IRIS DATASET VISUALIZATION(SEABORN,MATPLOTLIB)

I have created this Kernel for beginners who want to learn how to plot graphs with seaborn.This kernel is still a work in progress.I will be updating it further when I find some time.If you find my work useful please fo vote by clicking at the top of the page.Thanks for viewing.

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/dock
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

Importing pandas and Seaborn module

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

Importing Iris data set

In [3]:
```
iris=pd.read_csv(r'C:\23rd,24th\IRIS DATASET _ ADVANCE VISUALIZATION _ EDA 2\Iris
```

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

*Displaying data *

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 are any missing values *

In [8]:
```
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
```

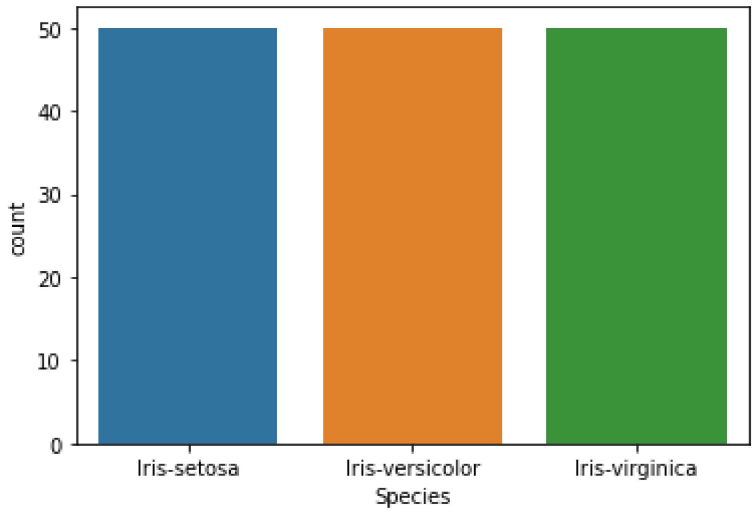
In [9]:
```
iris['Species'].value_counts()
```

Out[9]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

This data set 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 [10]:
```
sns.countplot('Species',data=iris)
plt.show()
```



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

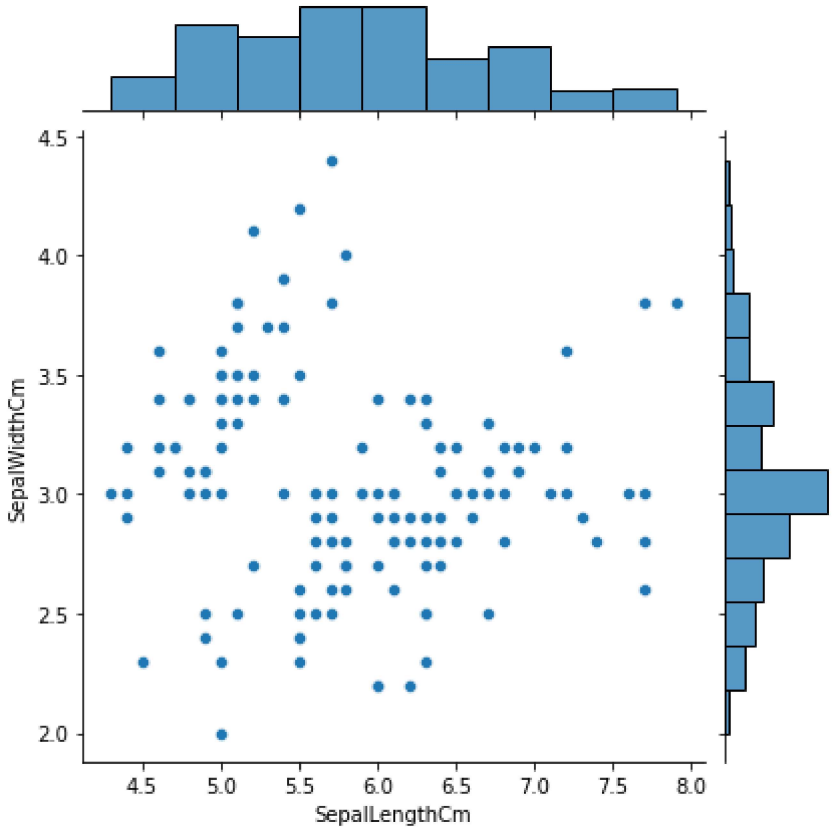
**4. Joint plot: * Jointplot is seaborn library specific and can be used to quickly visualize and analyze the relationship between two variables and describe their individual distributions on the same plot.

In [11]:
```
iris.head()
```

Out[11]:

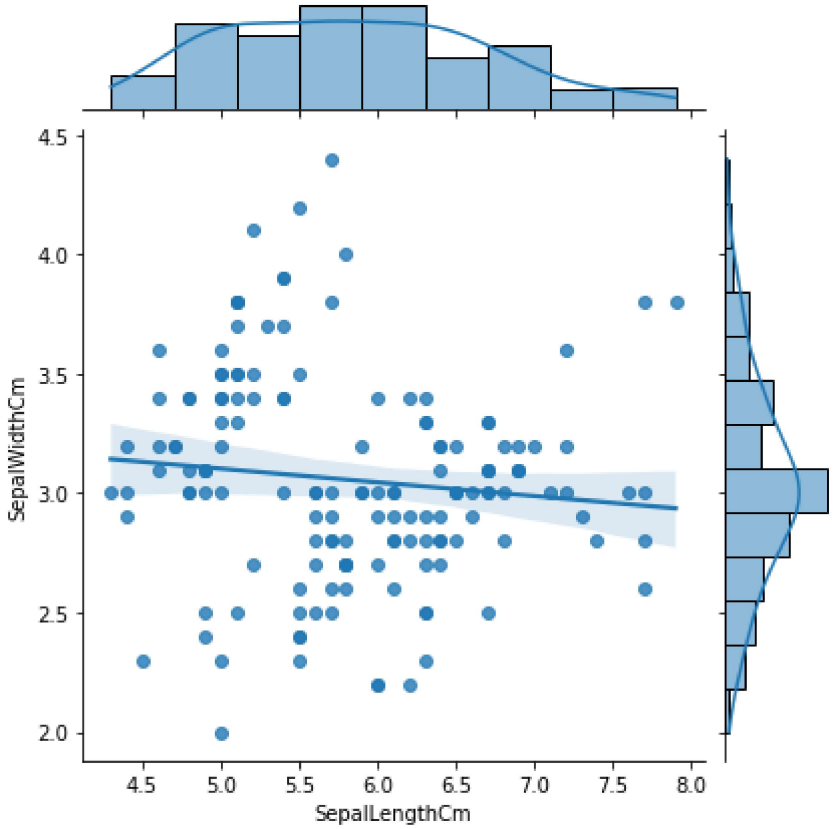
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [12]:
```
fig=sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',data=iris)
```

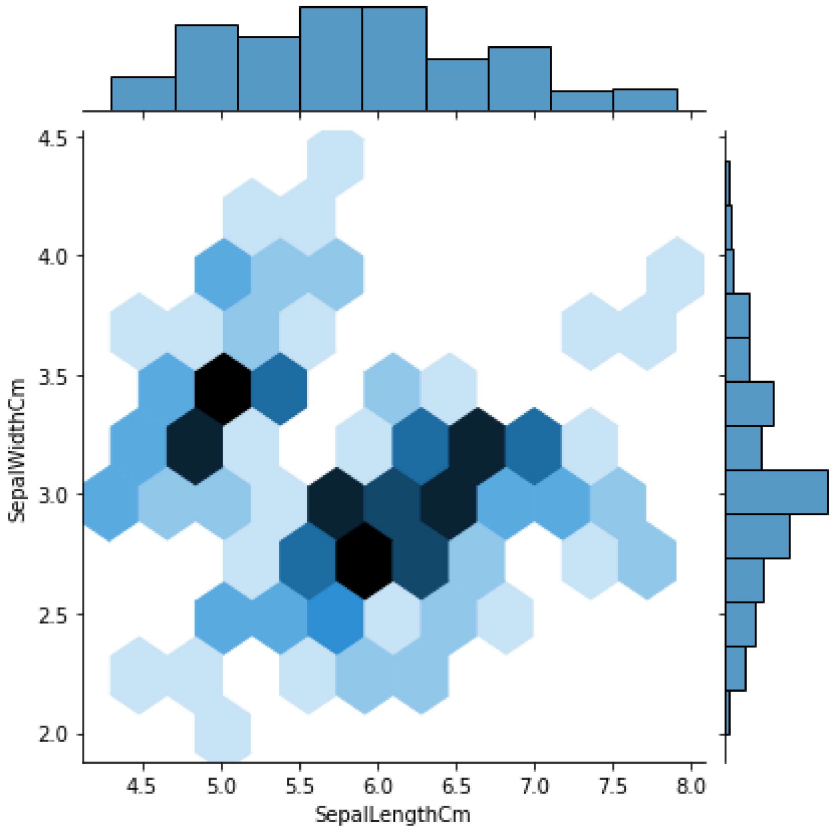


In [13]:
```
sns.jointplot("SepalLengthCm", "SepalWidthCm", data=iris, kind="reg")
```

Out[13]:
```
<seaborn.axisgrid.JointGrid at 0x1f118f3ee80>
```



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



5. FacetGrid Plot

In [15]:
```
import matplotlib.pyplot as plt
%matplotlib inline

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

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x1f118db1af0>
```

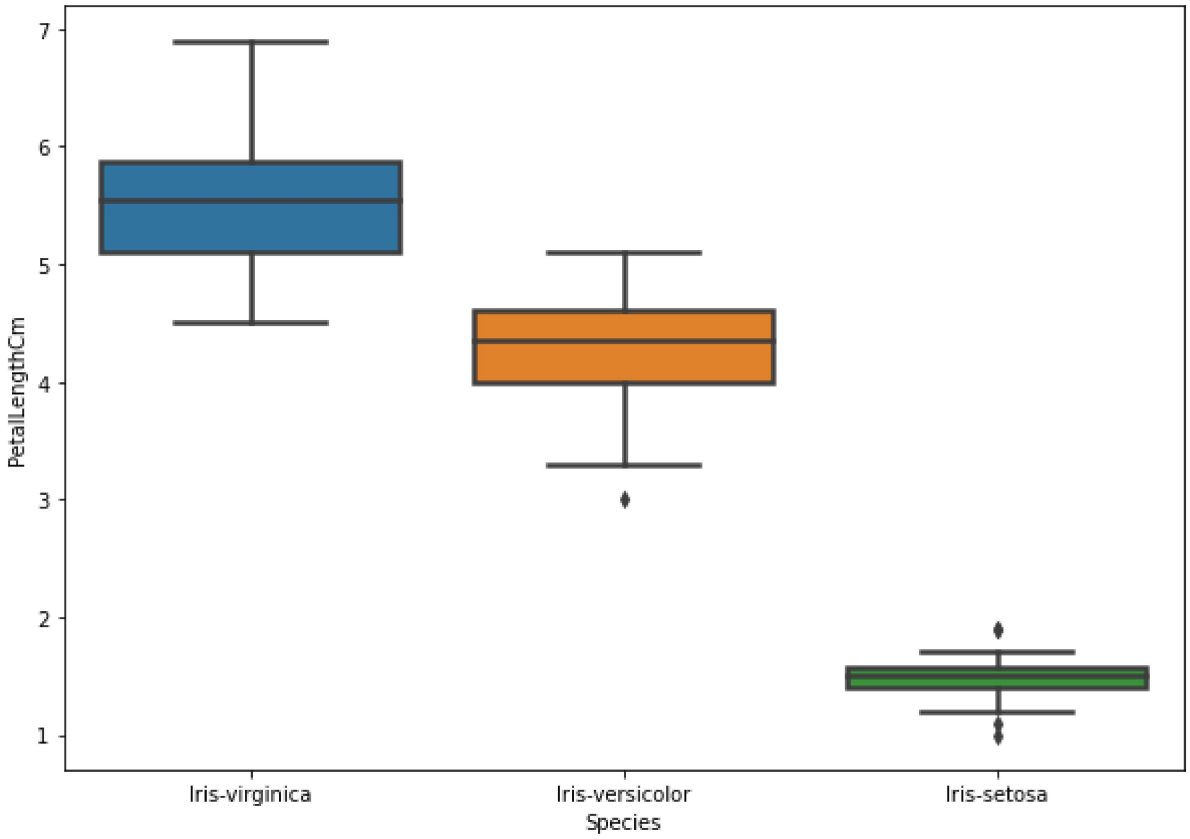
6. Boxplot or Whisker plot Box plot was was first introduced in year 1969 by Mathematician John Tukey.Box plot give a statical summary of the features being plotted.Top line represent the max value,top edge of box is third Quartile, middle edge represents the median,bottom edge represents the first quartile 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 [16]:
```
iris.head()
```

Out[16]:

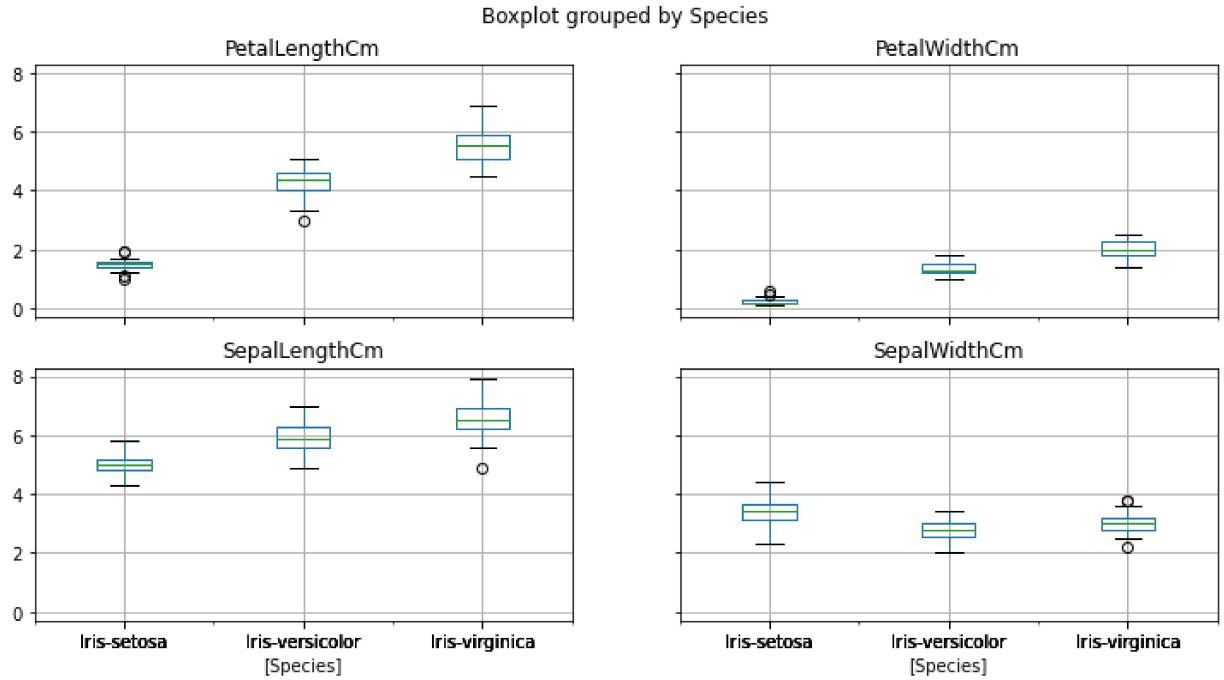
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [17]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxplot(x='Species',y='PetalLengthCm',data=iris,order=['Iris-virginica',
```



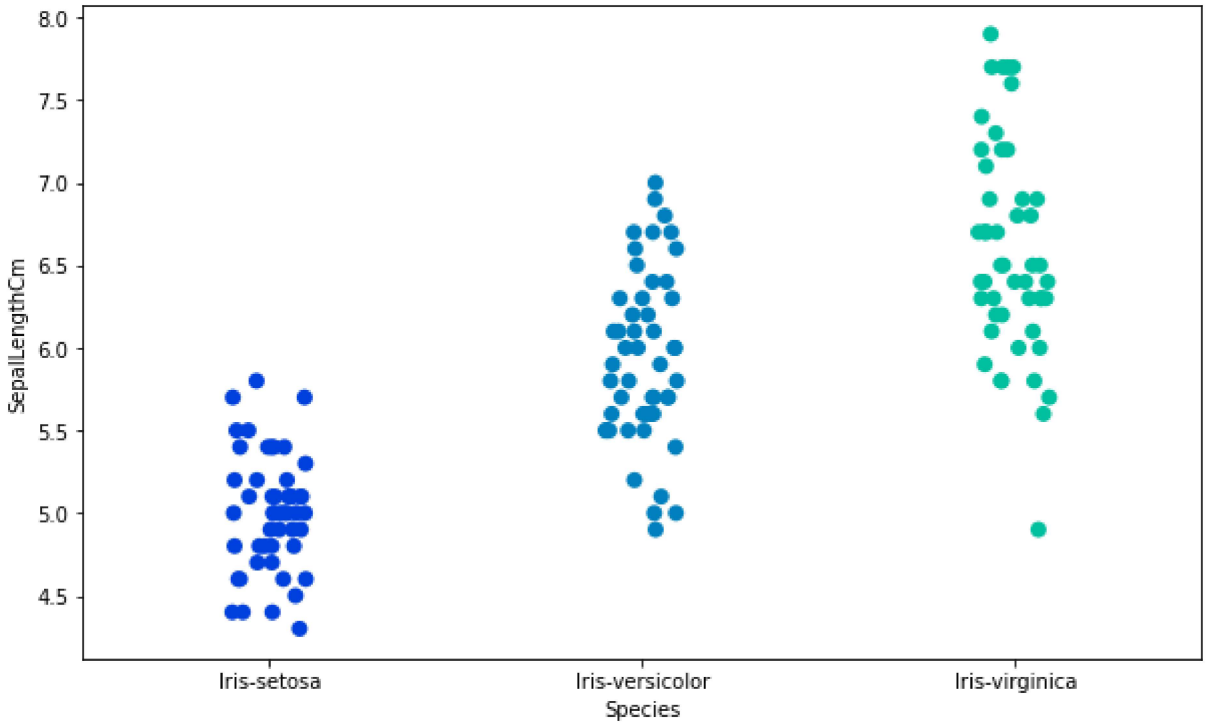
In [24]:
```
#iris.drop("Id", axis=1).boxplot(by="Species", figsize=(12, 6))
iris.boxplot(by="Species", figsize=(12, 6))
```

Out[24]:
```
array([[<AxesSubplot:title={'center':'PetalLengthCm'}, xlabel='[Species]'>,
        <AxesSubplot:title={'center':'PetalWidthCm'}, xlabel='[Species]'>],
       [<AxesSubplot:title={'center':'SepalLengthCm'}, xlabel='[Species]'>,
        <AxesSubplot:title={'center':'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



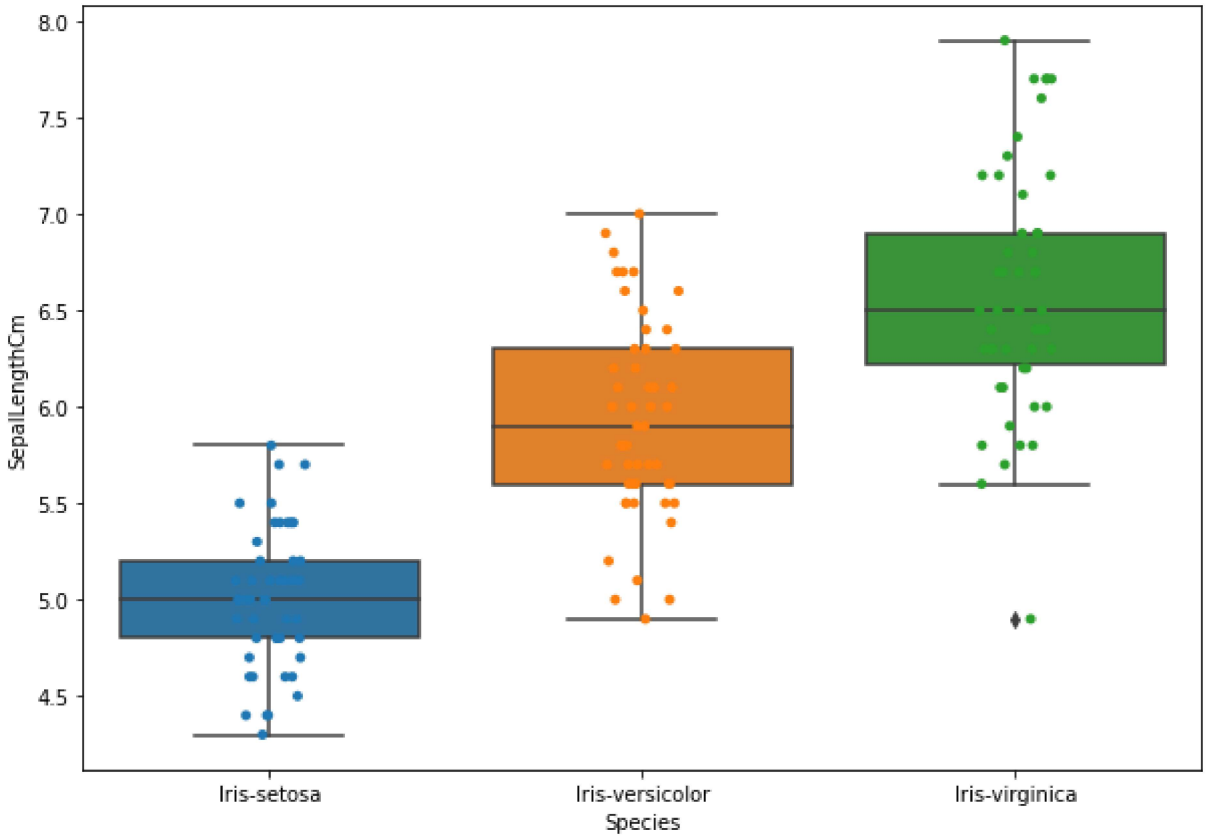
7. Strip plot

In [28]:
```
fig=plt.gcf()
fig.set_size_inches(10,6)
fig=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor='
```



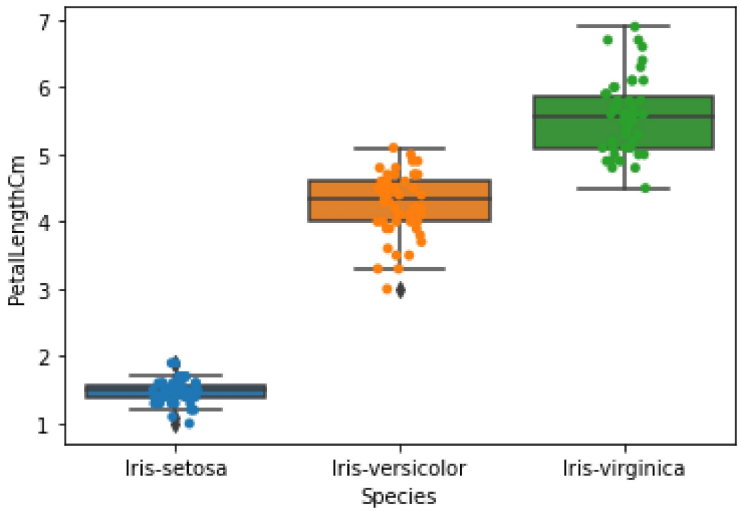
8. Combining Box and Strip Plots

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='
```



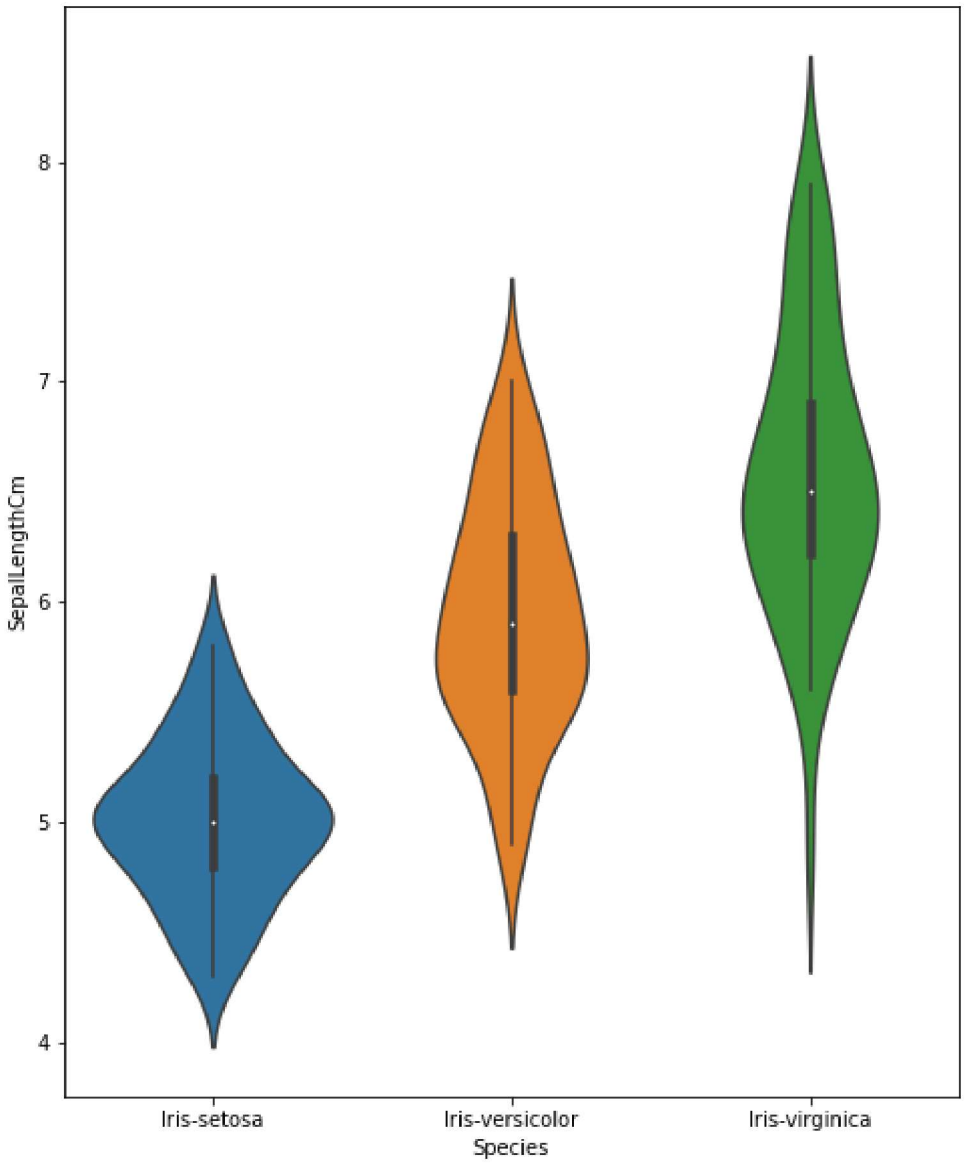
In [30]:
```
ax= sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
ax= sns.stripplot(x="Species", y="PetalLengthCm", data=iris, jitter=True, edgecol

boxtwo = ax.artists[2]
boxtwo.set_facecolor('yellow')
boxtwo.set_edgecolor('black')
boxthree=ax.artists[1]
boxthree.set_facecolor('red')
boxthree.set_edgecolor('black')
boxthree=ax.artists[0]
boxthree.set_facecolor('green')
boxthree.set_edgecolor('black')

plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [30], in <cell line: 4>()
      1 ax= sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
      2 ax= sns.stripplot(x="Species", y="PetalLengthCm", data=iris, jitter=Tru
e, edgecolor="gray")
----> 4 boxtwo = ax.artists[2]
      5 boxtwo.set_facecolor('yellow')
      6 boxtwo.set_edgecolor('black')

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:1368, in _AxesBase.
ArtistList.__getitem__(self, key)
   1367 def __getitem__(self, key):
-> 1368     return [artist
   1369             for artist in self._axes._children
   1370             if self._type_check(artist)][key]

IndexError: list index out of range
```



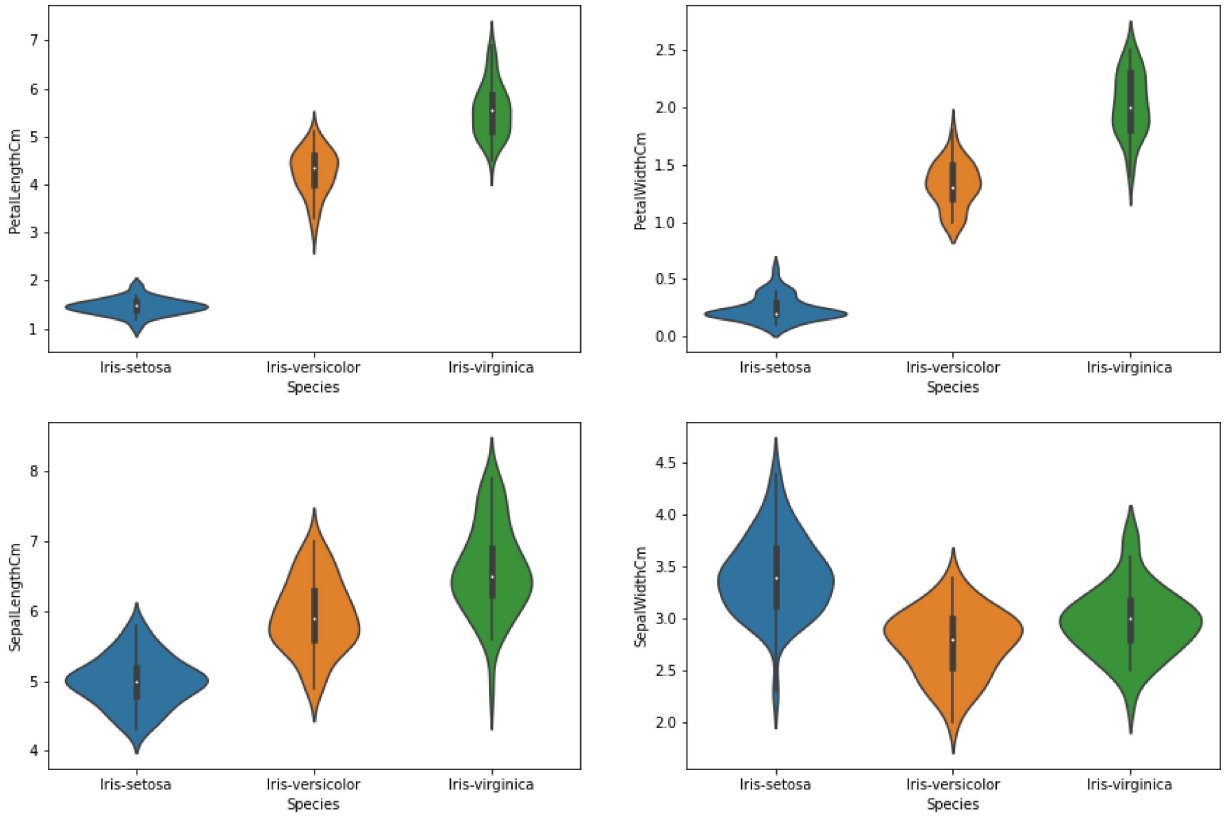
9. Violin Plot It is used to visualize the distribution of data and its probability distribution.This chart is a combination of a Box Plot and a Density Plot that is rotated and placed on each side, to show the distribution shape of the data. The thick black bar in the centre represents the interquartile 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 [34]:
```
fig=plt.gcf()
fig.set_size_inches(8,10)
fig=sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
```



In [35]:
```
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)
```

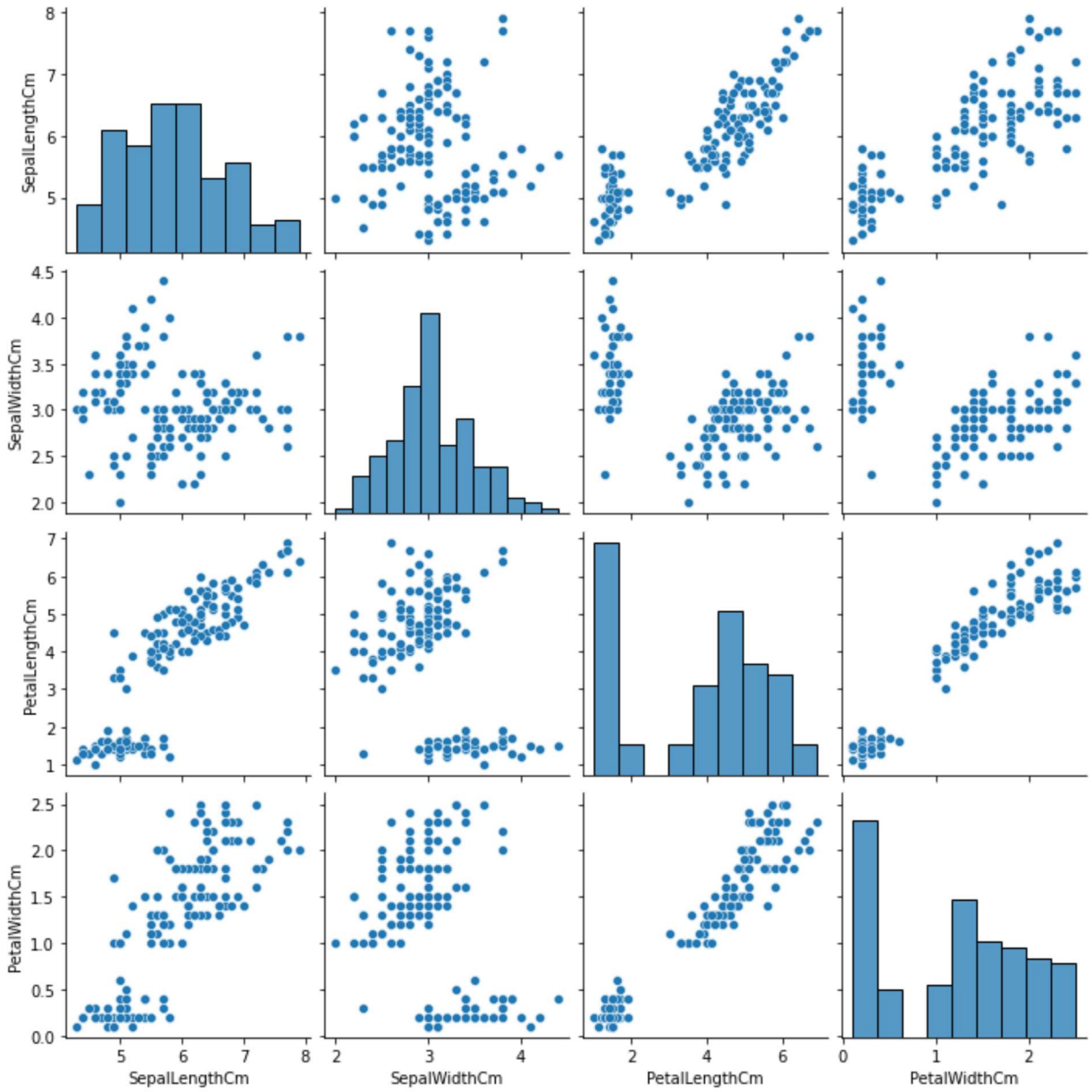
Out[35]: <AxesSubplot:xlabel='Species', ylabel='SepalWidthCm'>



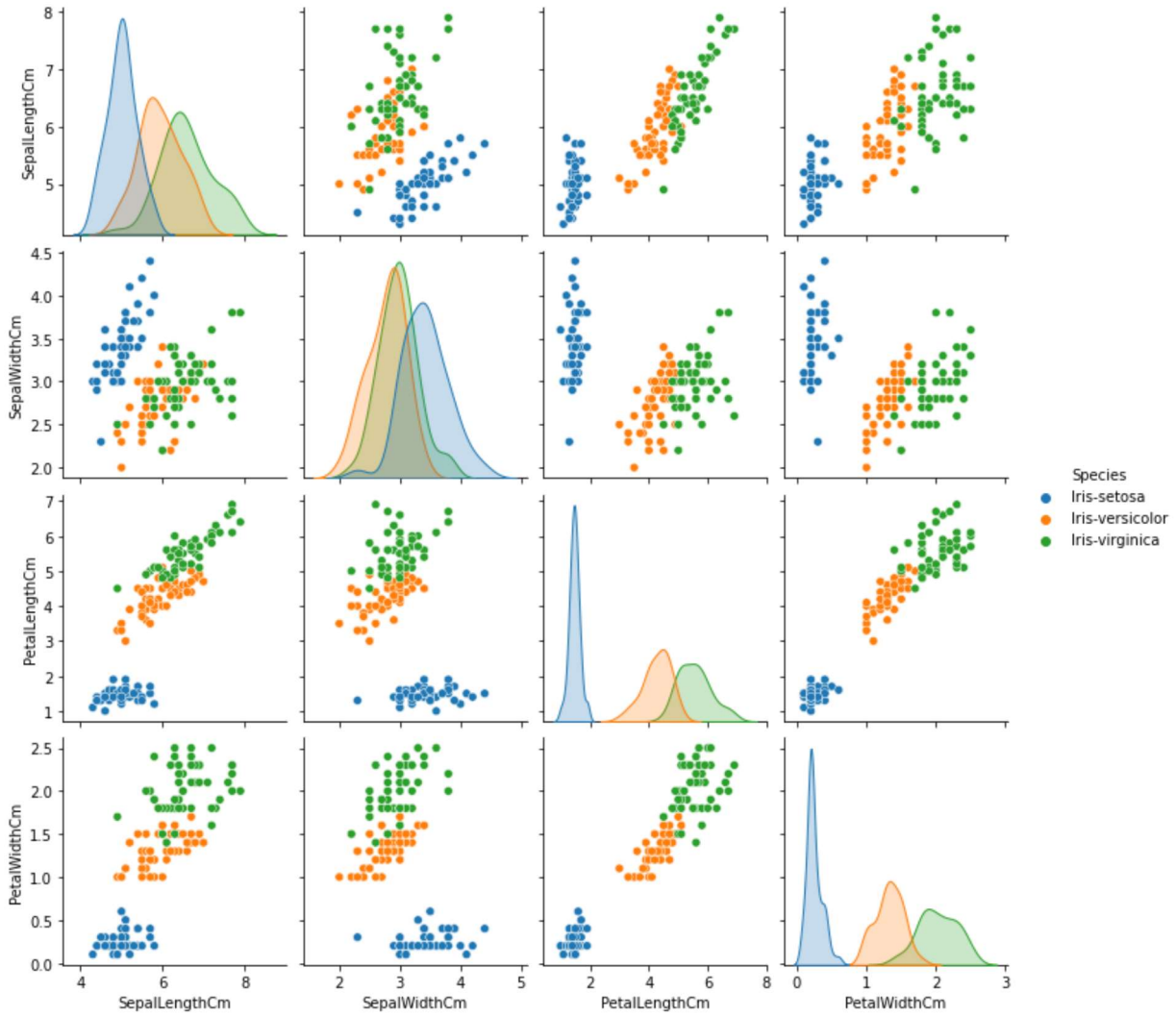
10. Pair Plot: A “pairs plot” is also known as a scatterplot, in which one variable in the same data row is matched with another variable's value, like this: Pairs plots are just elaborations on this, showing all variables paired with all the other variables.

In [36]:
```
sns.pairplot(data=iris,kind='scatter')
```

Out[36]:
```
<seaborn.axisgrid.PairGrid at 0x1f11f1420a0>
```

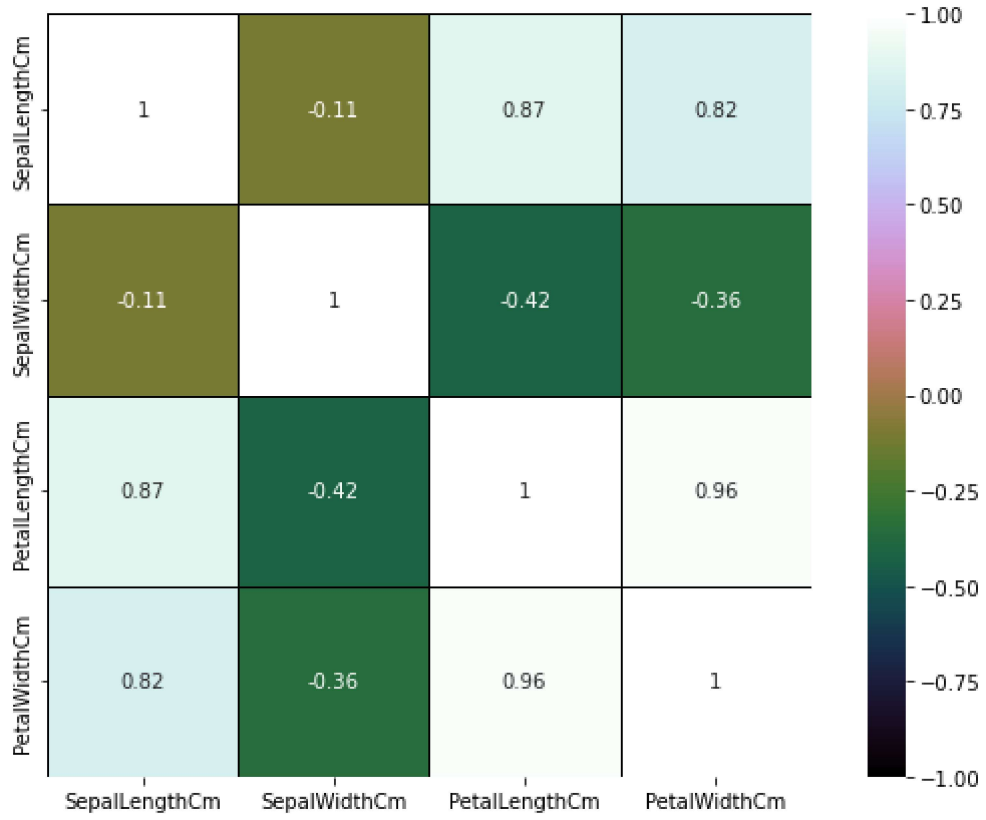


In [37]:
```
sns.pairplot(iris,hue='Species');
```



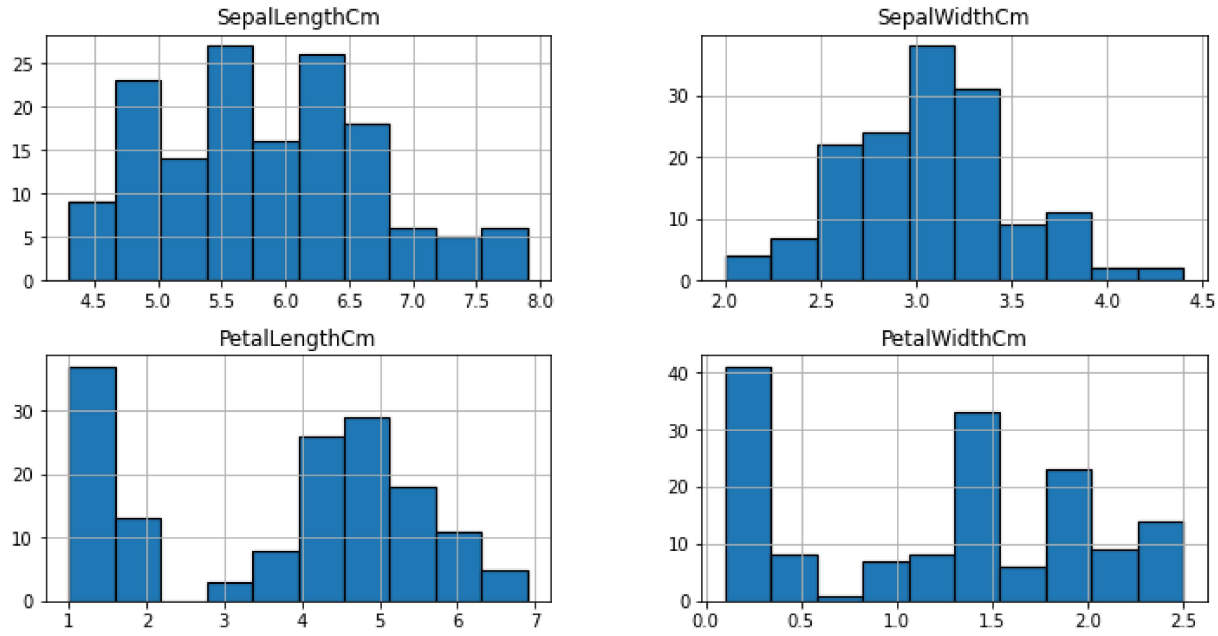
11. Heat map Heat map is used to find out the correlation between different features in the dataset.High positive or negative value shows that the features have high correlation.This helps us to select the parmeters for machine learning.

In [38]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.heatmap(iris.corr(),annot=True,cmap='cubehelix',linewidths=1,linecolor='k
```



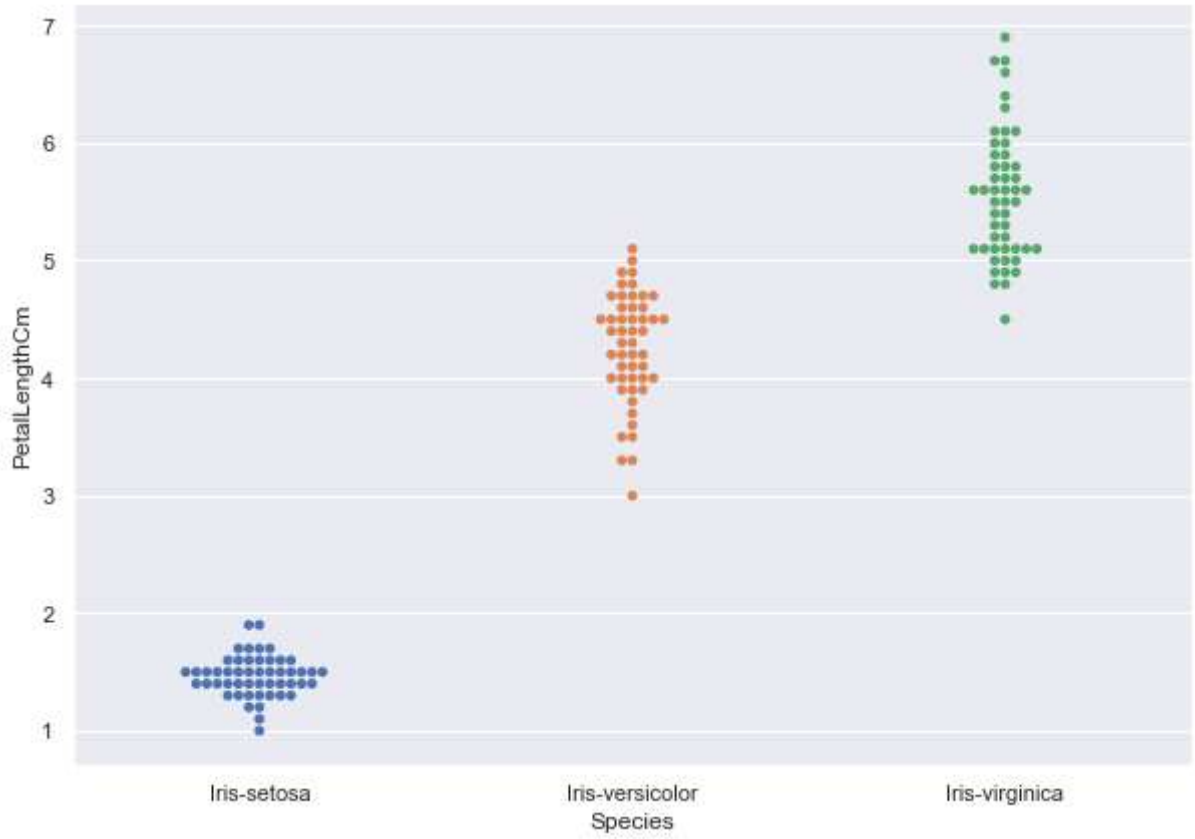
12. Distribution plot: The distribution plot is suitable for comparing range and distribution for groups of numerical data. Data is plotted as value points along an axis. You can choose to display only the value points to see the distribution of values, a bounding box to see the range of values, or a 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 [39]:
```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
```

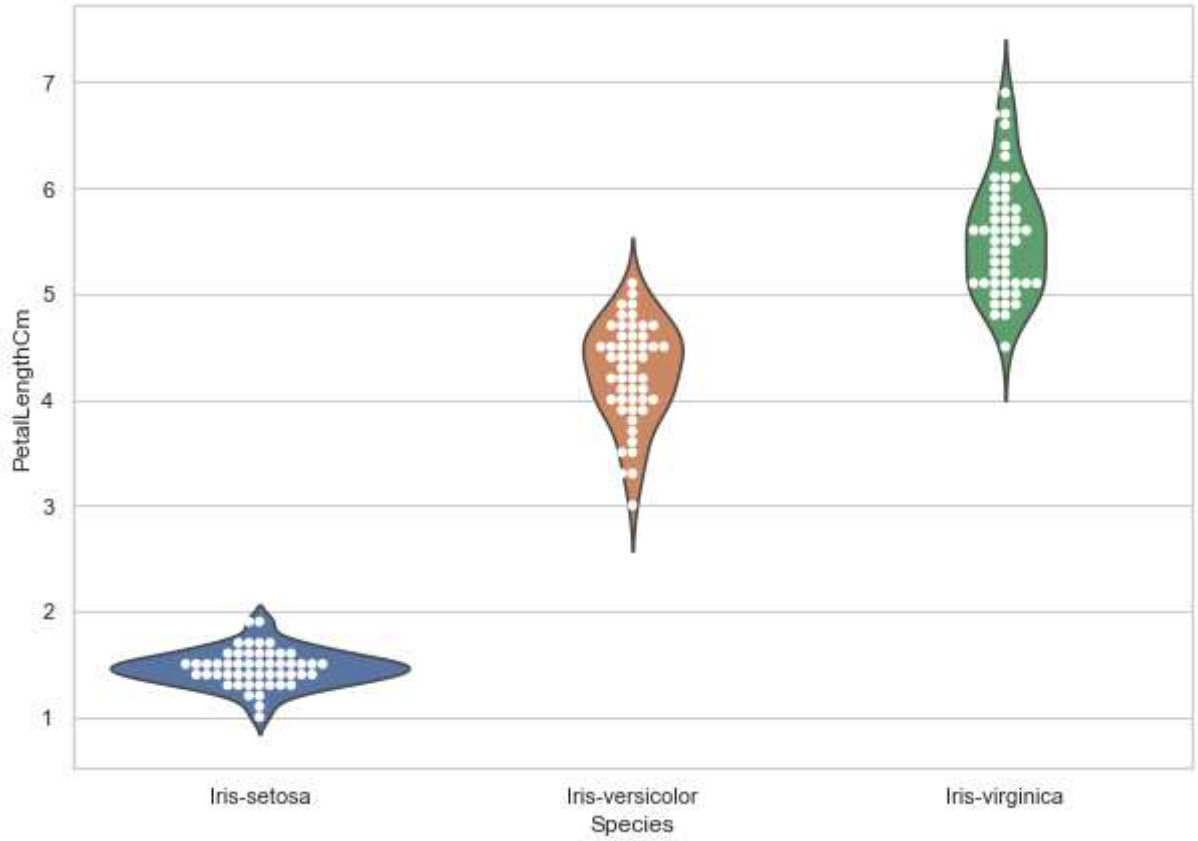


13. Swarm plot It looks a bit like a friendly swarm of bees buzzing about their hive. More importantly, each data point is clearly visible and no data are obscured by overplotting.A beeswarm plot improves upon the random jittering approach to move data points the minimum distance away from one another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

In [40]:
```
sns.set(style="darkgrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
fig = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris)
```

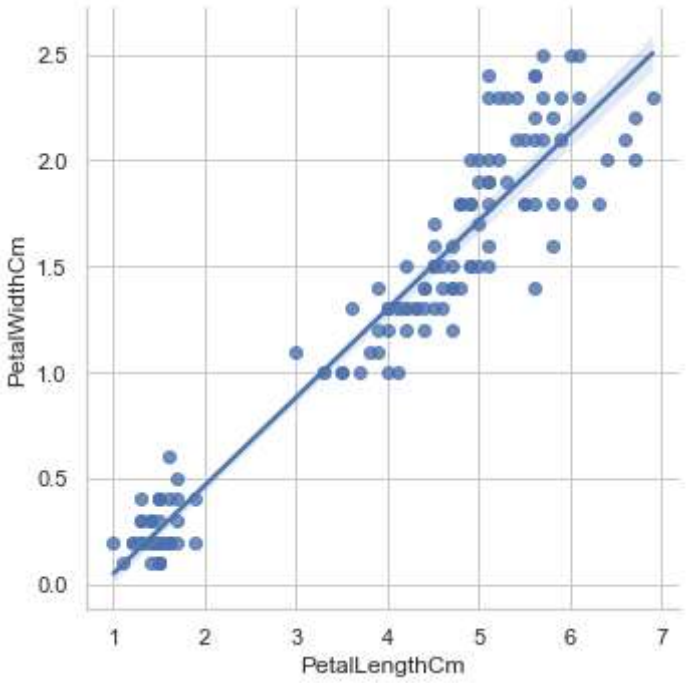


In [41]:
```
sns.set(style="whitegrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
ax = sns.violinplot(x="Species", y="PetalLengthCm", data=iris, inner=None)
ax = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris,color="white", edgec
```



17. LM PLot

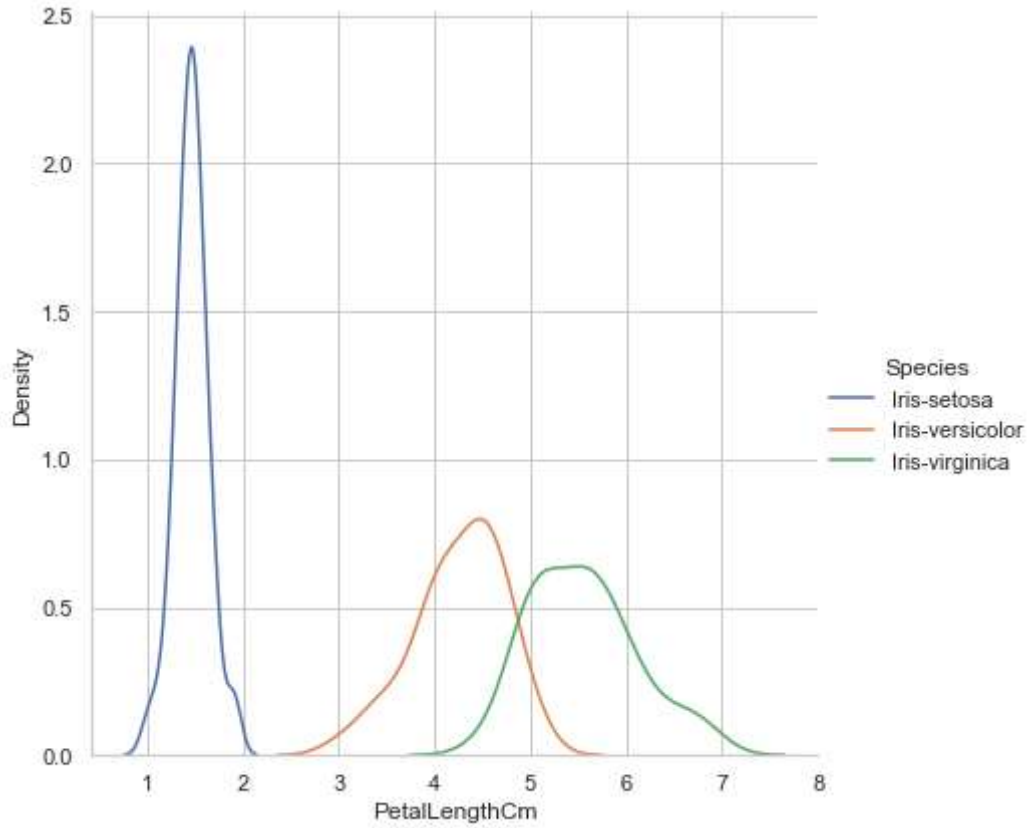
In [42]:
```
fig=sns.lmplot(x="PetalLengthCm", y="PetalWidthCm",data=iris)
```



18. FacetGrid

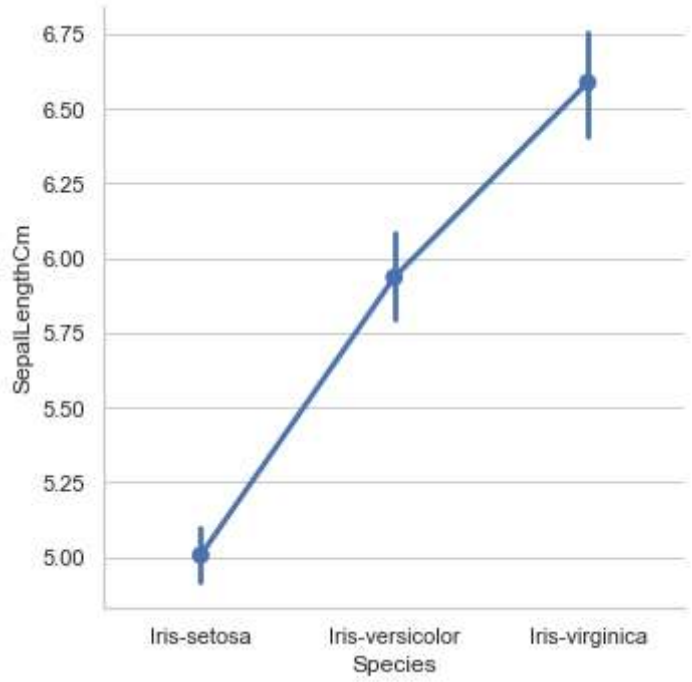
In [43]:
```
sns.FacetGrid(iris, hue="Species", size=6) \
   .map(sns.kdeplot, "PetalLengthCm") \
   .add_legend()
plt.ioff()
```

Out[43]:
```
<matplotlib.pyplot._IoffContext at 0x1f1203913a0>
```



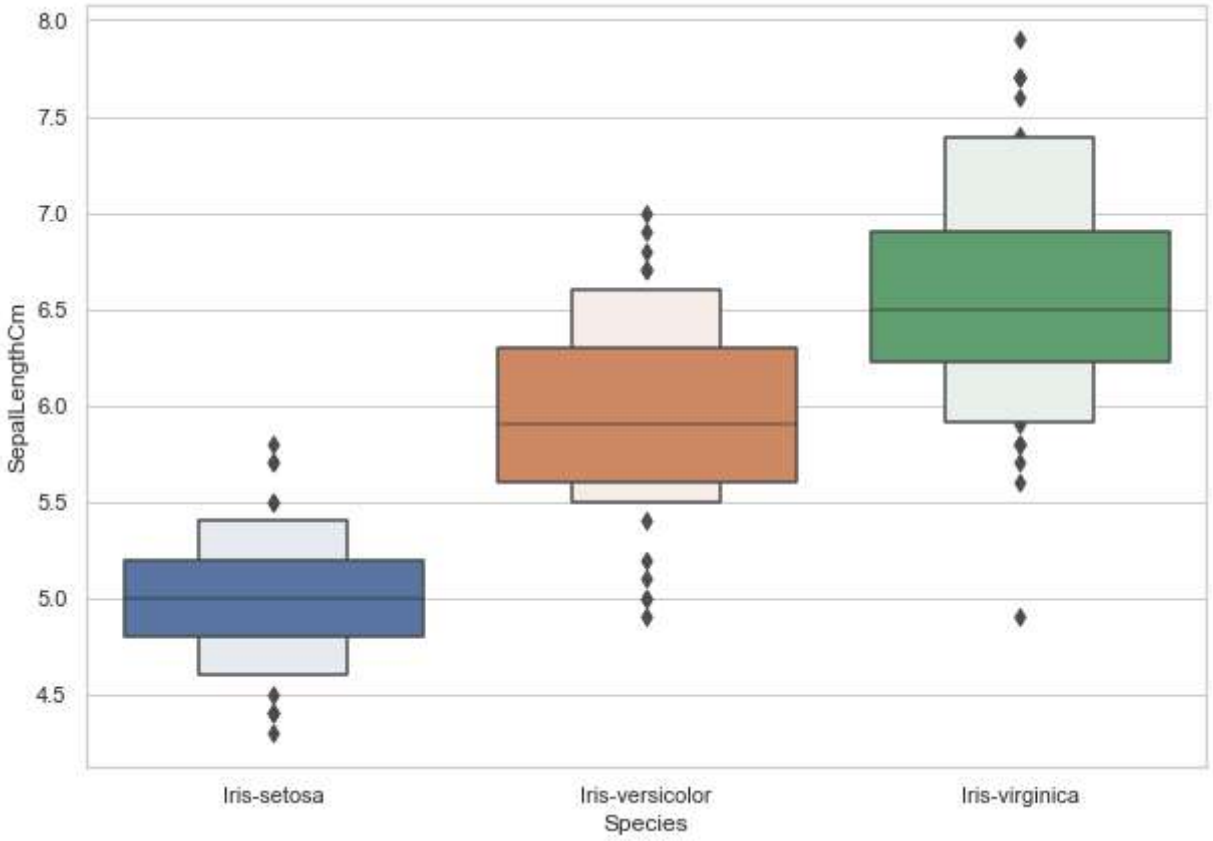
** 22. Factor Plot **

In [44]:
```
#f,ax=plt.subplots(1,2,figsize=(18,8))
sns.factorplot('Species','SepalLengthCm',data=iris)
plt.ioff()
plt.show()
#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])
```



** 23. Boxen Plot**

In [45]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxenplot(x='Species',y='SepalLengthCm',data=iris)
```

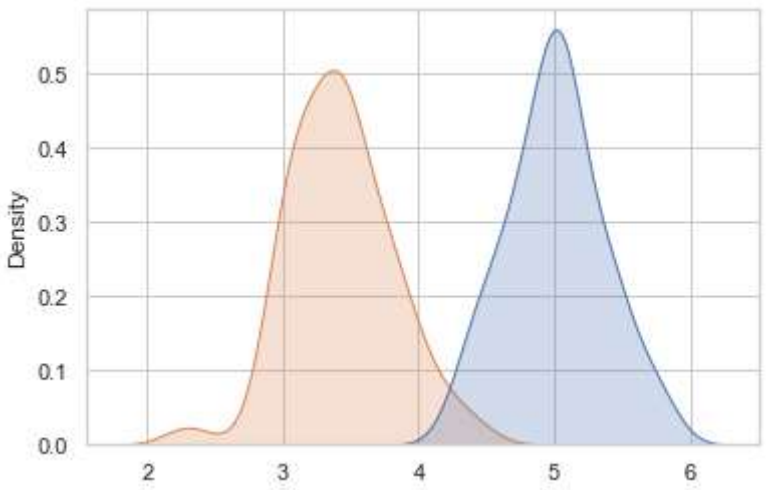


*28.KDE Plot *

In [46]:
```
# Create a kde plot of sepal_length versus sepal width for setosa species of flow
sub=iris[iris['Species']=='Iris-setosa']
sns.kdeplot(data=sub[['SepalLengthCm','SepalWidthCm']],cmap="plasma", shade=True,
plt.title('Iris-setosa')
plt.xlabel('Sepal Length Cm')
plt.ylabel('Sepal Width Cm')
```

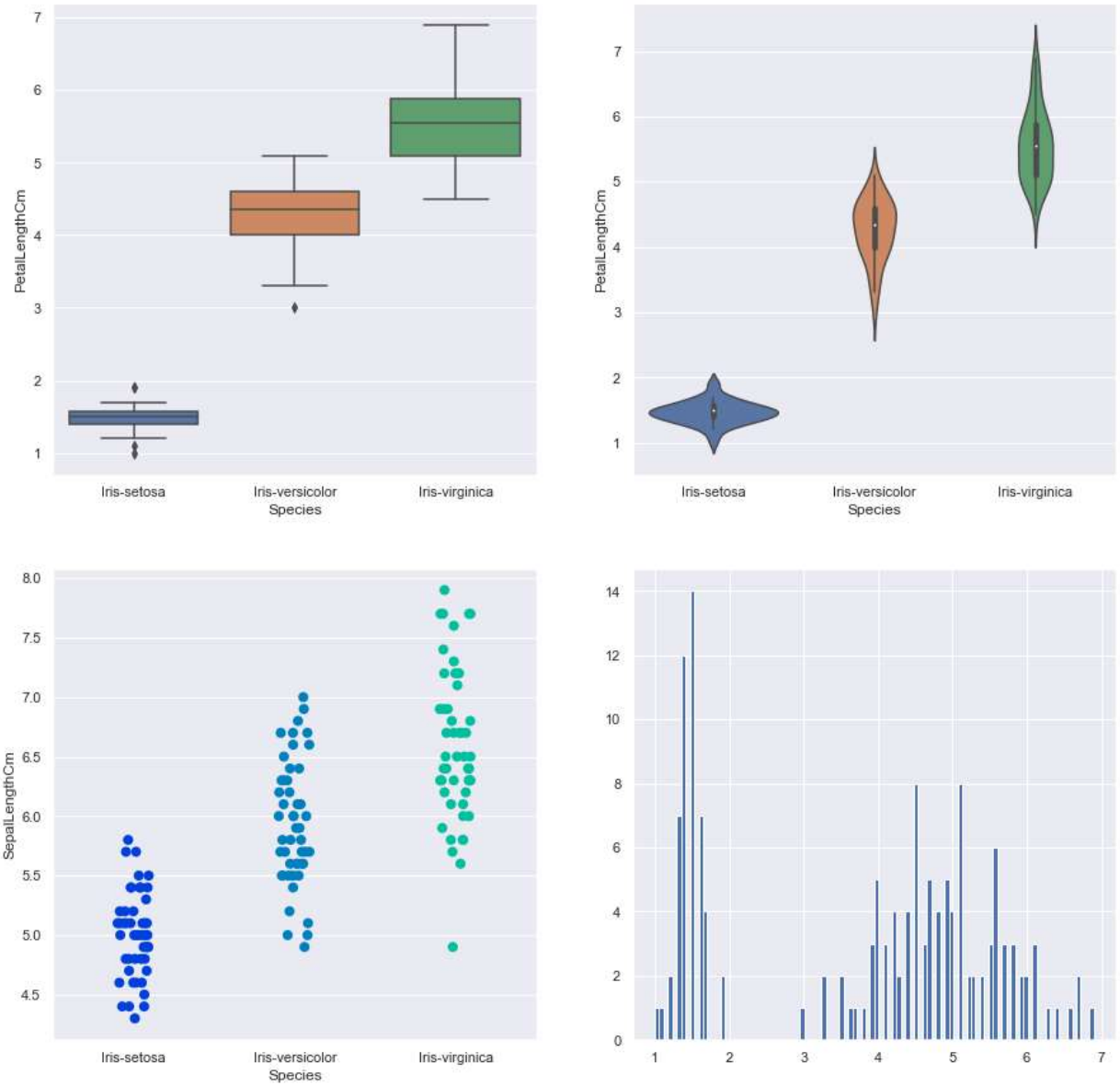
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [46], in <cell line: 3>()
      1 # Create a kde plot of sepal_length versus sepal width for setosa speci
es of flower.
      2 sub=iris[iris['Species']=='Iris-setosa']
----> 3 sns.kdeplot(data=sub[['SepalLengthCm','SepalWidthCm']],cmap="plasma", s
hade=True, shade_lowest=False)
      4 plt.title('Iris-setosa')
      5 plt.xlabel('Sepal Length Cm')

File ~\anaconda3\lib\site-packages\seaborn\_decorators.py:46, in _deprecate_pos
itional_args.<locals>.inner_f(*args, **kwargs)
     36     warnings.warn(
     37         "Pass the following variable{} as {}keyword arg{}: {}. "
     38         "From version 0.12, the only valid positional argument "
   (...)
     43         FutureWarning
     44     )
     45 kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46 return f(**kwargs)

File ~\anaconda3\lib\site-packages\seaborn\distributions.py:1770, in kdeplot(x,
y, shade, vertical, kernel, bw, gridsize, cut, clip, legend, cumulative, shade_
lowest, cbar, cbar_ax, cbar_kws, ax, weights, hue, palette, hue_order, hue_nor
m, multiple, common_norm, common_grid, levels, thresh, bw_method, bw_adjust, lo
g_scale, color, fill, data, data2, warn_singular, **kwargs)
   1767     if color is not None:
   1768         plot_kws["color"] = color
-> 1770     p.plot_univariate_density(
   1771         multiple=multiple,
   1772         common_norm=common_norm,
   1773         common_grid=common_grid,
   1774         fill=fill,
   1775         legend=legend,
   1776         warn_singular=warn_singular,
   1777         estimate_kws=estimate_kws,
   1778         **plot_kws,
   1779     )
   1781 else:
   1783     p.plot_bivariate_density(
   1784         common_norm=common_norm,
   1785         fill=fill,
   (...)
   1795         **kwargs,
   1796     )

File ~\anaconda3\lib\site-packages\seaborn\distributions.py:1054, in _Distribut
ionPlotter.plot_univariate_density(self, multiple, common_norm, common_grid, wa
```

```
rn_singular, fill, legend, estimate_kws, **plot_kws)
   1051     artist = partial(mpl.lines.Line2D, [], [])
   1053 ax_obj = self.ax if self.ax is not None else self.facets
-> 1054 self._add_legend(
   1055     ax_obj, artist, fill, False, multiple, alpha, plot_kws, {},
   1056 )

File ~\anaconda3\lib\site-packages\seaborn\distributions.py:146, in _Distributi
onPlotter._add_legend(self, ax_obj, artist, fill, element, multiple, alpha, art
ist_kws, legend_kws)
    144 for level in self._hue_map.levels:
    145     color = self._hue_map(level)
--> 146     handles.append(artist(
    147         **self._artist_kws(
    148             artist_kws, fill, element, multiple, color, alpha
    149         )
    150     ))
    151     labels.append(level)
    153 if isinstance(ax_obj, mpl.axes.Axes):

File ~\anaconda3\lib\site-packages\matplotlib\patches.py:113, in Patch.__init__
(self, edgecolor, facecolor, color, linewidth, linestyle, antialiased, hatch, f
ill, capstyle, joinstyle, **kwargs)
    110 self.set_joinstyle(joinstyle)
    112 if len(kwargs):
--> 113     self.update(kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\artist.py:1064, in Artist.update
(self, props)
   1062             func = getattr(self, f"set_{k}", None)
   1063             if not callable(func):
-> 1064                 raise AttributeError(f"{type(self).__name__!r} object "
   1065                                      f"has no property {k!r}")
   1066             ret.append(func(v))
   1067 if ret:

AttributeError: 'Patch' object has no property 'cmap'
```



30.Dashboard

In [47]:
```
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='g
#axes[1,1].hist(iris.hist,bin=10)
axes[1,1].hist(iris.PetalLengthCm,bins=100)
#k2.set(xlim=(-1,0.8))
plt.show()
```

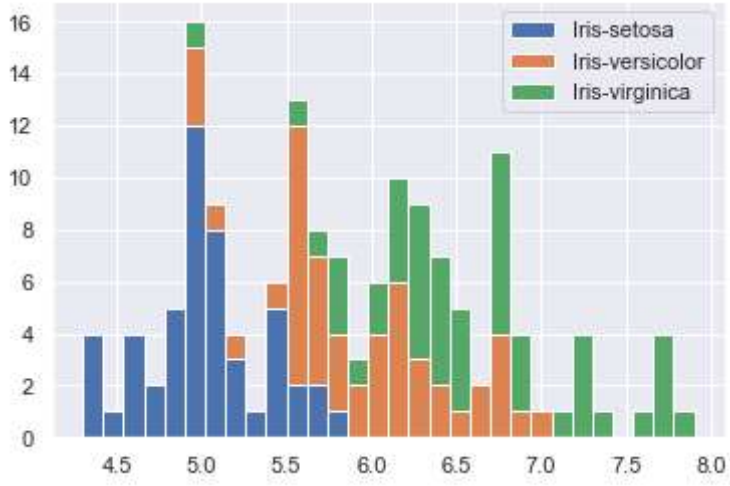


In the dashboard we have shown how to create multiple plots to foam a dashboard using Python.In this plot we have demonstrated how to plot Seaborn and Matplotlib plots on the same Dashboard.

31.Stacked Histogram

In [48]:
```
iris['Species'] = iris['Species'].astype('category')
#iris.head()
```

In [49]:
```
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()
```



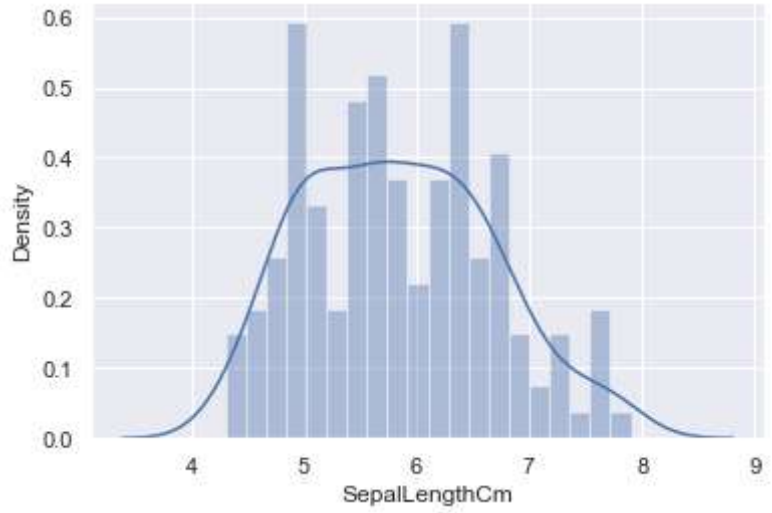
With Stacked Histogram we can see the distribution of Sepal Length of Different Species together.This shows us the range of Sepan Length for the three different Species of Iris Flower.

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

In [50]:
```
#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'],
```

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

In [51]:
```
sns.distplot(iris['SepalLengthCm'],kde=True,bins=20);
```



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