In [3]:
```
import numpy as np
```

In [4]:
```
import pandas as pd
```

In [5]:
```
import seaborn as sns
```

In [6]:
```
import matplotlib.pyplot as plt
```

In [7]:
```
import warnings
```

In [8]:
```
warnings.filterwarnings('ignore')
```

In [9]:
```
iris=pd.read_csv(r'C:\Users\Admin\Downloads\10th,11th (1)\10th,11th\IRIS DATA
```

In [10]:
```
iris
```

Out[10]:

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

Out[11]:

	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 [12]:
```
iris.drop('Id',axis=1,inplace=True)
```

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

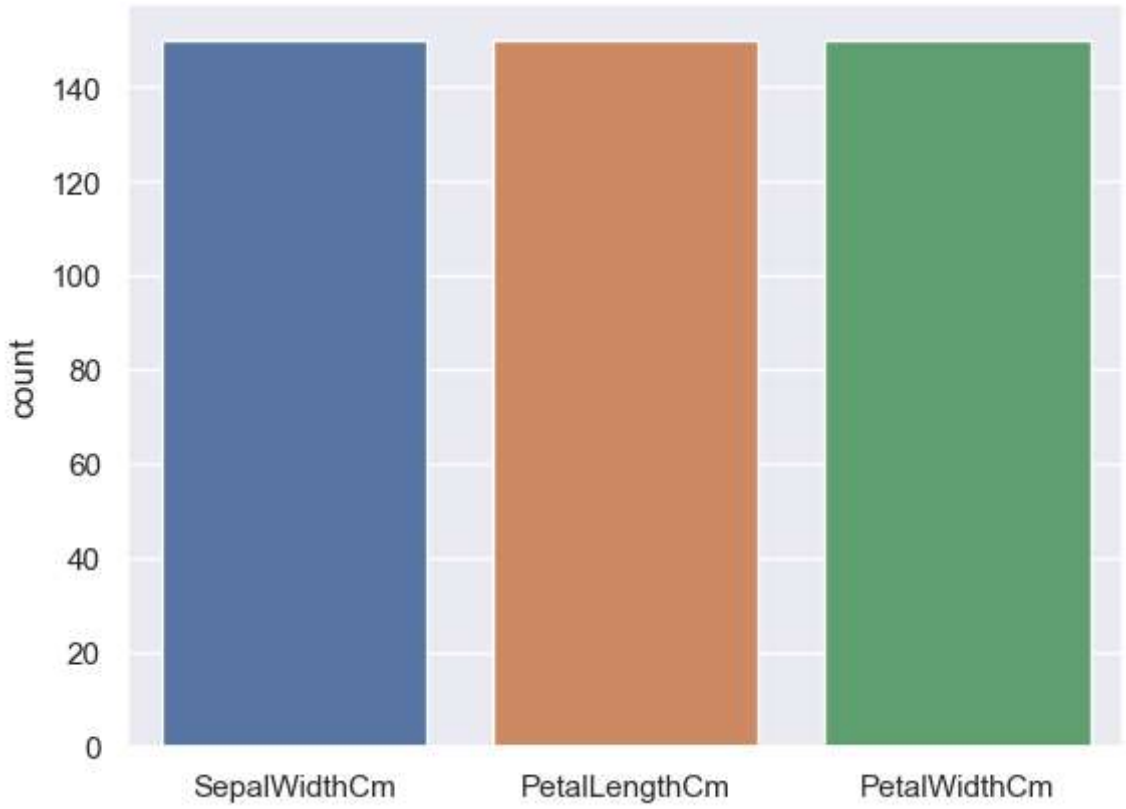
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  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 [15]:
```
iris['Species'].value_counts()
```

Out[15]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [79]:
```
sns.countplot(data=iris)
```

Out[79]: <Axes: ylabel='count'>

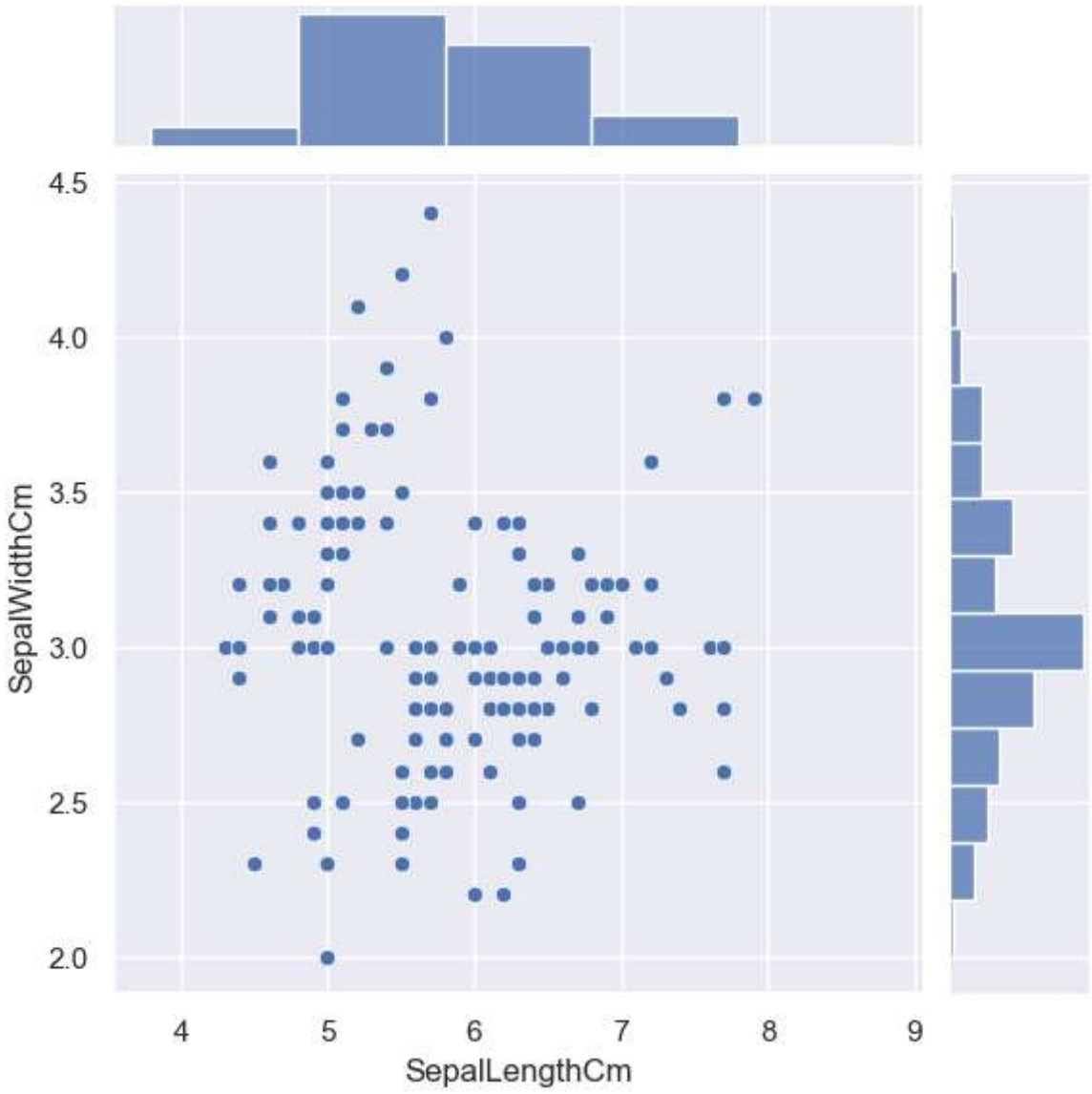


In [80]:
```
iris.head()
```

Out[80]:

	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 [81]:
```
fig=sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',data=iris)
```



** FacetGrid Plot**

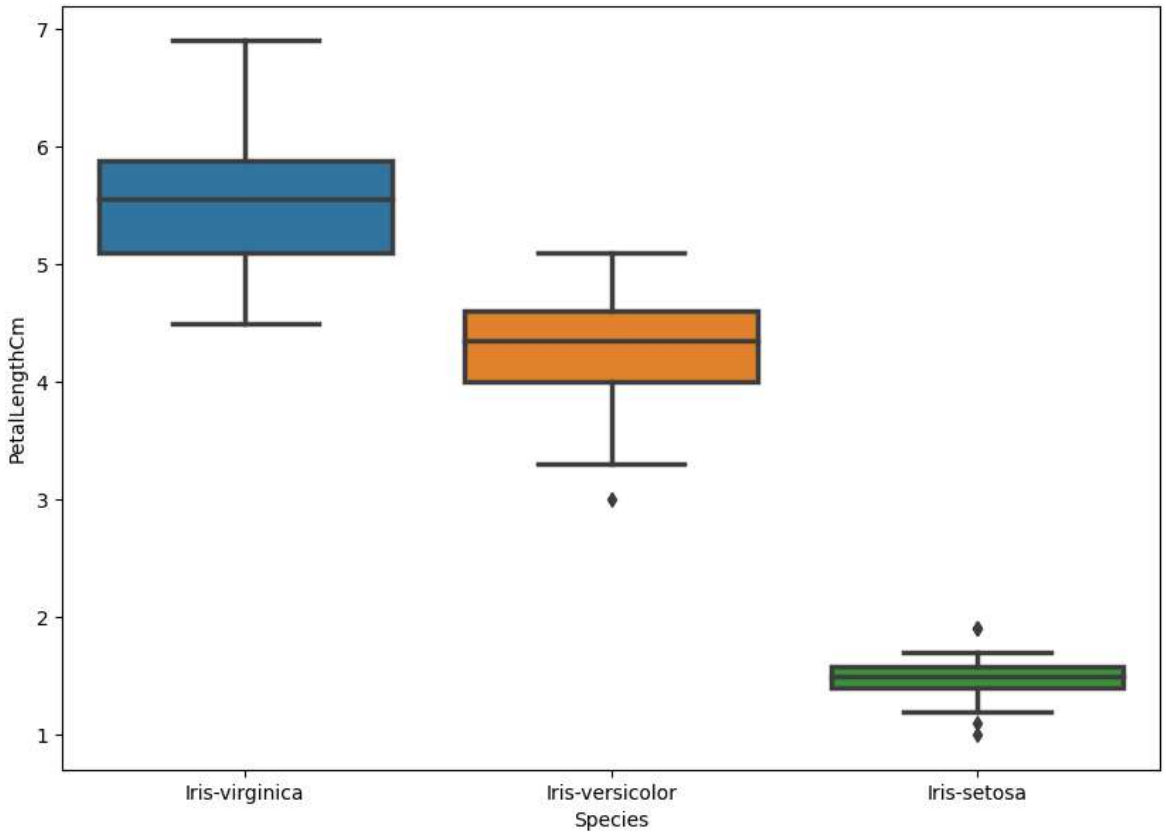
In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline
sns.FacetGrid(iris,hue='Species').map(plt.scatter,'SepalLengthCm','SepalWidth
```

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

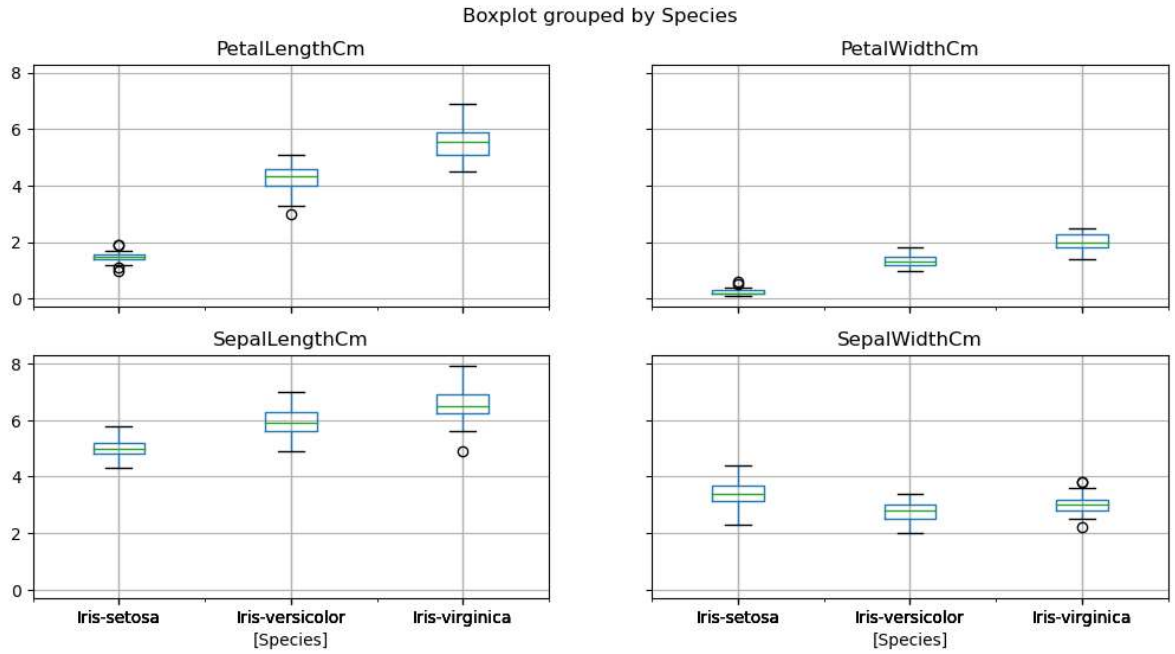
In [ ]:
```
iris.head()
```

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



In [18]:
```
iris.boxplot(by="Species", figsize=(12, 6))
```

Out[18]:
```
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)
```

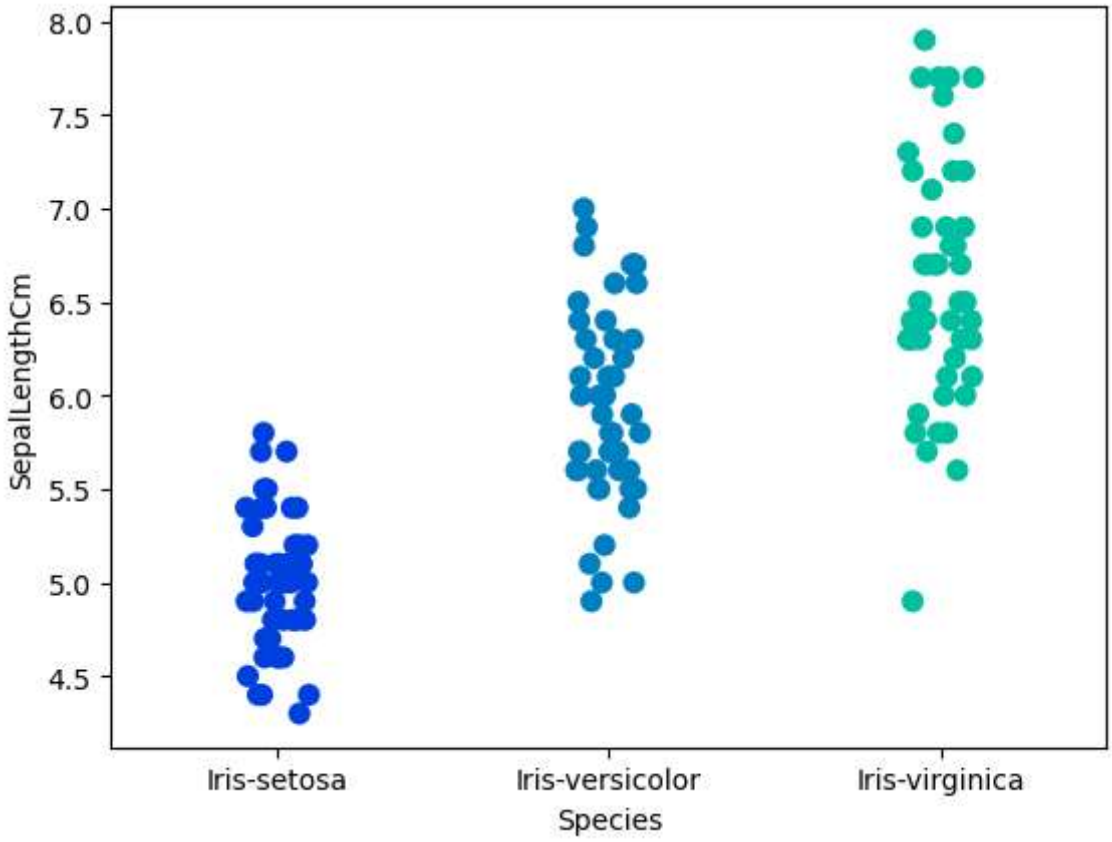


"7. Strip plot"

In [19]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
```

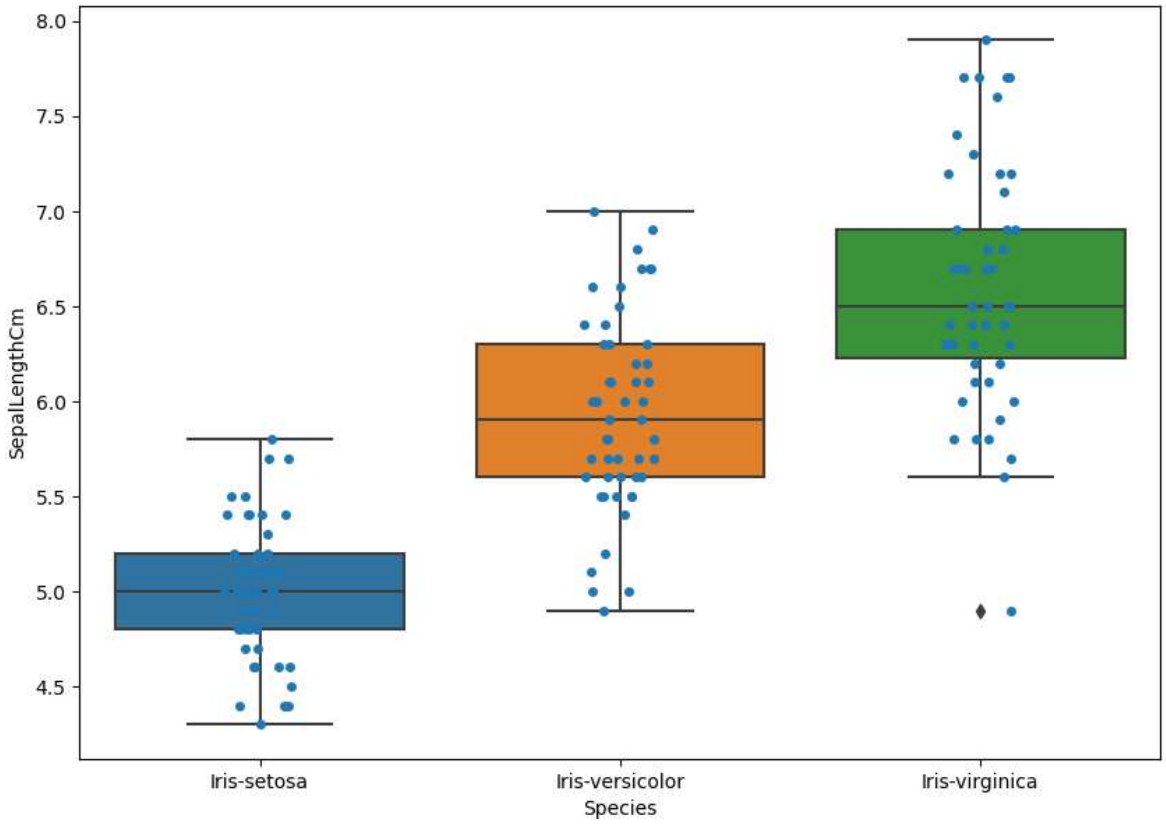
```
<Figure size 1000x700 with 0 Axes>
```

In [20]:
```
fig=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecol
```

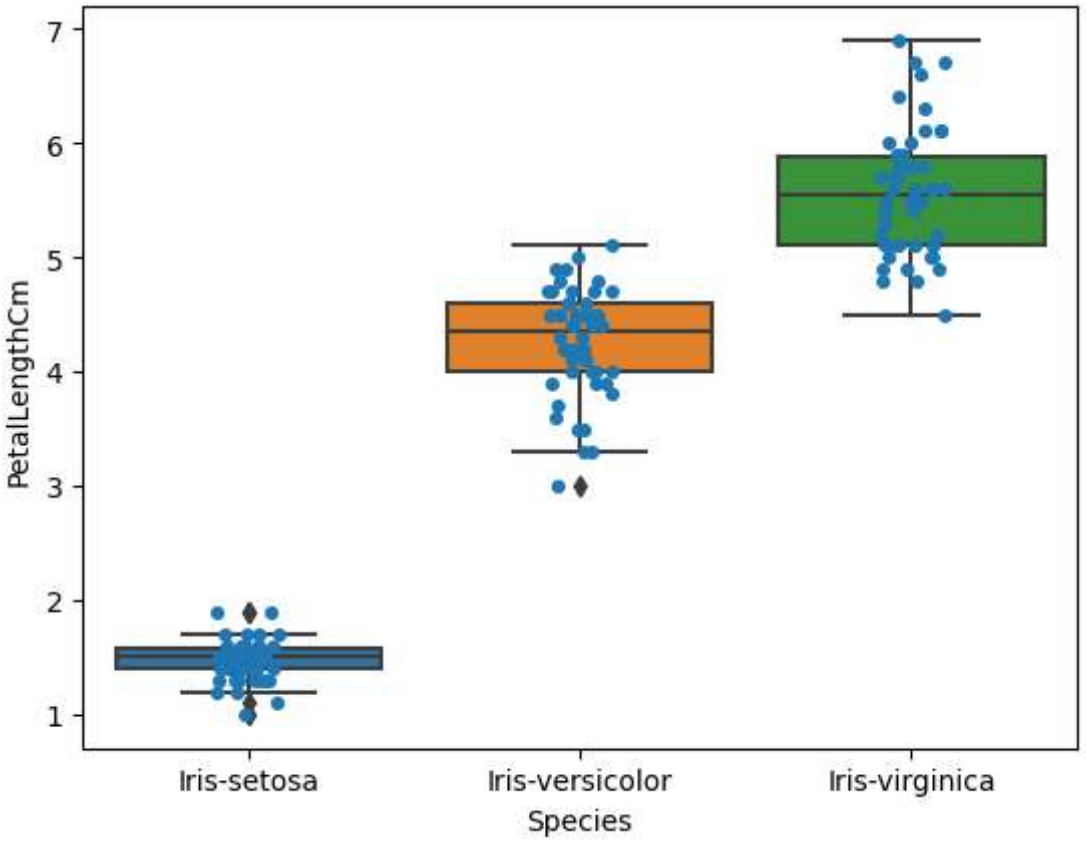


8. Combining Box and Strip Plots

In [21]:
```
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 [22]:
```
ax= sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
ax= sns.stripplot(x="Species", y="PetalLengthCm", data=iris, jitter=True, edg
```



In [23]:
```
# 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')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[23], line 4
      1 # boxtwo = ax.artists[2]
      2 # boxtwo.set_facecolor('yellow')
      3 # boxtwo.set_edgecolor('black')
----> 4 boxthree=ax.artists[1]
      5 boxthree.set_facecolor('red')
      6 boxthree.set_edgecolor('black')

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:1457, in _AxesBa
se.ArtistList.__getitem__(self, key)
   1456 def __getitem__(self, key):
-> 1457     return [artist
   1458             for artist in self._axes._children
   1459             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 [ ]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
```

In [ ]:
```
plt.figure(figsize=(15,10))
```

In [ ]:
```
plt.subplot(2,2,1)
```

In [ ]:
```
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
```

In [ ]:
```
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
```

In [ ]:
```
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
```

In [ ]:
```
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
```

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 t his: Pairs plots are just elaborations on this, showing all variables paired with all the other variables."

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

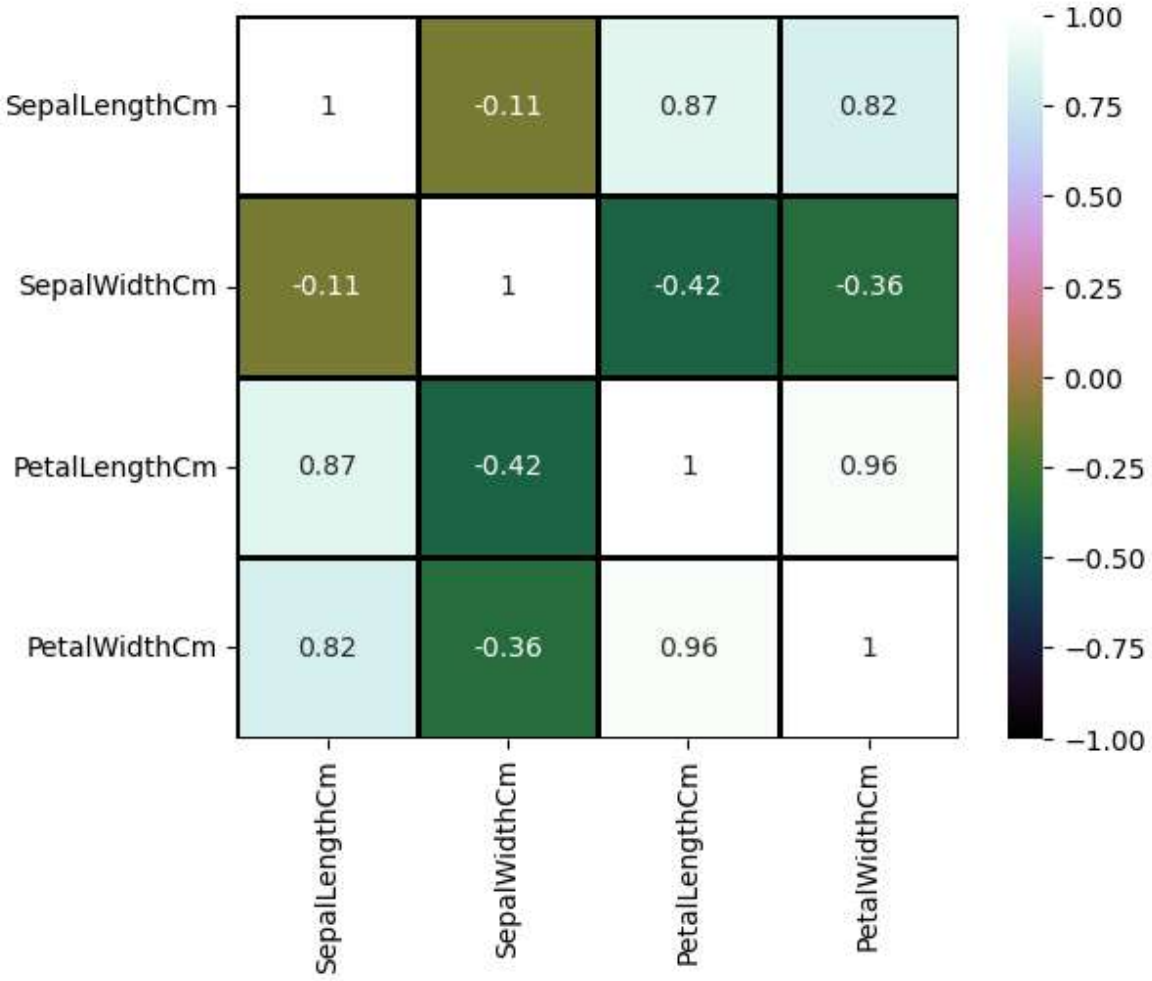
In [ ]:
```
sns.pairplot(iris,hue='Species')
```

"11. Heat map\n",

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

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

In [24]:
```
fig=sns.heatmap(iris.corr(),annot=True,cmap='cubehelix',linewidths=1,linecolo
```

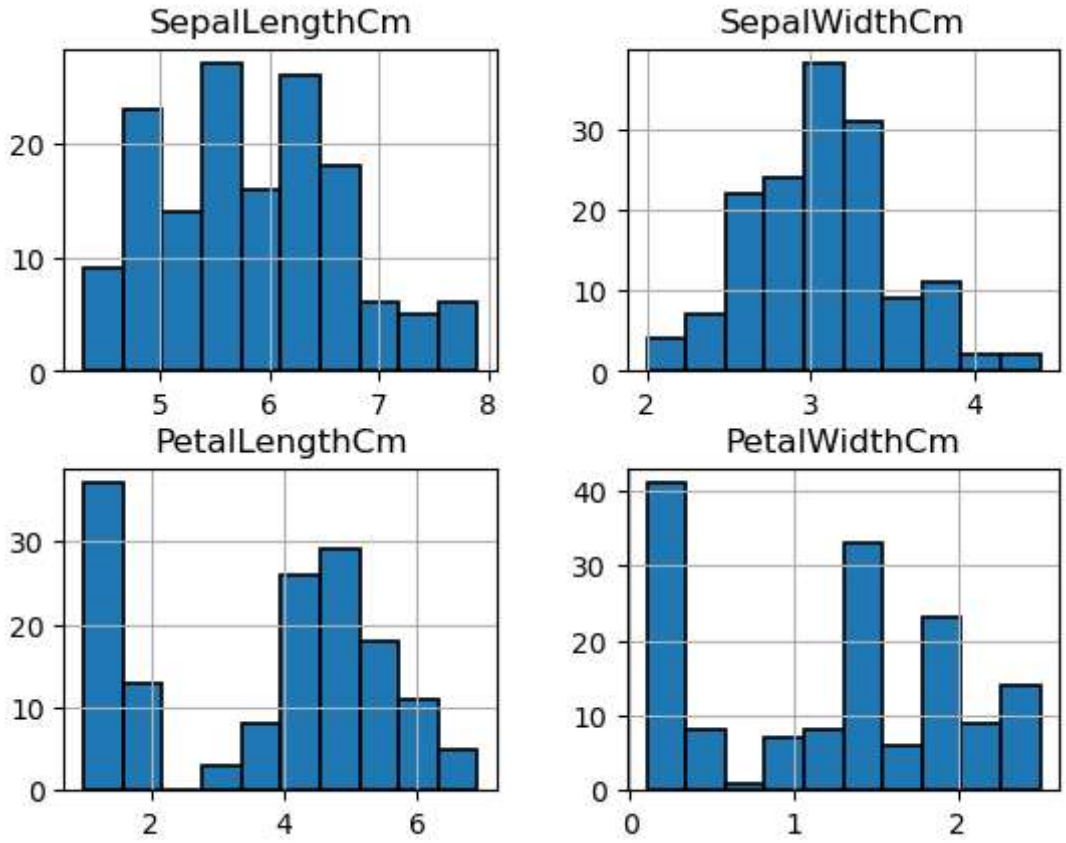


"**12. Distribution plot:**\n",

"The distribution plot is suitable for comparing range and distributi
on for groups of numerical data. Data is plotted as value points alon
g 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 rele
vant for detailed analysis of the data as it deals with a summary of
the data distribution."

In [25]:
```
iris.hist(edgecolor='black', linewidth=1.2)
```

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



In [26]:
```
fig=plt.gcf()
fig.set_size_inches(12,6)
```

```
<Figure size 1200x600 with 0 Axes>
```

"13. Swarm plot\n",

"It looks a bit like a friendly swarm of bees buzzing about their hiv e. More importantly, each data point is clearly visible and no data a re 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"

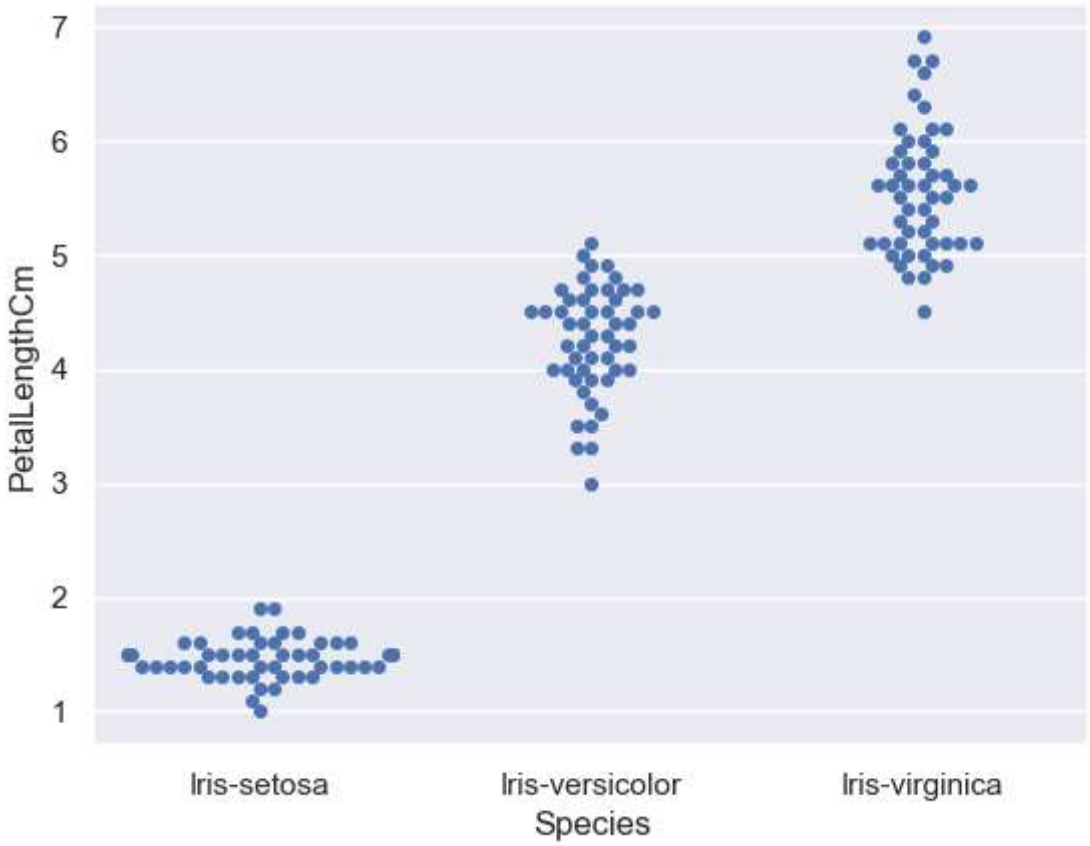
In [27]:
```
sns.set(style="darkgrid")
```

In [28]:
```
fig=plt.gcf()
```

```
<Figure size 640x480 with 0 Axes>
```

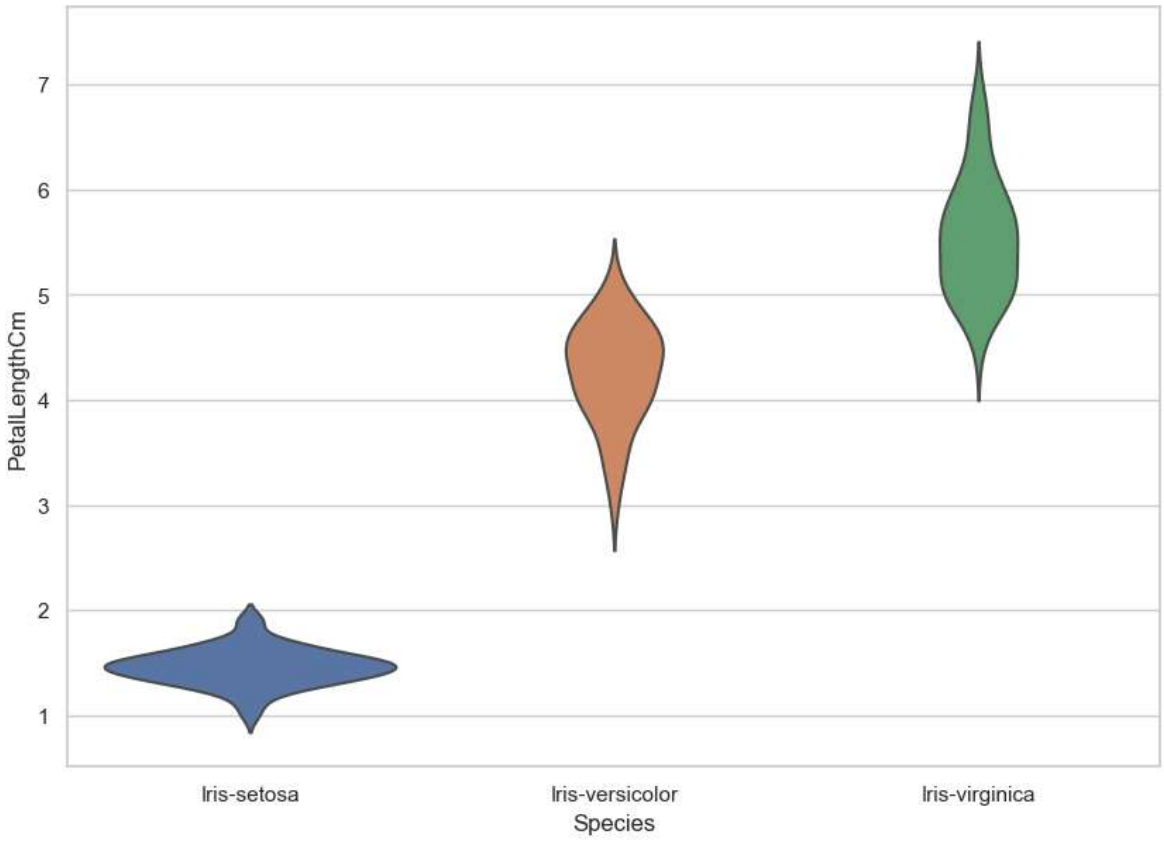
In [29]:
```
fig.set_size_inches(10,7)
```

In [30]:
```
fig = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris)
```

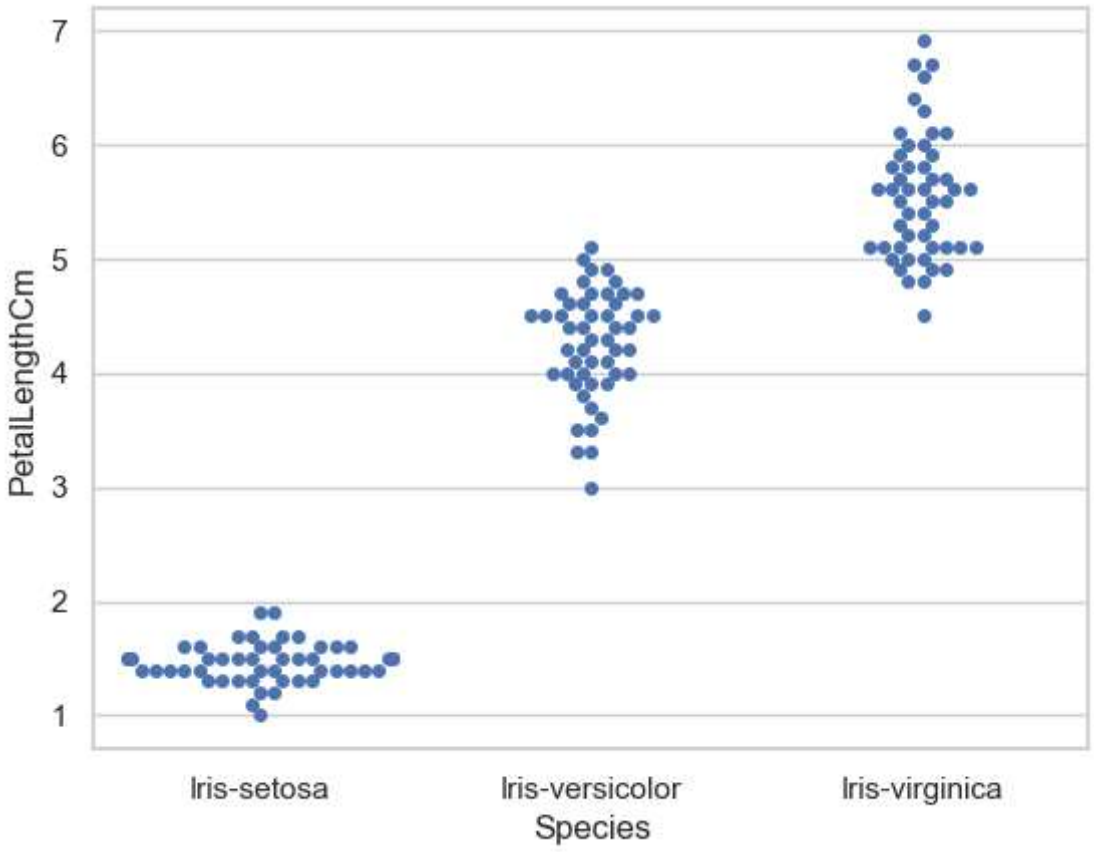


In [31]:
```
sns.set(style="whitegrid")
```

In [32]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
ax = sns.violinplot(x="Species", y="PetalLengthCm", data=iris, inner=None)
```

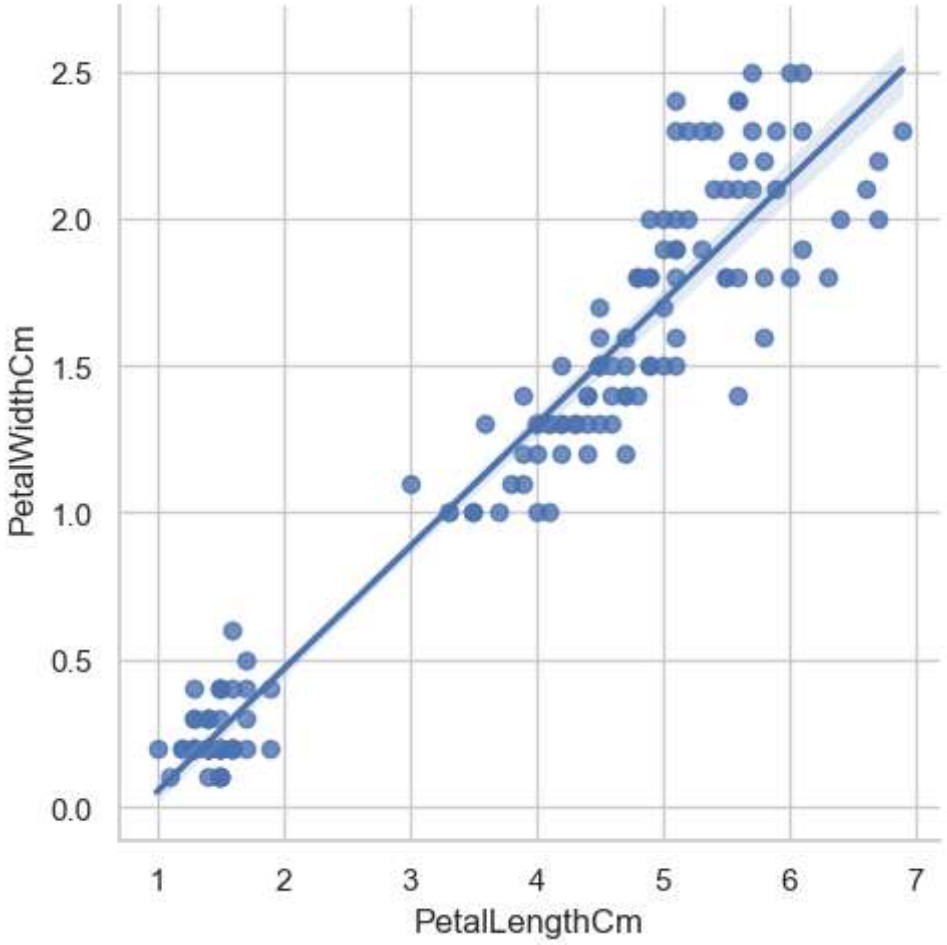


In [33]:
```
ax = sns.swarmplot(x="Species", y="PetalLengthCm", data=iris, edgecolor="blac
```



"17. LM PLot"

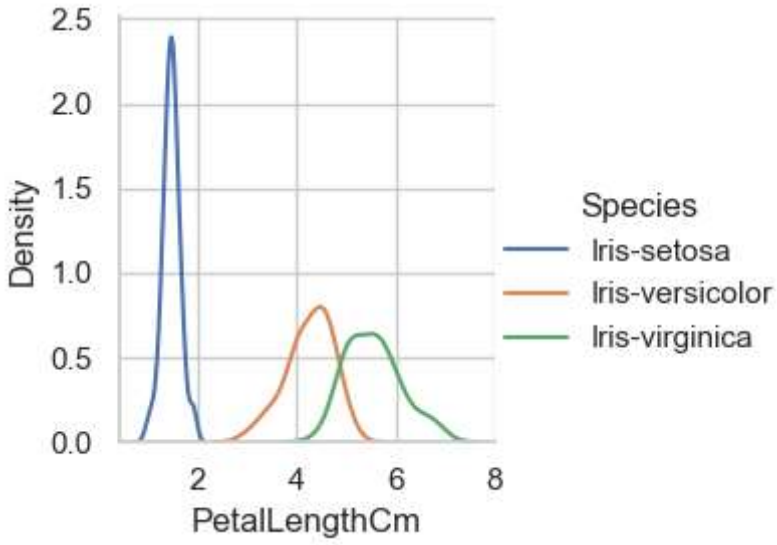
In [34]:
```
fig=sns.lmplot(x="PetalLengthCm", y="PetalWidthCm",data=iris)
```



"18. FacetGrid"

In [35]:
```
sns.FacetGrid(iris, hue="Species").map(sns.kdeplot, "PetalLengthCm").add_lege
```

Out[35]:
```
<seaborn.axisgrid.FacetGrid at 0x2459e700760>
```



In [36]:
```
plt.ion()
```

Out[36]:
```
<contextlib.ExitStack at 0x2459ec7dd80>
```

"** 22. Factor Plot **"

In [37]:
```
sns.factorplot('Species','SepalLengthCm',data=iris)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[37], line 1
----> 1 sns.factorplot('Species','SepalLengthCm',data=iris)

AttributeError: module 'seaborn' has no attribute 'factorplot'
```

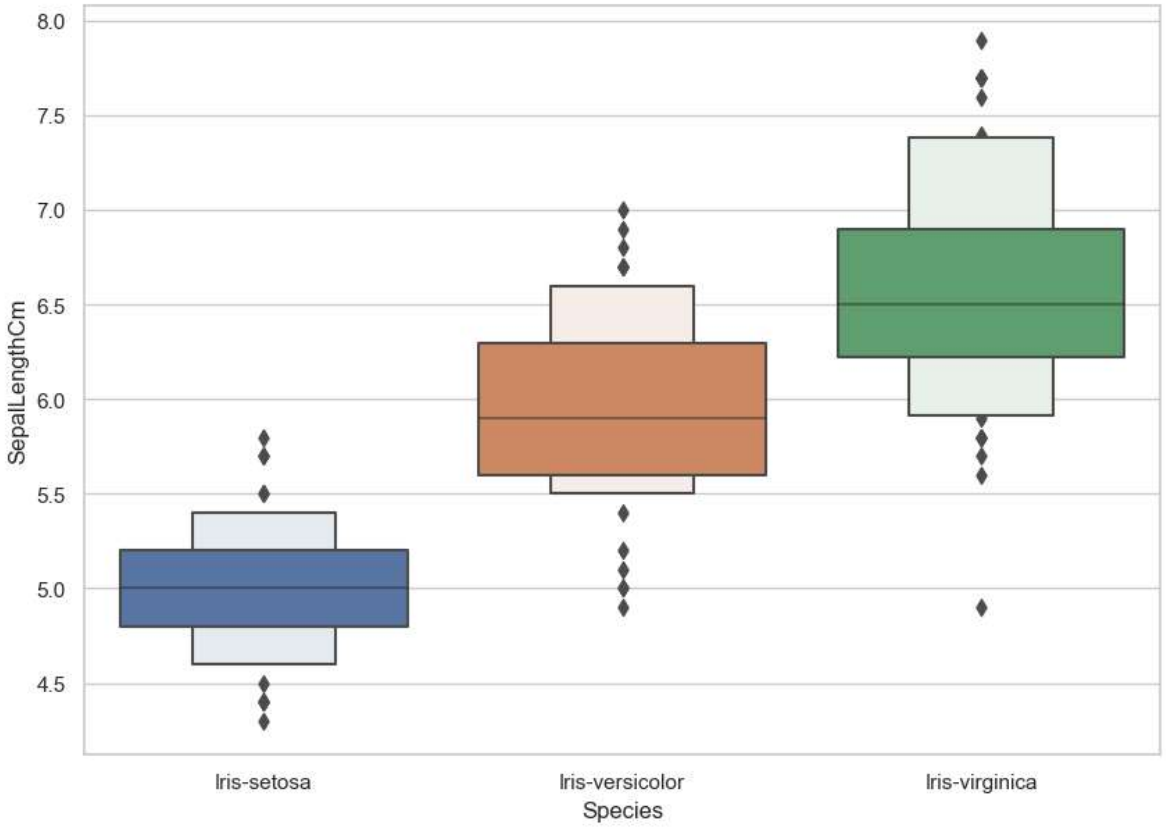
In [ ]:
```
plt.ioff()
plt.show()
```

In [38]:
```
sns.factorplot('Species','SepalLengthCm',data=iris,ax=ax[0][0])
```

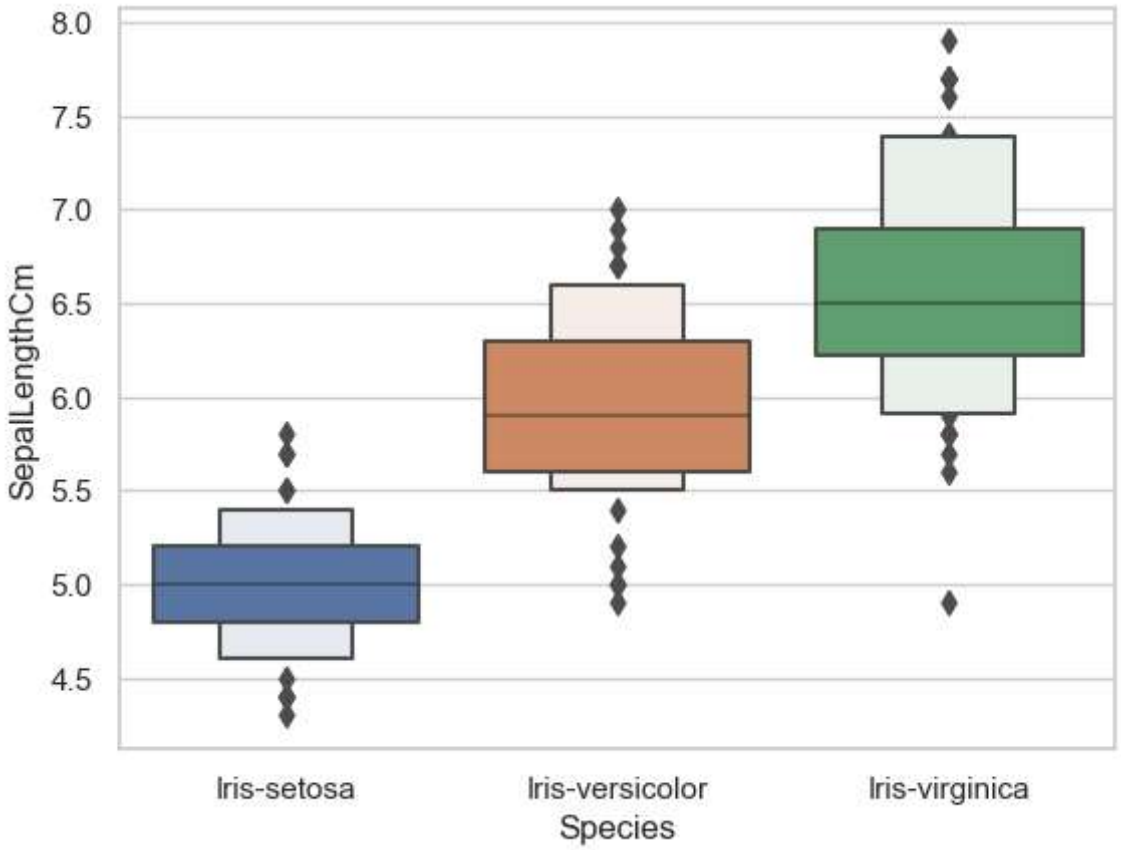
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[38], line 1
----> 1 sns.factorplot('Species','SepalLengthCm',data=iris,ax=ax[0][0])

AttributeError: module 'seaborn' has no attribute 'factorplot'
```

"** 23. Boxen Plot**"

In [39]:
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.boxenplot(x='Species',y='SepalLengthCm',data=iris)
```



In [40]:
```
fig=sns.boxenplot(x='Species',y='SepalLengthCm',data=iris)
```



"**28.KDE Plot **"

In [41]:
```
# Create a kde plot of sepal_length versus sepal width for setosa species of
sub=iris[iris['Species']=='Iris-setosa']
```

In [42]:

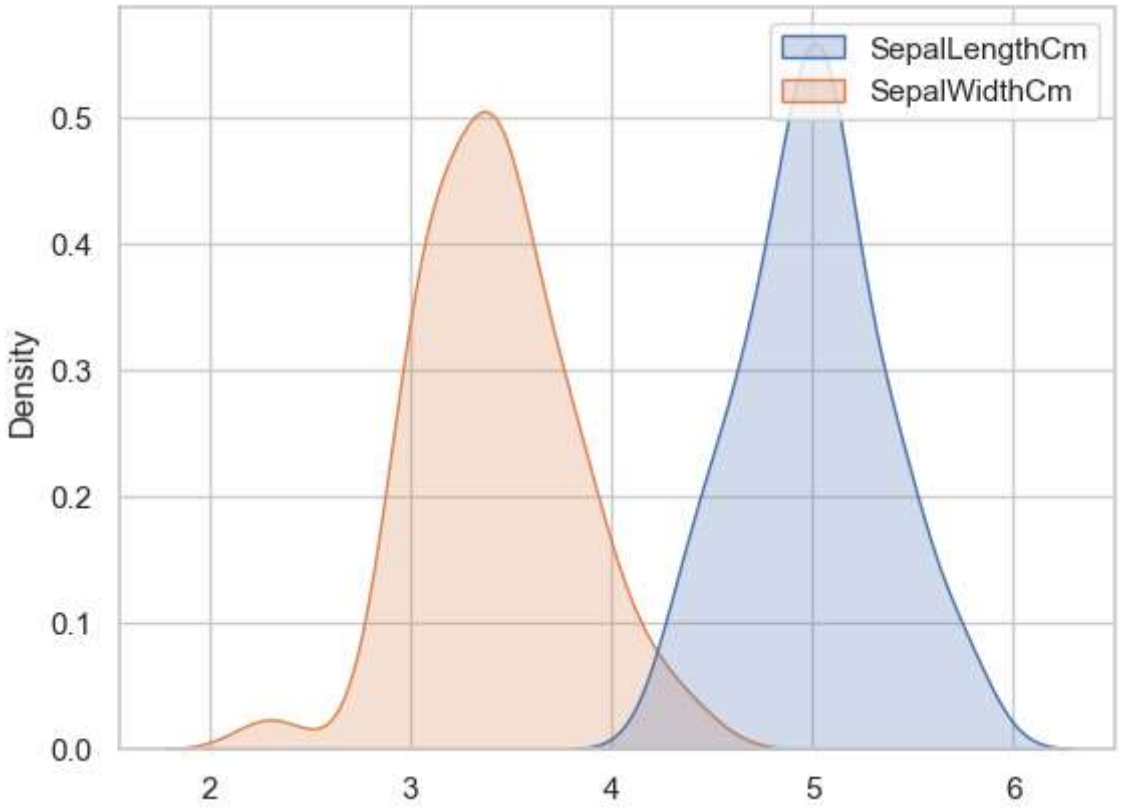
Out[42]:

	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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
30	4.8	3.1	1.6	0.2	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
34	4.9	3.1	1.5	0.1	Iris-setosa
35	5.0	3.2	1.2	0.2	Iris-setosa

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
36	5.5	3.5	1.3	0.2	Iris-setosa
37	4.9	3.1	1.5	0.1	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
40	5.0	3.5	1.3	0.3	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5.0	3.5	1.6	0.6	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
45	4.8	3.0	1.4	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5.0	3.3	1.4	0.2	Iris-setosa

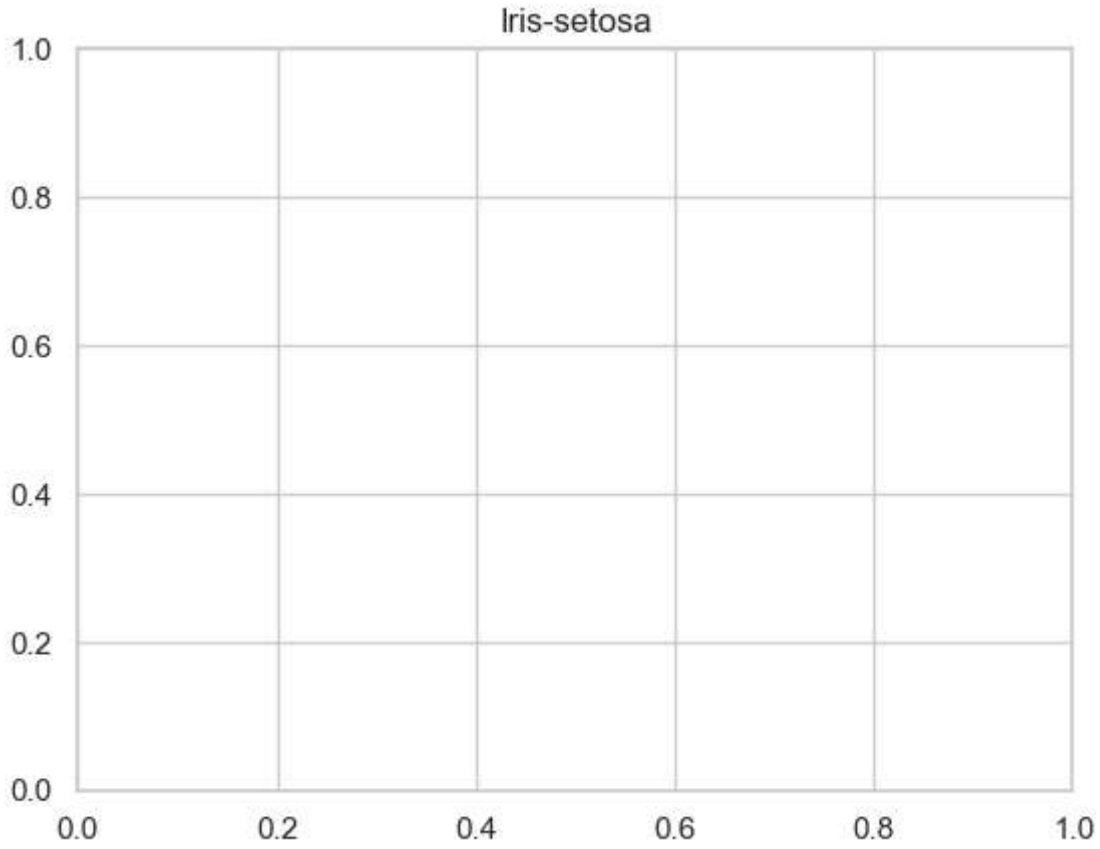
In [43]:
```
sns.kdeplot(data=sub[['SepalLengthCm','SepalWidthCm']], shade=True, shade_low
```

Out[43]: <Axes: ylabel='Density'>



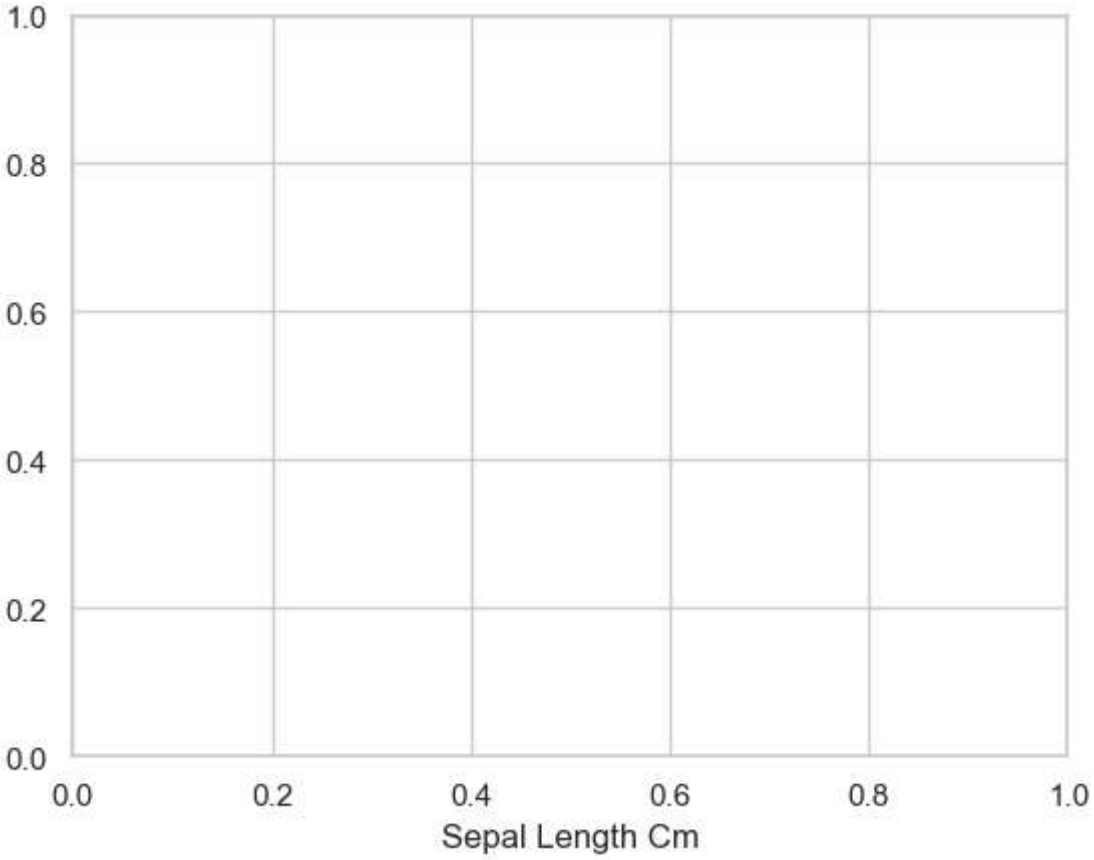
In [44]:
```
plt.title('Iris-setosa')
```

Out[44]:
```
Text(0.5, 1.0, 'Iris-setosa')
```



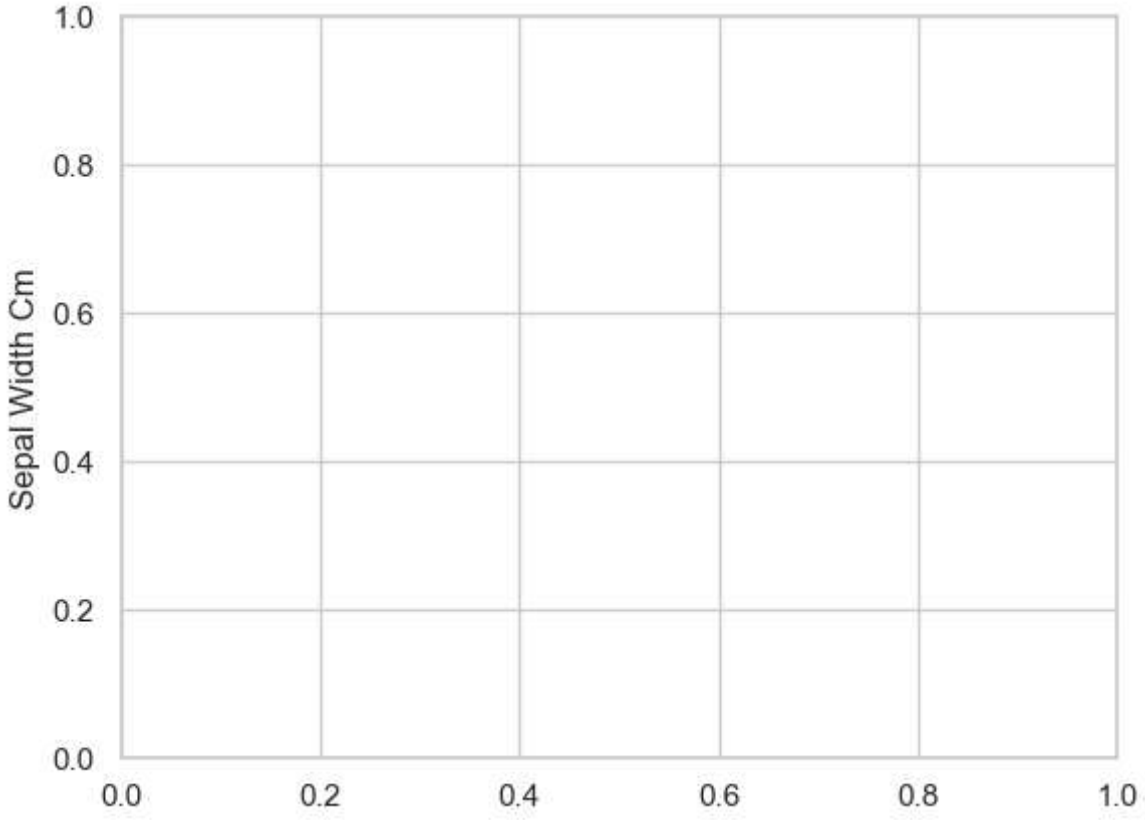
In [45]:
```
plt.xlabel('Sepal Length Cm')
```

Out[45]:
```
Text(0.5, 0, 'Sepal Length Cm')
```



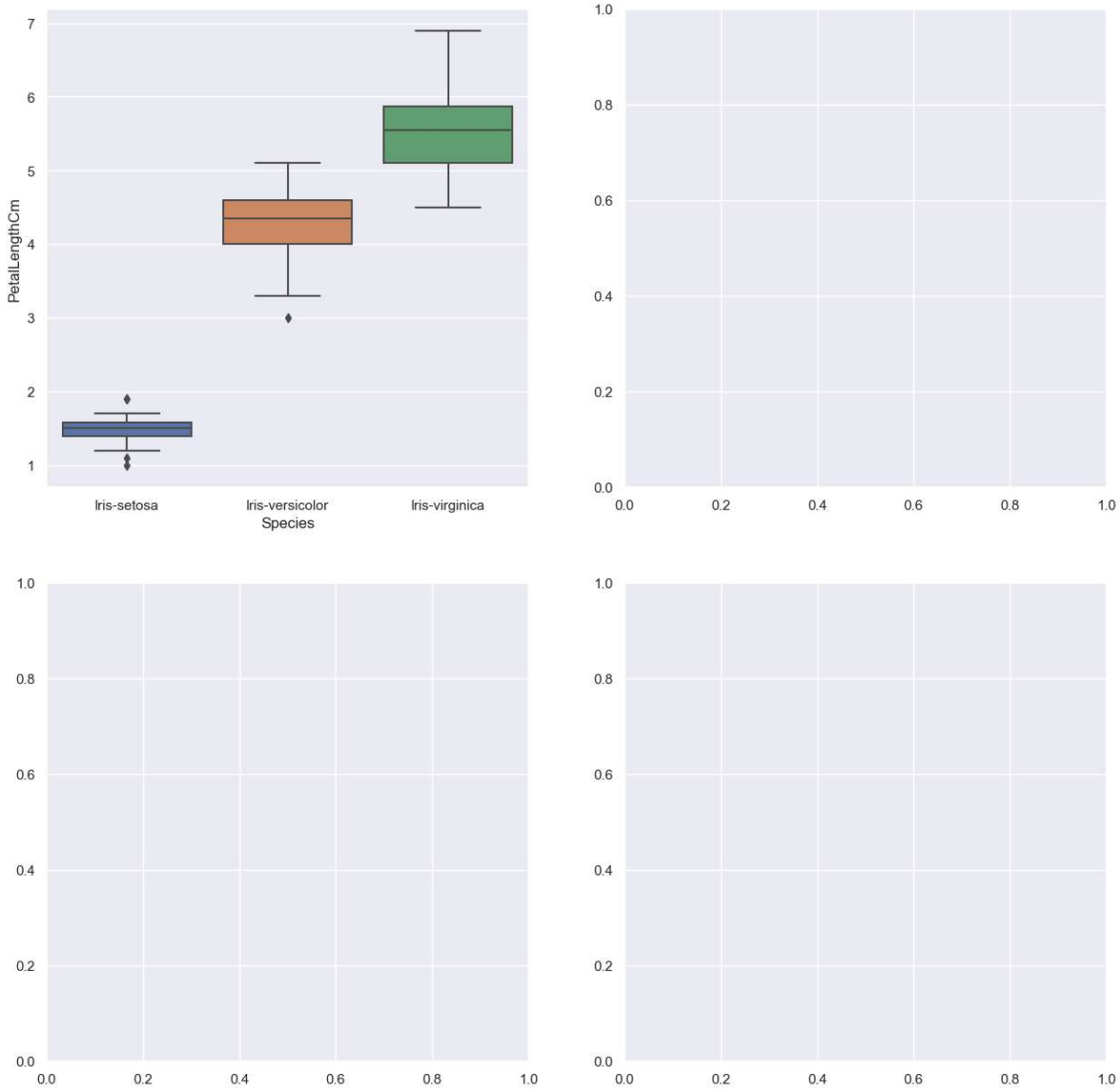
In [46]:
```
plt.ylabel('Sepal Width Cm')
```

Out[46]:
```
Text(0, 0.5, 'Sepal Width Cm')
```

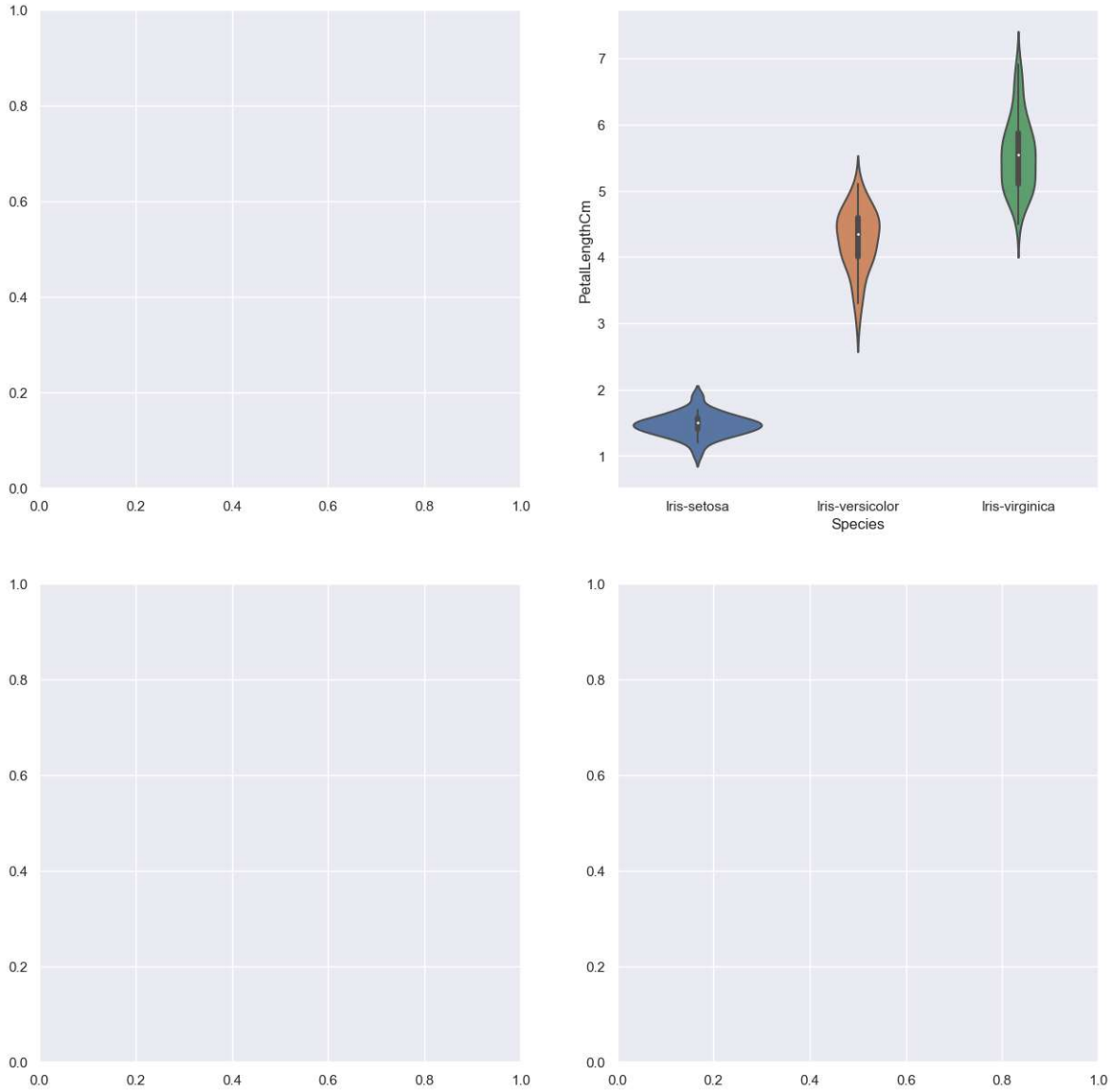


"30.Dashboard"

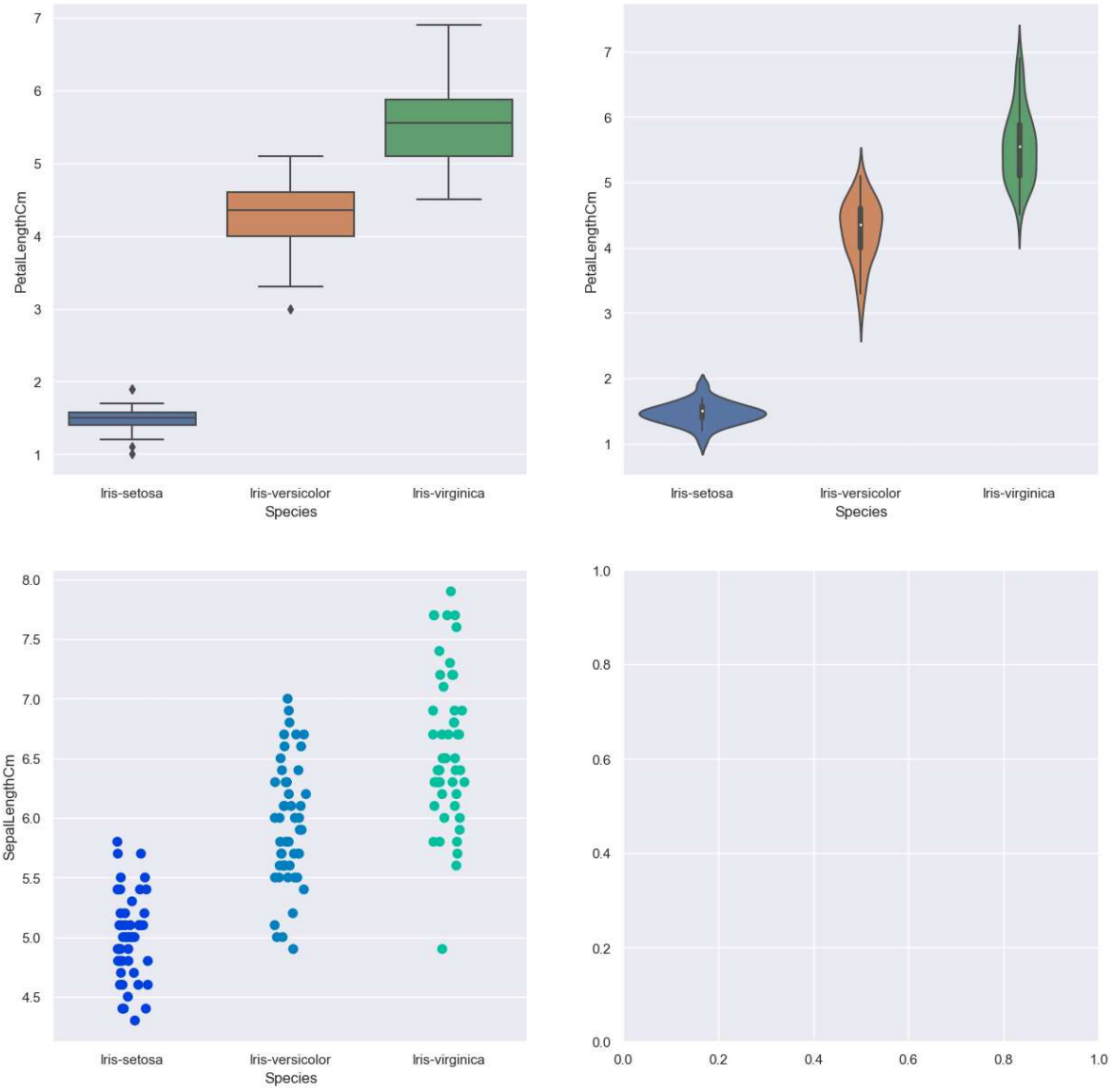
In [53]:
```
sns.set_style('darkgrid')
f,axes=plt.subplots(2,2,figsize=(15,15))
k1=sns.boxplot(x="Species", y="PetalLengthCm", data=iris,ax=axes[0,0])
```



In [55]:
```
f,axes=plt.subplots(2,2,figsize=(15,15))
k2=sns.violinplot(x='Species',y='PetalLengthCm',data=iris,ax=axes[0,1])
```



In [57]:
```
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,edgecolo
```



In [58]:
```
# In the dashboard we have shown how to create multiple plots to foam a dashb
```

In [59]:
```
# "**31.Stacked Histogram**"
```

In [60]:
```
iris['Species'] = iris['Species'].astype('category')
```

In [61]:
```
iris.info()
```

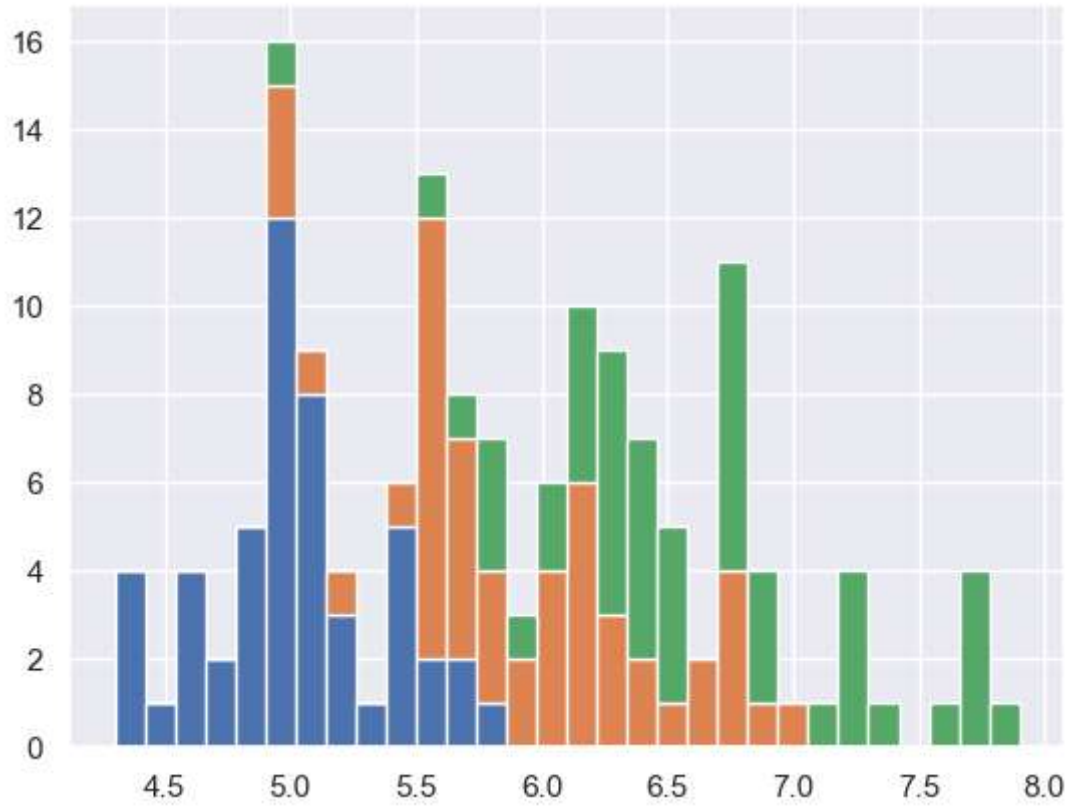
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype   
---  ------         --------------  -----   
 0   SepalLengthCm  150 non-null    float64 
 1   SepalWidthCm   150 non-null    float64 
 2   PetalLengthCm  150 non-null    float64 
 3   PetalWidthCm   150 non-null    float64 
 4   Species        150 non-null    category
dtypes: category(1), float64(4)
memory usage: 5.1 KB
```

In [62]:
```
list1=list()
```

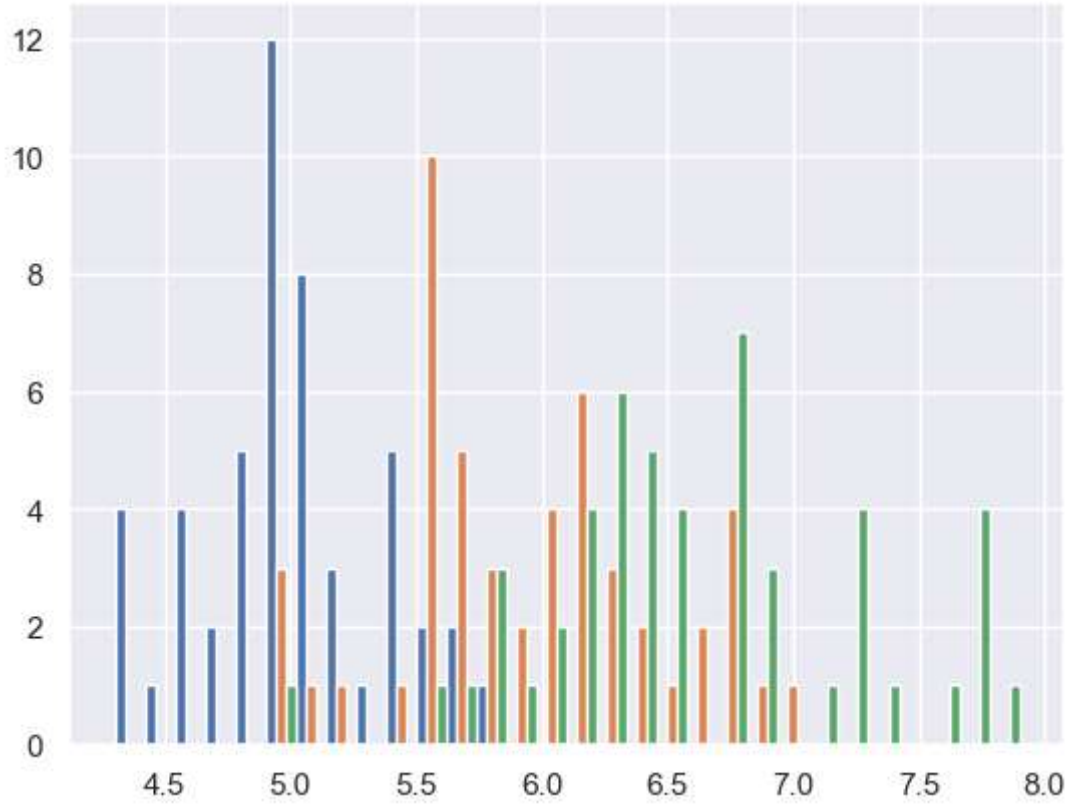
In [63]:
```
mylabels=list()
```

In [64]:
```
for gen in iris.Species.cat.categories:
    list1.append(iris[iris.Species==gen].SepalLengthCm)
    mylabels.append(gen)
```

In [65]:
```
h=plt.hist(list1,bins=30,stacked=True,rwidth=1,label=mylabels)
```



In [66]:
```
h1=plt.hist(list1,bins=30,stacked=False,rwidth=1,label=mylabels)
```



In [67]:
```
# With Stacked Histogram we can see the distribution of Sepal Length of Diffe
```

In [70]:
```
# **32.Area Plot:**
#     Area Plot gives us a visual representation of Various dimensions of Iri
```

In [71]:
```
iris['SepalLengthCm'] = iris['SepalLengthCm'].astype('category')
```

In [72]:
```
iris.head()
```

Out[72]:

	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 [73]:
```
iris.plot.area(y='SepalLengthCm',alpha=0.4,figsize=(12, 6))
```

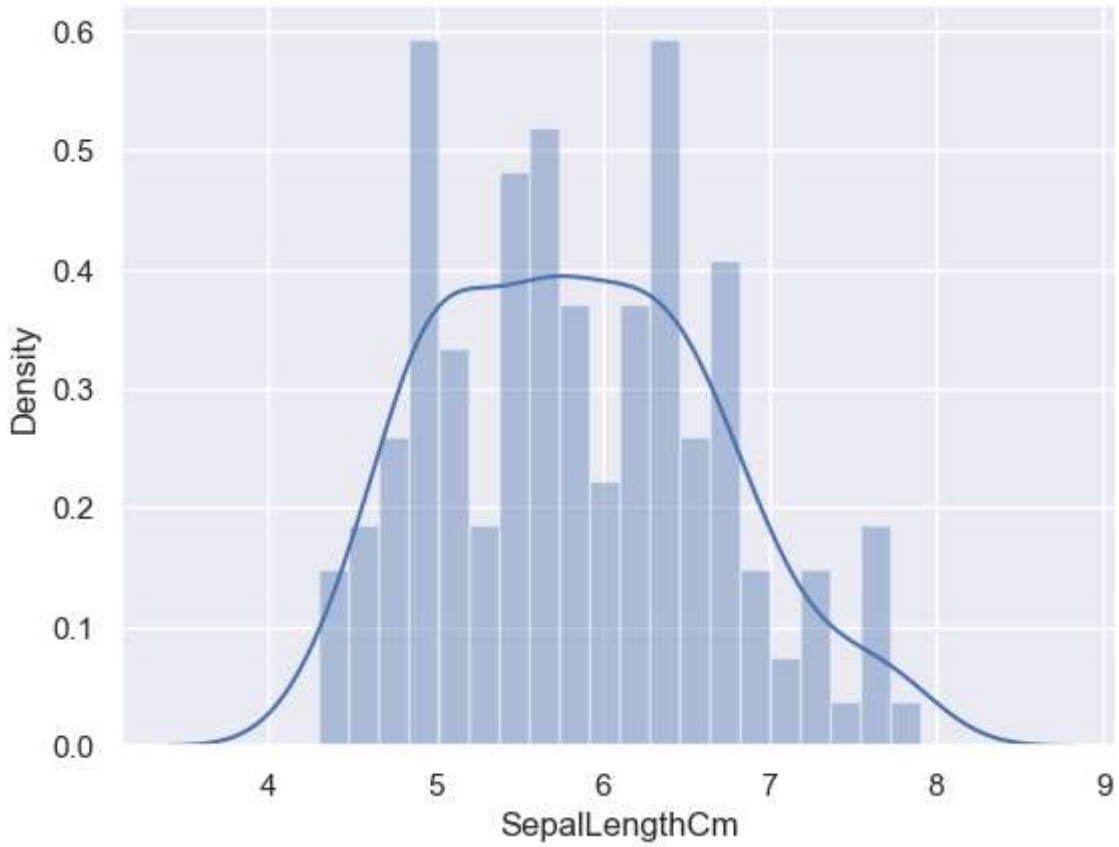
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[73], line 1
----> 1 iris.plot.area(y='SepalLengthCm',alpha=0.4,figsize=(12, 6))

File ~\anaconda3\lib\site-packages\pandas\plotting\_core.py:1557, in PlotAcc
essor.area(self, x, y, **kwargs)
   1486 def area(self, x=None, y=None, **kwargs) -> PlotAccessor:
   1487     """
   1488     Draw a stacked area plot.
   1489
   (...)
   1555         >>> ax = df.plot.area(x='day')
   1556     """
-> 1557     return self(kind="area", x=x, y=y, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\plotting\_core.py:1000, in PlotAcc
essor.__call__(self, *args, **kwargs)
    997             label_name = label_kw or data.columns
    998             data.columns = label_name
-> 1000 return plot_backend.plot(data, kind=kind, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\__init__.py:7
1, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File ~\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:450,
in MPLPlot.generate(self)
    448 def generate(self) -> None:
    449     self._args_adjust()
--> 450     self._compute_plot_data()
    451     self._setup_subplots()
    452     self._make_plot()

File ~\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:635,
in MPLPlot._compute_plot_data(self)
    633 # no non-numeric frames or series allowed
    634 if is_empty:
--> 635     raise TypeError("no numeric data to plot")
    637 self.data = numeric_data.apply(self._convert_to_ndarray)

TypeError: no numeric data to plot
```

In [74]:
```
iris.plot.area(y=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthC
```

Out[74]: <Axes: >

In [76]:
```
# "**33.Distplot:**\n",
#     "It helps us to look at the distribution of a single variable.Kde shows
```

In [77]:
```
sns.distplot(iris['SepalLengthCm'],kde=True,bins=20);
```



In [ ]: