IRIS Dataset Visualization (Seaborn, Matplotlib)

In [2]:
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
iris = pd.read_csv(r'C:\Users\HP\Desktop\DSAIML Course\Class Tasks & Notes\Panda
iris
```

Out[3]:

	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 [4]:
```
iris.head()
```

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

In [5]:
```
iris.tail()
```

Out[5]:

	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 [6]:
```
iris.isnull().any().any()
```

Out[6]:
```
False
```

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

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

In [8]:
```
iris.columns = ['Id', 'SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth','
iris.head()
```

Out[8]:

	Id	SepalLength	SepalWidth	PetalLength	PetalWidth	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 [9]:
```
iris.shape
```

Out[9]:
```
(150, 6)
```

In [10]:
```
iris.drop('Id', axis=1,inplace=True)
iris.head()
```

Out[10]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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 [11]:
```
iris.info()
```

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

In [12]:
```
iris.dtypes
```

Out[12]:
```
SepalLength    float64
SepalWidth     float64
PetalLength    float64
PetalWidth     float64
Species         object
dtype: object
```

In [13]:
```
iris.Species = iris.Species.astype('category')
```

In [14]:
```
iris.Species = iris.Species.astype('object')
```

In [15]:
```
iris.head()
```

Out[15]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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 [16]:
```
iris.dtypes
```

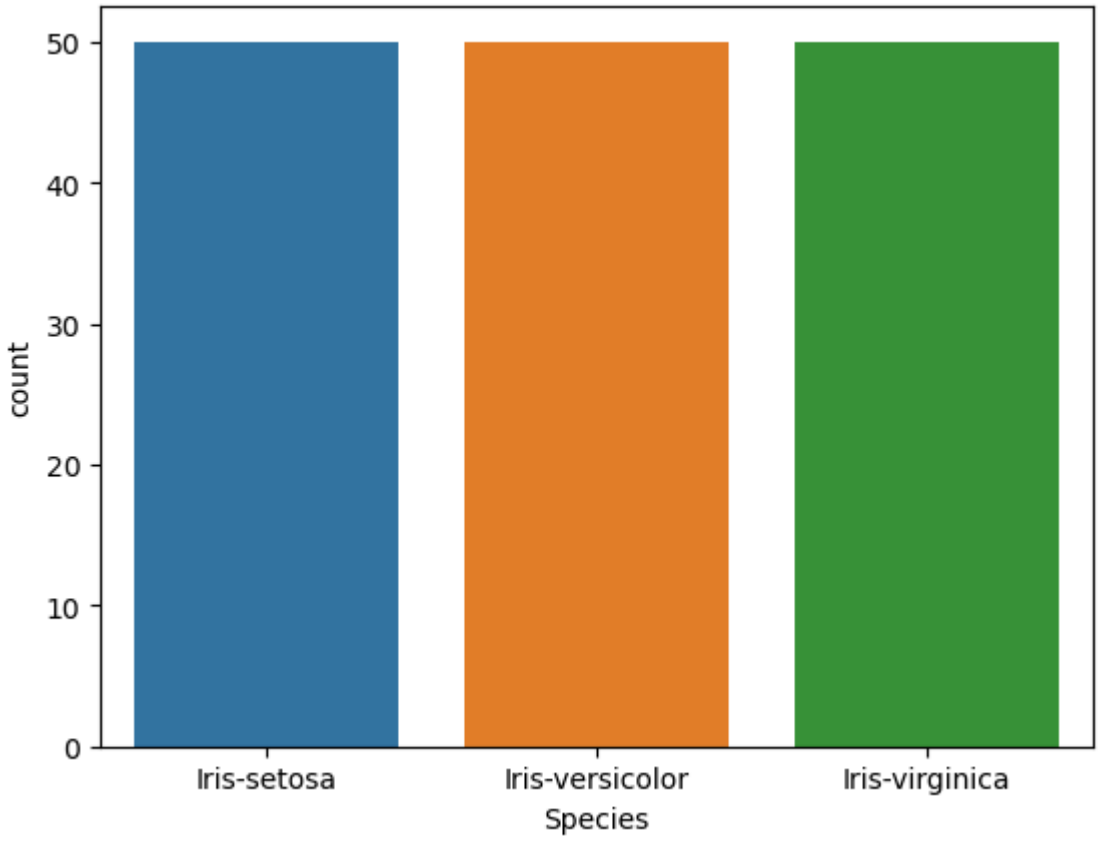
Out[16]:
```
SepalLength    float64
SepalWidth     float64
PetalLength    float64
PetalWidth     float64
Species         object
dtype: object
```

In [17]:
```
iris['Species'].value_counts()
```

Out[17]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

1. Bar plot = A bar plot is a chart that uses rectangular bars to represent data, where the length or height of each bar corresponds to the value of a variable, often used to compare categories or show counts.

In [19]:
```
sns.countplot(x='Species',data=iris, palette="tab10")
plt.show()
```



2. 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.

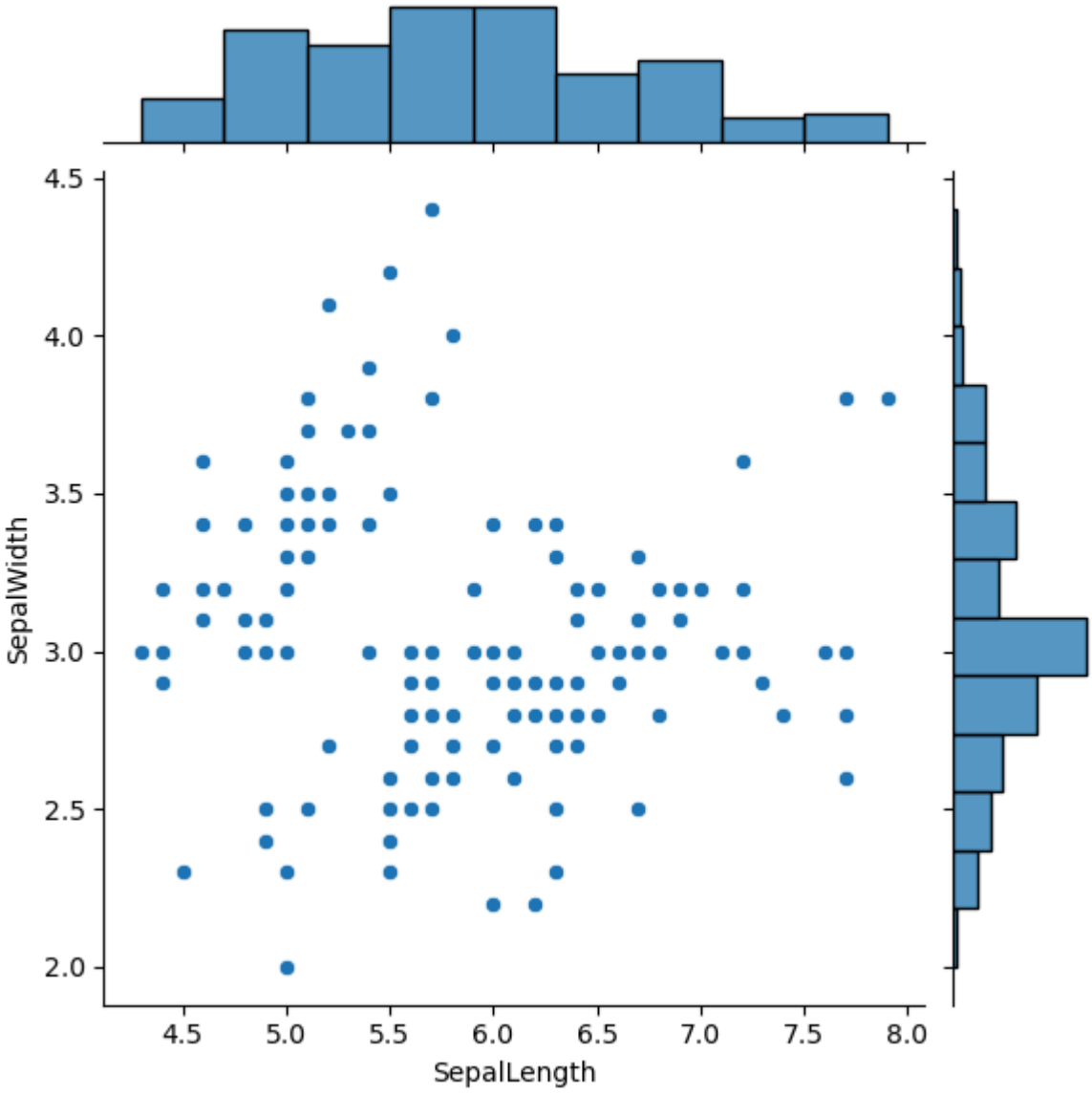
In [21]:
```
iris.head()
```

Out[21]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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

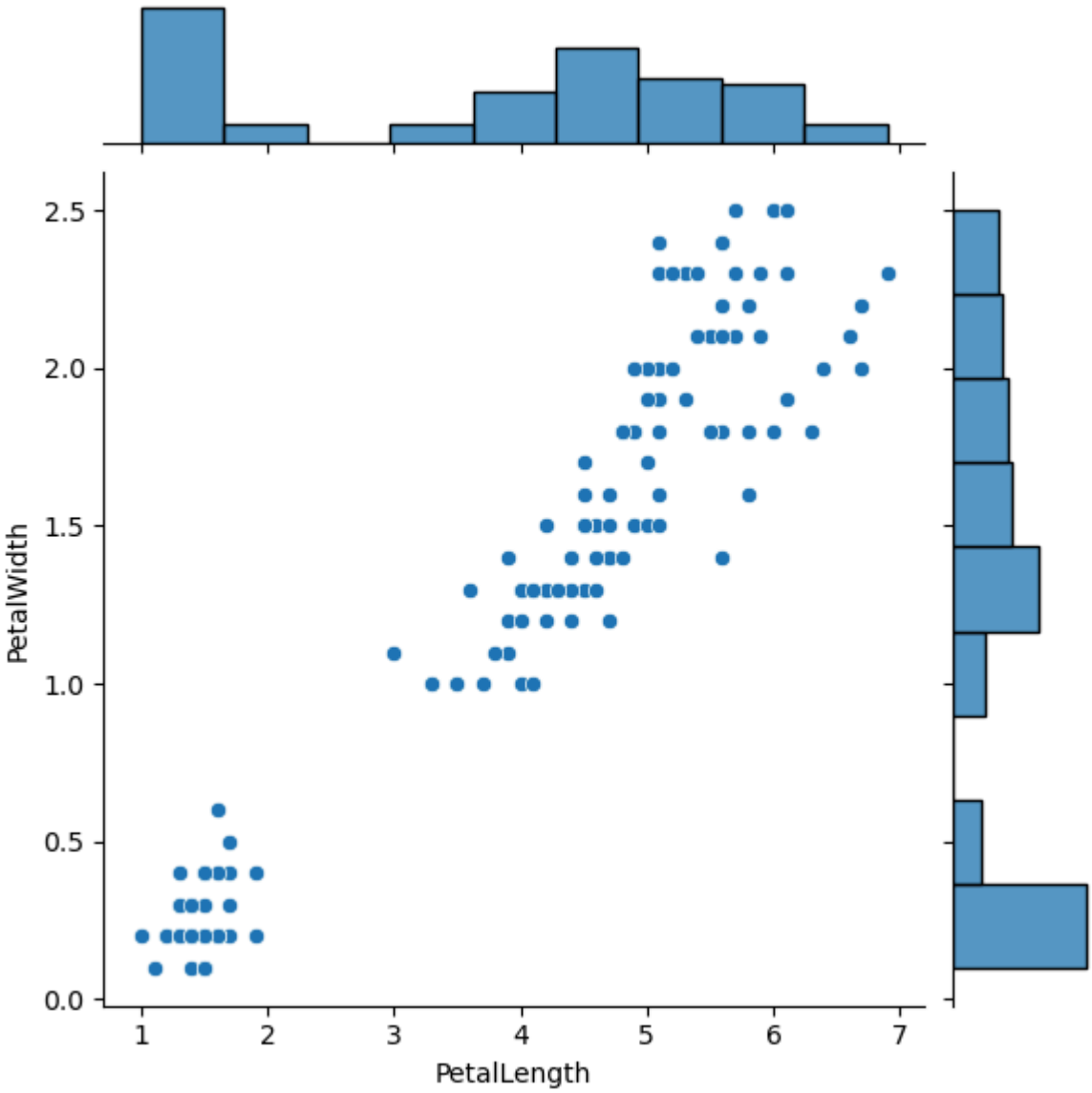
In [22]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris)
plt.show
```

Out[22]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [23]:
```
vis1 = sns.jointplot(x='PetalLength', y='PetalWidth', data=iris)
plt.show
```

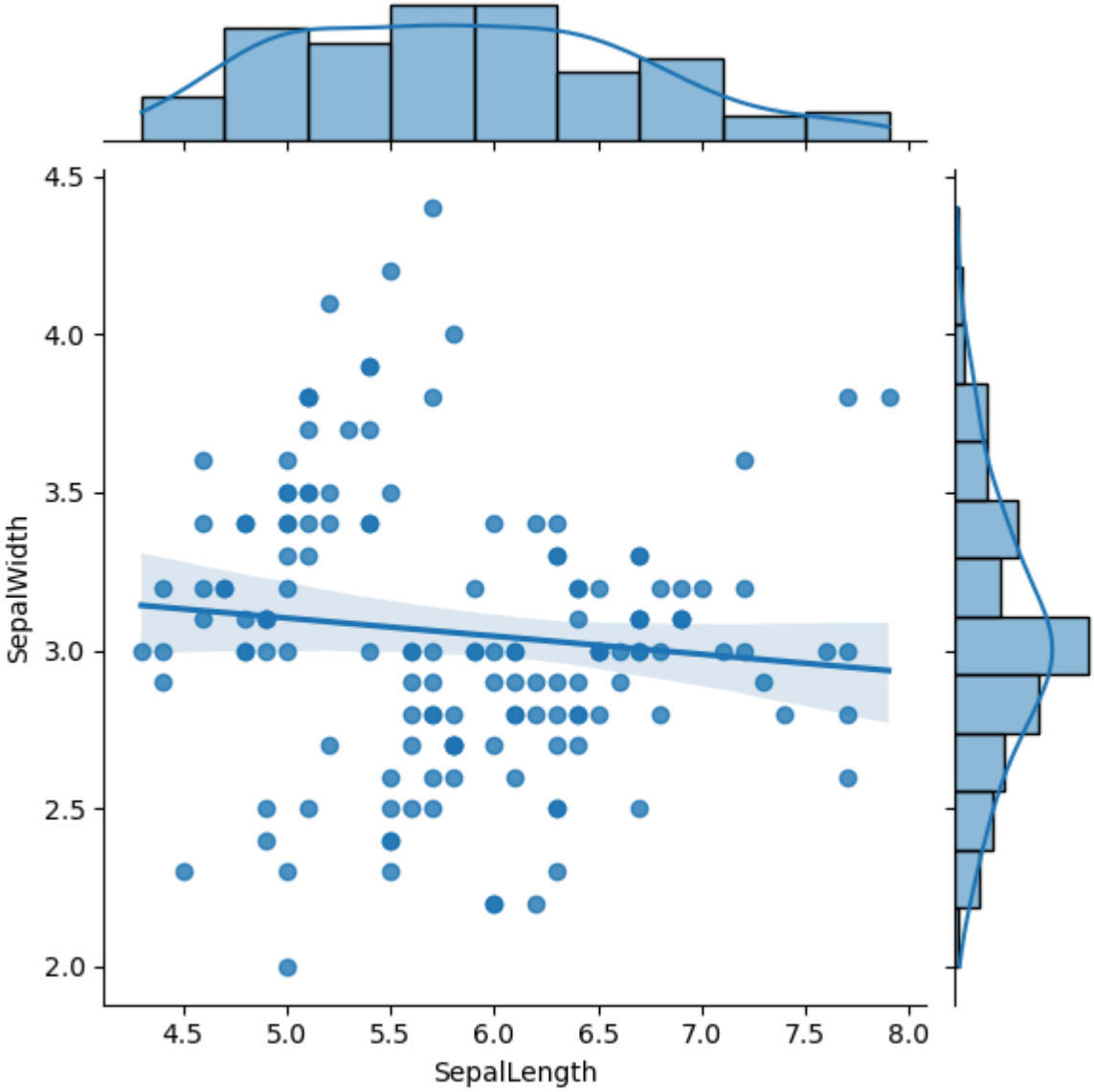
Out[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [24]:
```
# for "kind" we can use { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }
```

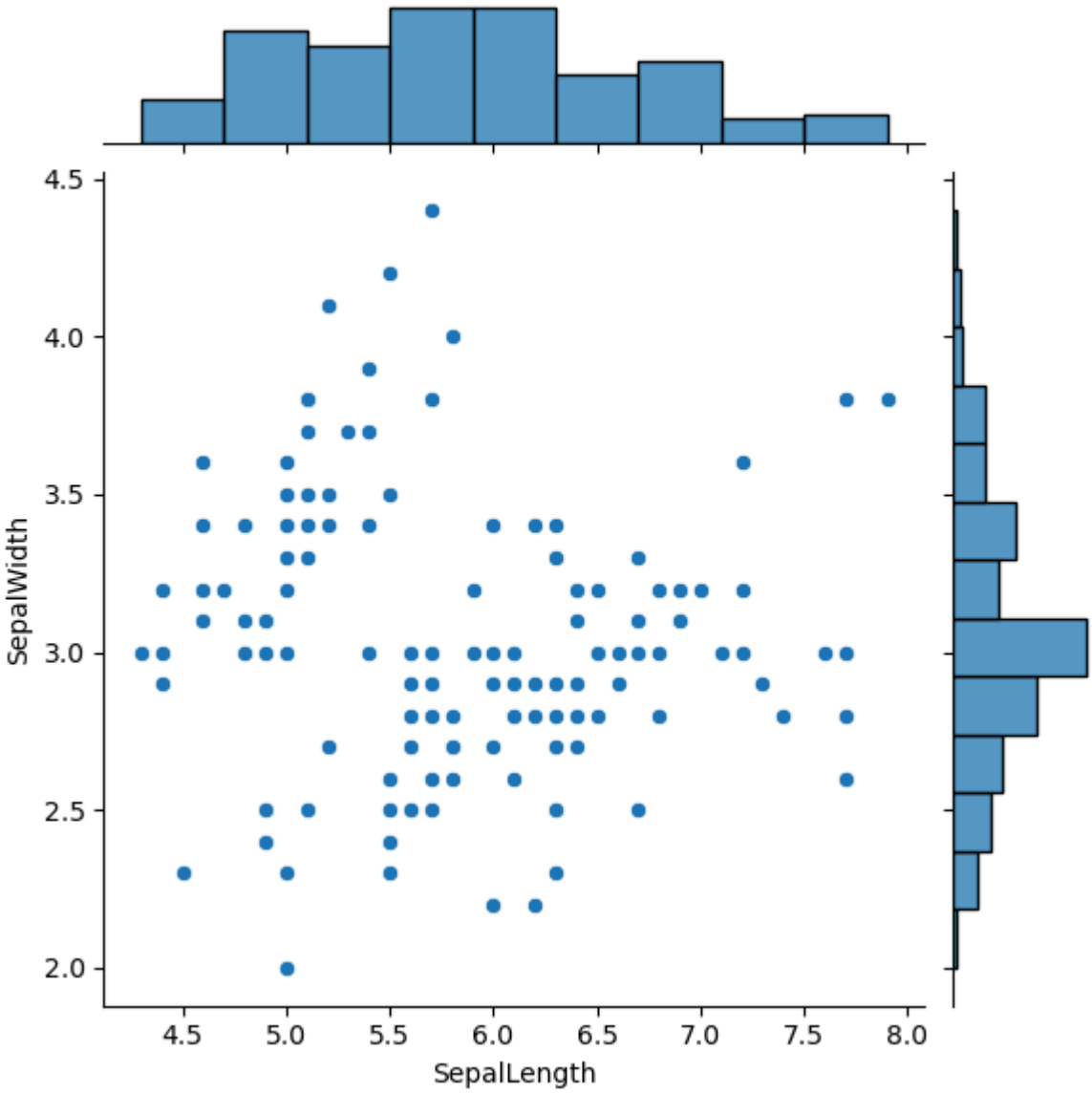
In [25]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='reg')
plt.show
```

Out[25]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



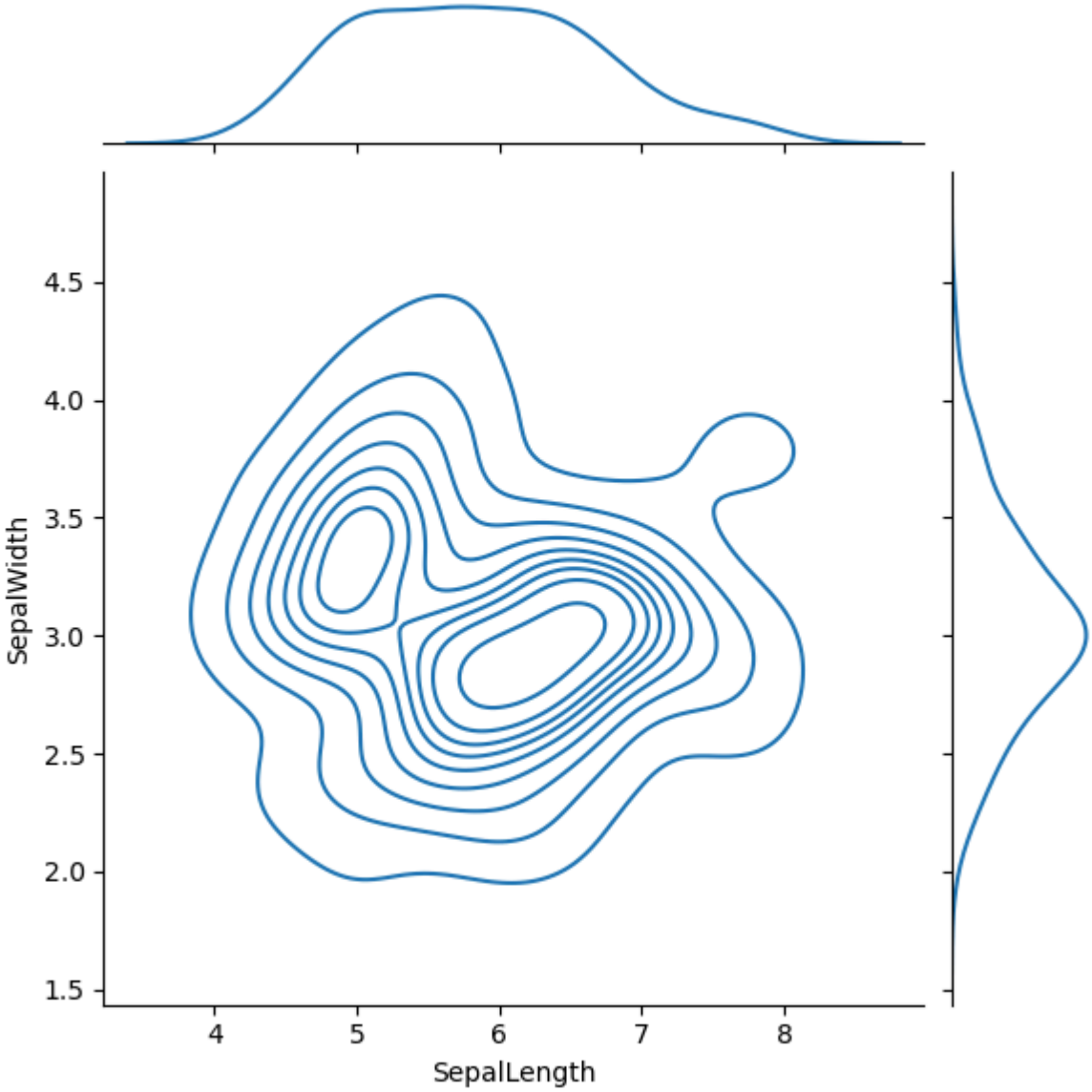
In [26]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='scatter')
plt.show
```

Out[26]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



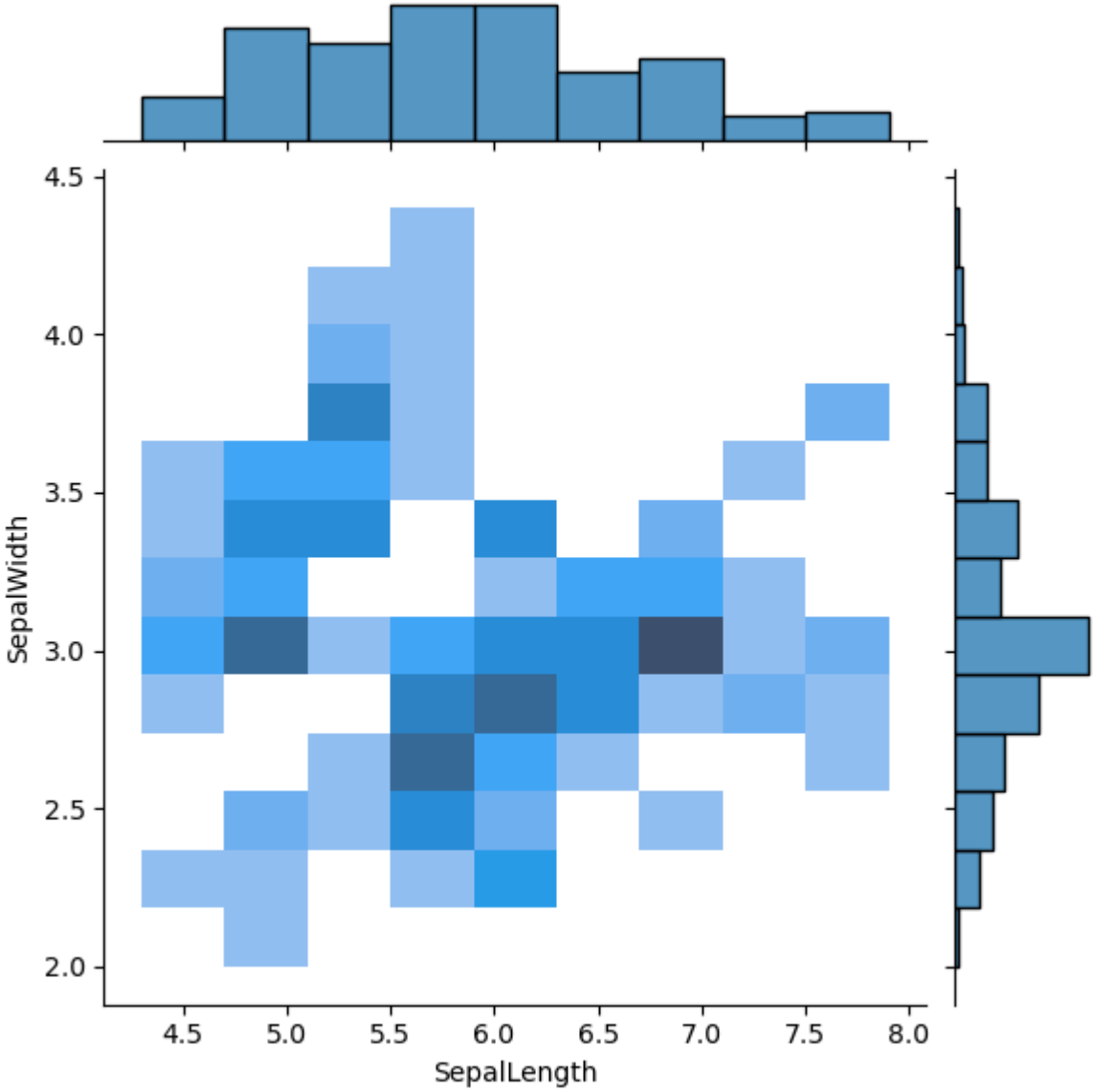
In [27]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='kde')
plt.show
```

Out[27]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



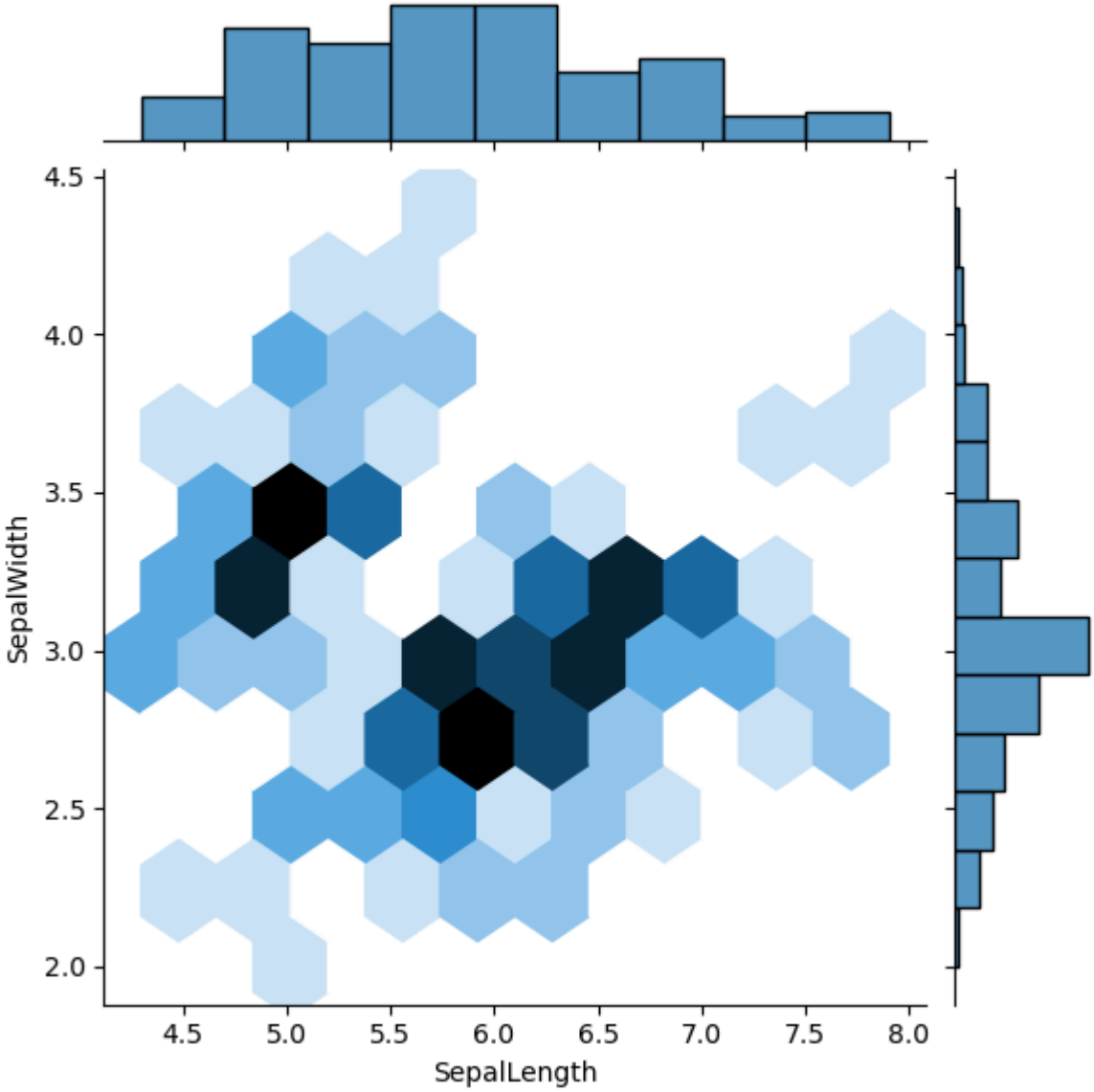
In [28]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='hist')
plt.show
```

Out[28]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



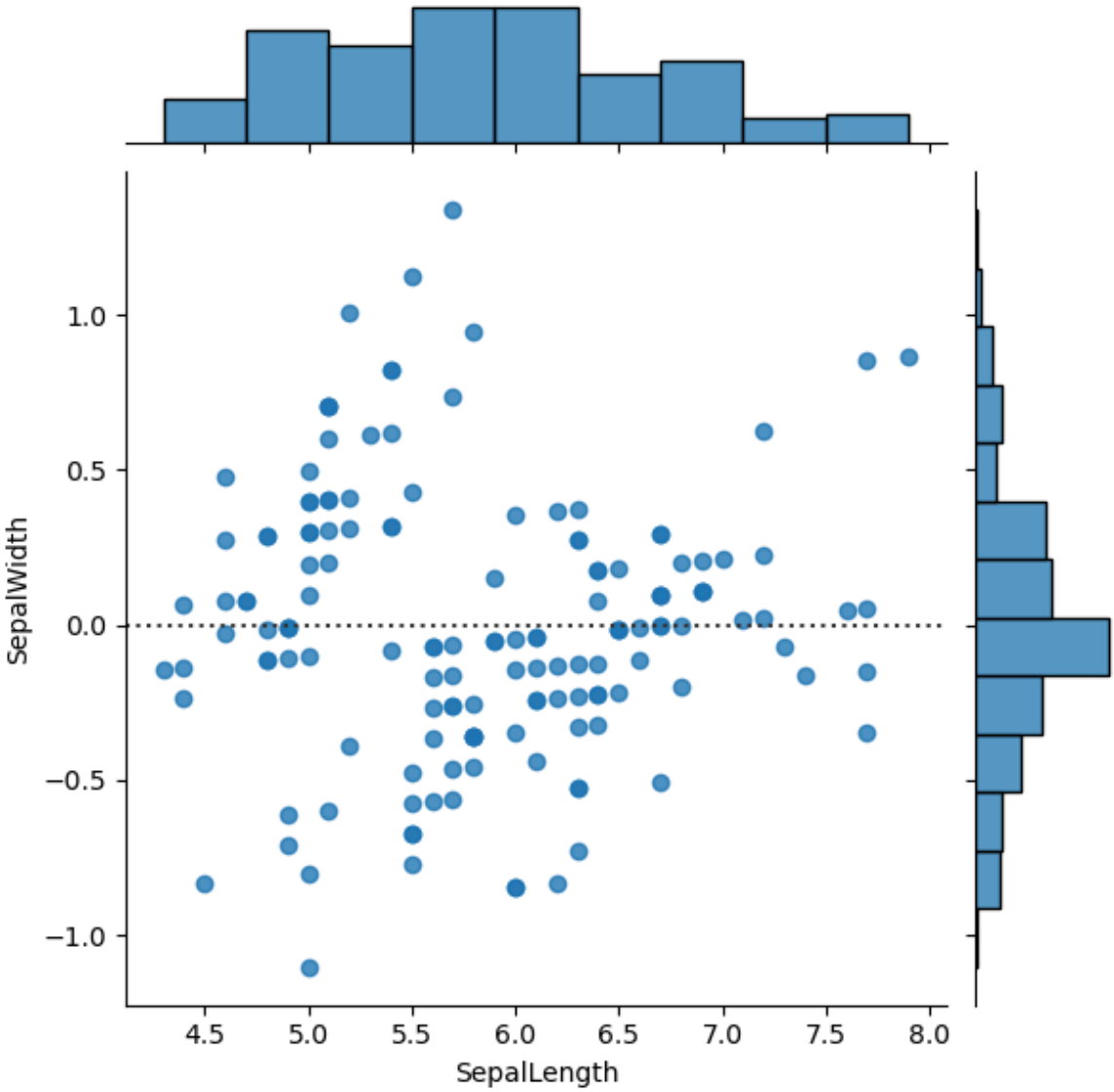
In [29]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='hex')
plt.show
```

Out[29]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [30]:
```
vis1 = sns.jointplot(x='SepalLength', y='SepalWidth', data=iris, kind='resid')
plt.show
```

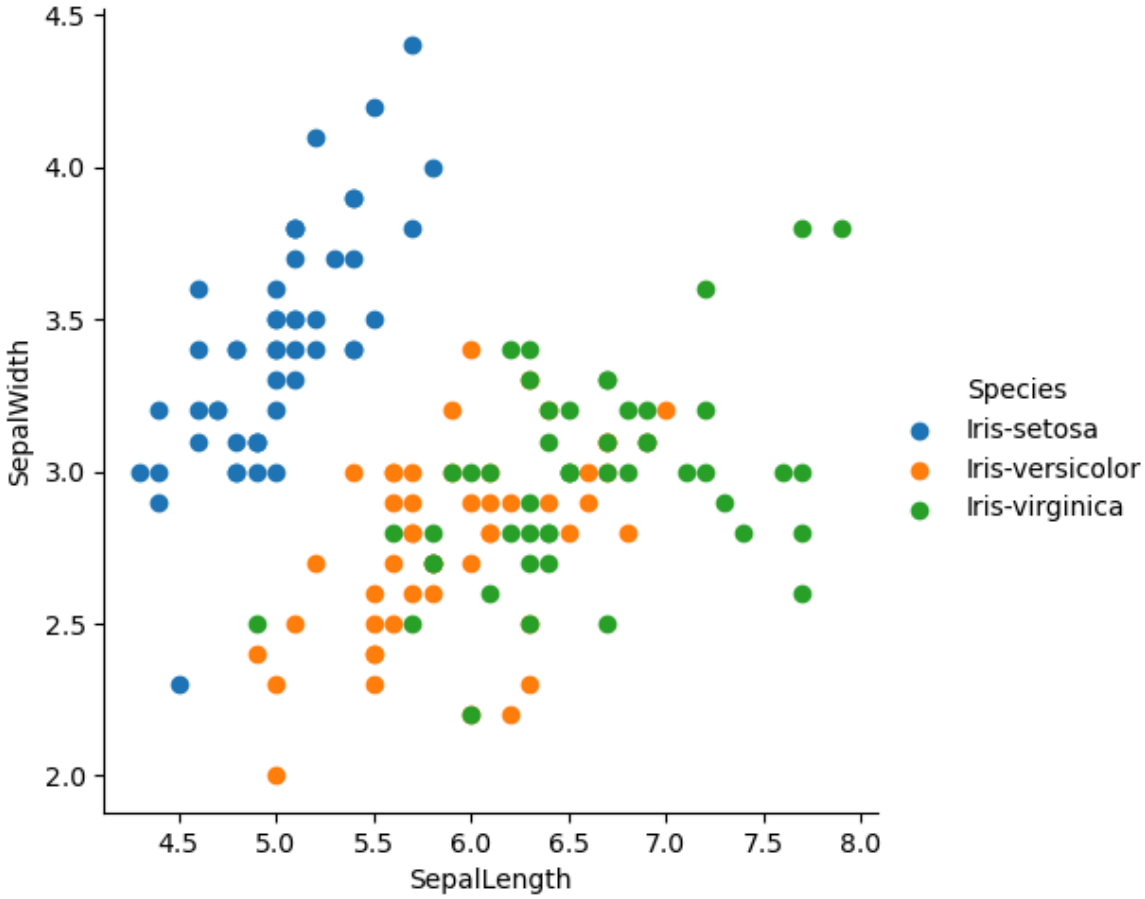
Out[30]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



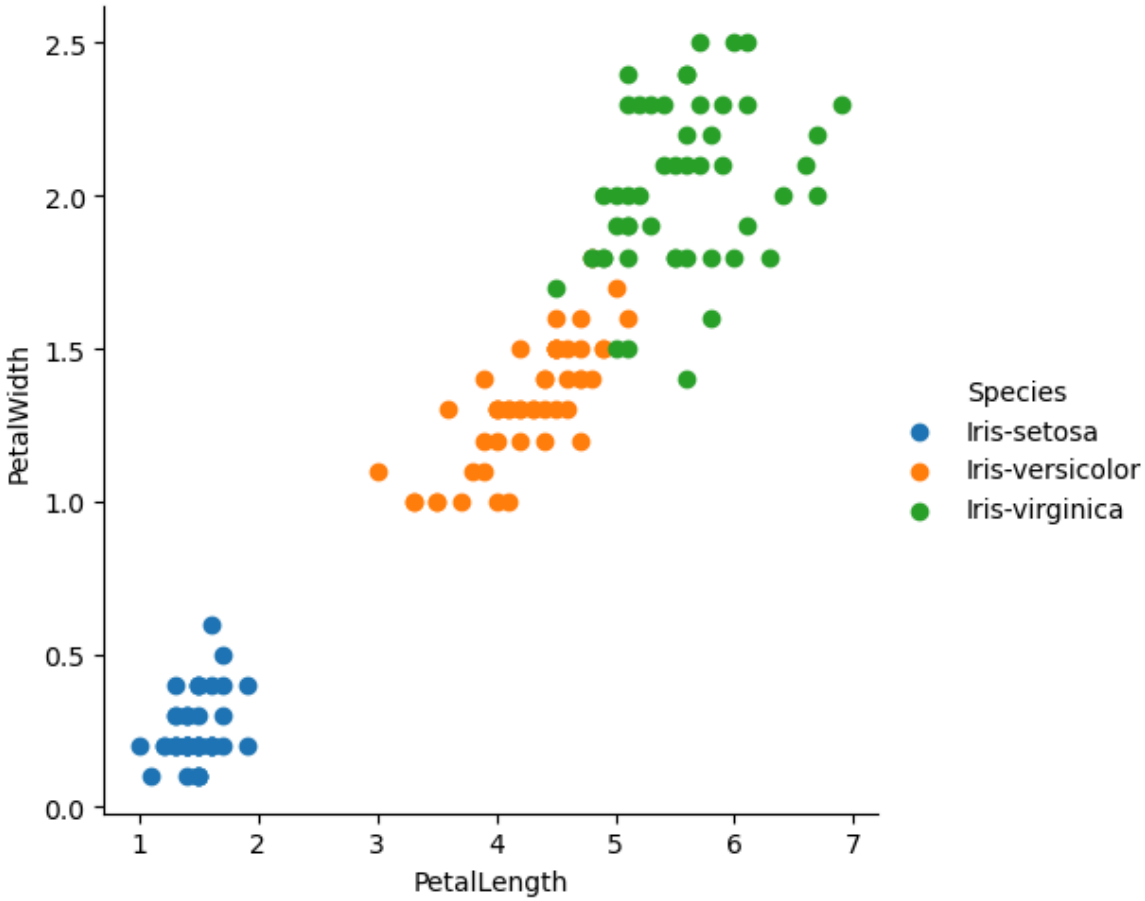
3. FacetGrid plot = A FacetGrid plot in Seaborn is a way to create a grid of subplots, where each subplot shows a subset of the data based on one or more categorical variables, making it easy to compare patterns across different groups.

In [32]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [33]:
```
sns.FacetGrid(iris, hue='Species', height=5)\
.map(plt.scatter,'SepalLength','SepalWidth')\
.add_legend()
plt.show()
```



In [34]:
```
sns.FacetGrid(iris, hue='Species', height=5)\
.map(plt.scatter,'PetalLength','PetalWidth')\
.add_legend()
plt.show()
```



4. 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.

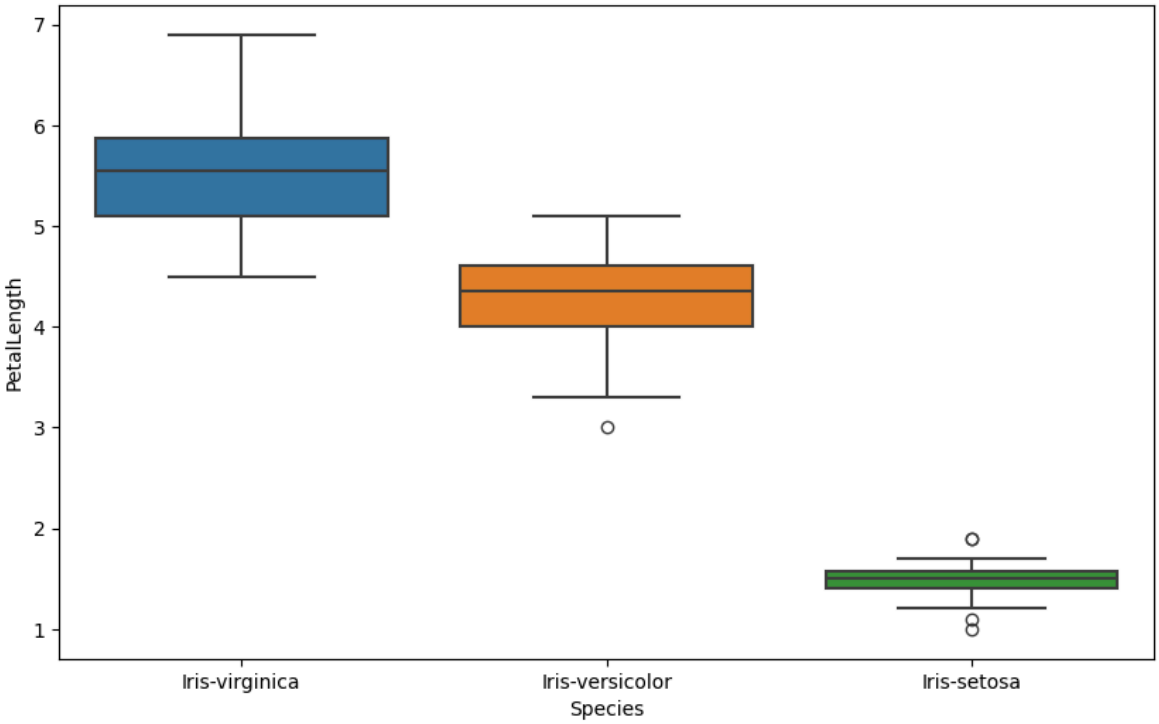
In [36]:
```
iris.head()
```

Out[36]:

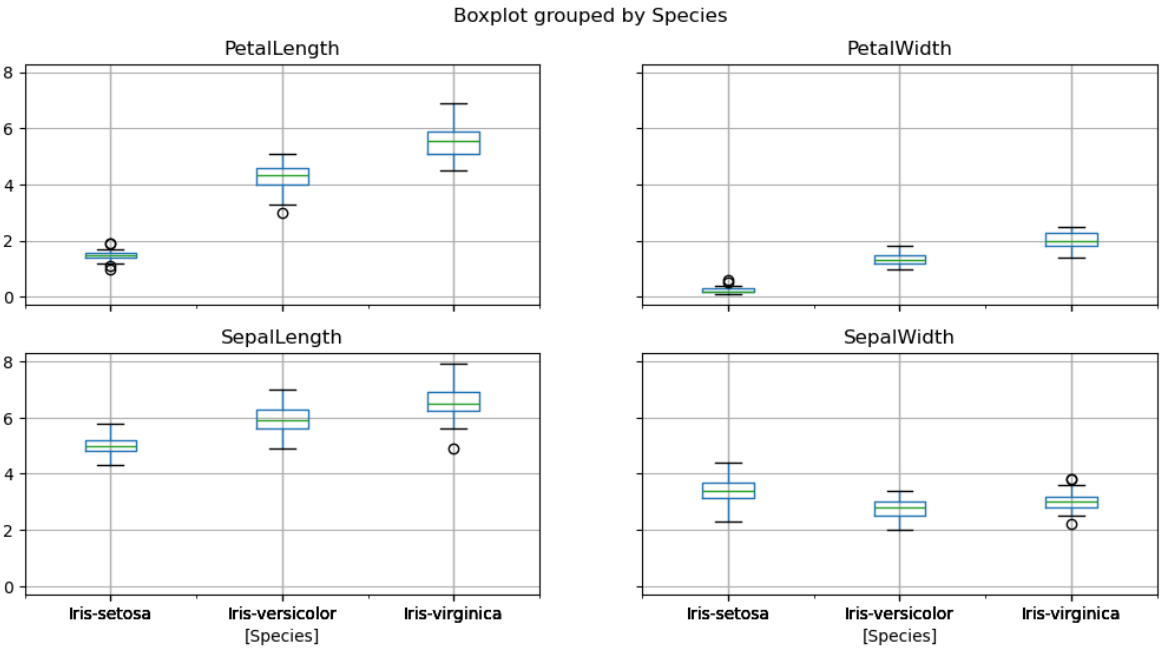
	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [37]:
```
fig=plt.gcf() # it gives current figure. if there is no figure, it gives empty p
fig.set_size_inches(10,6)
fig=sns.boxplot(x='Species',y='PetalLength',data=iris,order=['Iris-virginica','I
plt.show()
# Orientation: 'v' for vertical boxes (default for y=numerical)
```

```
# Linewidth = Thickness of the boxplot lines
# dodge = No dodging (relevant only with hue, which isn't used here)
# palette: color scheme,'tab10' is a Matplotlib qualitative palette
```



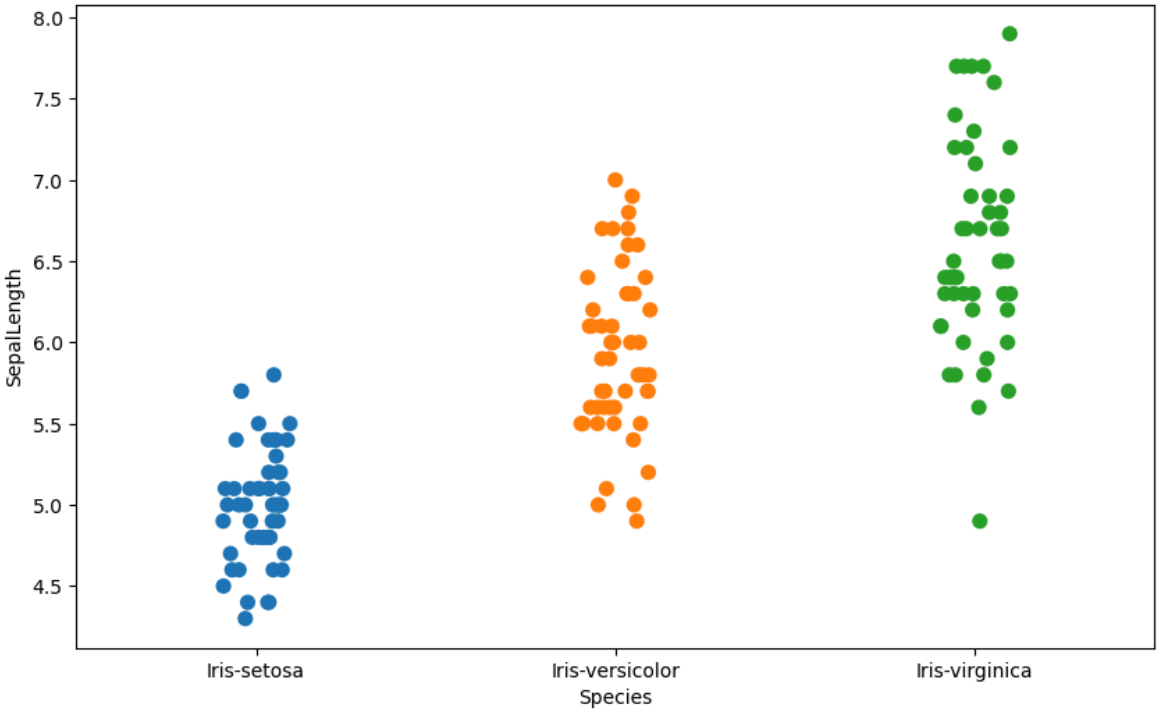
In [38]:
```
iris.boxplot(by='Species', figsize=(12,6))
plt.show()
```



5. Strip Plot = a strip plot offers a simple yet effective way to visualize the spread and distribution of individual data points.

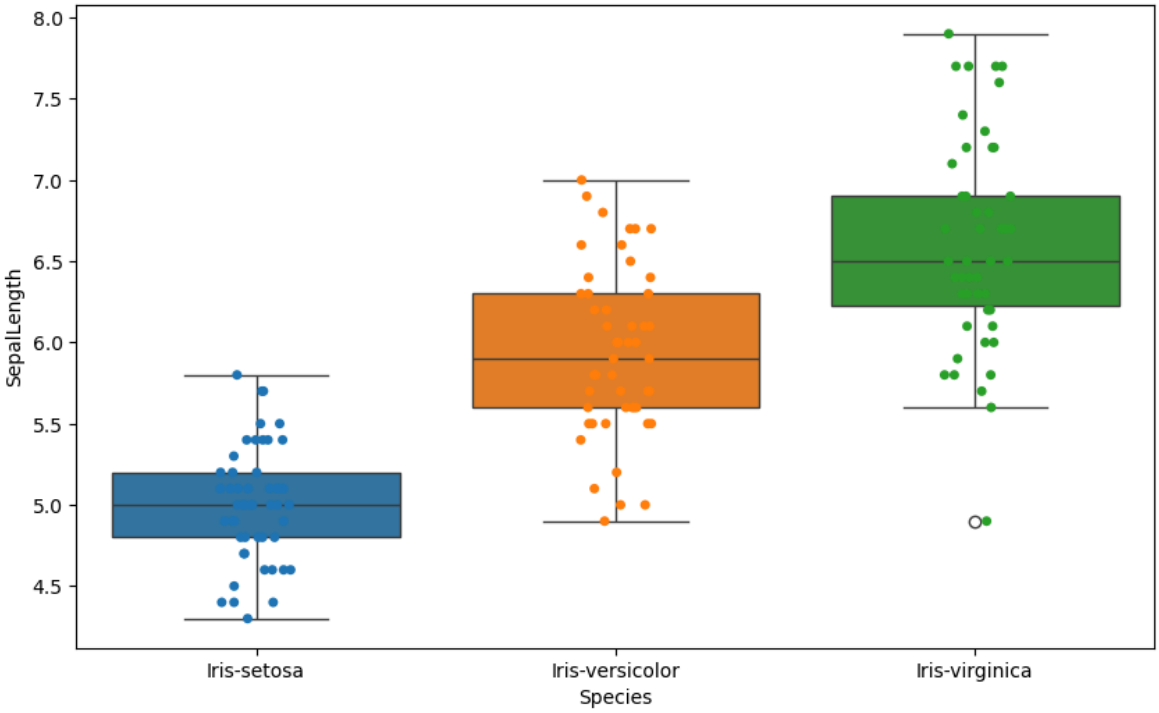
In [40]:
```
fig=plt.gcf()
fig.set_size_inches(10,6)
fig=sns.stripplot(x='Species',y='SepalLength',data=iris,jitter=True,edgecolor='g
```

```
plt.show()
# jitter = it gives only the points which are present in straight line.
```



6. Combination of Box and Strip Plots

In [42]:
```
fig=plt.gcf()
fig.set_size_inches(10,6)
fig=sns.boxplot(x='Species',y='SepalLength',data=iris, palette='tab10')
fig=sns.stripplot(x='Species',y='SepalLength',data=iris,jitter=True,edgecolor='g
plt.show()
```



In [50]:
```
ax= sns.boxplot(x="Species", y="PetalLength", data=iris)
ax= sns.stripplot(x="Species", y="PetalLength", data=iris, jitter=True, edgecolo
```

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

plt.show() # incomplete
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[50], line 4
      1 ax= sns.boxplot(x="Species", y="PetalLength", data=iris)
      2 ax= sns.stripplot(x="Species", y="PetalLength", data=iris, jitter=True, e
dgecolor="gray")
----> 4 boxthree=ax.artists[2]
      5 boxthree.set_facecolor('red')
      6 boxthree.set_edgecolor('black')

File C:\anaconda\Lib\site-packages\matplotlib\axes\_base.py:1453, in _AxesBase.Ar
tistList.__getitem__(self, key)
   1452 def __getitem__(self, key):
-> 1453     return [artist
   1454             for artist in self._axes._children
   1455             if self._type_check(artist)][key]

IndexError: list index out of range
```

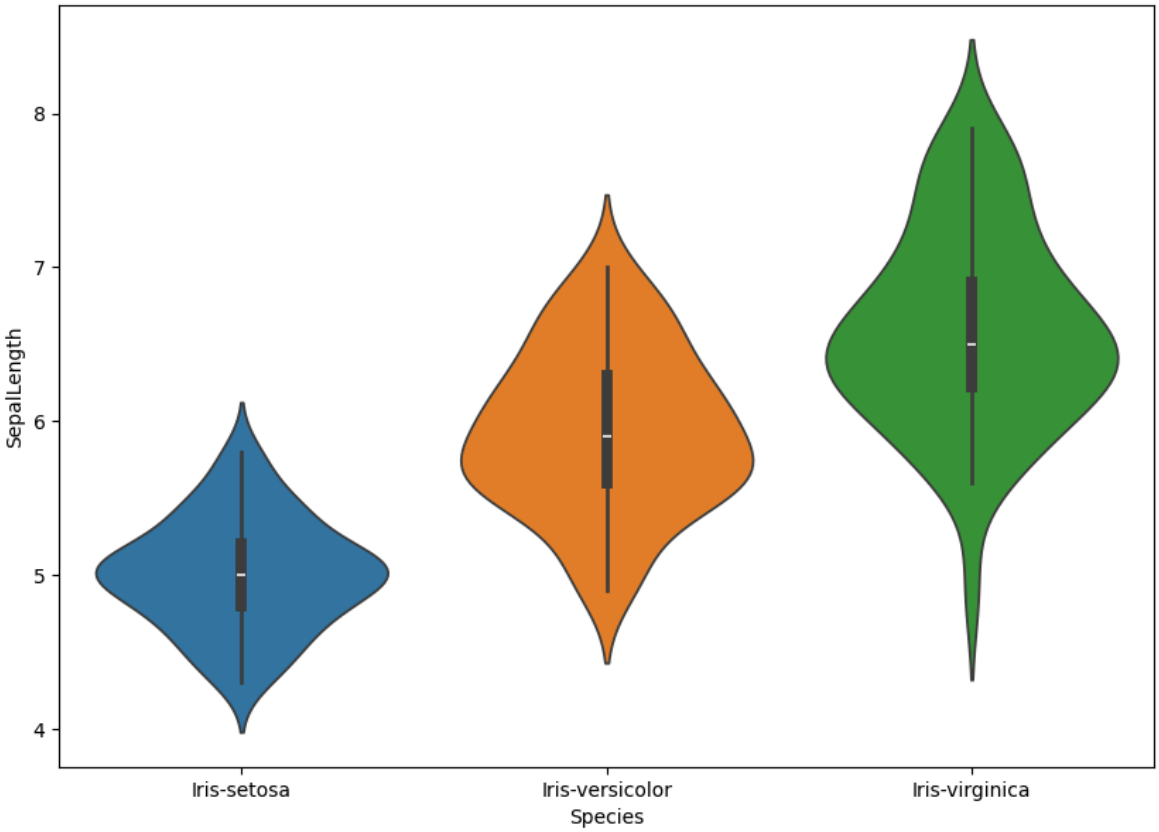
7. Violin Plot = It is used to visualize the distribution of data and its probability distribution.This chart is a combination of a Box Plot and a Density Plot that is rotated and placed on each side, to show the distribution shape of the data. The thick black bar in the centre represents the interquartile 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 [44]:
```
iris.head()
```

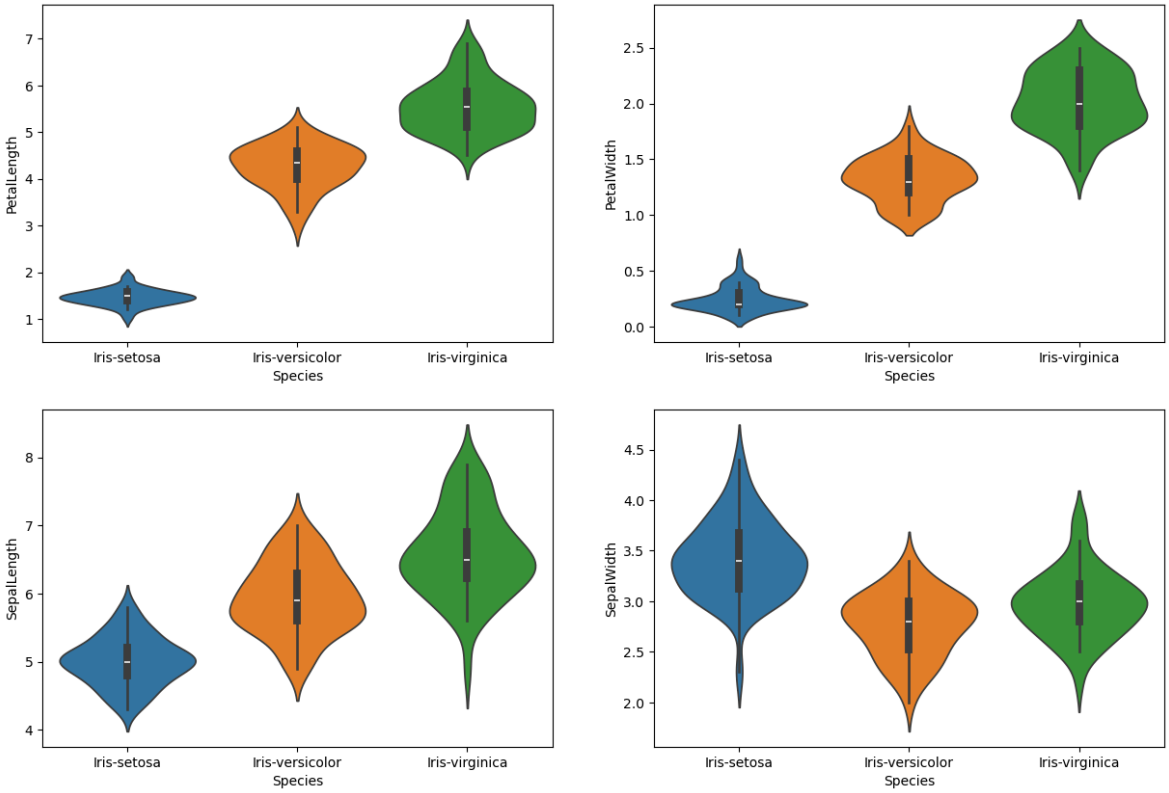
Out[44]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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 [56]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.violinplot(x='Species',y='SepalLength',data=iris, palette = 'tab10')
plt.show()
```

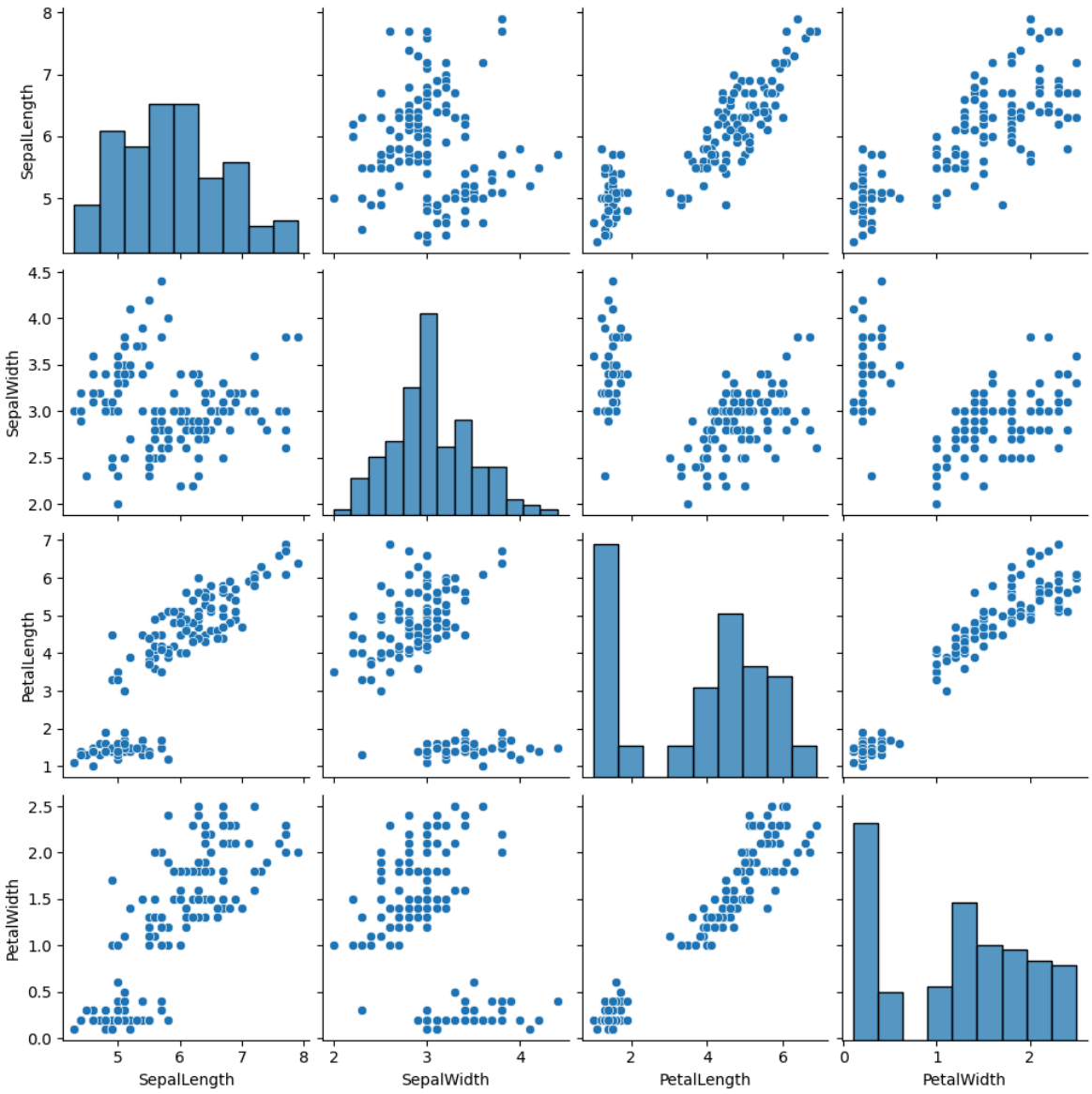


In [62]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLength',data=iris, palette='tab10')
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidth',data=iris, palette='tab10')
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLength',data=iris, palette='tab10')
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidth',data=iris, palette='tab10')
plt.show()
```

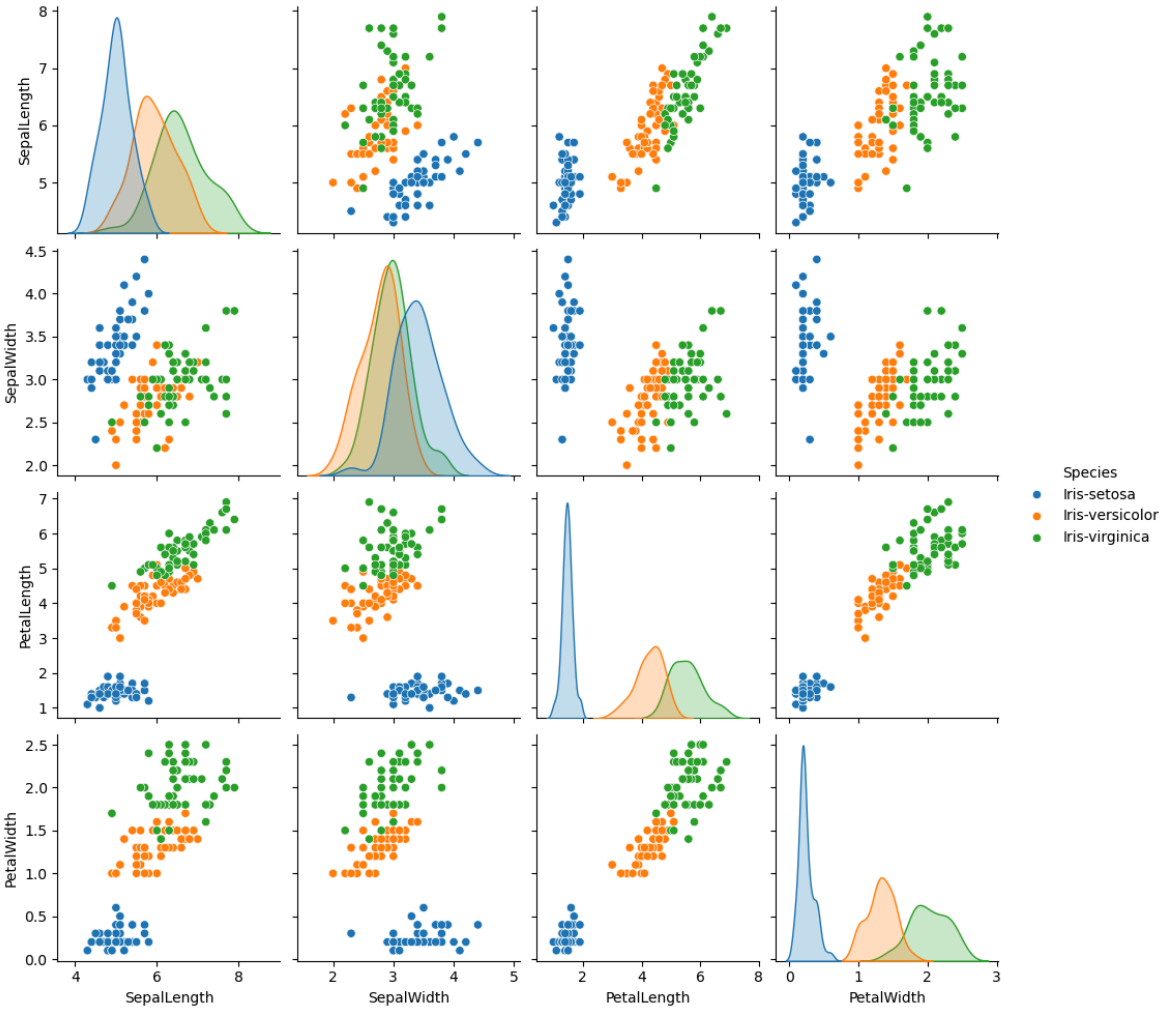


8. Pair Plot = A “pairs plot” is also known as a scatterplot, in which one variable in the same data row is matched with another variable's value, like this: Pairs plots are just elaborations on this, showing all variables paired with all the other variables.

In [69]:
```
sns.pairplot(data=iris,kind='scatter')
plt.show()
```

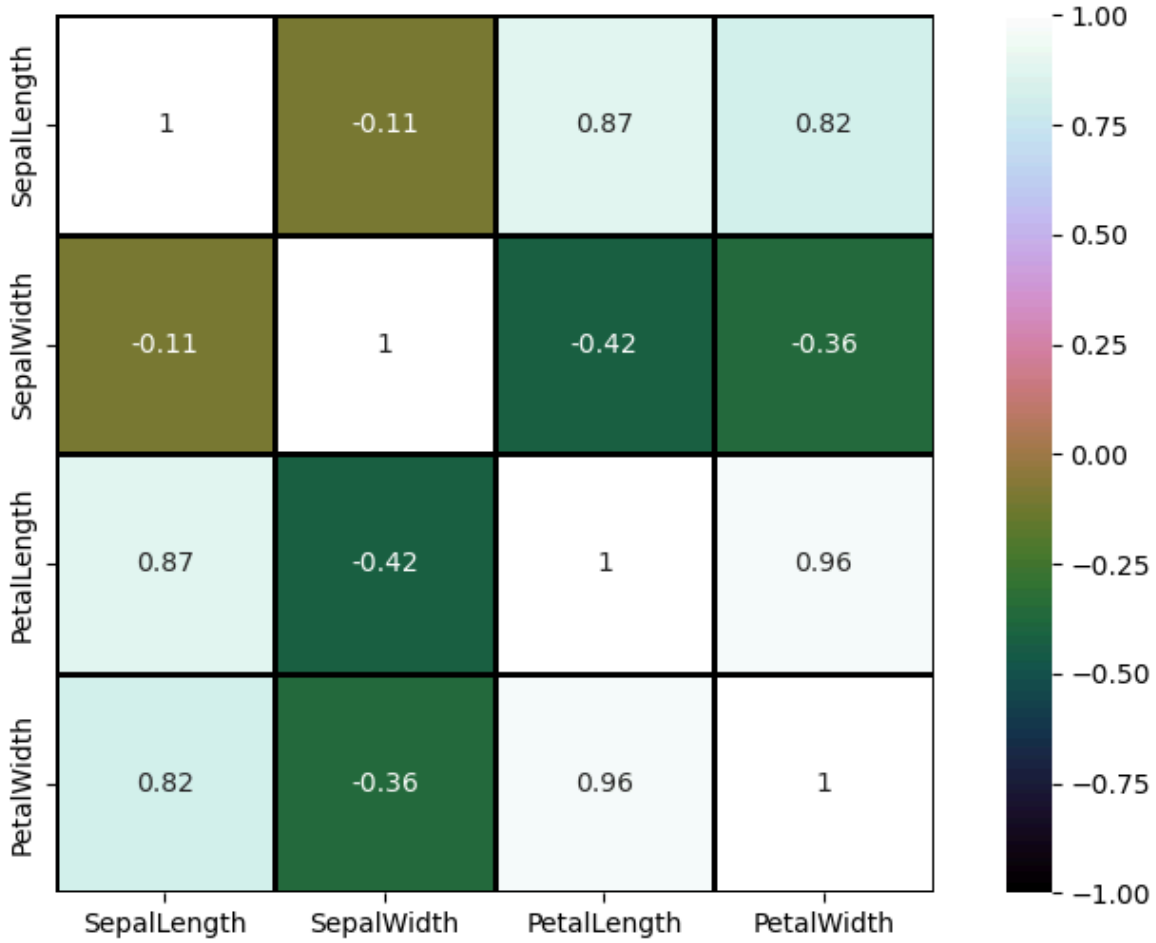


In [71]:
```
sns.pairplot(data=iris,kind='scatter', hue='Species')
plt.show()
```



9. 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 [84]:
```
fig=plt.gcf()
fig.set_size_inches(10,6)
numerical_iris = iris.select_dtypes(include=['number'])
fig=sns.heatmap(numerical_iris.corr(),annot=True,cmap='cubehelix',linewidths=1,l
plt.show()
```

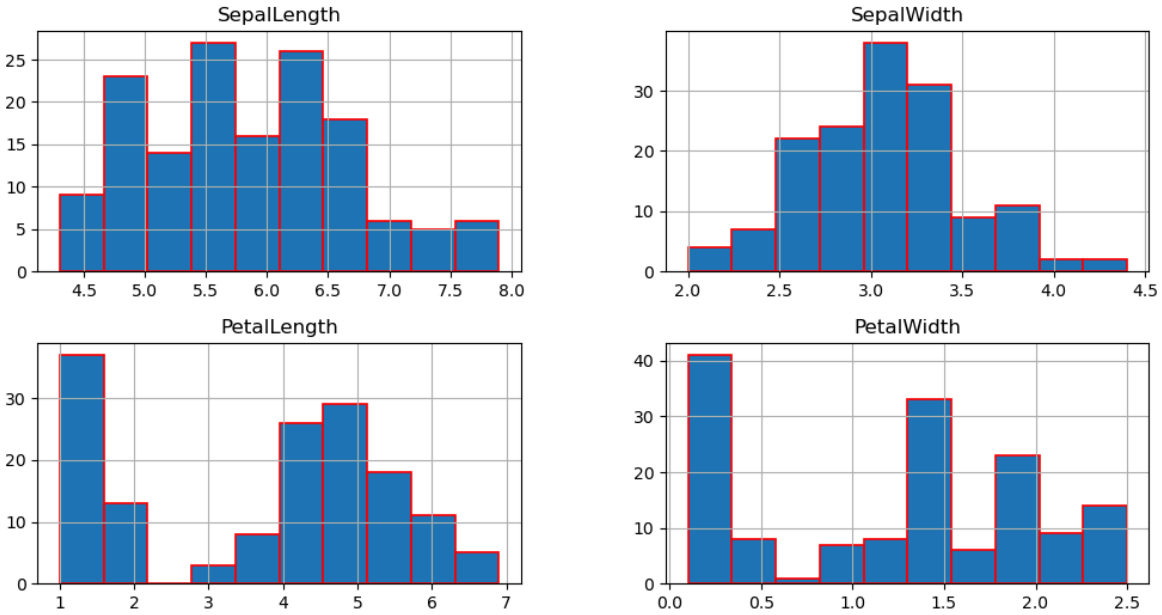


In [82]:
```
iris['Species'].value_counts()
```

Out[82]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

10. 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.

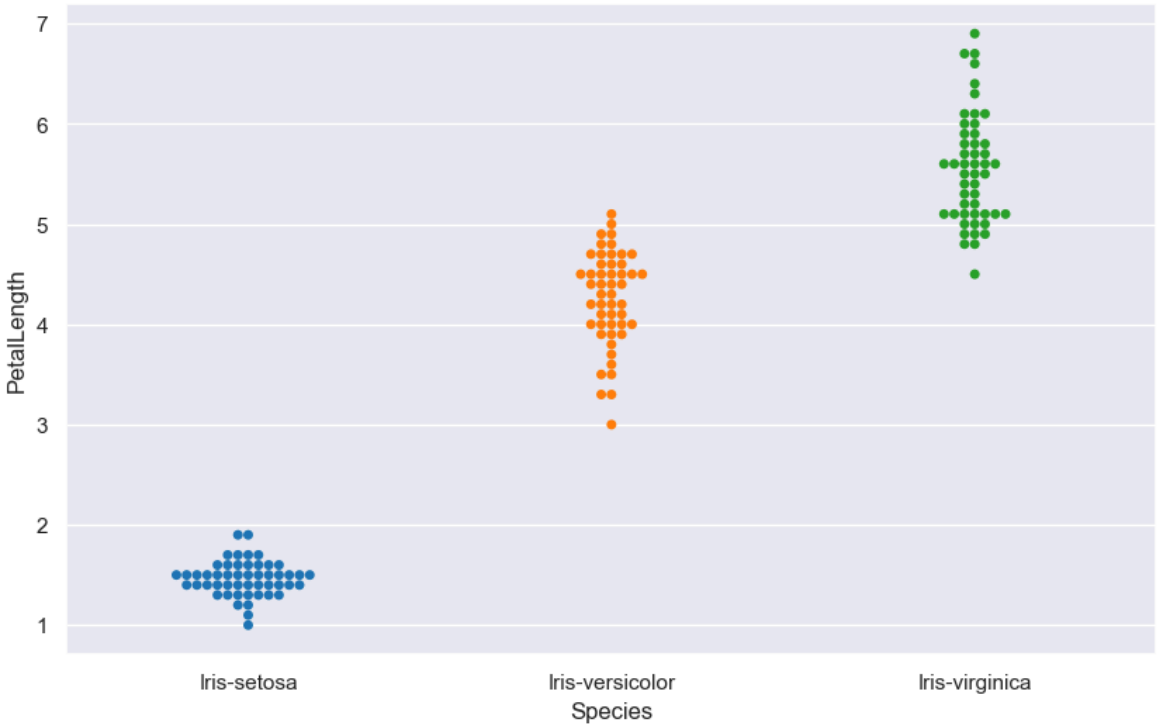
In [96]:
```
iris.hist(edgecolor='red', linewidth=1.2,)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```



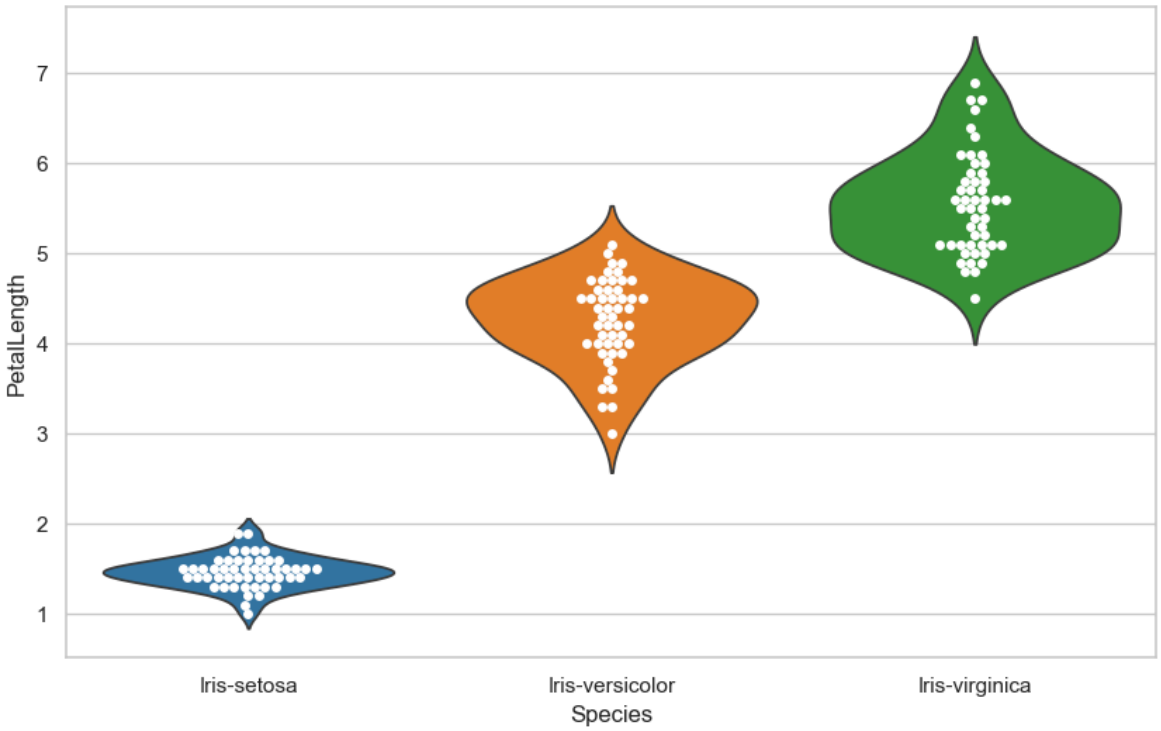
11. Swarm Plot = It looks a bit like a friendly swarm of bees buzzing about their hive. More importantly, each data point is clearly visible and no data are obscured by overplotting.A beeswarm plot improves upon the random jittering approach to move data points the minimum distance away from one another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

In [103…

```
sns.set(style="darkgrid")
fig=plt.gcf()
fig.set_size_inches(10,6)
fig = sns.swarmplot(x="Species", y="PetalLength", data=iris, palette = 'tab10')
plt.show()
```

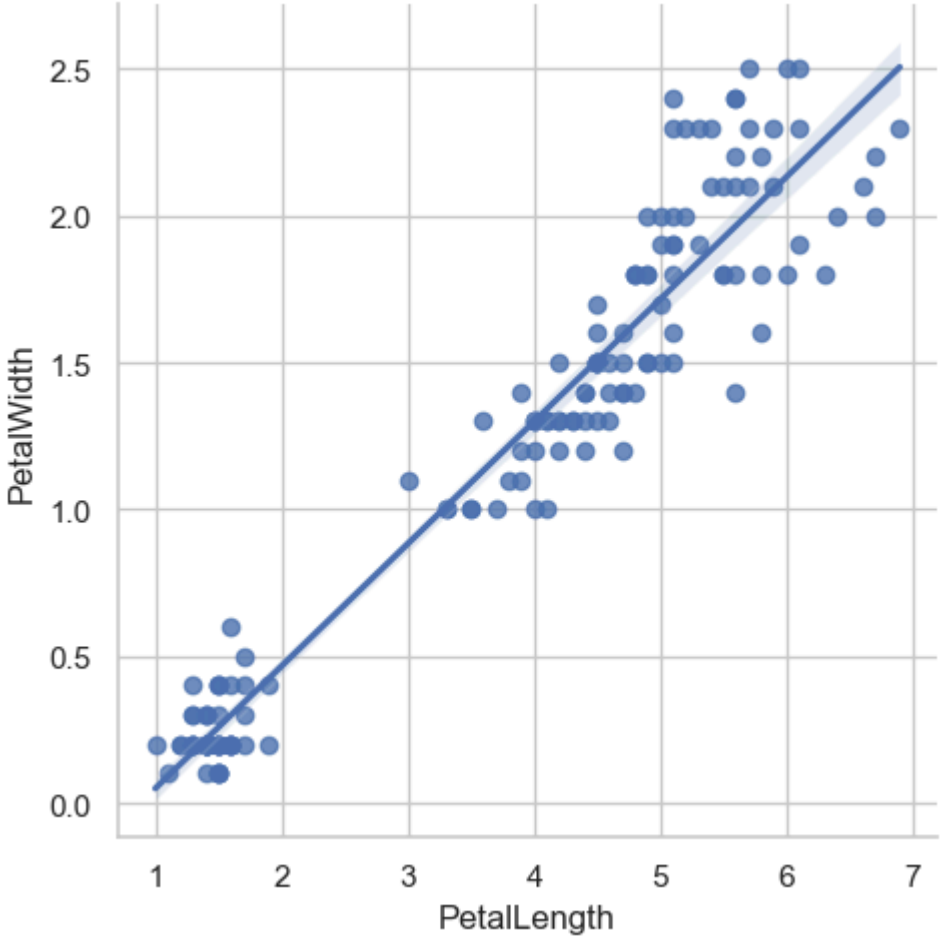


In [111…
```
sns.set(style="whitegrid")
fig=plt.gcf()
fig.set_size_inches(10,6)
ax = sns.violinplot(x="Species", y="PetalLength", data=iris,palette='tab10', inn
ax = sns.swarmplot(x="Species", y="PetalLength", data=iris,color="white", edgecc
plt.show()
# inner = by default it gives black line beneath the white dots. so we are remov
```



12. Linear Mode plot

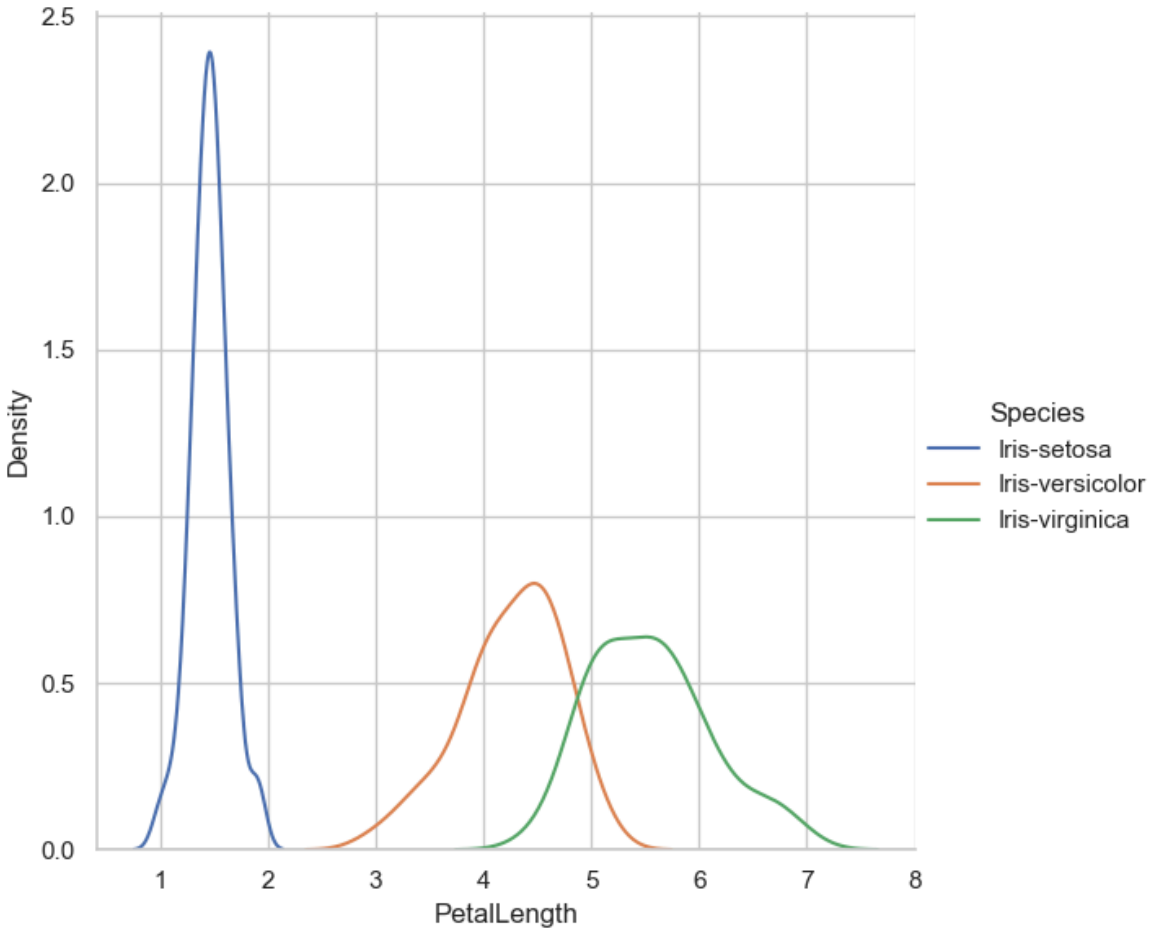
In [114…
```
fig=sns.lmplot(x="PetalLength", y="PetalWidth",data=iris)
plt.show()
```



13. FacetGrid

In [123…

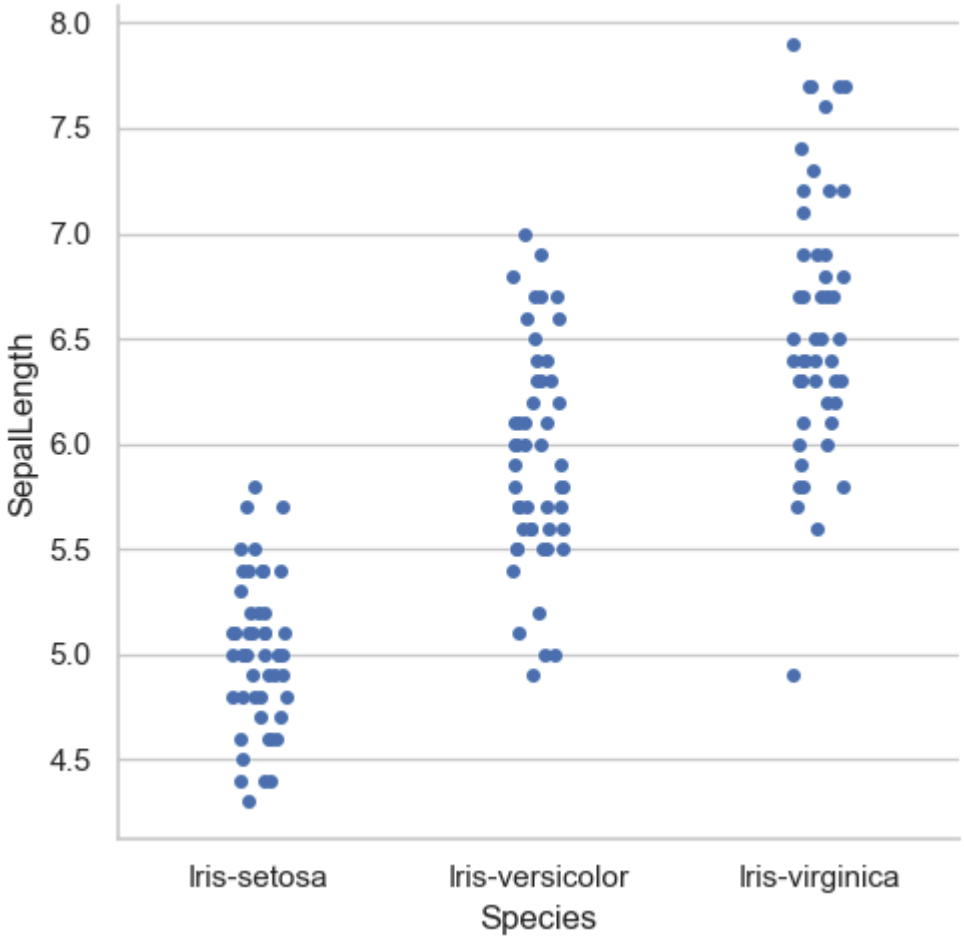
```
sns.FacetGrid(iris, hue="Species", height=6) \
   .map(sns.kdeplot, "PetalLength") \
   .add_legend()
plt.show()
```



14. Factor Plot = it is replaced with catplot()

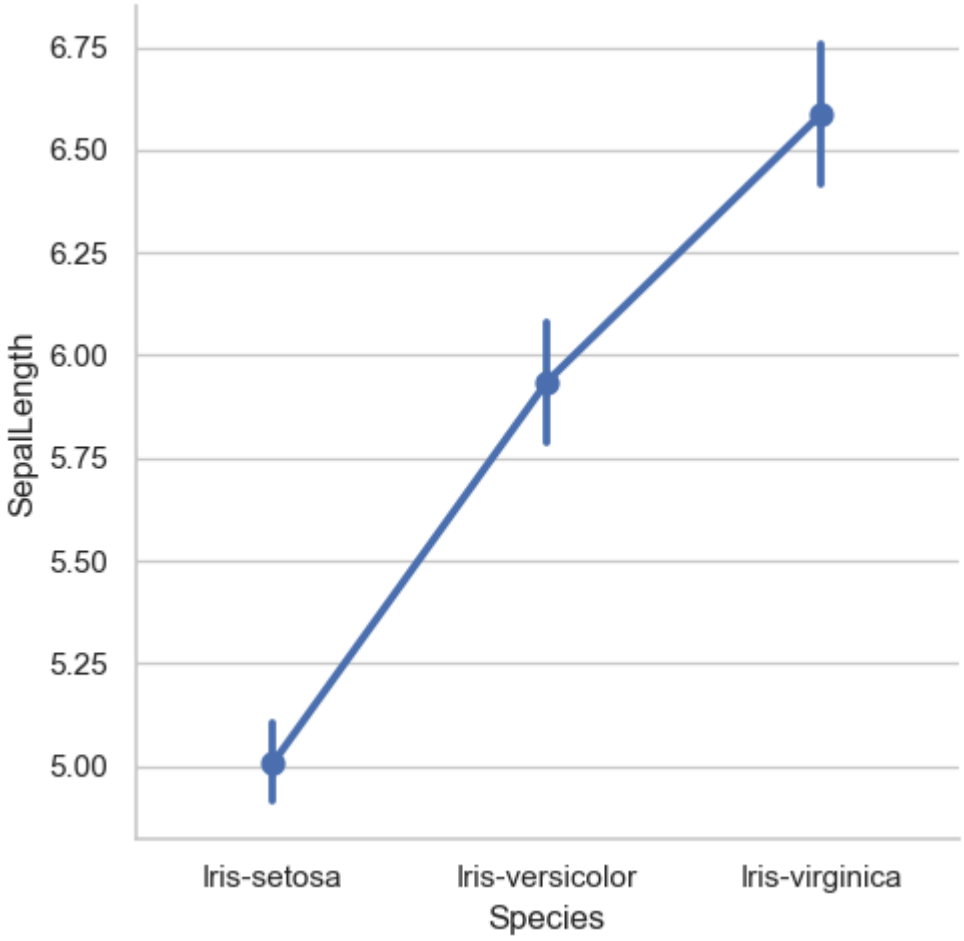
In [161…

```
sns.catplot(x='Species',y='SepalLength', data=iris,)
plt.show()
```



In [163…

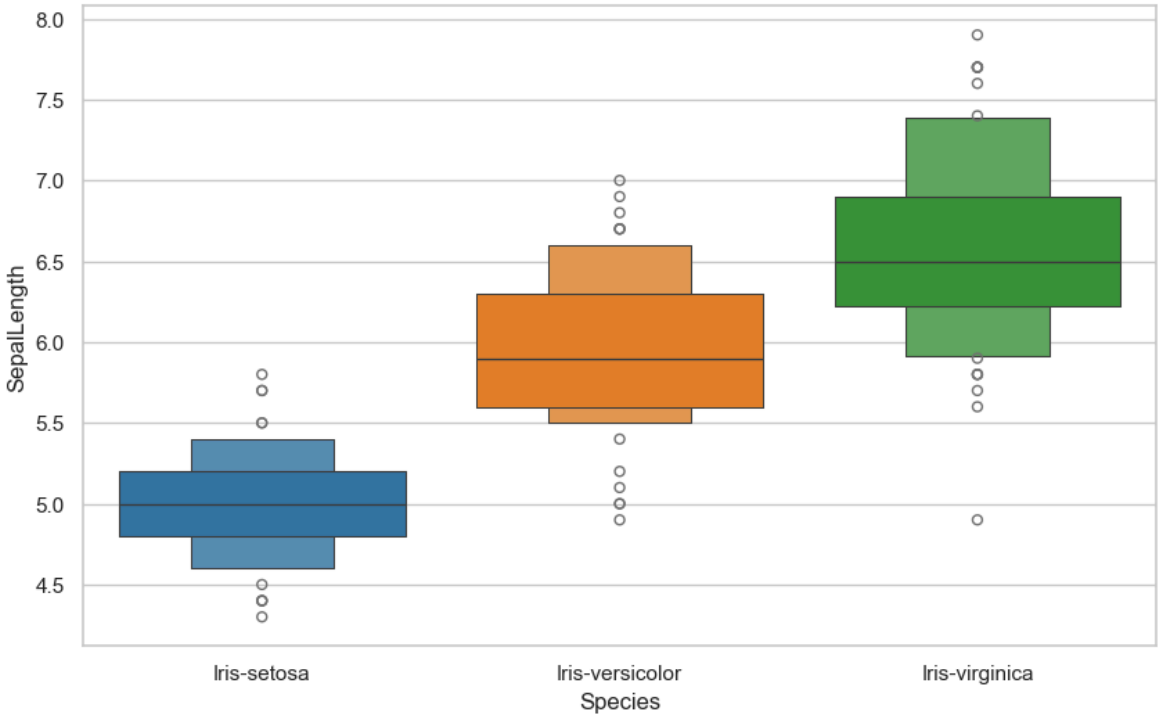
```
sns.catplot(x='Species',y='SepalLength', data=iris, kind='point')
plt.show()
```



15. Boxen Plot = a "boxen plot" (also known as a "letter-value plot") is a visualization that provides a more detailed view of the distribution of a numerical variable compared to a traditional box plot. it's almost similar to boxplot

In [170…

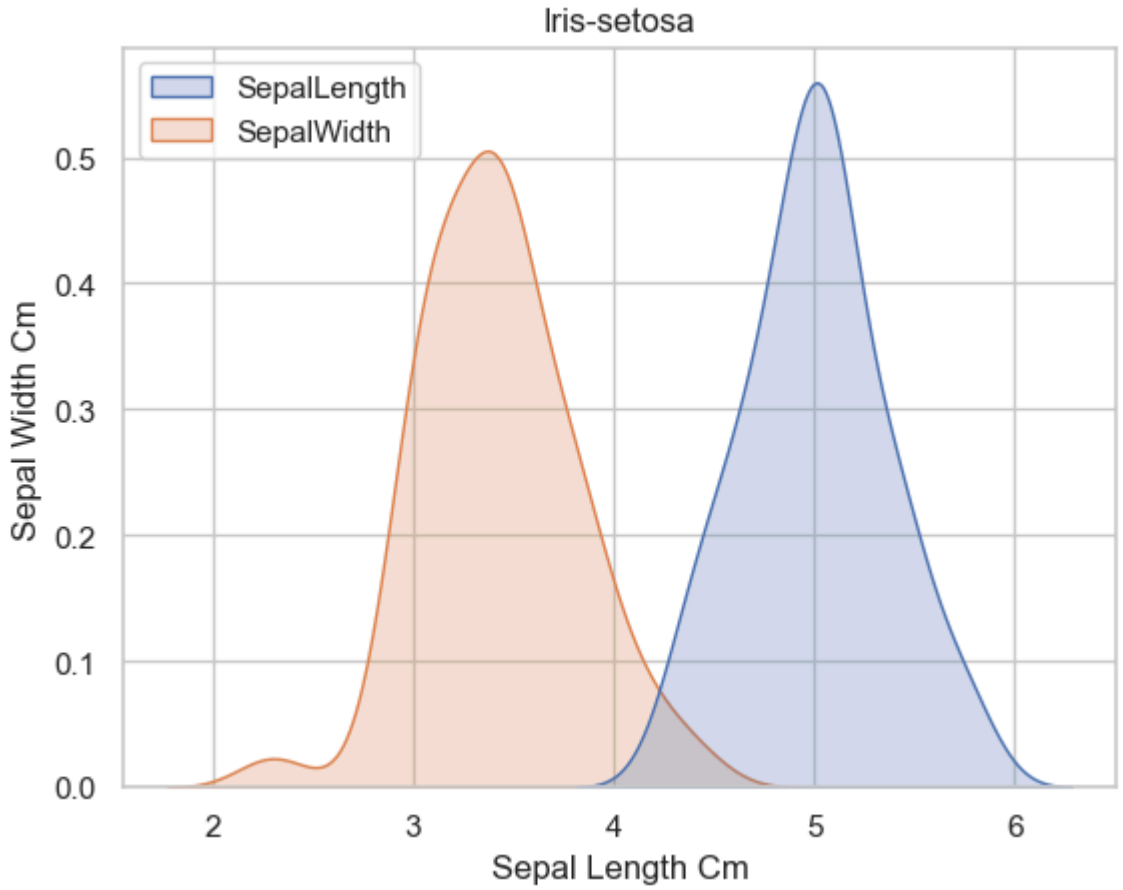
```
fig=plt.gcf()
fig.set_size_inches(10,6)
fig=sns.boxenplot(x='Species',y='SepalLength',data=iris, palette = 'tab10')
plt.show()
```



16. KDE plot

In [181…

```
sub=iris[iris['Species']=='Iris-setosa']
sns.kdeplot(data=sub[['SepalLength','SepalWidth']],color="plasma", shade=True, s
plt.title('Iris-setosa')
plt.xlabel('Sepal Length Cm')
plt.ylabel('Sepal Width Cm')
plt.show()
```

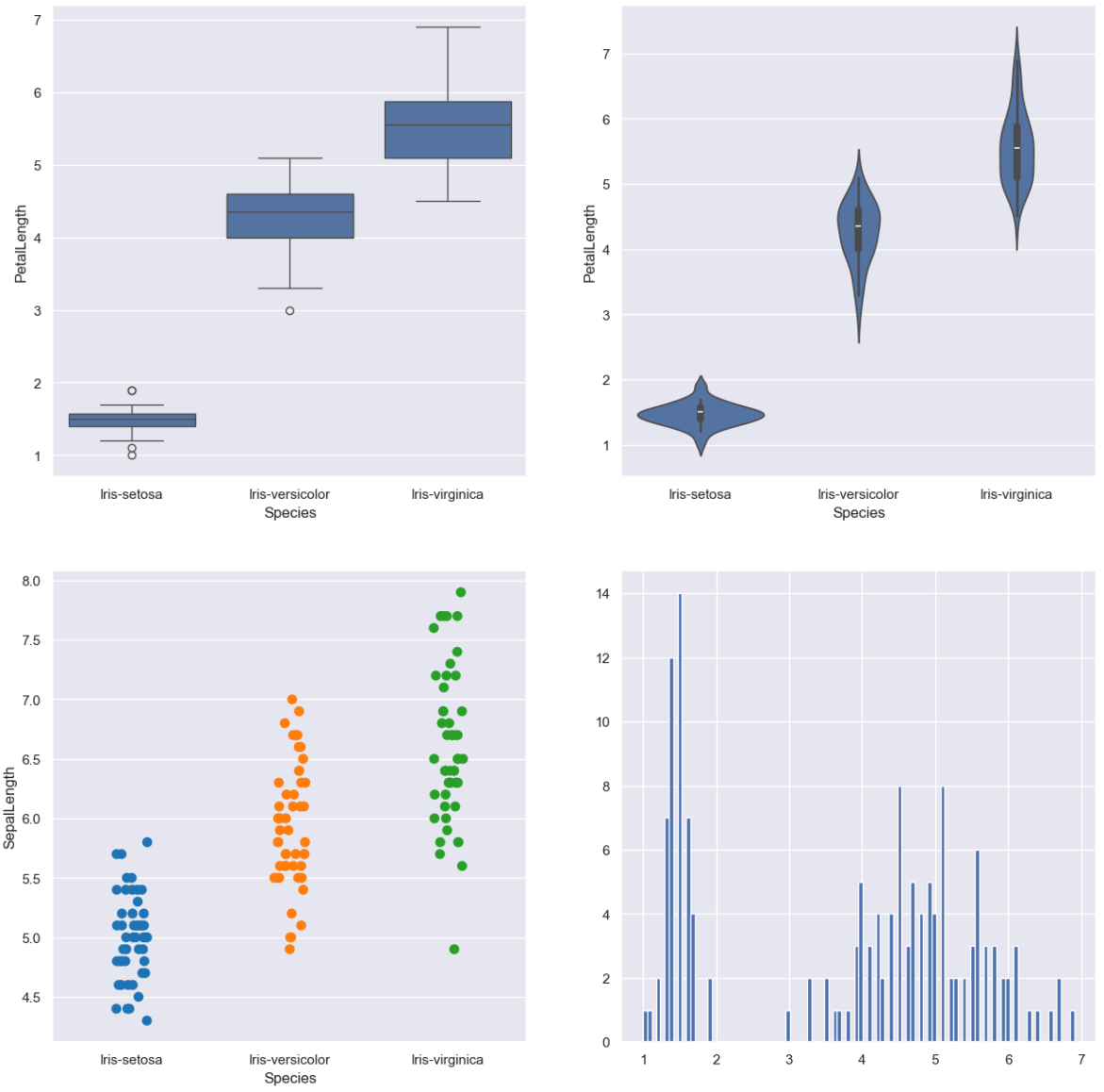


17. Dashboard

In [188…

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

k1=sns.boxplot(x="Species", y="PetalLength", data=iris,ax=axes[0,0])
k2=sns.violinplot(x='Species',y='PetalLength',data=iris,ax=axes[0,1])
k3=sns.stripplot(x='Species',y='SepalLength',data=iris,jitter=True,edgecolor='gr
#axes[1,1].hist(iris.hist,bin=10)
axes[1,1].hist(iris.PetalLength,bins=100)
#k2.set(xlim=(-1,0.8))
plt.show()
```



18. Stacked Histogram

In [191…

```
iris['Species'] = iris['Species'].astype('category')
```

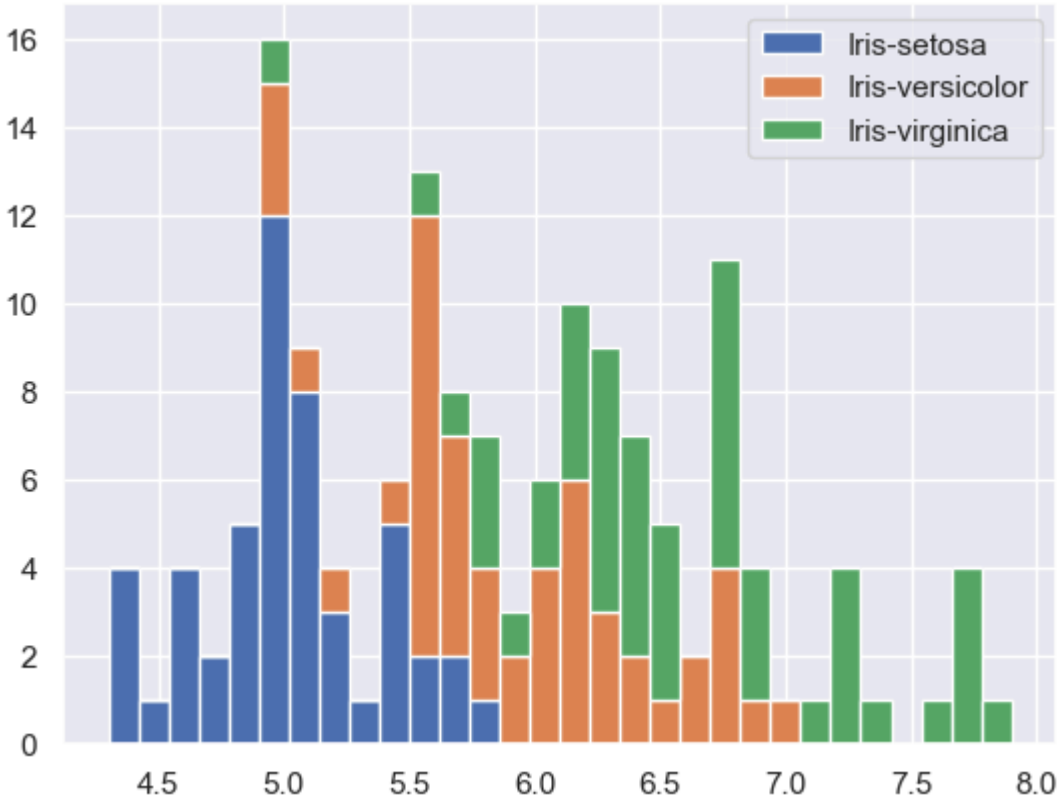
In [193…

```
iris.dtypes
```

Out[193…
```
SepalLength     float64
SepalWidth      float64
PetalLength     float64
PetalWidth      float64
Species        category
dtype: object
```

In [197…
```
list1=list()
mylabels=list()
for gen in iris.Species.cat.categories:
    list1.append(iris[iris.Species==gen].SepalLength)
    mylabels.append(gen)

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



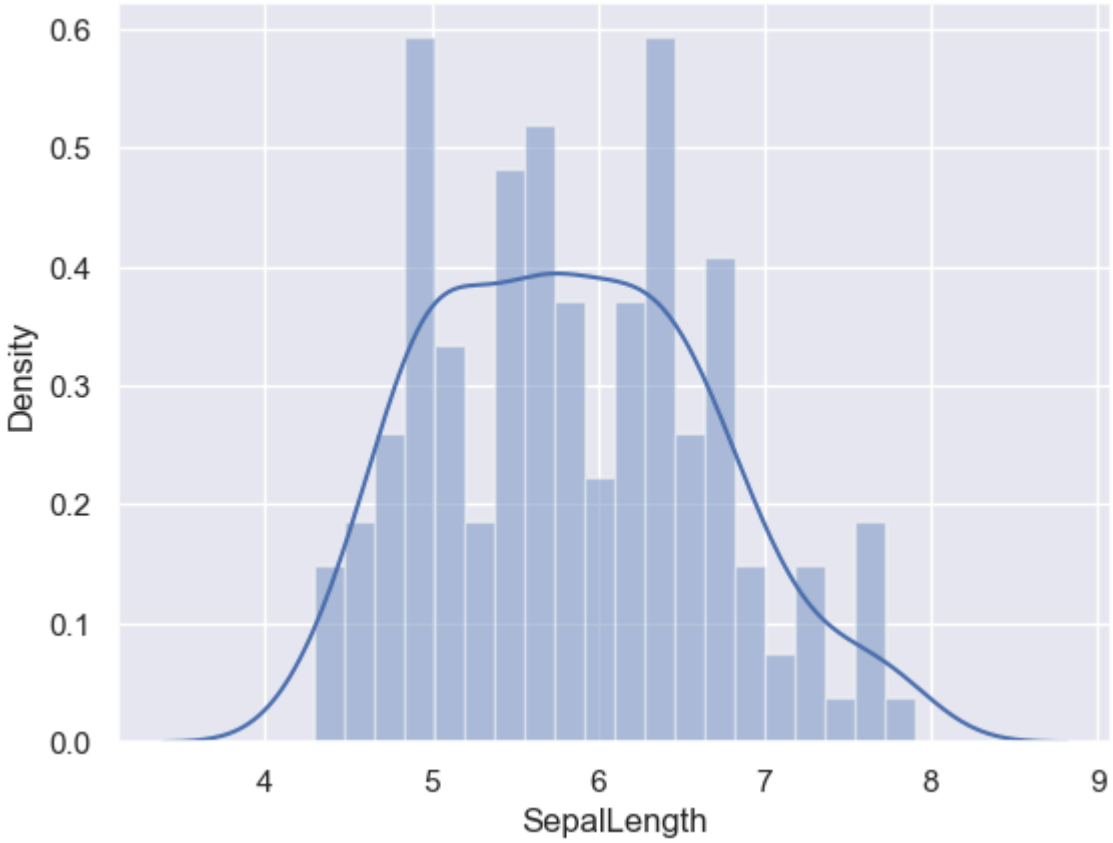
19. Area Plot = Area Plot gives us a visual representation of Various dimensions of Iris flower and their range in dataset.

In [200…
```
iris.plot.area(y=['SepalLength','SepalWidth','PetalLength','PetalWidth'],alpha=0
plt.show()
# alpha = it controls the transparency of the filled areas.
```

20. Distplot

In [203…

```
sns.distplot(iris['SepalLength'],kde=True,bins=20);
plt.show()
```



EDA completed