Matplotlib

- **Low-level control:**Provides granular control over every aspect of the plot (size, axes, ticks, titles, etc.).
- **Versatile plotting:** Supports various plot types like line plots, bar charts, histograms, scatter plots, pie charts, and more.
- **Customizability:** You can customize every detail of the plot, from colors to font sizes and figure layout.
- **2D and 3D plotting:** Can generate both 2D and 3D plots.
- **Static and interactive plots:** Suitable for static image exports (e.g., PNG, PDF) and interactive plots in environments like Jupyter Notebooks.
- **Multiple output formats:** Matplotlib can save figures in many formats such as PNG, JPG, SVG, and PDF.
- **Annotations:** Allows you to add text, arrows, and other annotations to your plots for clarity.
- **Subplots:** Supports arranging multiple plots in one figure using subplots().

Seaborn

- **Built on Matplotlib:** Seaborn simplifies complex visualizations and enhances Matplotlib's capabilities.
- **Works well with Pandas:** Seaborn can automatically handle Pandas DataFrames and perform aggregations.
- **Beautiful default styles:** Predefined themes (e.g., darkgrid, whitegrid) make plots more visually appealing.
- **Simplified syntax:** Functions like pairplot, heatmap, and boxplot allow for easy, quick visualizations.
- **Statistical plots:** Focuses on visualizing statistical relationships (e.g., correlation, distribution).
- **Color palettes:** Comes with various built-in color schemes for better customization.
- **Specialized plots:** Includes pair plots, joint plots, violin plots, and KDE plots.
- **Automatic legend and labels:** Seaborn automatically adds legends and axis labels to the plots based on the dataset.

In [123]:
```
import seaborn as sb
import matplotlib.pyplot as plt
iris=sb.load_dataset('iris')
print(iris.head())
```

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

1. General Statistics Plot (Matplotlib or Seaborn)

The goal of this task is to give a general statistical summary of the Iris dataset. This can be done using:

Seaborn's pairplot:

It visualizes pairwise relationships between variables (like sepal length, sepal width, petal length, petal width), showing scatter plots and histograms.

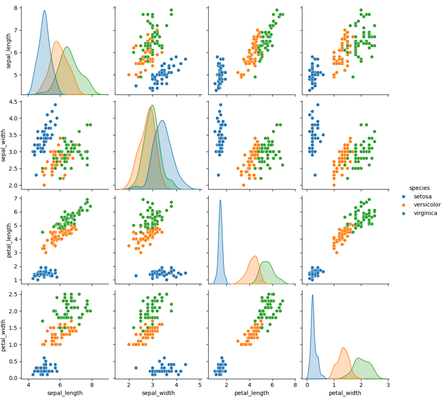
Pandas' describe() function:

It provides a statistical summary of the data, including count, mean, min, max, standard deviation, and quartiles for each numeric feature.

Purpose:

To understand the overall distribution and relationships among features of the Iris dataset.

In [21]:
```
#General statistical summary using seaborn's pairplot
sb.pairplot(iris, hue='species')
plt.show()
# Alternatively, to use Pandas' describe()
print(iris.describe())
```




```
       sepal_length  sepal_width  petal_length  petal_width
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
```

Pie Plot for Species Frequency:

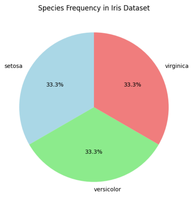
This task is to create a pie chart to display the frequency of the three species in the Iris dataset: Setosa, Versicolor, and Virginica.

Pie charts are circular graphs divided into sectors, where each sector represents a proportion of the whole dataset. It helps show the proportion of each species in the dataset.

Purpose:

To easily visualize how the three species are distributed in the dataset.

In [31]:
```
# Pie chart for species frequency
species_counts = iris['species'].value_counts()
plt.figure(figsize=(6,6))
# Pie chart
plt.pie(species_counts, labels=species_counts.index, autopct='%1.1f%%', startangle=90, colors=['lightblue', 'lightgreen', 'lightcoral'])
plt.title('Species Frequency in Iris Dataset')
plt.show()
```



Relationship Between Sepal Length and Width:

Relationship Between Sepal Length and Sepal Width

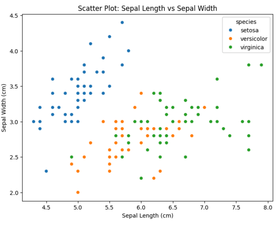
This task involves creating a scatter plot to find the relationship between sepal length and sepal width.

Scatter plots are used to show the relationship or correlation between two variables. Each point represents a data observation.

Purpose:

To determine if there is a linear, non-linear, or no relationship between sepal length and width.

In [5]:
```
# Scatter plot to find the relationship between Sepal Length and Sepal Width
plt.figure(figsize=(6,6))
sb.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris)
plt.title('Scatter Plot: Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



Distribution of Sepal and Petal Features:

Here, you will create a plot to show how sepal length, sepal width, petal length, and petal width are distributed.

Histograms or KDE plots (Kernel Density Estimate) can be used to visualize the distribution of data. These plots provide insights into how each feature is spread and whether they are skewed or normally distributed.

Purpose:

To understand how each feature (sepal and petal measurements) is distributed across the dataset.

In [ ]:
```
plt.figure(figsize=(12,10))
    for i, feature in enumerate(['sepal_length', 'sepal_width', 'petal_length', 'petal_width']):
        plt.subplot(2, 2, i+1)
        sb.histplot(iris[feature], kde=True)
        plt.title(f'Distribution of {feature}')
    plt.tight_layout()
    plt.show()
```

Jointplot of Sepal Length vs Sepal Width:

A joint plot is a combination of a scatter plot with the individual distributions of sepal length and sepal width displayed on the same plot.

The scatter plot shows the relationship between two variables, while histograms (or KDE plots) along the axes show the individual distributions of these variables.

Purpose:

To analyze the relationship between the two variables along with their respective distributions.

In [ ]:
```
sb.jointplot(x='sepal_length', y='sepal_width', data=iris, hue='species', kind='scatter')
    plt.show()
```

KDE Plot for Setosa Species (Sepal Length vs Sepal Width):

This task involves creating a KDE plot (Kernel Density Estimate) for sepal length versus sepal width for the Setosa species.

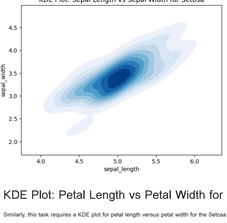
KDE plots are a smoothed approximation of a histogram. They help visualize the probability density of the variables.

Purpose:

To see how the Setosa species' sepal measurements are distributed and where the highest density of observations occurs.

In [57]:
```
setosa = iris[iris['species'] == 'setosa']

# KDE plot: Sepal Length vs Sepal Width for Setosa
sb.kdeplot(x='sepal_length', y='sepal_width', data=setosa, cmap='Blues', fill=True, thresh=0.05)
plt.title('KDE Plot: Sepal Length vs Sepal Width for Setosa')
plt.show()
```



KDE Plot: Petal Length vs Petal Width for Setosa

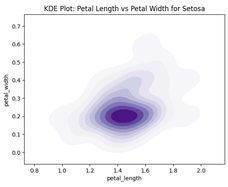
Similarly, this task requires a KDE plot for petal length versus petal width for the Setosa species.

It will show the density distribution of petal measurements for the Setosa species.

Purpose:

To visualize how the petal length and width are distributed in Setosa species and identify patterns.

In [61]:
```
# KDE plot: Petal Length vs Petal Width for Setosa
sb.kdeplot(x='petal_length', y='petal_width', data=setosa, cmap='Purples', fill=True, thresh=0.05)
plt.title('KDE Plot: Petal Length vs Petal Width for Setosa')
plt.show()
```


KDE Plot: Petal Length vs Petal Width for Setosa
petal_width
petal_length
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.2
1.4
1.6
1.8
2.0