Univariate and Bivariate Analysis in Machine Learning

1. **Load the Iris dataset into a Pandas DataFrame.**

```
df = pd.read_csv('iris_data.csv')
df.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
```

2. **Perform univariate analysis on all numeric columns (sepal length,sepal width, petal length, petal width).**
   - **Calculate summary statistics.**

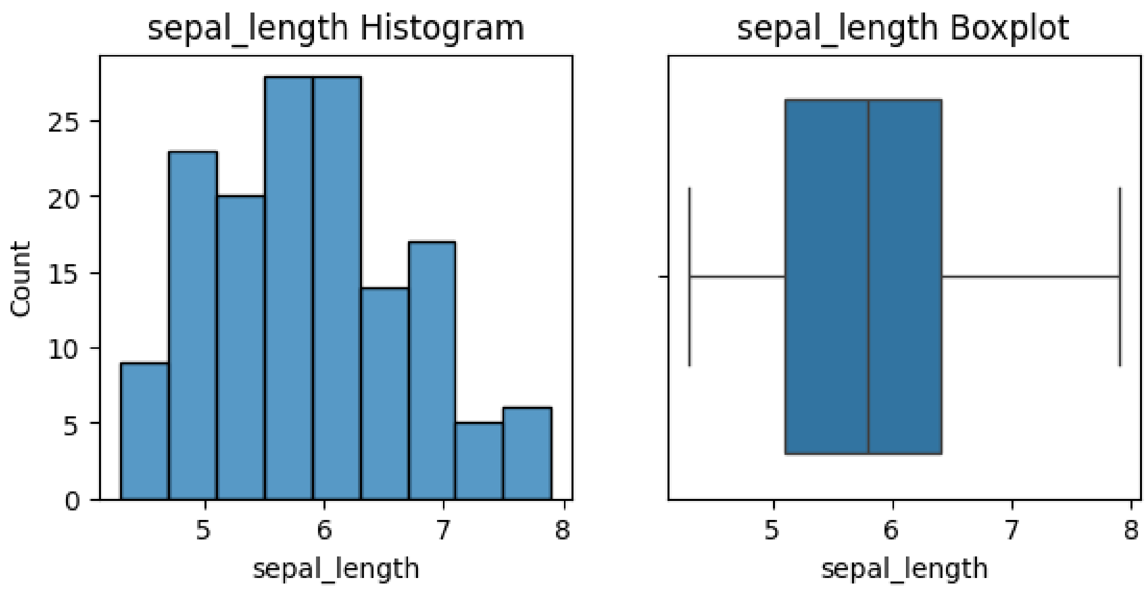
```
    #print(df.describe())

for i in df.columns:
    if df[i].dtype == 'float64':
        print(f'{i}:')
        print(f'Median: {df[i].median()}')
        print(f'Mode: {df[i].mode()}')
        print(f'Variance: {df[i].var()}')
        print(f'Standard Deviation: {df[i].std()}')
        print(f'Mean: {df[i].mean()}')
        print(f'Range: {df[i].max() - df[i].min()}')
        print('\n')
```

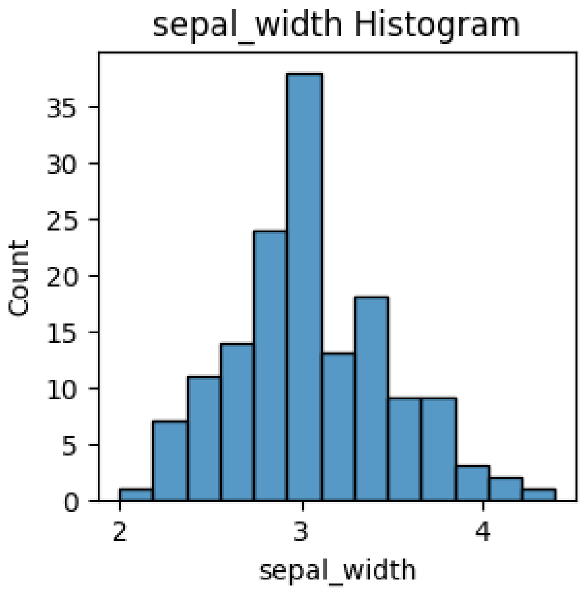
```
sepal_length:
Median: 5.8
Mode: 0    5.0
Name: sepal_length, dtype: float64
Variance: 0.6856935123042507
Standard Deviation: 0.828066127977863
Mean: 5.843333333333334
Range: 3.6000000000000005


sepal_width:
Median: 3.0
Mode: 0    3.0
Name: sepal_width, dtype: float64
Variance: 0.1880040268456376
Standard Deviation: 0.4335943113621737
Mean: 3.0540000000000003
Range: 2.4000000000000004


petal_length:
Median: 4.35
Mode: 0    1.5
```

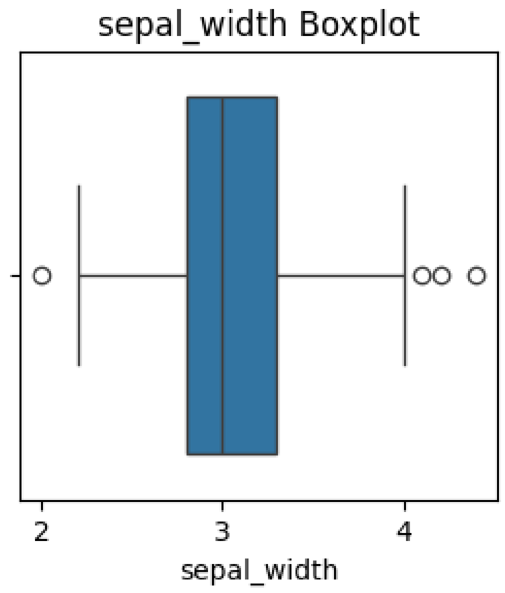
```
Name: petal_length, dtype: float64
Variance: 3.113179418344519
Standard Deviation: 1.7644204199522626
Mean: 3.758666666666666
Range: 5.9


petal_width:
Median: 1.3
Mode: 0    0.2
Name: petal_width, dtype: float64
Variance: 0.582414317673378
Standard Deviation: 0.7631607417008411
Mean: 1.1986666666666668
Range: 2.4
```

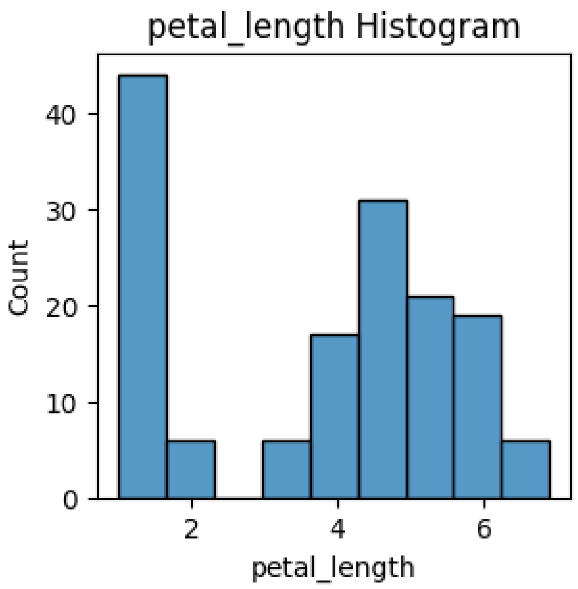
- **Create histograms and box plots for each variable.**

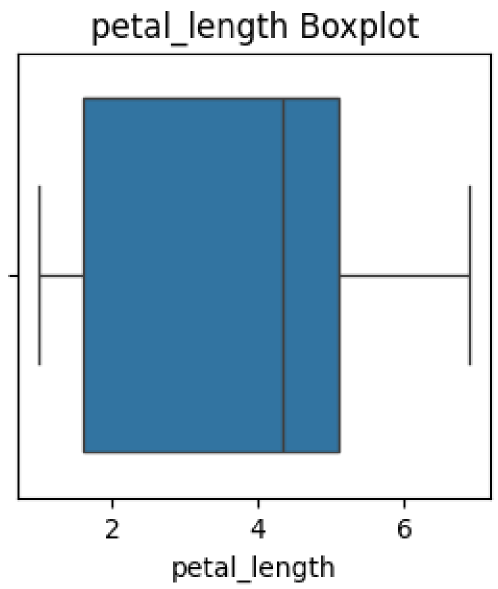
```
for i in df.columns:
    if df[i].dtype == 'float64':
        plt.figure(figsize=(7, 3))
        plt.subplot(1, 2, 1)
        sns.histplot(df[i])
        plt.title(f'{i} Histogram')
        plt.subplot(1, 2, 2)
        sns.boxplot(x=df[i])
        plt.title(f'{i} Boxplot')
        plt.show()
```

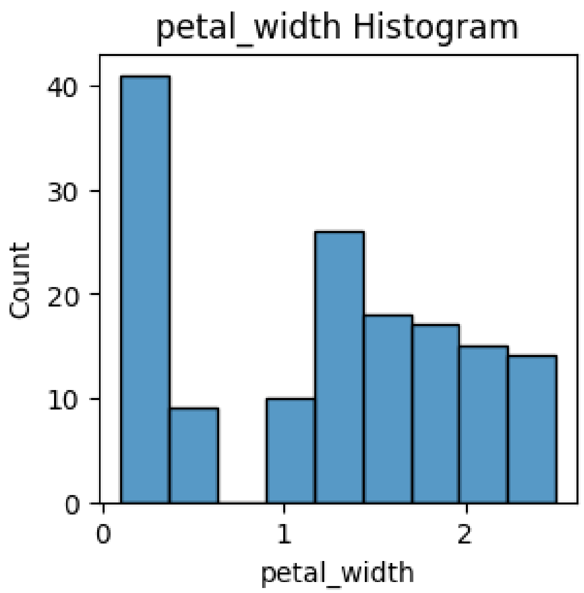


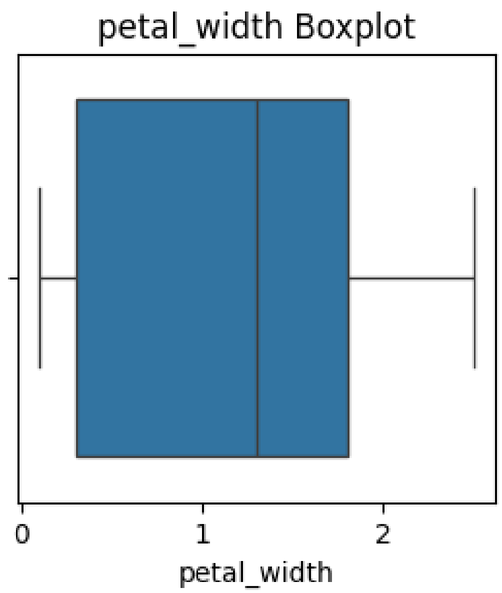

sepal_width Histogram
Count
sepal_width
0
5
10
15
20
25
30
35
2
3
4


sepal_width Boxplot
sepal_width
2
3
4


petal_length Histogram
Count
petal_length
0
10
20
30
40
2
4
6


petal_length Boxplot
petal_length
2
4
6

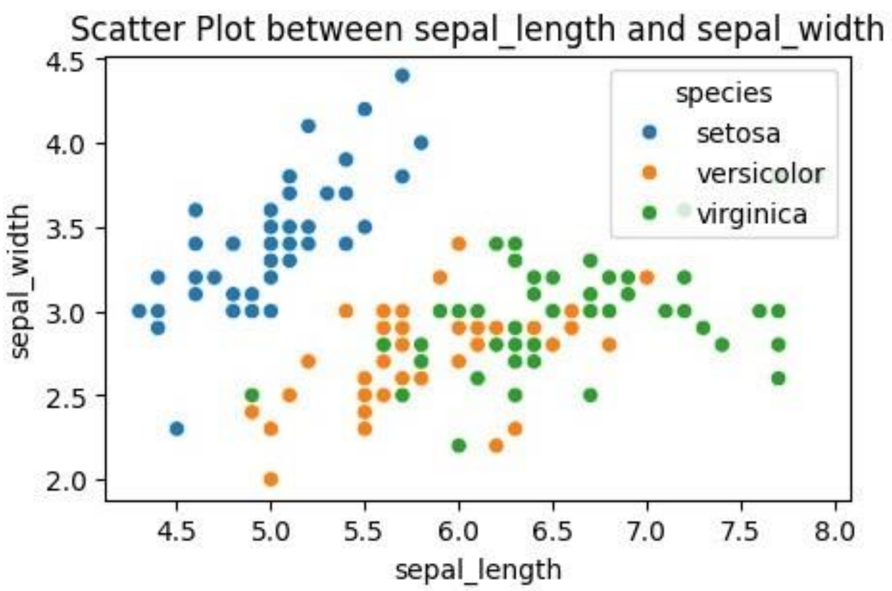

petal_width Histogram
Count
petal_width
0
10
20
30
40
0
1
2

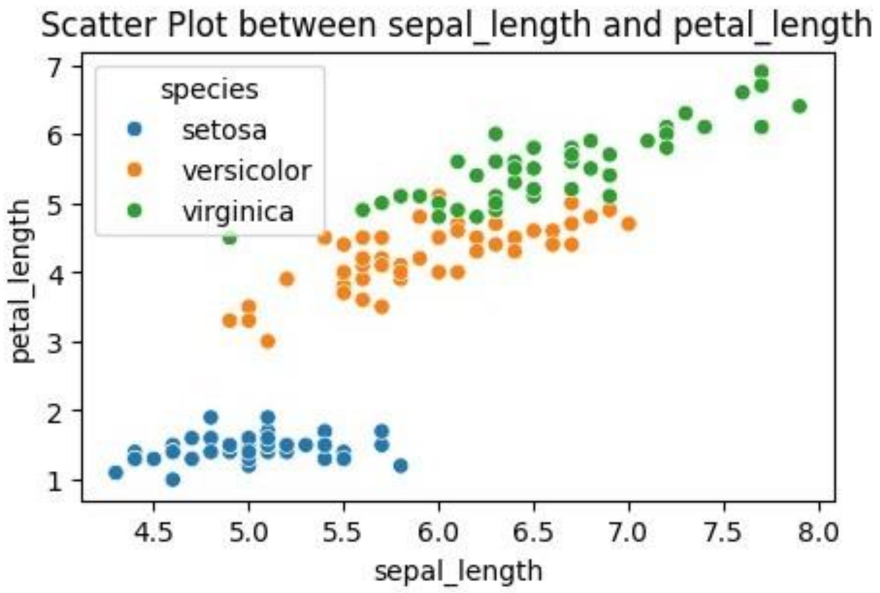

petal_width Boxplot
petal_width
0
1
2

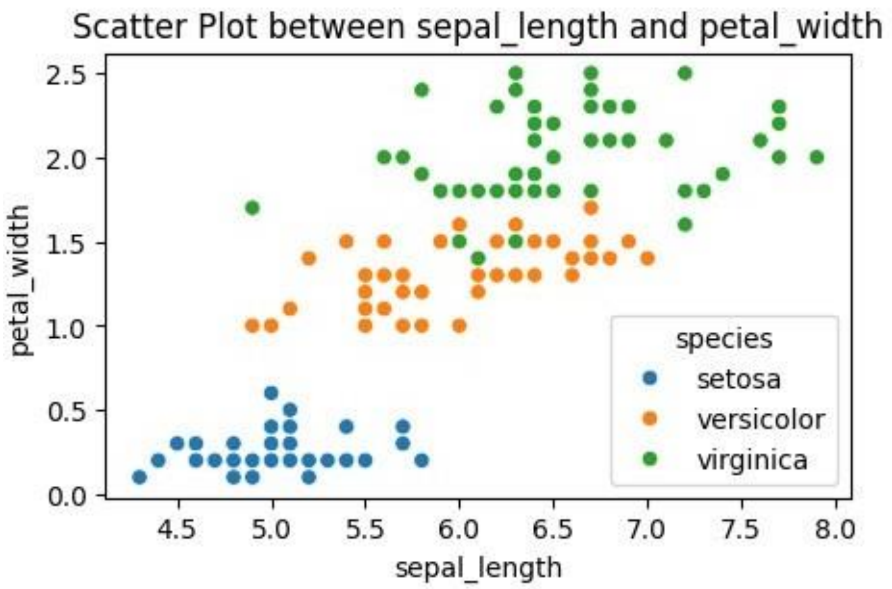
3. Perform bivariate analysis to explore relationships between pairsof variables.

- **Create scatter plots for pairs of variables.**

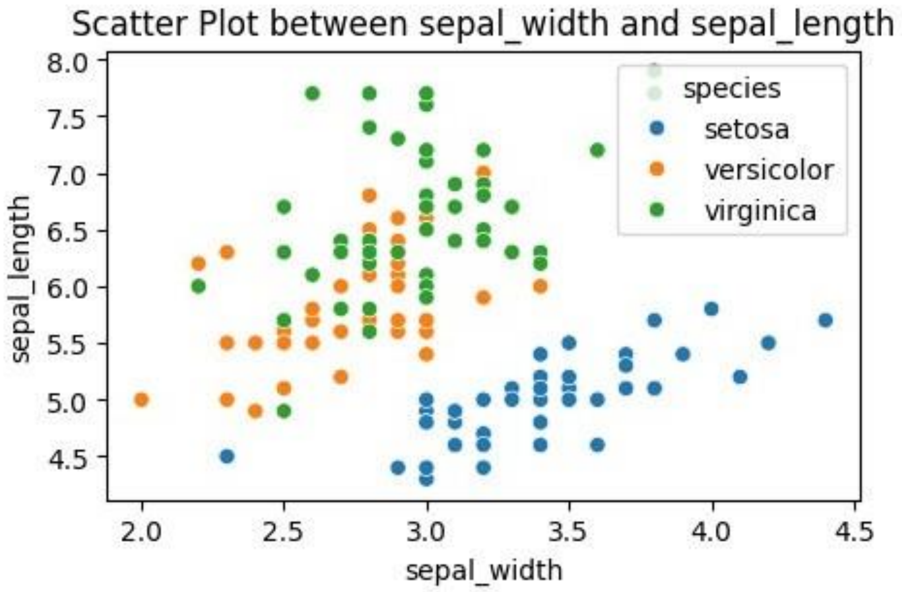
```
for i in df.columns:
    if df[i].dtype == 'float64':
        for j in df.columns:
            if df[j].dtype == 'float64' and i != j:
                plt.figure(figsize=(5, 3))
                sns.scatterplot(x=df[i], y=df[j], hue = df['species'])
                plt.title(f'Scatter Plot between {i} and {j}')
                plt.show()
```



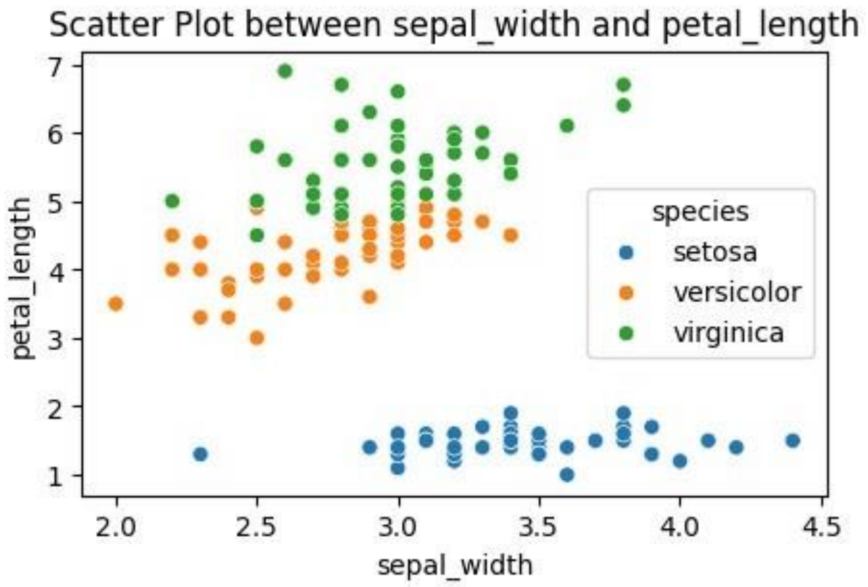




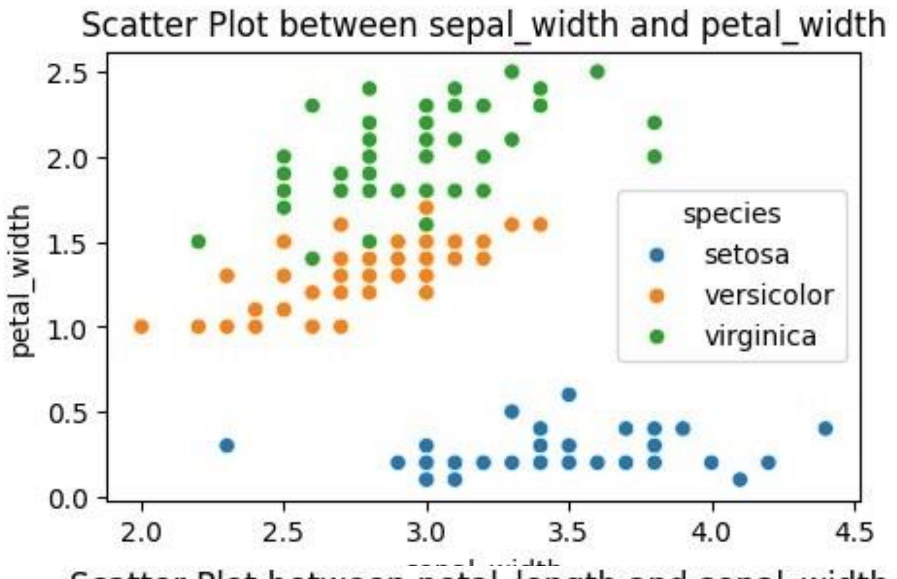
Scatter Plot between sepal_length and petal_width
petal_width
sepal_length
species
setosa
versicolor
virginica

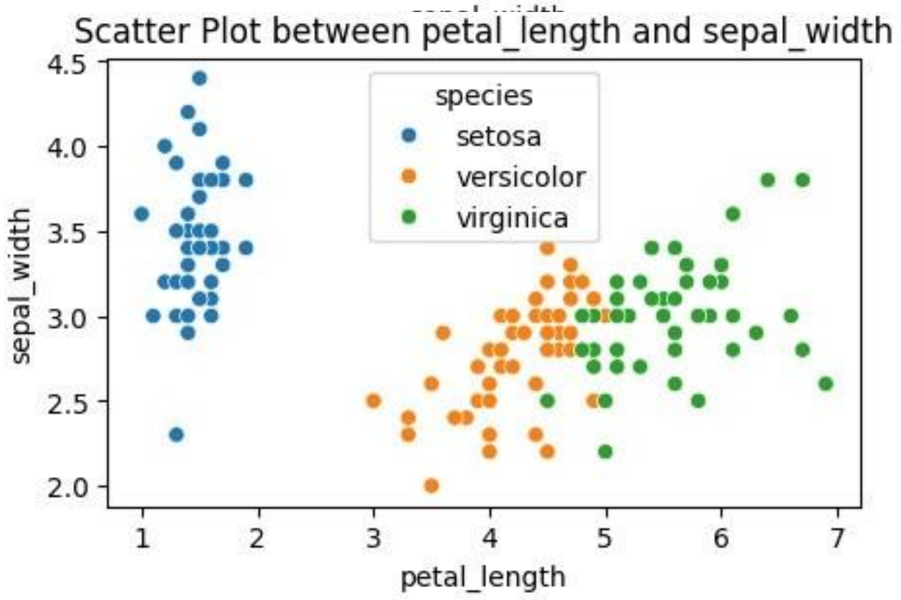


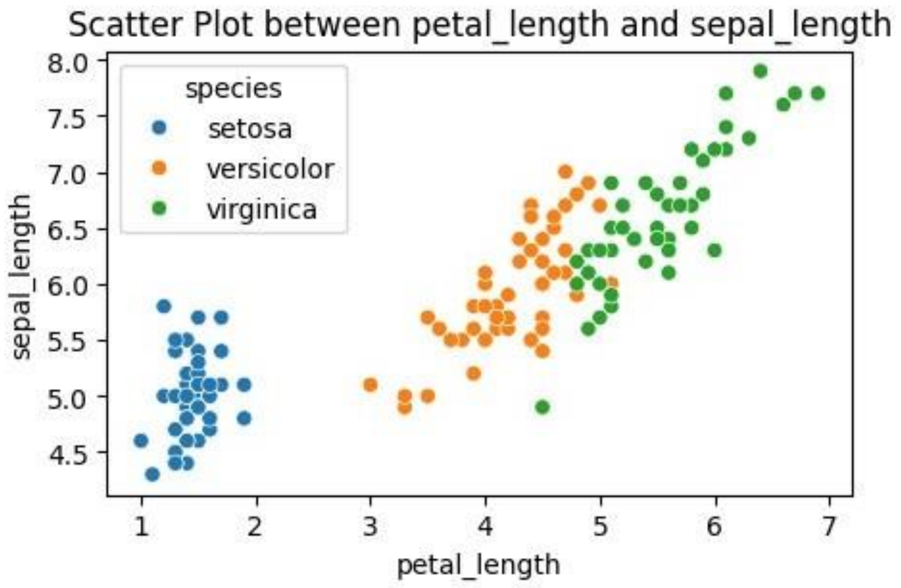
Scatter Plot between sepal_width and sepal_length
sepal_length
sepal_width
species
setosa
versicolor
virginica

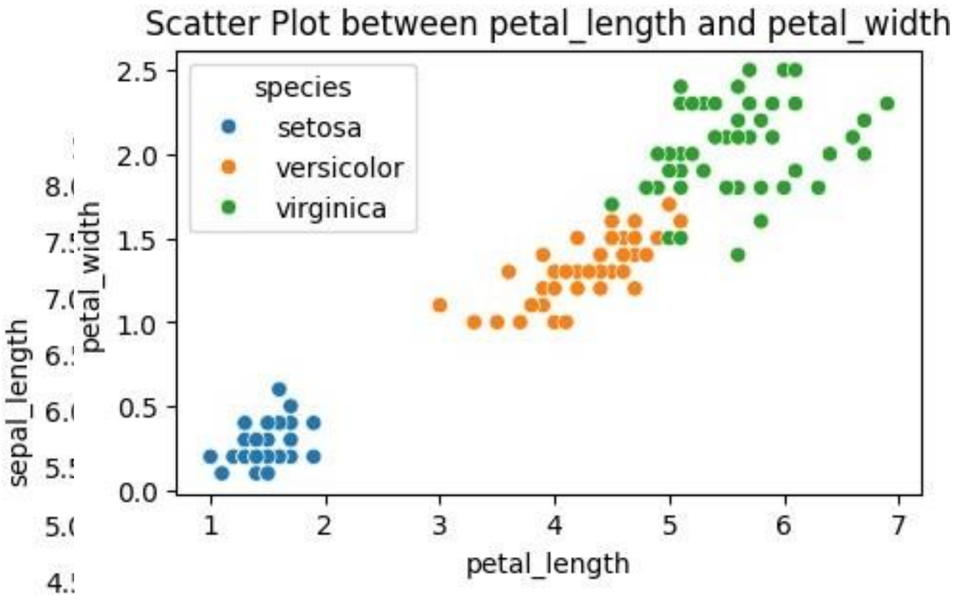


Scatter Plot between sepal_width and petal_length
petal_length
sepal_width
species
setosa
versicolor
virginica

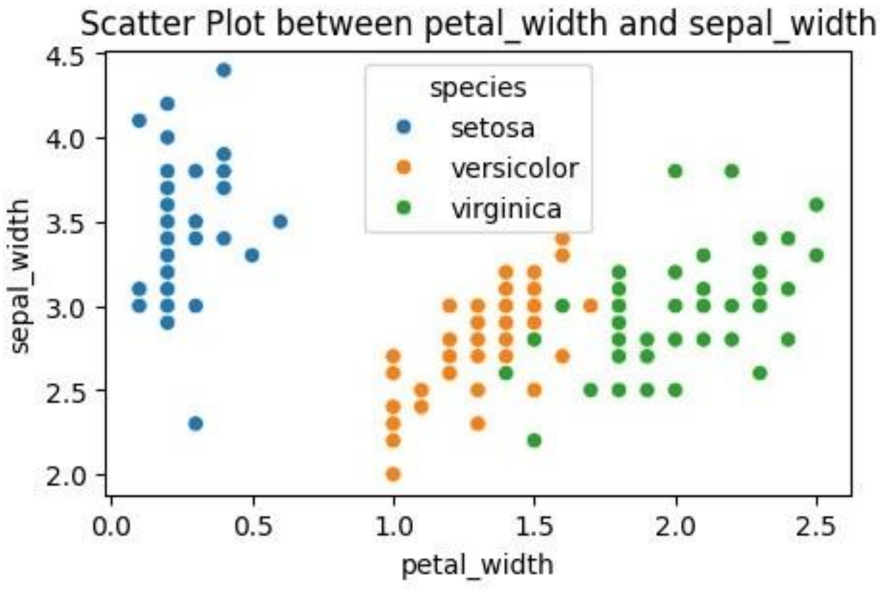

Scatter Plot between sepal_width and petal_width
petal_width
sepal_width
species
setosa
versicolor
virginica


Scatter Plot between petal_length and sepal_width
sepal_width
petal_length
species
setosa
versicolor
virginica

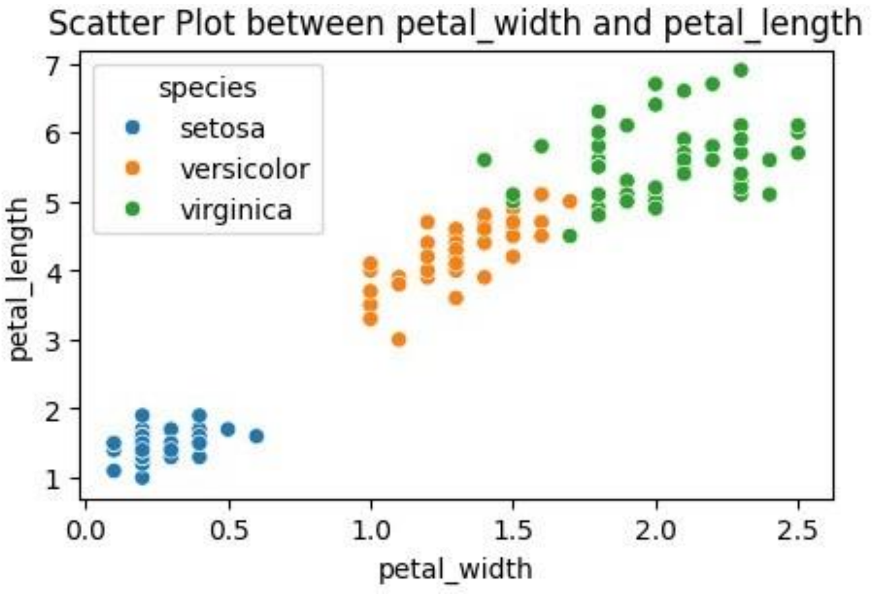

Scatter Plot between petal_length and sepal_length
sepal_length
petal_length
species
setosa
versicolor
virginica



Scatter Plot between petal_length and petal_width
species
setosa
versicolor
virginica
petal_width
petal_length
sepal_length



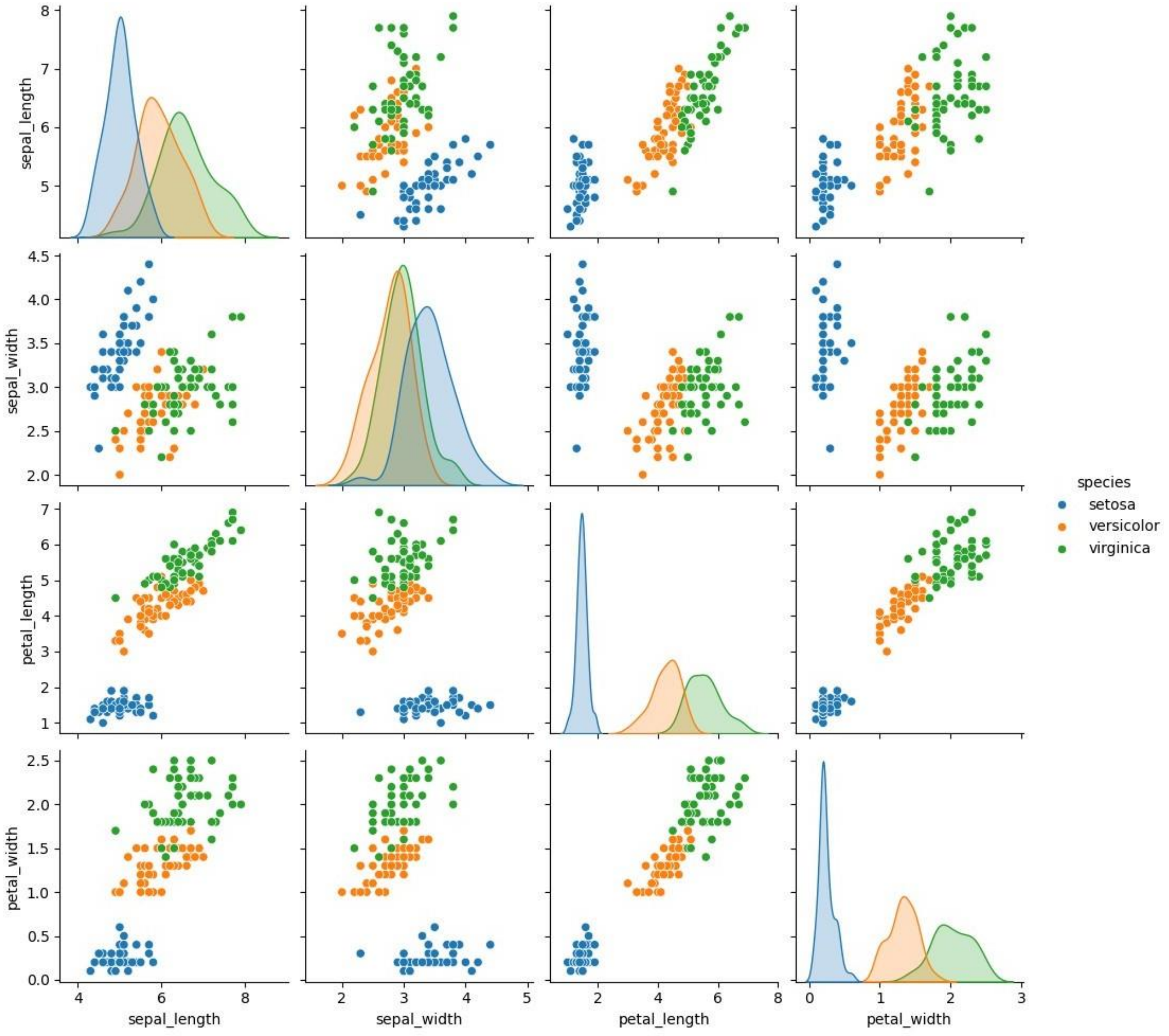
Scatter Plot between petal_width and sepal_width
species
setosa
versicolor
virginica
sepal_width
petal_width



Scatter Plot between petal_width and petal_length
species
setosa
versicolor
virginica
petal_length
petal_width

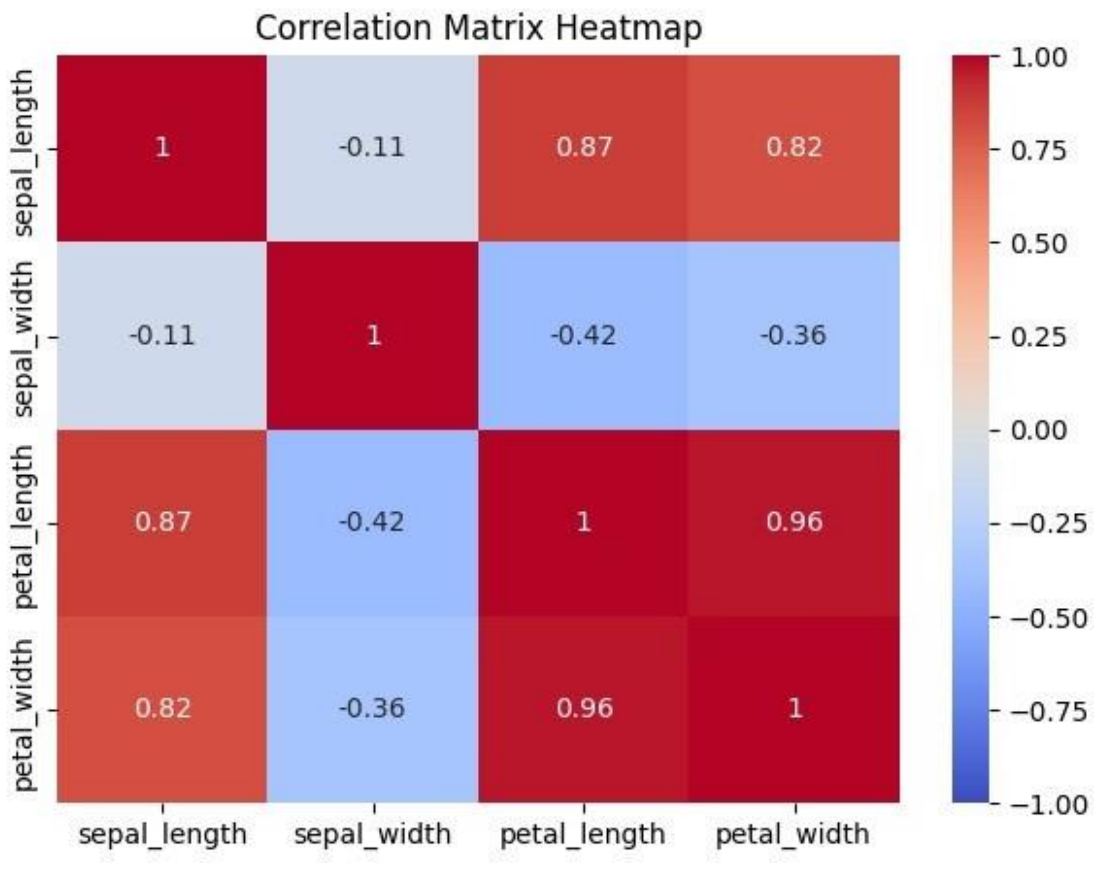
- **Create a pair plot to visualize relationships between all pairs ofvariables.**

```
sns.pairplot(df, hue='species')
plt.show()
```



- **Calculate and visualize the correlation matrix.**

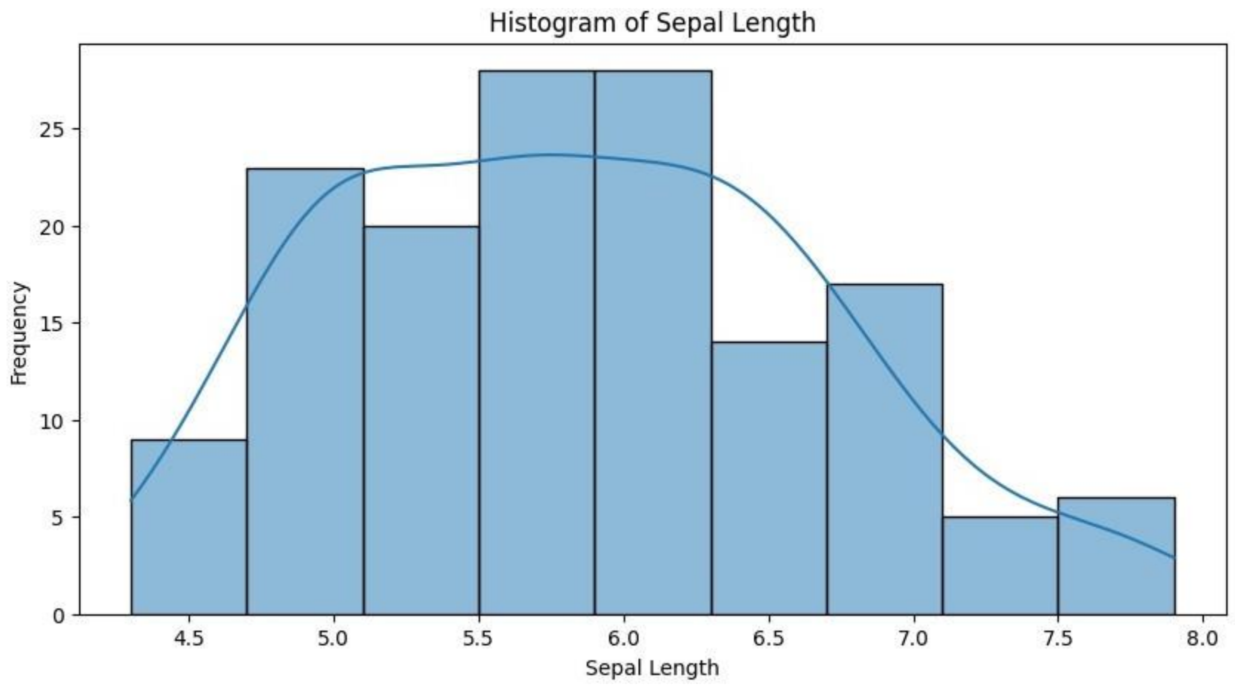
```
df = df.drop(['species'], axis=1)

corr_matrix = df.corr()

plt.figure(figsize=(7, 5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



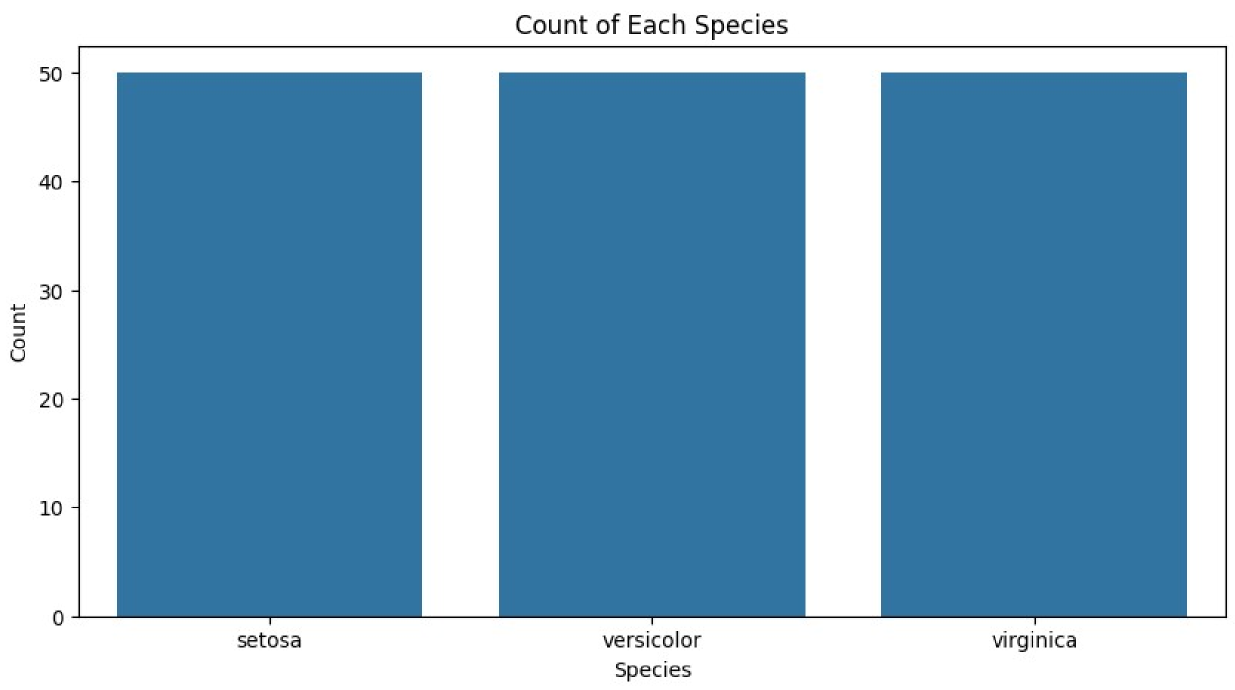
Exploring Various Types of Visual Representations in Python

1. **Create each type of visualization (histogram, bar chart, line chart,bar and line chart, pie chart, heatmap, pair plot, box plot).**

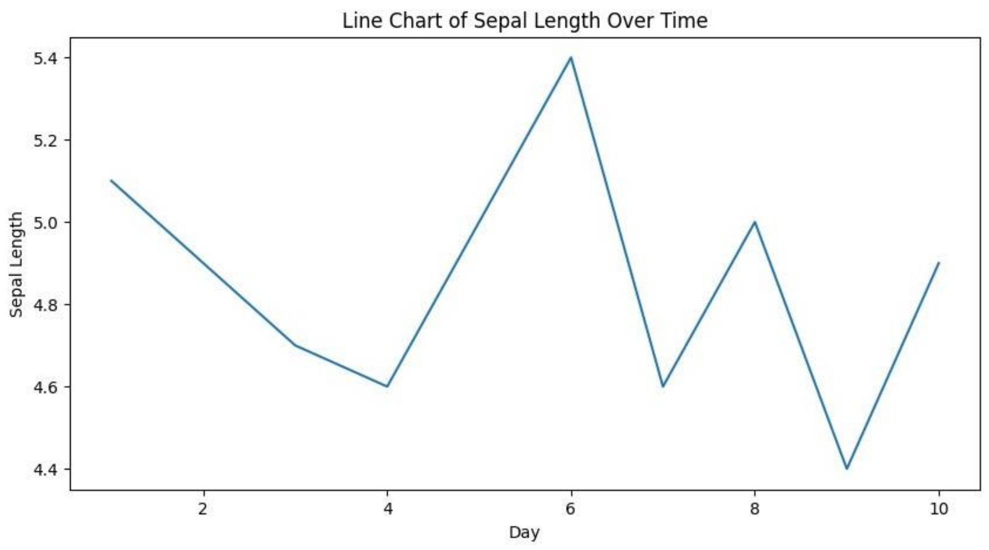
```
# Histogram
plt.figure(figsize=(10, 5))
sns.histplot(df['sepal_length'], kde=True)
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.show()
```



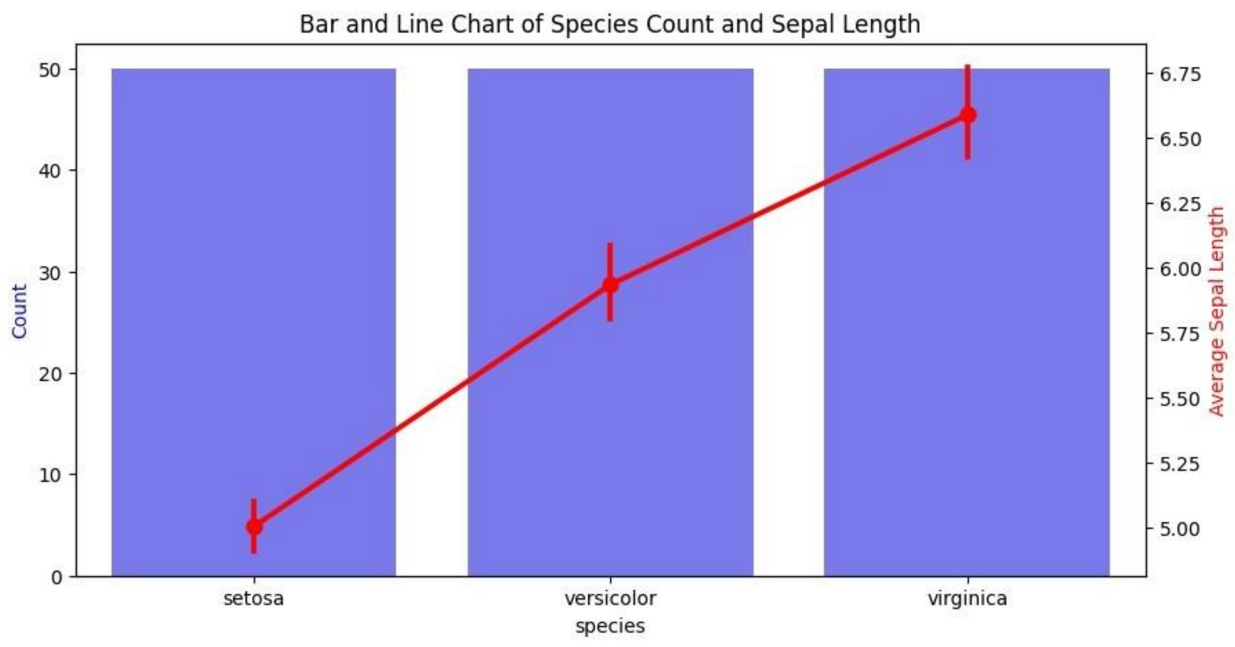
```
df = pd.read_csv('iris_data.csv')
# Bar Chart
plt.figure(figsize=(10, 5))
sns.countplot(x='species', data=df)
plt.title('Count of Each Species')
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```



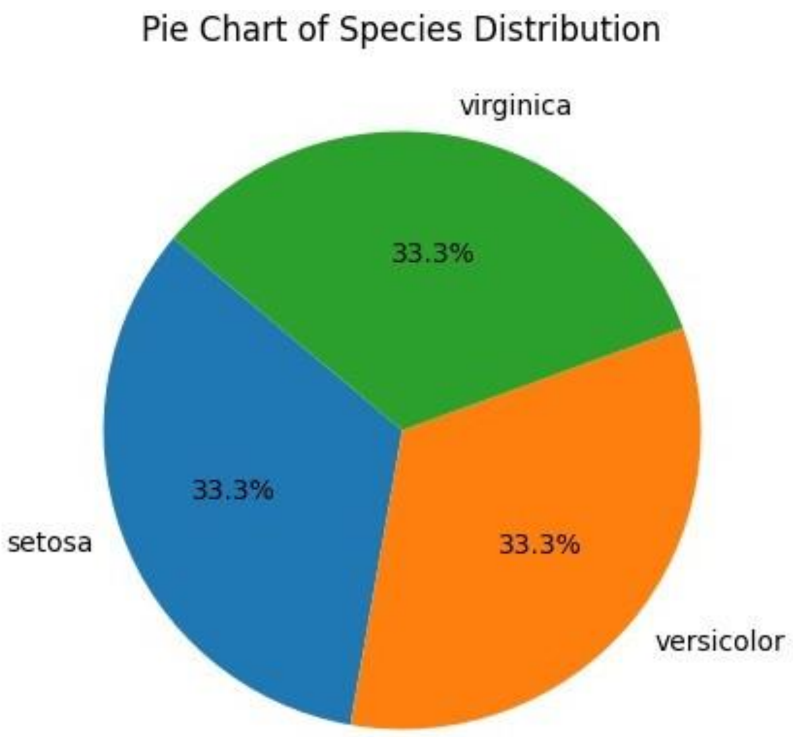
```python
# Simulating data over time
time_data = pd.DataFrame({
    'day': range(1, 11),
    'sepal_length': df['sepal_length'][:10]
})

# Line Chart
plt.figure(figsize=(10, 5))
sns.lineplot(x='day', y='sepal_length', data=time_data)
plt.title('Line Chart of Sepal Length Over Time')
plt.xlabel('Day')
plt.ylabel('Sepal Length')
plt.show()
```



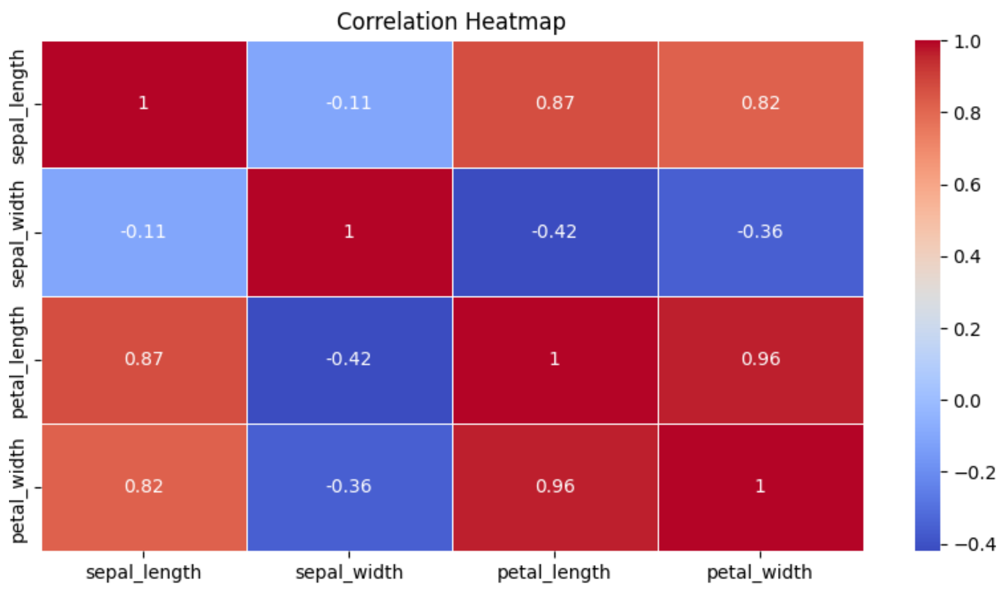
```
fig, ax1 = plt.subplots(figsize=(10, 5))

# Bar chart for count of species
sns.countplot(x='species', data=df, ax=ax1, alpha=0.6, color='b')
ax1.set_ylabel('Count', color='b')

# Line chart for average sepal length of species
ax2 = ax1.twinx()
sns.pointplot(x='species', y='sepal_length', data=df, ax=ax2,
color='r', markers='o', linestyles='-')
ax2.set_ylabel('Average Sepal Length', color='r')
plt.title('Bar and Line Chart of Species Count and Sepal Length')
plt.show()
```



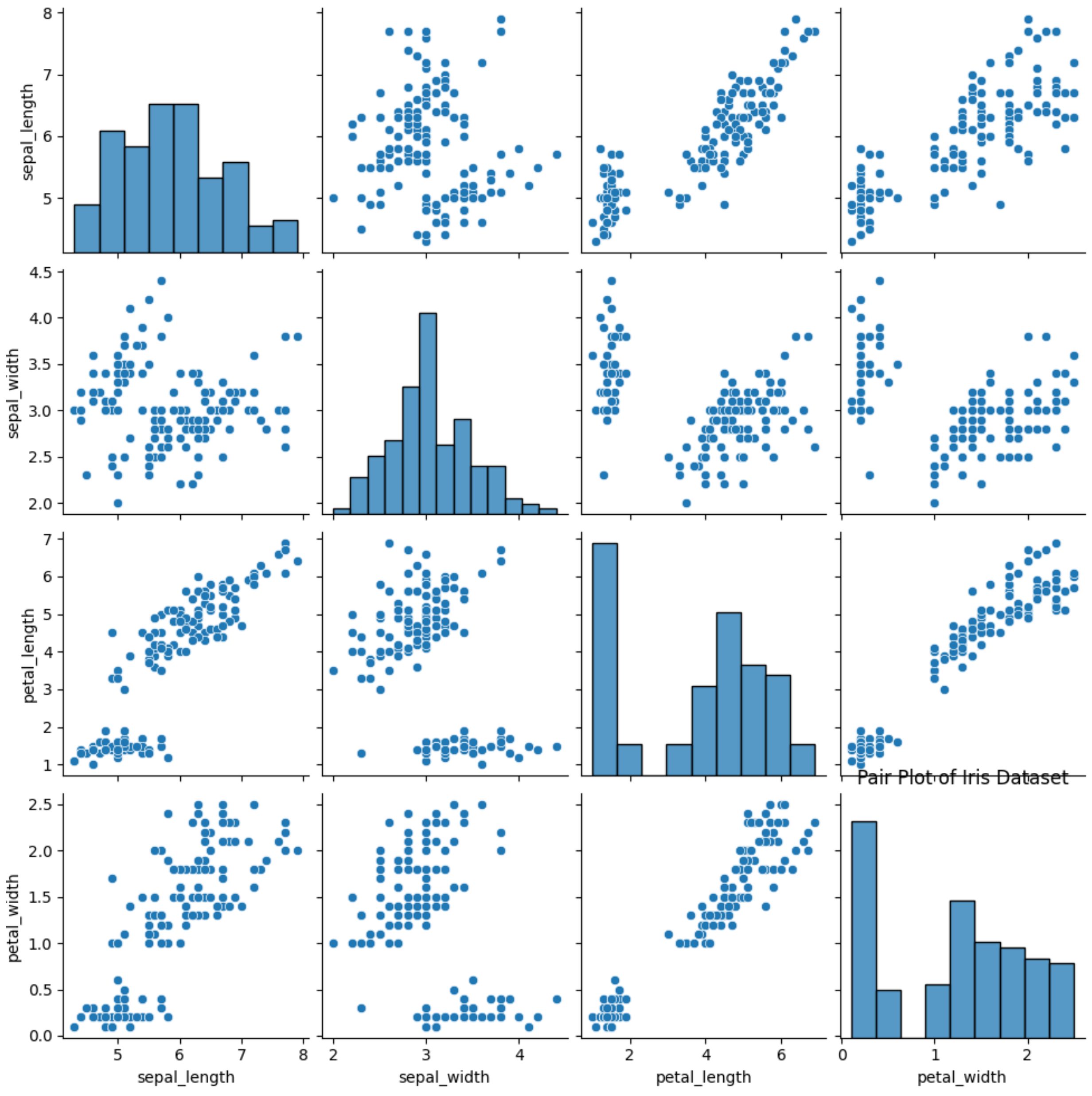
```
# Pie Chart
species_count = df['species'].value_counts()
plt.figure(figsize=(10, 5))
plt.pie(species_count, labels=species_count.index, autopct='%1.1f%%',
startangle=140)
plt.title('Pie Chart of Species Distribution')
plt.show()
```



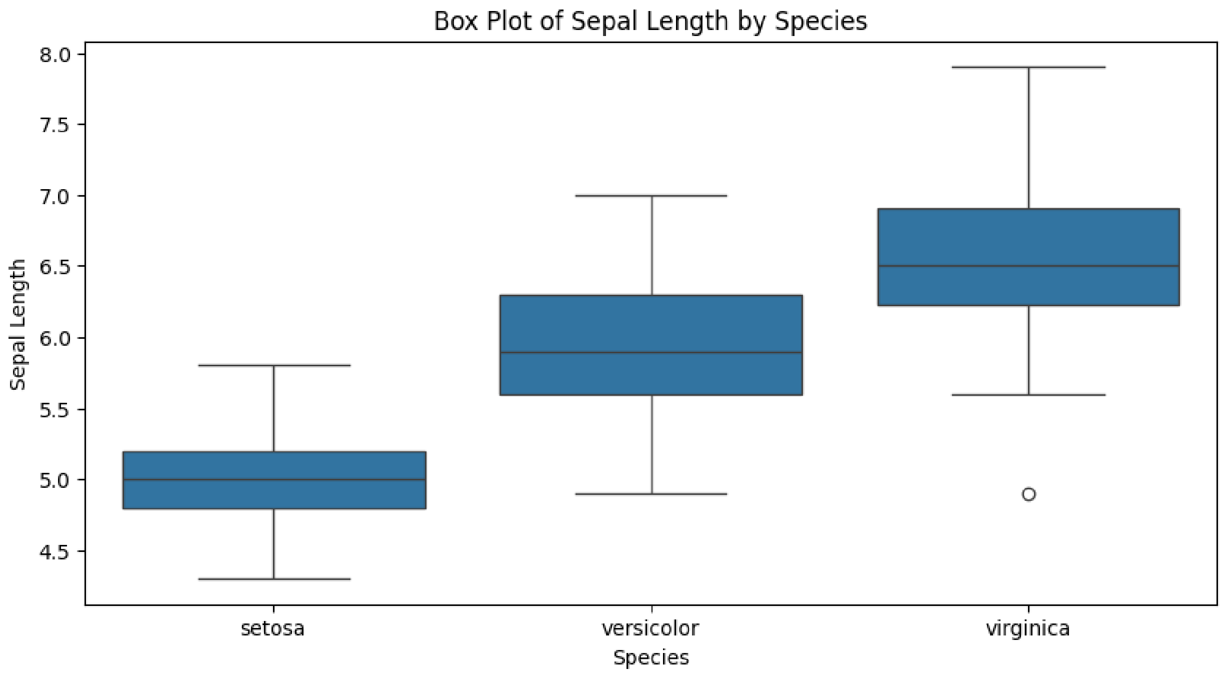
```python
# Heatmap
df = df.drop(['species'], axis=1)
plt.figure(figsize=(10, 5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



```
df = pd.read_csv('iris_data.csv')
# Pair Plot
sns.pairplot(df)
plt.title('Pair Plot of Iris Dataset')
plt.show()
```



```python
# Box Plot
plt.figure(figsize=(10, 5))
sns.boxplot(x='species', y='sepal_length', data=df)
plt.title('Box Plot of Sepal Length by Species')
plt.xlabel('Species')
plt.ylabel('Sepal Length')
plt.show()
```



2. Write a brief explanation of the purpose of each visualization.

Here's a brief overview of the purposes of these common visualizations:

1. Histogram: A histogram shows the distribution of a single continuous variable by dividing it into intervals (bins) and displaying the frequency of data points in each bin. It's useful for visualizing the spread, central tendency, and skewness of a dataset.

2. Bar Chart: A bar chart uses rectangular bars to represent categorical data. The height (or length) of each bar shows the frequency or value of each category, making it easy to compare different categories side-by-side.

3. Line Chart: A line chart connects data points with a line to show trends over time or another continuous variable. It's ideal for visualizing changes, patterns, or fluctuations over intervals, often used for time series data.

4. Bar and Line Chart: This combination chart includes both bars and lines, usually on a shared axis, to show relationships between different types of data in a single plot. It can compare different data series or reveal trends alongside category-specific values.

5. Pie Chart: A pie chart displays categorical data as slices of a circular “pie,” with each slice representing a proportion of the total. It’s used for showing relative percentages or parts of a whole but is generally best for data with a small number of categories.

6. Heatmap: A heatmap uses color intensities to show the values of data points within a matrix or grid. It's great for visualizing the relationship between two variables in a compact, intuitive format, especially in correlation matrices or geographic data.

7. Pair Plot: A pair plot (scatterplot matrix) shows pairwise relationships between multiple variables in a dataset. It's a powerful tool for exploring possible correlations and patterns between variables at a glance.

8. Box Plot: A box plot shows the distribution of data based on a five-number summary (minimum, first quartile, median, third quartile, and maximum). It highlights outliers and skewness, making it useful for comparing distributions across multiple gr