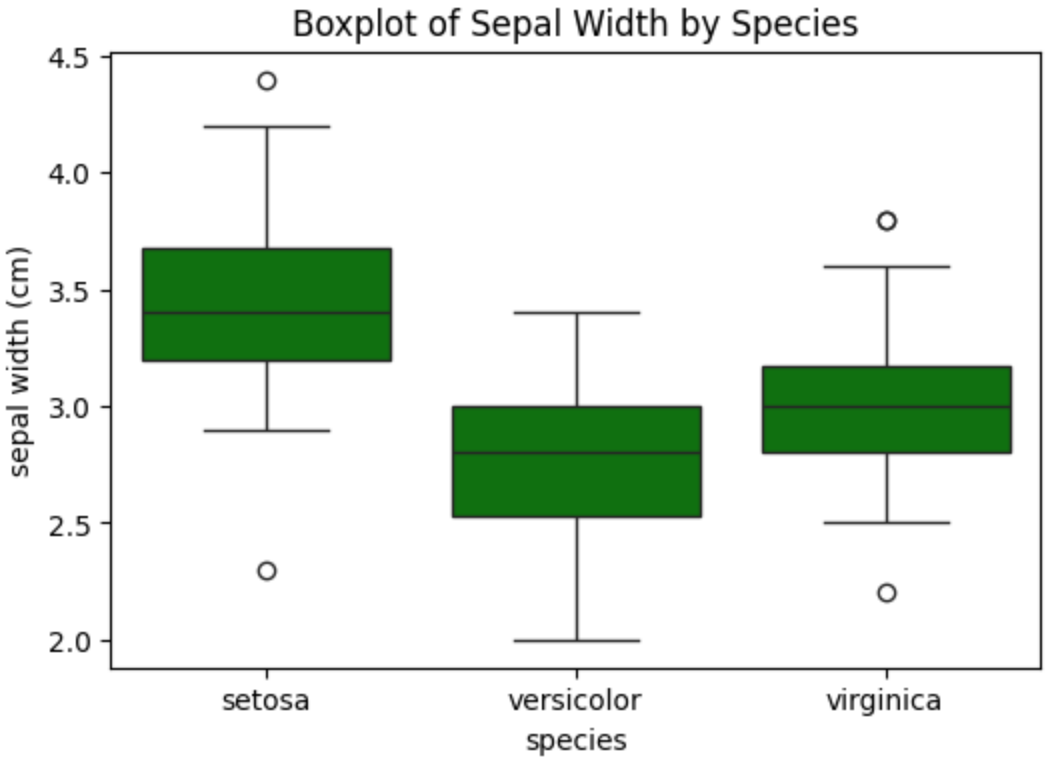
Purpose: To identify potential outliers and the spread of each feature.

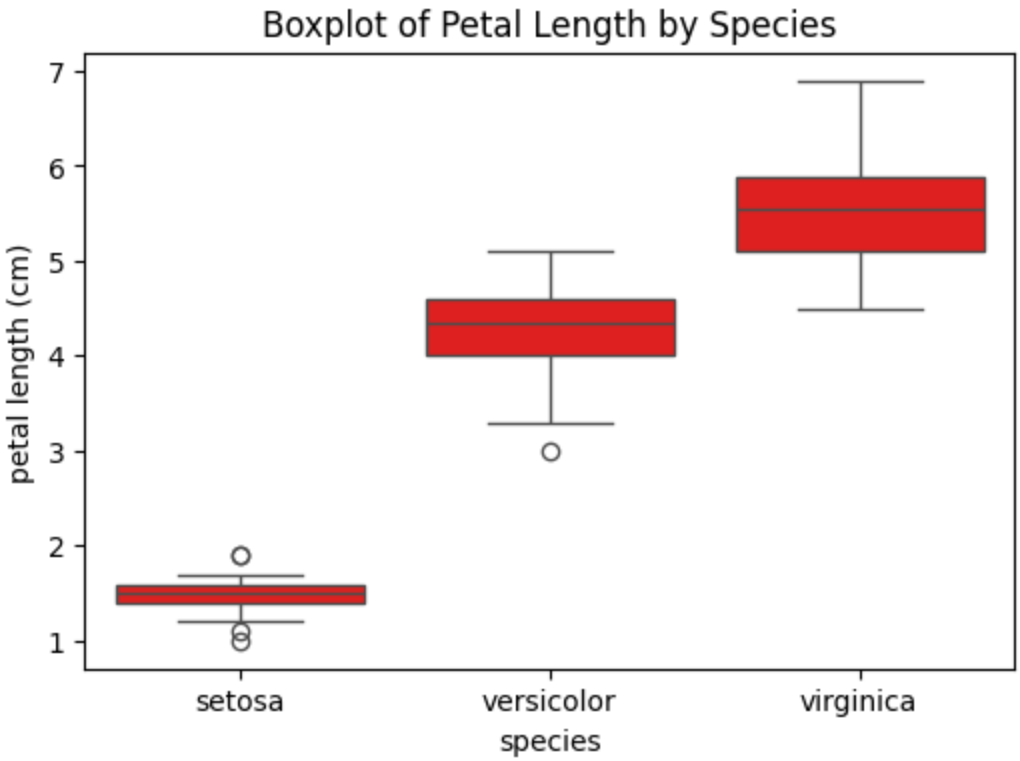
Insight: Boxplots allow us to observe how data is distributed around the median, revealing skewness, the interquartile range, and outliers. For example, petal width and length often have a greater spread and visible outliers, especially when comparing across species.

In [44]:
```
# again load the data for boxplot
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virg


plt.figure(figsize=(6, 4))
sns.boxplot(x='species', y='sepal length (cm)', data=iris_df,color='yellow',)
plt.title("Boxplot of Sepal Length by Species")
plt.show()


plt.figure(figsize=(6, 4))
sns.boxplot(x='species', y='sepal width (cm)', data=iris_df,color='green')
plt.title("Boxplot of Sepal Width by Species")
plt.show()


plt.figure(figsize=(6, 4))
sns.boxplot(x='species', y='petal length (cm)', data=iris_df,color='red')
plt.title("Boxplot of Petal Length by Species")
plt.show()


plt.figure(figsize=(6, 4))
sns.boxplot(x='species', y='petal width (cm)', data=iris_df,color='orange')
```

```
plt.title("Boxplot of Petal Width by Species")
plt.show()
```







CORRELATION MATRIX

In [18]:
```
Iris_data=iris_df[['petal length (cm)','petal width (cm)','sepal length (cm)','sepa
```

In [19]:
```
Iris_data
```

Out[19]:

	petal length (cm)	petal width (cm)	sepal length (cm)	sepal width (cm)
0	1.4	0.2	5.1	3.5
1	1.4	0.2	4.9	3.0
2	1.3	0.2	4.7	3.2
3	1.5	0.2	4.6	3.1
4	1.4	0.2	5.0	3.6
...	...	...	...	...
145	5.2	2.3	6.7	3.0
146	5.0	1.9	6.3	2.5
147	5.2	2.0	6.5	3.0
148	5.4	2.3	6.2	3.4
149	5.1	1.8	5.9	3.0

150 rows × 4 columns

It calculates and displays the correlation matrix of the Iris_data dataset. A *correlation matrix*is a table showing the correlation coefficients between pairs of features. Each coefficient ranges from -1 to 1, where:

1 indicates a perfect positive correlation: as one feature increases, the other does as well.

-1 indicates a perfect negative correlation: as one feature increases, the other decreases.

0 means no correlation: there's no linear relationship between the features.

In this matrix, the diagonal values are always 1 because each feature is perfectly correlated with itself.

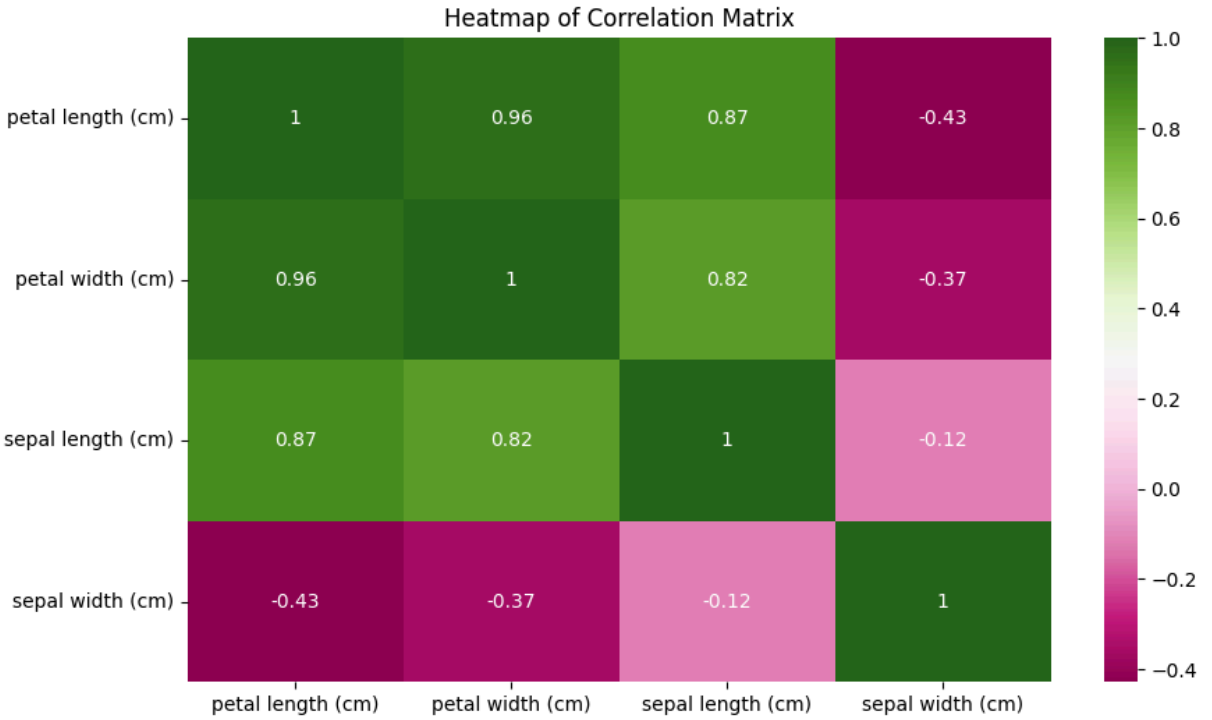
CORRELAGION WITH VARIABLES

In [20]:
```
corr_matrix = Iris_data.corr()
print(corr_matrix)
```

```
                   petal length (cm)  petal width (cm)  sepal length (cm)  \
petal length (cm)           1.000000          0.962865           0.871754
petal width (cm)            0.962865          1.000000           0.817941
sepal length (cm)           0.871754          0.817941           1.000000
sepal width (cm)           -0.428440         -0.366126          -0.117570

                   sepal width (cm)
petal length (cm)         -0.428440
petal width (cm)          -0.366126
sepal length (cm)         -0.117570
sepal width (cm)           1.000000
```

HEAPMAP OF COORELATION MATRIX

In [21]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='PiYG')
plt.title("Heatmap of Correlation Matrix")
plt.show()
```



This **heatmap** visually shows the strength and direction of relationships between the features in the Iris dataset.

Each cell in the heatmap represents the correlation value between two features, ranging from -1 (strong negative correlation) to +1 (strong positive correlation).

Positive values indicate that as one feature increases, the other also tends to increase

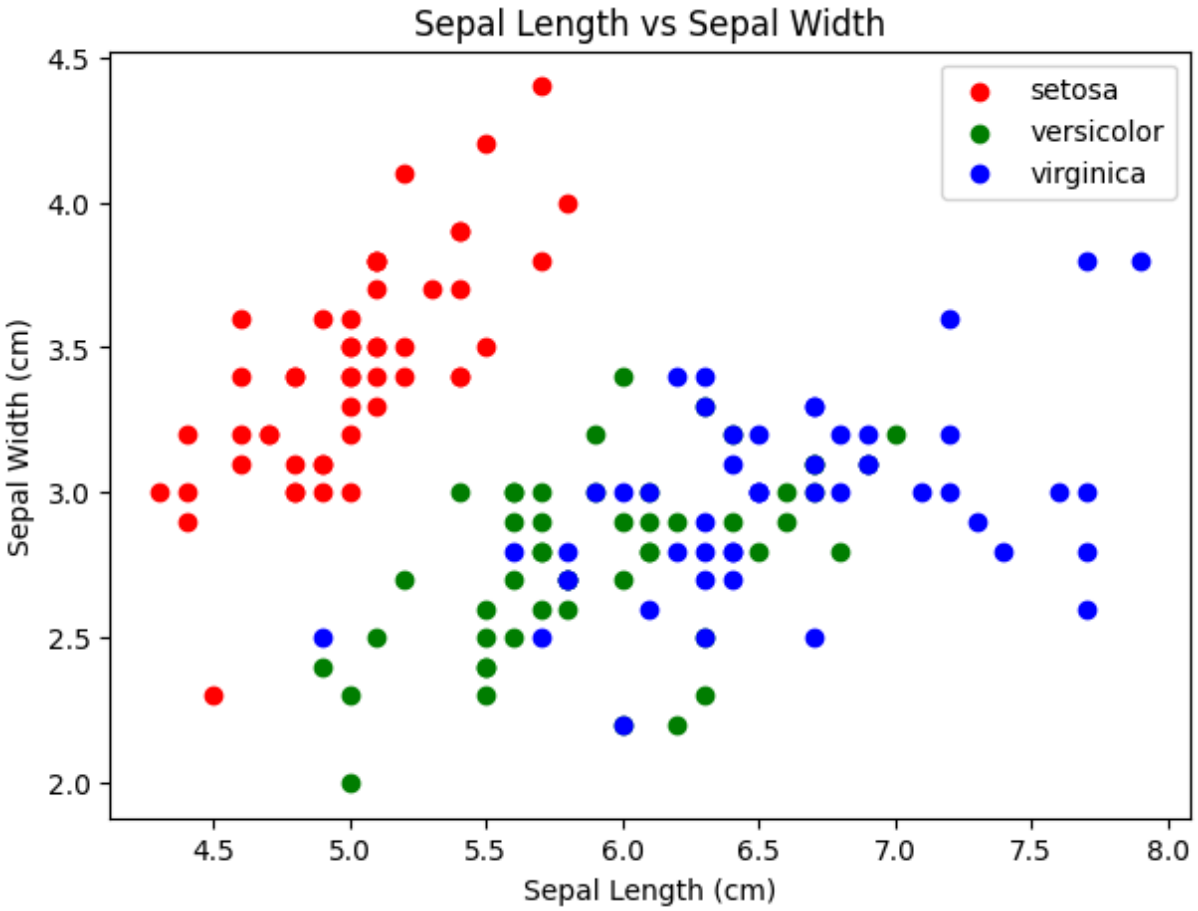
Negative values indicate an inverse relationship.

This analysis helps identify which features may be more closely related, which is useful for selecting features or understanding patterns in the data.

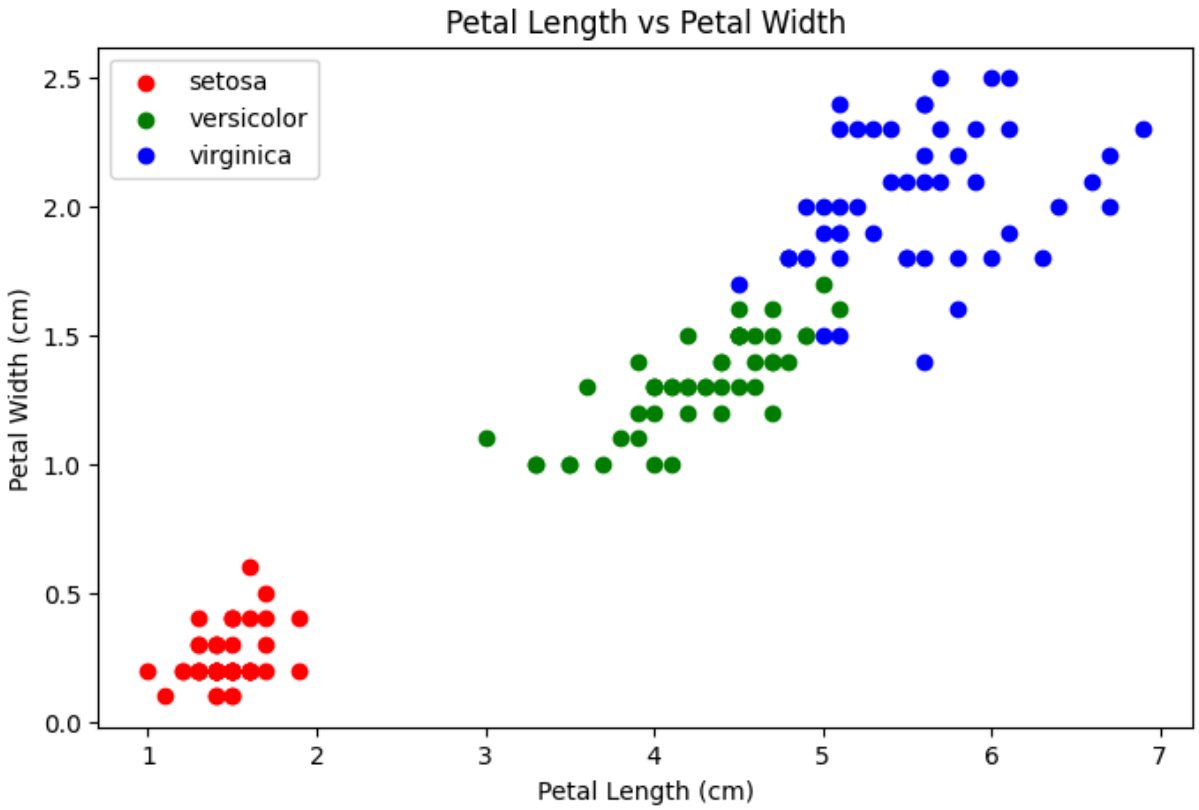
SCATTER PLOT USING Matplolib

In [31]:
```
# Scatter plot for Sepal length vs Sepal width
plt.figure(figsize=(7, 5))
colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}
for species, color in colors.items():
    subset = iris_df[iris_df['species'] == species]
    plt.scatter(subset['sepal length (cm)'], subset['sepal width (cm)'], label=spec
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
```

```
plt.title('Sepal Length vs Sepal Width')
plt.legend()
plt.show()
```

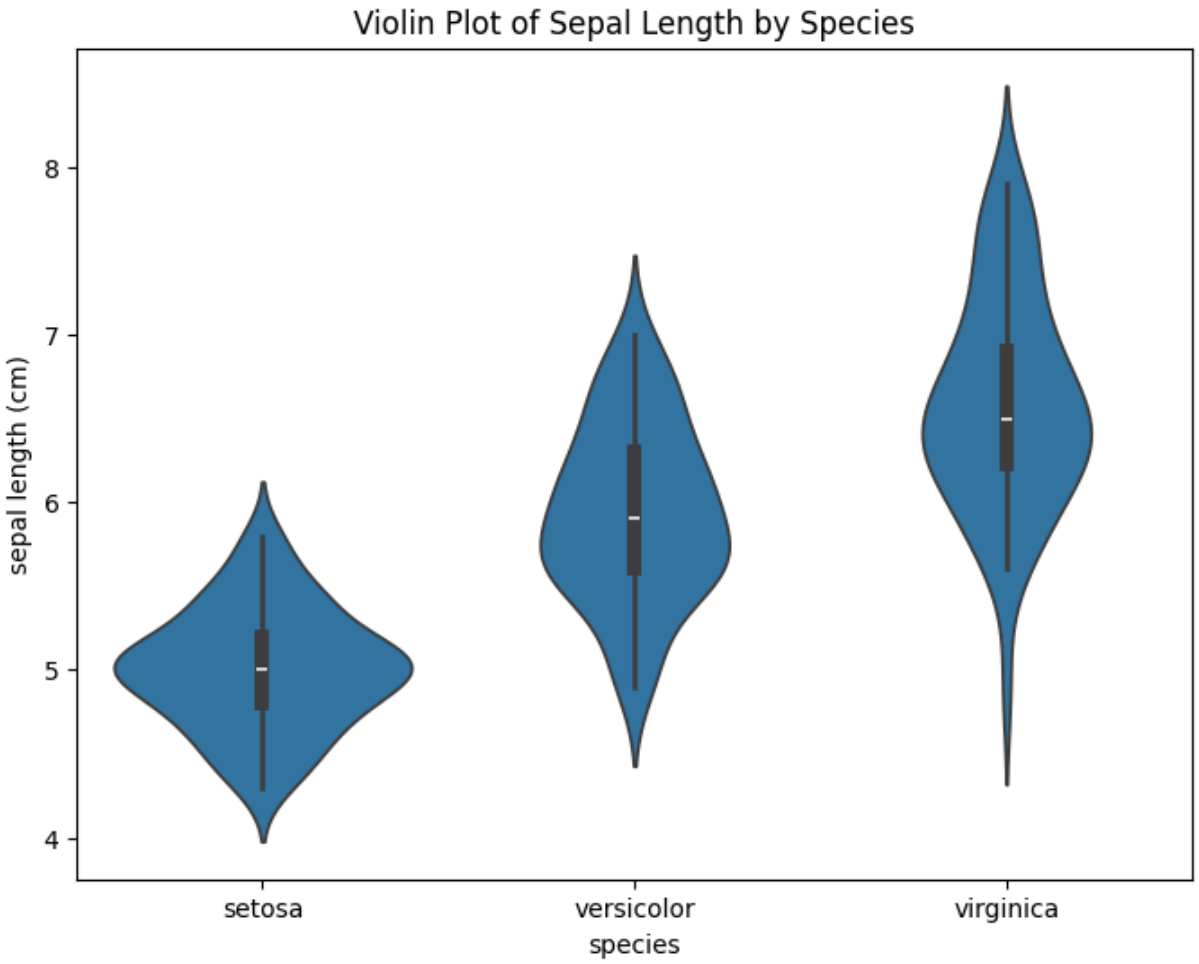


In [30]:
```
# Scatter plot for Petal length vs Petal width
plt.figure(figsize=(8, 5))
for species, color in colors.items():
    subset = iris_df[iris_df['species'] == species]
    plt.scatter(subset['petal length (cm)'], subset['petal width (cm)'], label=spec
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('Petal Length vs Petal Width')
plt.legend()
plt.show()
```

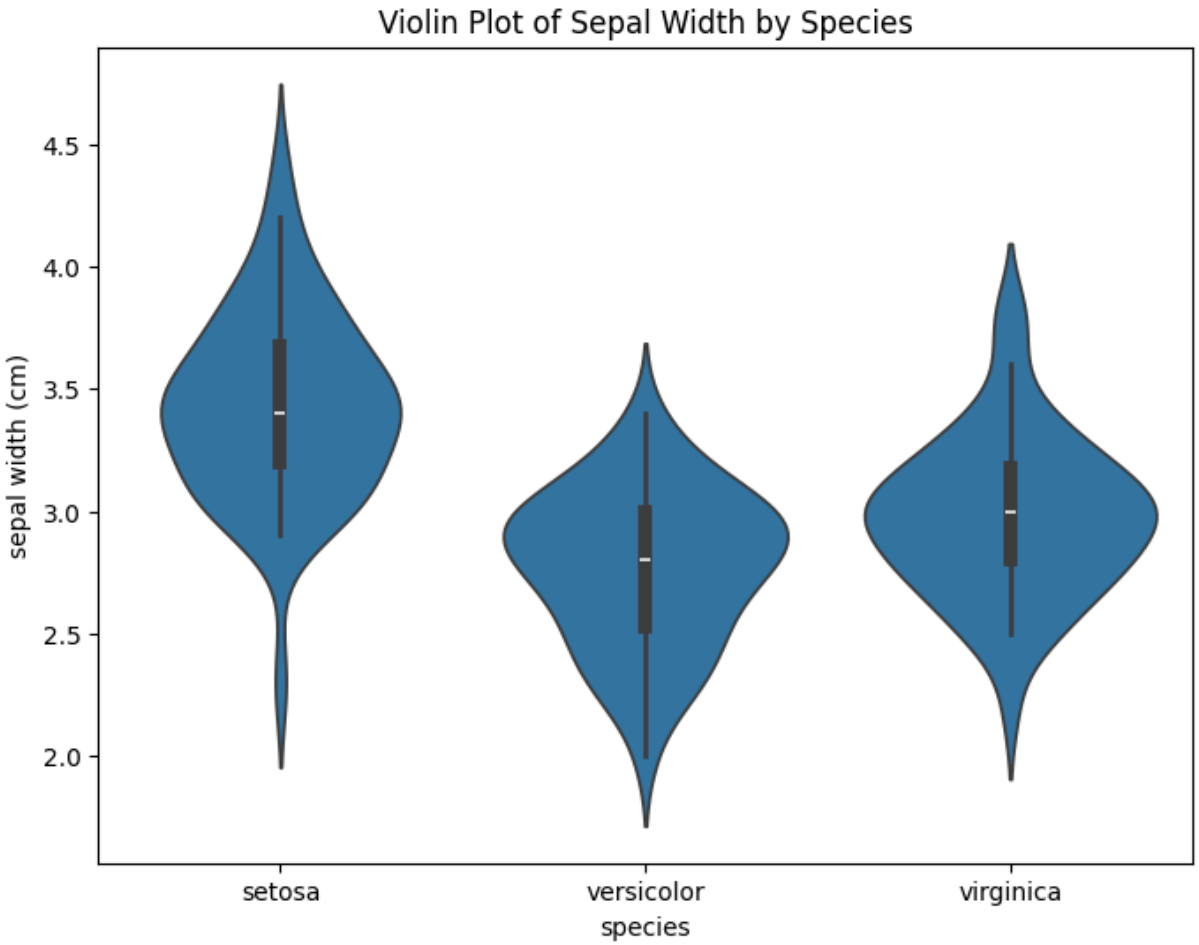


Violin plot for better understanding of distribution and density

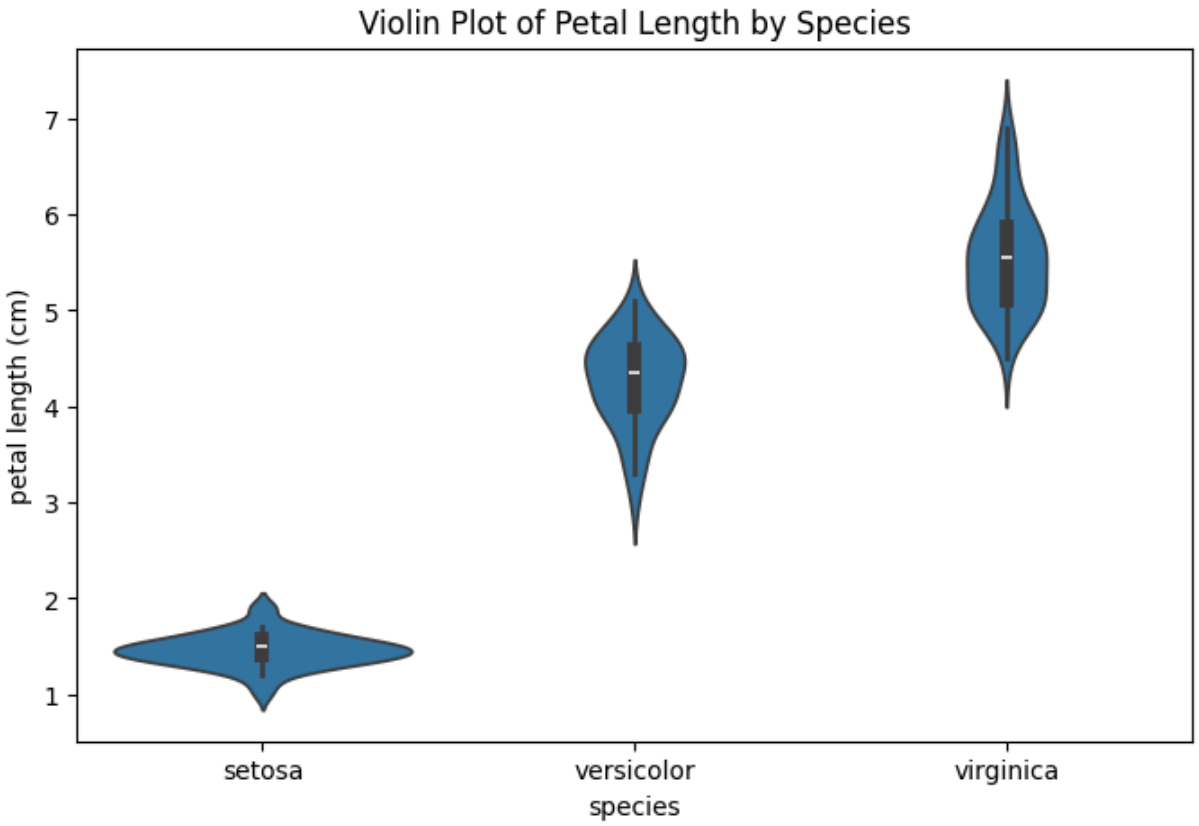
In [29]:
```
plt.figure(figsize=(8,6))
sns.violinplot(x='species', y='sepal length (cm)', data=iris_df)
plt.title("Violin Plot of Sepal Length by Species")
plt.show()
```



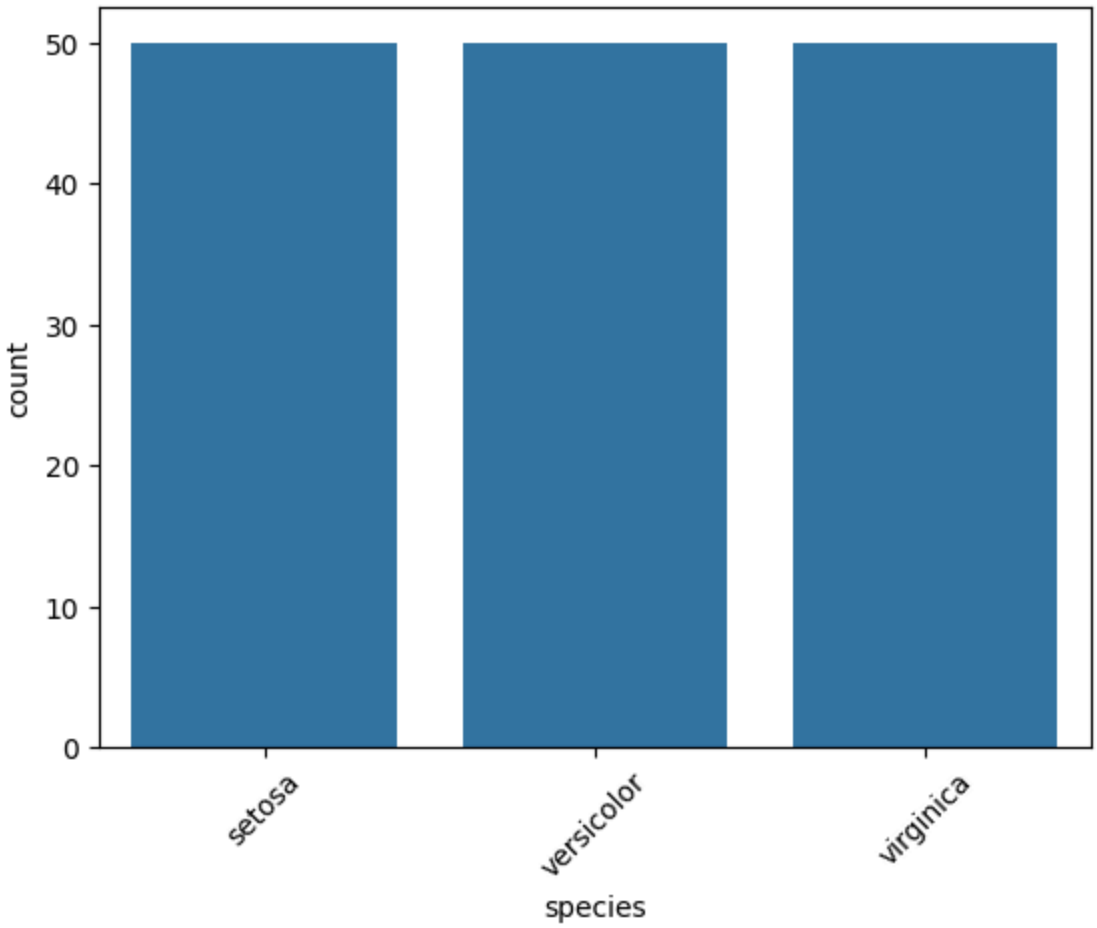
In [27]:
```
plt.figure(figsize=(8, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=iris_df)
plt.title("Violin Plot of Sepal Width by Species")
plt.show()
```



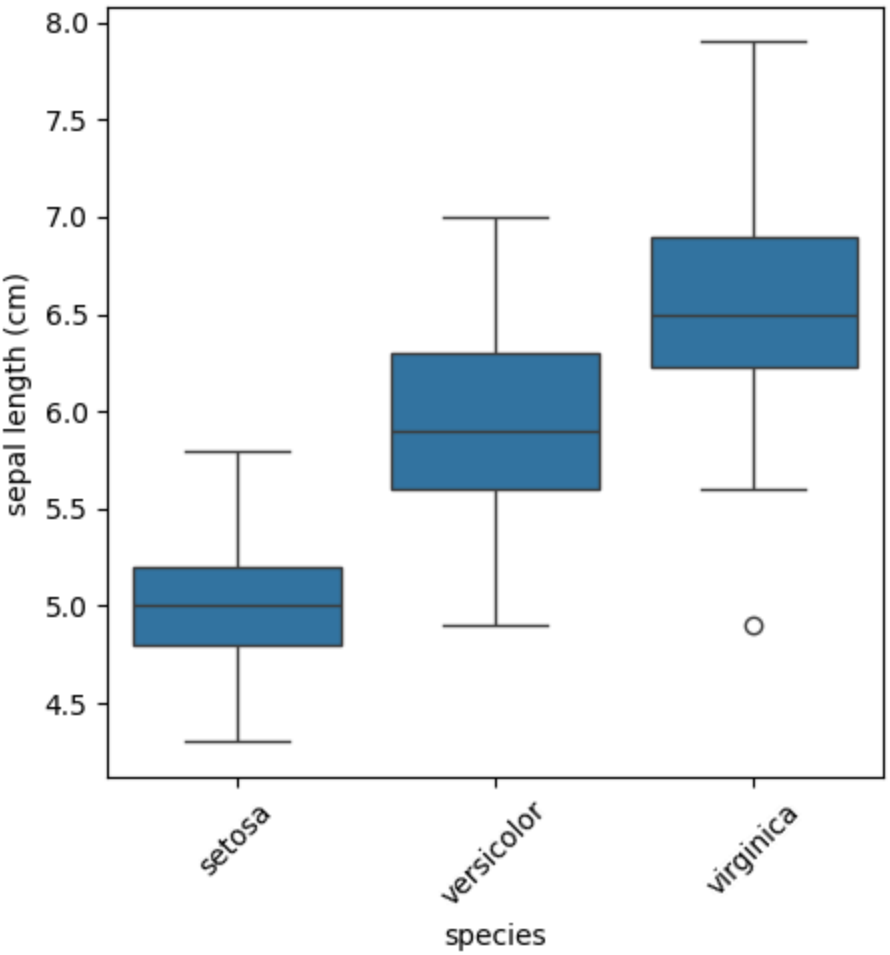
In [28]:
```
plt.figure(figsize=(8, 5))
sns.violinplot(x='species', y='petal length (cm)', data=iris_df)
plt.title("Violin Plot of Petal Length by Species")
plt.show()
```



In [37]:
```
sns.countplot(x='species', data=df)
plt.xticks(rotation=45)
plt.show()
```



In [40]:
```
plt.figure(figsize=(5,5))
sns.boxplot(x='species', y='sepal length (cm)', data=df)
plt.xticks(rotation=45)
plt.show()
```



INSIGHTS

Insights from the Iris Dataset:

1. Setosa species has the smallest petal length and width, which is significantly different from Versicolor and Virginica.

2. Virginica species generally has larger sepal and petal dimensions compared to Setosa and Versicolor

4. There is a strong positive correlation between petal length and petal width, indicating that as petal length increases, petal width also increases

6. Sepal width shows a weak correlation with other features, suggesting it might not be a strong predictor of species on its own.

Expected Output

Summary Statistics: A table displaying basic statistics (mean, std, min, max, etc.) of the features.

Histograms: Visual representations of the distributions of the numerical features.

Boxplots: Visualizations showing the spread of data and potential outliers.

Scatter Plots: Charts showing the relationships between pairs of features.

Correlation Heatmap: A heatmap illustrating the strength of the relationships between numerical features.

Bar Plots and Boxplots for Categorical Data: Visualizations displaying the distribution of species and how numerical features vary across them.

CONCLUSION

The **Iris Dataset** offers valuable insights into the characteristics of iris flowers, providing a clear example of how data science techniques can be applied to explore and visualize data. Through EDA, we identified the distribution of features, correlations between them, and visualized outliers. This analysis provides a deeper understanding of the dataset, which is essential for building machine learning models for classification or clustering tasks.

The findings of this EDA can also serve as a foundation for further analysis or modeling, such as applying classification algorithms to predict species based on the flower's attributes.

THIS IS MY PROJECT CREATED BY - SAYEEDA MISBAH ILYAS

Github account for more infomation : https://github.com/codemisba

Linkdin account : https://www.linkedin.com/in/sayeeda-misbah-ilyas/