Visualizing Data in Python

When working with a new dataset, one of the most useful things to do is to begin to visualize the data. By using **tables**, **histograms**, **boxplots**, **scatter plots** and other visual tools, we can get a better idea of what the data may be trying to tell us, and we can gain insights into the data that we may have not discovered otherwise.

In this notebook will use the Seaborn data processing library, which is a higher-level interface to **Matplotlib** that can be used to simplify many visualization tasks

The **Seaborn** provides visualisations tools that will allow to explore data from a graphical perspective.

Acknowledgments

- Data from https://www.coursera.org/ from the course "Understanding and Visualizing Data with Python" by University of Michigan

In [ ]:

Importing libraries

In [40]:
```
# Import the packages that we will be using
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing data

In [41]:
```
from sklearn import datasets
iris = datasets.load_iris()
```

In [42]:
```
# Define the col names for the iris dataset
col_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'Flower']

# Dataset url
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Load the dataset from URL
iris_df = pd.read_csv(url, header=None, names=col_names)
print(iris_df.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width       Flower
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

In [ ]:

Exploring the content of the data set

Get a general 'feel' of the data

In [43]:
```
# Display the first few rows of the dataframe
print(iris_df.head())

# Get summary statistics of the dataframe
print(iris_df.describe(include='all'))

# Check for missing values
print(iris_df.isnull().sum())
```

```
   sepal_length  sepal_width  petal_length  petal_width       Flower
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
        sepal_length  sepal_width  petal_length  petal_width       Flower
count     150.000000   150.000000    150.000000   150.000000          150
unique           NaN          NaN           NaN          NaN            3
top              NaN          NaN           NaN          NaN  Iris-setosa
freq             NaN          NaN           NaN          NaN           50
mean        5.843333     3.054000      3.758667     1.198667          NaN
std         0.828066     0.433594      1.764420     0.763161          NaN
min         4.300000     2.000000      1.000000     0.100000          NaN
25%         5.100000     2.800000      1.600000     0.300000          NaN
50%         5.800000     3.000000      4.350000     1.300000          NaN
75%         6.400000     3.300000      5.100000     1.800000          NaN
max         7.900000     4.400000      6.900000     2.500000          NaN
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
Flower          0
dtype: int64
```

Frequency tables

The `value_counts()` method can be used to determine the number of times that each distinct value of a variable occurs in a data set. In statistical terms, this is the "frequency distribution" of the variable. The `value_counts()` method produces a table with two columns. The first column contains all distinct observed values for the variable. The second

column contains the number of times each of these values occurs. Note that the table returned by `value_counts()` is actually a **Pandas** data frame, so can be further processed using any Pandas methods for working with data frames.

In [44]:
```
# Number of times that each distinct value occurs in each column of the iris_df DataFr
print(iris_df['Flower'].value_counts())
```

```
Flower
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [ ]:
```

```

In [45]:
```
# Proportion of each distinct value of 'Flowers' in the data set

flower_proportions = iris_df['Flower'].value_counts(normalize=True)
print(flower_proportions)
```

```
Flower
Iris-setosa        0.333333
Iris-versicolor    0.333333
Iris-virginica     0.333333
Name: proportion, dtype: float64
```

Note that the `value_counts()` method excludes missing values. We confirm this below by adding up observations to your data frame with some missing values and then computing `value_counts()` and comparing this to the total number of rows in the data set, which is 28. This tells us that there are 28 - (21+6) = 1 missing values for this variable (other variables may have different numbers of missing values).

In [46]:
```
total_observations = iris_df.shape[0]
# total number of null observations in Age
null_observations_age = iris_df['sepal_length'].isnull().sum()
print(f'Total number of null observations in sepal_length: {null_observations_age}')

# Total number of counts in Age (excluding missing values)
total_counts_age = iris_df['sepal_length'].count()
print(f'Total number of counts in sepal_length (excluding missing values): {total_cour

print(f'Total number of observations: {total_observations}')
```

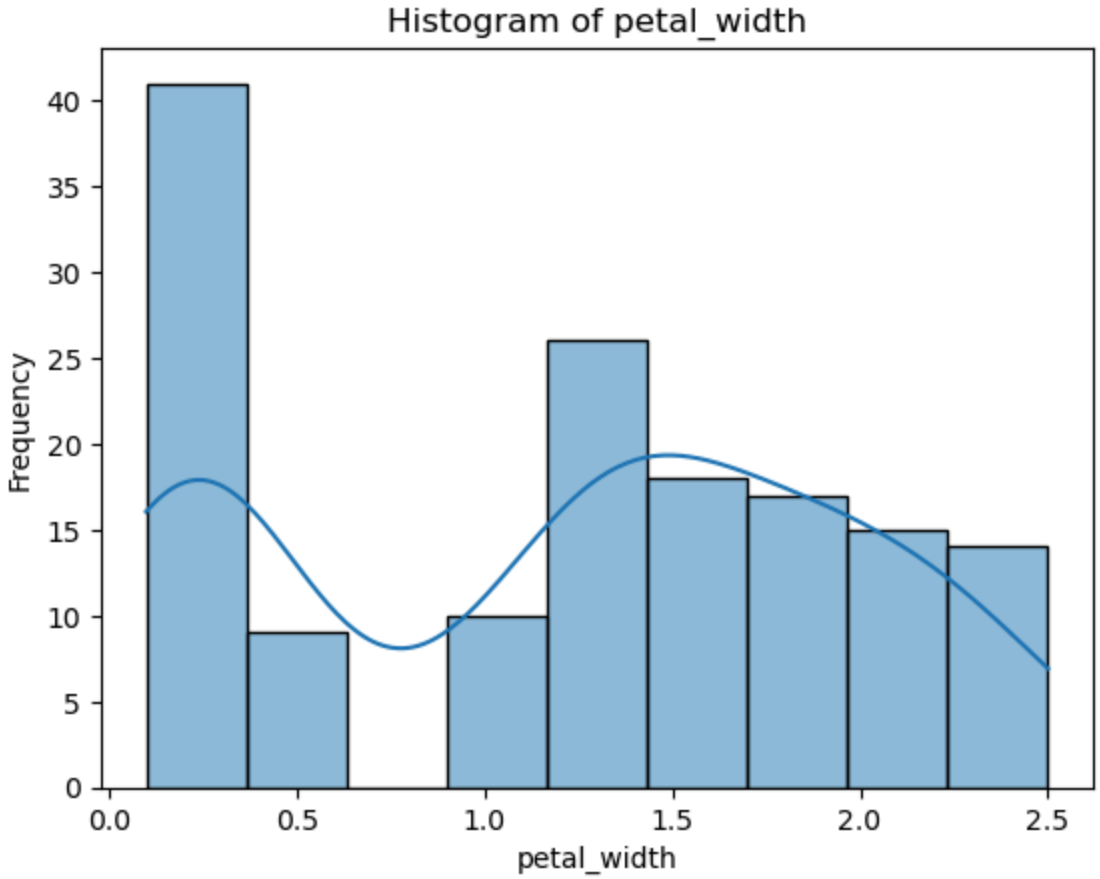
```
Total number of null observations in sepal_length: 0
Total number of counts in sepal_length (excluding missing values): 150
Total number of observations: 150
```

Histogram

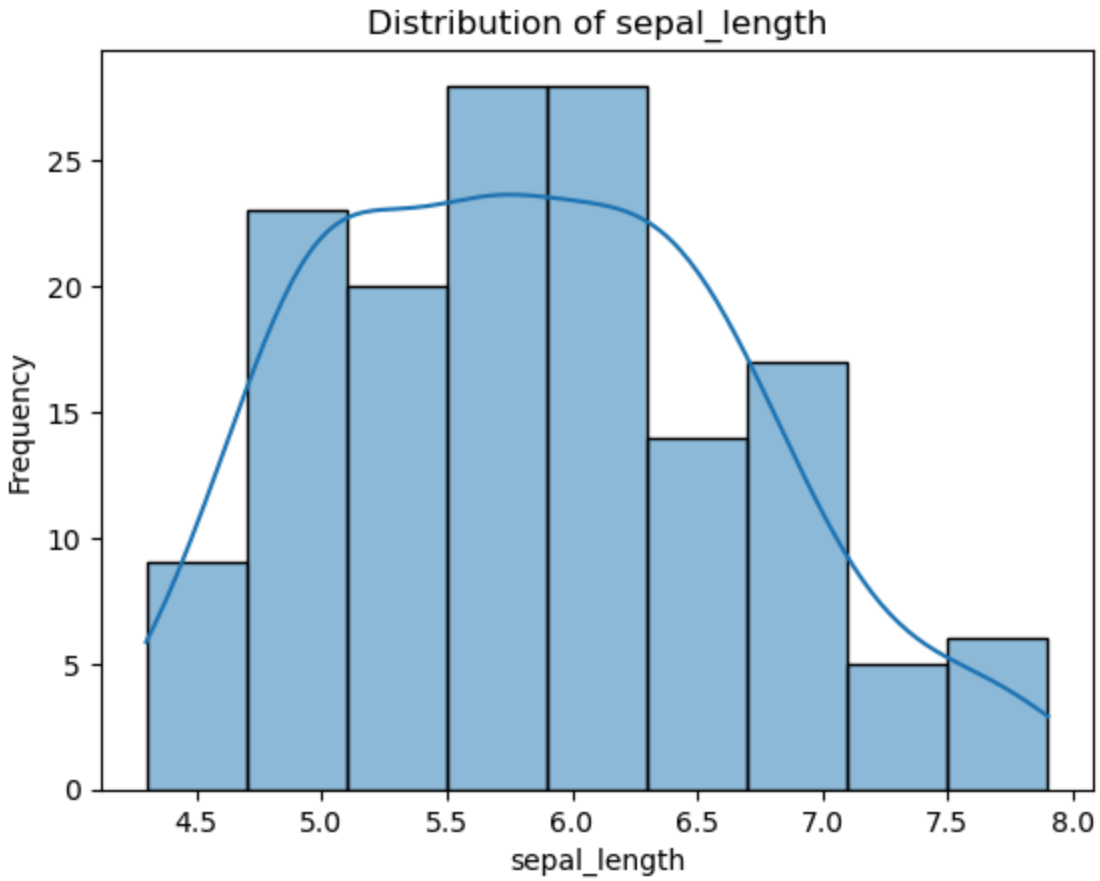
It is often good to get a feel for the shape of the distribution of the data.

In [47]:
```
# Plot histogram for the petal_width column
sns.histplot(iris_df['petal_width'], kde=True)
```

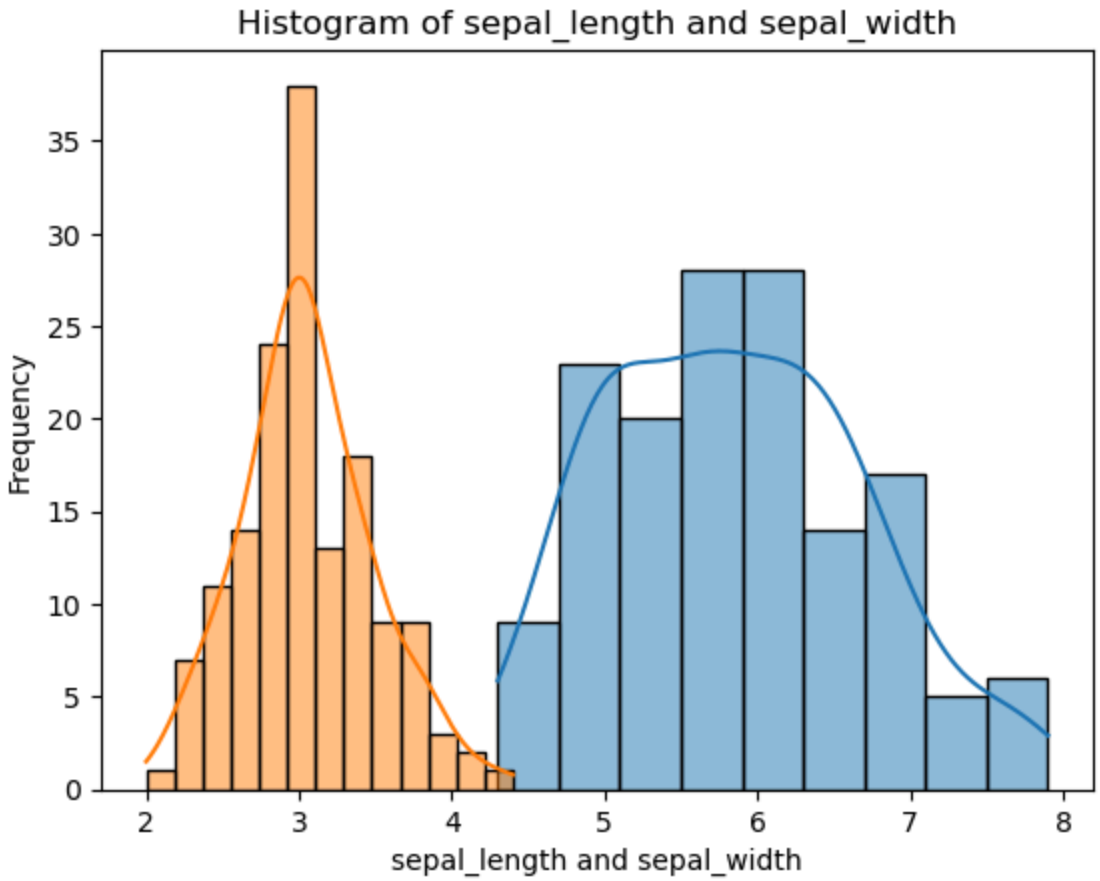
```
plt.xlabel('petal_width')
plt.ylabel('Frequency')
plt.title('Histogram of petal_width')
plt.show()
```



In [48]:
```
# Plot distribution of the sepal_length column
sns.histplot(iris_df['sepal_length'], kde=True)
plt.xlabel('sepal_length')
plt.ylabel('Frequency')
plt.title('Distribution of sepal_length')
plt.show()
```



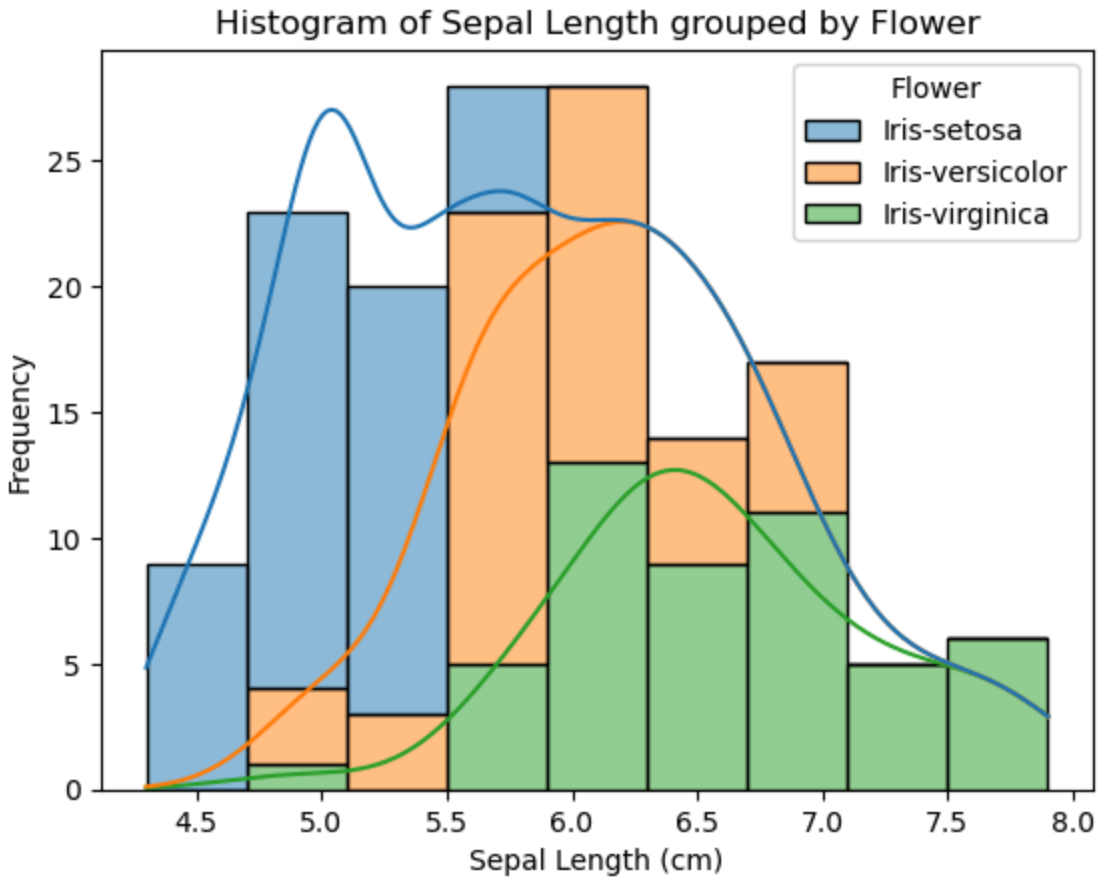
In [49]:
```
# Plot histogram of both the sepal_length and sepal_width columns
sns.histplot(iris_df['sepal_length'], kde=True)
sns.histplot(iris_df['sepal_width'], kde=True)
plt.xlabel('sepal_length and sepal_width')
plt.ylabel('Frequency')
plt.title('Histogram of sepal_length and sepal_width')
plt.show()
```



Histograms plotted by groups

While looking at a single variable is interesting, it is often useful to see how a variable changes in response to another. Thus, we can create a histograms of one quantitative variable grouped by another categorical variables.

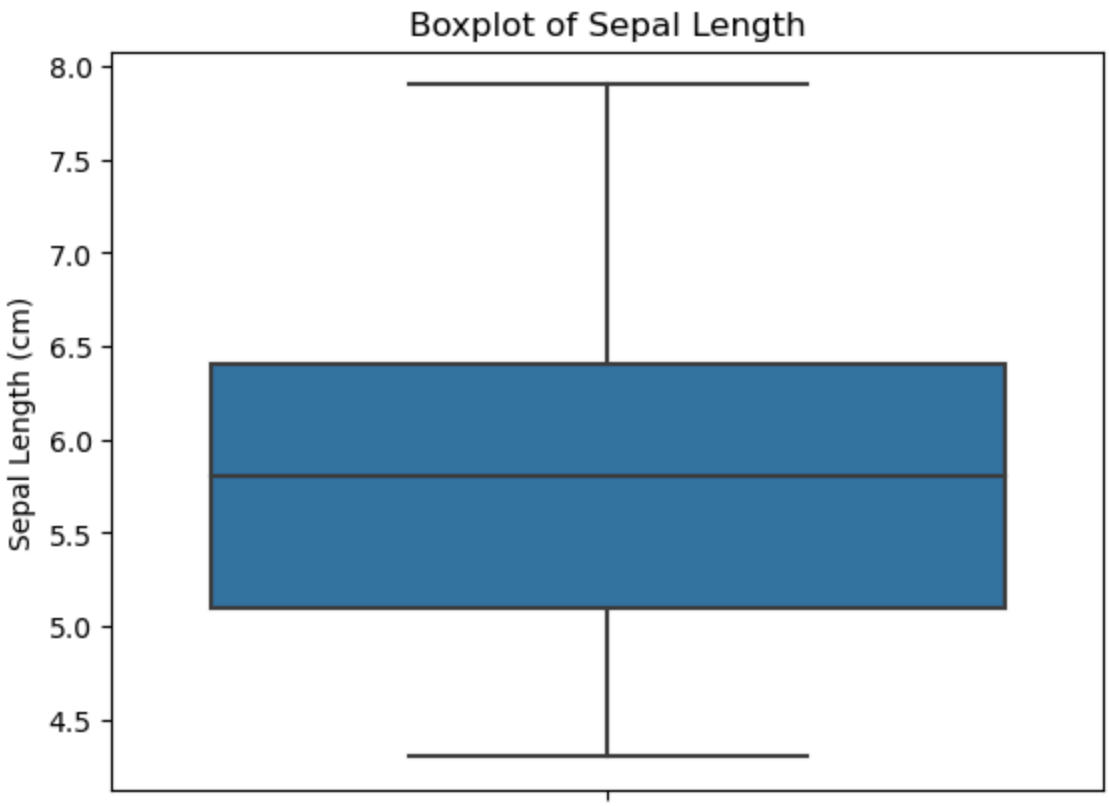
In [50]:
```
# Create histograms of the "sepal_length" grouped by "Flower"
sns.histplot(data=iris_df, x='sepal_length', hue='Flower', multiple='stack', kde=True)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.title('Histogram of Sepal Length grouped by Flower')
plt.show()
```



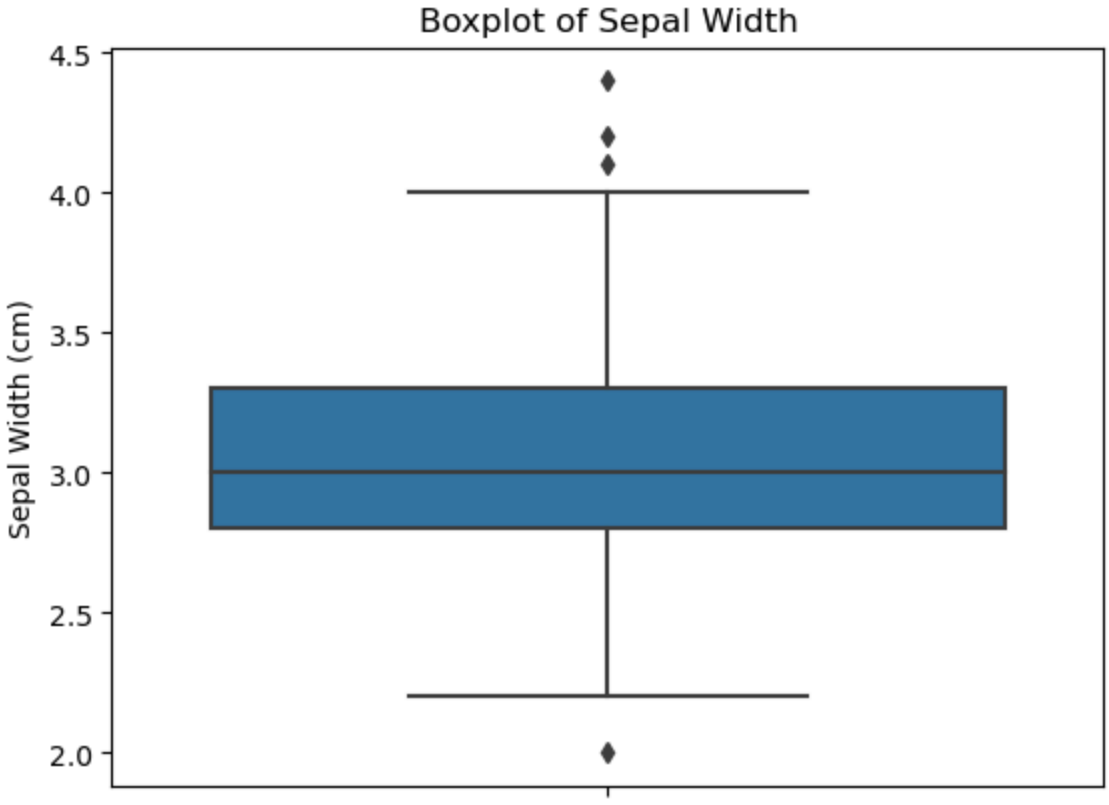
Boxplots

Boxplots do not show the shape of the distribution, but they can give us a better idea about the center and spread of the distribution as well as any potential outliers that may exist. Boxplots and Histograms often complement each other and help an analyst get more information about the data

In [51]:
```
sns.boxplot(data=iris_df, y='sepal_length')
plt.ylabel('Sepal Length (cm)')
plt.title('Boxplot of Sepal Length')
plt.show()
```

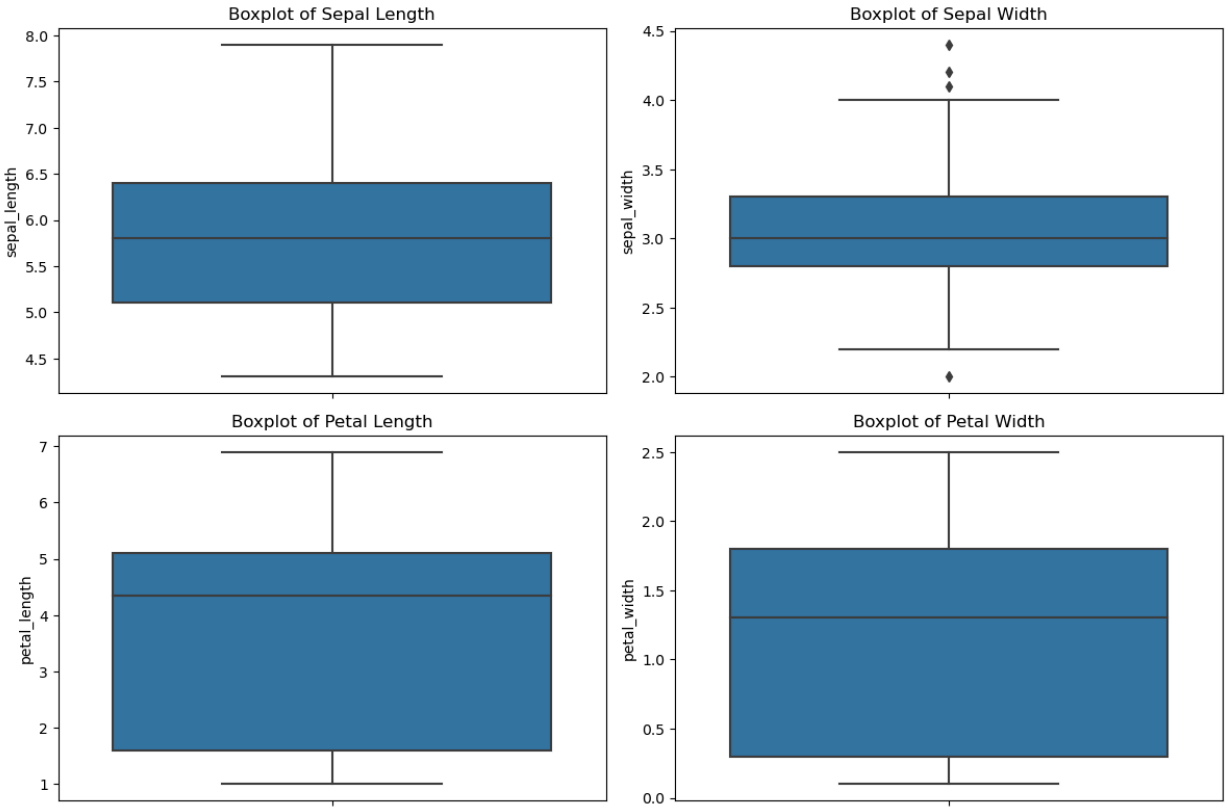


In [52]:
```
sns.boxplot(data=iris_df, y='sepal_width')
plt.ylabel('Sepal Width (cm)')
plt.title('Boxplot of Sepal Width')
plt.show()
```



In [53]:
```
plt.figure(figsize=(12, 8))
```

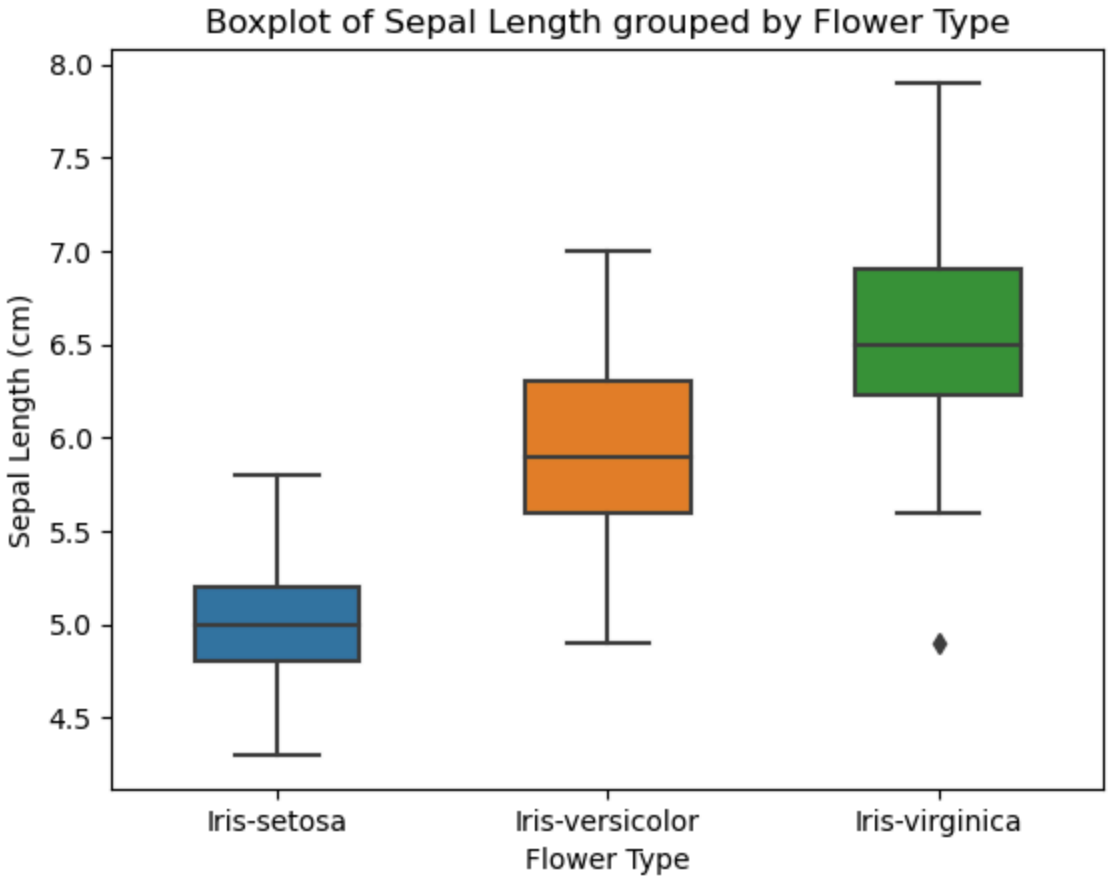
```
# Boxplot for sepal length
plt.subplot(2, 2, 1)
sns.boxplot(data=iris_df, y='sepal_length')
plt.title('Boxplot of Sepal Length')

# Boxplot for sepal width
plt.subplot(2, 2, 2)
sns.boxplot(data=iris_df, y='sepal_width')
plt.title('Boxplot of Sepal Width')

# Boxplot for petal length
plt.subplot(2, 2, 3)
sns.boxplot(data=iris_df, y='petal_length')
plt.title('Boxplot of Petal Length')

# Boxplot for petal width
plt.subplot(2, 2, 4)
sns.boxplot(data=iris_df, y='petal_width')
plt.title('Boxplot of Petal Width')

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



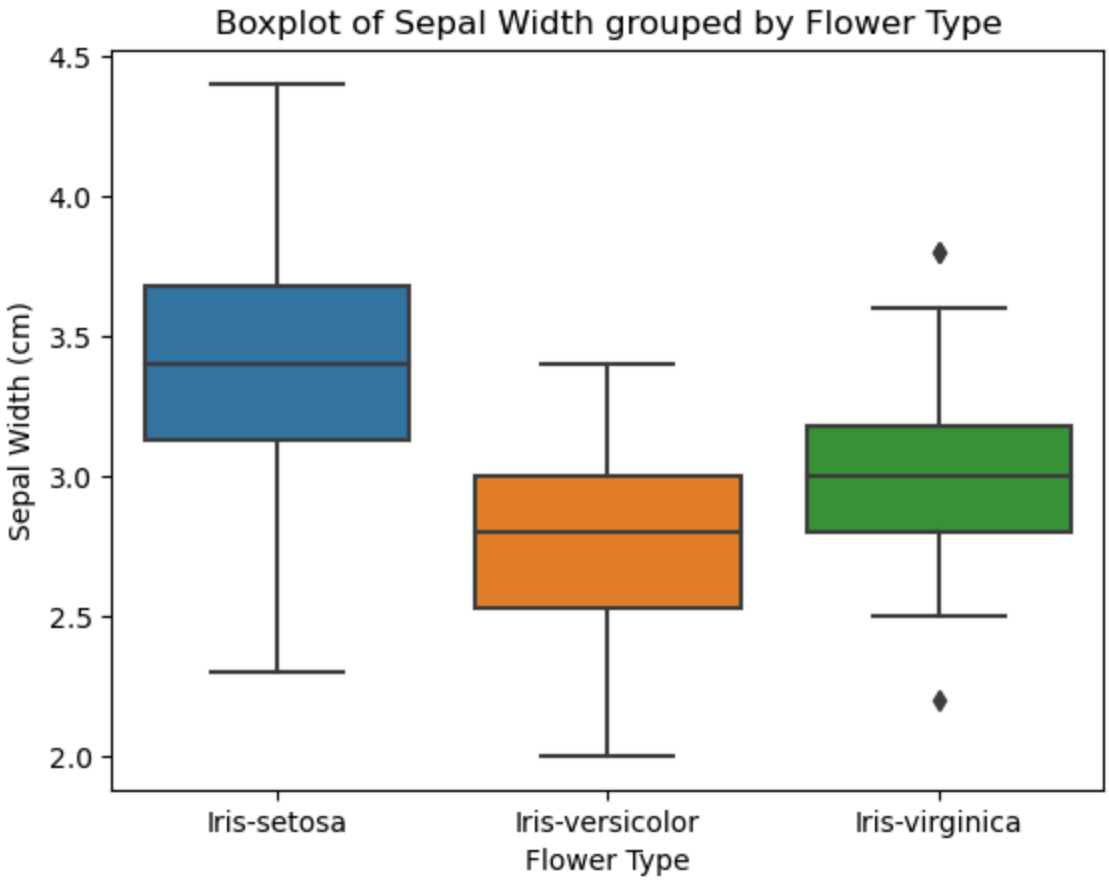
Boxplots plotted by groups

While looking at a single variable is interesting, it is often useful to see how a variable changes in response to another. Thus, we can create a side-by-side boxplots of one quantitative variable grouped by another categorical variables.

In [54]:
```
# Create side-by-side boxplots of the "sepal_length" grouped by "Flower"
sns.boxplot(data=iris_df, x='Flower', y='sepal_length', width=0.5, dodge=True)
plt.xlabel('Flower Type')
plt.ylabel('Sepal Length (cm)')
plt.title('Boxplot of Sepal Length grouped by Flower Type')
plt.show()
```



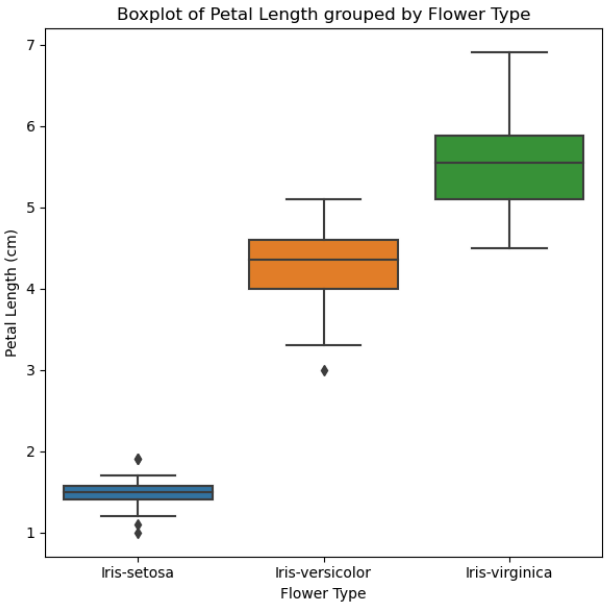
In [55]:
```
# Create side-by-side boxplots of the "sepal_width" grouped by "Flower"
sns.boxplot(data=iris_df, x='Flower', y='sepal_width')
plt.xlabel('Flower Type')
plt.ylabel('Sepal Width (cm)')
plt.title('Boxplot of Sepal Width grouped by Flower Type')
plt.show()
```

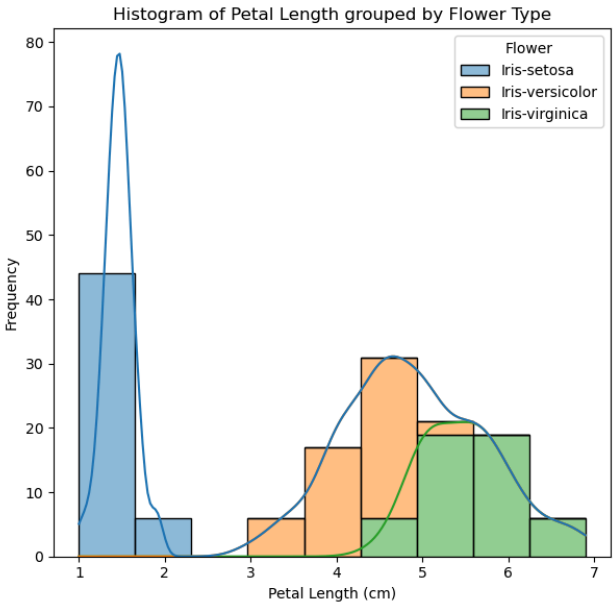


Histograms and boxplots plotted by groups

We cal also create both boxplots and histograms of one quantitative variable grouped by another categorical variables

In [56]:
```
# Create a boxplot and histogram of the "petal_length" grouped by "Flower"
# Boxplot of the petal_length grouped by Flower
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(data=iris_df, x='Flower', y='petal_length')
plt.xlabel('Flower Type')
plt.ylabel('Petal Length (cm)')
plt.title('Boxplot of Petal Length grouped by Flower Type')

# Histogram of the petal_length grouped by Flower
plt.subplot(1, 2, 2)
sns.histplot(data=iris_df, x='petal_length', hue='Flower', multiple='stack', kde=True)
plt.xlabel('Petal Length (cm)')
plt.ylabel('Frequency')
plt.title('Histogram of Petal Length grouped by Flower Type')

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

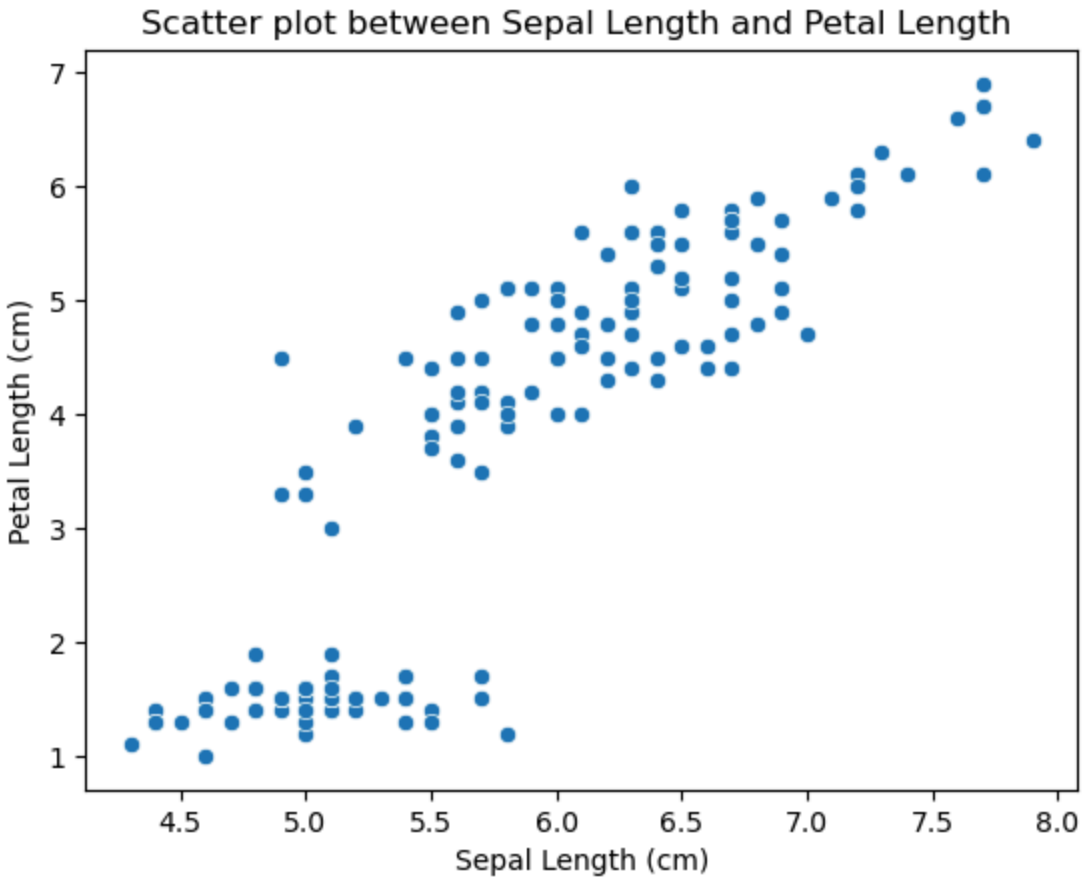




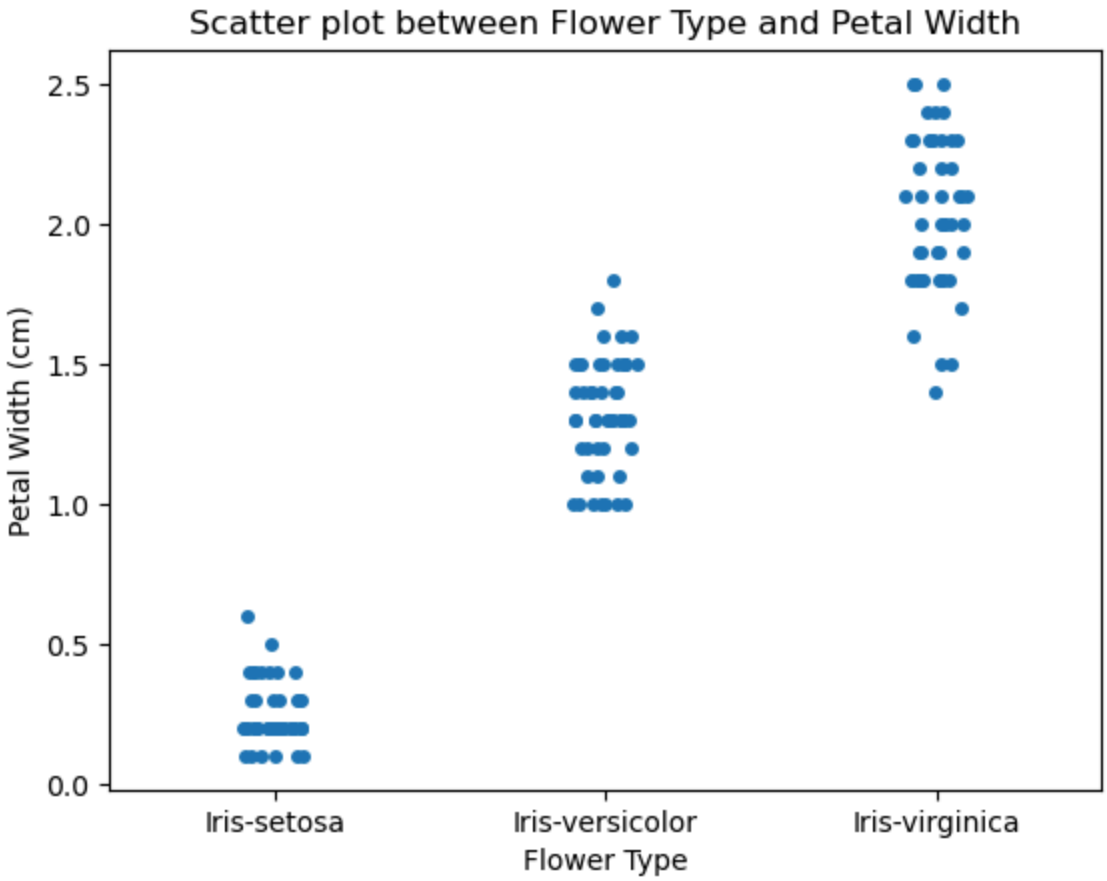
Scatter plot

Plot values of one variable versus another variable to see how they are correlated

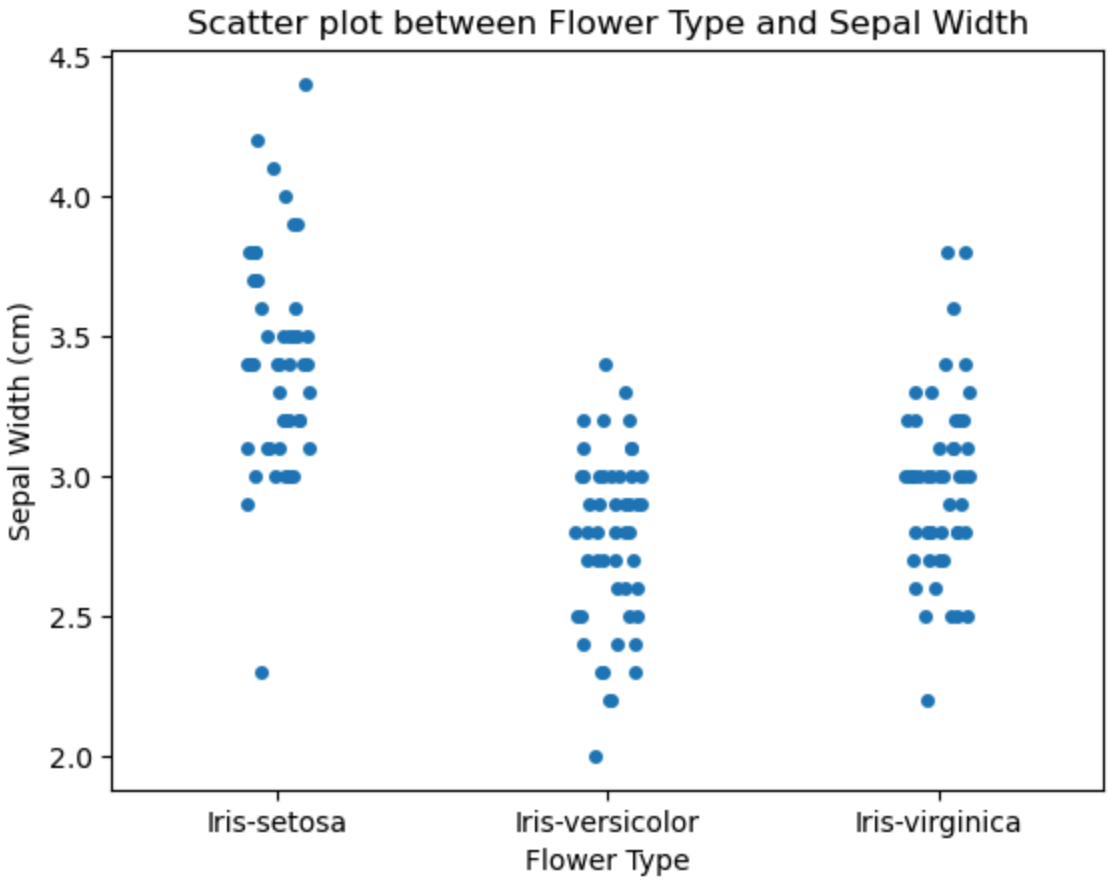
In [61]:
```
# Scatter plot between sepal_length and petal_length
sns.scatterplot(data=iris_df, x='sepal_length', y='petal_length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.title('Scatter plot between Sepal Length and Petal Length ')
plt.show()
```



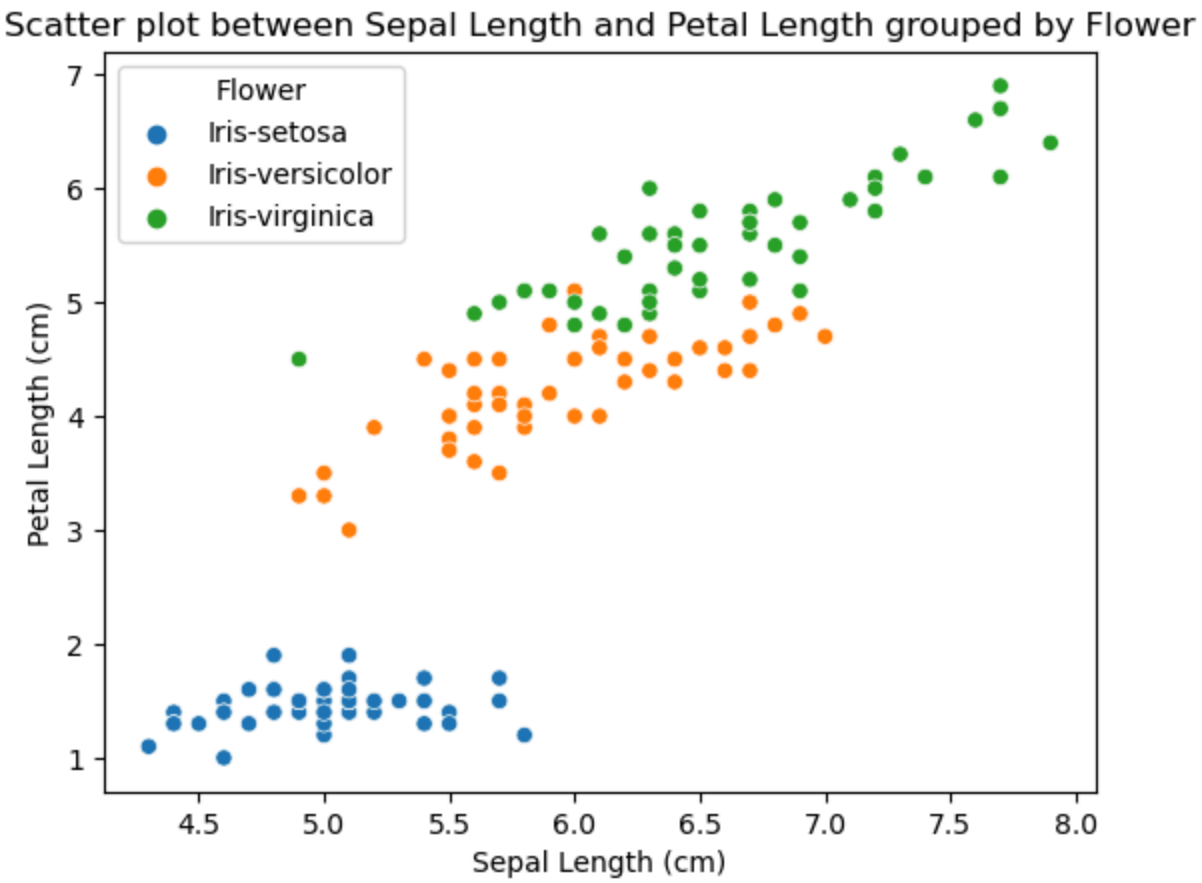
In [59]:
```
# Scatter plot between Flower (categorical) and petal_width (numerical)
sns.stripplot(data=iris_df, x='Flower', y='petal_width', jitter=True)
plt.xlabel('Flower Type')
plt.ylabel('Petal Width (cm)')
plt.title('Scatter plot between Flower Type and Petal Width')
plt.show()
```



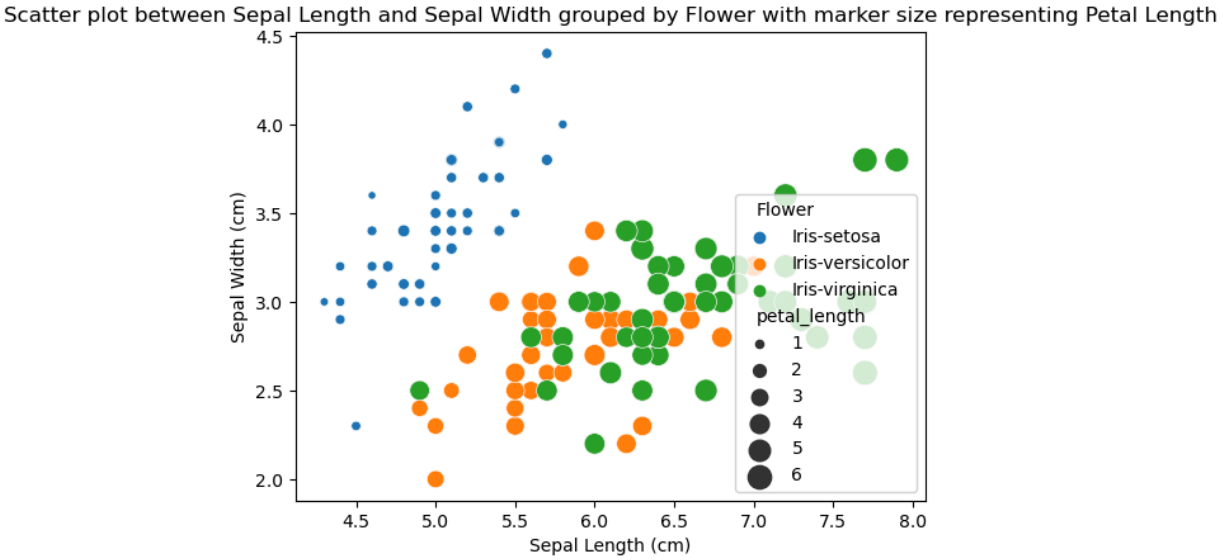
In [63]:
```
# Scatter plot between Flower (categorical) and sepal_width (numerical)
sns.stripplot(data=iris_df, x='Flower', y='sepal_width', jitter=True)
plt.xlabel('Flower Type')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter plot between Flower Type and Sepal Width')
plt.show()
```



In [64]:
```
# Scatter plot between sepal_length and petal_length grouped by Flower
sns.scatterplot(data=iris_df, x='sepal_length', y='petal_length', hue='Flower')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.title('Scatter plot between Sepal Length and Petal Length grouped by Flower')
plt.show()
```

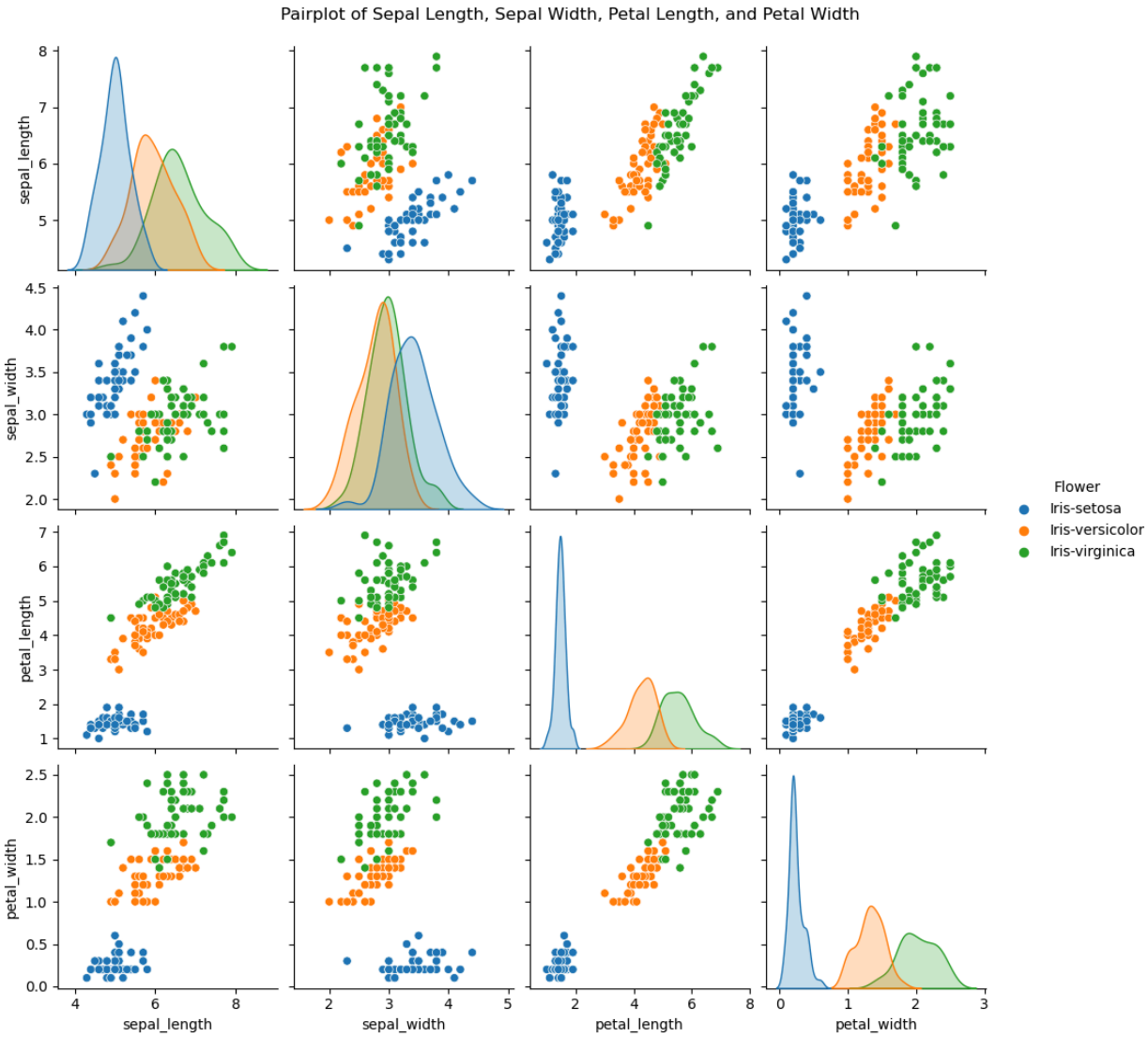


In [65]:
```
# Scatter plot between sepal_length and sepal_width grouped by Flower with marker size
sns.scatterplot(data=iris_df, x='sepal_length', y='sepal_width', hue='Flower', size='p
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter plot between Sepal Length and Sepal Width grouped by Flower with ma
plt.show()
```



In [66]:
```
# Pairplot: Scatterplot of sepal_length, sepal_width, petal_length, petal_width
sns.pairplot(iris_df, vars=['sepal_length', 'sepal_width', 'petal_length', 'petal_widt
plt.suptitle('Pairplot of Sepal Length, Sepal Width, Petal Length, and Petal Width', y
plt.show()
```

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



Final remarks

- Visualizing your data using **tables**, **histograms**, **boxplots**, **scatter plots** and other tools is essential to carry put analysis and extract conclusions
- There are several ways to do the same thing
- The **Seaborn** package provides visualisations tools that allow to explore data from a graphical perspective

Activity: work with the iris dataset

Repeat this tutorial with the iris data set and respond to the following inquiries

1. Plot the histograms for each of the four quantitative variables

1. Plot the histograms for each of the quantitative variables

1. Plot the boxplots for each of the quantitative variables

1. Plot the boxplots of the petal width grouped by type of flower

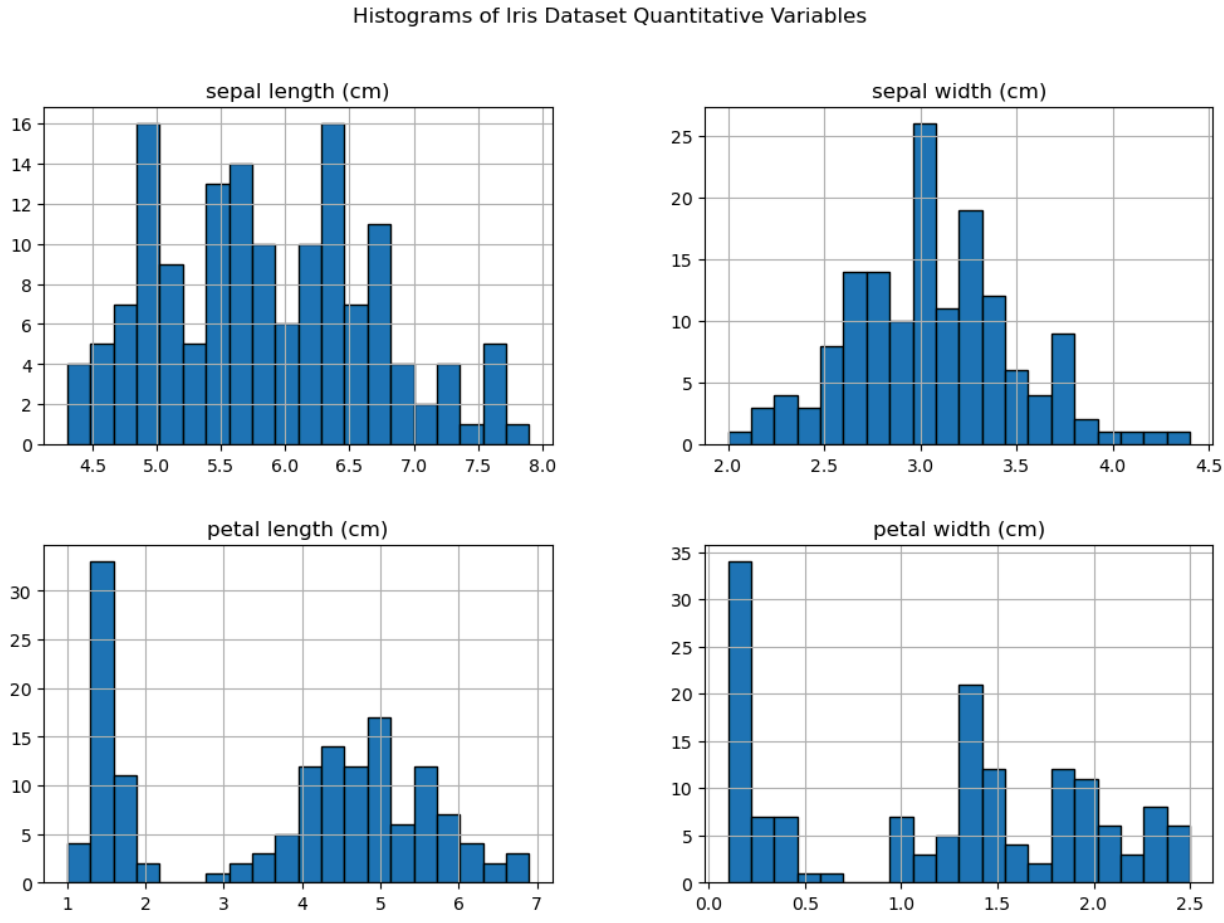
1. Plot the boxplots of the setal length grouped by type of flower

1. Provide a description (explaination from your observations) of each of the quantitative variables

In [67]:
```
from sklearn import datasets
iris = datasets.load_iris()
```

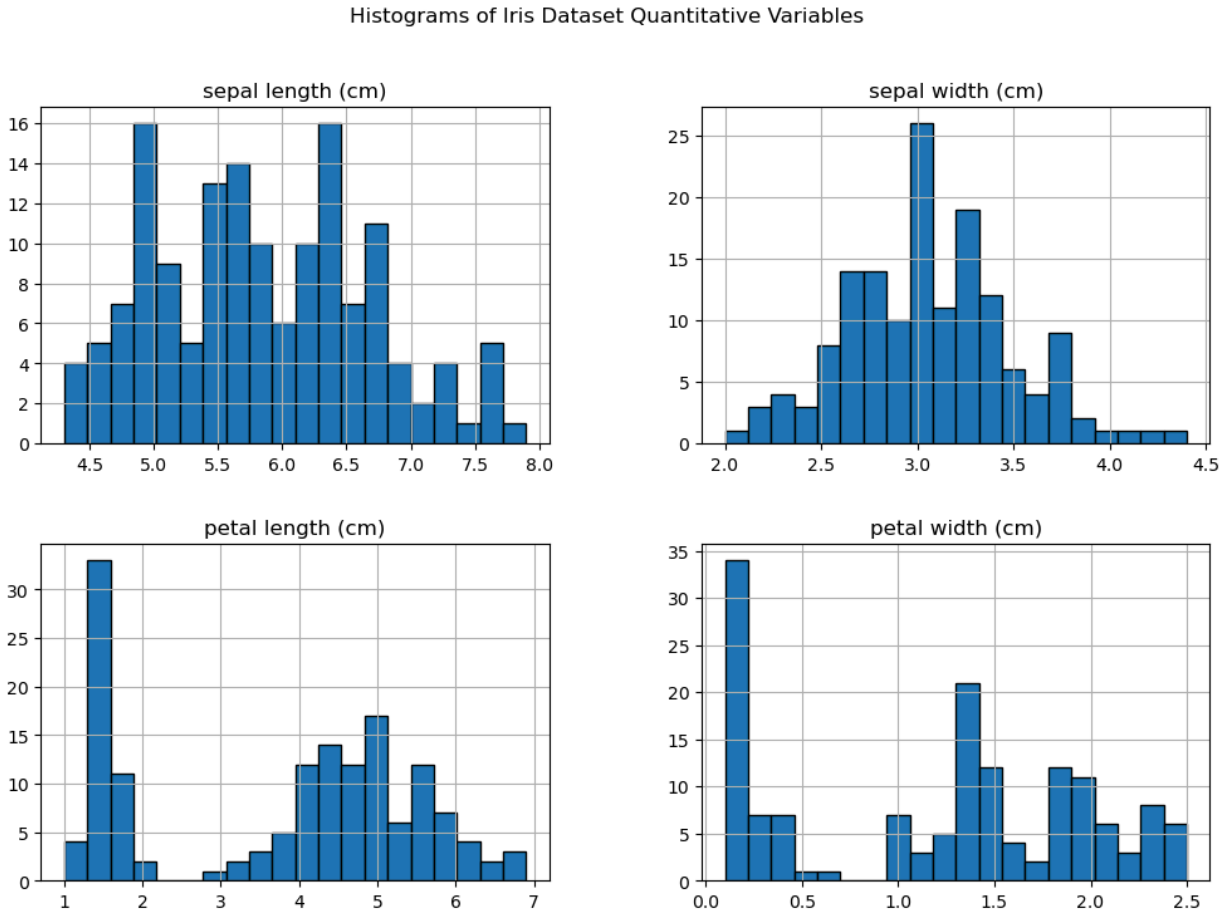
1. Plot the histograms for each of the four quantitative variables

In [68]:
```
# Convert the iris dataset to a pandas DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Plot histograms for each quantitative variable
iris_df.hist(bins=20, figsize=(12, 8), layout=(2, 2), edgecolor='black')
plt.suptitle('Histograms of Iris Dataset Quantitative Variables')
plt.show()
```



1. Plot the histograms for each of the quantitative variables

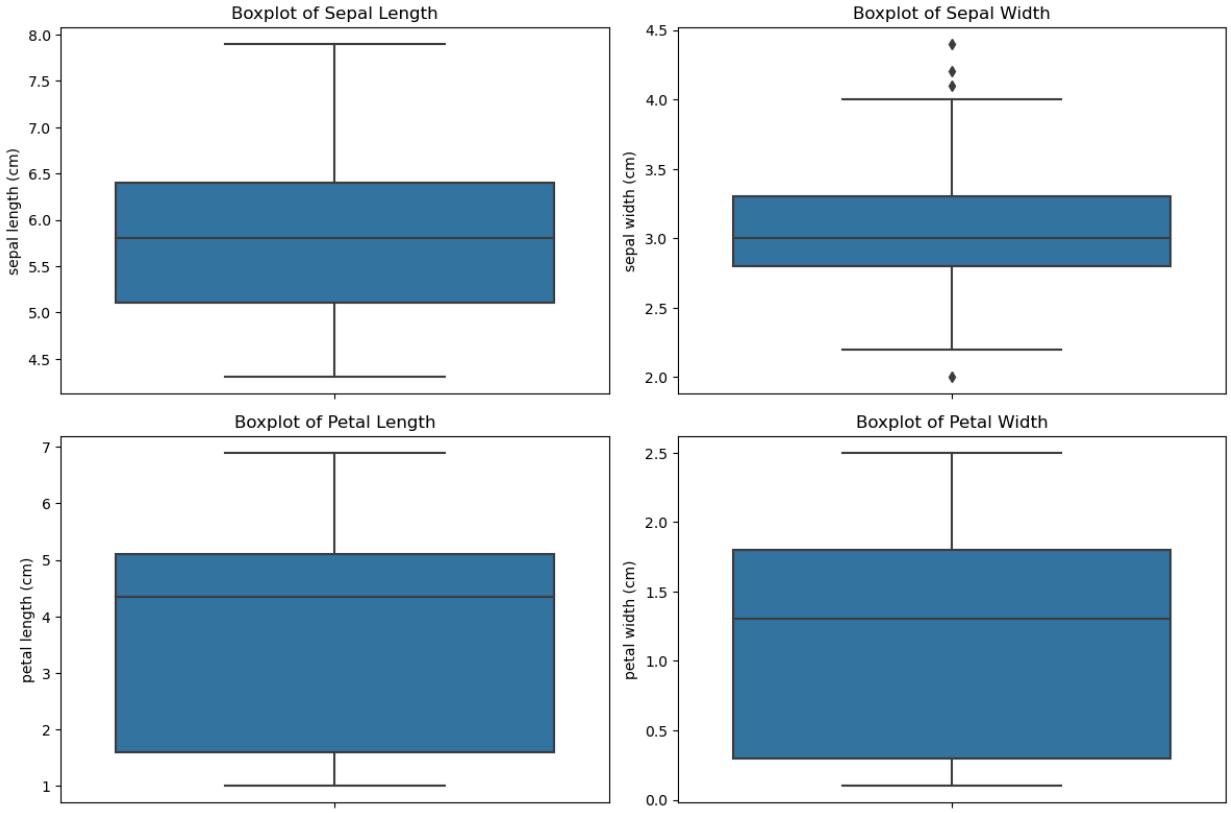
In [69]:
```
# Plot histograms for each quantitative variable
iris_df.hist(bins=20, figsize=(12, 8), layout=(2, 2), edgecolor='black')
plt.suptitle('Histograms of Iris Dataset Quantitative Variables')
plt.show()
```



1. Plot the boxplots for each of the quantitative variables

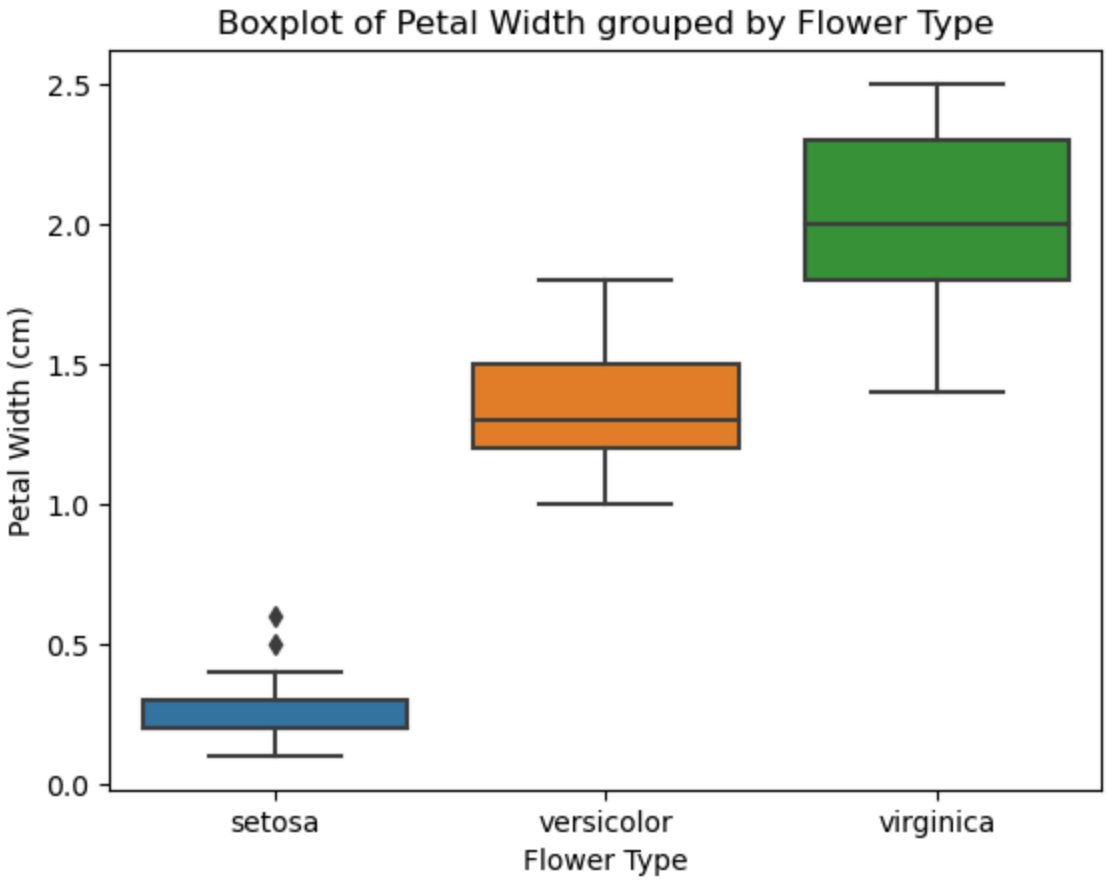
In [70]:
```
# Plot boxplots for each quantitative variable
plt.figure(figsize=(12, 8))

# Boxplot for sepal length
plt.subplot(2, 2, 1)
sns.boxplot(data=iris_df, y='sepal length (cm)')
plt.title('Boxplot of Sepal Length')

# Boxplot for sepal width
plt.subplot(2, 2, 2)
sns.boxplot(data=iris_df, y='sepal width (cm)')
plt.title('Boxplot of Sepal Width')

# Boxplot for petal length
plt.subplot(2, 2, 3)
sns.boxplot(data=iris_df, y='petal length (cm)')
plt.title('Boxplot of Petal Length')
```

```
# Boxplot for petal width
plt.subplot(2, 2, 4)
sns.boxplot(data=iris_df, y='petal width (cm)')
plt.title('Boxplot of Petal Width')

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



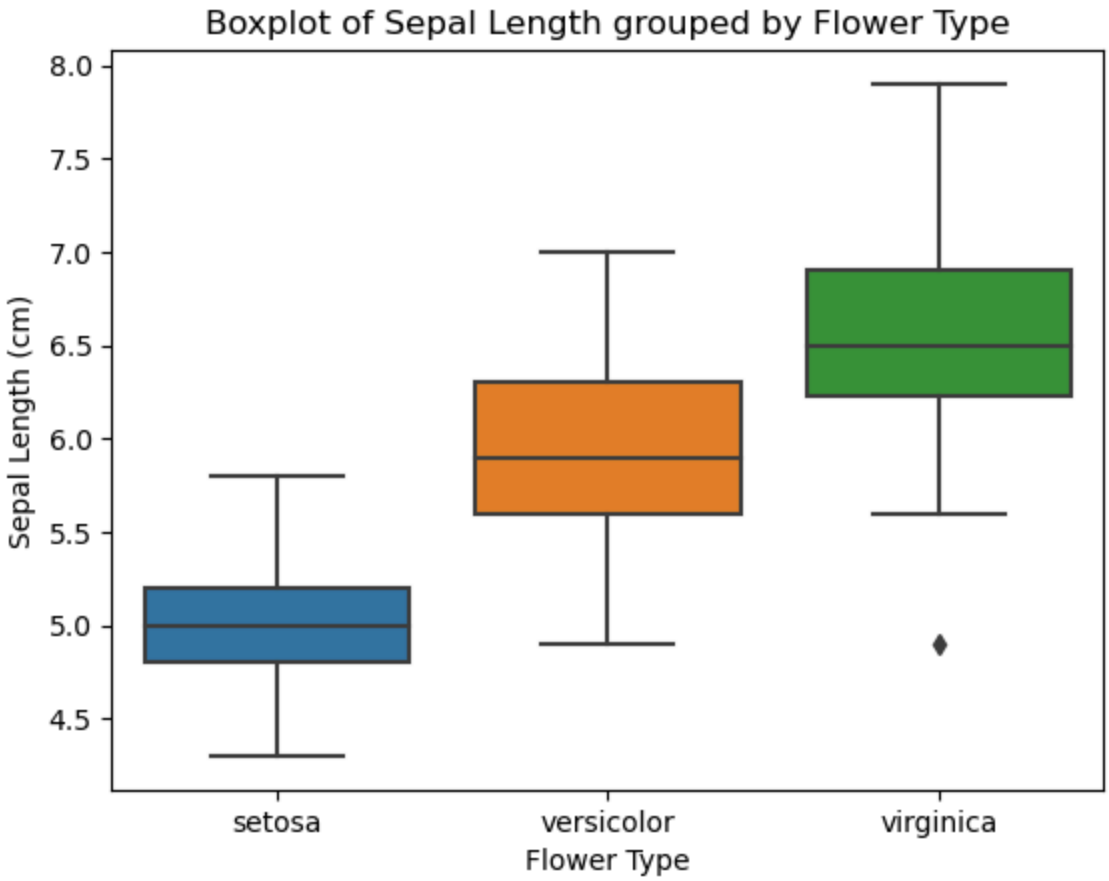
1. Plot the boxplots of the petal width grouped by type of flower

In [71]:
```
# Add the target variable to the DataFrame
iris_df['flower_type'] = iris.target

# Map the target variable to the flower names
iris_df['flower_type'] = iris_df['flower_type'].map({0: 'setosa', 1: 'versicolor', 2:

# Plot the boxplots of the petal width grouped by type of flower
sns.boxplot(data=iris_df, x='flower_type', y='petal width (cm)')
plt.xlabel('Flower Type')
plt.ylabel('Petal Width (cm)')
plt.title('Boxplot of Petal Width grouped by Flower Type')
plt.show()
```



1. Plot the boxplots of the setal length grouped by type of flower

In [7]:
```
# Plot the boxplots of the sepal length grouped by type of flower
sns.boxplot(data=iris_df, x='flower_type', y='sepal length (cm)')
plt.xlabel('Flower Type')
plt.ylabel('Sepal Length (cm)')
plt.title('Boxplot of Sepal Length grouped by Flower Type')
plt.show()
```



1. Provide a description (explaination from your observations) of each of the quantitative variables

The Iris dataset consists of four quantitative variables: sepal length, sepal width, petal length, and petal width, all measured in centimeters. Sepal length ranges from 4.3 to 7.9 cm, with `Iris-setosa` generally having shorter sepals and `Iris-virginica` the longest. Sepal width ranges from 2.0 to 4.4 cm, where `Iris-setosa` tends to have wider sepals, while `Iris-virginica` has narrower ones. Petal length, ranging from 1.0 to 6.9 cm, and petal width, from 0.1 to 2.5 cm, are the most distinctive features, clearly separating the species. `Iris-setosa` has the smallest petals, while `Iris-virginica` has the largest, with `Iris-versicolor` falling between them, making these variables essential for classification.