IRIS DATASET VISUALIZATION(SEABORN,MATPLOTLIB)

In [2]:

```
# importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
# importing iris dataset

iris=pd.read_csv(r"C:\Users\Jan Saida\OneDrive\Documents\Desktop\Excel sheets\Iris.csv")
iris
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [5]:

```
iris.head()
```

Out[5]:

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

In [6]:

```
iris.drop('Id',axis=1, inplace = True)
```

In [7]:
```
iris.head()
```

Out[7]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

checking if there any missing values

In [9]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [10]:
```
iris['Species'].value_counts()
```

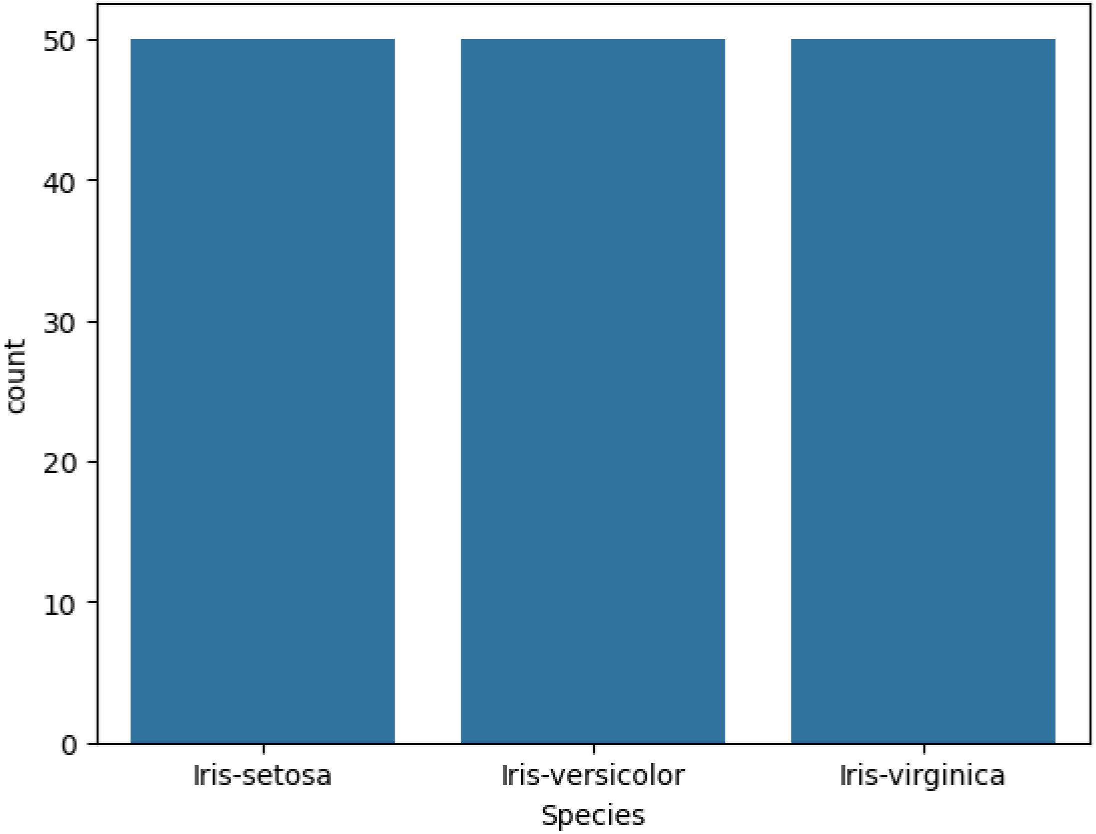
Out[10]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

This dataset has three varities of iris plant

2.Bar Plot :

Here the frequency of the observation is plotted.In this case we are plotting the frequency of the three species in the Iris Dataset

In [13]:
```
sns.countplot(x ='Species',data=iris)
plt.show()
# Use x='Species' for horizontal bars
```



We can see that there are 50 samples each of all the Iris Species in the data set.

3.Joint Plot :

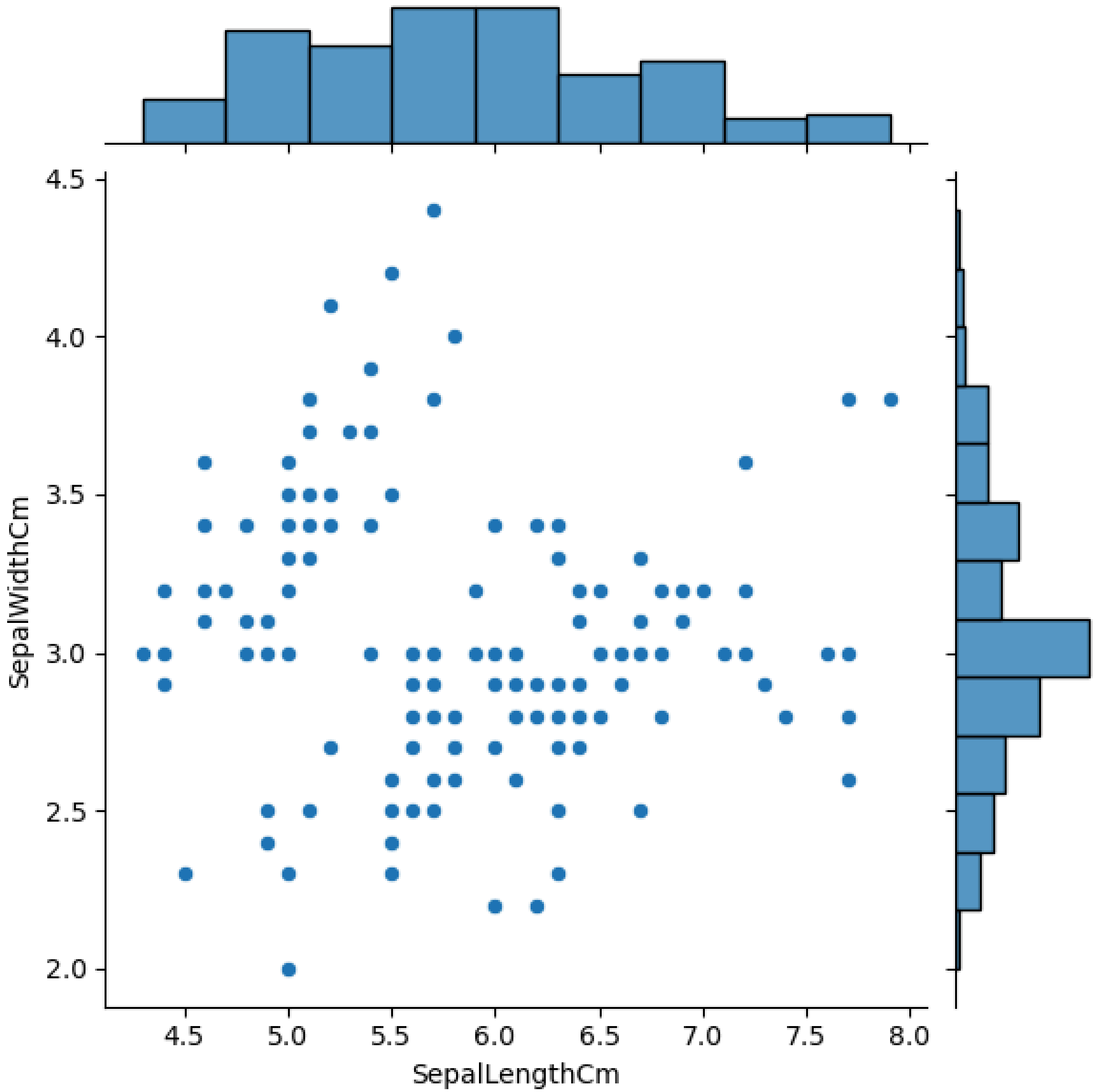
Jointplot is seaborn library specific and can be used to quickly visualize and analyze the relationship between two variables and describe their individual distributions on the same plot.

In [16]:
```
iris.head()
```

Out[16]:

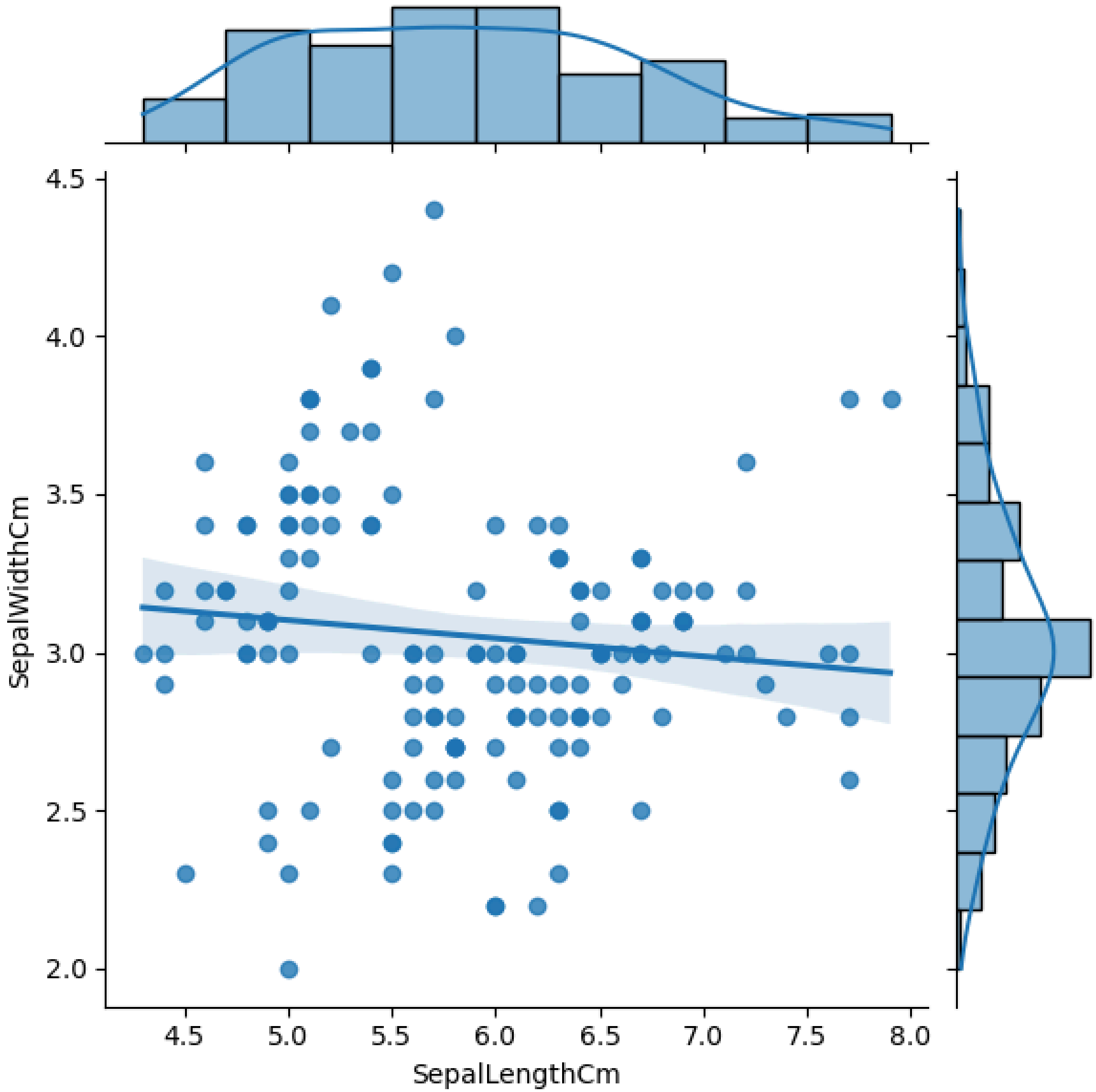
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [17]:
```
fig = sns.jointplot(x = 'SepalLengthCm',y = 'SepalWidthCm', data = iris)
```

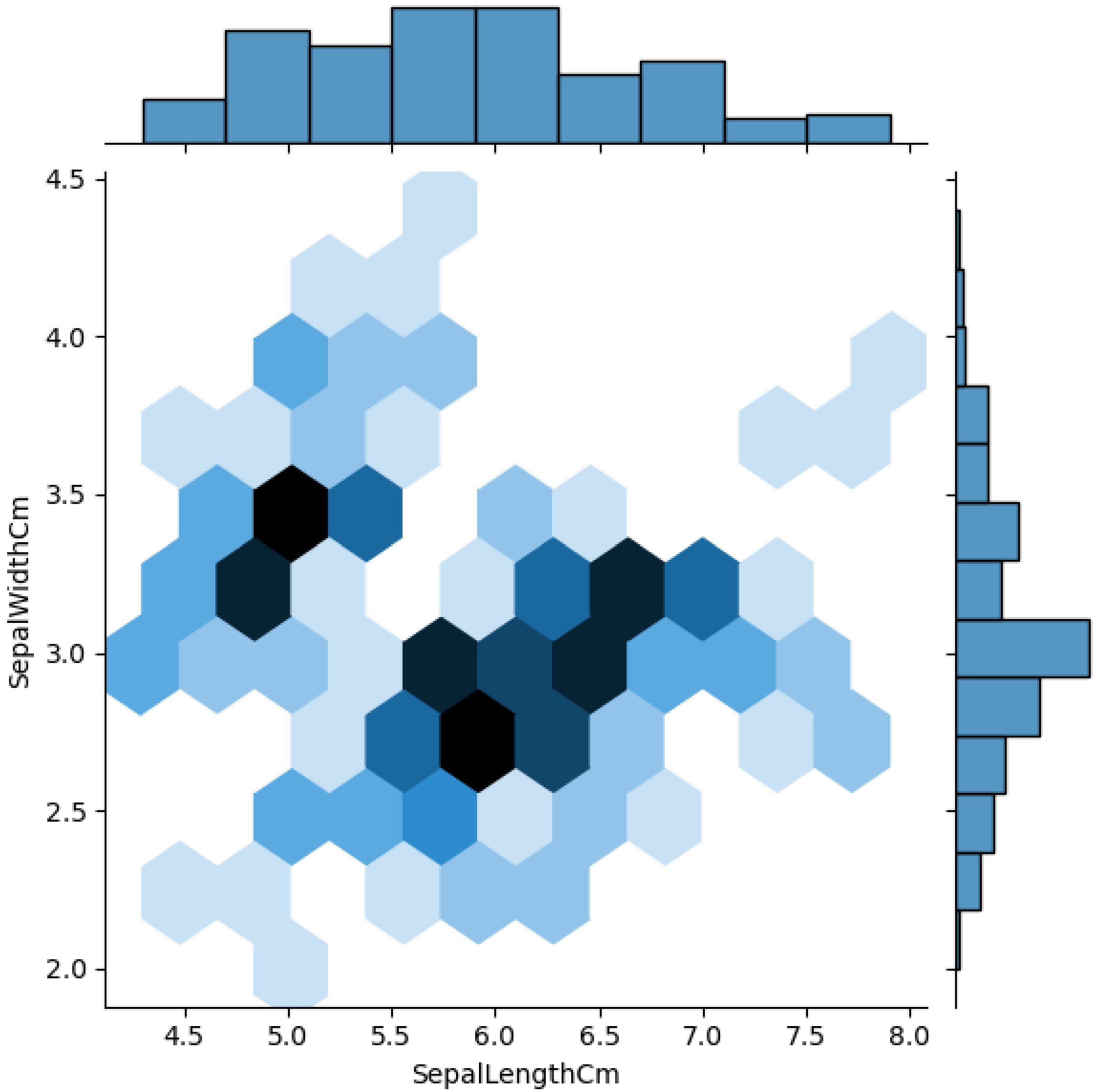


In [18]:
```
sns.jointplot(x = "SepalLengthCm", y = "SepalWidthCm", data=iris, kind="reg")

# use axis as x = [column] and y = [column]
# otherwise it will show error sometimes
```

Out[18]: <seaborn.axisgrid.JointGrid at 0x2042a835fa0>



In [19]:
```
fig=sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',kind='hex',data=iris)
```

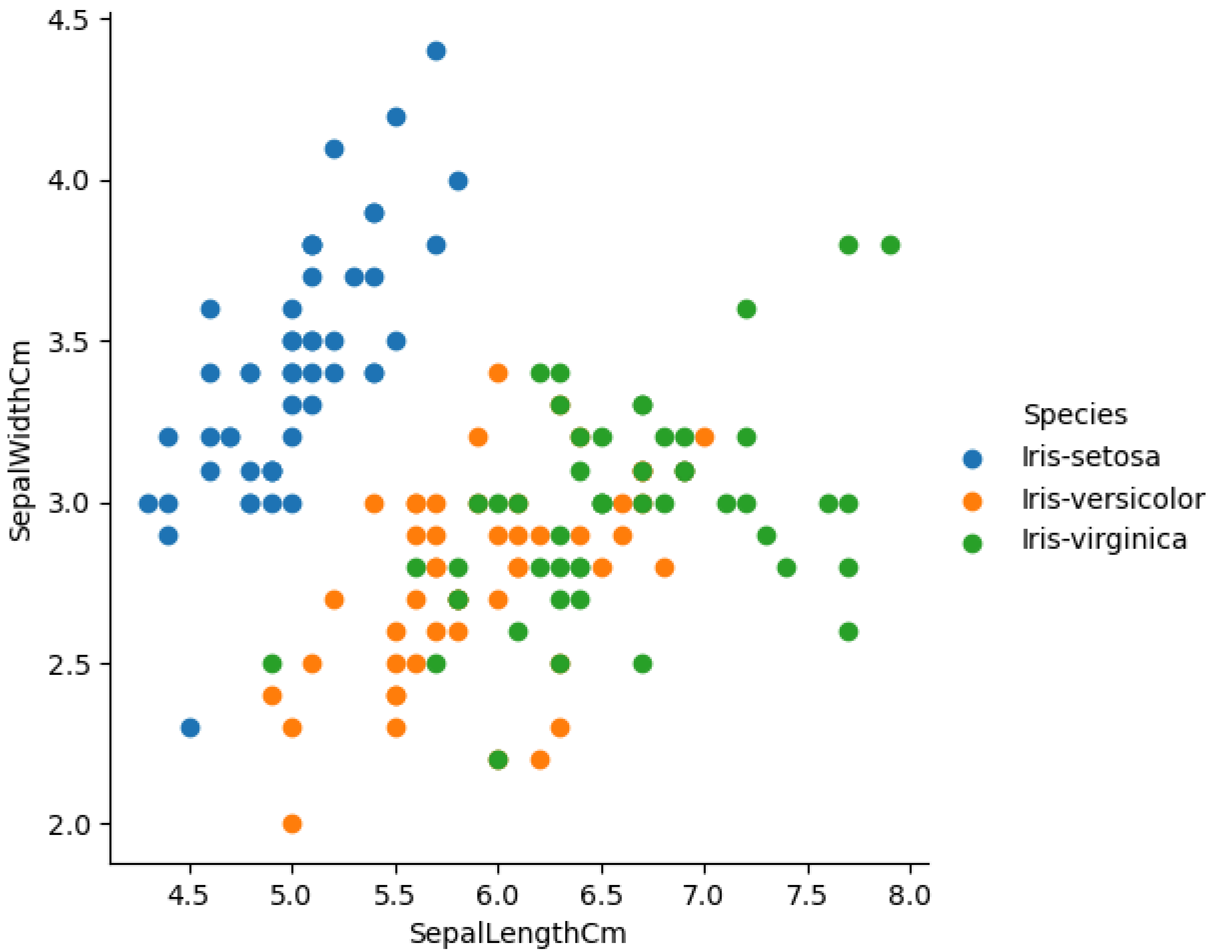


4. FacetGrid Plot

In [21]:

```
import matplotlib.pyplot as plt
%matplotlib inline

# Assuming you have the 'iris' dataset loaded
sns.FacetGrid(iris, hue='Species', height=5) \
    .map(plt.scatter, 'SepalLengthCm', 'SepalWidthCm') \
    .add_legend()
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x2042a40da00>



5. Boxplot or Whisker plot

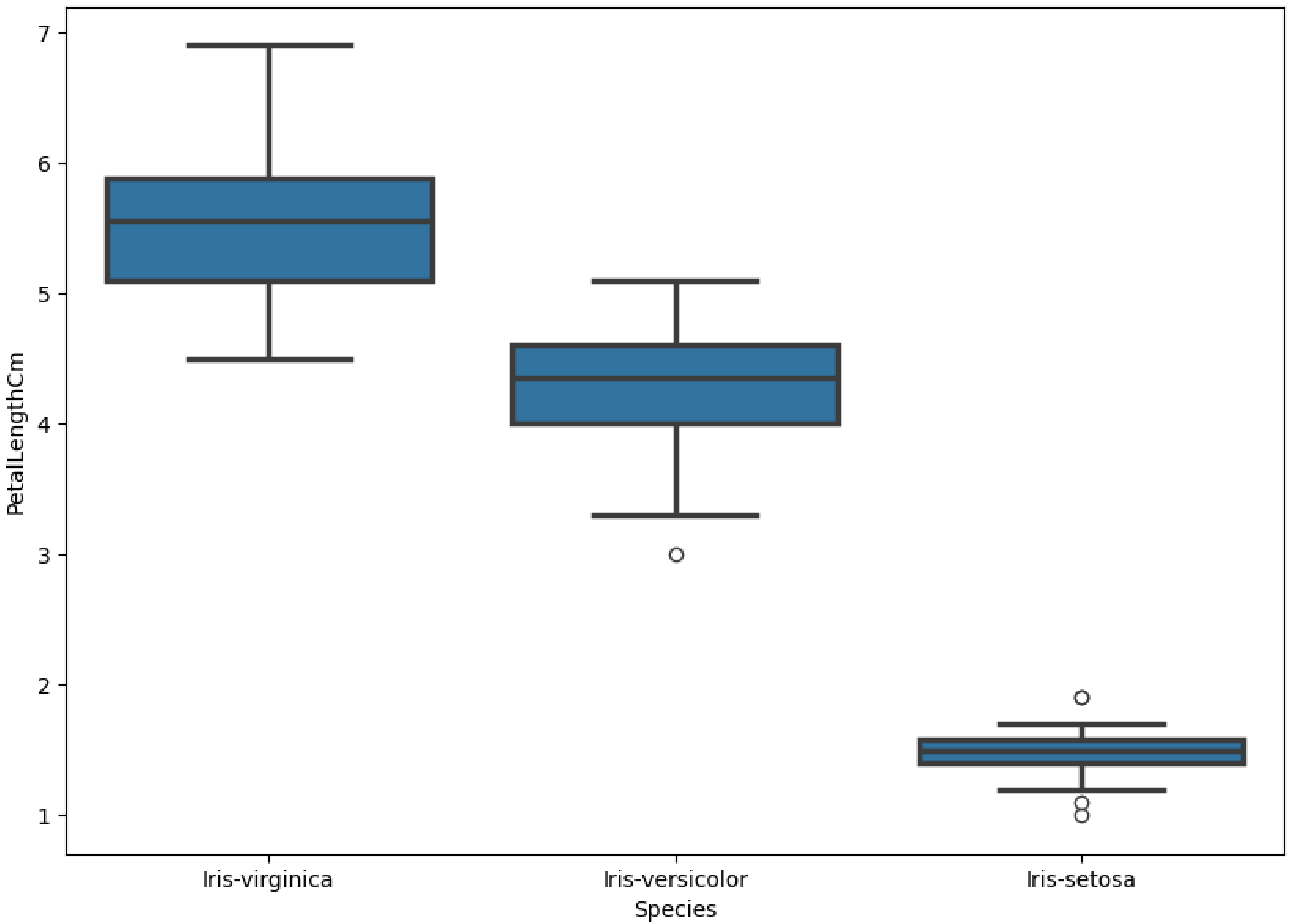
Box plot was was first introduced in year 1969 by Mathematician John Tukey.Box plot give a statical summary of the features being plotted.Top line represent the max value,top edge of box is third Quartile, middle edge represents the median,bottom edge represents the first quartile value.The bottom most line respresent the minimum value of the feature.The height of the box is called as Interquartile range.The black dots on the plot represent the outlier values in the data.

In [23]:
```
iris.head()
```

Out[23]:

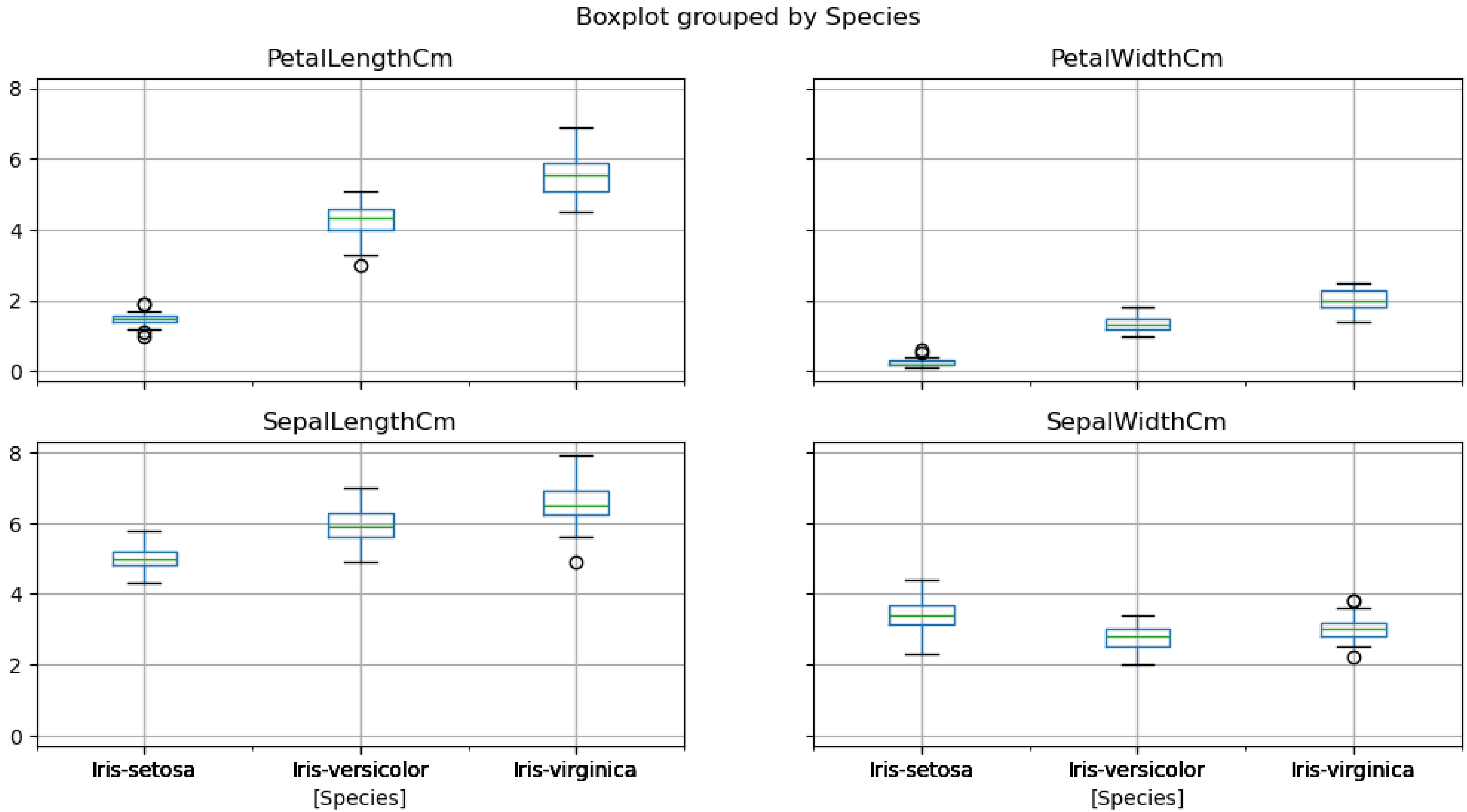
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [24]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxplot(x='Species',y='PetalLengthCm',data=iris,order=['Iris-virginica','Iris-versicolor','Iris-setosa'],linewidth=2.5
```



In [25]:
```
#iris.drop("Id", axis=1).boxplot(by="Species", figsize=(12, 6))
iris.boxplot(by="Species", figsize=(12, 6))
```

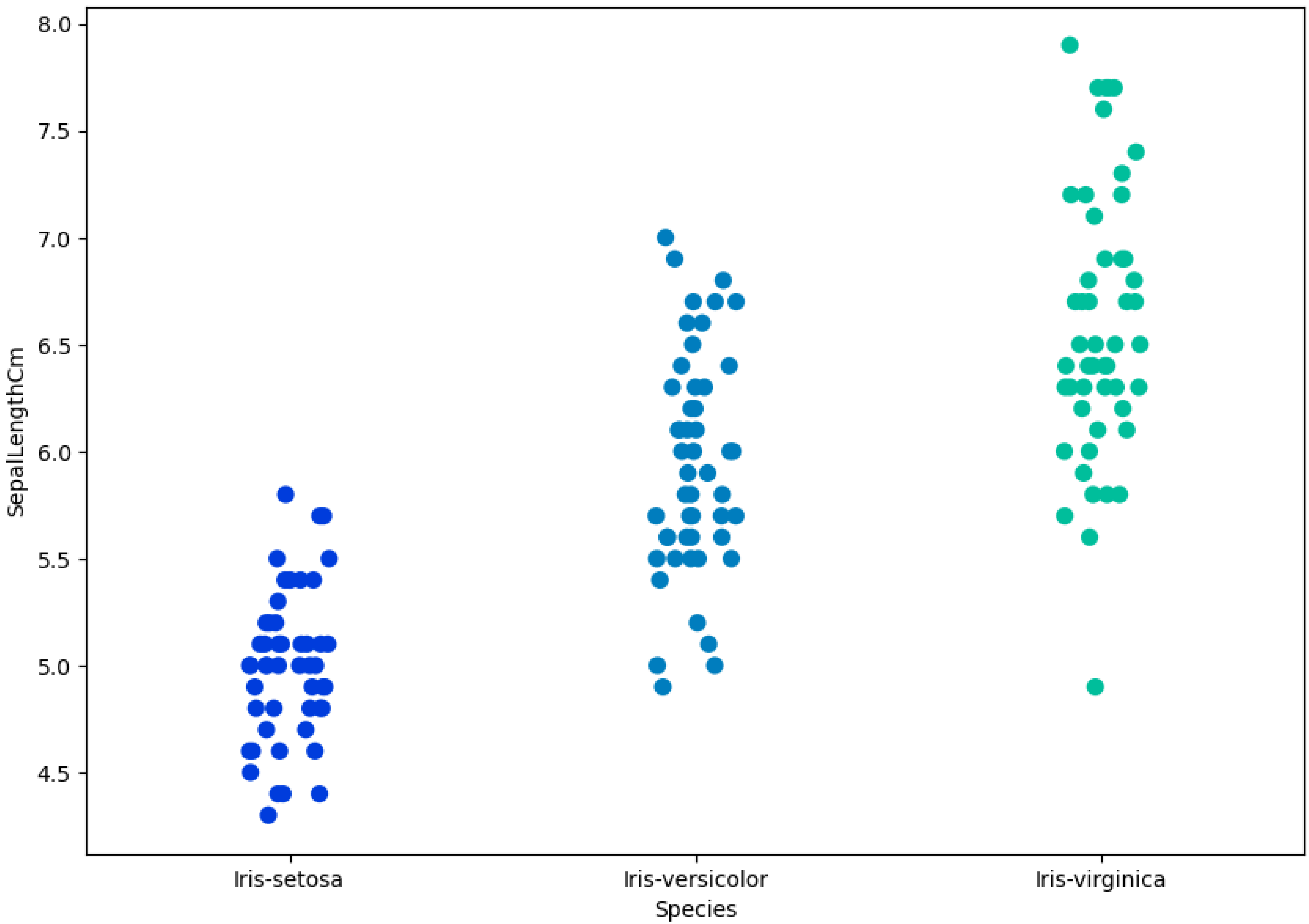
Out[25]:
```
array([[<Axes: title={'center': 'PetalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'PetalWidthCm'}, xlabel='[Species]'>],
       [<Axes: title={'center': 'SepalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



6. Strip plot

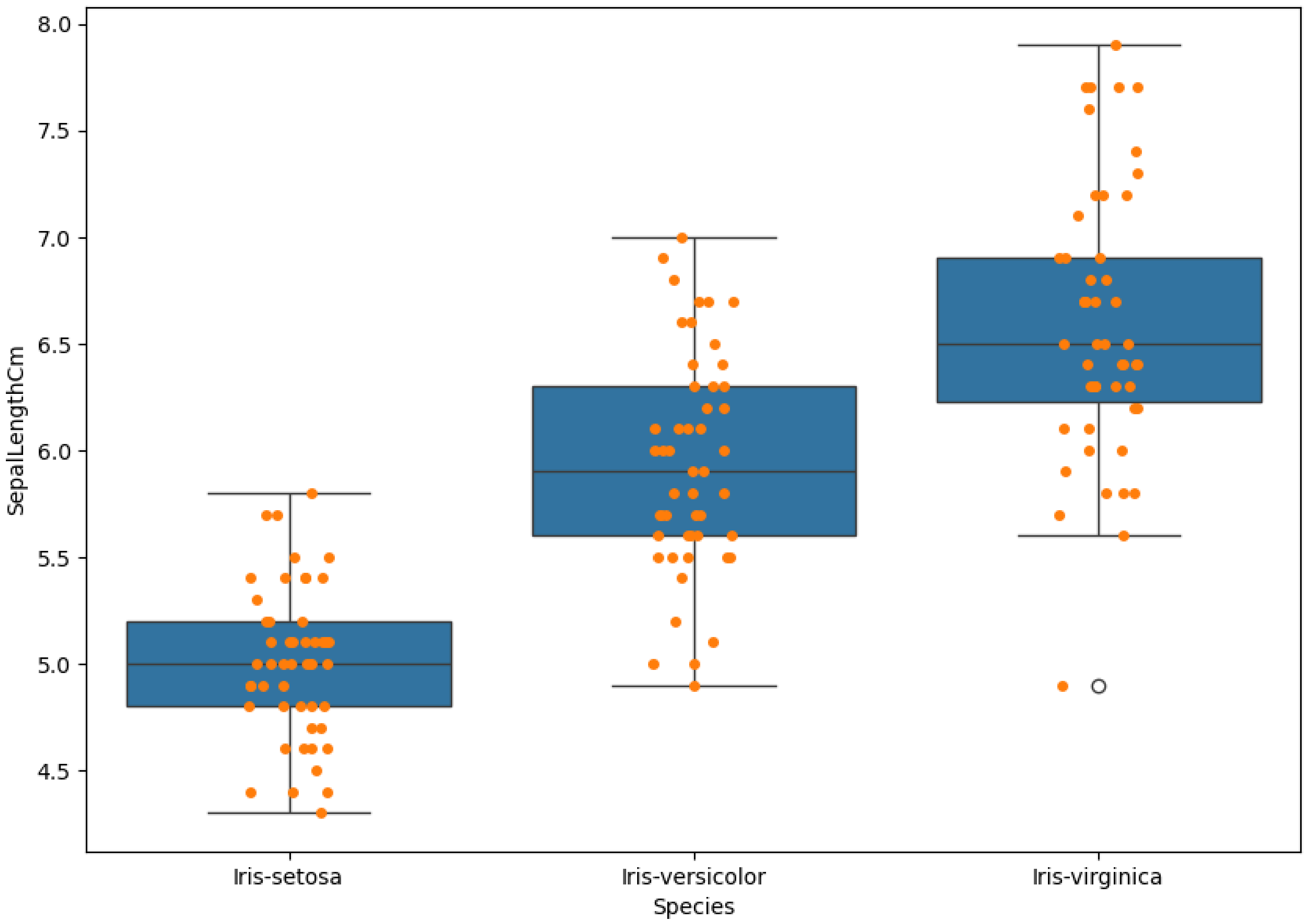
In [27]:

```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor='gray',size=8,palette='winter',orient='v')
```



7. Combining Box and Strip Plots

In [29]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxplot(x='Species',y='SepalLengthCm',data=iris)
fig=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor='gray')
```

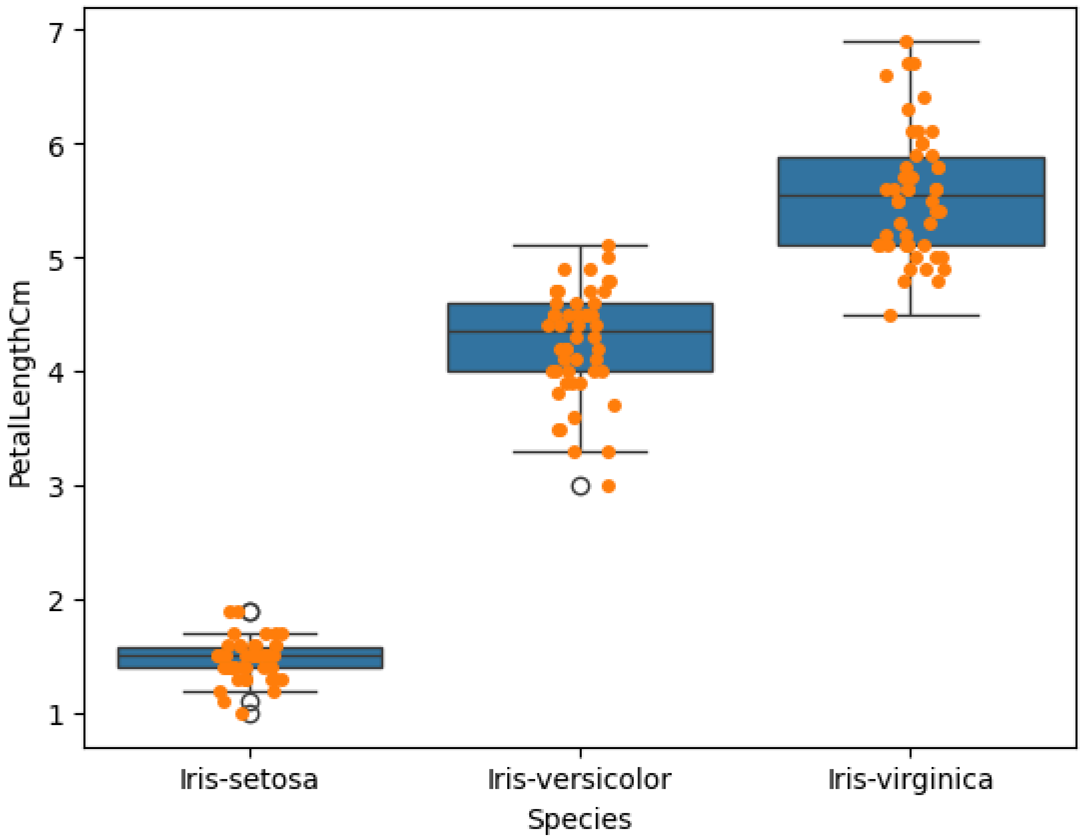


In [30]:

```
# Create the boxplot and stripplot
ax = sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
ax = sns.stripplot(x="Species", y="PetalLengthCm", data=iris, jitter=True, edgecolor="gray")
```

```python
# Print the artists to check how many boxes exist
print(len(ax.artists))  # Check how many boxes there are

# Safely modify the box colors if they exist
if len(ax.artists) > 0:
    boxone = ax.artists[0]
    boxone.set_facecolor('green')
    boxone.set_edgecolor('black')

if len(ax.artists) > 1:
    boxtwo = ax.artists[1]
    boxtwo.set_facecolor('yellow')
    boxtwo.set_edgecolor('black')

if len(ax.artists) > 2:
    boxthree = ax.artists[2]
    boxthree.set_facecolor('red')
    boxthree.set_edgecolor('black')
```

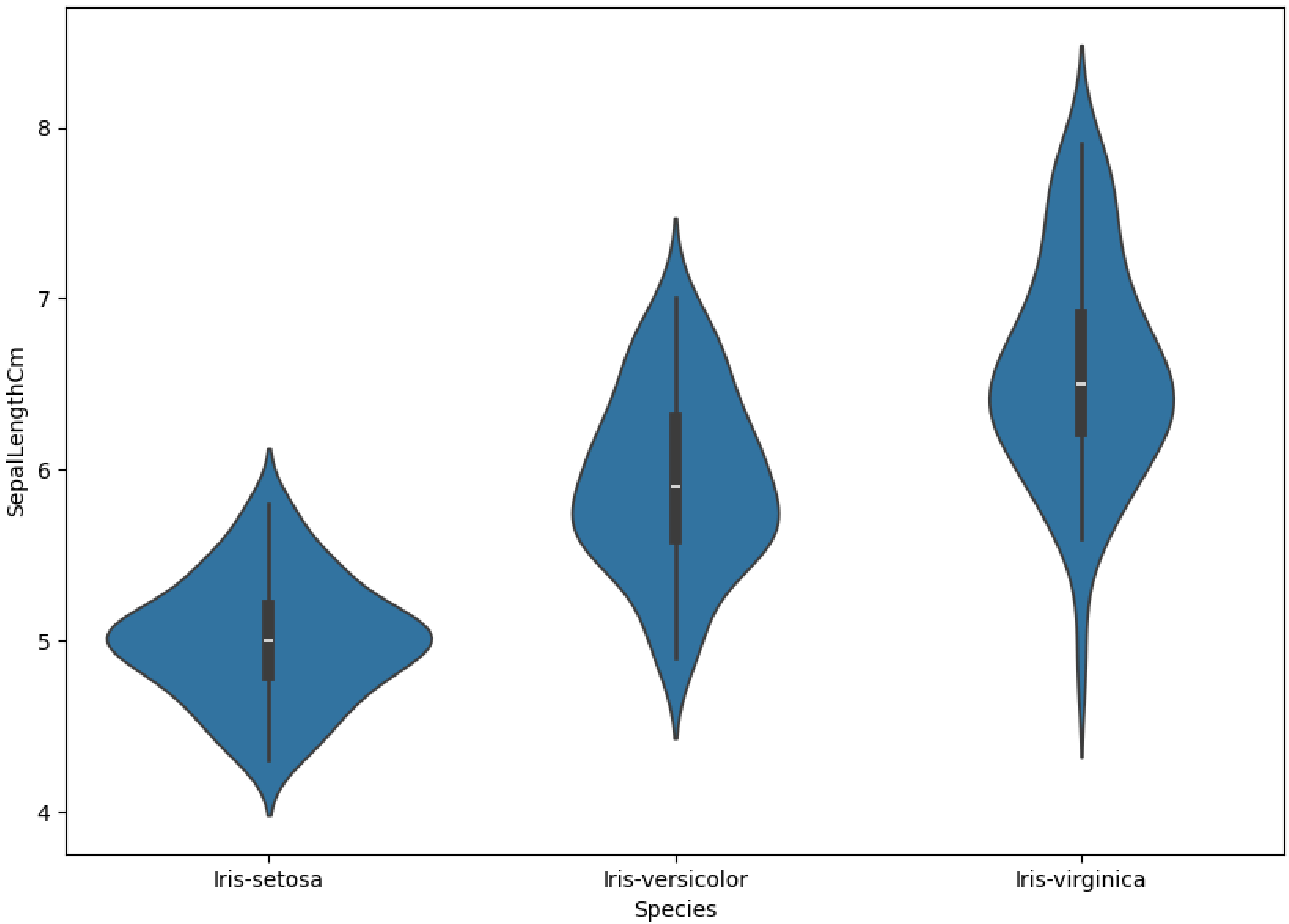
```
0
```



8. Violin Plot

It is used to visualize the distribution of data and its probability distribution.This chart is a combination of a Box Plot and a Density Plot that is rotated and placed on each side, to show the distribution shape of the data. The thick black bar in the centre represents the interquartile range, the thin black line extended from it represents the 95% confidence intervals, and the white dot is the median.Box Plots are limited in their display of the data, as their visual simplicity tends to hide significant details about how values in the data are distributed

In [32]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
```

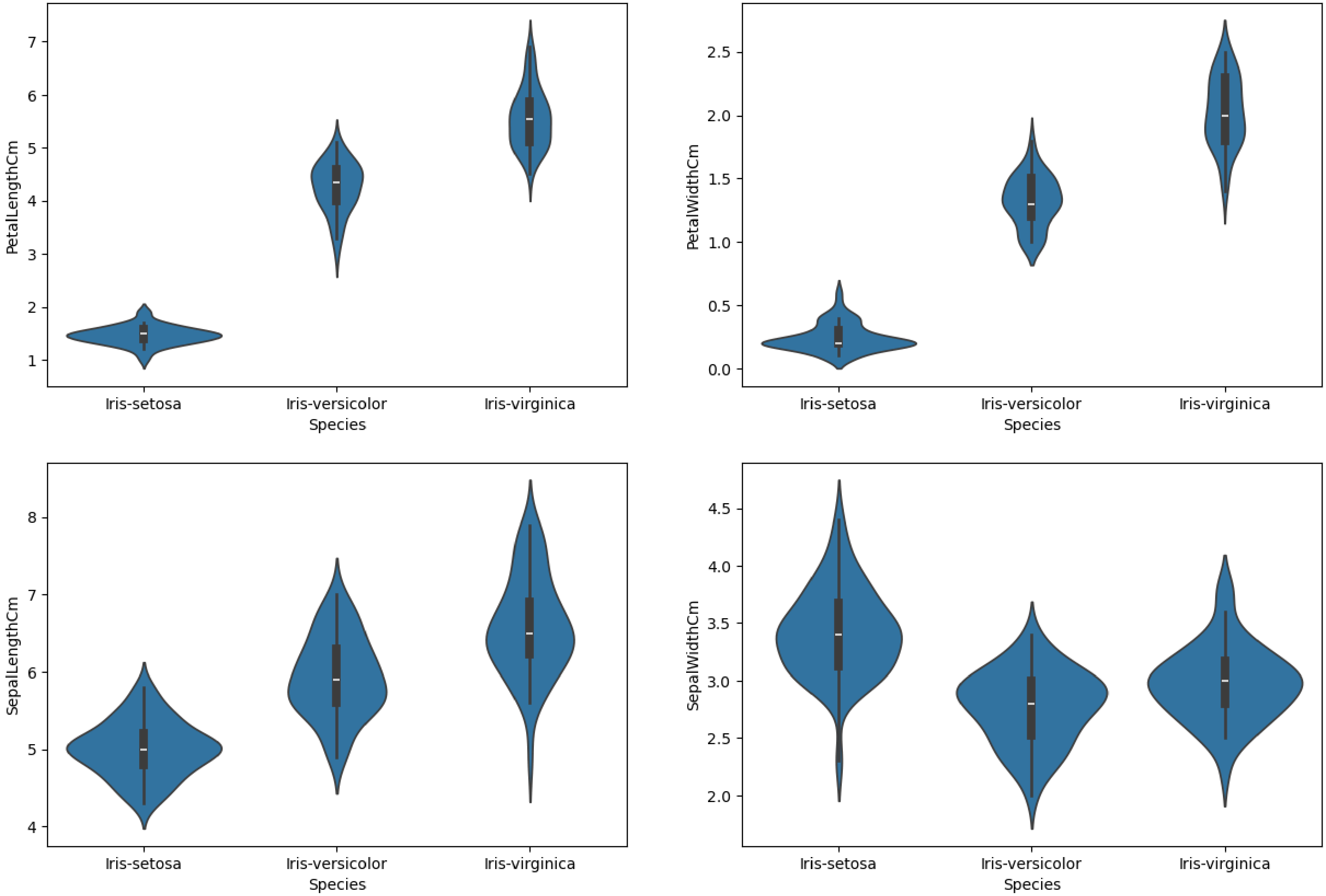


In [33]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
```

```python
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
```

Out[33]:

```
<Axes: xlabel='Species', ylabel='SepalWidthCm'>
```

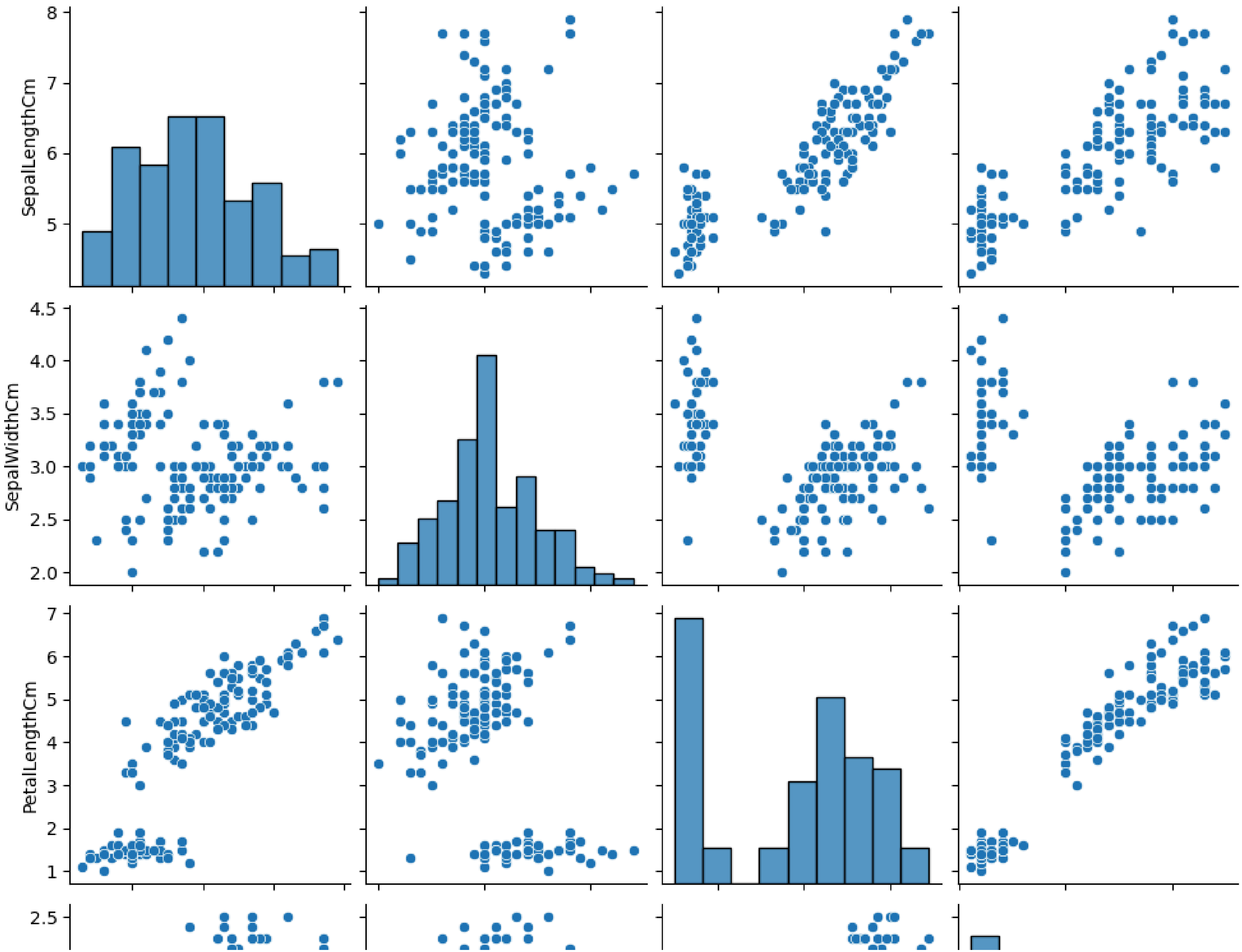

PetalLengthCm
PetalWidthCm
SepalLengthCm
SepalWidthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

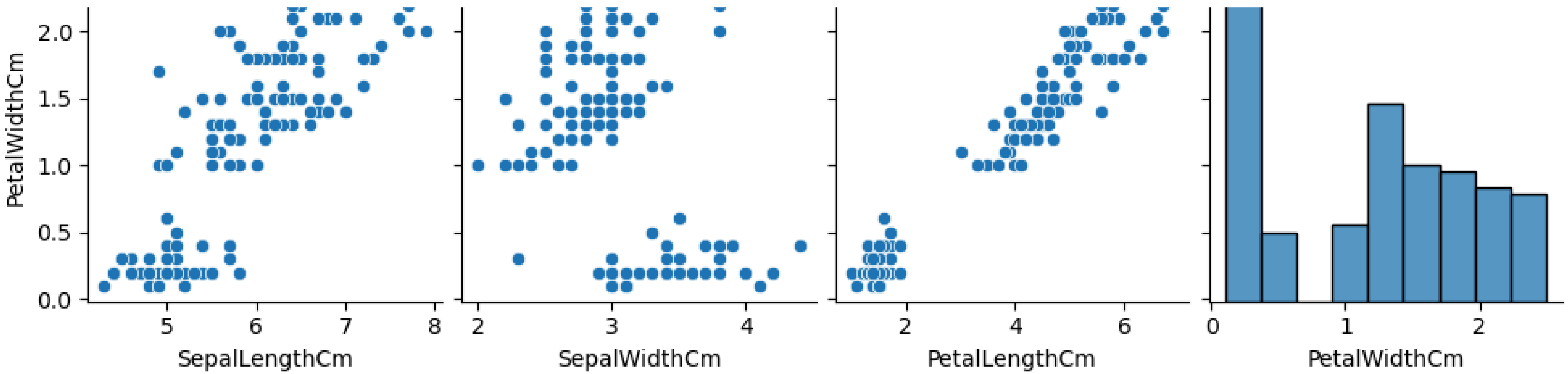
10. Pair Plot:

A “pairs plot” is also known as a scatterplot, in which one variable in the same data row is matched with another variable's value, like this: Pairs plots are just elaborations on this, showing all variables paired with all the other variables.

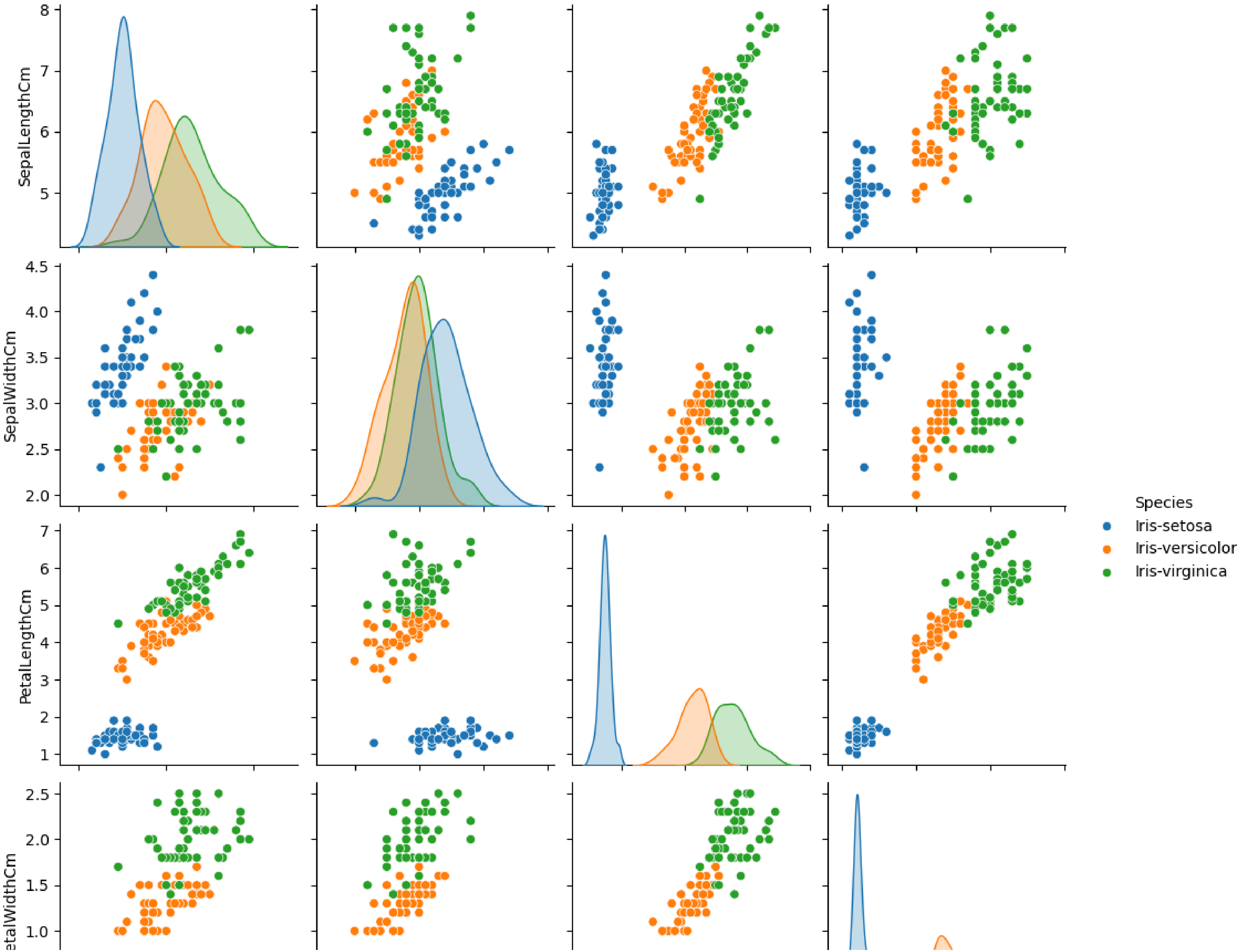
In [35]:
```
sns.pairplot(data=iris,kind='scatter')
```

Out[35]:
```
<seaborn.axisgrid.PairGrid at 0x2042d1fedb0>
```

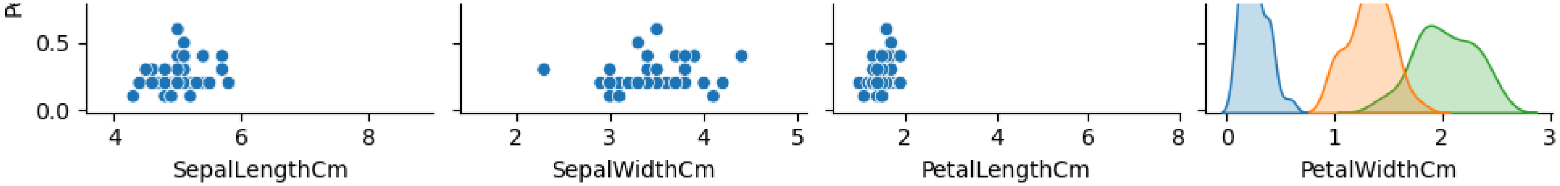

SepalLengthCm
SepalWidthCm
PetalLengthCm
8
7
6
5
4.5
4.0
3.5
3.0
2.5
2.0
7
6
5
4
3
2
1
2.5



In [36]:
```
sns.pairplot(iris,hue='Species');
```



SepalLengthCm
SepalWidthCm
PetalLengthCm
etalWidthCm
Species
Iris-setosa
Iris-versicolor
Iris-virginica

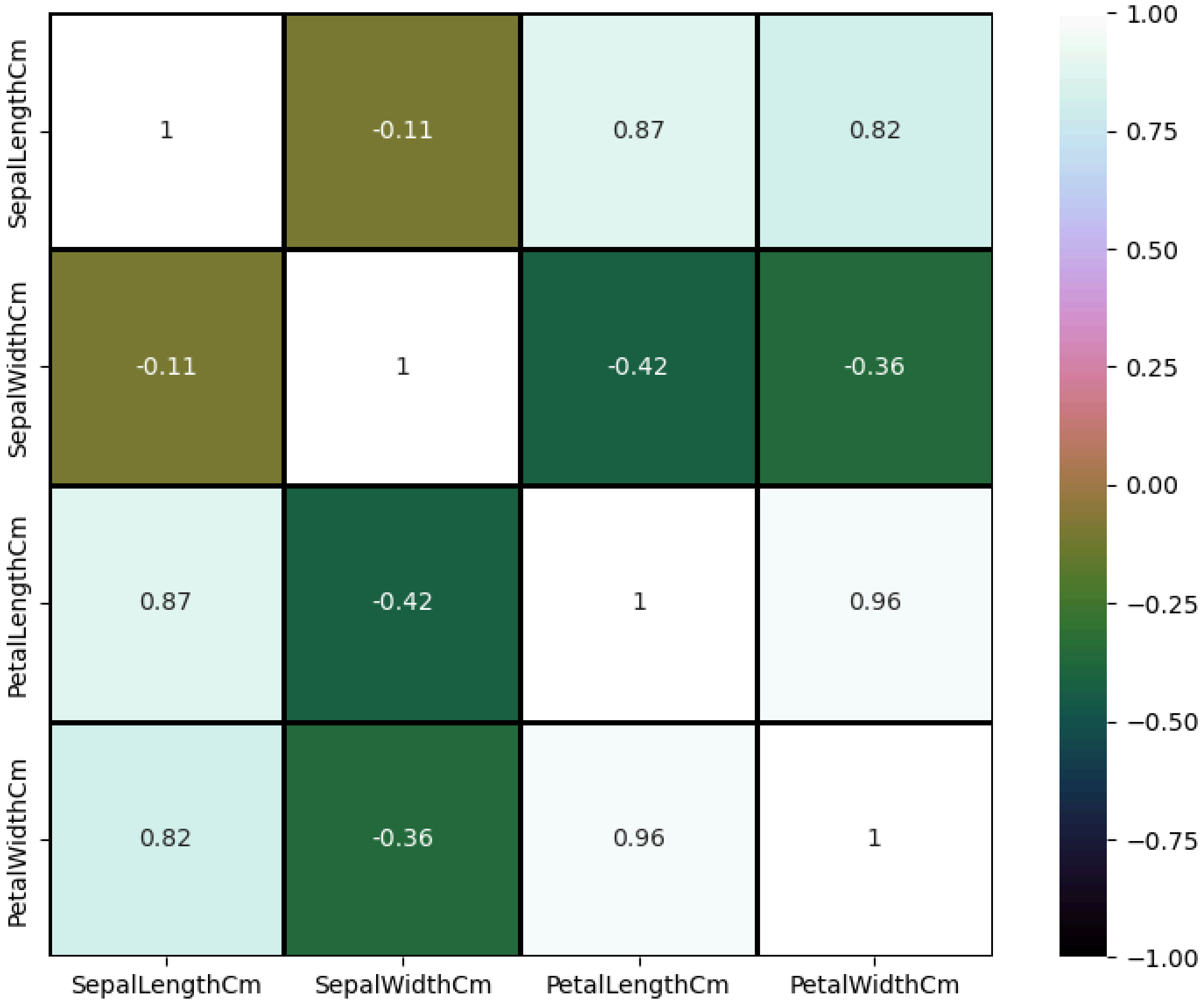


11. Heat map

Heat map is used to find out the correlation between different features in the dataset.High positive or negative value shows that the features have high correlation.This helps us to select the parmeters for machine learning.

In [38]:
```
# Exclude the 'Species' column to calculate correlation on numeric data only
numeric_data = iris.select_dtypes(include=['float64', 'int64'])

# Plot the heatmap
fig = plt.gcf()
fig.set_size_inches(10, 7)
sns.heatmap(numeric_data.corr(), annot=True, cmap='cubehelix', linewidths=1, linecolor='k', square=True, mask=False,
            vmin=-1, vmax=1, cbar_kws={"orientation": "vertical"}, cbar=True)
```

Out[38]: <Axes: >

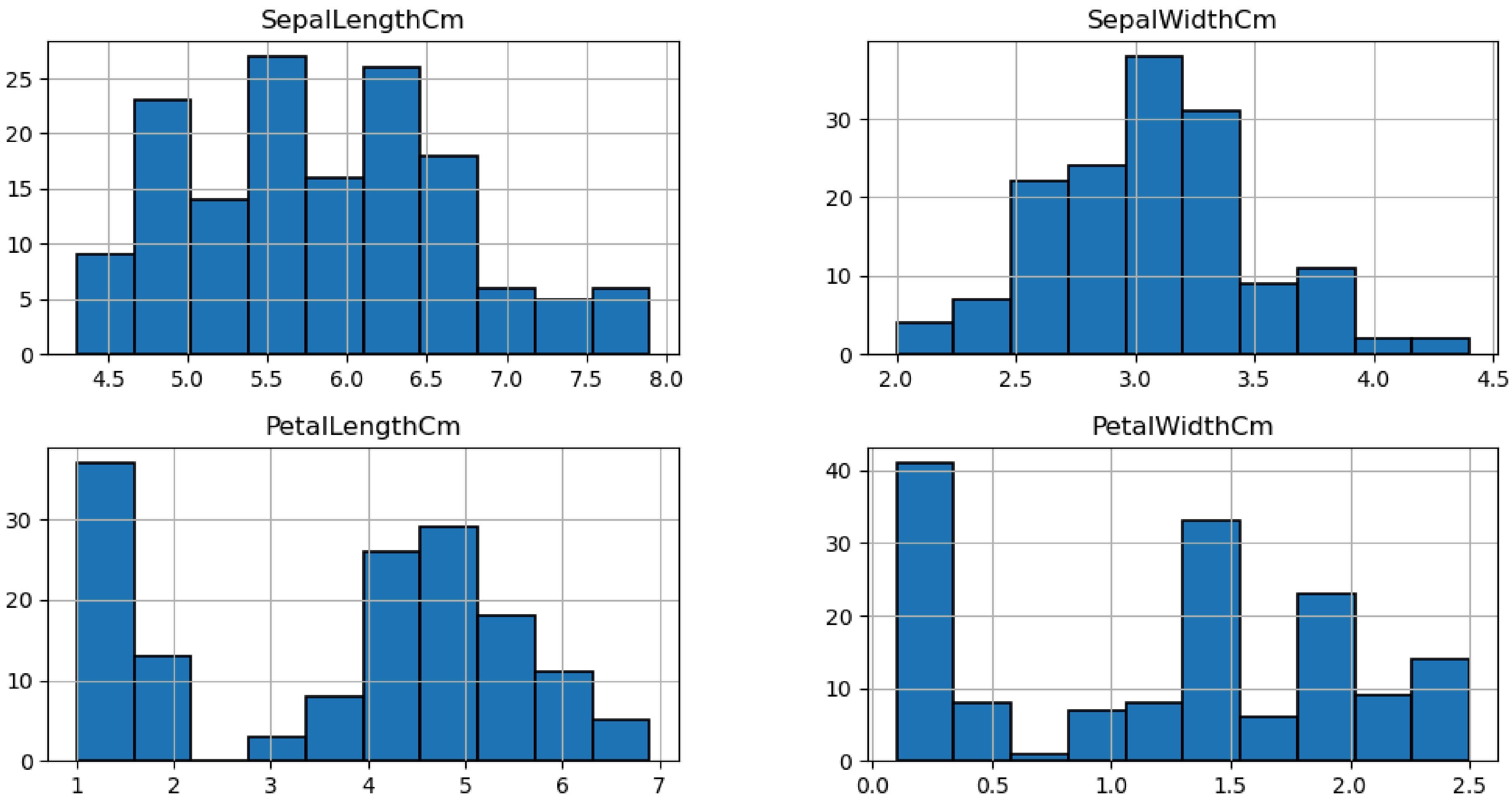


12. Distribution plot:

The distribution plot is suitable for comparing range and distribution for groups of numerical data. Data is plotted as value points along an axis. You can choose to display only the value points to see the distribution of values, a bounding box to see the range of values, or a combination of both as shown here.The distribution plot is not relevant for detailed analysis of the data as it deals with a summary of the data distribution.

In [40]:

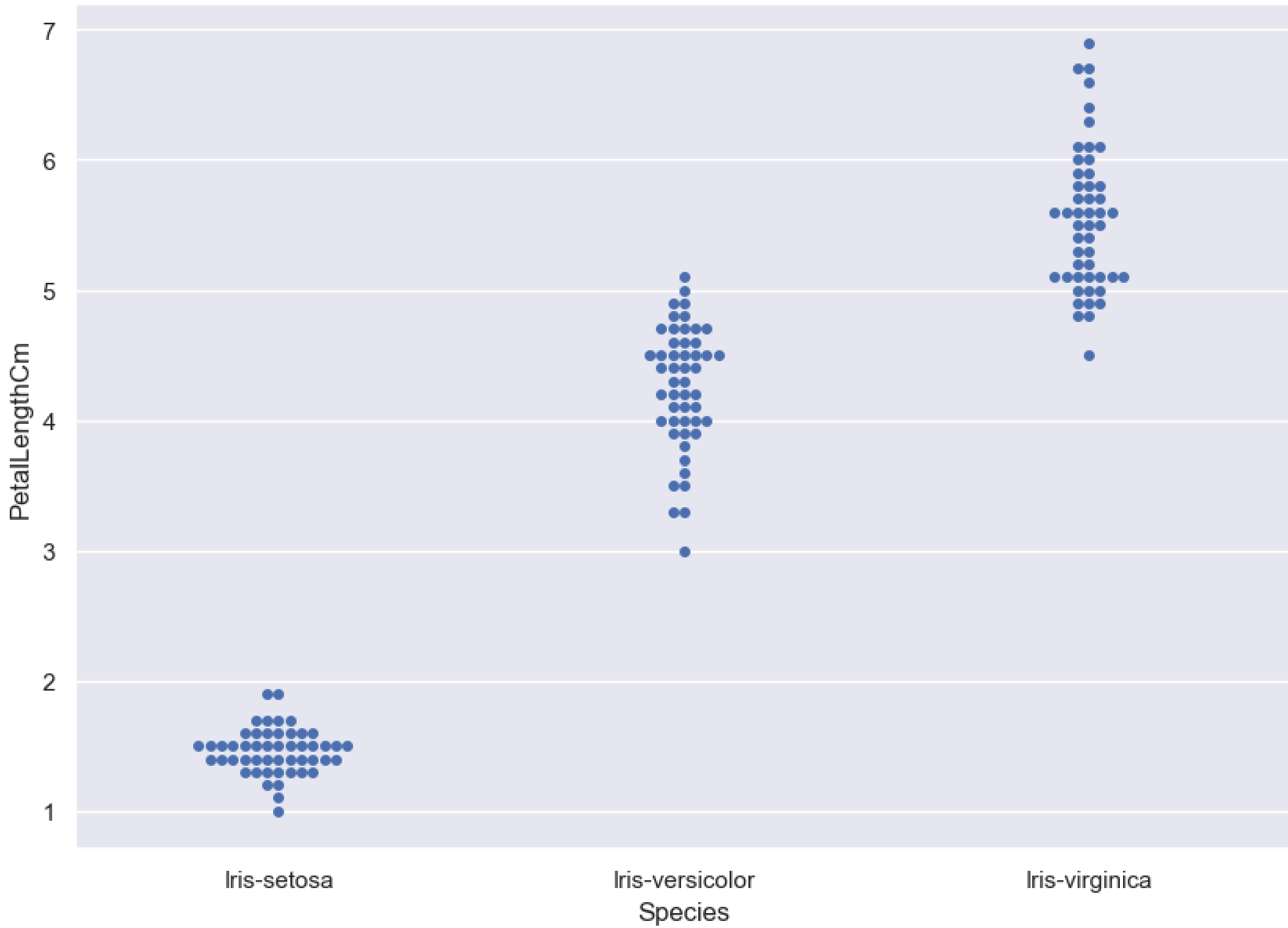
```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
```



13. Swarm plot

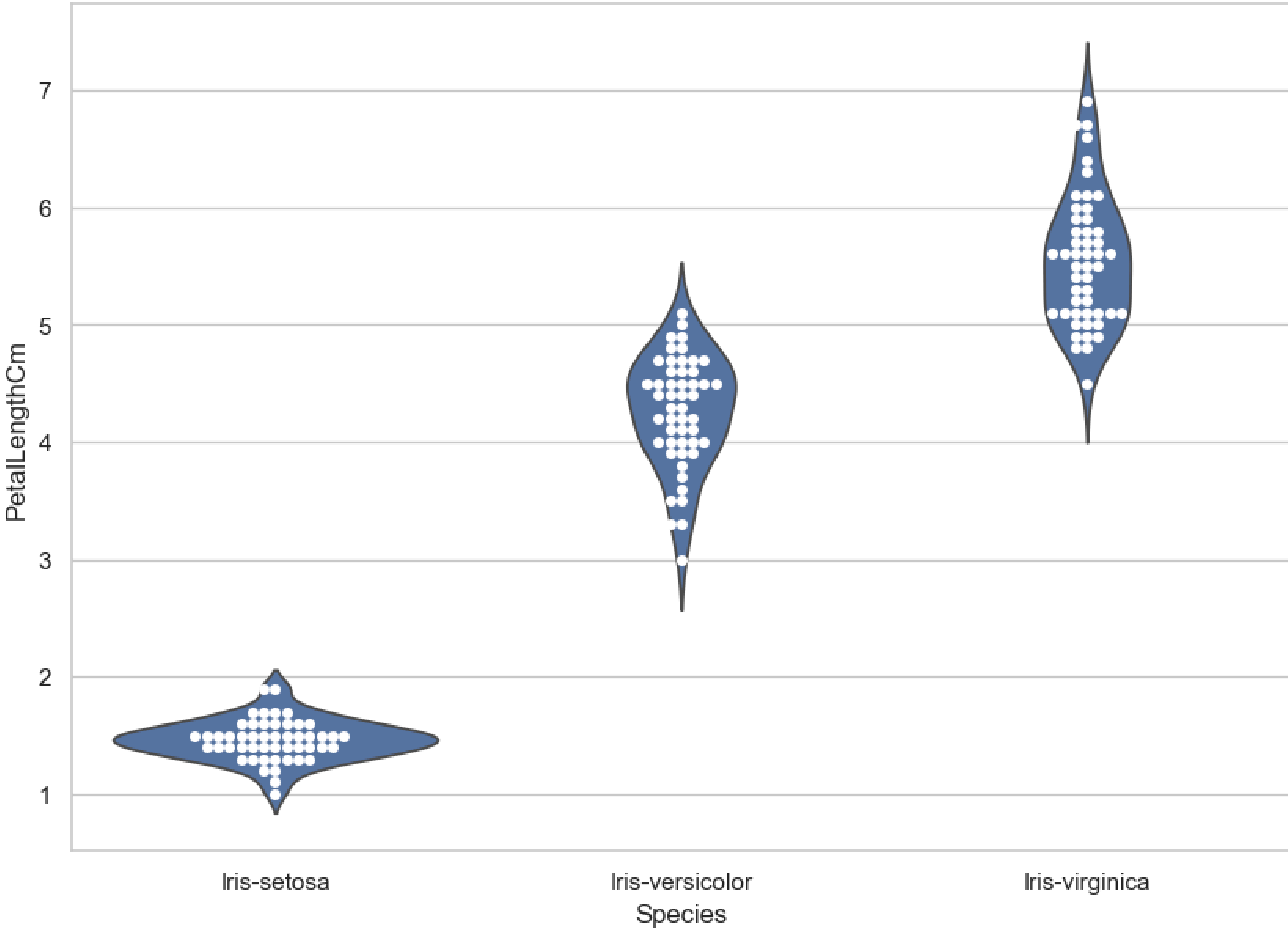
It looks a bit like a friendly swarm of bees buzzing about their hive. More importantly, each data point is clearly visible and no data are obscured by overplotting.A beeswarm plot improves upon the random jittering approach to move data points the minimum distance away from one another to avoid overlays. The result is a plot where you can see each distinct data point, like shown in below plot

In [42]:
```
sns.set(style="darkgrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
fig = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris)
```



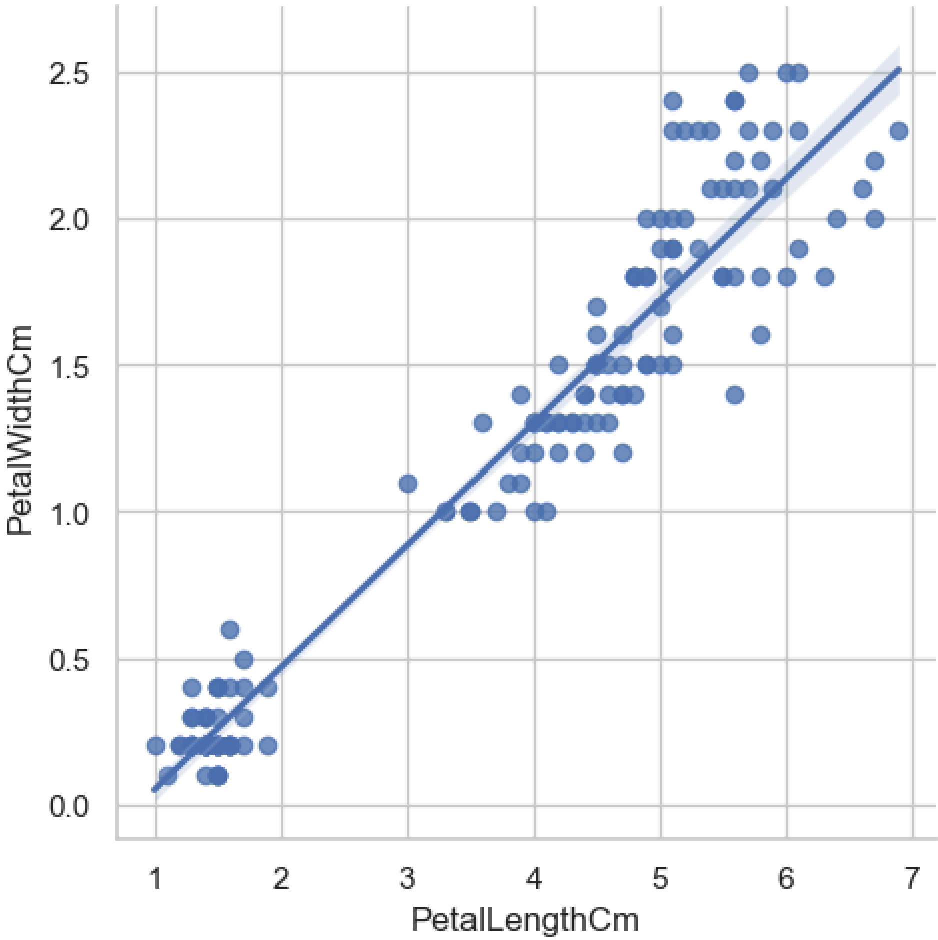
In [43]:
```
sns.set(style="whitegrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
```

```python
ax = sns.violinplot(x="Species", y="PetalLengthCm", data=iris, inner=None)
ax = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris,color="white", edgecolor="black")
```



14. LM PLot

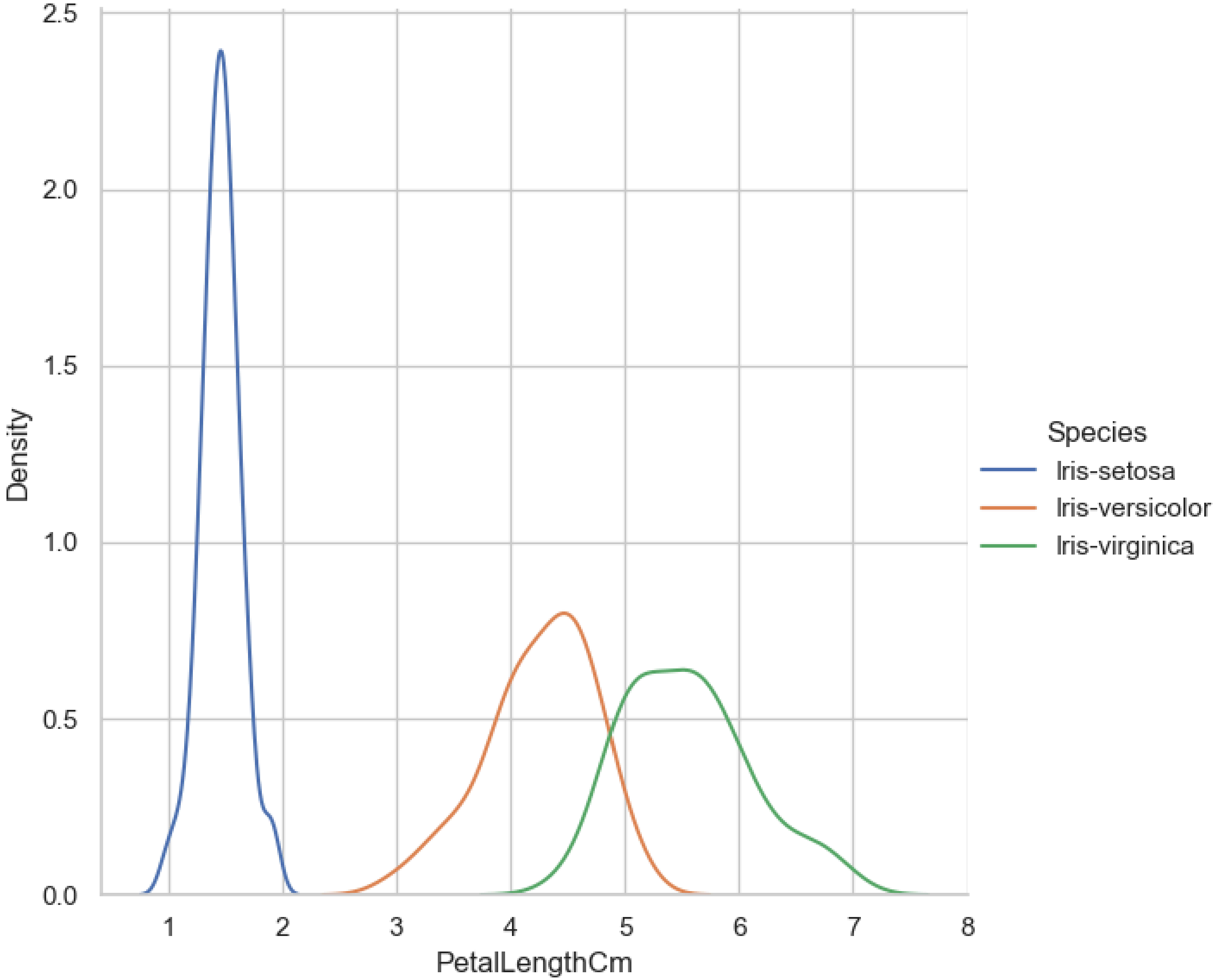
In [45]:
```
fig=sns.lmplot(x="PetalLengthCm", y="PetalWidthCm",data=iris)
```



15. FacetGrid

In [47]:
```python
# Create the FacetGrid and KDE plot
sns.FacetGrid(iris, hue="Species", height=6) \
    .map(sns.kdeplot, "PetalLengthCm") \
    .add_legend()
```

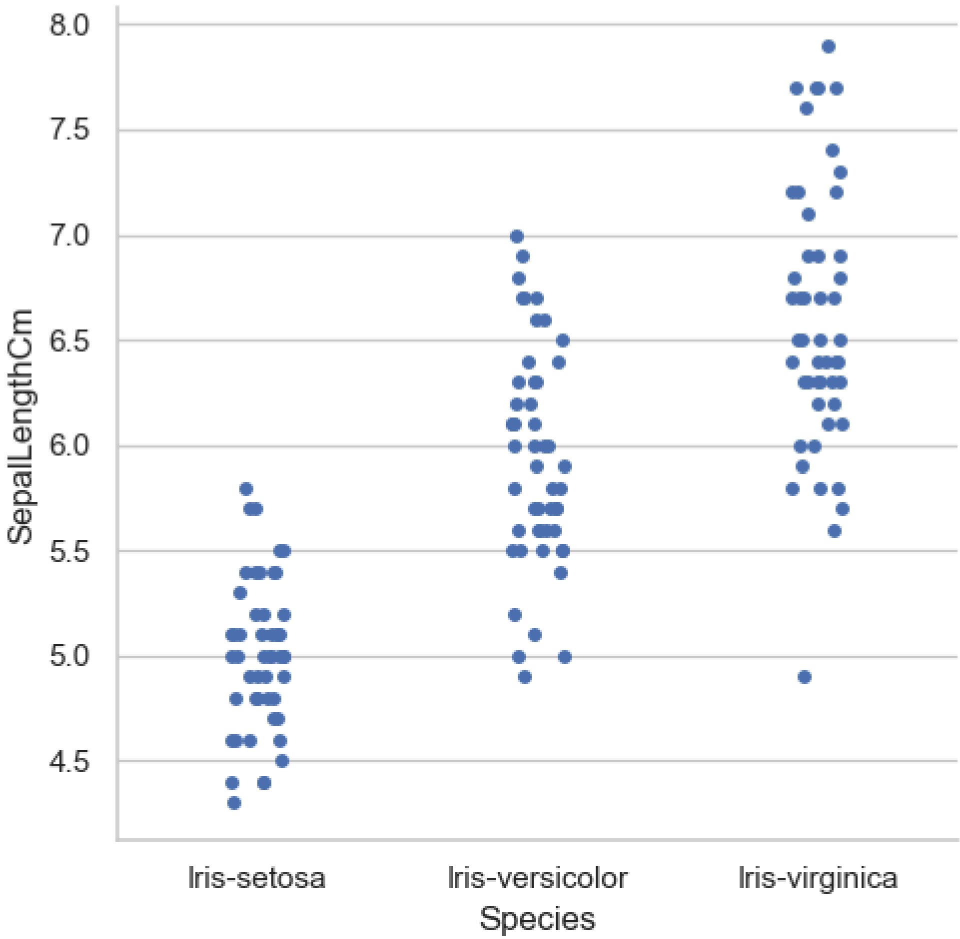
Out[47]:
```
<seaborn.axisgrid.FacetGrid at 0x2042f7e73b0>
```



** 16. Factor Plot **

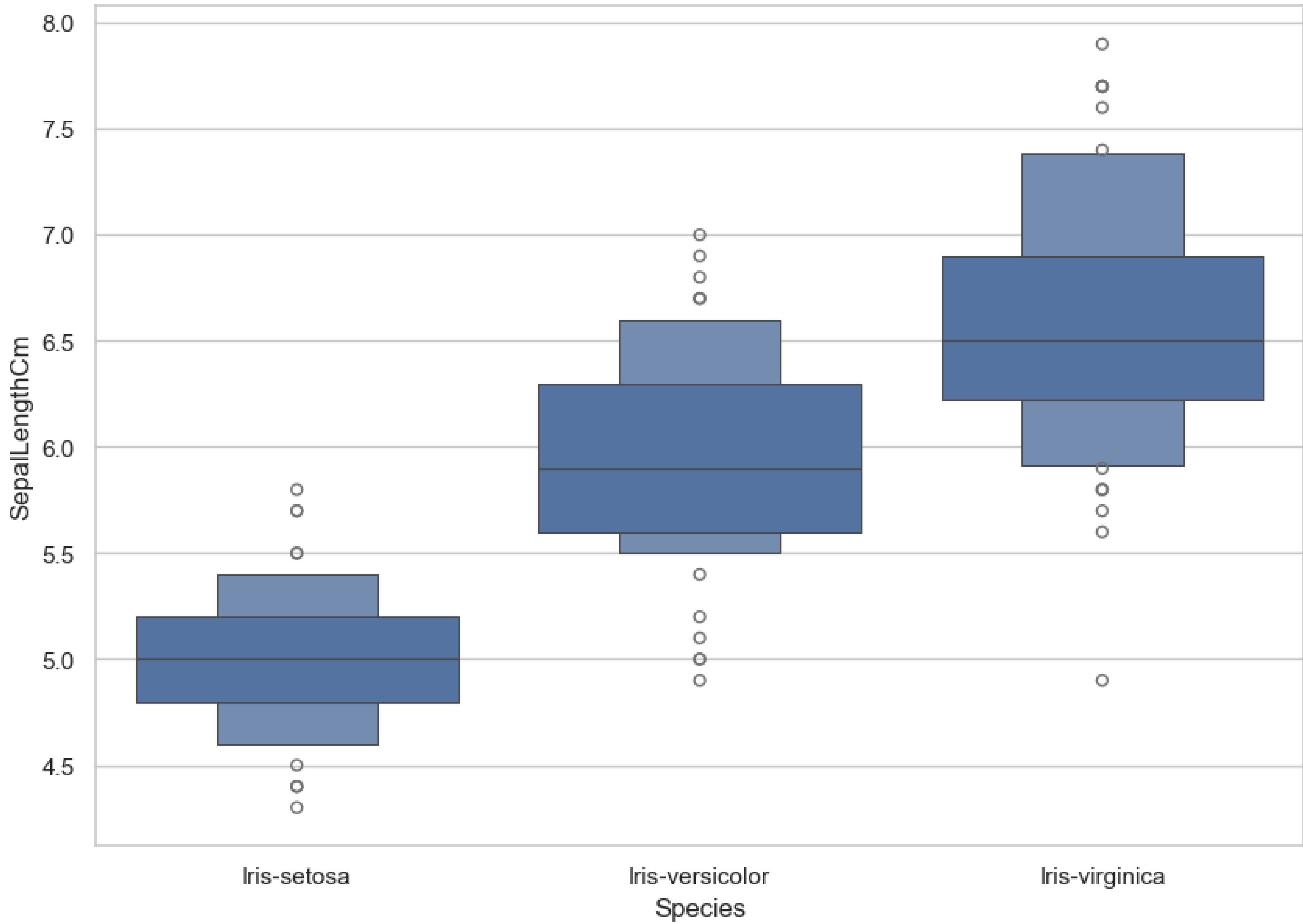
In [49]:
```
# Create a categorical plot (equivalent to factorplot)
sns.catplot(x='Species', y='SepalLengthCm', data=iris)

# Turn off interactive plotting mode (optional)
plt.ioff()
#sns.factorplot('Species','SepalLengthCm',data=iris,ax=ax[0][0])
#sns.factorplot('Species','SepalWidthCm',data=iris,ax=ax[0][1])
#sns.factorplot('Species','PetalLengthCm',data=iris,ax=ax[1][0])
#sns.factorplot('Species','PetalWidthCm',data=iris,ax=ax[1][1])
```

Out[49]: <contextlib.ExitStack at 0x2042a76c0e0>



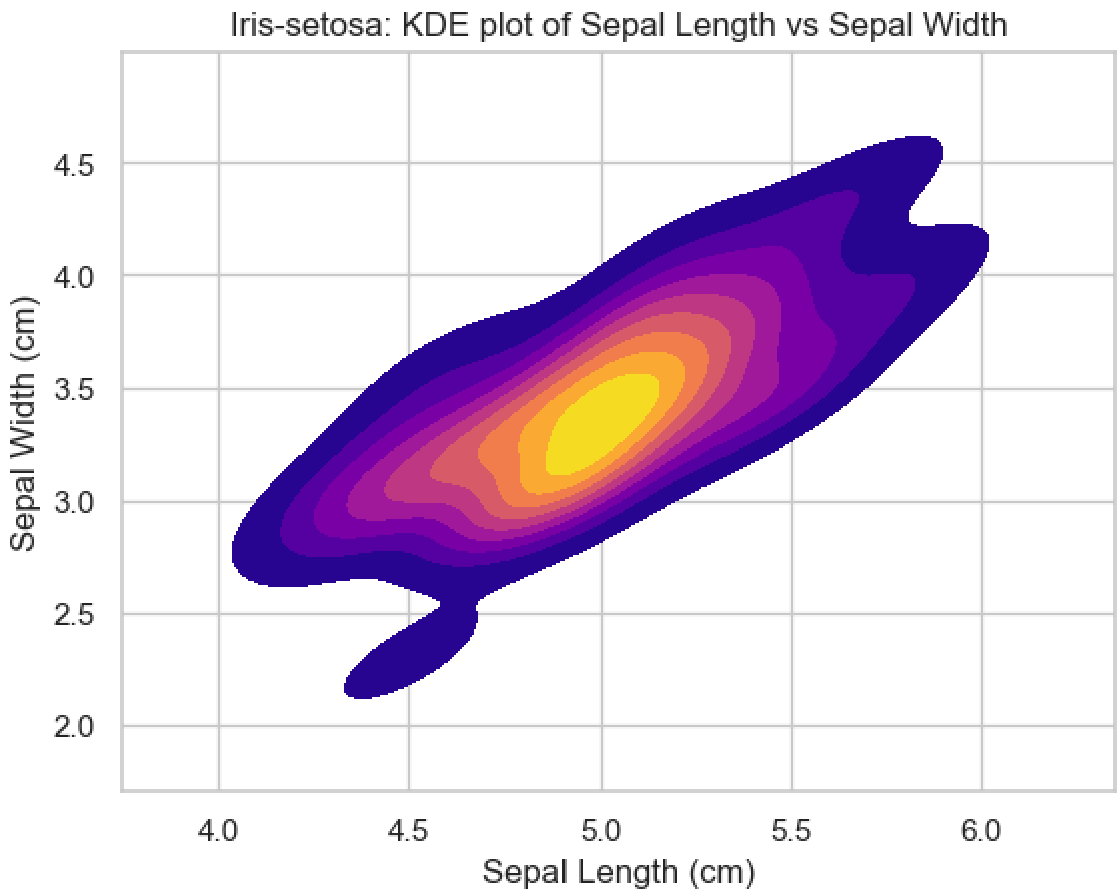
17. Boxen Plot

In [51]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxenplot(x='Species',y='SepalLengthCm',data=iris)
plt.show()
```


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
SepalLengthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

18.KDE Plot

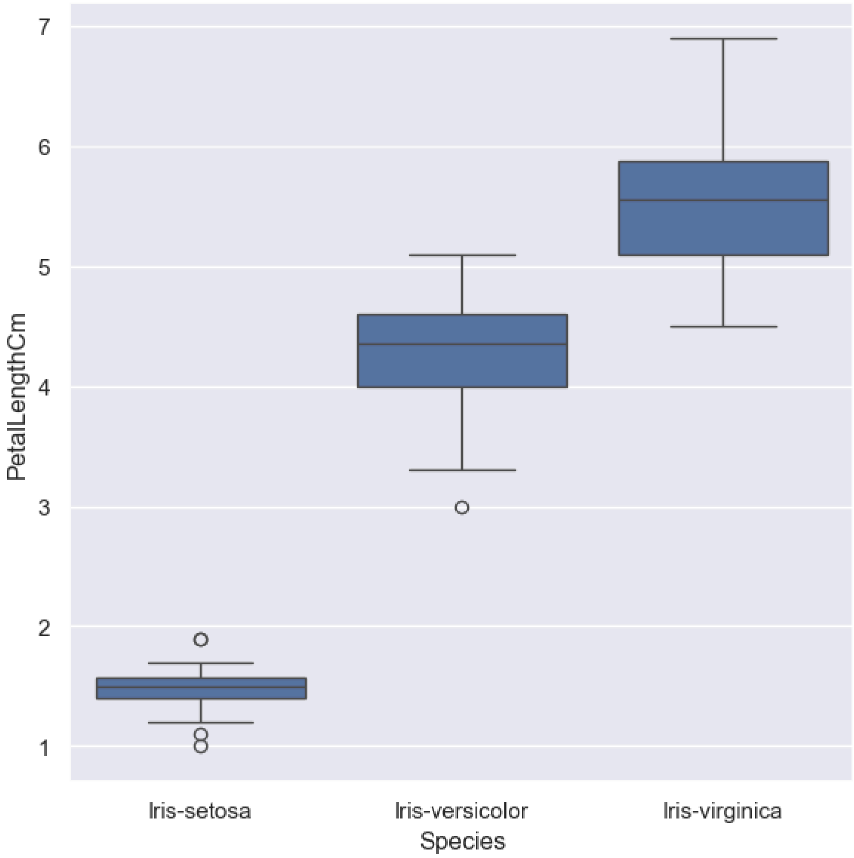
In [53]:
```
# Filter the data for Iris-setosa
sub = iris[iris['Species'] == 'Iris-setosa']

# Create the KDE plot for SepalLengthCm vs. SepalWidthCm
sns.kdeplot(data=sub, x='SepalLengthCm', y='SepalWidthCm', cmap="plasma", shade=True, shade_lowest=False)

# Add title and labels
plt.title('Iris-setosa: KDE plot of Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



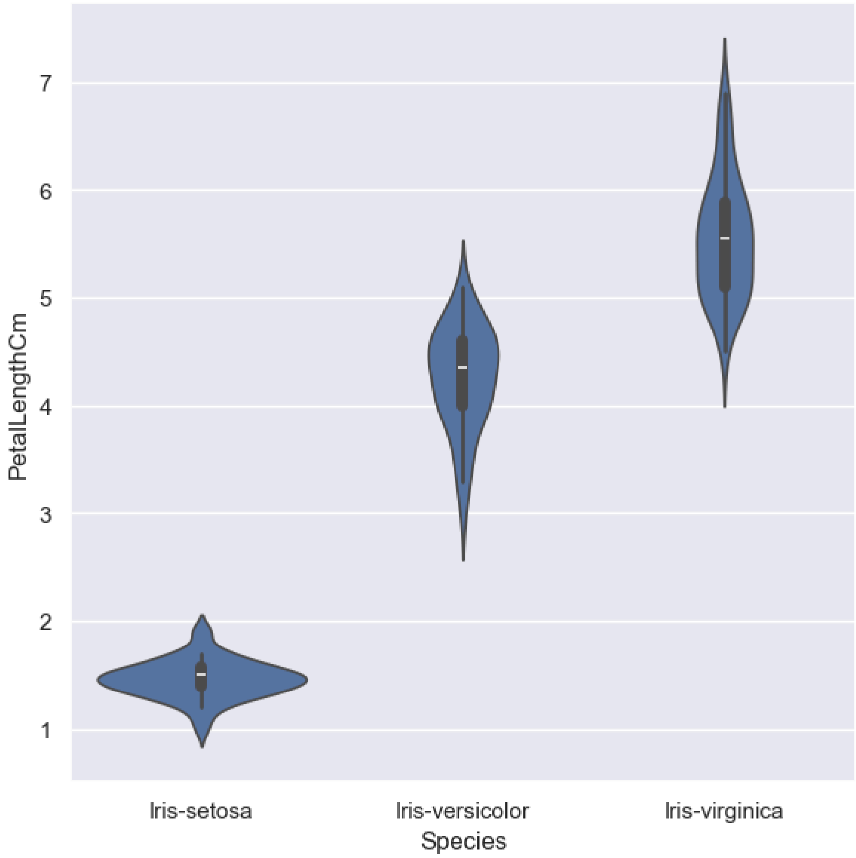
DashBoard

In [55]:

```
sns.set_style('darkgrid')
f,axes=plt.subplots(2,2,figsize=(15,15))

k1=sns.boxplot(x="Species", y="PetalLengthCm", data=iris,ax=axes[0,0])
k2=sns.violinplot(x='Species',y='PetalLengthCm',data=iris,ax=axes[0,1])
k3=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor='gray',size=8,palette='winter',orient='v',ax=ax
#axes[1,1].hist(iris.hist,bin=10)
axes[1,1].hist(iris.PetalLengthCm,bins=100)
#k2.set(xlim=(-1,0.8))
plt.show()
```



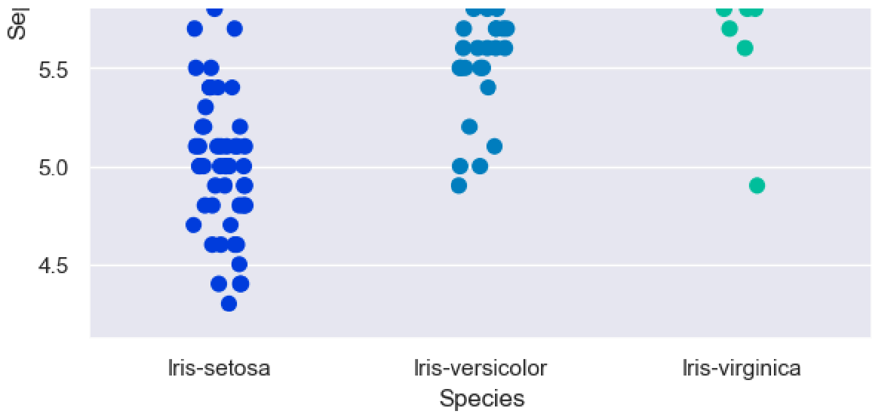
7
6
5
4
3
2
1
PetalLengthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

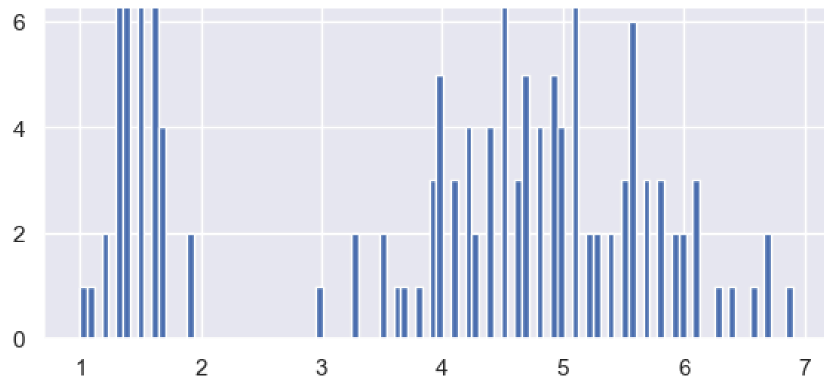


7
6
5
4
3
2
1
PetalLengthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

8.0
7.5
7.0
6.5
6.0

14
12
10
8



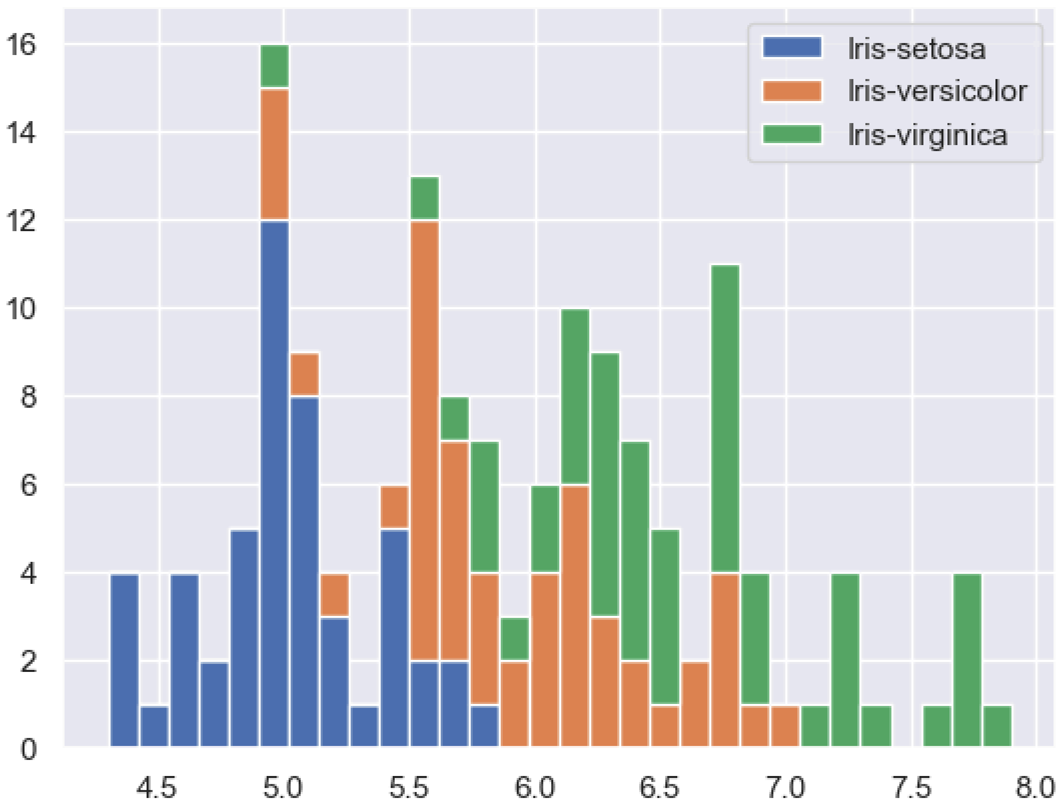


In the dashboard we have shown how to create multiple plots to foam a dashboard using Python.In this plot we have demonstrated how to plot Seaborn and Matplotlib plots on the same Dashboard.

Stacked Histogram

In [58]:
```
iris['Species'] = iris['Species'].astype('category')
#iris.head()
```

In [59]:
```
list1=list()
mylabels=list()
for gen in iris.Species.cat.categories:
    list1.append(iris[iris.Species==gen].SepalLengthCm)
    mylabels.append(gen)

h=plt.hist(list1,bins=30,stacked=True,rwidth=1,label=mylabels)
plt.legend()
plt.show()
```

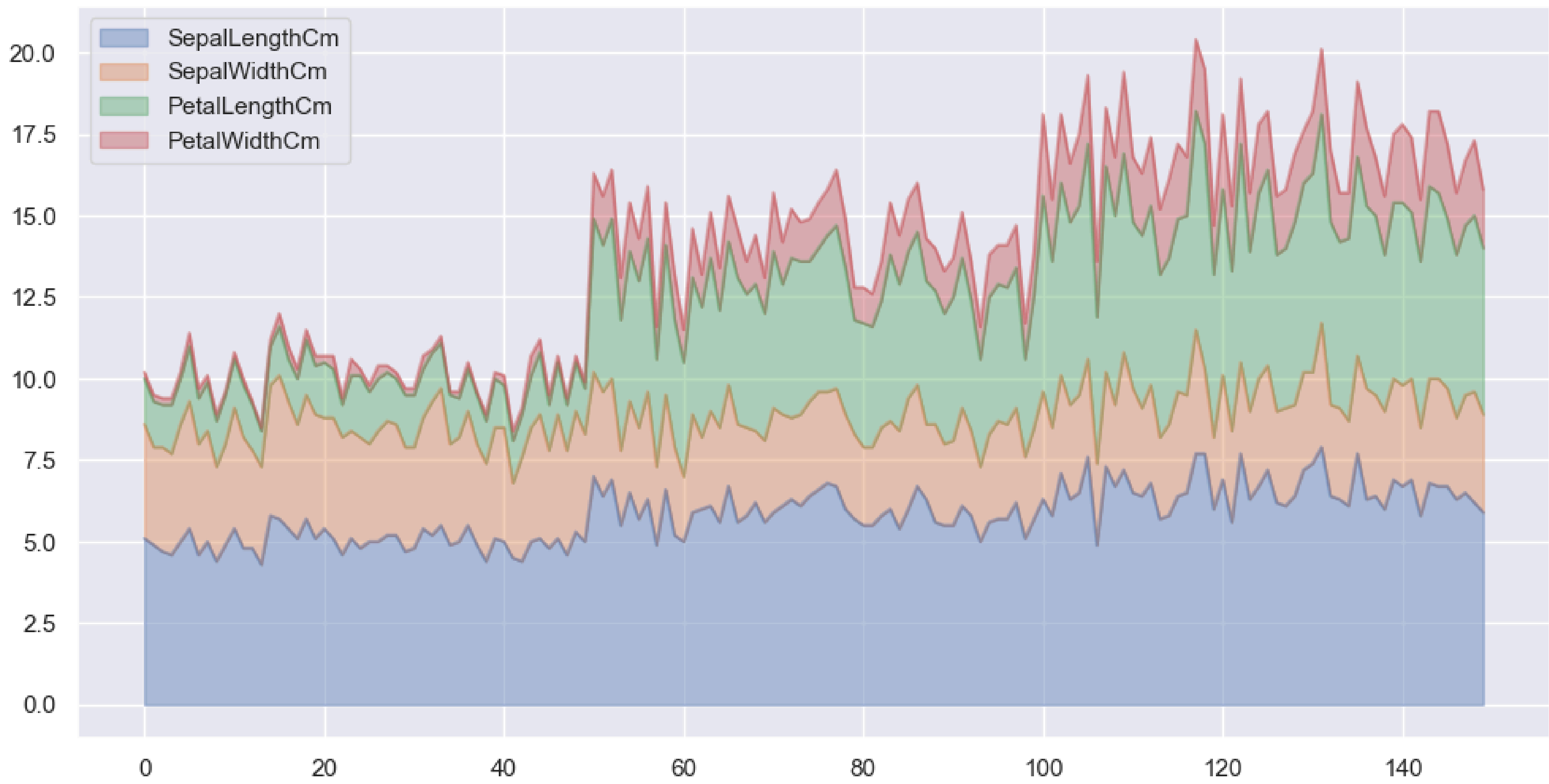


With Stacked Histogram we can see the distribution of Sepal Length of Different Species together.This shows us the range of Sepan Length for the three different Species of Iris Flower.

Area Plot

Area Plot gives us a visual representation of Various dimensions of Iris flower and their range in dataset.

In [62]:
```
#iris['SepalLengthCm'] = iris['SepalLengthCm'].astype('category')
#iris.head()
#iris.plot.area(y='SepalLengthCm',alpha=0.4,figsize=(12, 6));
iris.plot.area(y=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'],alpha=0.4,figsize=(12, 6));
plt.show()
```

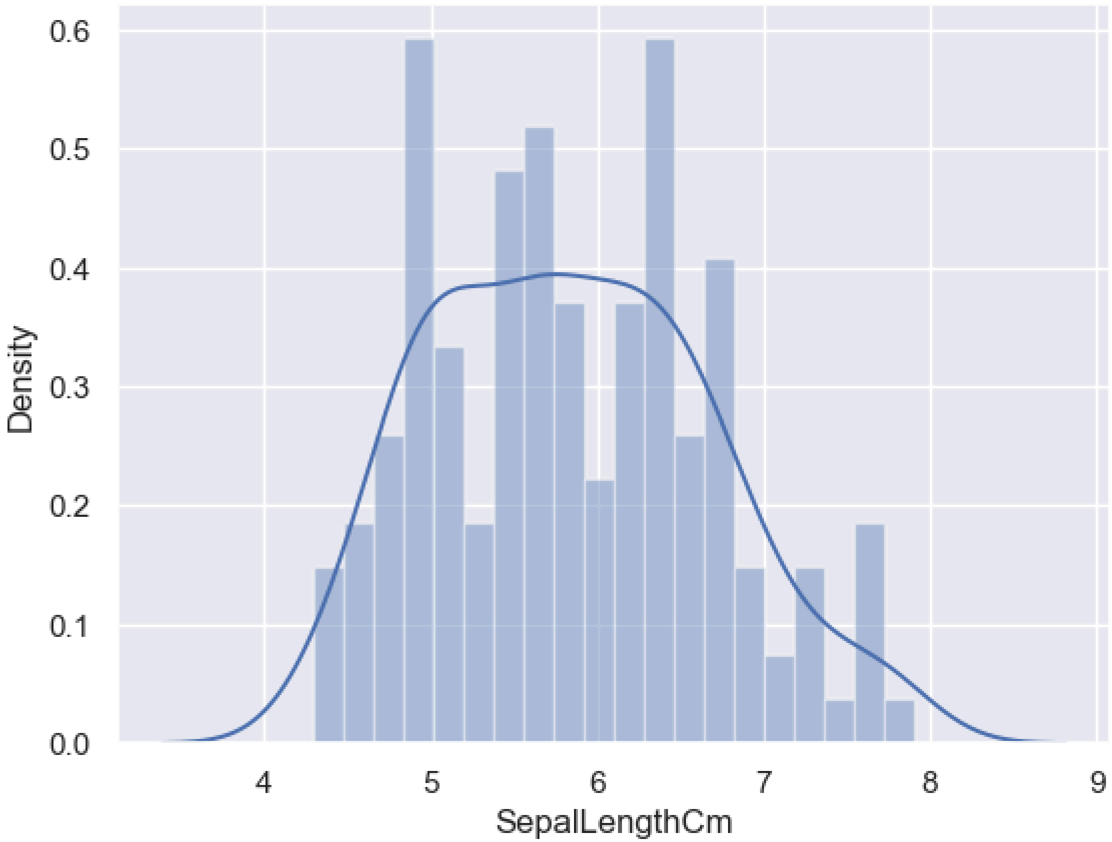


Distplot:

It helps us to look at the distribution of a single variable.Kde shows the density of the distribution

In [64]:
```
sns.distplot(iris['SepalLengthCm'],kde=True,bins=20);
```

In [65]:
```
plt.show()
```




```
In [ ]:
```