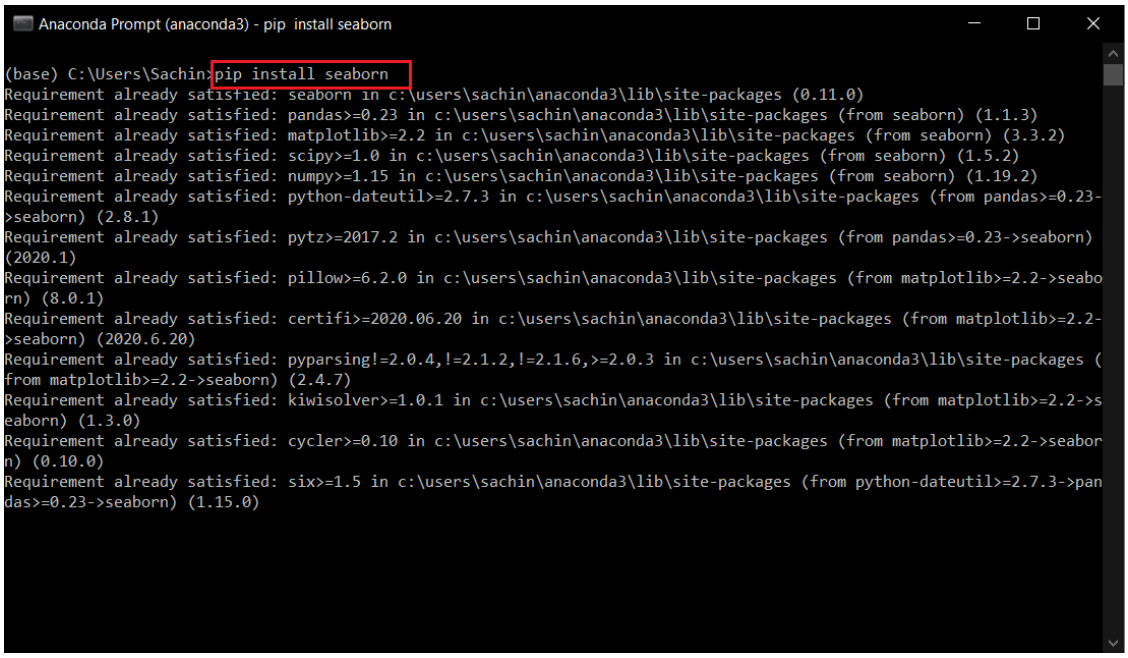
Seaborn

Seaborn is a library mostly used for statistical plotting in Python. It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.

Installing Seaborn

To install Python Seaborn, go to your command prompt and type "pip install seaborn". Once the installation is completed, go to your IDE (For example: PyCharm) or Anaconda Jupyter Notebook and simply import it by typing: **import seaborn as sns**.



How to import Seaborn

In order to start using Seaborn and all of the plot available in Seaborn, You will need to import it. This can be easily done with this import statement.

In [1]:
```
import seaborn as sns
```

NOTE: We shorten seaborn to sns in order to save time and also to keep code standardized so that anyone working with your code can easily understand and run it.

Importing libraries and loading Dataset

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
iris=pd.read_csv("D:\Python learning track and notes\Dataset\Iris.csv")
iris
```

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

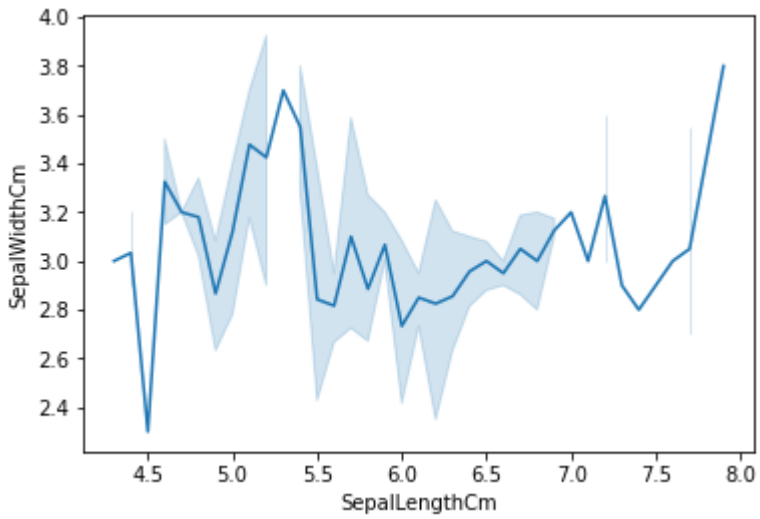
150 rows × 6 columns

Lineplot

It represents the change in a quantity with respect to another quantity. This variation is usually plotted in a two-dimensional XY plane. If the relation including any two measures can be expressed utilizing a straight line in a graph, then such graphs are called linear graphs. Thus, the line plot is also called a linear plot.

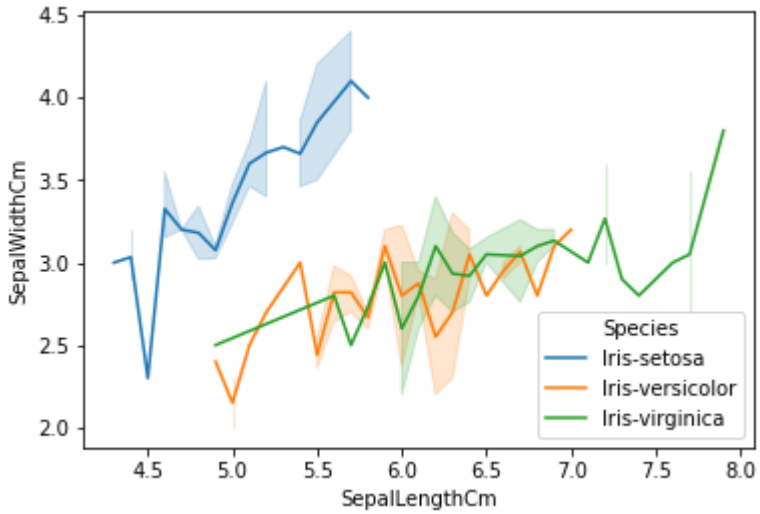
In [3]:
```
sns.lineplot(x="SepalLengthCm", y="SepalWidthCm", data=iris)
```

Out[3]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



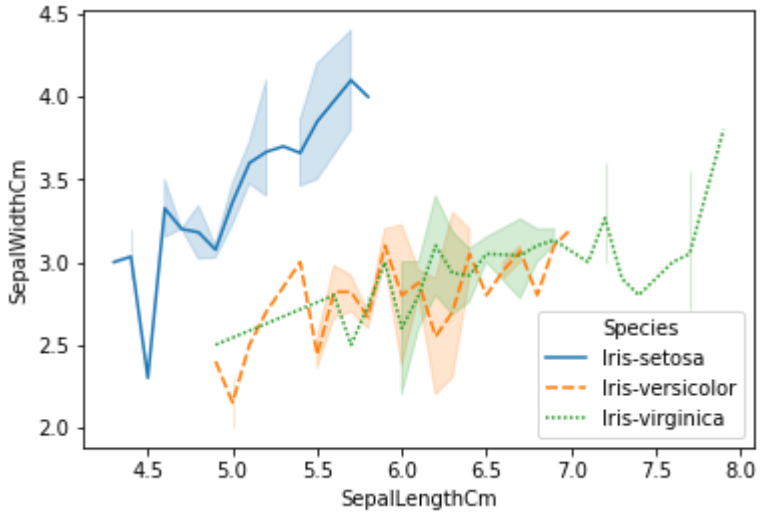
In [4]:
```
sns.lineplot(x="SepalLengthCm", y="SepalWidthCm", data=iris, hue="Specie
```

Out[4]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



In [5]:
```
sns.lineplot(x="SepalLengthCm", y="SepalWidthCm", data=iris, hue="Specie
```

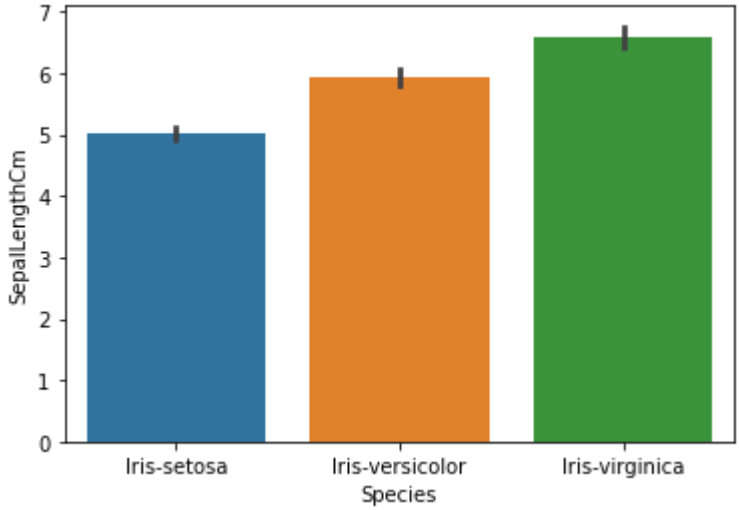
Out[5]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



Barplot

A barplot is basically used to aggregate the categorical data according to some methods and by default it's the mean. It can also be understood as a visualization of the group by action. To use this plot we choose a categorical column for the x-axis and a numerical column for the y-axis, and we see that it creates a plot taking a mean per categorical column.

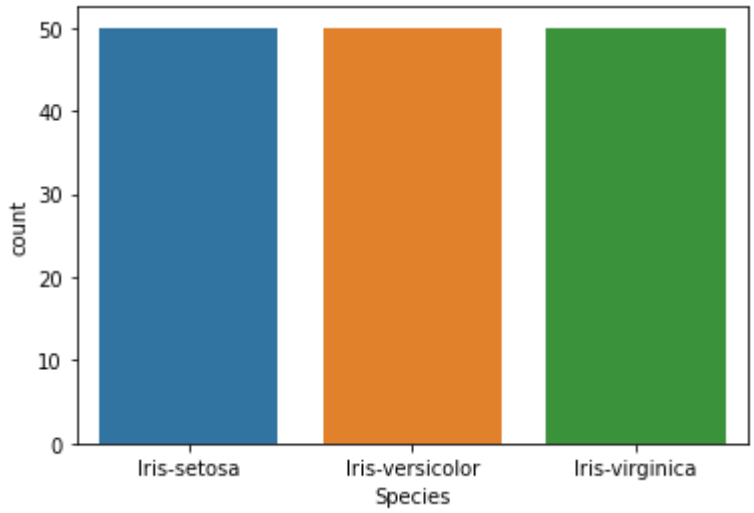
In [6]:
```
sns.barplot(x='Species', y='SepalLengthCm', data=iris)
plt.show()
```



Count plot

A countplot method is used to Show the counts of observations in each categorical bin using bars.

In [7]:
```
sns.countplot(x='Species', data=iris)
plt.show()
```



Box

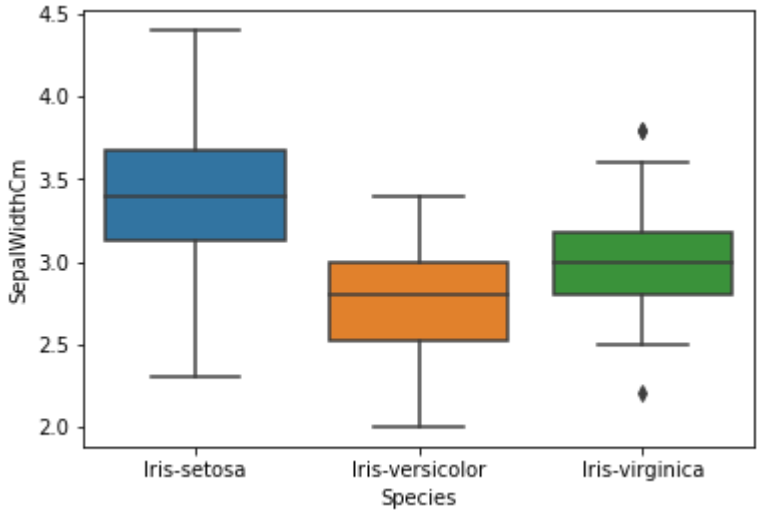
Box Plot is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detect the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups. Boxplot summarizes a sample data using 25th, 50th and 75th percentiles. These percentiles are also known as the lower quartile, median and upper quartile.

A box plot consist of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%

- Third Quartile or 75%
- Maximum

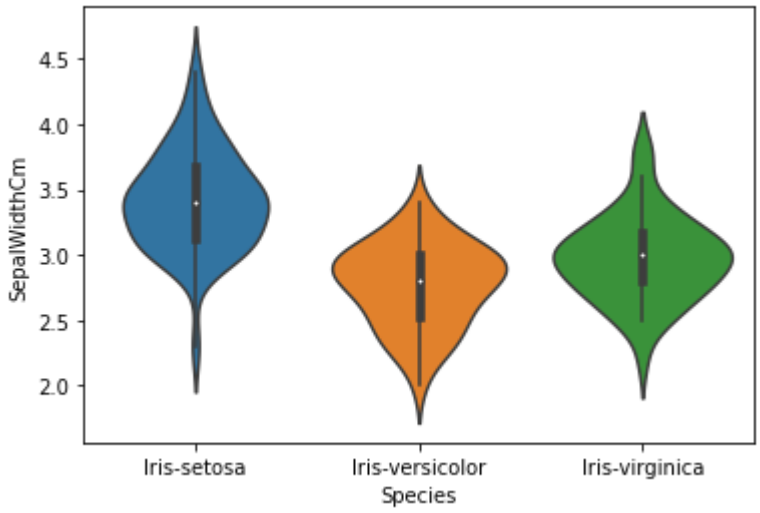
In [8]:
```
sns.boxplot(x='Species', y='SepalWidthCm', data=iris)
plt.show()
```



Violinplot

A violin plot plays a similar activity that is pursued through whisker or box plot do. As it shows several quantitative data across one or more categorical variables. It can be an effective and attractive way to show multiple data at several units. A “wide-form” Data Frame helps to maintain each numeric column which can be plotted on the graph.

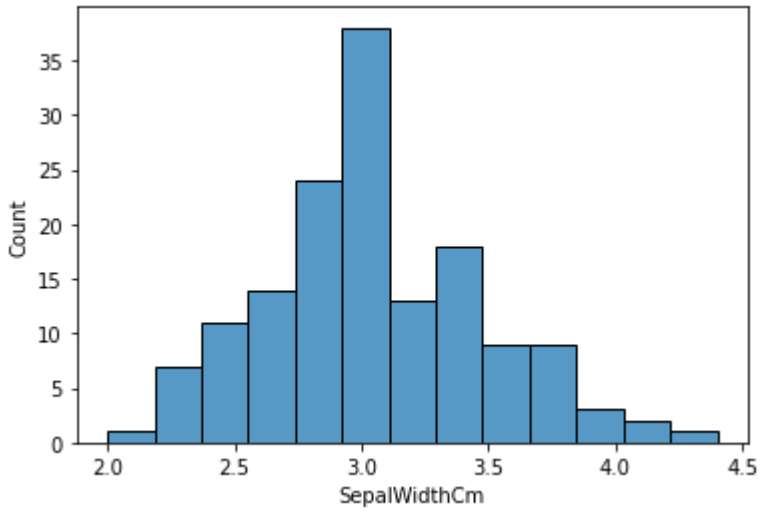
In [9]:
```
sns.violinplot(x='Species', y='SepalWidthCm', data=iris)
plt.show()
```



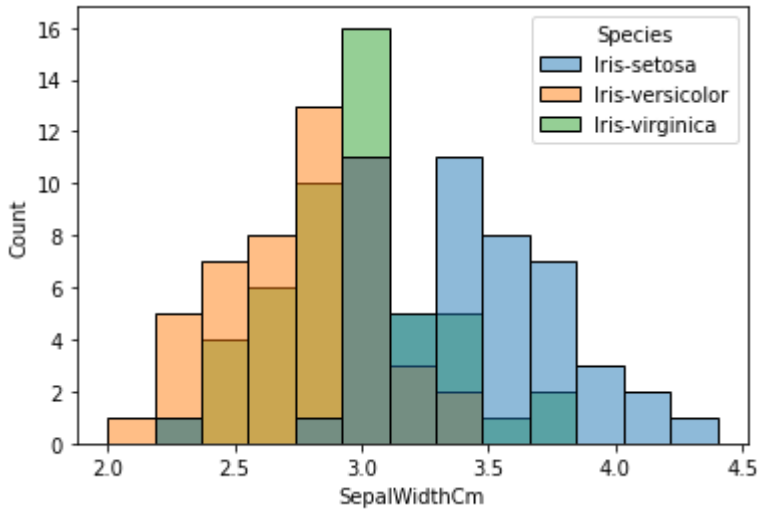
Histogram

When the grouped data are represented horizontally in a chart with the help of bars, then such graphs are called horizontal bar graphs, where the bars show the measure of data. The data is depicted here along the x-axis of the graph, and the length of the bars denote the values.

In [10]:
```
sns.histplot(x='SepalWidthCm', data=iris)
plt.show()
```



In [11]:
```
sns.histplot(x='SepalWidthCm', data=iris, hue='Species')
plt.show()
```

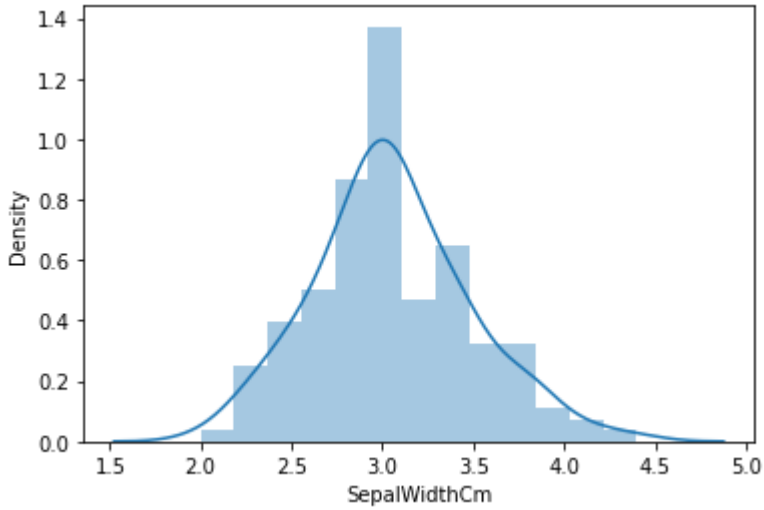


Distplot

It is used basically for univariant set of observations and visualizes it through a histogram i.e. only one observation and hence we choose one particular column of the dataset.

In [12]:
```
sns.distplot(iris['SepalWidthCm'])
plt.show()
```

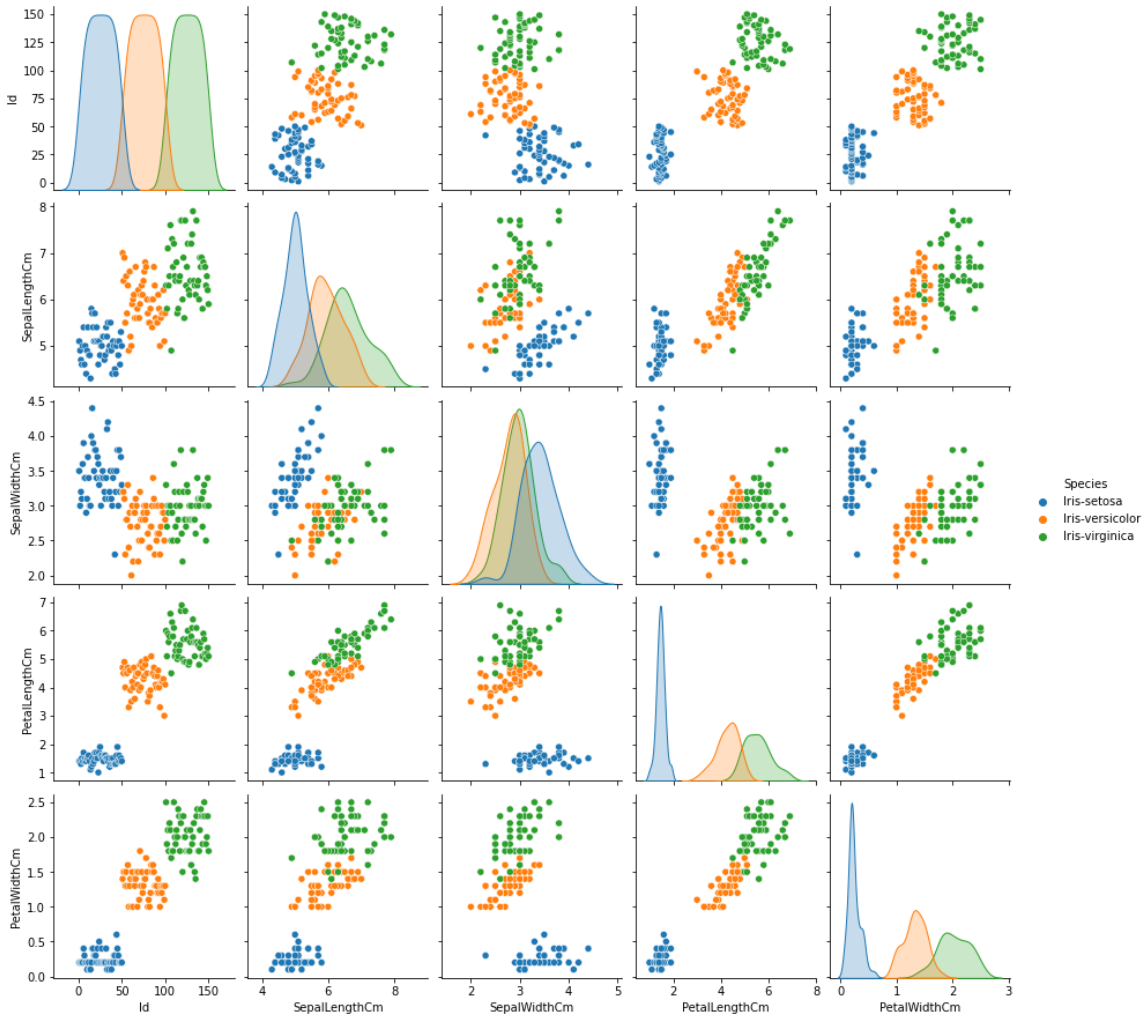
```
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:255
1: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-leve
l function for histograms).
  warnings.warn(msg, FutureWarning)
```



Pairplot

To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This shows the relationship for (n, 2) combination of variable in a DataFrame as a matrix of plots and the diagonal plots are the univariate plots.

In [13]:
```
sns.pairplot(data=iris, hue='Species')
plt.show()
```



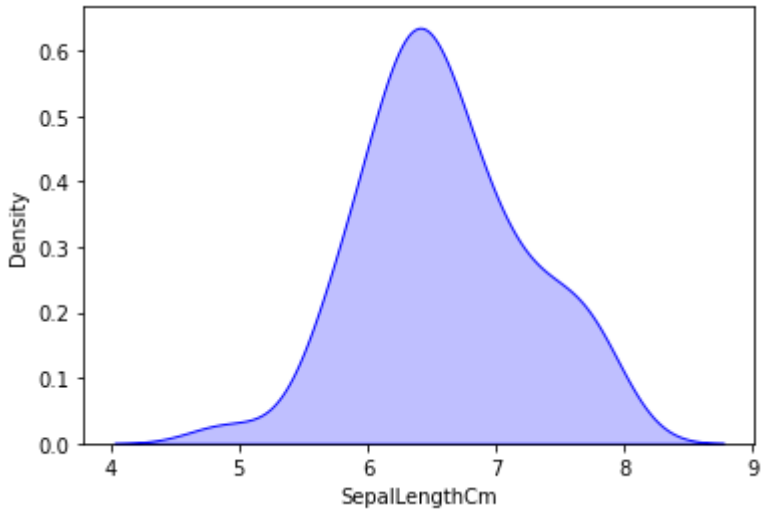
KDE Plot

KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

In [14]:
```
sns.kdeplot(iris.loc[(iris['Species']=='Iris-virginica'),'SepalLengthCm'
```

```
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:948:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  scout = self.ax.fill_between([], [], **plot_kws)
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:991:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  artist = ax.fill_between(
```

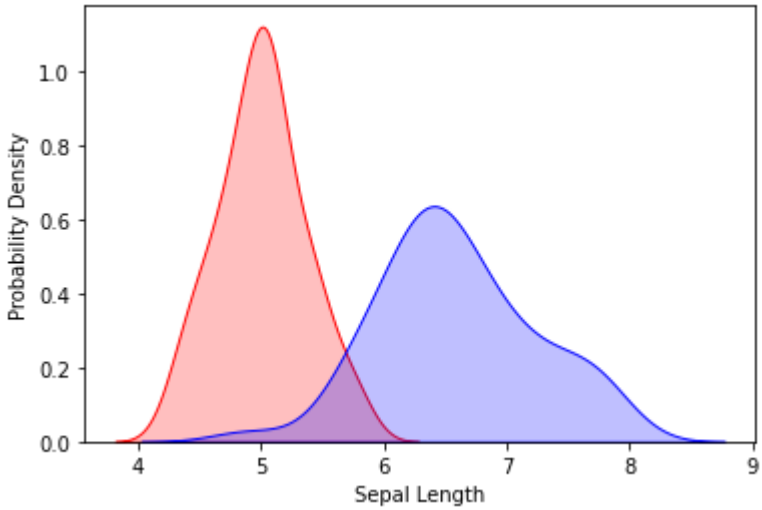
Out[14]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='Density'>
```



In [15]:
```
sns.kdeplot(iris.loc[(iris['Species']=='Iris-setosa'),'SepalLengthCm'],
sns.kdeplot(iris.loc[(iris['Species']=='Iris-virginica'),'SepalLengthCm'
plt.xlabel('Sepal Length')
plt.ylabel('Probability Density')
```

```
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:948:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  scout = self.ax.fill_between([], [], **plot_kws)
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:991:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  artist = ax.fill_between(
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:948:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  scout = self.ax.fill_between([], [], **plot_kws)
C:\Users\Sachin\anaconda3\lib\site-packages\seaborn\distributions.py:991:
MatplotlibDeprecationWarning: Case-insensitive properties were deprecated
in 3.3 and support will be removed two minor releases later
  artist = ax.fill_between(
```

Out[15]:
```
Text(0, 0.5, 'Probability Density')
```



In [16]:
```
sns.kdeplot(x='SepalLengthCm', y='SepalWidthCm', color="red",data=iris,sh
plt.show()
```