25-07-2025 (TASK)

PROJECT - IRIS DATASET - VISUALIZATION(SEABORN , MATPLOTLIB)

import required libraries & SEABORN module

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

importing IRIS dataset

In [2]:
```
iris = pd.read_csv(r"C:\Users\Hanshu\Desktop\excel data\Iris.csv")
iris                                                              # here iri
```

Out[2]:

	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 top 5 rows by using head() method

In [3]:

```
iris.head()
```

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

In [4]:

```
iris.columns
```

Out[4]:

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

now drop 'ID' column from dataset

In [5]:
```
iris.drop('Id', axis=1 , inplace=True)
```

now check dropped or not

In [6]:
```
iris.head()
```

Out[6]:

	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

now check, if there any missing values

In [7]:
```
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 [8]:
```
iris.isnull().sum()      # it will show , how many missing values are in each co
```

Out[8]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [9]:
```
iris['Species'].value_counts()    # it show, how many times each SPECIES appears
```

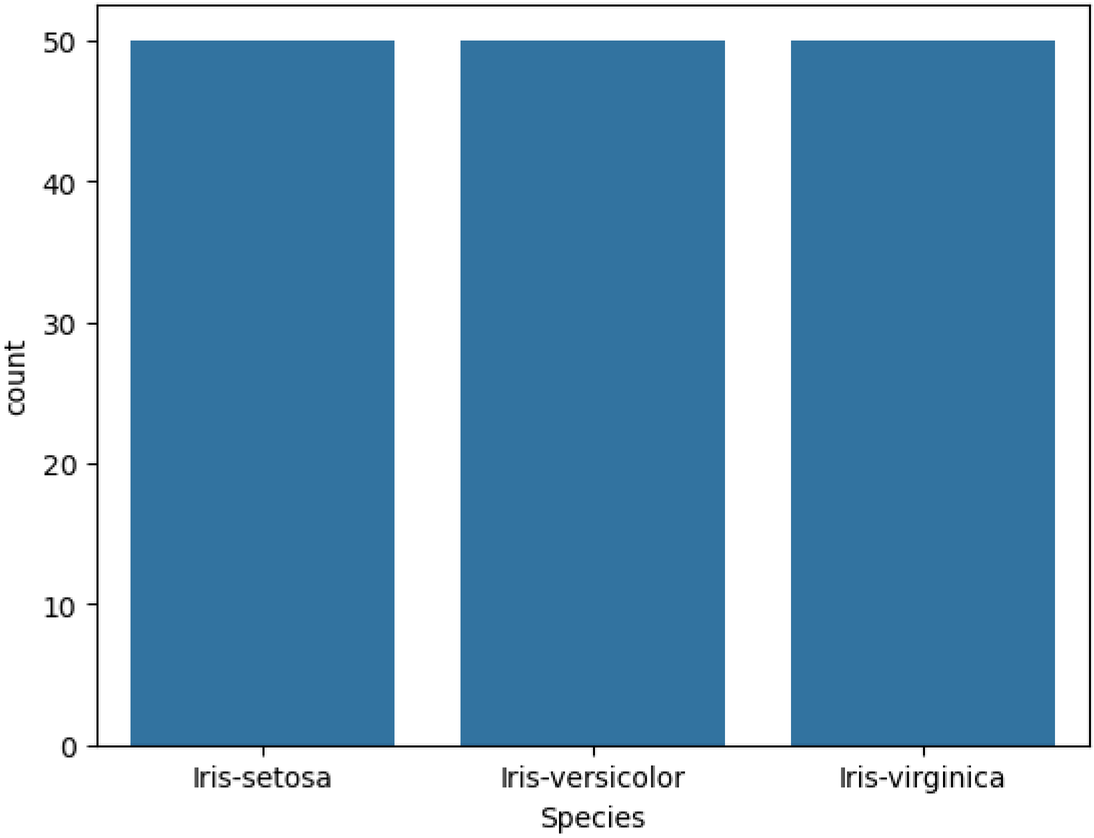
Out[9]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

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(x = '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.

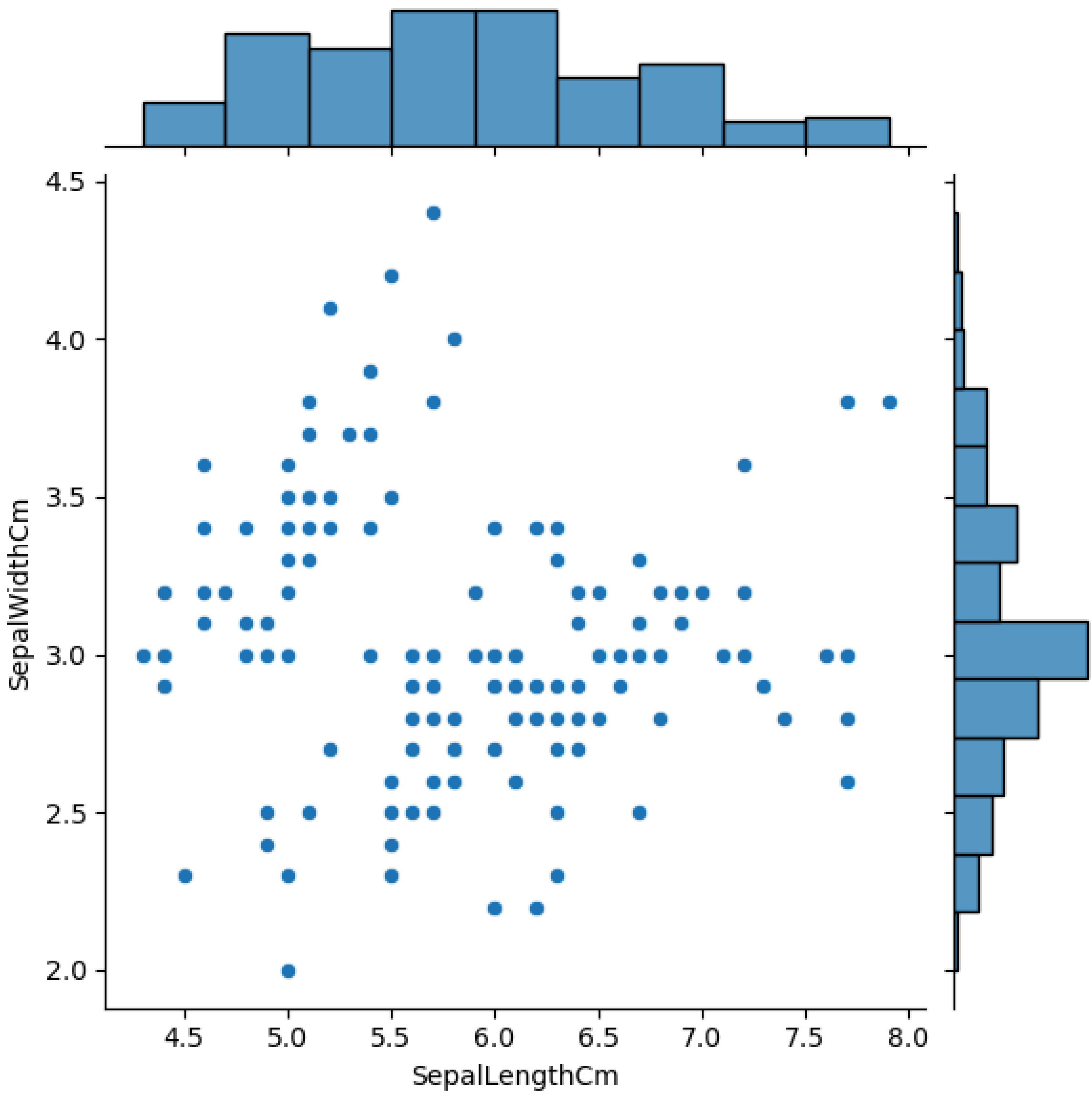
In [11]:
```
iris.head()
```

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

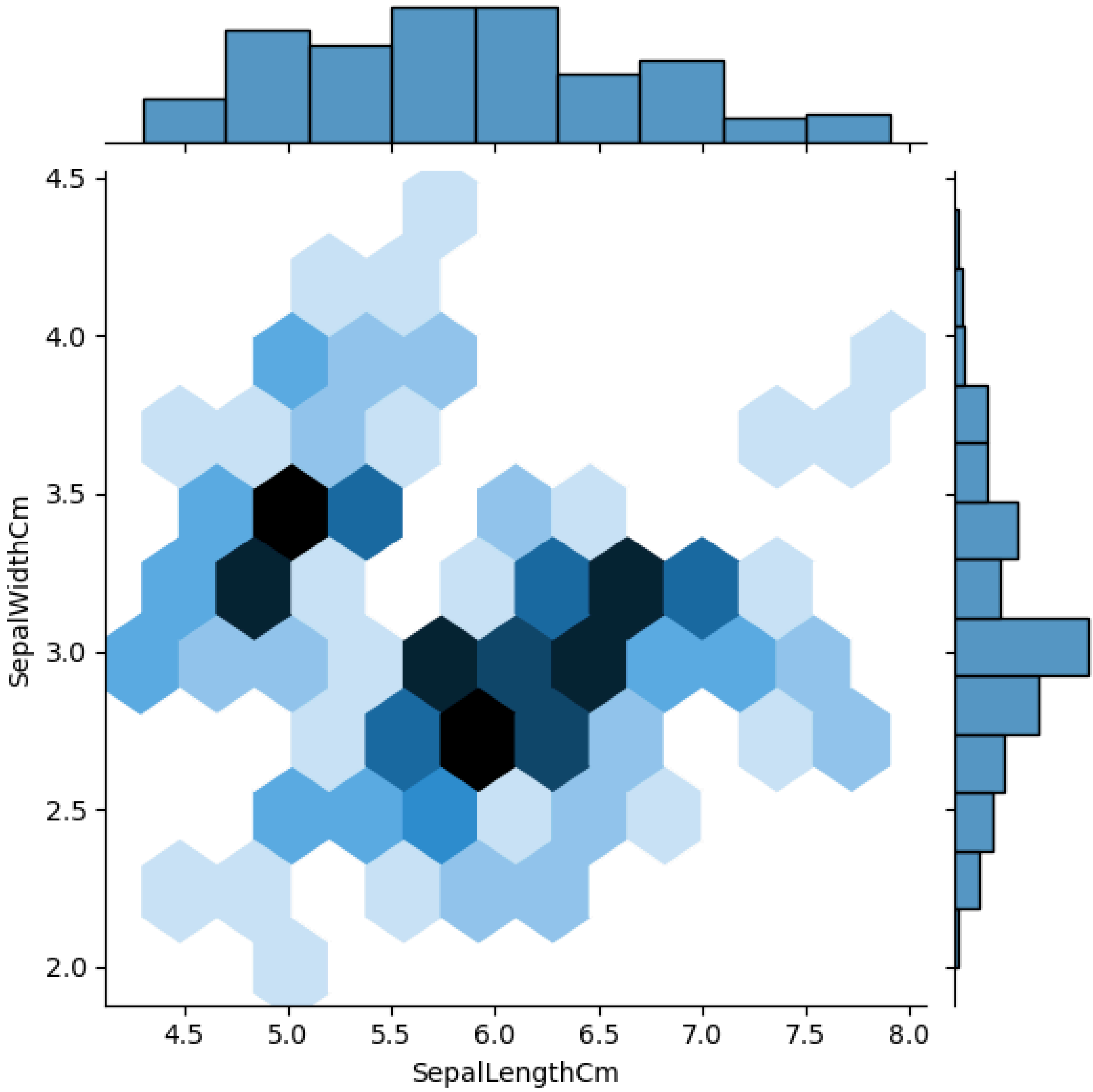
In [12]:

```
fig = sns.jointplot(x= 'SepalLengthCm' , y = 'SepalWidthCm' , data=iris )
```



In [13]:

```
fig = sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',kind='hex',data=iris)

# kind='hex' --> creates a hexbin plot (usefull for visualizing the density of p
```

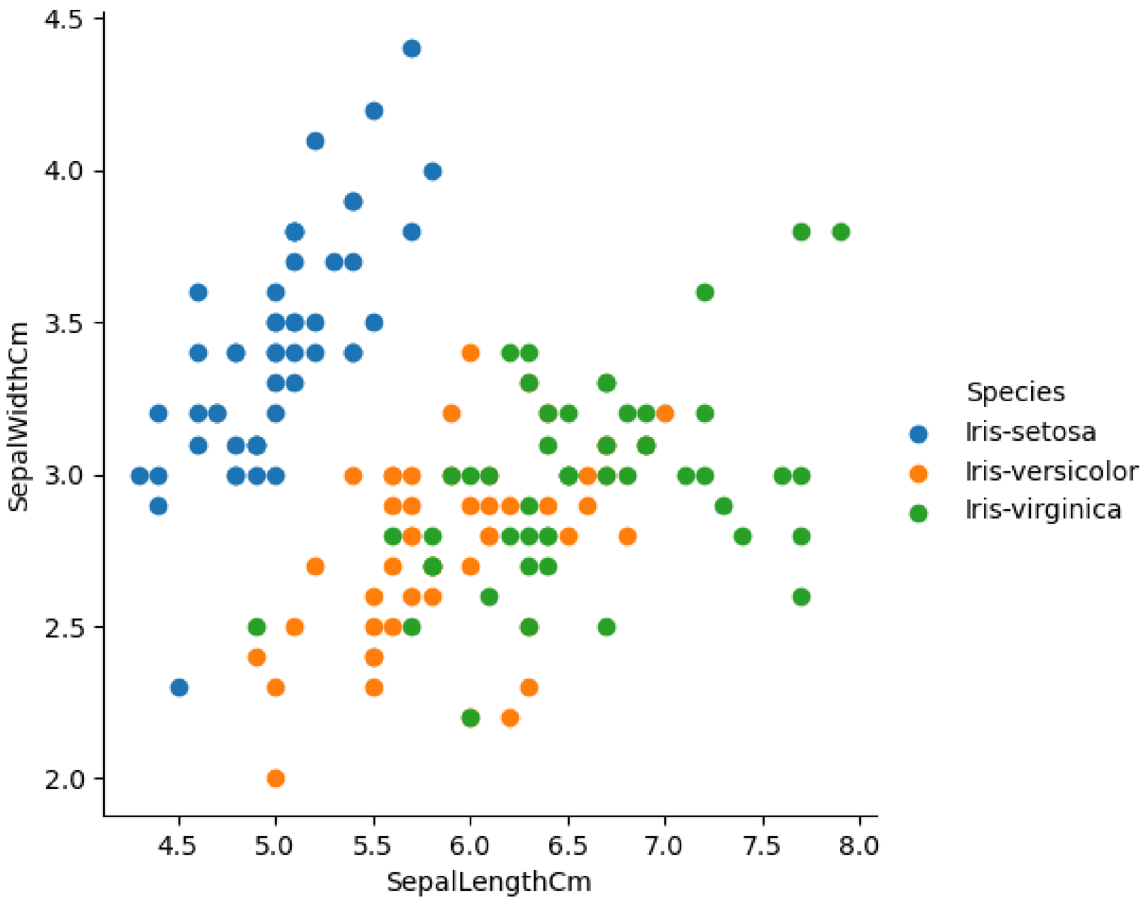


5. FaceGrid Plot

In [14]:
```
import matplotlib.pyplot as plt
%matplotlib inline

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

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x280ecd7f440>
```

In [15]:
```
plt.show()
```



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.

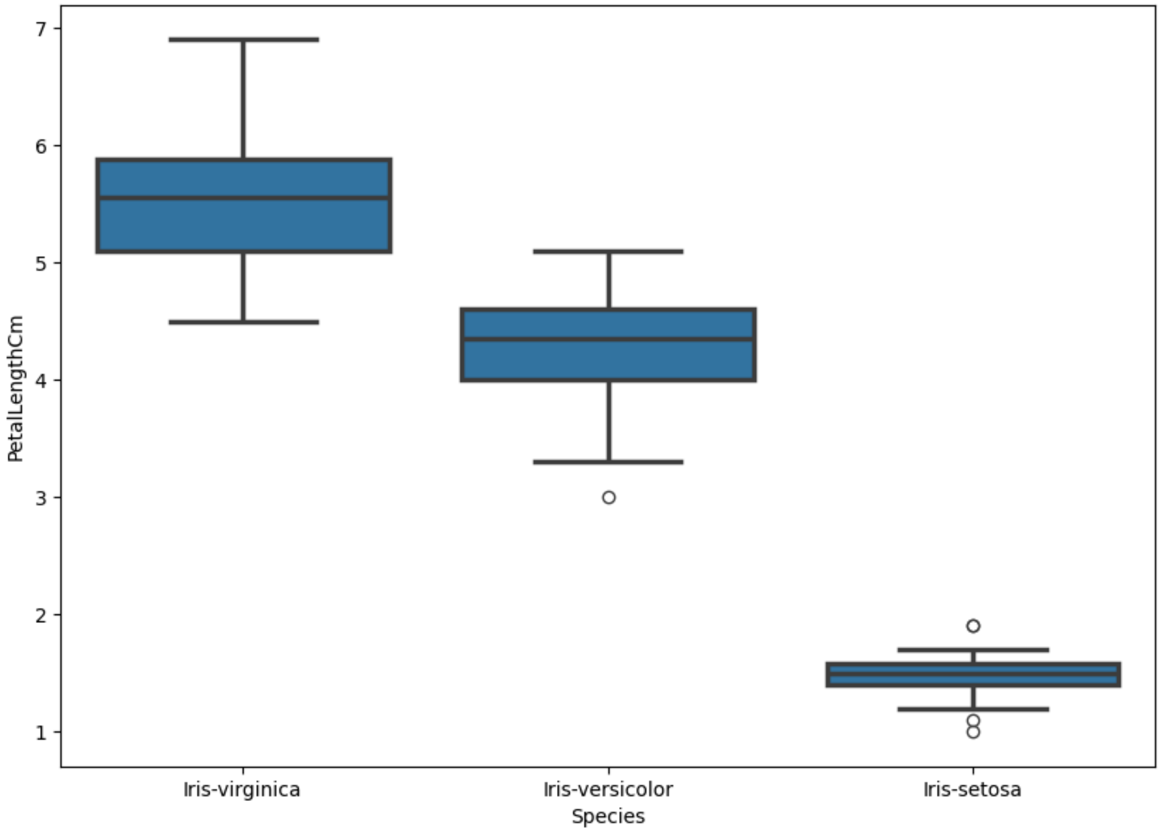
In [16]:
```
iris.head()
```

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
```

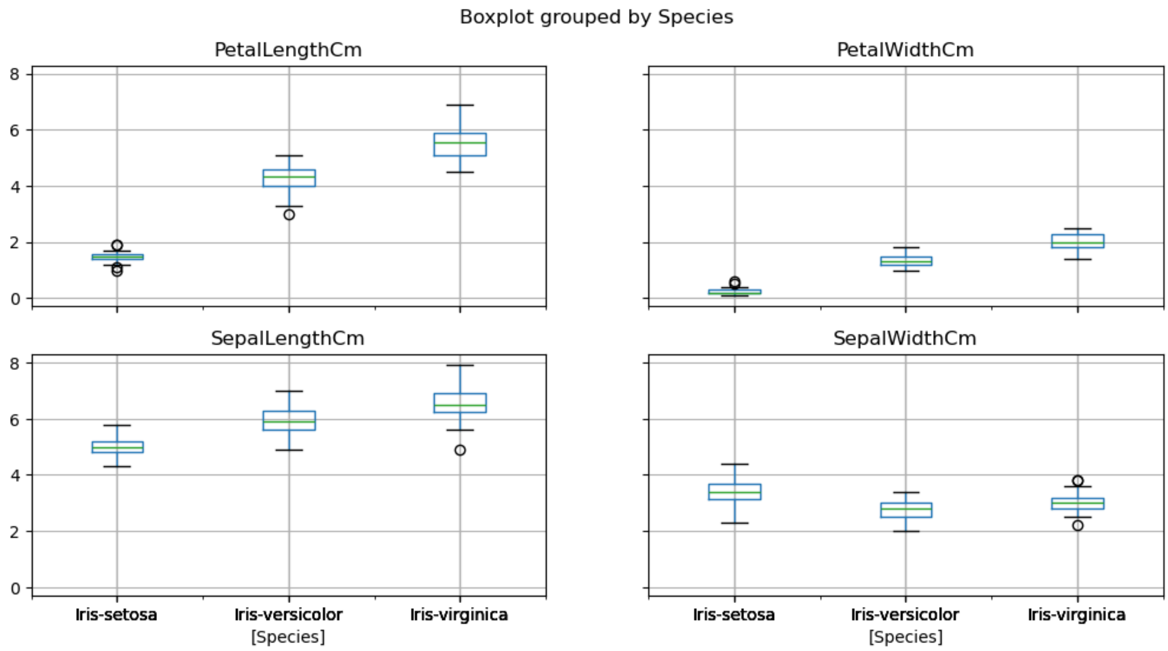
In [18]:
```
plt.show()
```



In [19]:
```python
#iris.drop("Id", axis=1).boxplot(by="Species", figsize=(12, 6))

iris.boxplot(by='Species', figsize=(12,6))
```

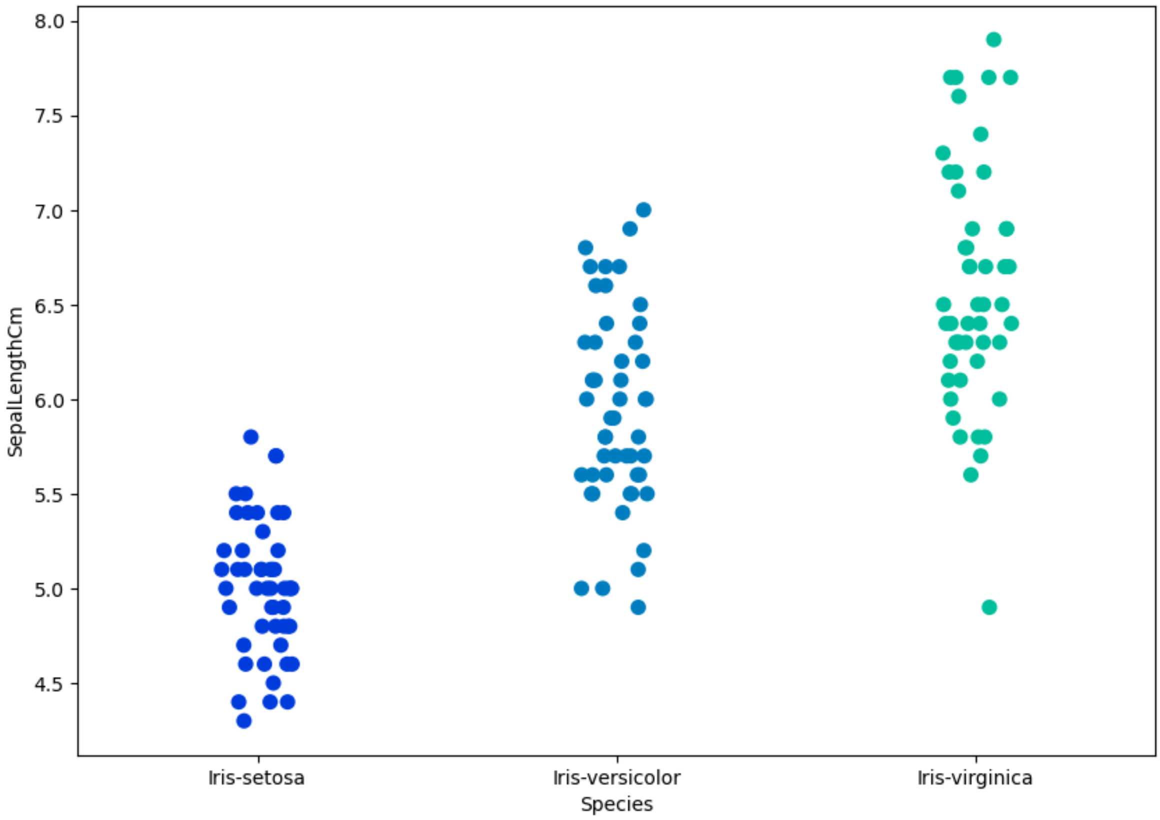
Out[19]:
```
array([[<Axes: title={'center': 'PetalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'PetalWidthCm'}, xlabel='[Species]'>],
       [<Axes: title={'center': 'SepalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```

In [20]:
```python
plt.show()
```



7. strip plot

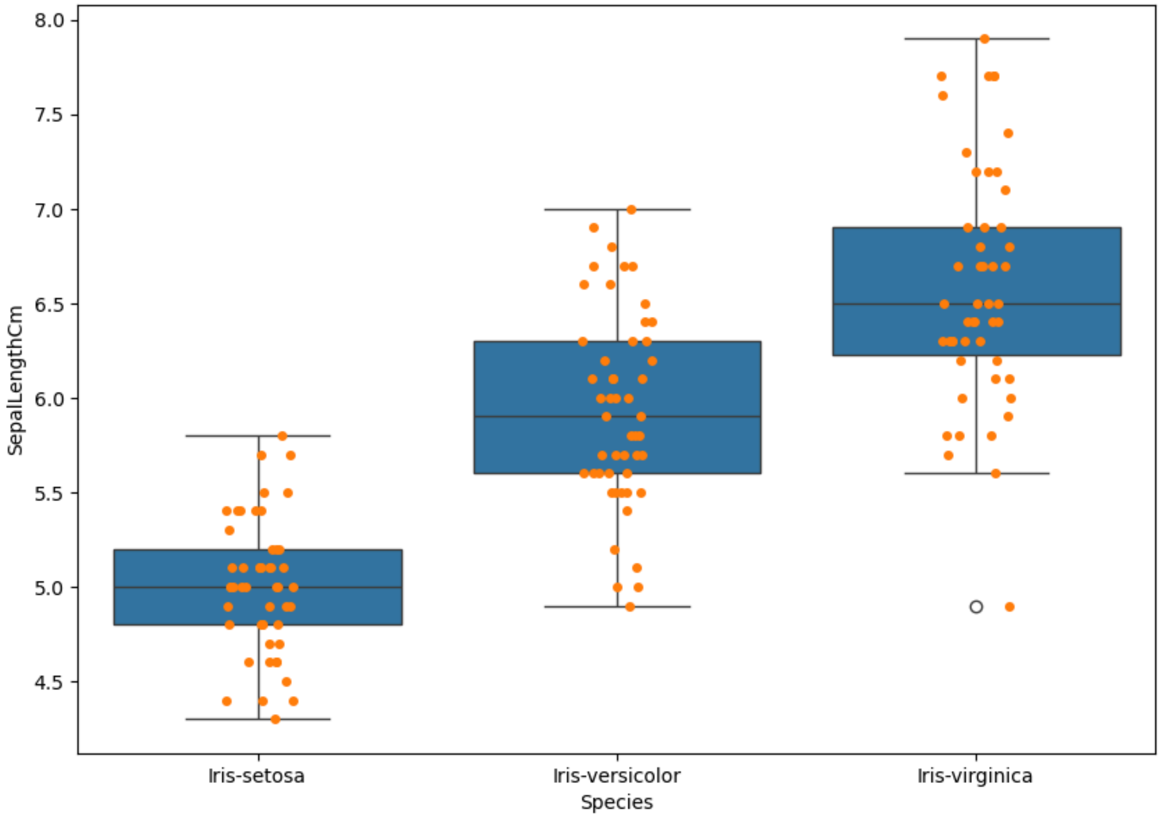
In [21]:
```
fig = plt.gcf()
fig.set_size_inches(10,7)
fig=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor=
plt.show()
```



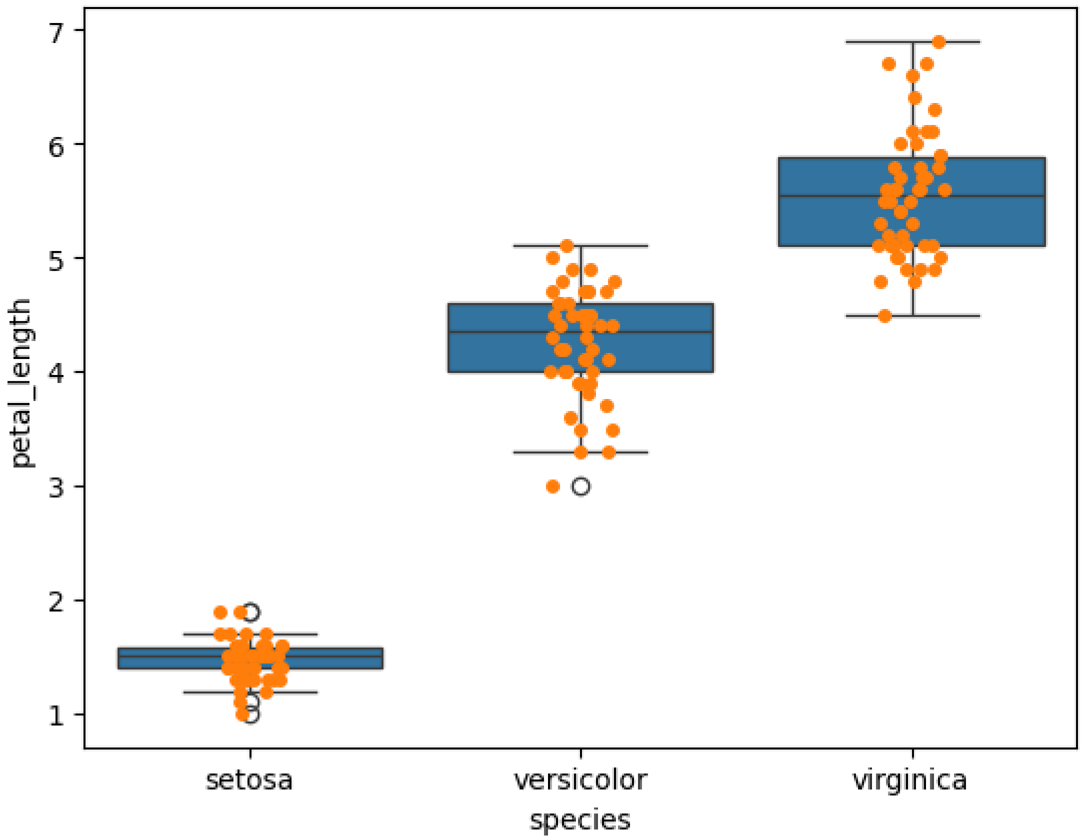
8. Combining Box and Strip Plots

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,edgecol
```

In [23]:
```
plt.show()
```



In [24]:
```
# Load iris dataset
iris = sns.load_dataset("iris")

# Create the boxplot and stripplot
ax = sns.boxplot(x="species", y="petal_length", data=iris)
sns.stripplot(x="species", y="petal_length", data=iris, jitter=True, edgecolor="

# Change colors of the boxplot elements
colors = ['green', 'red', 'yellow']  # setosa, versicolor, virginica

for i, box in enumerate(ax.artists):
    box.set_facecolor(colors[i])
    box.set_edgecolor('black')

plt.show()
```



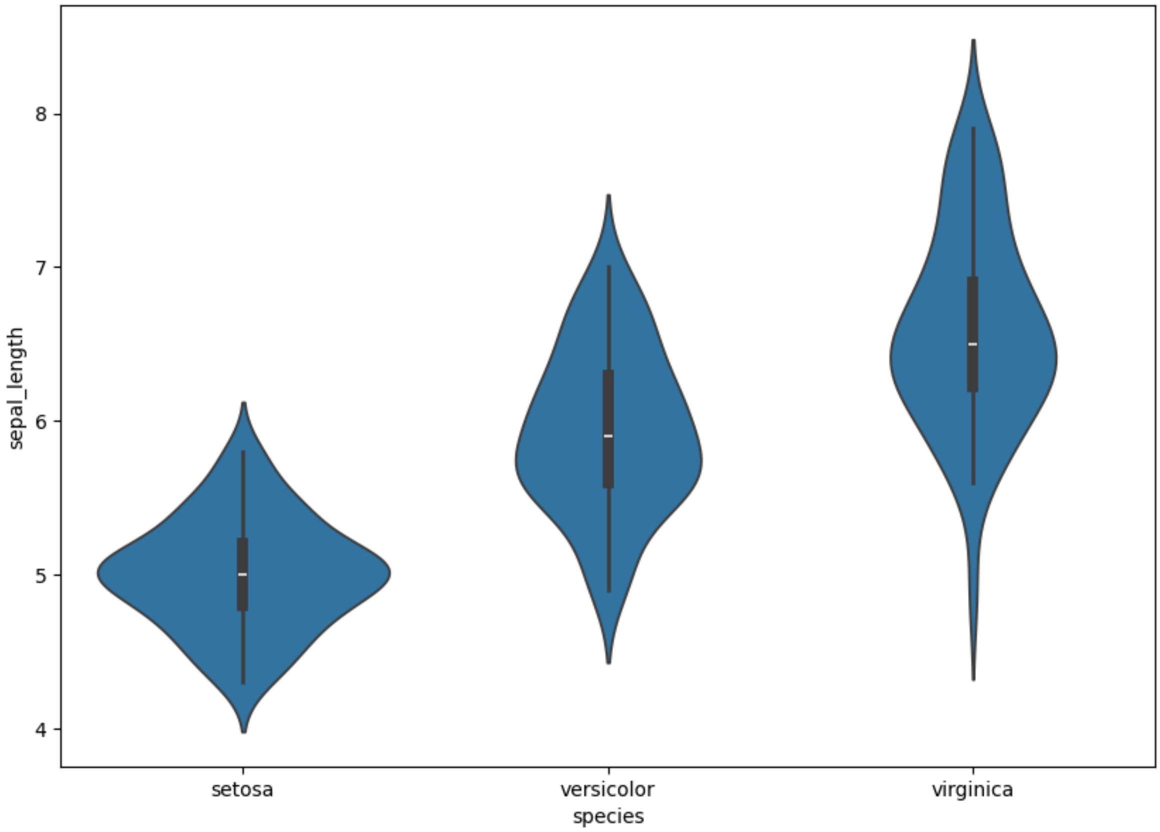
9. Violin Plot

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 [25]:
```
print(iris.columns)
```

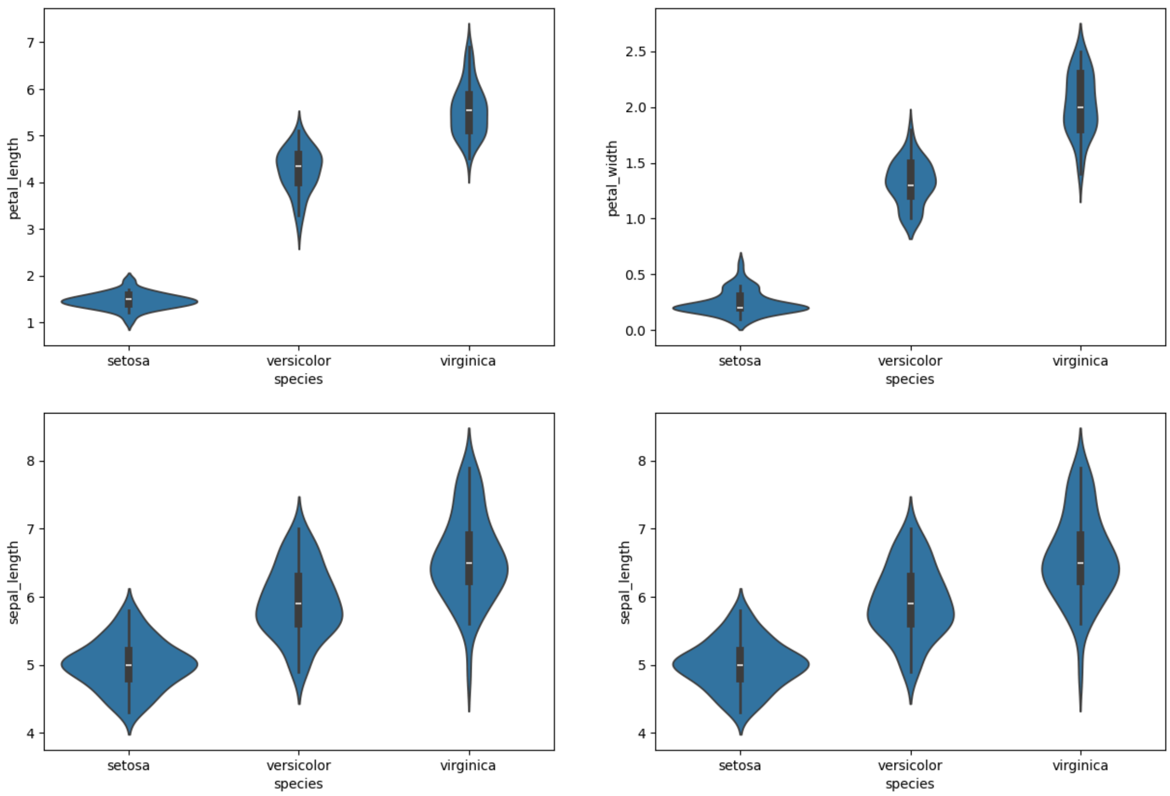
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [26]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.violinplot(x='species',y='sepal_length',data=iris)
plt.show()
```



In [27]:

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='species',y='petal_length',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='species',y='petal_width',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='species',y='sepal_length',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='species',y='sepal_length',data=iris)
plt.show()
```



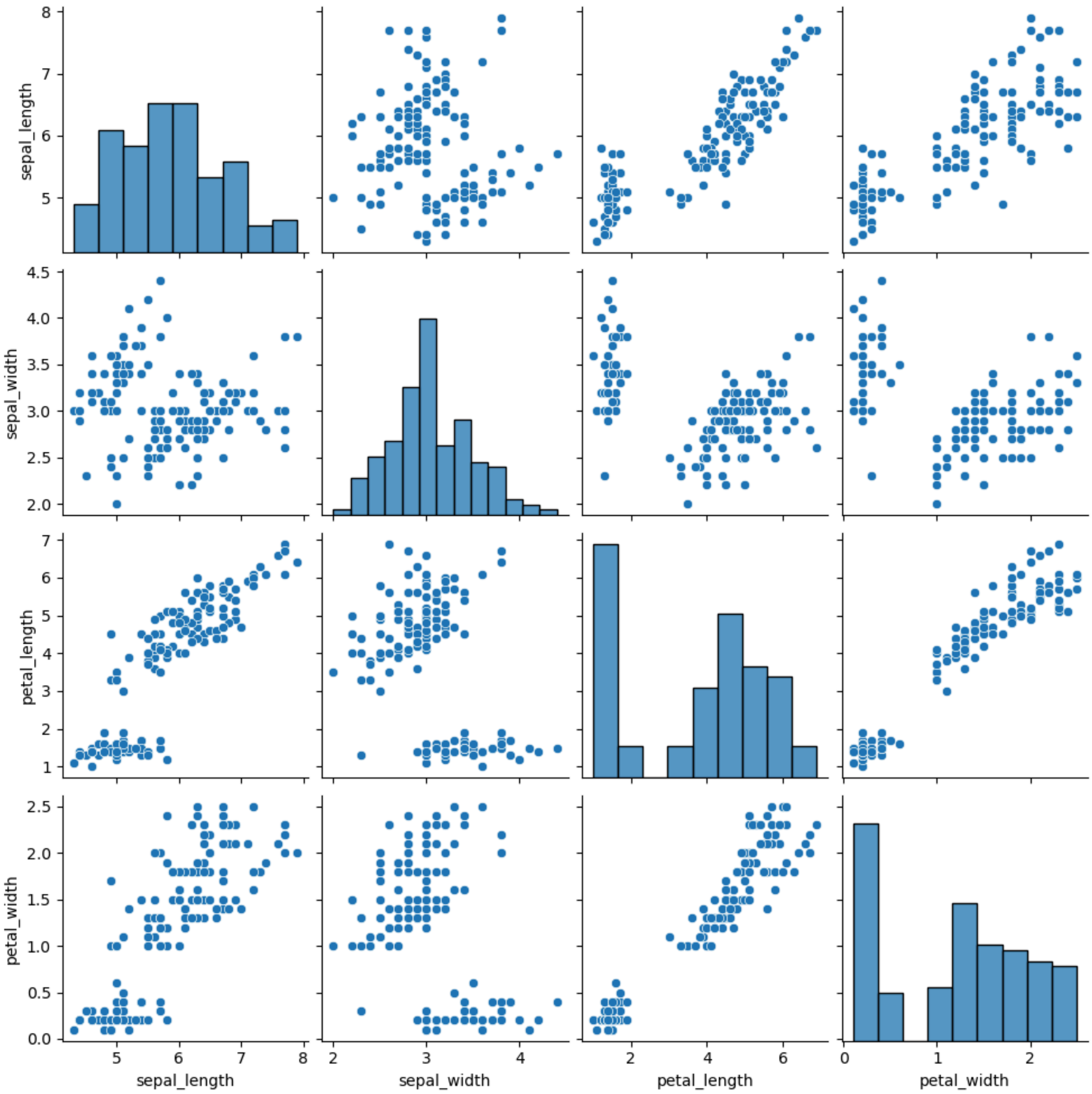
10. Pair Plot

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.

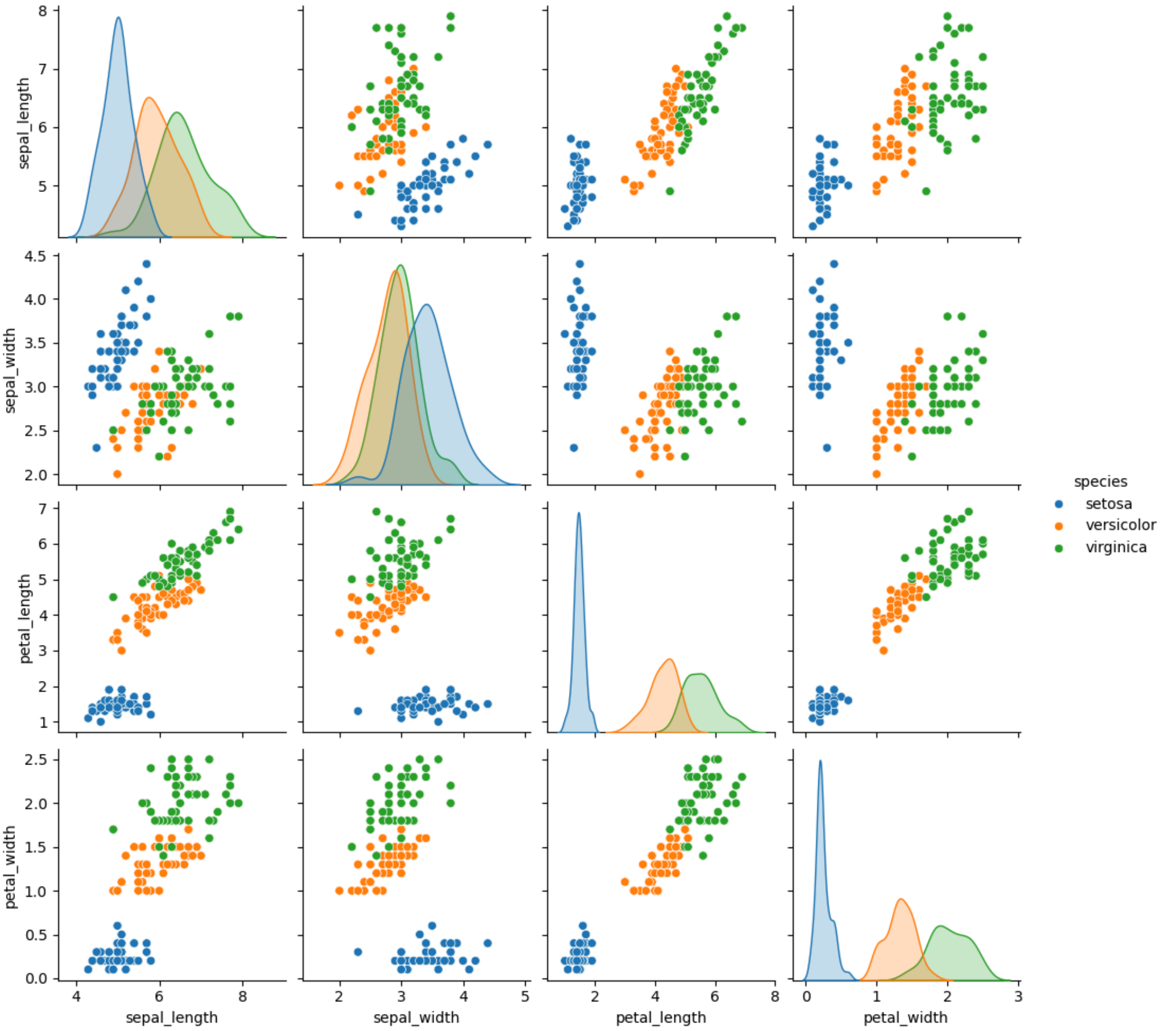
In [28]:
```
sns.pairplot(data=iris,kind= 'scatter')
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x280eceb9550>

In [29]:
```
plt.show()
```



In [30]:
```
sns.pairplot(data=iris, hue='species')
plt.show()
```



11. Het 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 [31]:
```
print(iris.dtypes)
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

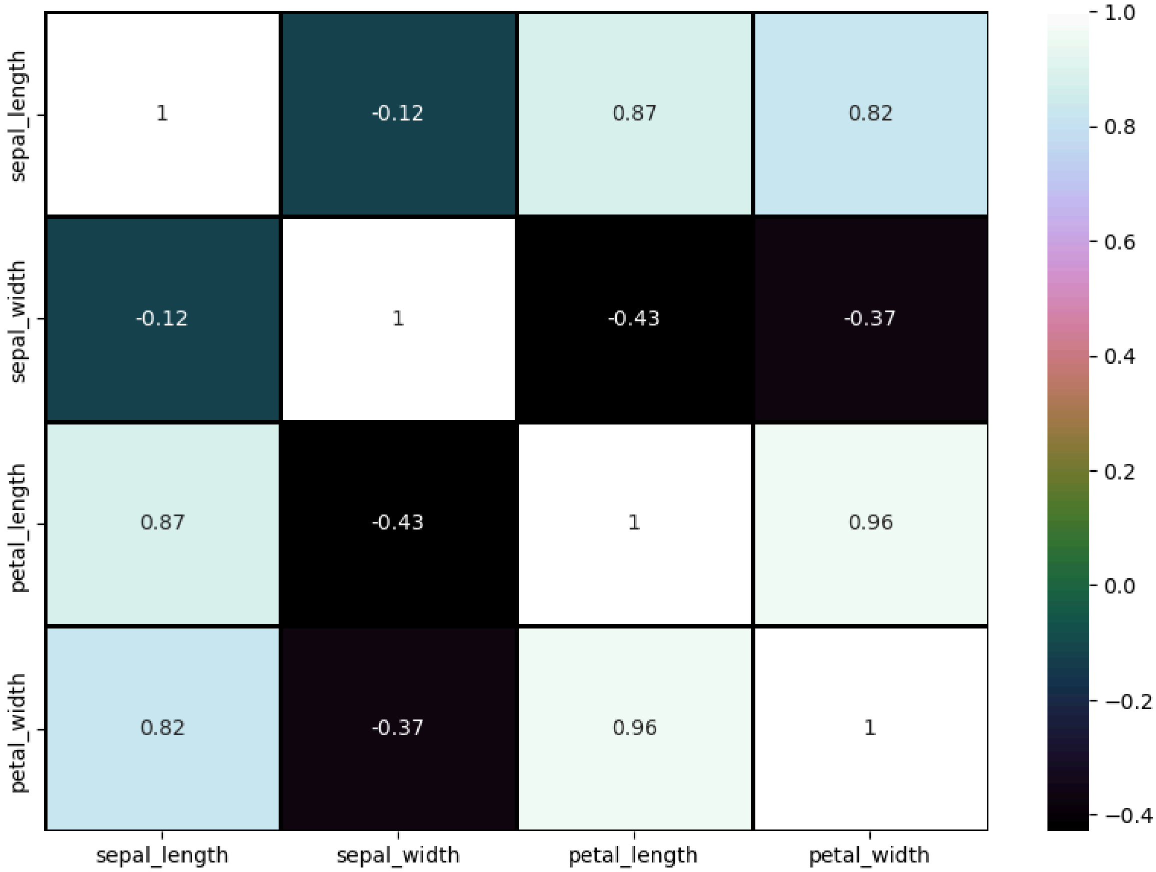
In [32]:
```
iris_numeric = iris.select_dtypes(include='number')
iris_numeric.corr()
```

Out[32]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

In [33]:

```
# Select only numeric columns
iris_numeric = iris.select_dtypes(include='number')

# Create heatmap
plt.figure(figsize=(10,7))
sns.heatmap(iris_numeric.corr(), annot=True, cmap='cubehelix', linewidths=1, lin
plt.show()
```



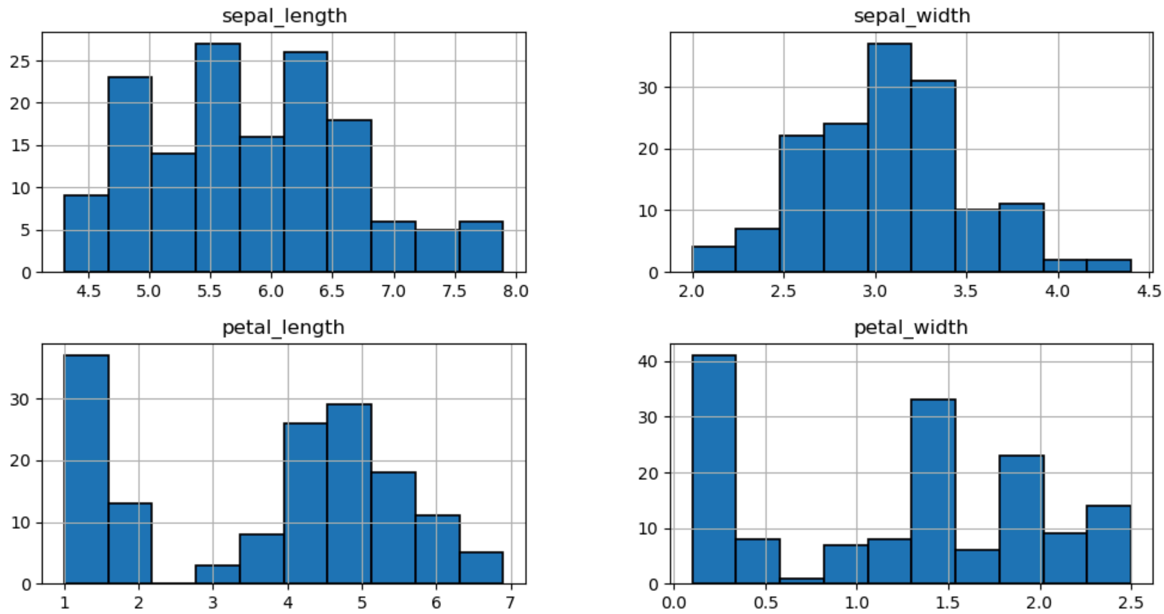
12. Distribution plot

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 [34]:

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

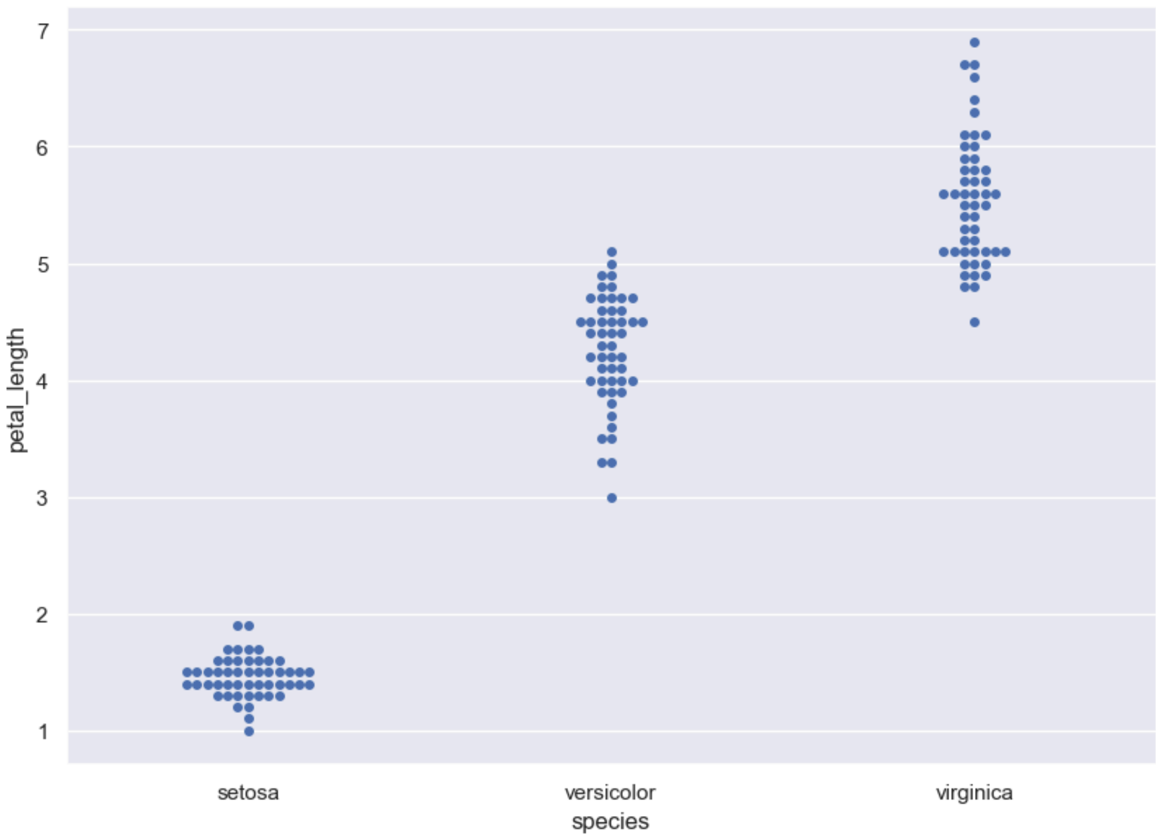
```
fig.set_size_inches(12,6)
plt.show()
```



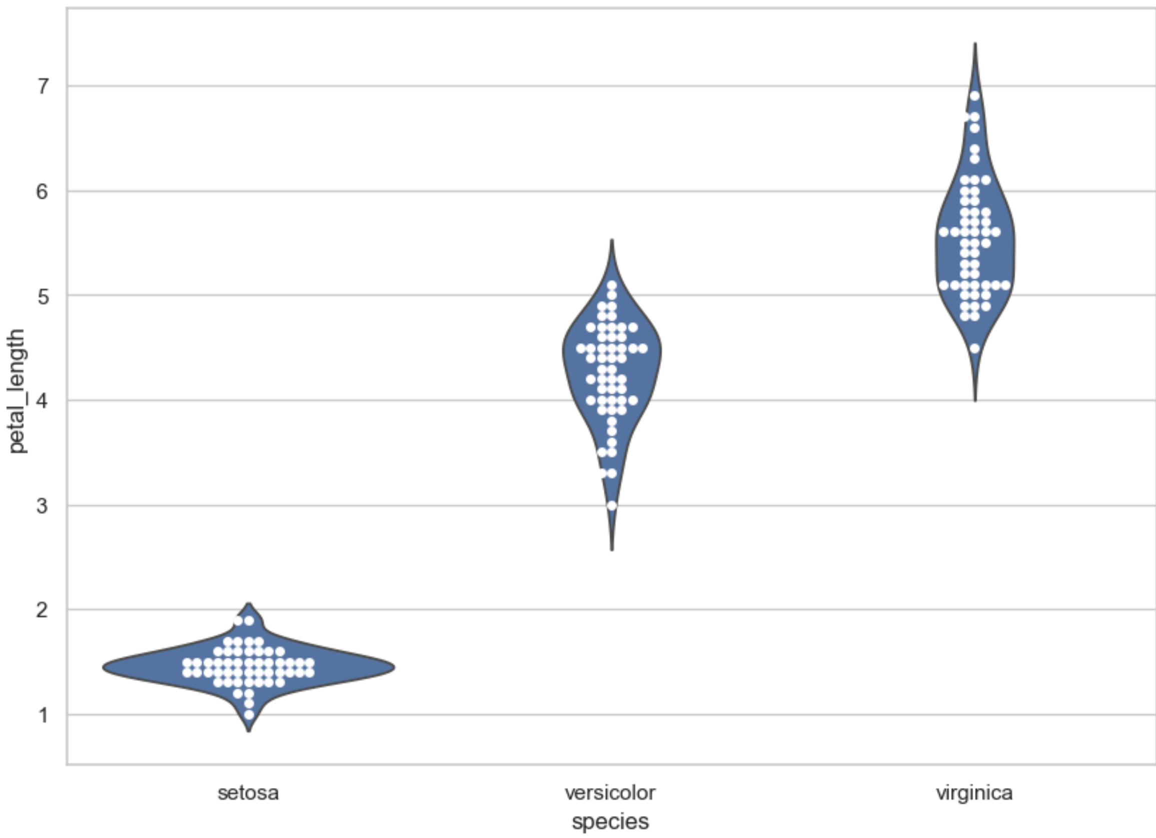
13. Swarm plot

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 [35]:
```
sns.set(style='darkgrid')
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.swarmplot(x='species', y='petal_length', data=iris)
plt.show()
```

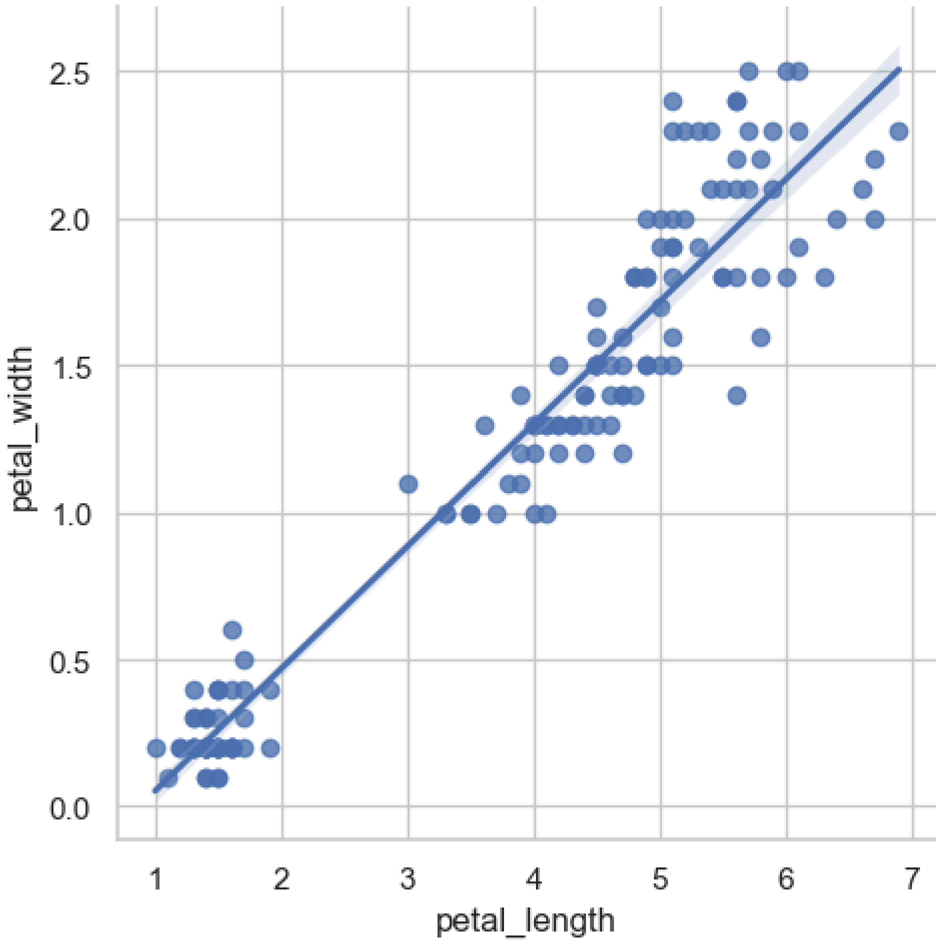


In [36]:
```
sns.set(style='whitegrid')
fig=plt.gcf()
fig.set_size_inches(10,7)
ax = sns.violinplot(x="species", y="petal_length", data=iris, inner=None)
ax = sns.swarmplot(x="species", y="petal_length", data=iris,color="white", edgec
plt.show()
```



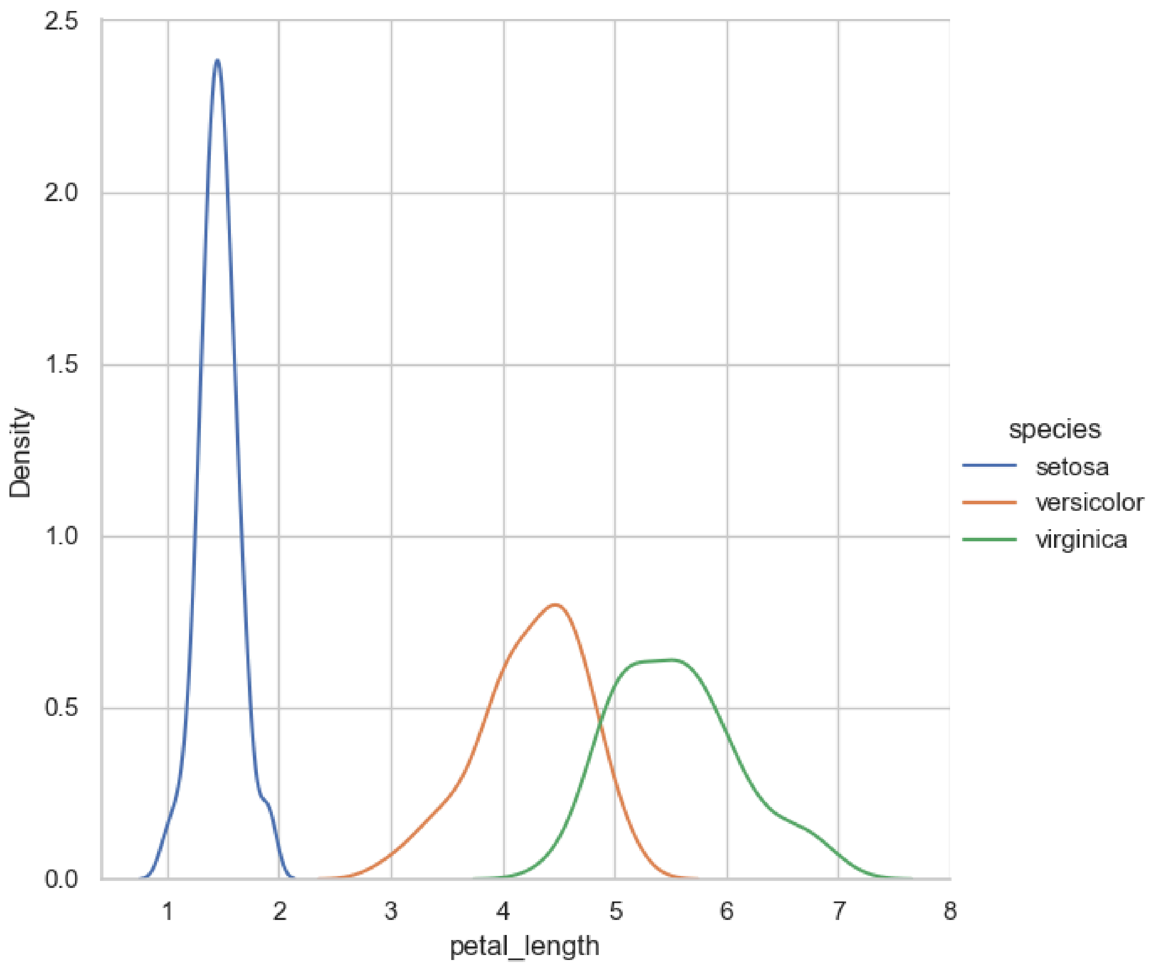
17.LM PLOT

In [37]:
```
fig=sns.lmplot(x='petal_length', y='petal_width',data=iris)
plt.show()
```



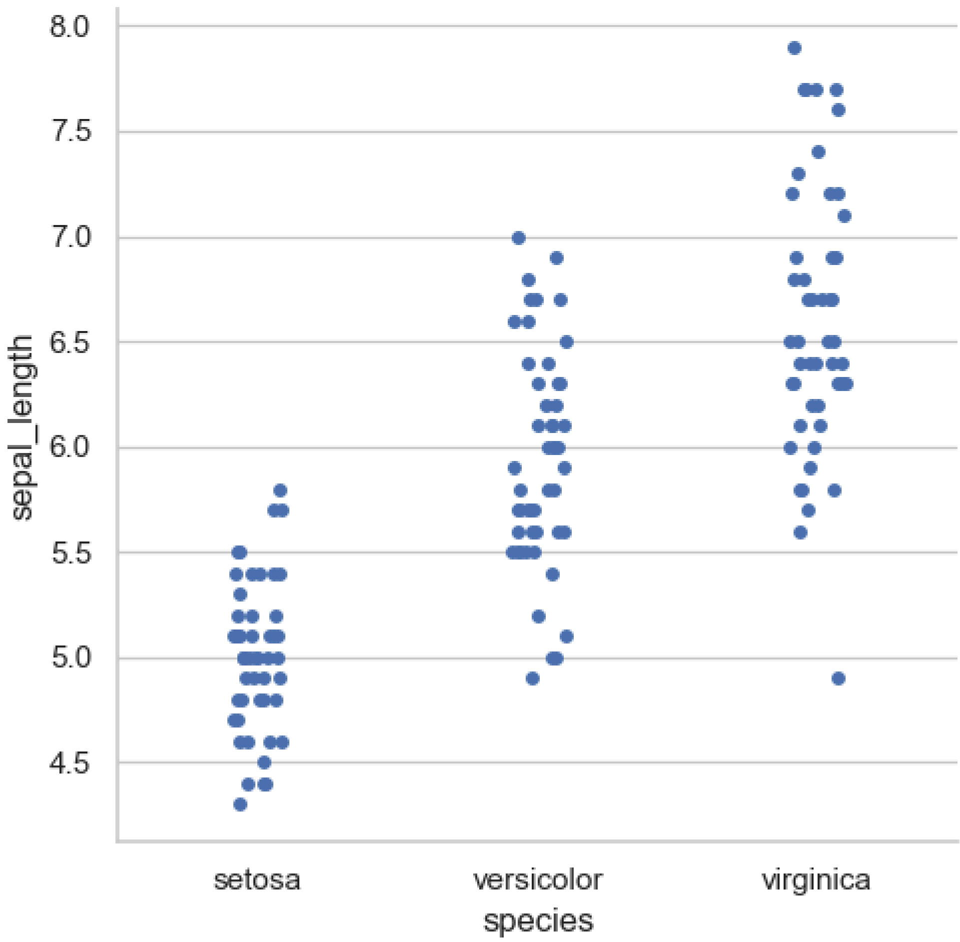
18. FacetGrid

In [38]:
```
sns.FacetGrid(iris, hue='species', height=6)\
   .map(sns.kdeplot , 'petal_length') \
   .add_legend()
plt.show()
```



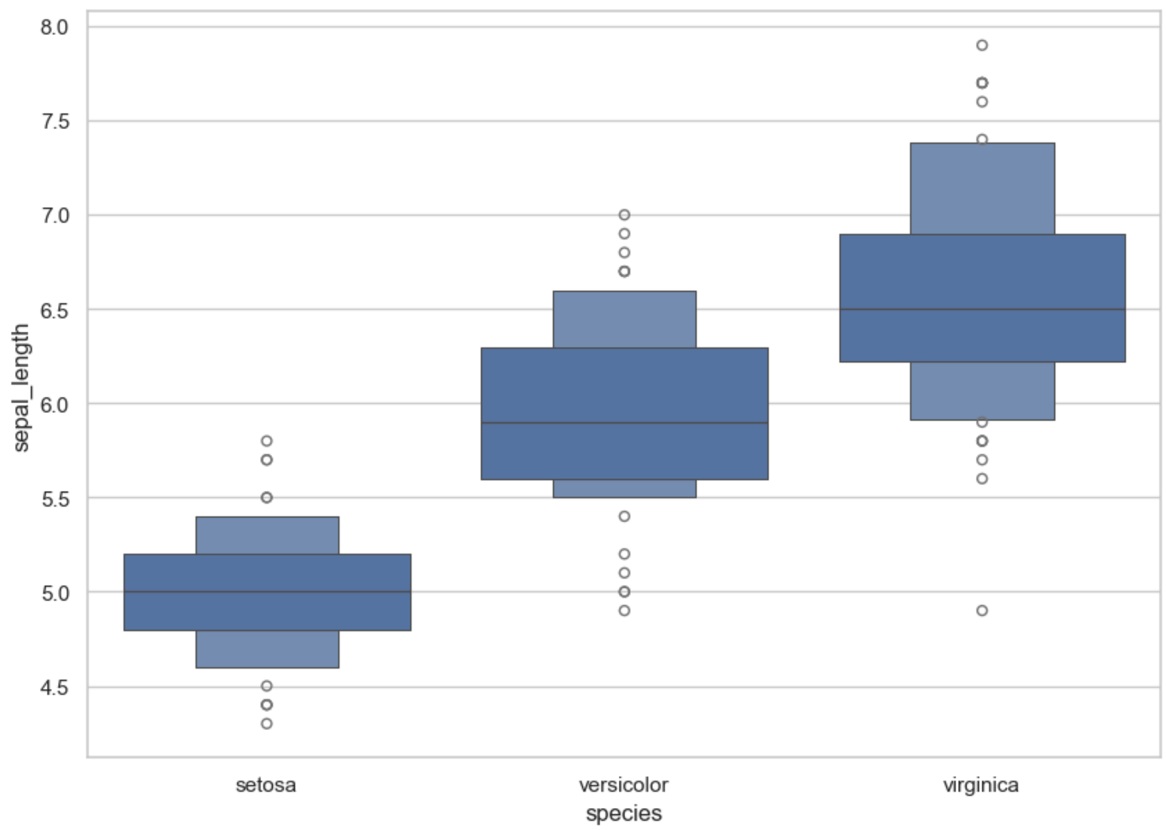
22. Factor Plot

In [39]:
```
# catplot (or) factor plot

sns.catplot(x='species', y='sepal_length', data=iris)
plt.ioff()
plt.show()
```



23 . Boxen Plot

In [40]:
```
fig = plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxenplot(x= 'species' , y='sepal_length', data=iris)
plt.show()
```

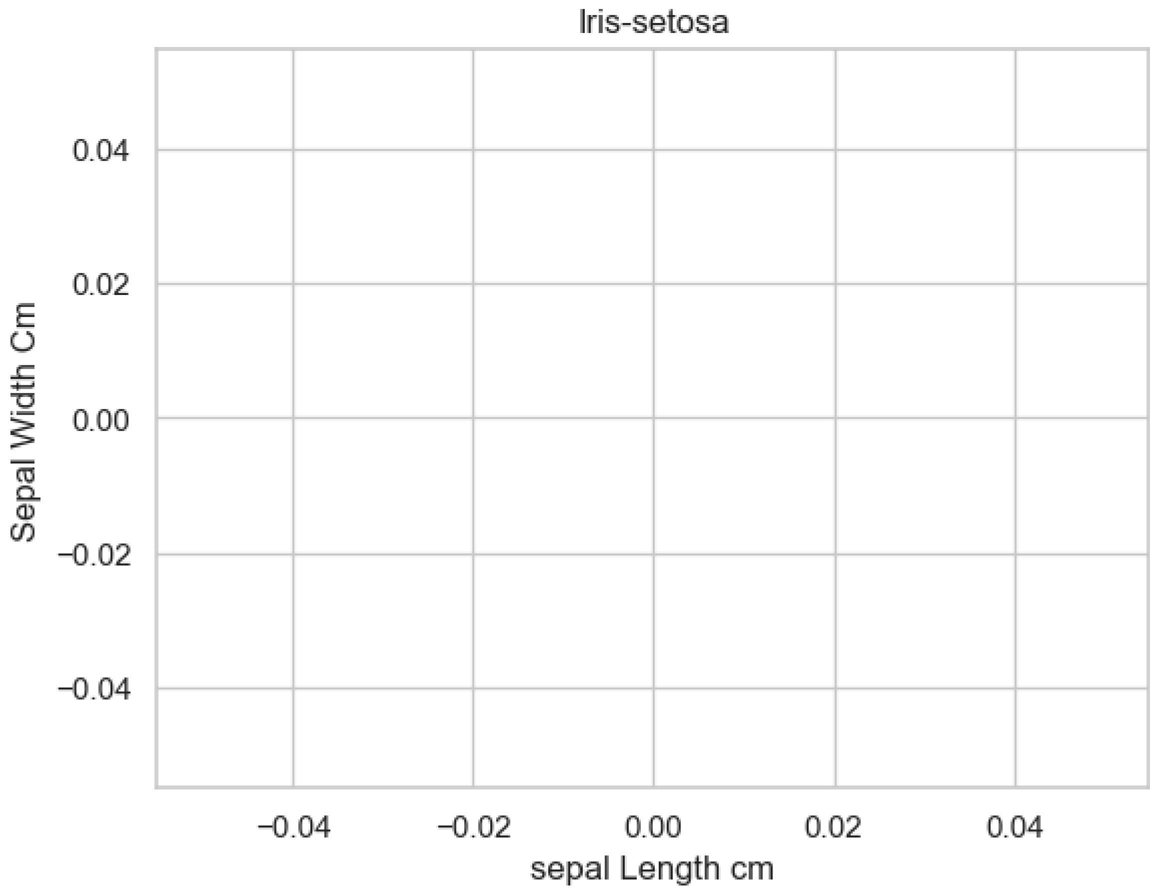


In [41]:
```
print(sns.__version__)
```

```
0.13.2
```

28.KDE Plot

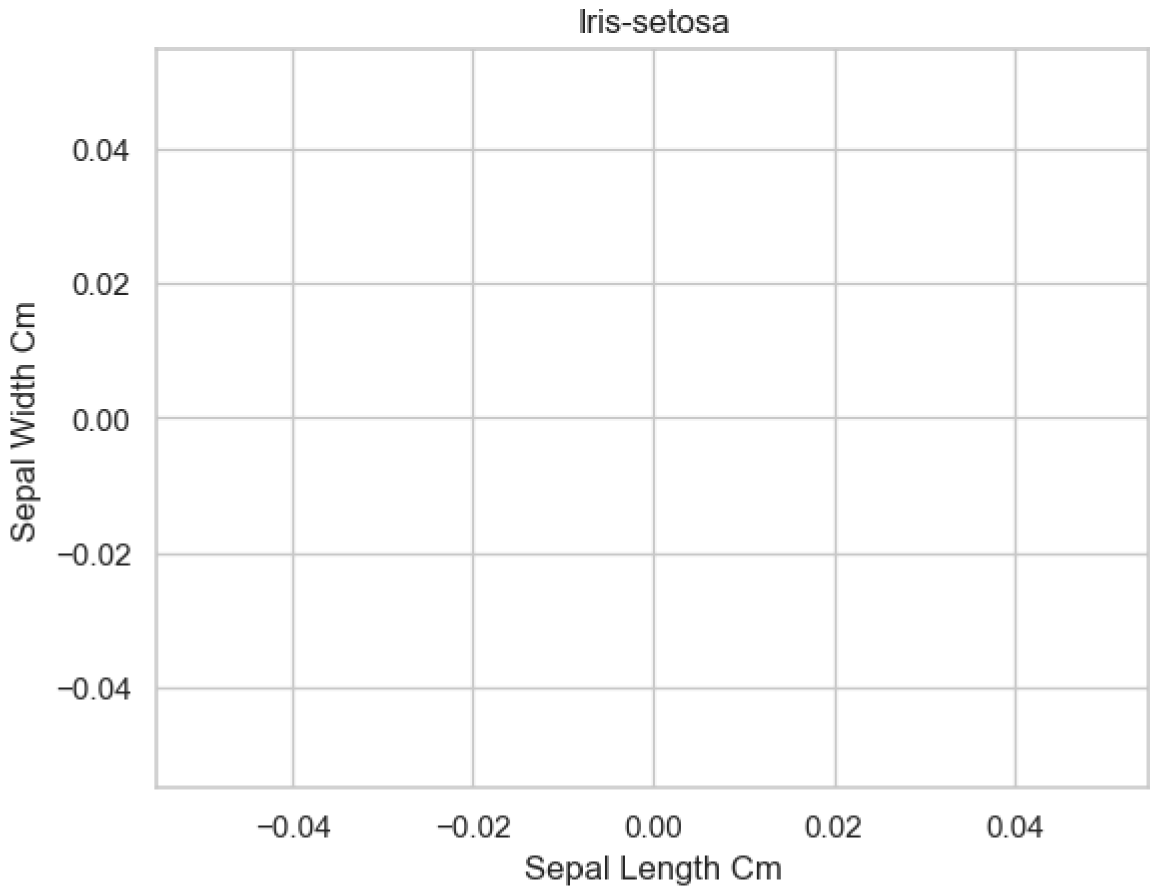
In [43]:
```
sub = iris[iris['species'] == 'Iris-setosa']

sns.kdeplot(
    x=sub['sepal_length'],
    y=sub['sepal_width'],
    cmap="plasma",
    fill=True,
           # instead of shade=True
)

plt.title('Iris-setosa')
plt.xlabel('sepal Length cm')
plt.ylabel('Sepal Width Cm')
plt.show()
```



In [44]:
```
# Create a kde plot of sepal_length versus sepal width for setosa species of flo

sub=iris[iris['species']=='Iris-setosa']
sns.kdeplot(data=sub[['sepal_length','sepal_width']],cmap="plasma", shade=True,
plt.title('Iris-setosa')
plt.xlabel('Sepal Length Cm')
plt.ylabel('Sepal Width Cm')
```

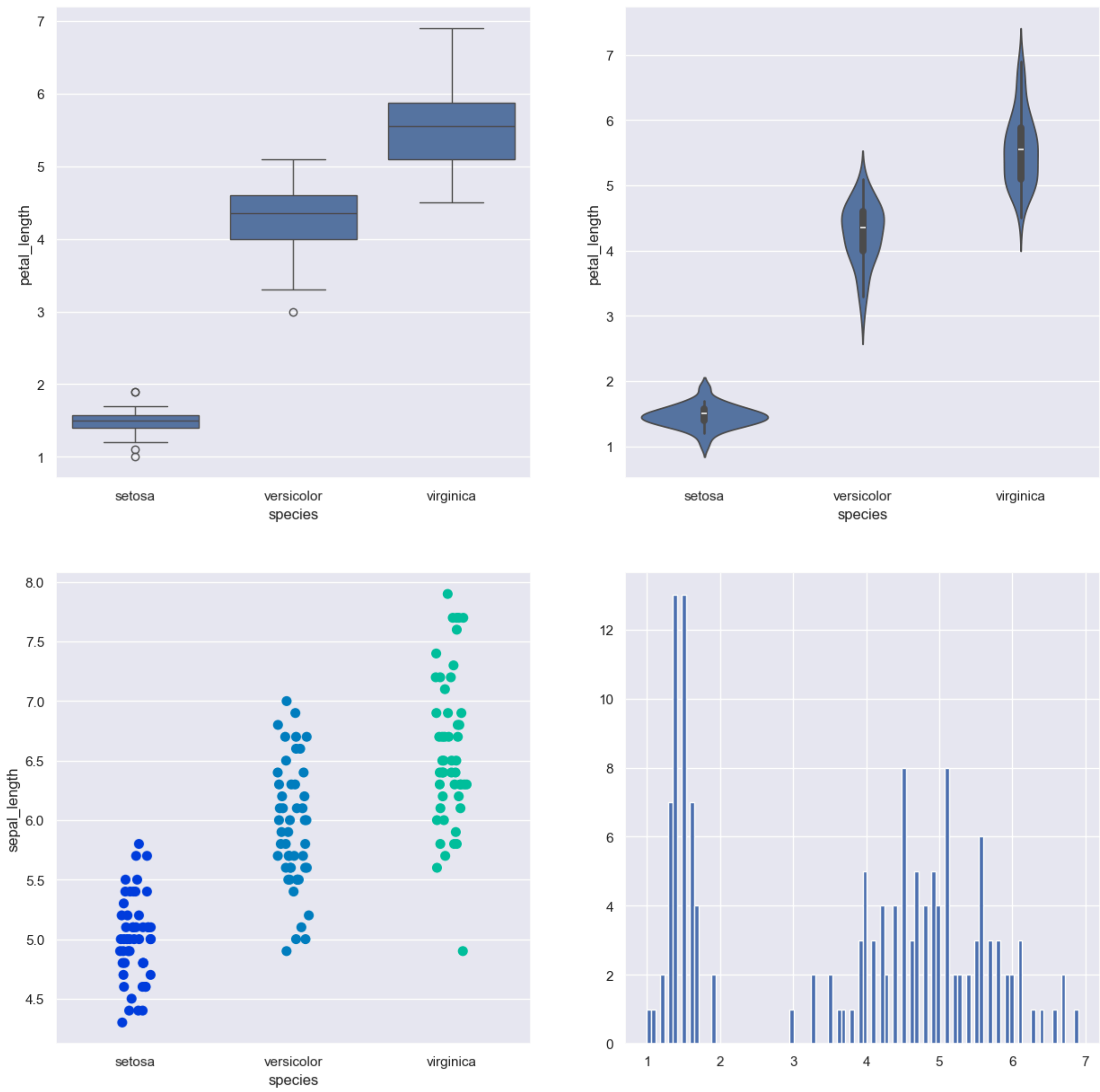
Out[44]:
```
Text(0, 0.5, 'Sepal Width Cm')
```

In [45]:
```
plt.show()
```



30. Dashbord

In [47]:
```
sns.set_style('darkgrid')
f,axes=plt.subplots(2,2,figsize=(15,15))

k1=sns.boxplot(x="species", y="petal_length", data=iris,ax=axes[0,0])
k2=sns.violinplot(x='species',y='petal_length',data=iris,ax=axes[0,1])
k3=sns.stripplot(x='species',y='sepal_length',data=iris,jitter=True,edgecolor='g

#axes[1,1].hist(iris.hist,bin=10)
axes[1,1].hist(iris.petal_length,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

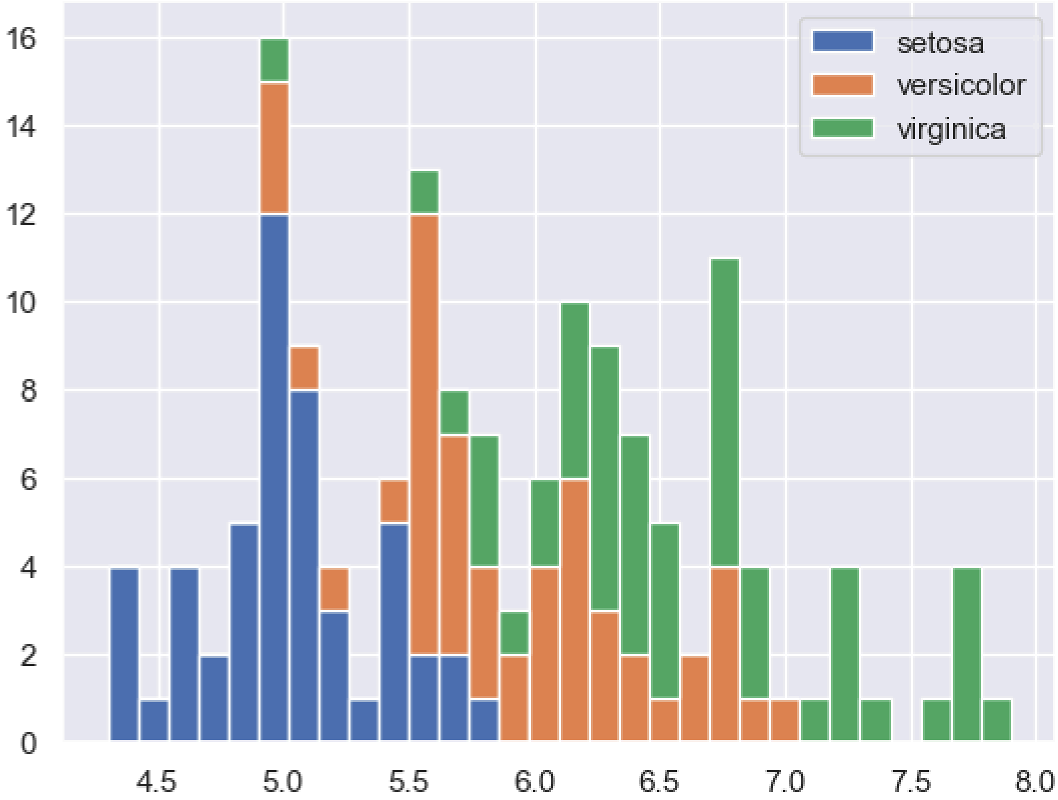
In [48]:
```
iris['species'] = iris['species'].astype('category')
```

In [49]:
```
iris.head()
```

Out[49]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [51]:
```
list1=list()
mylabels=list()
for gen in iris.species.cat.categories:
    list1.append(iris[iris.species==gen].sepal_length)
    mylabels.append(gen)

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

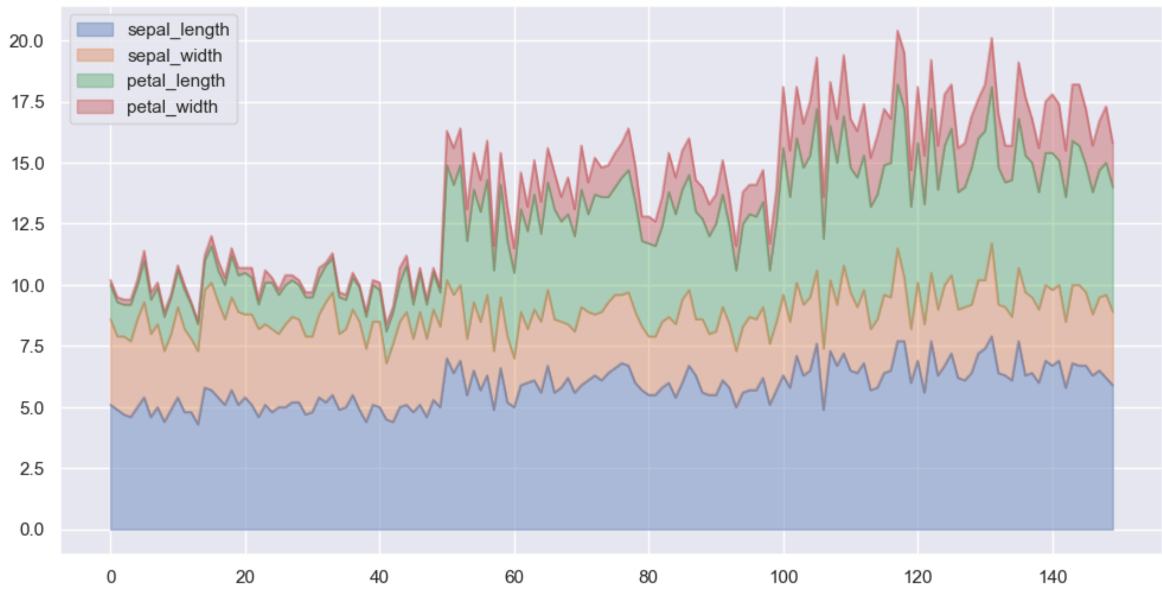


With Stacked Histogram we can see the distribution of Sepal Length of Different Species together.This shows us the range of Sepan Length for the three different Species of Iris Flower.

32. Area Plot

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

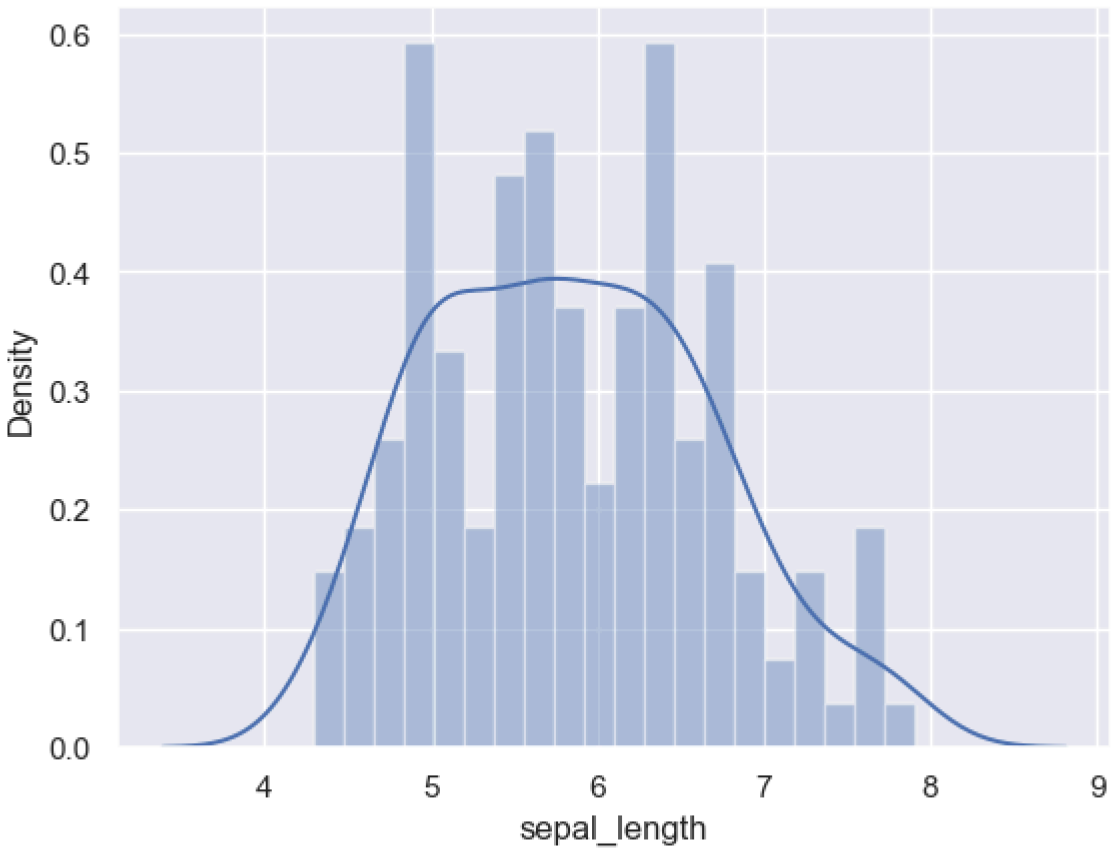
In [53]:
```
#iris['SepalLengthCm'] = iris['SepalLengthCm'].astype('category')
#iris.head()
#iris.plot.area(y='SepalLengthCm',alpha=0.4,figsize=(12, 6));

iris.plot.area(y=['sepal_length','sepal_width','petal_length','petal_width'],alp
plt.show()
```



33. Dist Plot

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

In [54]:
```
sns.distplot(iris['sepal_length'],kde=True,bins=20)
plt.show()
```



EDA complted

In [ ]:
```

```