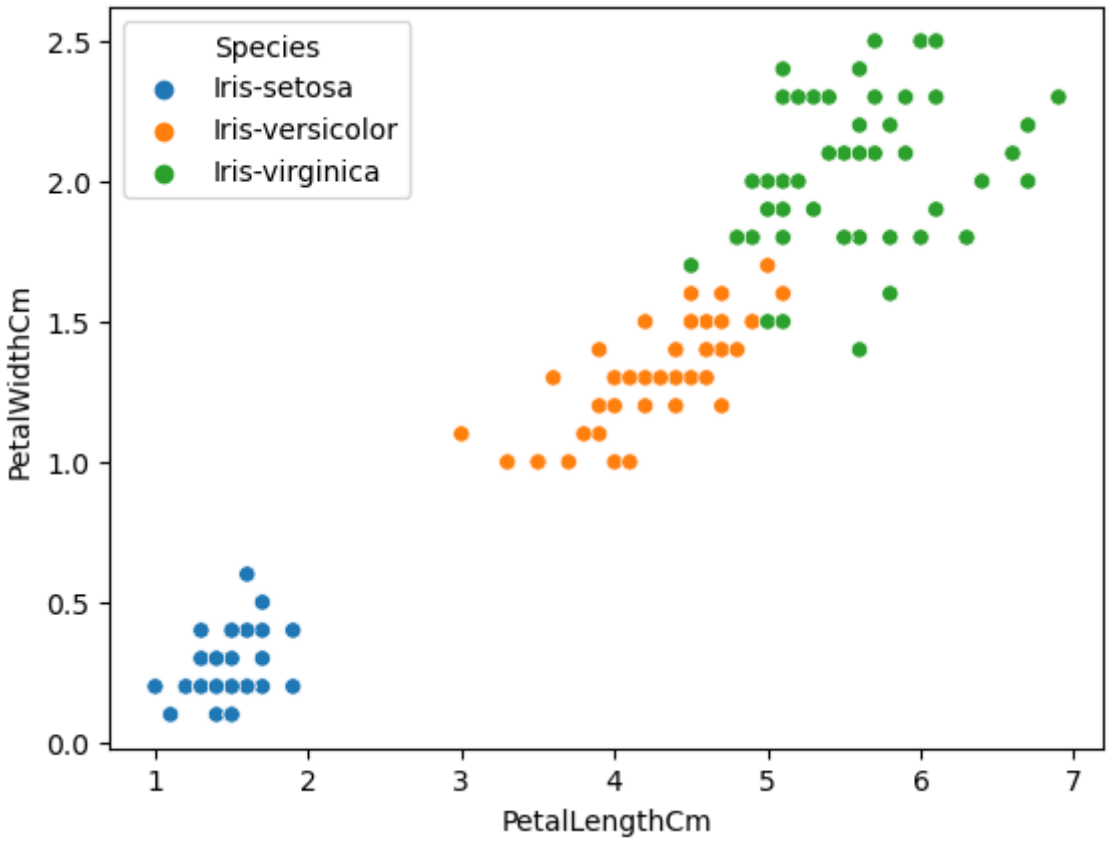
In [5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
iris_df = pd.read_csv("iris.csv")
```

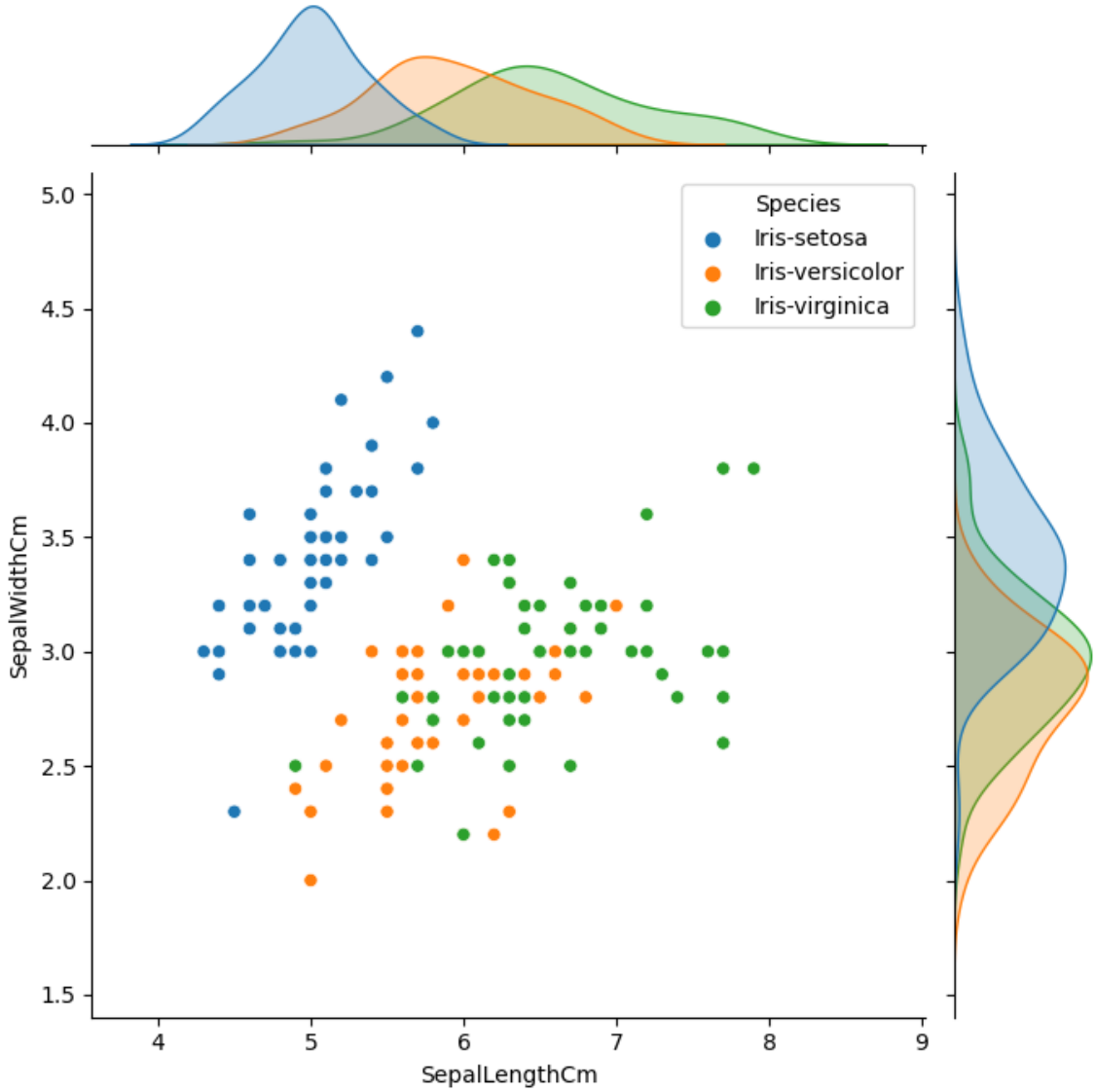
In [3]:

```
sns.scatterplot(iris_df["PetalLengthCm"], iris_df["PetalWidthCm"], hue=iris_df["Species"]
plt.show()
```



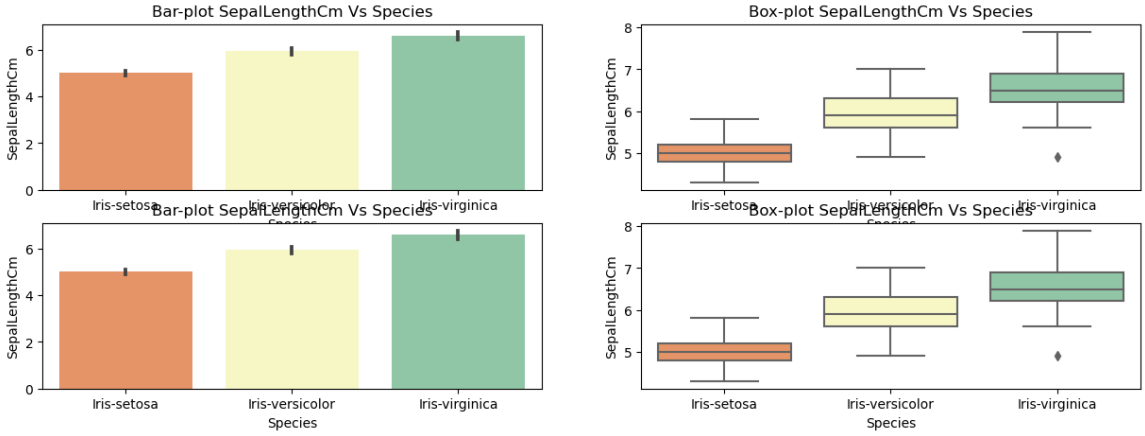
In [6]:

```
sns.jointplot(data = iris_df, x="SepalLengthCm", y="SepalWidthCm", size=7, hue = "Species
plt.show()
```



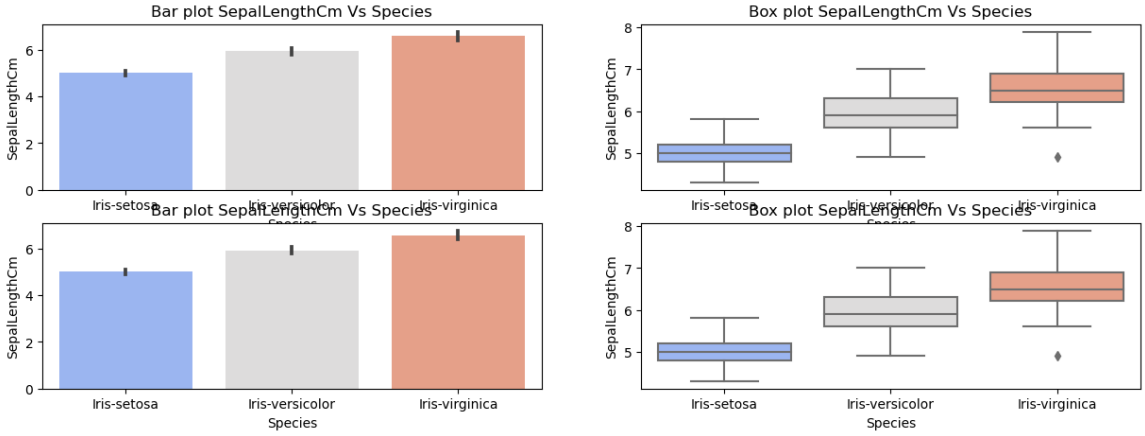
In [7]:

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

plt.figure(figsize=(15, 5))

plt.subplot(2, 2, 1)
sns.barplot(x="Species", y="SepalLengthCm", data=iris_df, palette="Spectral")
plt.title("Bar-plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 2)
sns.boxplot(x="Species", y="SepalLengthCm", data=iris_df, palette="Spectral")
plt.title("Box-plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 3)
sns.barplot(x="Species", y="SepalLengthCm", data=iris_df, palette="Spectral")
plt.title("Bar-plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 4)
sns.boxplot(x="Species", y="SepalLengthCm", data=iris_df, palette="Spectral")
plt.title("Box-plot SepalLengthCm Vs Species")

plt.show()
```



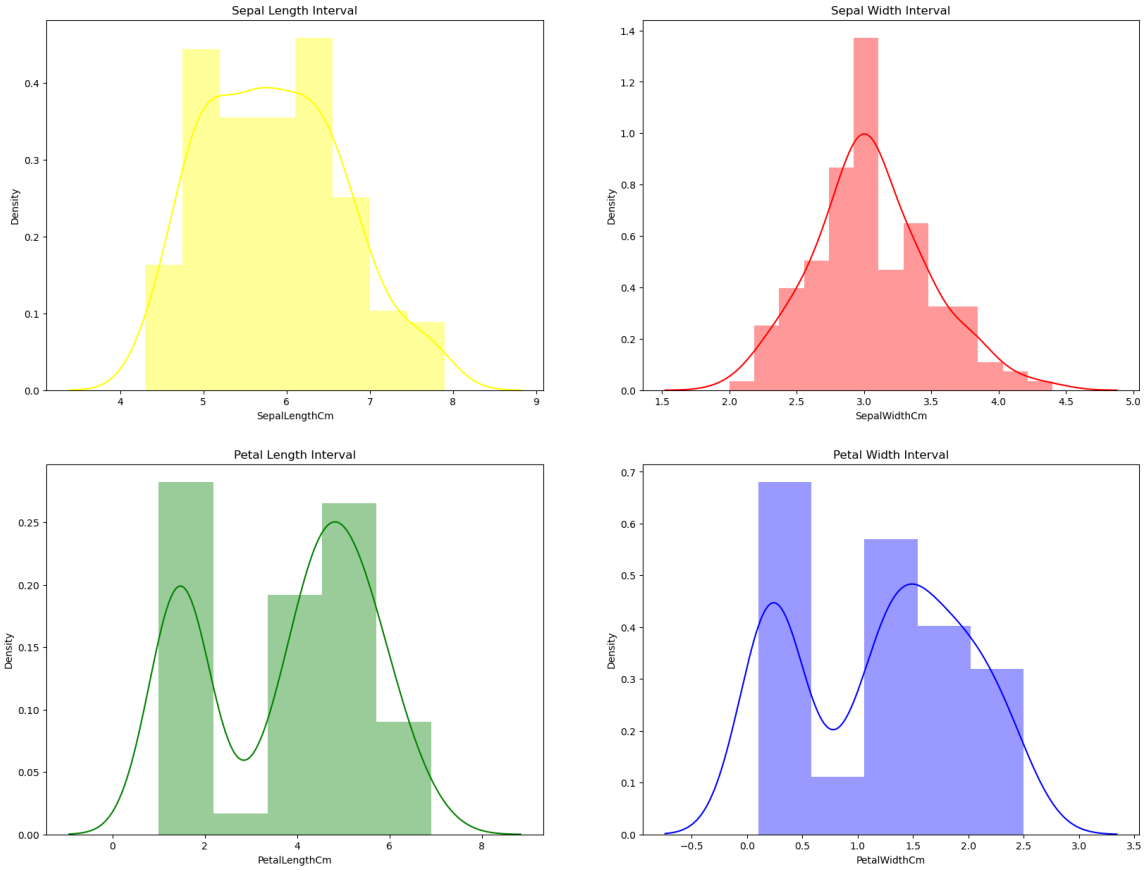
In [8]:

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

plt.figure(figsize=(15, 5))

plt.subplot(2, 2, 1)
sns.barplot(x="Species", y="SepalLengthCm", data=iris_df, palette="coolwarm")
plt.title("Bar plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 2)
sns.boxplot(x="Species", y="SepalLengthCm", data=iris_df, palette="coolwarm")
plt.title("Box plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 3)
sns.barplot(x="Species", y="SepalLengthCm", data=iris_df, palette="coolwarm")
plt.title("Bar plot SepalLengthCm Vs Species")

plt.subplot(2, 2, 4)
sns.boxplot(x="Species", y="SepalLengthCm", data=iris_df, palette="coolwarm")
plt.title("Box plot SepalLengthCm Vs Species")

plt.show()
```



In [9]:

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

plt.figure(figsize=(20, 15))

plt.subplot(2, 2, 1)
sns.distplot(iris_df["SepalLengthCm"], color="yellow").set_title("Sepal Length Interval")

plt.subplot(2, 2, 2)
sns.distplot(iris_df["SepalWidthCm"], color="red").set_title("Sepal Width Interval")

plt.subplot(2, 2, 3)
sns.distplot(iris_df["PetalLengthCm"], color="green").set_title("Petal Length Interval")

plt.subplot(2, 2, 4)
sns.distplot(iris_df["PetalWidthCm"], color="blue").set_title("Petal Width Interval")

plt.show()
```



In [7]:

```
from sklearn.preprocessing import LabelEncoder

# Create a LabelEncoder instance
le = LabelEncoder()

# Fit and transform the "Species" column
iris_df["Species"] = le.fit_transform(iris_df["Species"])

# Display the updated DataFrame
iris_df.head()
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

In [8]:

```
unique_species = iris_df['Species'].unique()
print(unique_species)
```

```
[0 1 2]
```

In [9]:

```
# Select the features (X) from the first 4 columns of the DataFrame
X = iris_df.iloc[:, [0, 1, 2, 3]]

# Display the first few rows of X
print(X.head())

# Select the target variable (y) from the last column of the DataFrame
y = iris_df.iloc[:, -1]

# Display the first few rows of y
print(y.head())

# Print the shape of X and y
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm
0   1            5.1           3.5            1.4
1   2            4.9           3.0            1.4
2   3            4.7           3.2            1.3
3   4            4.6           3.1            1.5
4   5            5.0           3.6            1.4
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
Shape of X: (150, 4)
Shape of y: (150,)
```

In [11]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [12]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Create a Logistic Regression model
lr = LogisticRegression()

# Fit the model to the training data
lr.fit(x_train, y_train)

print("Logistic regression successfully implemented")

# Predict on the test data
y_pred = lr.predict(x_test)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)

# Accuracy Score
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy is:", accuracy * 100)

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Logistic regression successfully implemented
Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy is: 100.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [15]:

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier

# Create and train a Decision Tree model
dtree = DecisionTreeClassifier()
dtree.fit(x_train, y_train)  # Assuming x_train and y_train are your training data

# Now you can use the model to make predictions as shown in the previous code.

input_data = (4.9, 3.0, 1.4, 0.2)

# Convert the input data to a numpy array
input_data_as_nparray = np.asarray(input_data)

# Reshape the data as we are predicting the label for only one instance
input_data_reshaped = input_data_as_nparray.reshape(1, -1)

# Make the prediction using the decision tree model
prediction = dtree.predict(input_data_reshaped)

print("The category is:", prediction)
```

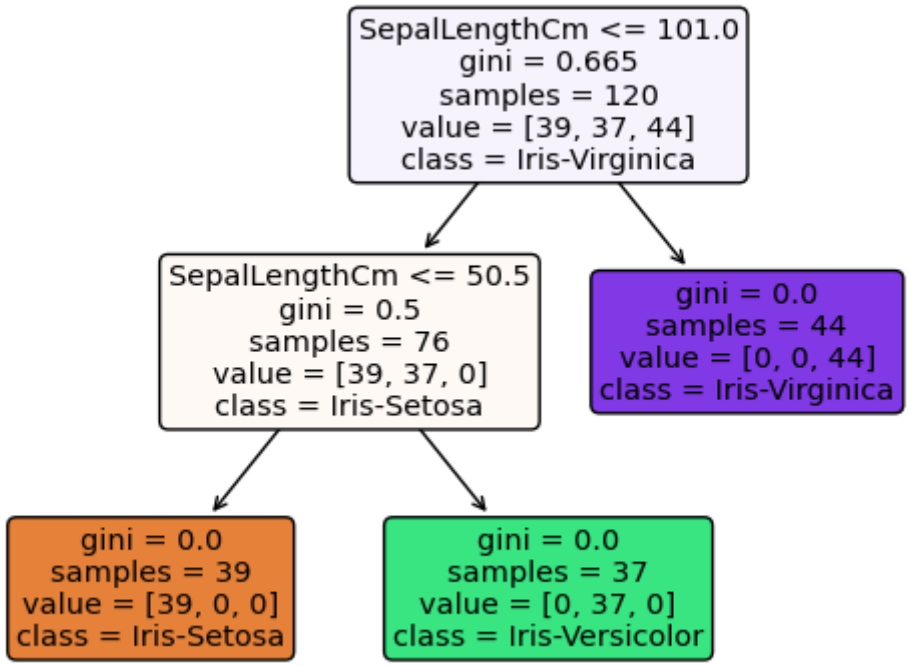
```
The category is: [0]
```

In [16]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Create and train a Decision Tree model
dtree = DecisionTreeClassifier()
dtree.fit(x_train, y_train)  # Replace x_train and y_train with your actual training data

# Define the feature names and class names
feature_names = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]
class_names = ['Iris-Setosa', 'Iris-Versicolor', 'Iris-Virginica']

# Visualize the Decision Tree
plot_tree(dtree, feature_names=feature_names, class_names=class_names, filled=True, round

# You may need to import the necessary libraries for visualization, such as matplotlib.
# plt.show()  # Uncomment this line if you want to display the tree visualization using m
```

Out[16]:

```
[Text(0.6, 0.8333333333333334, 'SepalLengthCm <= 101.0\ngini = 0.665\nsamp
les = 120\nvalue = [39, 37, 44]\nclass = Iris-Virginica'),
 Text(0.4, 0.5, 'SepalLengthCm <= 50.5\ngini = 0.5\nsamples = 76\nvalue =
[39, 37, 0]\nclass = Iris-Setosa'),
 Text(0.2, 0.16666666666666666, 'gini = 0.0\nsamples = 39\nvalue = [39, 0,
0]\nclass = Iris-Setosa'),
 Text(0.6, 0.16666666666666666, 'gini = 0.0\nsamples = 37\nvalue = [0, 37,
0]\nclass = Iris-Versicolor'),
 Text(0.8, 0.5, 'gini = 0.0\nsamples = 44\nvalue = [0, 0, 44]\nclass = Iri
s-Virginica')]
```



In [17]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Create and train a Decision Tree model
dtree = DecisionTreeClassifier()
dtree.fit(x_train, y_train)  # Replace x_train and y_train with your actual training data

# Define the feature names and class names
feature_names = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]
class_names = ['Iris-Setosa', 'Iris-Versicolor', 'Iris-Virginica']

# Visualize the Decision Tree
plt.figure(figsize=(10, 10))
plot_tree(dtree, feature_names=feature_names, class_names=class_names, filled=True)

plt.show()
```