Exploratory Data Analysis

load a CSV dataset from a local file using `pandas.read_csv,,, use basic Pandas functions for Exploratory Data Analysis-EDA describe and discriminate between basic data types such as categorical, quantitative, continuous, discrete, ordinal, nominal and identifier

[Objective 01] LOAD THE DATASET AND EXPLORE

Overview

Steps for loading a dataset:

1. Learn as much as you can about the dataset:

- Number of rows
- Number of columns
- Column headers (Is there a "data dictionary"?)
- Is there missing data?
- Open the raw file and look at it - it may not be formatted the way you expect.

2. Try loading the dataset using `pandas.read_csv()` and if things aren't acting the way that you expect, investigate until you can get it loading correctly.

3. Keep in mind that functions like `pandas.read_csv()` have a lot of optional parameters that might help us change the way that data is read in. If you get stuck, google, read the documentation, and try things out.

4. You might need to type out column headers by hand if they are not provided in a neat format in the original dataset. It can be a drag.

Follow Along

Learn about the dataset and look at the raw file.

Attempt to load it.

In [62]:

```
column_headers = ["Id", "SepalLengthCm", "SepalWidthCm",
                  "PetalLengthCm", "PetalWidthCm", "Species"]

# Load dataset with headers
df = pd.read_csv("iris_dirty.csv", names=column_headers, header=None)
```

```
# Show first 5 rows
df.head()
```

Out[62]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	NaN	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1.0	5.1	3.5	1.4	0.2	Setosa
2	2.0	4.9	3	1.4	0.2	setosa
3	3.0	4.7	3.2	1.3	0.2	setosa
4	4.0	4.6	3.1	1.5	0.2	setosa

In [63]:

```
import pandas as pd
import numpy as np
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline

# Load dataset
df = pd.read_csv("iris_dirty.csv")
```

In [64]:

```
#Print the first 5 rows:
df.head()
```

Out[64]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	Setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	NaN	3.6	1.4	0.2	setosa

In [65]:

```
#Print the last 5 rows:
df.tail()
```

Out[65]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	NaN	3.0	5.1	1.8	virginica

In [66]:

```
#Can you print the first 7 rows?
```

```
df.head(7)
```

Out[66]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	Setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	NaN	3.6	1.4	0.2	setosa
5	6	5.4	NaN	1.7	0.4	setosa
6	7	4.6	3.4	1.4	0.3	setosa

In [67]:

```
# Shape of dataset
print("Shape of dataset:", df.shape)
```

```
Shape of dataset: (150, 6)
```

In [68]:

```
# Column names and data types
print("\nInfo:")
print(df.info())
```

```
Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    150 non-null    int64  
 1   Sepal.Length  142 non-null    float64
 2   Sepal.Width   139 non-null    float64
 3   Petal.Length  150 non-null    float64
 4   Petal.Width   150 non-null    float64
 5   Species       150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

In [69]:

```
# Summary statistics for numerical columns
print("\nDescribe:")
print(df.describe())
```

```
Describe:
       Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count  150.000000    142.000000   139.000000    150.000000   150.000000
mean    75.500000      5.840141     3.043165      3.758000     1.199333
std     43.445368      0.831905     0.438550      1.765298     0.762238
min      1.000000      4.300000     2.000000      1.000000     0.100000
25%     38.250000      5.100000     2.800000      1.600000     0.300000
50%     75.500000      5.800000     3.000000      4.350000     1.300000
75%    112.750000      6.400000     3.300000      5.100000     1.800000
max    150.000000      7.900000     4.400000      6.900000     2.500000
```

In [70]:

```
# Check missing values
print("\nMissing values:")
print(df.isnull().sum())
```

```
Missing values:
Unnamed: 0       0
Sepal.Length     8
Sepal.Width     11
Petal.Length     0
Petal.Width      0
Species          0
dtype: int64
```

In [71]:

```
# Unique values for categorical columns
print("\nUnique values for categorical columns:")
for col in df.select_dtypes(include="object").columns:
    print(col, df[col].unique())
```

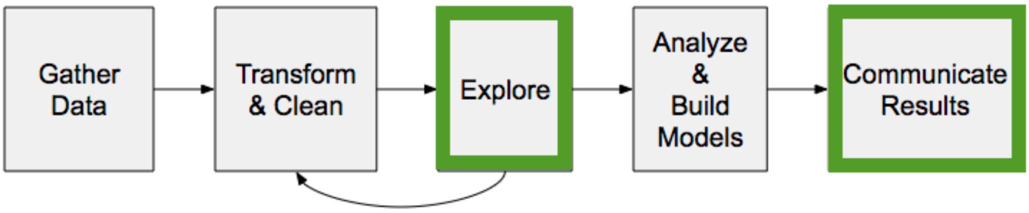
```
Unique values for categorical columns:
Species ['Setosa' 'setosa' 'SETOSA' 'versicolor' 'Versicolor' 'VERSICOLOR'
 'virginica' 'VIRGINICA']
```

In [72]:

```
rows, cols = df.shape
print(f"Dimensions of dataset: {rows} rows × {cols} columns")
```

```
Dimensions of dataset: 150 rows × 6 columns
```

Objective 02 - Use basic Pandas functions for Exploratory Data Analysis (EDA)



Overview

> Exploratory Data Analysis (EDA) refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations

Exploratory Data Analysis is often the first thing that we'll do when starting out with a new dataset.

In [73]:

```
#Learn more about the variables in the dataset using info function
print("\nInfo():")
print(df.info())
```

```
Info():
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    150 non-null    int64  
 1   Sepal.Length  142 non-null    float64
 2   Sepal.Width   139 non-null    float64
 3   Petal.Length  150 non-null    float64
 4   Petal.Width   150 non-null    float64
 5   Species       150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

Learn more about each variable

In [74]:

```
#Determine the data types
df.dtypes
```

Out[74]:

```
Unnamed: 0        int64
Sepal.Length    float64
Sepal.Width     float64
Petal.Length    float64
Petal.Width     float64
Species          object
dtype: object
```

In [75]:

```
# Summary Statistics - using (describe function)
# check if there are non-numeric column
df.describe()
```

Out[75]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	142.000000	139.000000	150.000000	150.000000
mean	75.500000	5.840141	3.043165	3.758000	1.199333
std	43.445368	0.831905	0.438550	1.765298	0.762238
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [76]:

```
df.describe(include='all')
```

Out[76]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
count	150.000000	142.000000	139.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	8
top	NaN	NaN	NaN	NaN	NaN	virginica
freq	NaN	NaN	NaN	NaN	NaN	49
mean	75.500000	5.840141	3.043165	3.758000	1.199333	NaN
std	43.445368	0.831905	0.438550	1.765298	0.762238	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [77]:

```
# try to exclude non numeric value
df.select_dtypes(include=['object']).columns
```

Out[77]:

```
Index(['Species'], dtype='object')
```

In [78]:

```
# include all

df['Species'].value_counts()
```

Out[78]:

```
Species
virginica     49
setosa        48
versicolor    48
Setosa         1
SETOSA         1
Versicolor     1
VERSICOLOR     1
VIRGINICA      1
Name: count, dtype: int64
```

In [79]:

```
# accesss a specific column of the dataframe
# Access 'Species' column
print(df['Species'].head())

# Value counts for 'Species' column
print(df['Species'].value_counts())
```

```
0    Setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: Species, dtype: object
Species
virginica     49
setosa        48
versicolor    48
Setosa         1
SETOSA         1
Versicolor     1
VERSICOLOR     1
VIRGINICA      1
Name: count, dtype: int64
```

In [80]:

```
#sort by values (any specific column)
df.sort_values(by="Sepal.Length").head()
```

Out[80]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
13	14	4.3	3.0	1.1	0.1	setosa
42	43	4.4	3.2	1.3	0.2	setosa
38	39	4.4	3.0	1.3	0.2	setosa
8	9	4.4	2.9	1.4	0.2	setosa
41	42	4.5	2.3	1.3	0.3	setosa

In [81]:

```
# check for missing values
# the number of missing values in each column
df.isnull()
```

Out[81]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	True	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	True	False	False	False	False

150 rows × 6 columns

In [82]:

```
df.isnull().sum()
```

Out[82]:

```
Unnamed: 0       0
Sepal.Length     8
Sepal.Width     11
Petal.Length     0
Petal.Width      0
Species          0
dtype: int64
```

In [83]:

```
df.isnull().sum().sum()
```

Out[83]:

```
np.int64(19)
```

In [84]:

```
# Total number of missing cells in the entire dataset
total_missing = df.isnull().sum().sum()
print("Total missing cells in dataset:", total_missing)
```

```
Total missing cells in dataset: 19
```

In [85]:

```
# try dropping rows from the dataset inplace
df.dropna(inplace=True)
```

In [86]:

```
# axis=1 to look through column headers and not row index
#Drop ID variable
df.drop("Sepal.Length", axis=1, inplace=True)

#inplace
#drop(column_name, axis=1, inplace=True)# Columns gets removed
```

Recap - what do each of these things do???

- df.shape
- df.head()

- df.dtypes
- df.describe()
- Numeric
- Non-Numeric
- df['column'].value_counts()
- df.isnull().sum()
- df.isnull().sum().sum()
- df.drop()

Objective 03 Describe and discriminate between basic data types

Overview

One of the cornerstones of Exploratory Data Analysis (EDA) is being able to identify variable types. We will need different statistical methods to display and describe each of these different types of data.

In [92]:

```
df = pd.read_csv("iris_dirty.csv", names=column_headers, header=None)
df.head()
```

Out[92]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	NaN	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1.0	5.1	3.5	1.4	0.2	Setosa
2	2.0	4.9	3	1.4	0.2	setosa
3	3.0	4.7	3.2	1.3	0.2	setosa
4	4.0	4.6	3.1	1.5	0.2	setosa

In [ ]:

```
df.select_dtypes(include=[np.number]).skew()
print(df.head())
```

```
    Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm  Species
0  NaN  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width  Species
1  1.0           5.1          3.5           1.4          0.2   Setosa
2  2.0           4.9            3           1.4          0.2   setosa
3  3.0           4.7          3.2           1.3          0.2   setosa
4  4.0           4.6          3.1           1.5          0.2   setosa
```

In [99]:

```
df = df[df["SepalWidthCm"] != "Sepal.Width"].reset_index(drop=True)
```

In [100…

```
df["SepalWidthCm"] = pd.to_numeric(df["SepalWidthCm"], errors="coerce")
```

In [101…

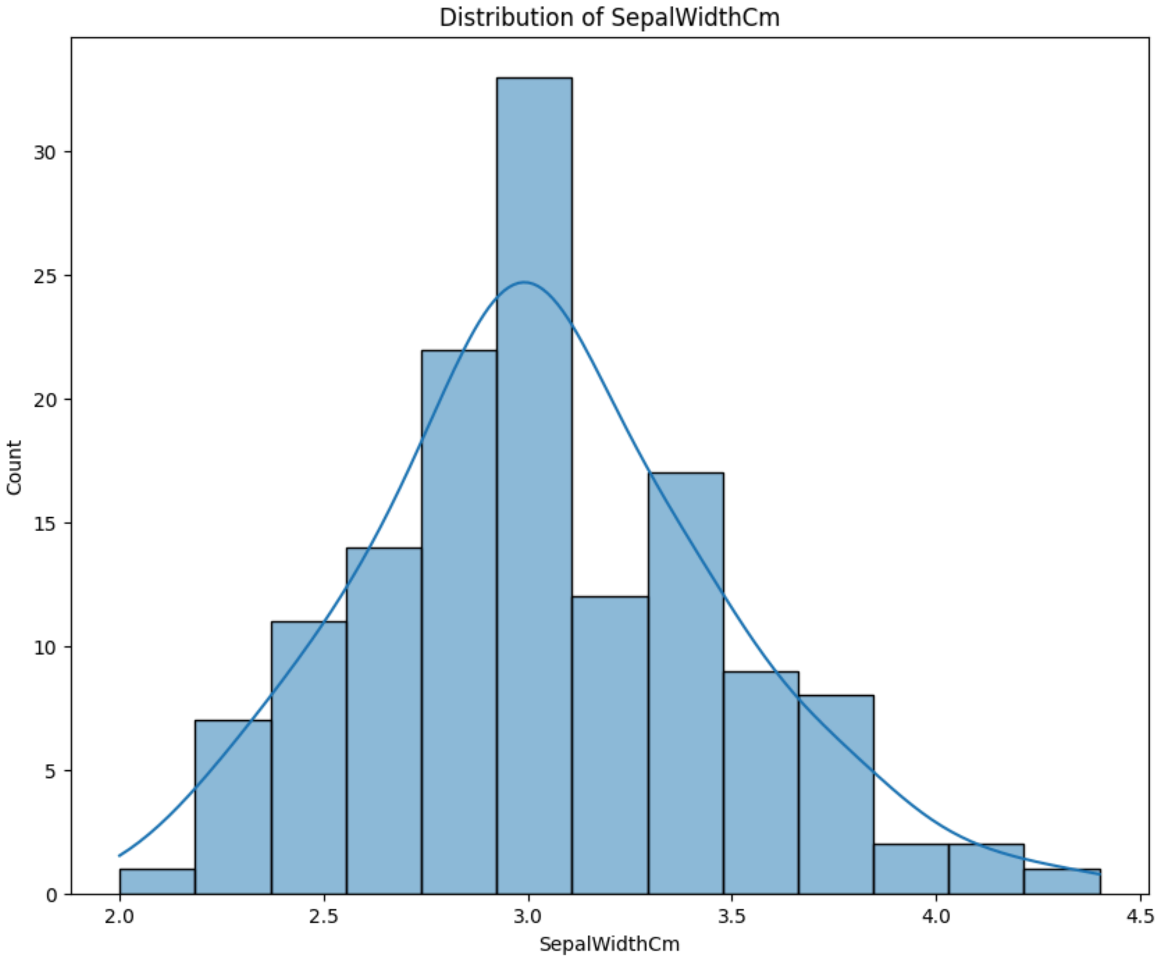
```
print(df["SepalWidthCm"].skew())
```

```
0.3406404145264381
```

In [102…

```
plt.figure(figsize= (10,8))
sns.histplot(df["SepalWidthCm"], kde=True)
```

```
plt.title("Distribution of SepalWidthCm")
plt.show()
```



In [103…

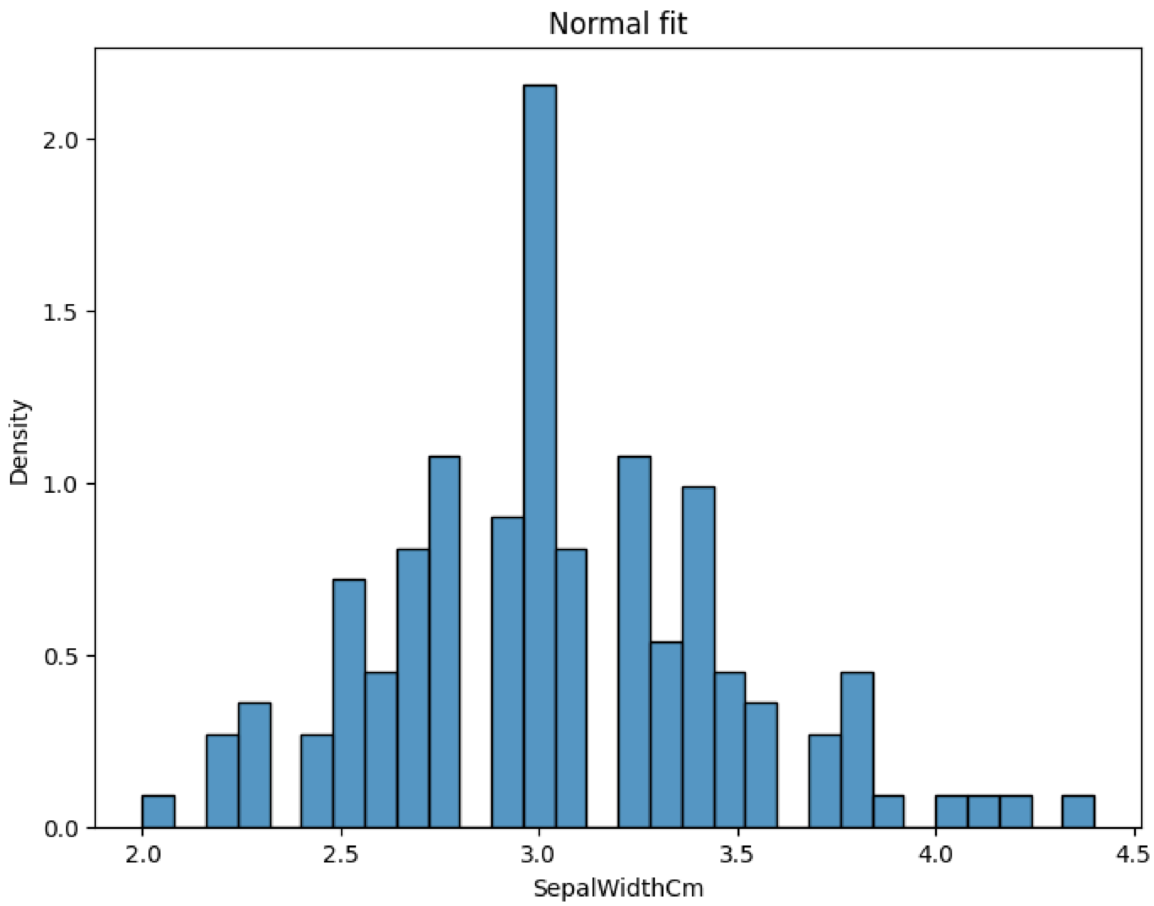
```
df["SepalWidthCm"].kurt()
```

Out[103…

```
np.float64(0.27127989325185053)
```

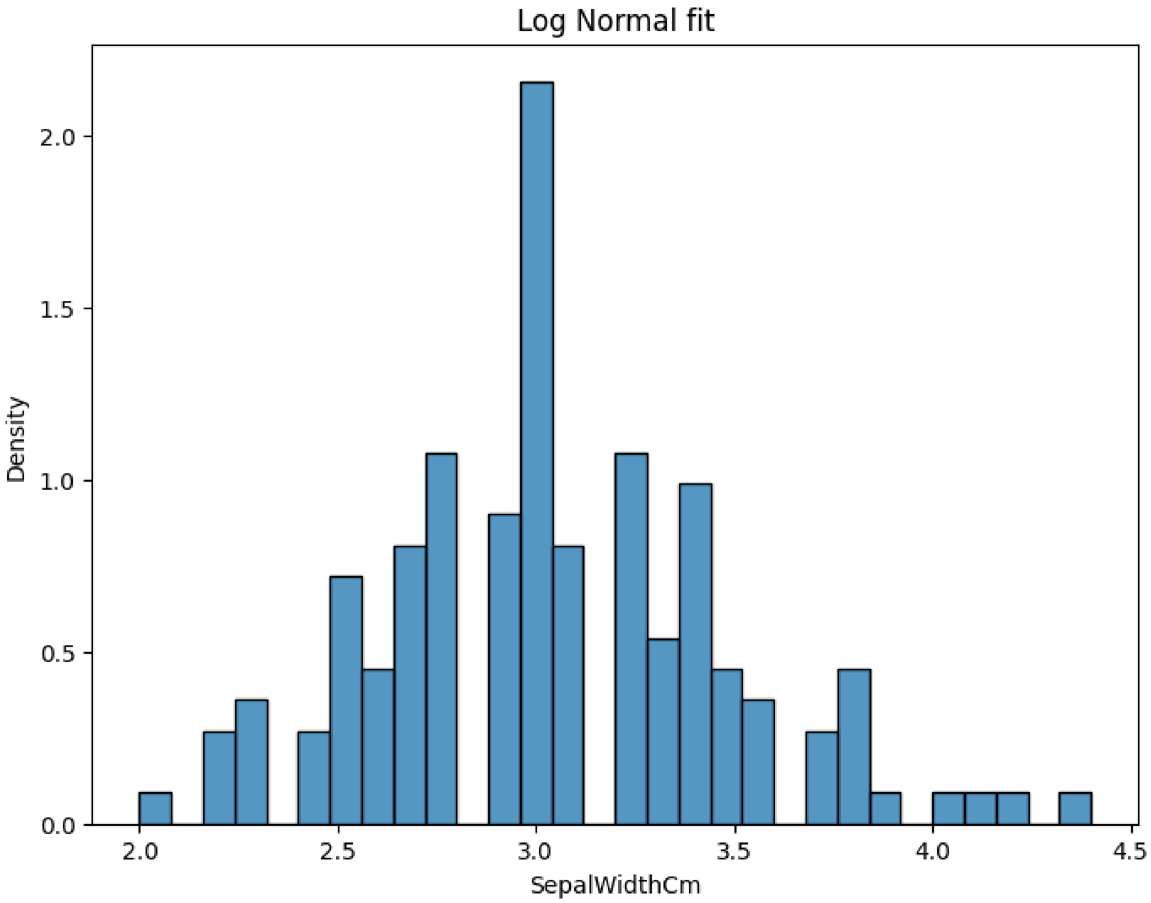
In [104…

```
y = df['SepalWidthCm']
plt.figure(figsize=(8,6))
sns.histplot(y, kde=False, stat="density", bins=30)
plt.title('Normal fit')
plt.show()
```



In [106…

```
plt.figure(figsize=(8,6))
sns.histplot(y, kde=False, stat="density", bins=30)
plt.title('Log Normal fit')
plt.show()
```



In [107…

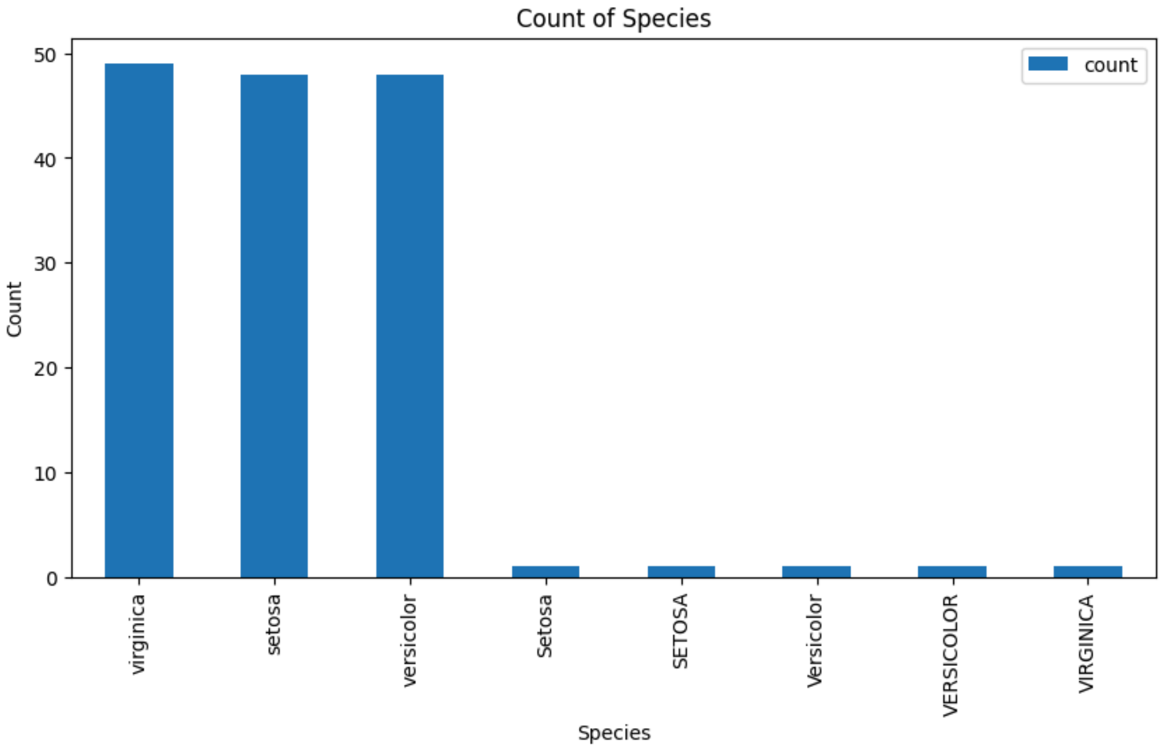
```
df['Species'].value_counts()
```

Out[107…

```
Species
virginica     49
setosa        48
versicolor    48
Setosa         1
SETOSA         1
Versicolor     1
VERSICOLOR     1
VIRGINICA      1
Name: count, dtype: int64
```

In [108…

```
df['Species'].value_counts().plot(kind = 'bar', figsize=(10,5))
plt.title('Count of Species')
plt.xlabel('Species')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [110…

```
num_cols = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidth
df[num_cols] = df[num_cols].apply(pd.to_numeric, errors="coerce")
```

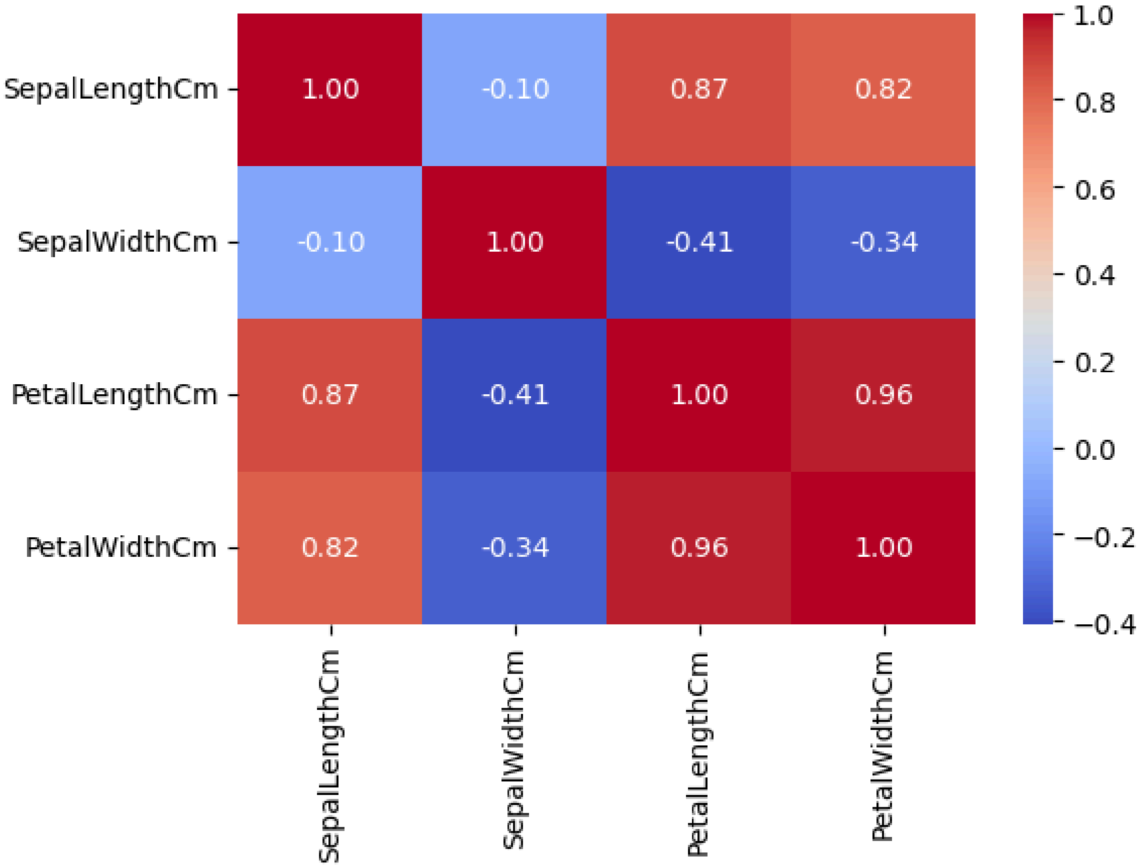
In [111…

```
corr_matrix = df[num_cols].corr()
print(corr_matrix)
```

```
               SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
SepalLengthCm       1.000000     -0.096690       0.872044      0.817492
SepalWidthCm       -0.096690      1.000000      -0.406541     -0.342267
PetalLengthCm       0.872044     -0.406541       1.000000      0.962865
PetalWidthCm        0.817492     -0.342267       0.962865      1.000000
```

In [112…

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,4))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.show()
```

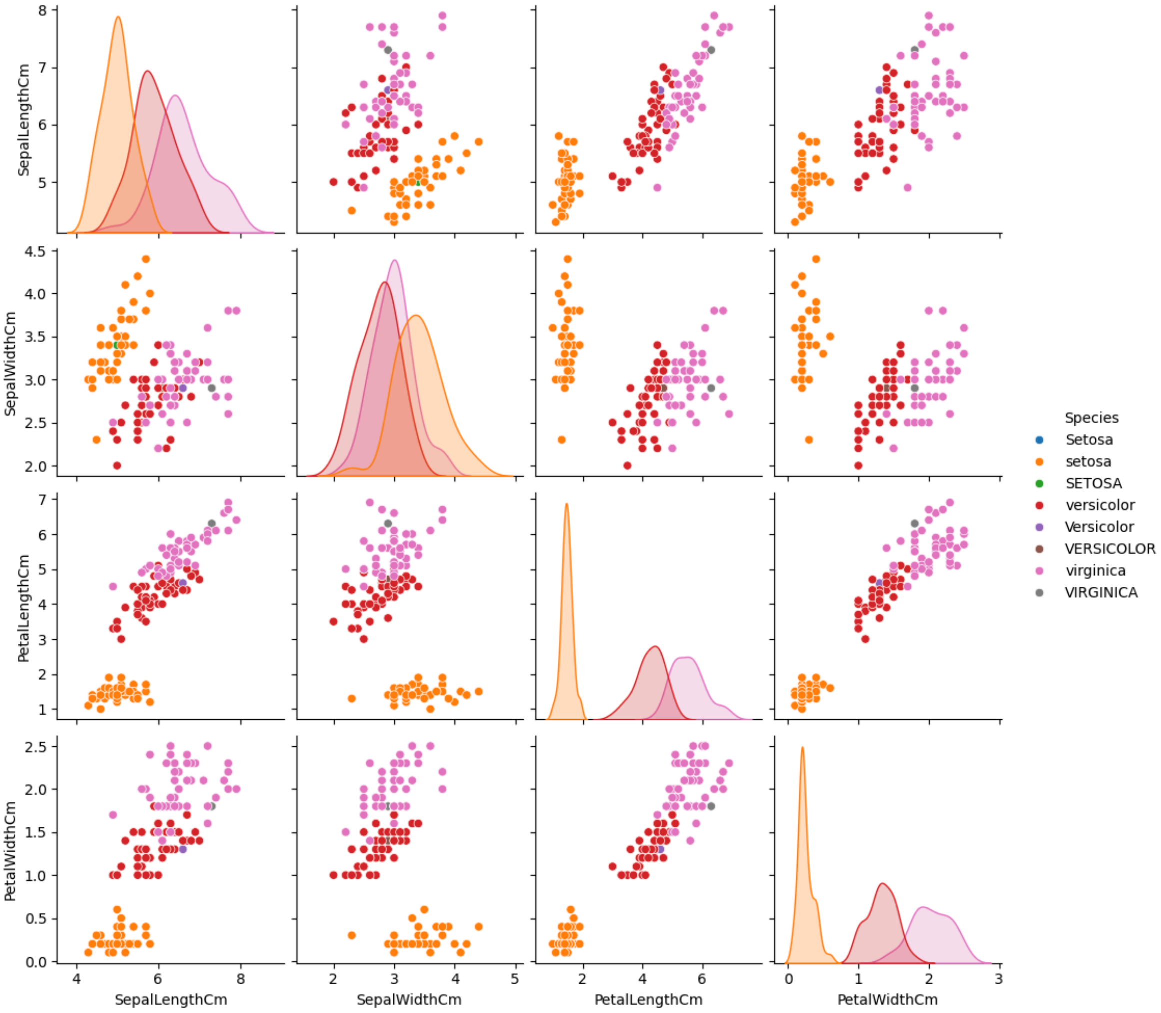


In [114…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Pick the useful columns
num_cols = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidth

# Pairplot with species as hue (color by species)
sns.pairplot(df[num_cols + ["Species"]], hue="Species")
```

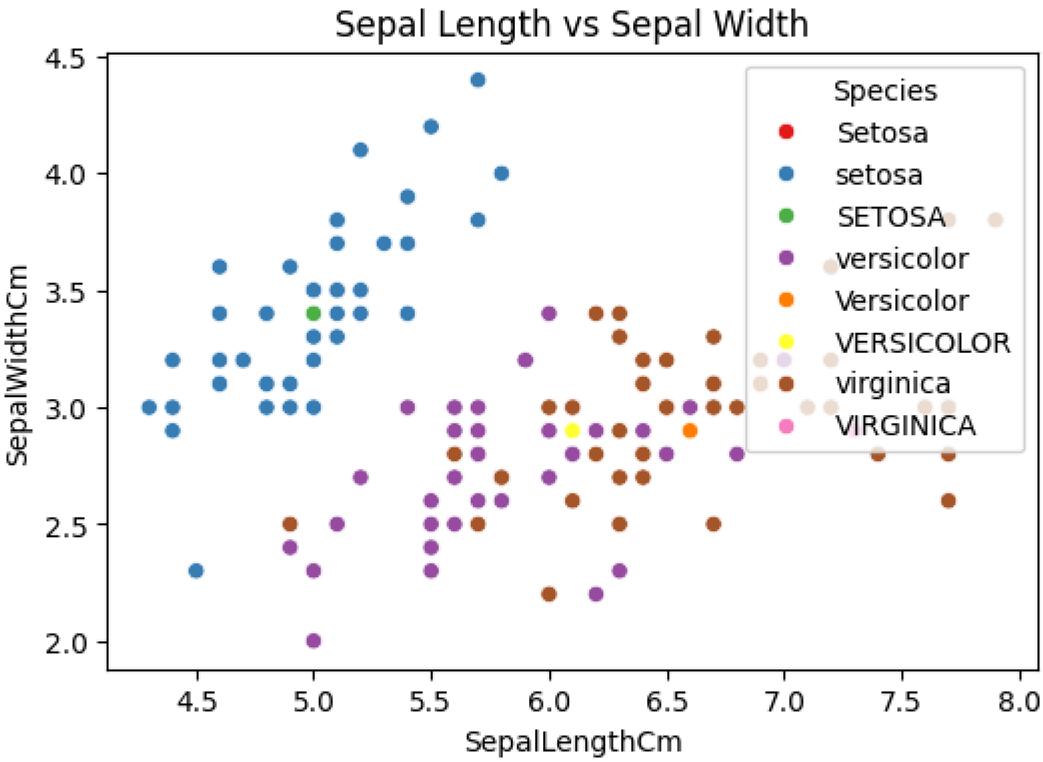
Out[114…

```
<seaborn.axisgrid.PairGrid at 0x12ce323c0>
```



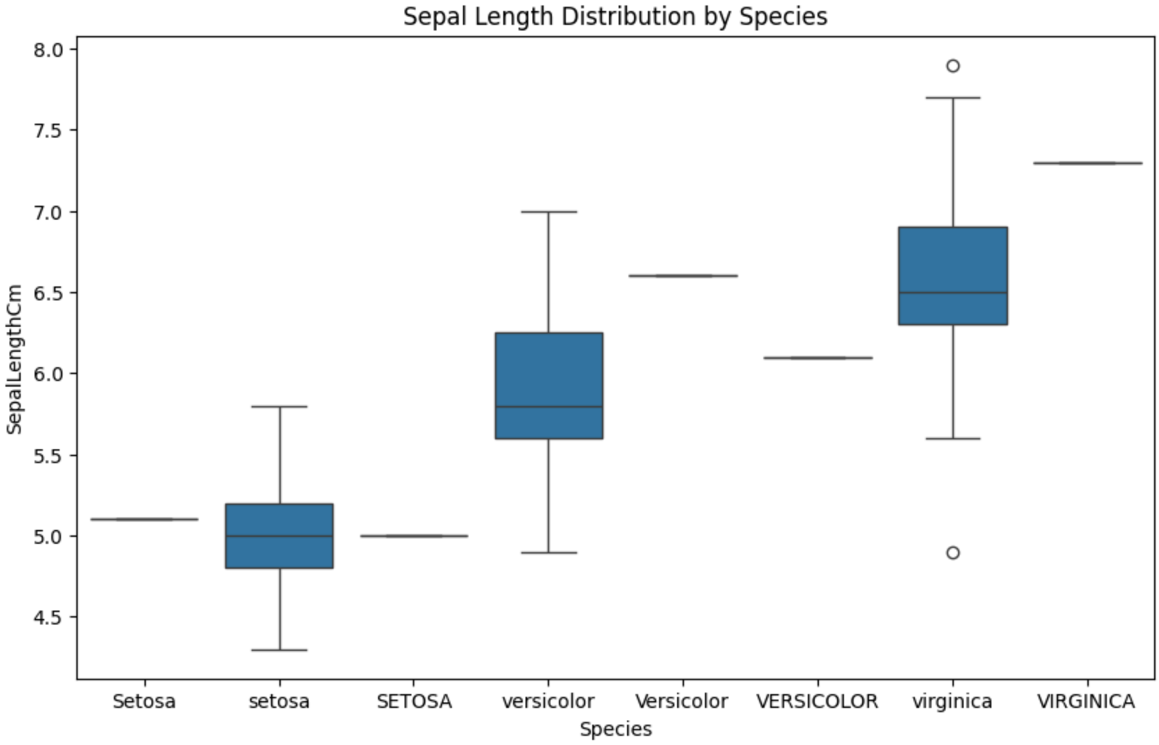
In [115…

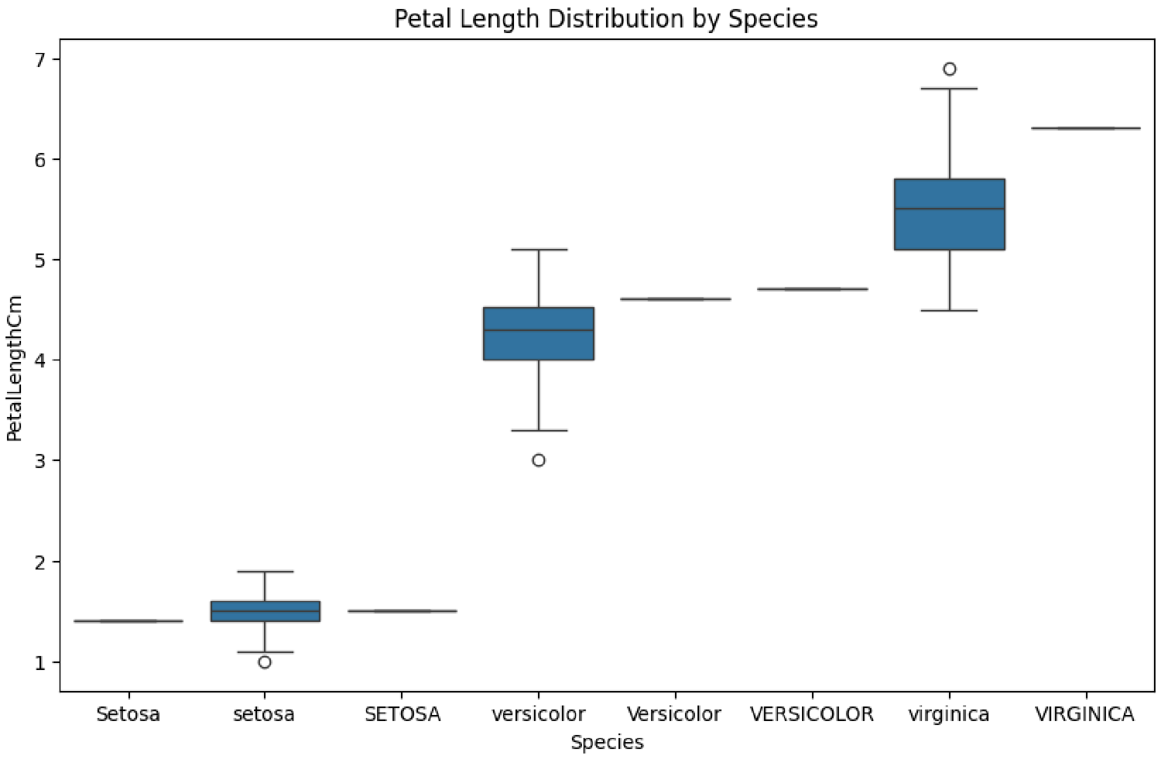
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(6,4))
sns.scatterplot(
    data=df,
    x="SepalLengthCm",
    y="SepalWidthCm",
    hue="Species",
    palette="Set1"
)
plt.title("Sepal Length vs Sepal Width")
plt.show()
```



In [116…

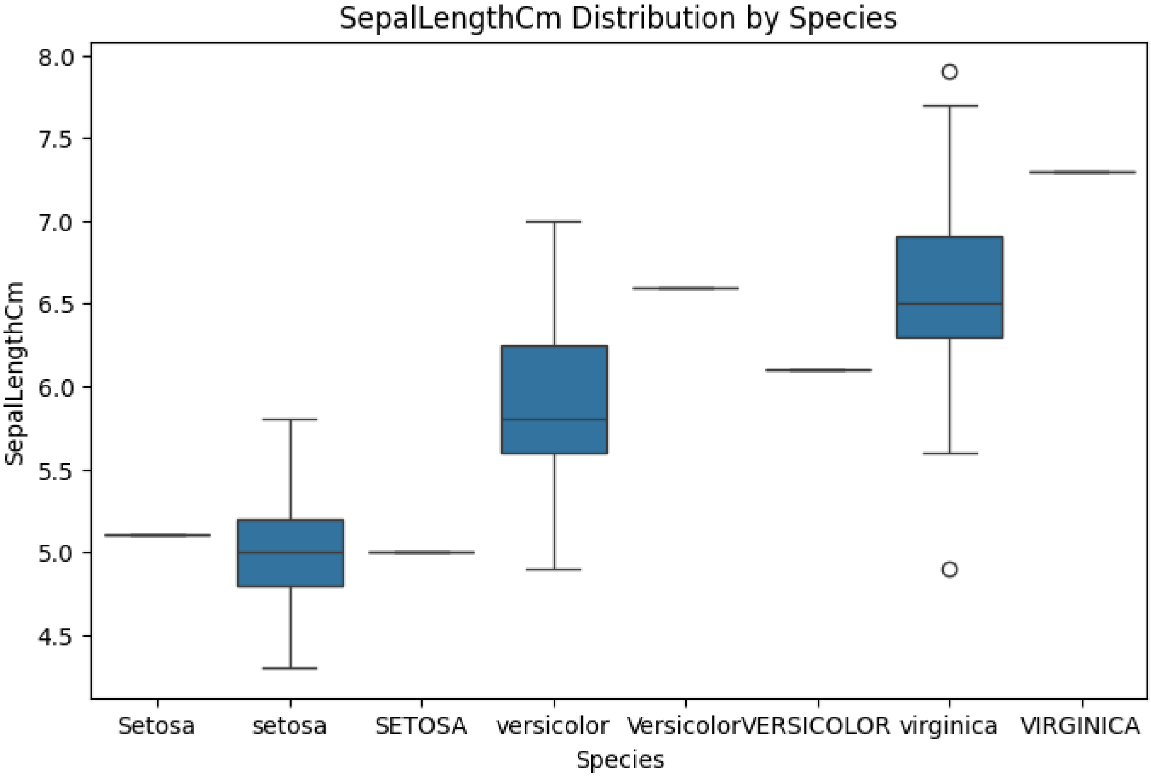
```
plt.figure(figsize=(10,6))
sns.boxplot(data=df, x="Species", y="SepalLengthCm")
plt.title("Sepal Length Distribution by Species")
plt.show()

plt.figure(figsize=(10,6))
sns.boxplot(data=df, x="Species", y="PetalLengthCm")
plt.title("Petal Length Distribution by Species")
plt.show()
```

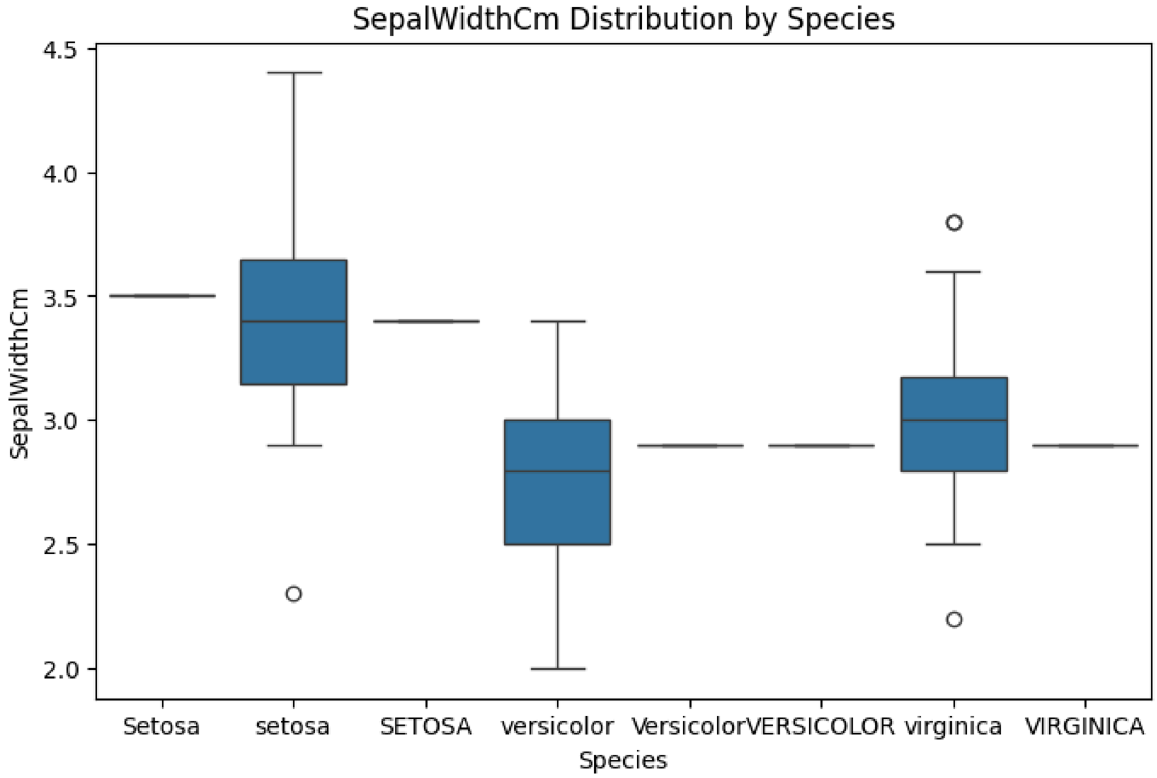


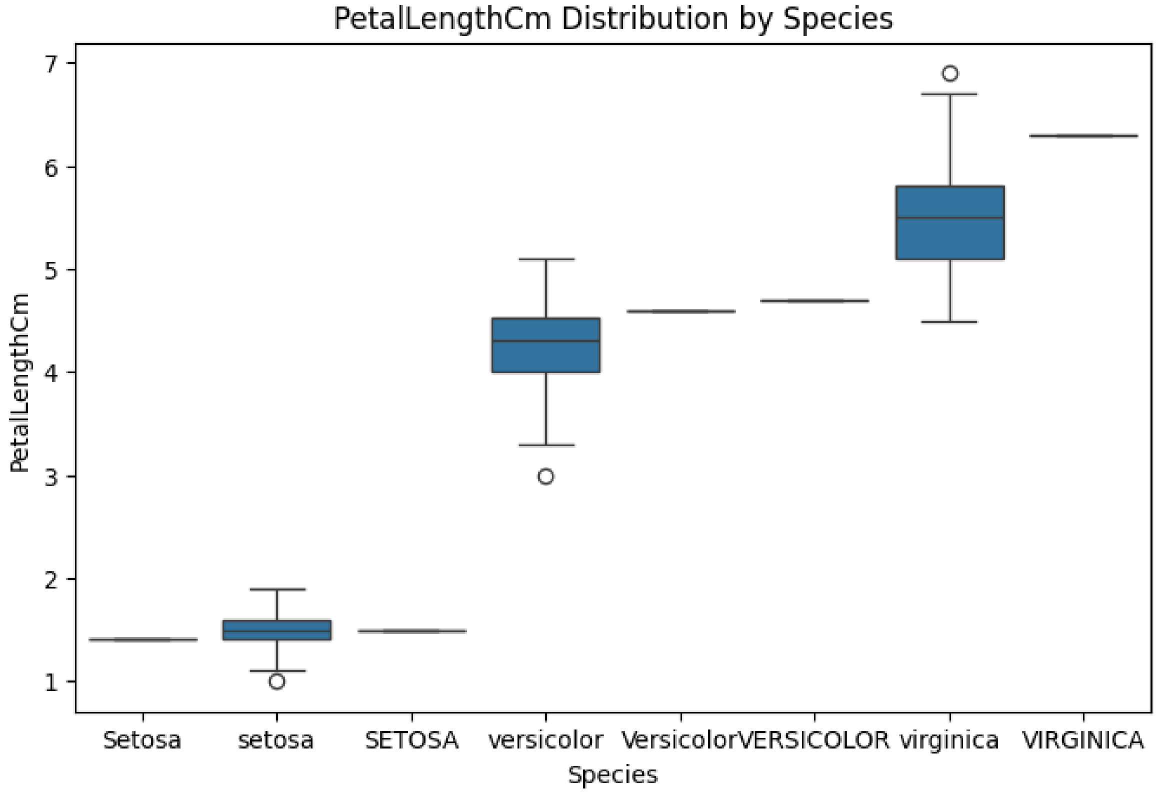


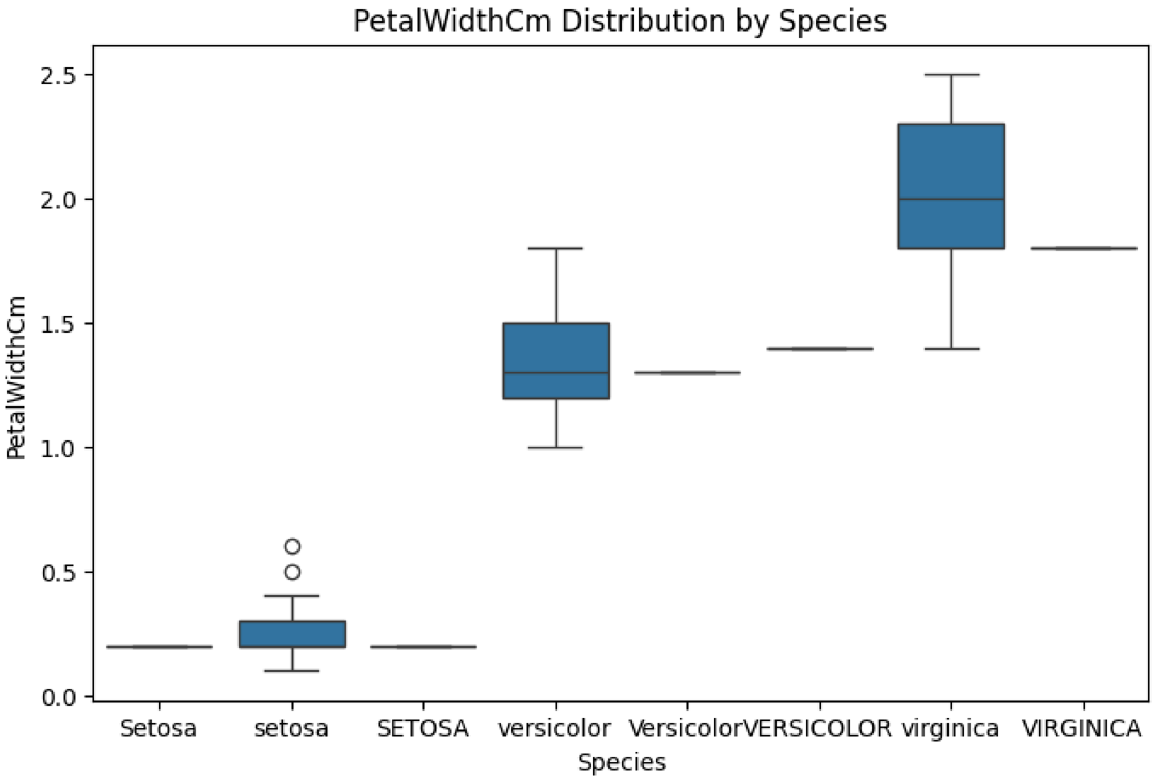
In [117…

```
num_cols = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidth

for col in num_cols:
    plt.figure(figsize=(8,5))
    sns.boxplot(data=df, x="Species", y=col)
    plt.title(f"{col} Distribution by Species")
    plt.show()
```




SepalWidthCm Distribution by Species
SepalWidthCm
4.5
4.0
3.5
3.0
2.5
2.0
Setosa
setosa
SETOSA
versicolor
Versicolor
VERSICOLOR
virginica
VIRGINICA
Species


PetalLengthCm Distribution by Species
PetalLengthCm
7
6
5
4
3
2
1
Setosa
setosa
SETOSA
versicolor
Versicolor
VERSICOLOR
virginica
VIRGINICA
Species


PetalWidthCm Distribution by Species
PetalWidthCm
Species
0.0
0.5
1.0
1.5
2.0
2.5
Setosa
setosa
SETOSA
versicolor
Versicolor
VERSICOLOR
virginica
VIRGINICA

```
In [ ]:
```