IRIS DATASET VISUALIZATION(SEABORN,MATPLOTLIB)¶

Importing libraries

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing,
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
```

1. importing Iris data set

In [3]:
```
iris=pd.read_csv(r'D:\Data Science with AI\29th-jan-2024\IRIS DATASET _ ADVANC
```

In [4]:
```
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.tail()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [7]:
```
iris.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [8]:
```
iris.drop('Id',axis=1,inplace=True)
```

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

Out[9]:

	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 [10]:
```
iris.info()
```

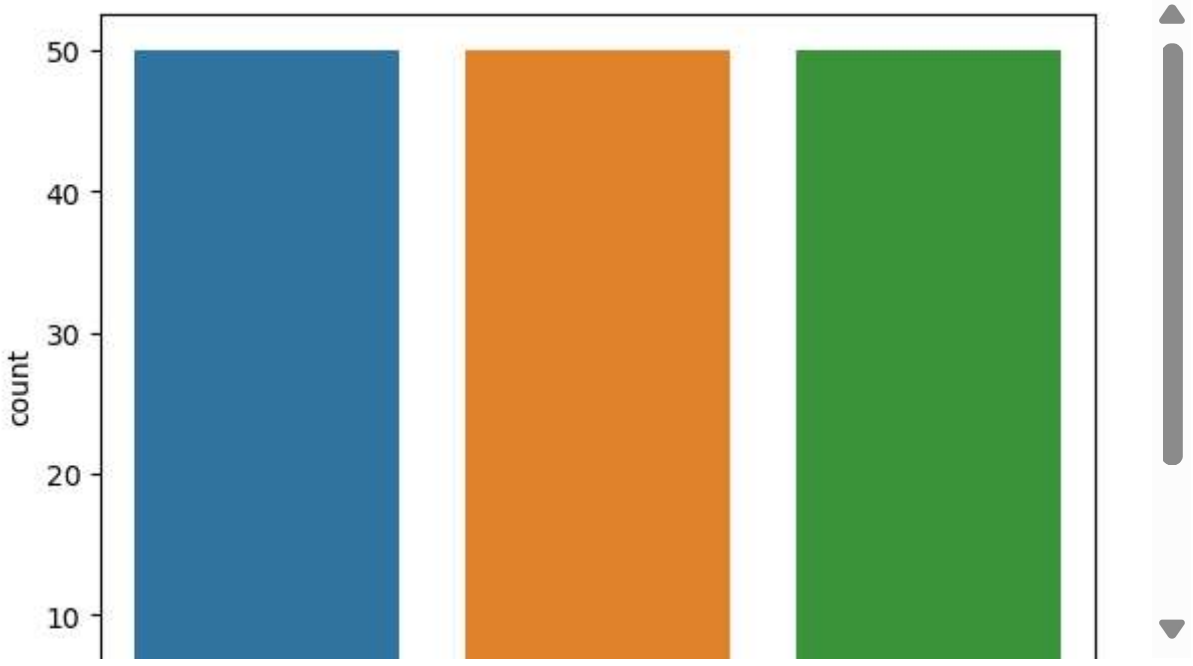
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 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 [11]:
```
iris['Species'].value_counts()
```

Out[11]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

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 [12]:
```
sns.countplot(x='Species',data=iris)
plt.show()
```



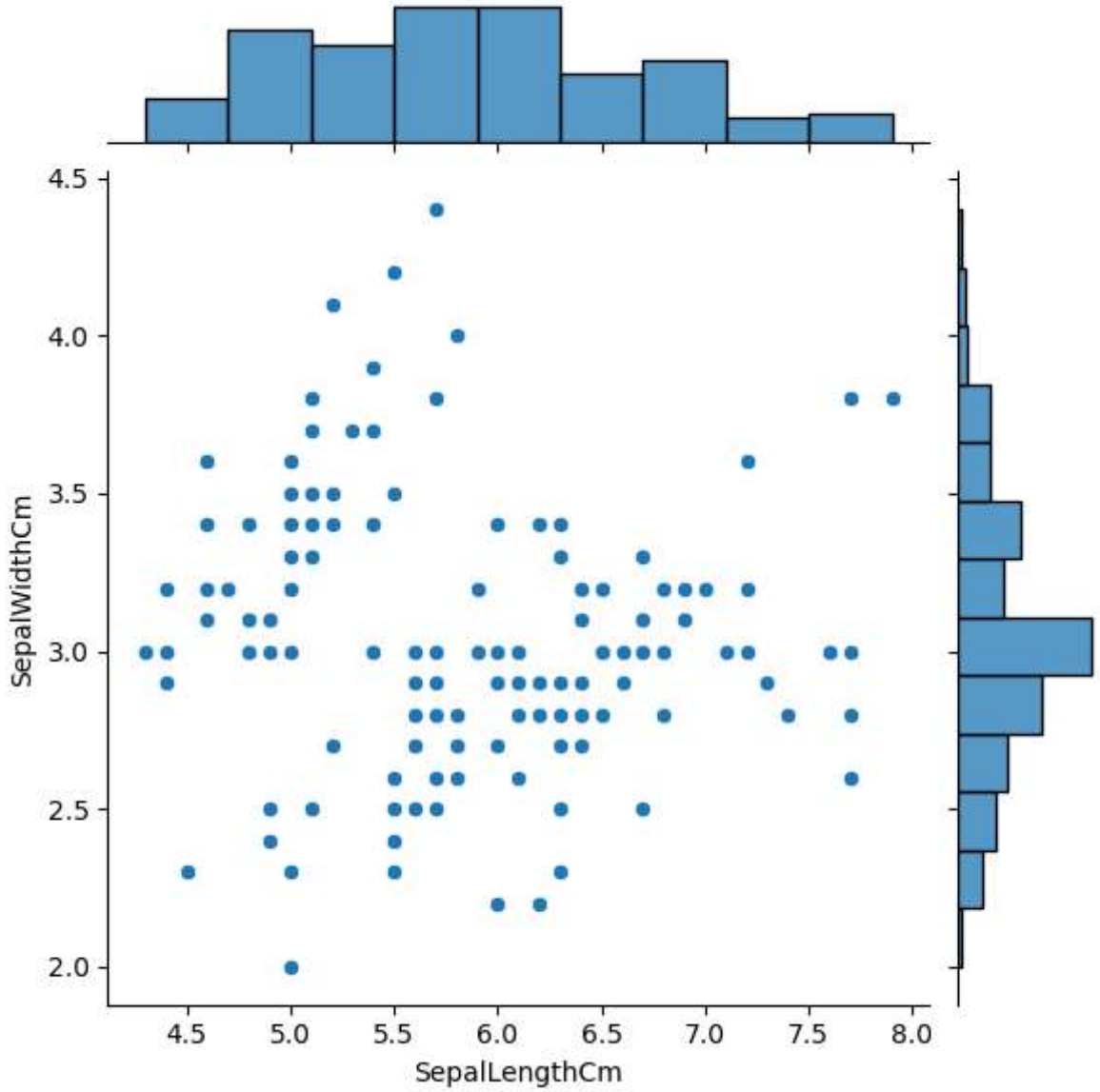
3. Joint plot:- Jointplot is seaborn library specific and can be used to quickly visualize and analyze the relationship between two variables

In [13]:
```
iris.head()
```

Out[13]:

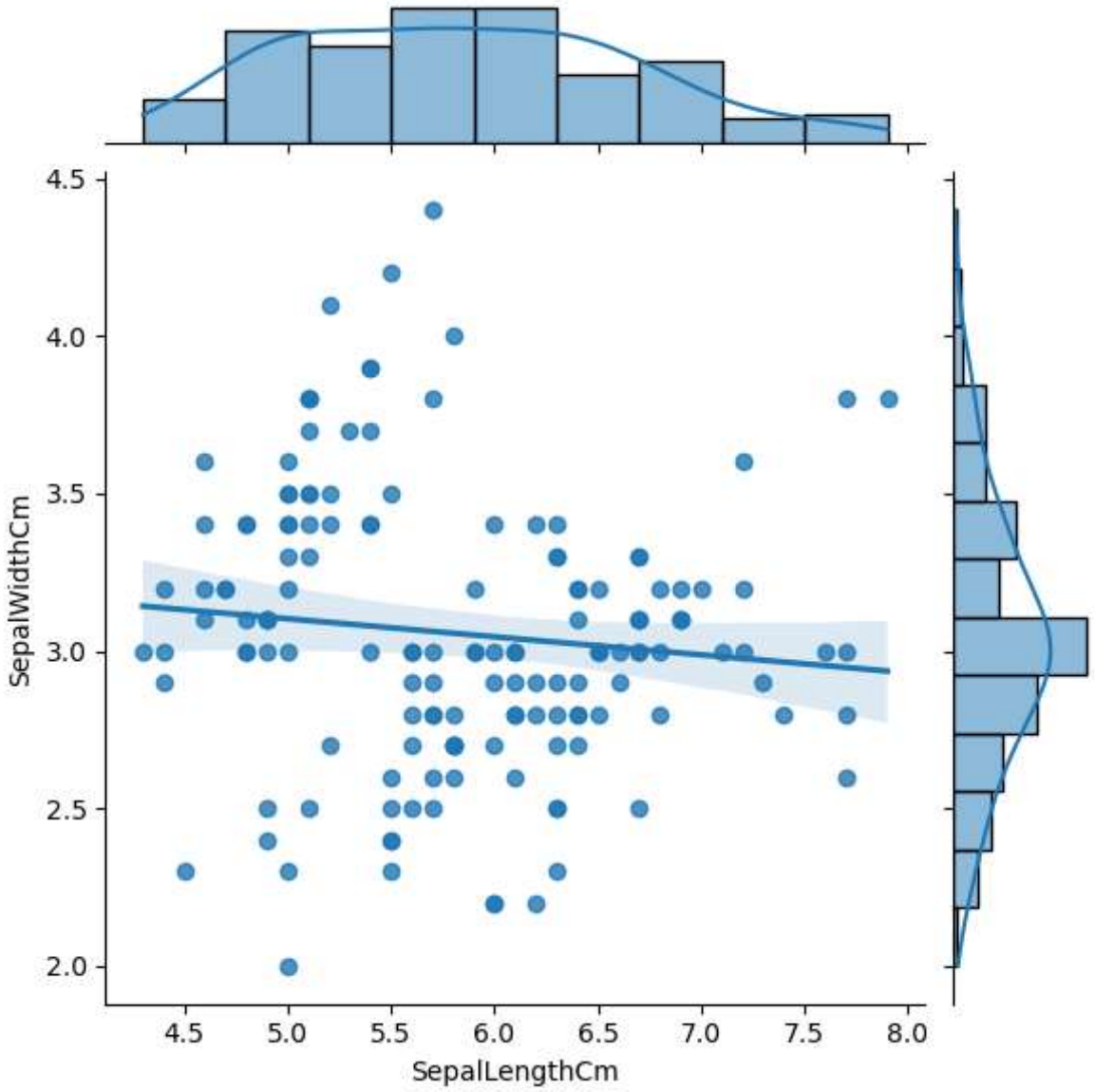
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [14]:
```
fig=sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',data=iris)
```

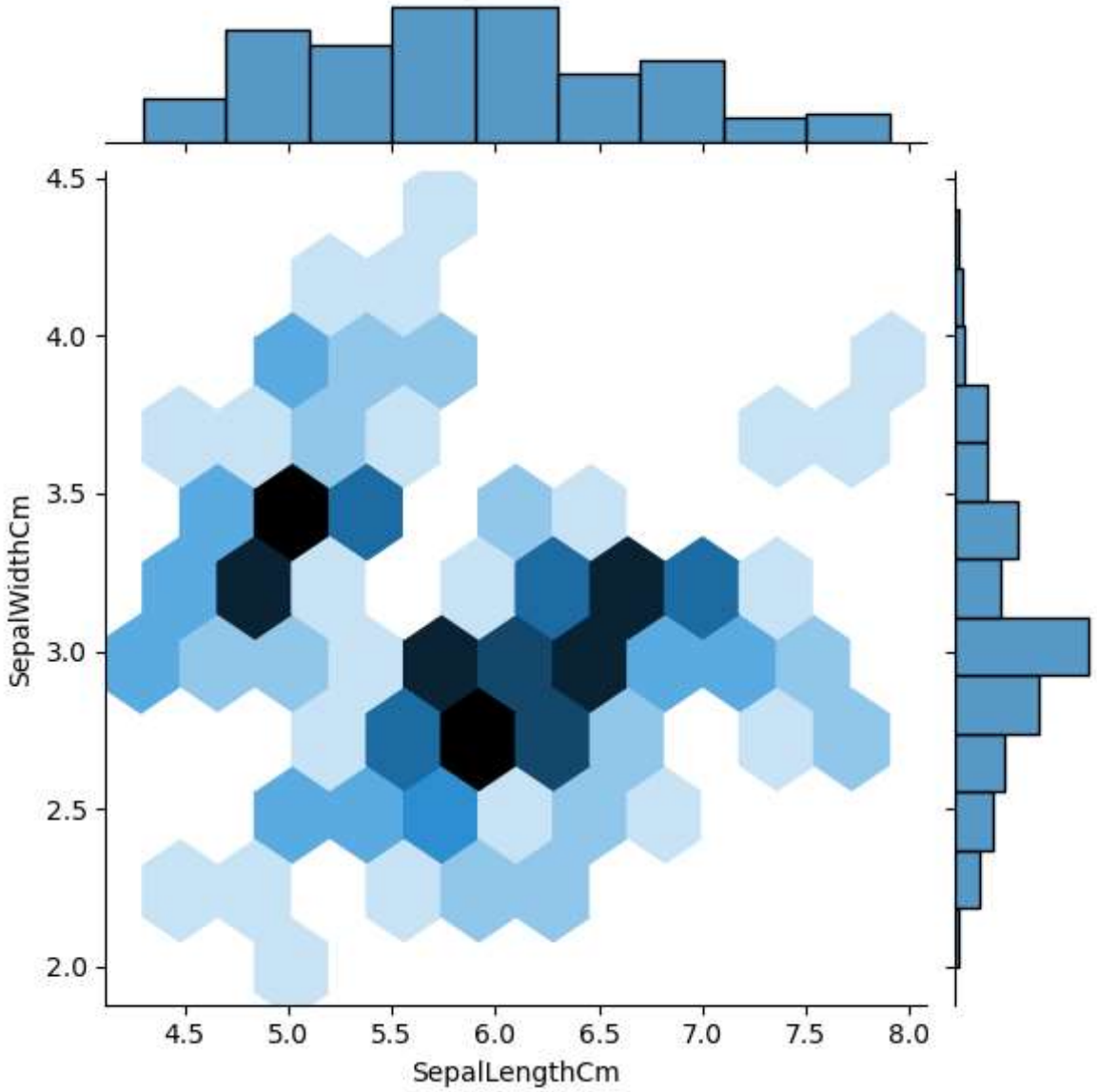


In [15]:
```
sns.jointplot(x="SepalLengthCm" ,y= "SepalWidthCm" , data=iris ,kind="reg")
```

Out[15]:
```
<seaborn.axisgrid.JointGrid at 0x25c074fee50>
```



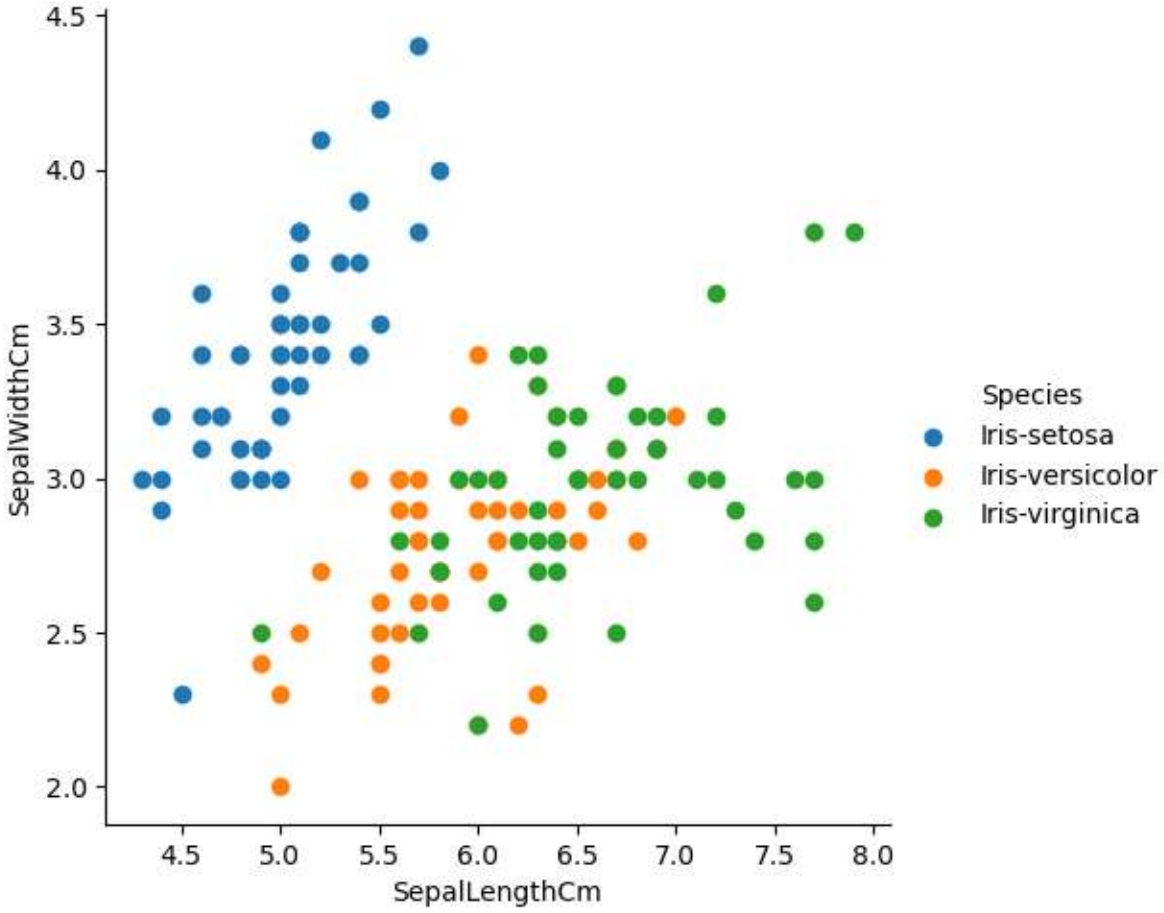
In [16]:
```
fig=sns.jointplot(x='SepalLengthCm' ,y='SepalWidthCm' ,kind='hex' ,data=iris)
```



4. Facetgrid Plot

In [17]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

sns.FacetGrid(iris, hue='Species', height=5)\
    .map(plt.scatter, 'SepalLengthCm', 'SepalWidthCm')\
    .add_legend()
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x25c07a01810>
```



5. Boxplot or Whisker Plot

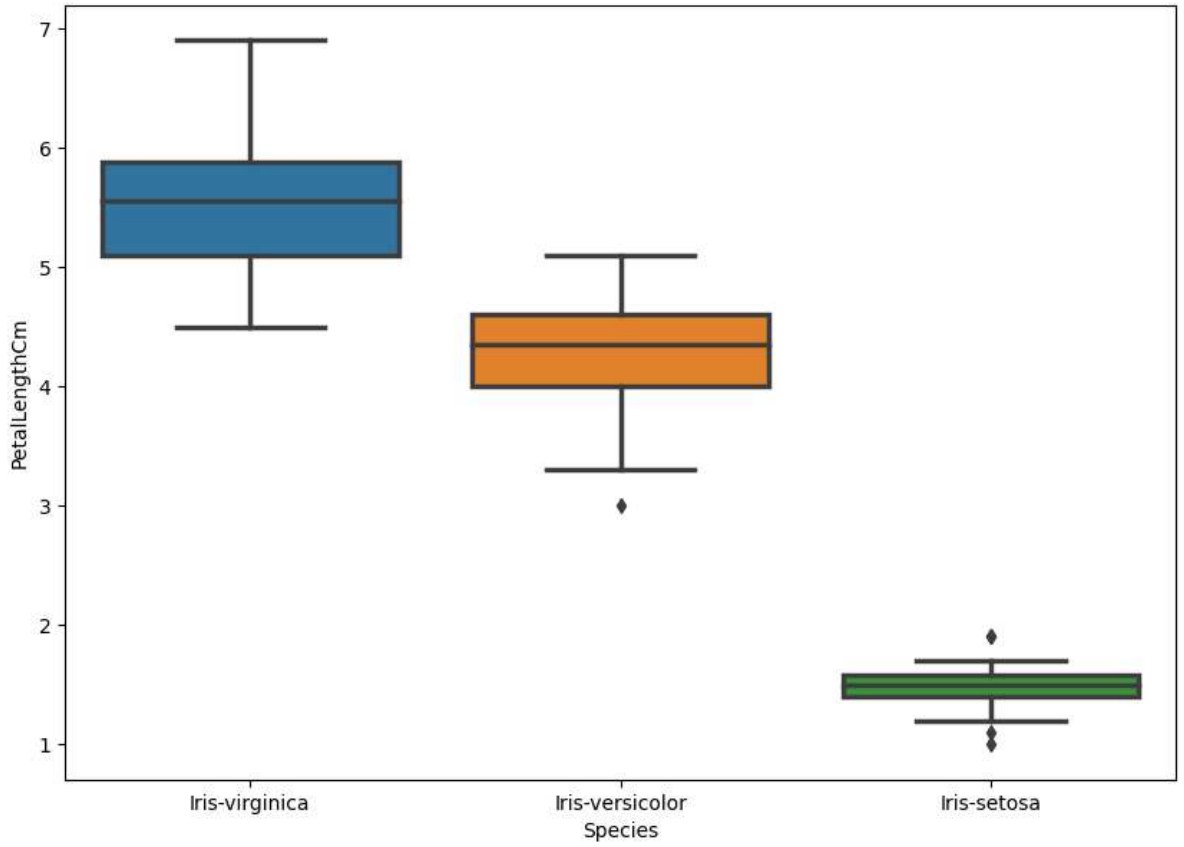
Box plot was was first introduced in year 1969 by Mathematician John Tukey.Box plot give a statical summary of the features being plotted.Top line represent the max value,top edge of box is third Quartile, middle edge represents the median,bottom edge represents the first quartile value.The bottom most 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 [18]:
```
iris.head()
```

Out[18]:

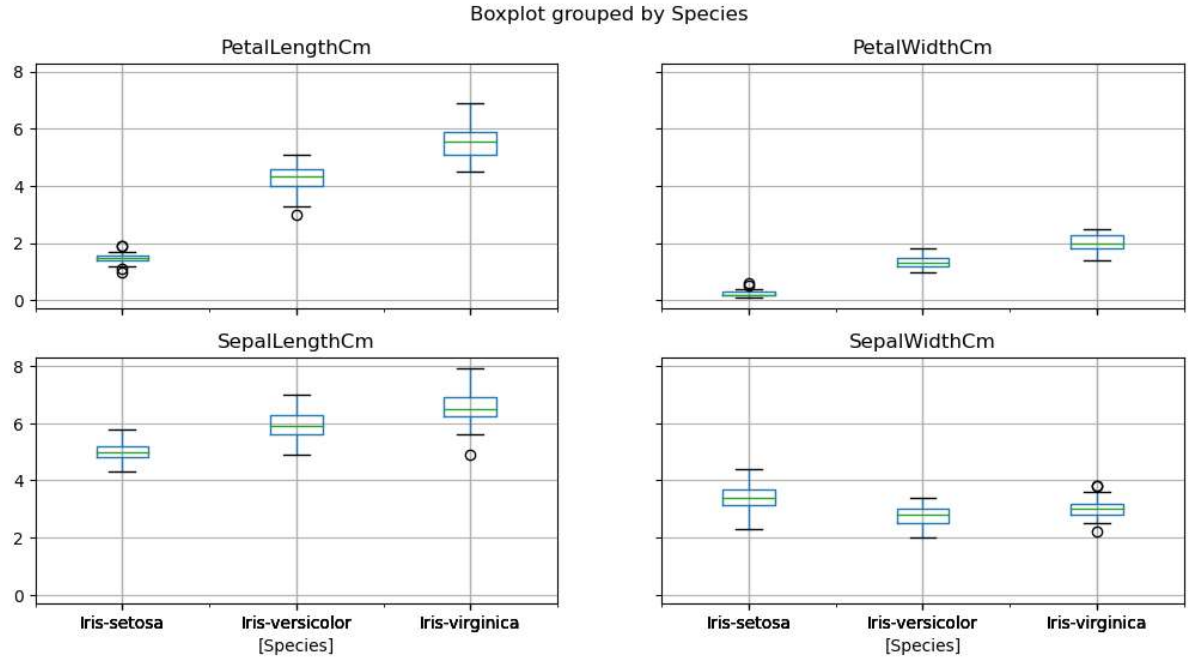
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [19]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxplot(x='Species' ,y='PetalLengthCm' ,data=iris ,order=['Iris-virgin
```



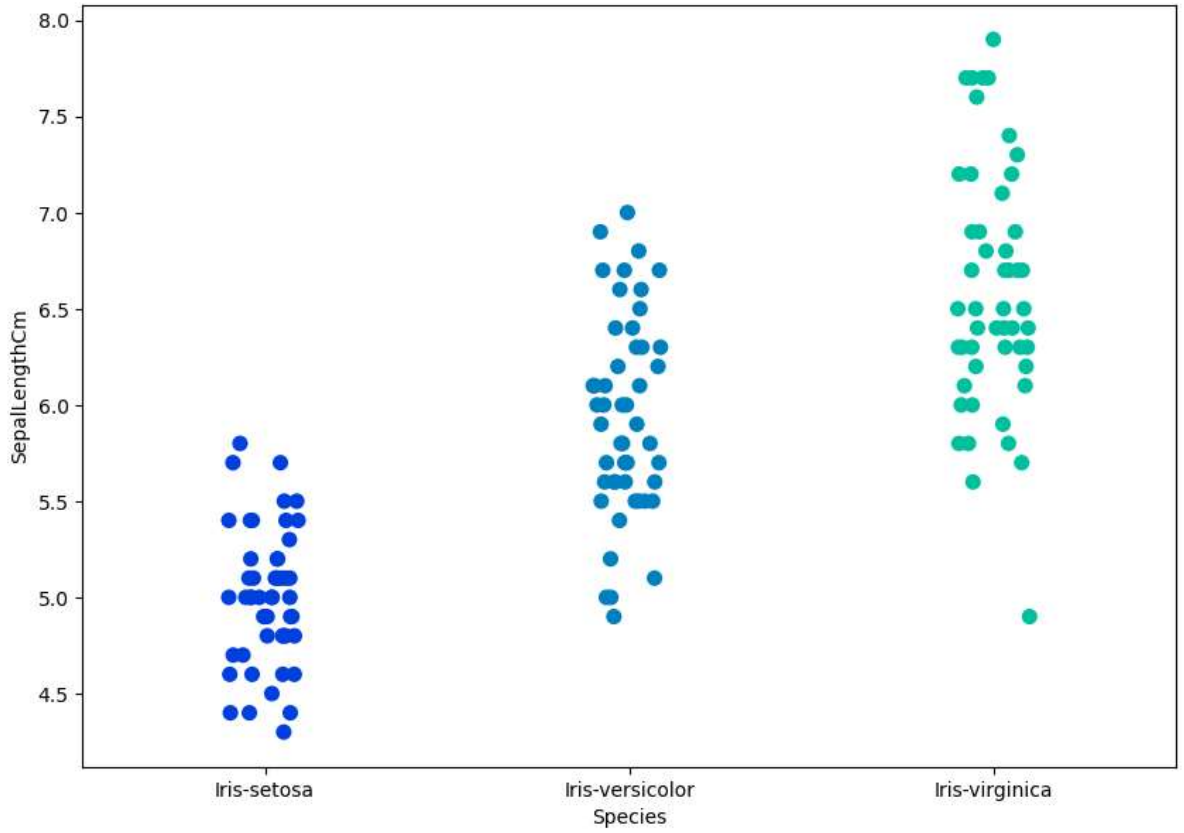
In [20]:
```
#iris.drop("Id" , axis=1).boxplot(by="Species" , figsize=(12,6))
iris.boxplot(by="Species" , figsize=(12,6))
```

Out[20]:
```
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)
```



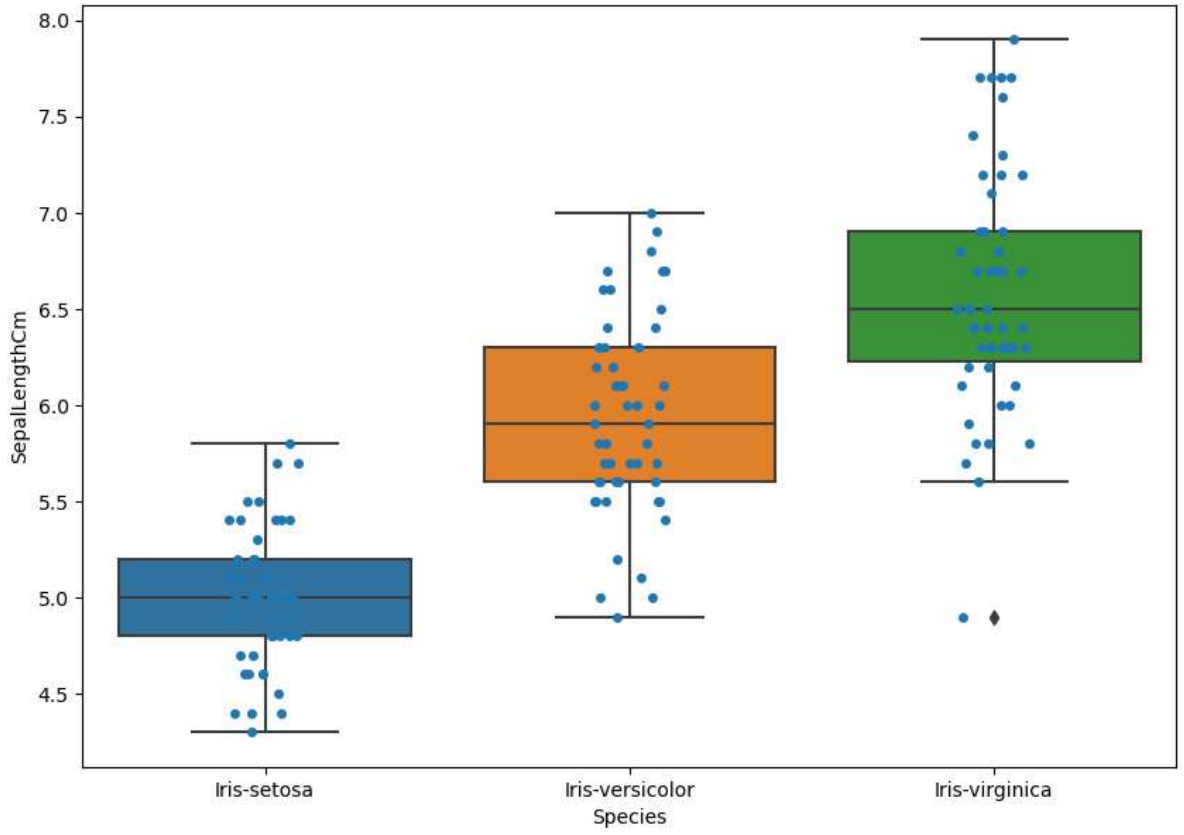
6.Strip Plot

In [21]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.stripplot(x='Species' ,y='SepalLengthCm' ,data=iris ,jitter=True,edgec
```

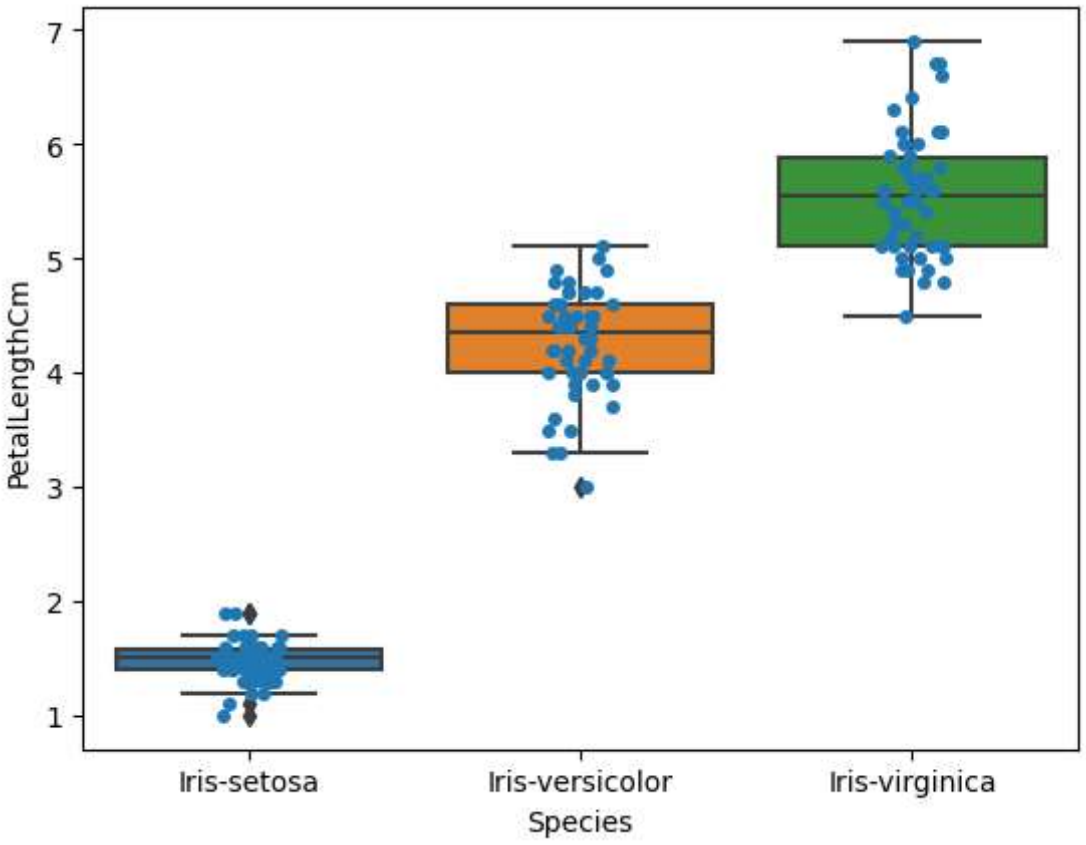


7. Combining Box and Strip Plots

In [22]:
```
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 ,edgeco
```

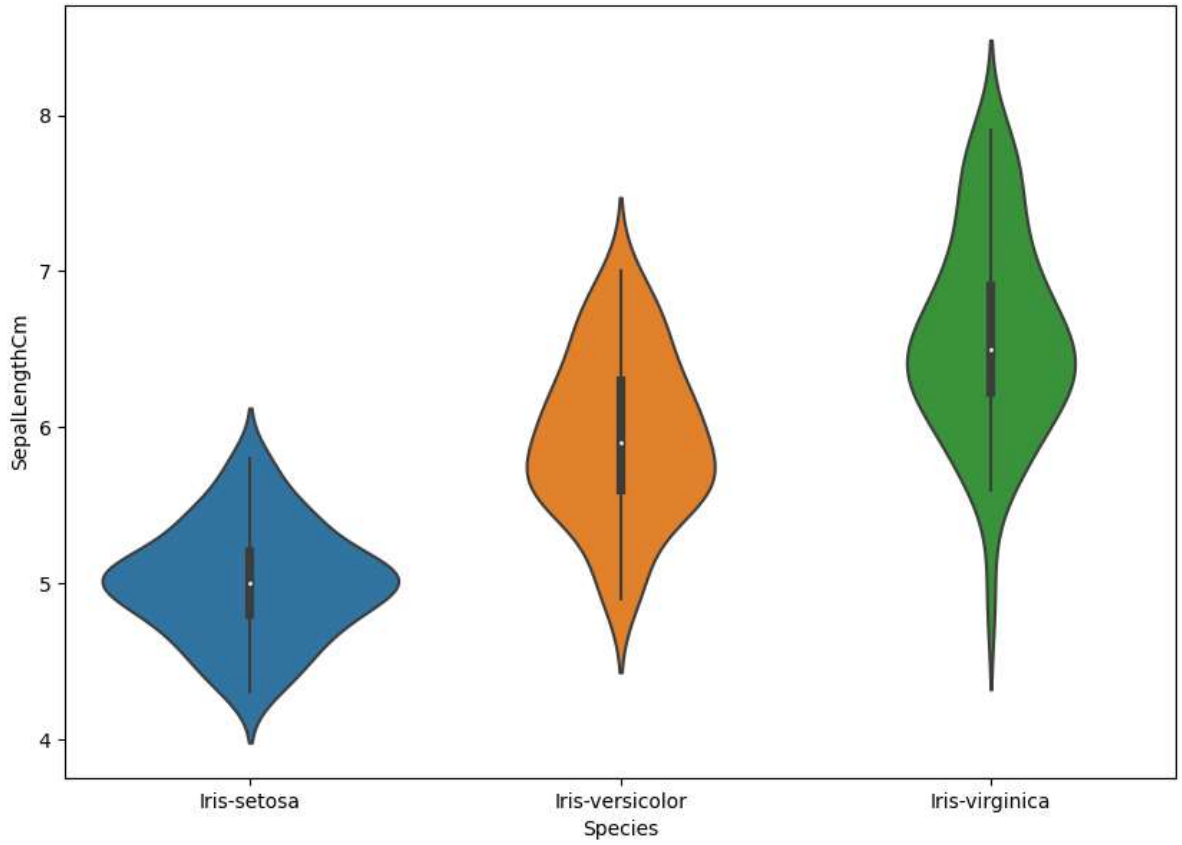


In [23]:
```
# Assuming 'iris' is a DataFrame containing the required data

ax = sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
ax = sns.stripplot(x="Species", y="PetalLengthCm", data=iris, jitter=True, edg

# Check the number of artists created by the boxplot
num_boxes = len(ax.artists)

# Access the elements in a safe way
if num_boxes > 2:
    boxtwo = ax.artists[2]
    boxtwo.set_facecolor('yellow')
    boxtwo.set_edgecolor('black')

if num_boxes > 1:
    boxthree = ax.artists[1]
    boxthree.set_facecolor('red')
    boxthree.set_edgecolor('black')

if num_boxes > 0:
    boxone = ax.artists[0]
    boxone.set_facecolor('green')
    boxone.set_edgecolor('black')

plt.show()
```



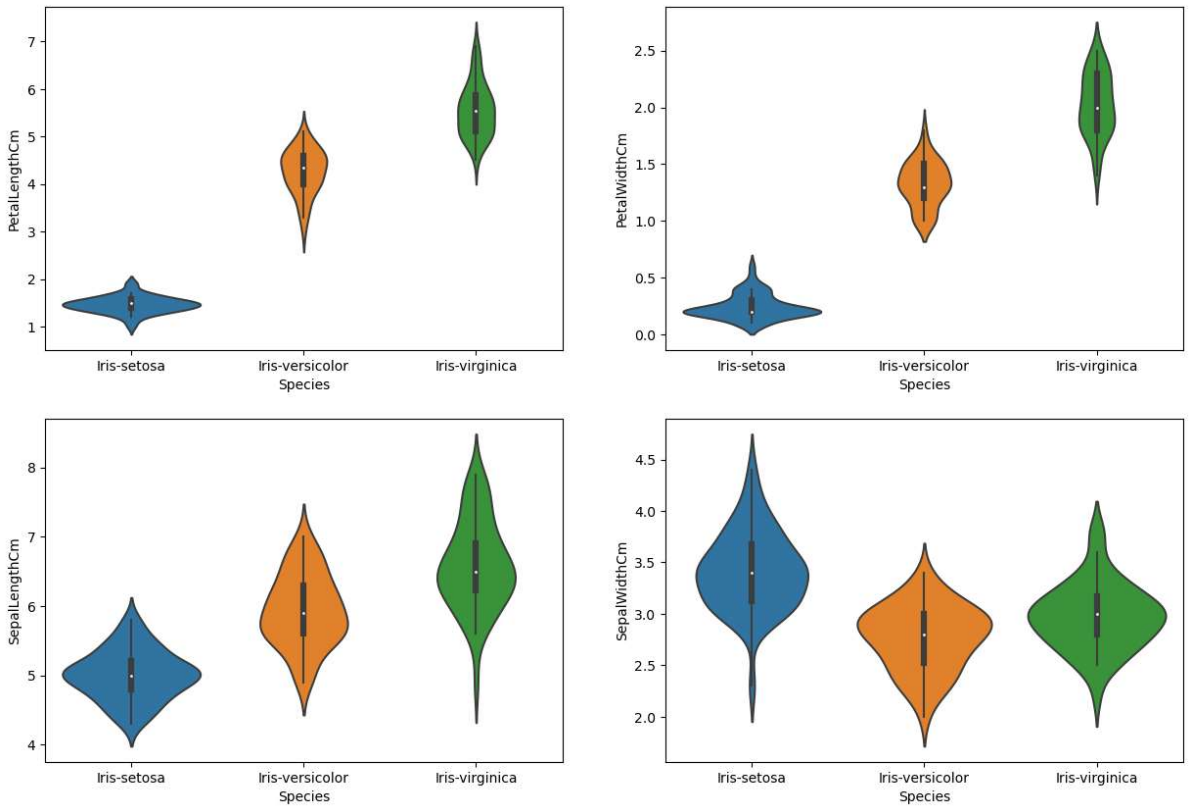
8. Violin Plot:- It is used to visualize the distribution of data and its probability distribution.This chart is a combination of a Box Plot and a Density Plot that is rotated and placed on each side, to show the distribution shape of the data. The thick black bar in the centre represents the interquartile 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 [24]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
```



In [25]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
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)
```

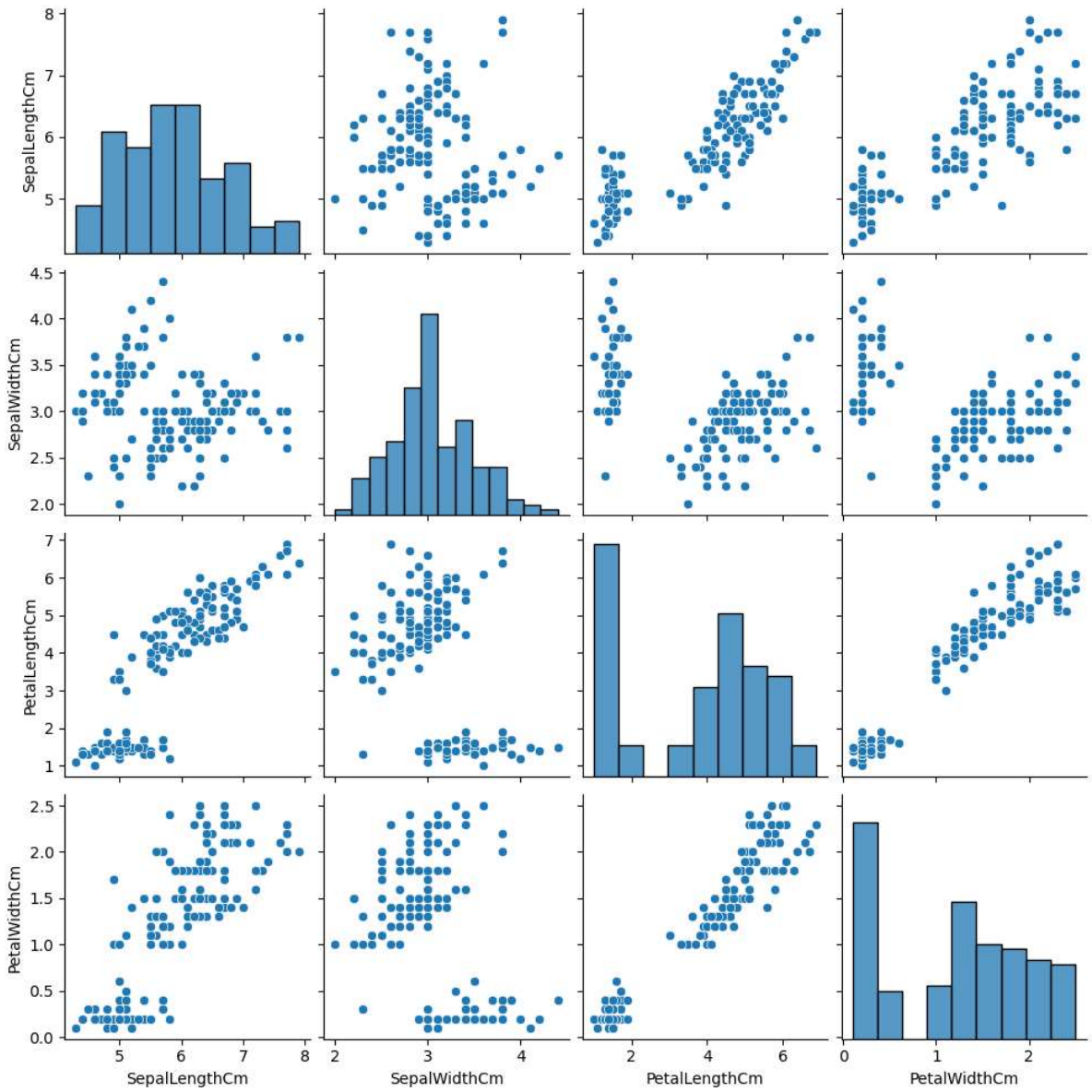
Out[25]: <Axes: xlabel='Species', ylabel='SepalWidthCm'>



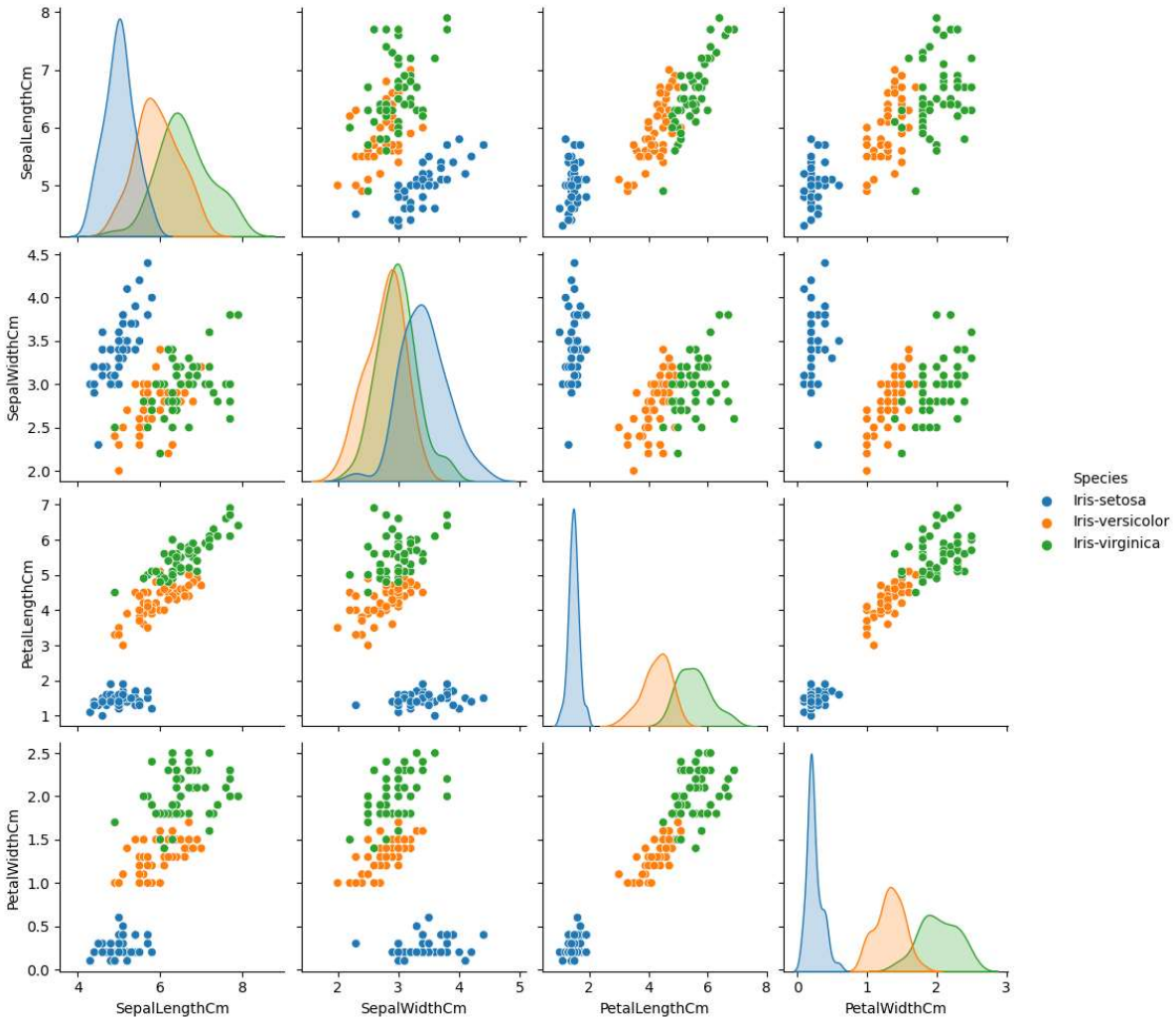
19. Pair Plot:- A “pairs plot” is also known as a scatterplot, in which one variable in the same data row is matched with another variable's value, like this: Pairs plots are just elaborations on this, showing all variables paired with all the other variables.

In [26]:
```
sns.pairplot(data=iris,kind='scatter')
```

Out[26]:
```
<seaborn.axisgrid.PairGrid at 0x25c08925150>
```



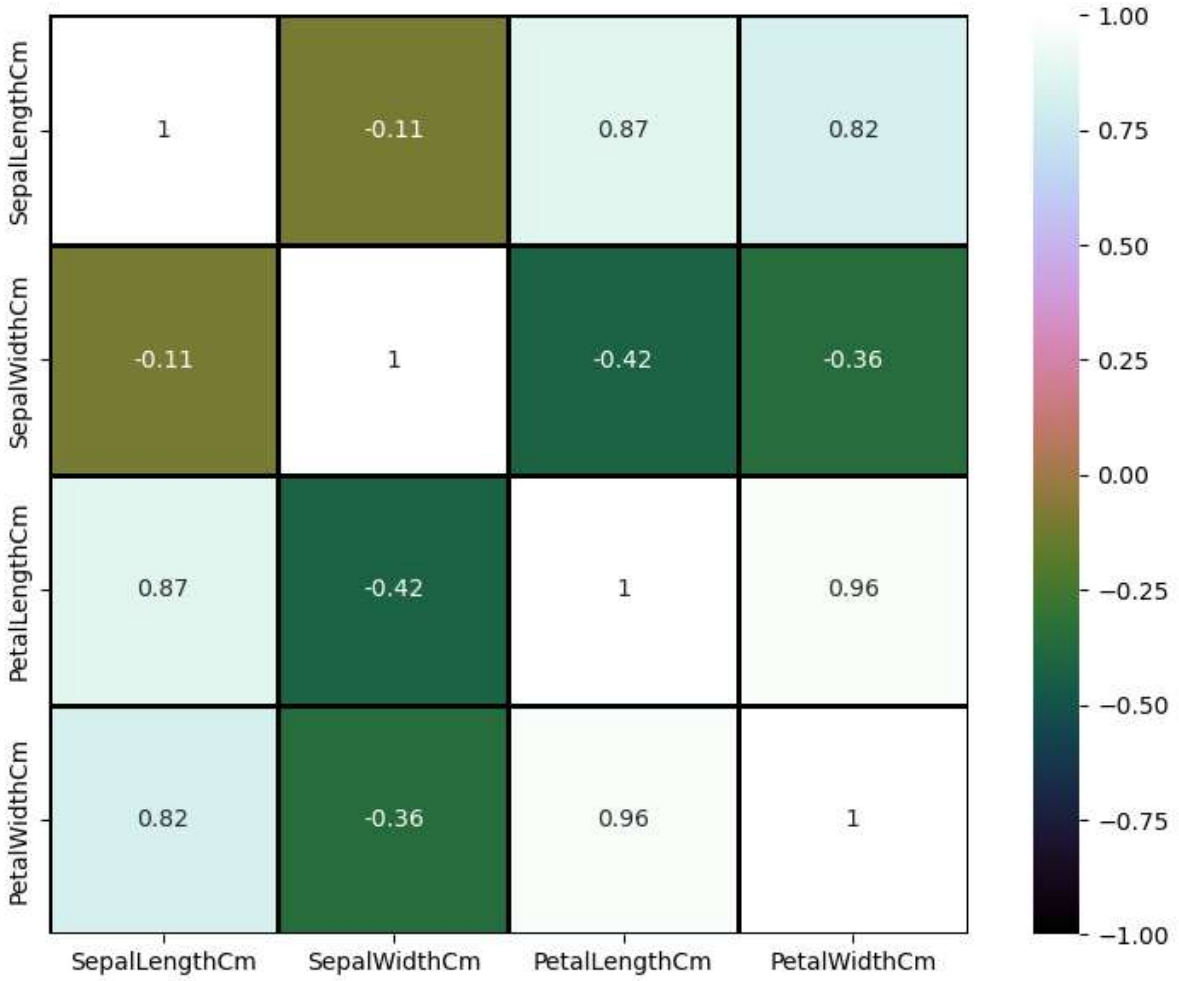
In [27]:
```
sns.pairplot(data=iris,hue='Species');
```



In [ ]:

10. 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.

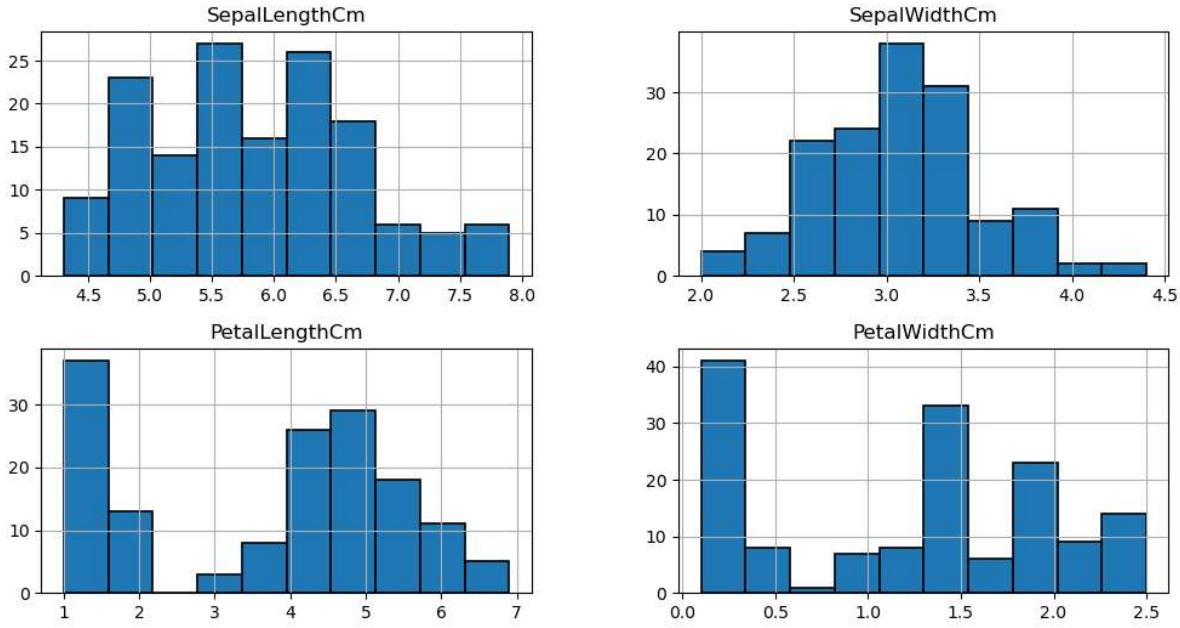
In [28]:
```
# Assuming 'iris' is a DataFrame containing the required data

# Exclude non-numeric columns before calculating correlation matrix
numeric_columns = iris.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numeric_columns.corr()

# Create the heatmap
fig = plt.gcf()
fig.set_size_inches(10, 7)
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='cubehelix', linewi

plt.show()
```



11. 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

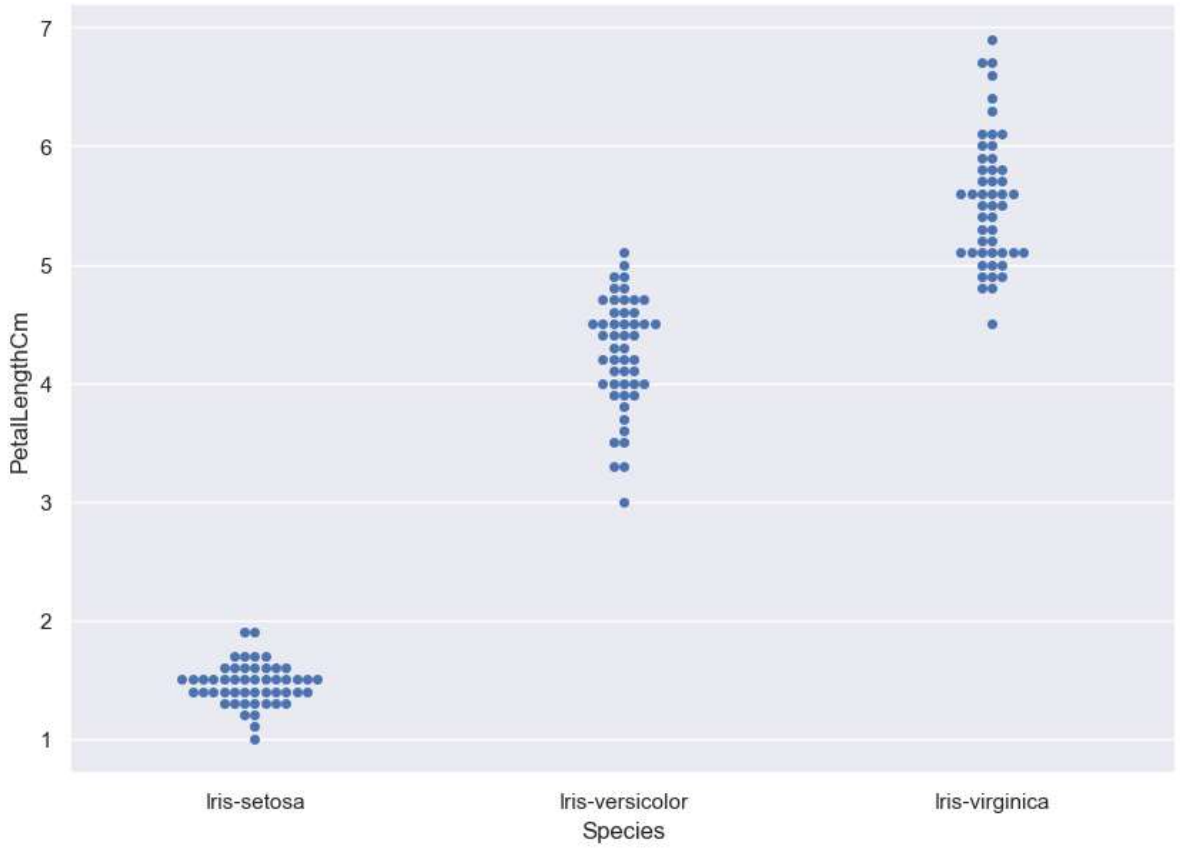
value points to see the distribution of values, a bounding box to see the range of values, or a combination of both as shown here.The distribution plot is not relevant for detailed analysis of the data as it deals with a summary of the data distribution.

In [29]:
```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
```

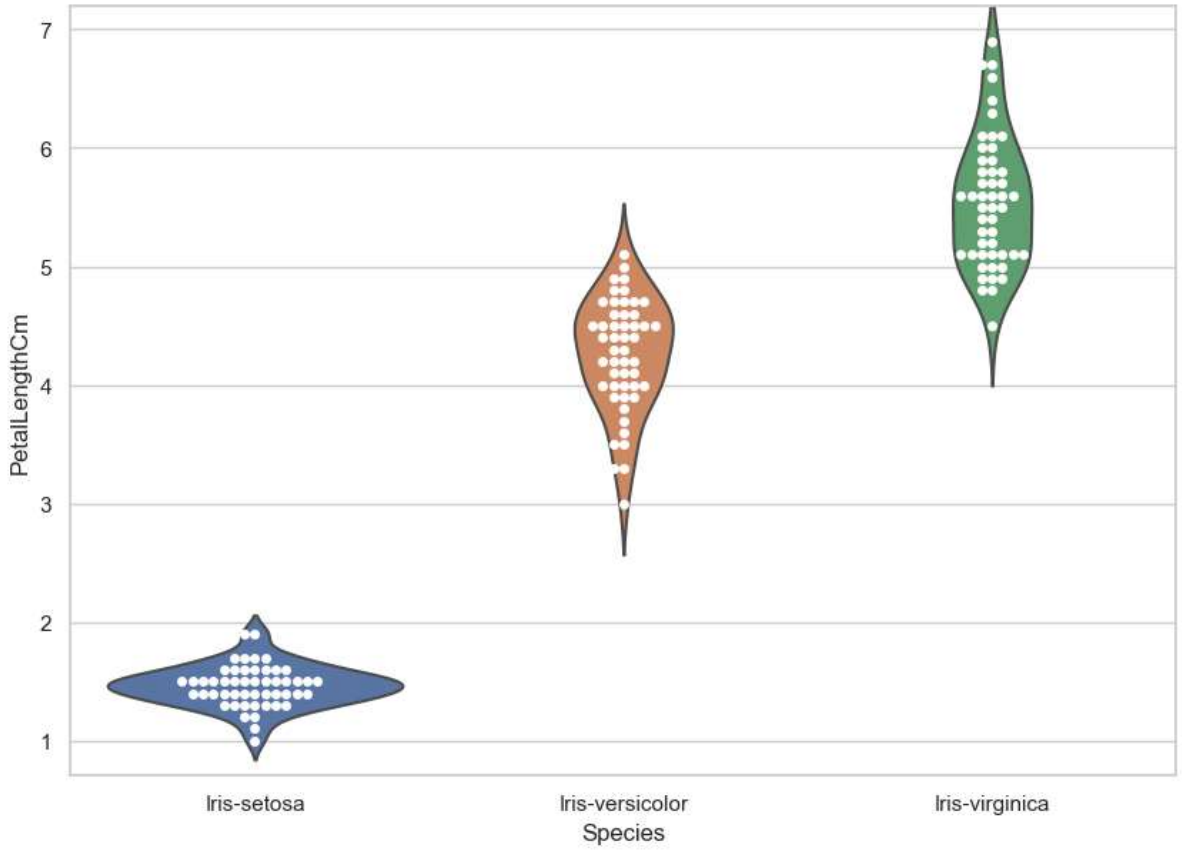


12. Swarm plot:- It looks a bit like a friendly swarm of bees buzzing about their hive. More importantly, each data point is clearly visible and no data are obscured by overplotting.A beeswarm plot improves upon the random jittering approach to move data points the minimum distance away from one another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

In [30]:
```
sns.set(style="darkgrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
fig = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris)
```

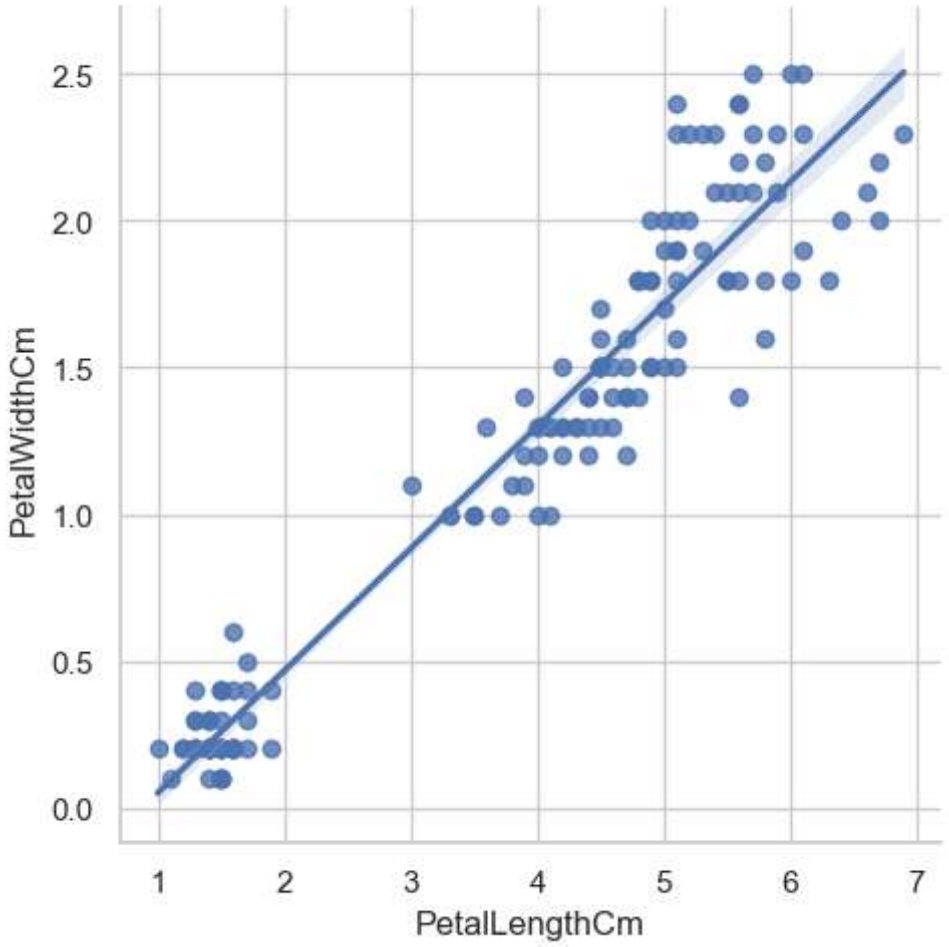


In [31]:
```
sns.set(style="whitegrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
ax = sns.violinplot(x="Species", y="PetalLengthCm", data=iris, inner=None)
ax = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris,color="white", ed
```



13. LM Plot

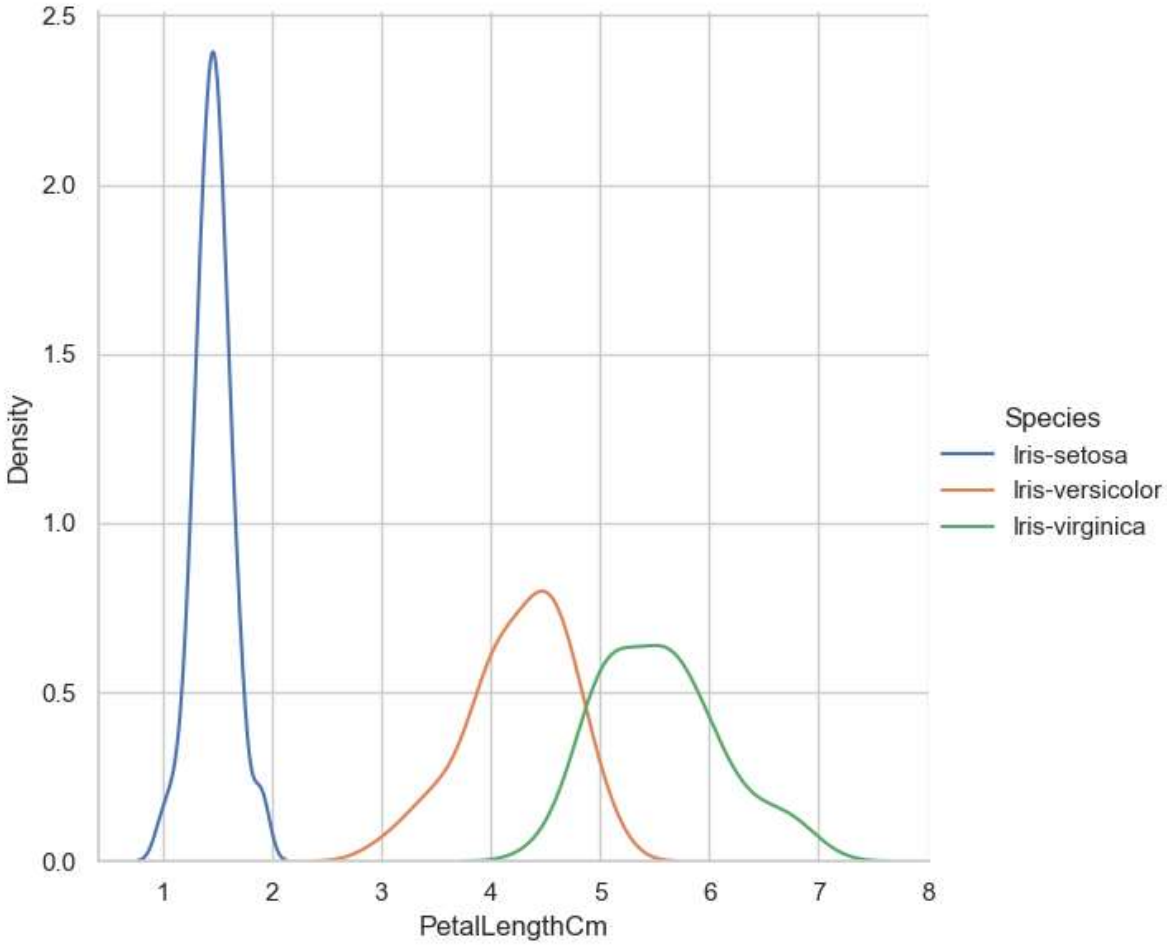
In [32]:
```
fig=sns.lmplot(x="PetalLengthCm" , y="PetalWidthCm" ,data=iris)
```



14. Facetgrid

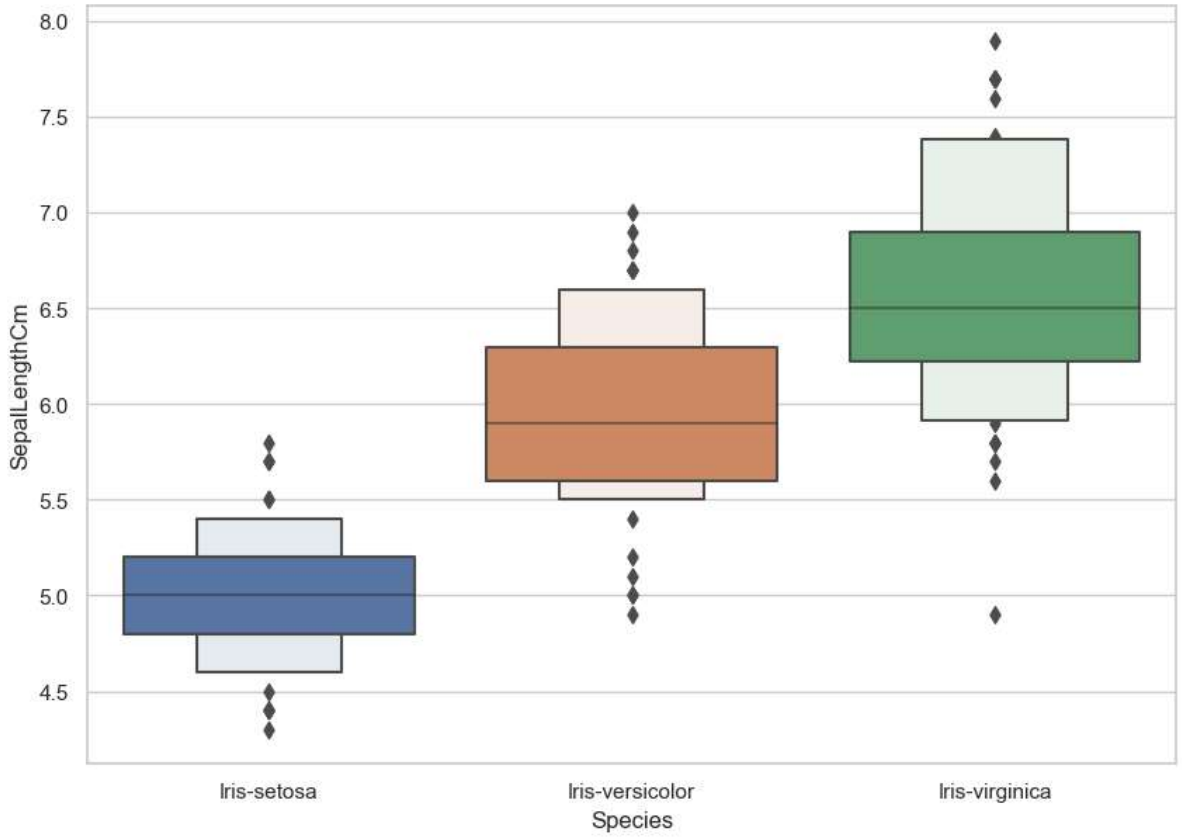
In [33]:
```
sns.FacetGrid(iris ,hue="Species" ,height=6) \
.map(sns.kdeplot ,"PetalLengthCm") \
.add_legend()
plt.ioff()
```

Out[33]:
```
<contextlib.ExitStack at 0x25c0d102510>
```



** 15. Boxen Plot**

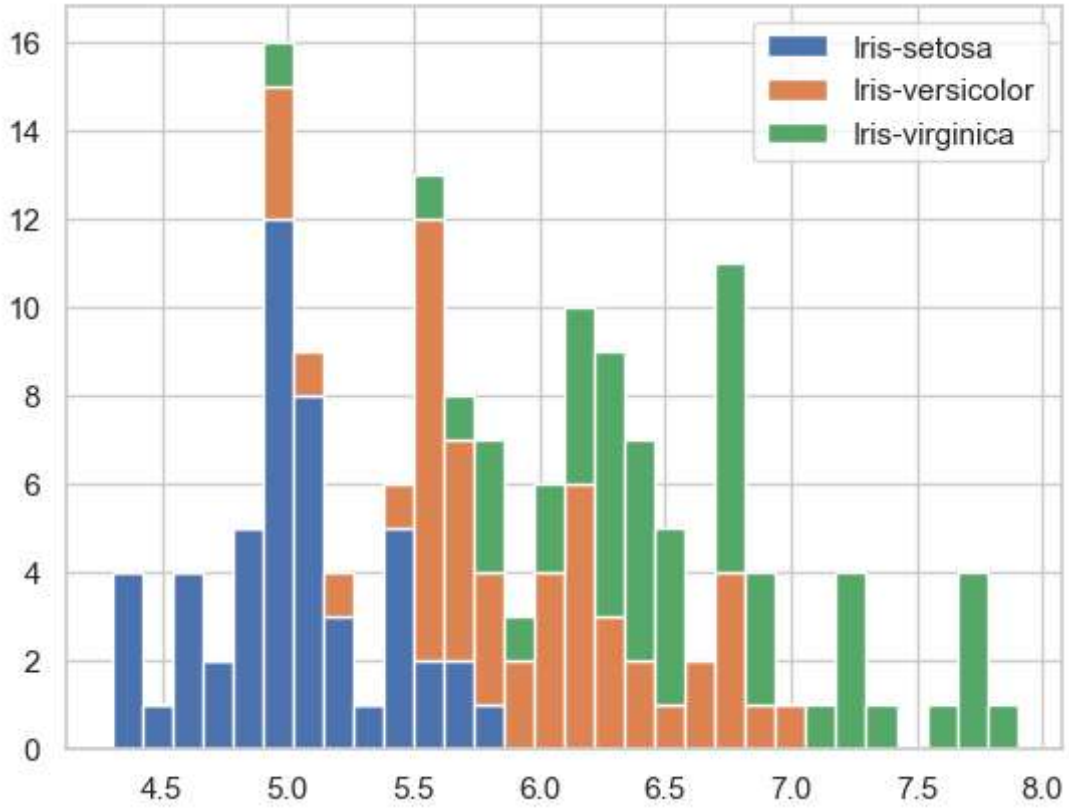
In [46]:
```
fig = plt.gcf()
fig.set_size_inches(10, 7)
fig = sns.boxenplot(x='Species', y='SepalLengthCm', data=iris)
plt.show()
```



16.Stacked Histogram

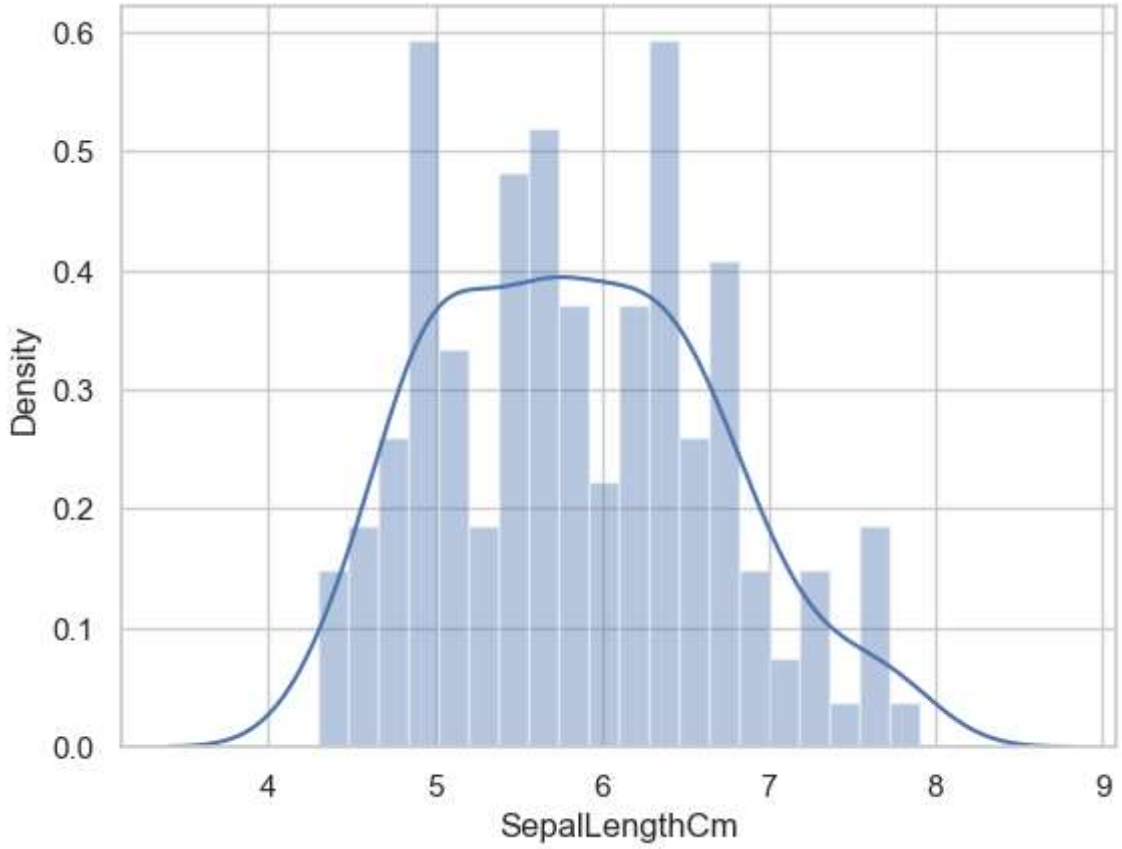
In [47]:
```
iris['Species'] = iris['Species'].astype('category')
#iris.head()
```

In [48]:
```
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()
```



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

In [49]:
```
sns.distplot(iris['SepalLengthCm'], kde=True, bins=20)
plt.show()
```



In [43]:
```
# THIS IS ALL ABOUT EDA COMPLETE
```