Out[50]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.6
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.6
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.5
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
...	...	...	...	...	...	...	...	...	...	.
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.7
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.7
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.7
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.6

1599 rows × 12 columns

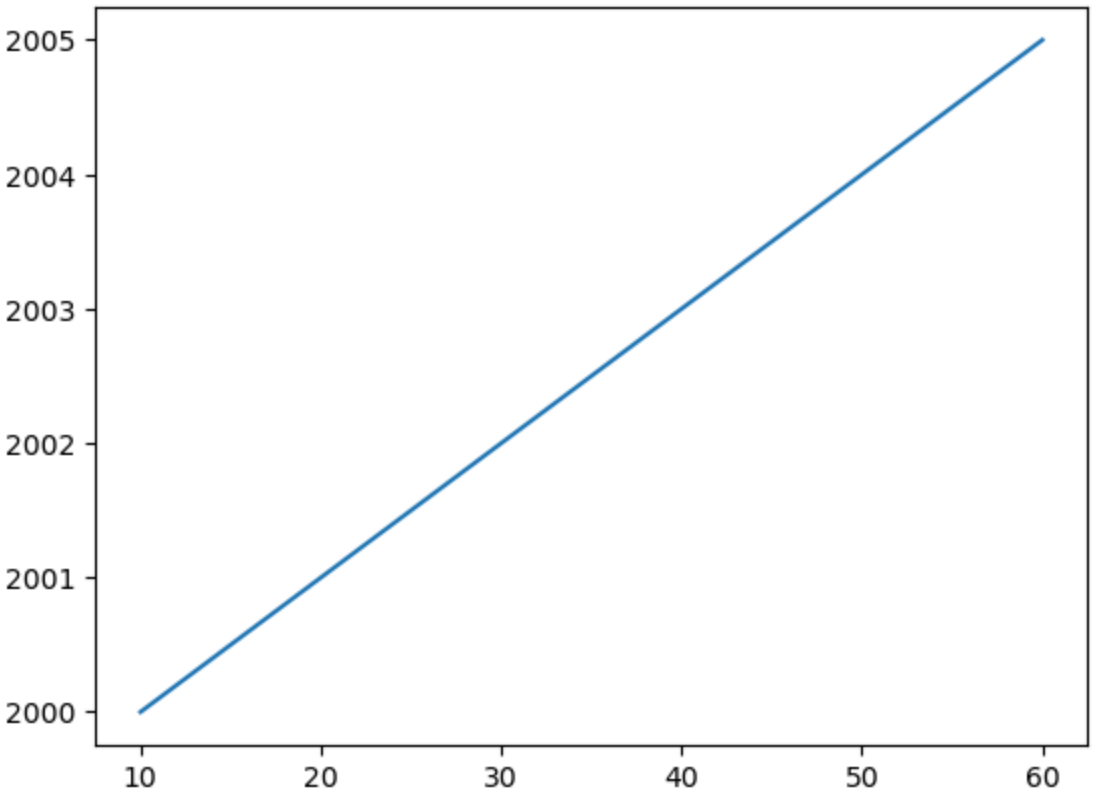
Matplotlib :- It is Data Visualization Tool.

Used to plot different graphs.

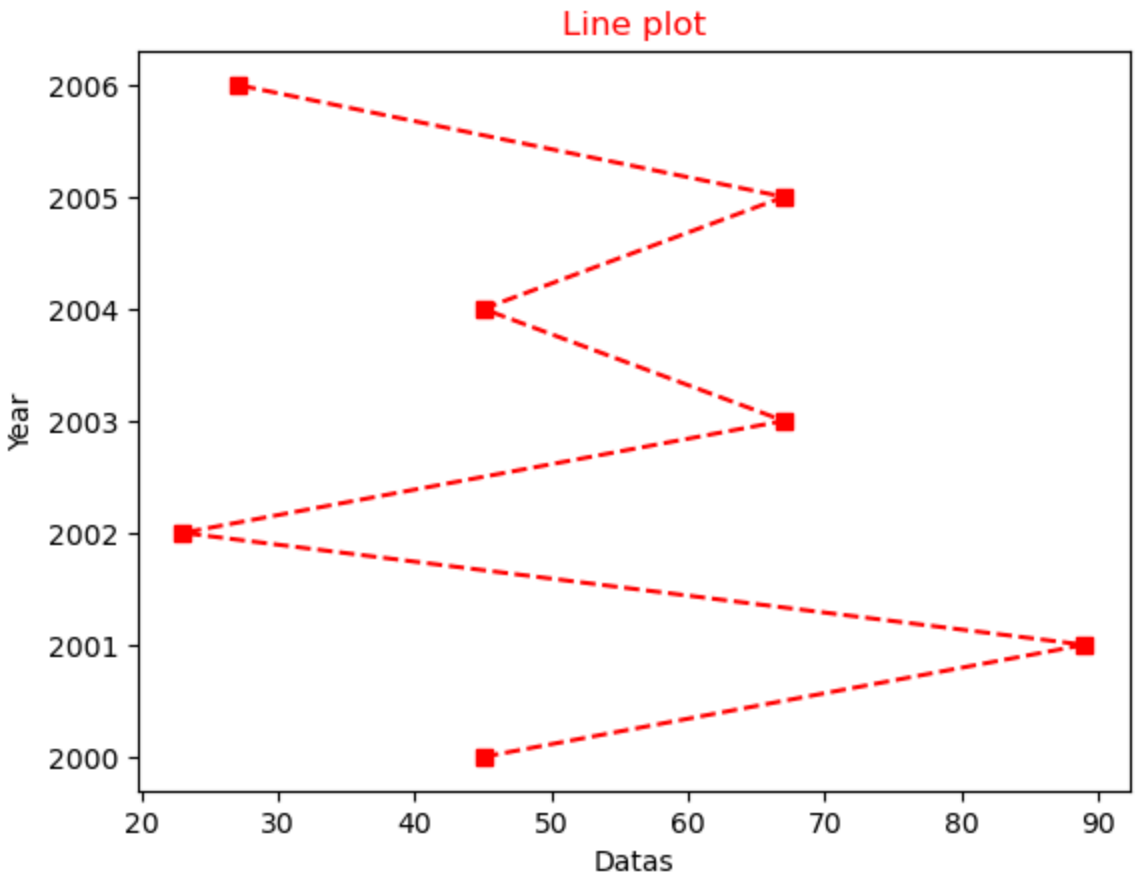
In [1]:
```
# Import the libraries:-
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Line Plot :- is used to visualize a particular trend over time.

In [52]:
```
x = [10,20,30,40,50,60]
y = [2000,2001,2002,2003,2004,2005]
plt.plot(x,y) # to the graph using x & y data.
plt.show()  # to plot the graph
```

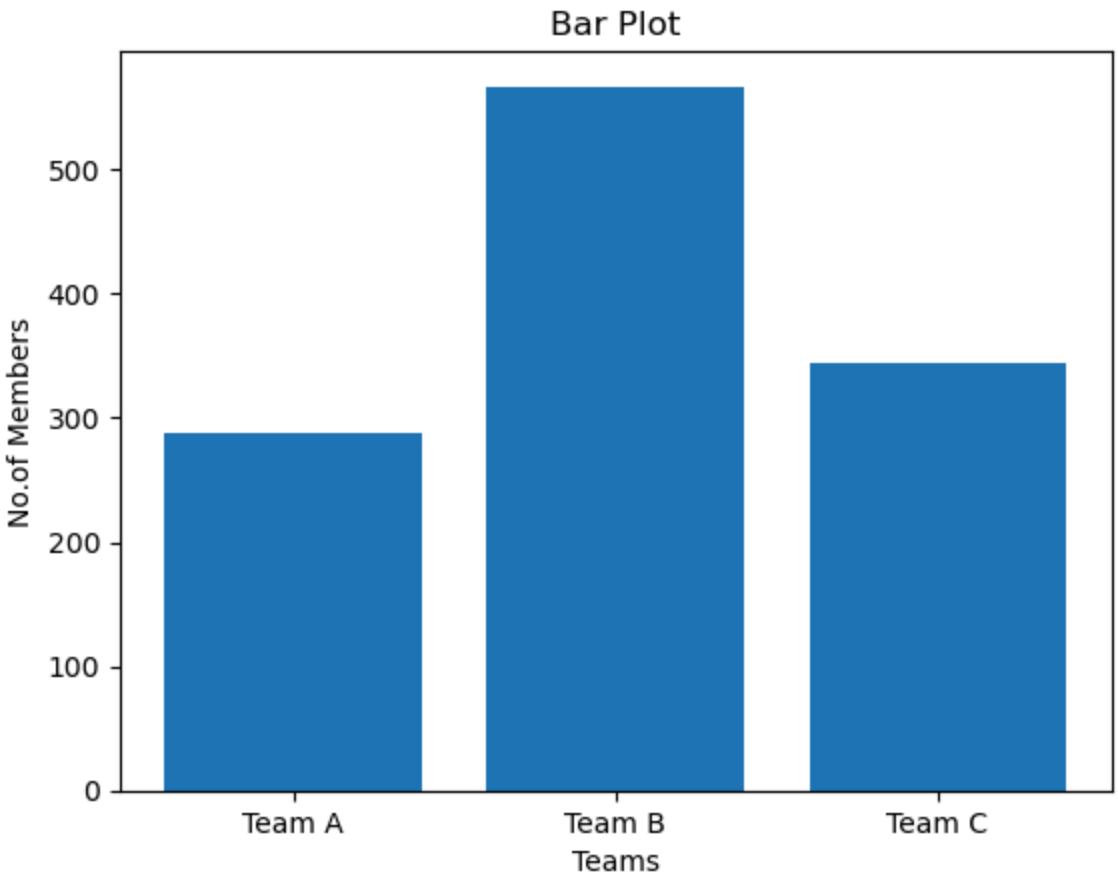


In [61]:
```
x = [45,89,23,67,45,67,27]
y = [2000,2001,2002,2003,2004,2005,2006]
plt.title('Line plot',c='red') # gives a heading to the graph
plt.plot(x,y,marker='s',linestyle='--',c='r')
plt.xlabel('Datas') # gives a name to x-axis
plt.ylabel('Year') # gives a name to y-axis
plt.show()
```

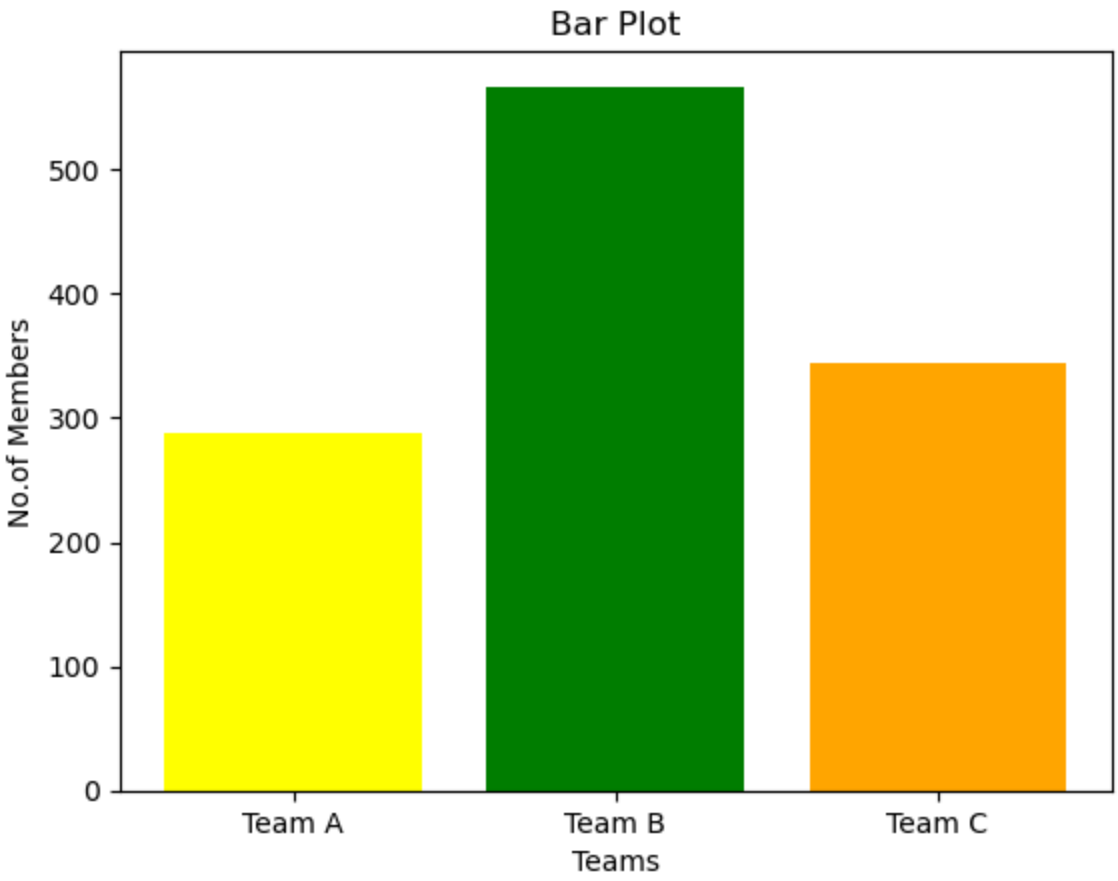


Bar plot:- used for comparision of categories.

In [62]:
```
x = ['Team A','Team B','Team C']
y = [288,567,345]
plt.bar(x,y) # plot a bar graph
plt.title('Bar Plot')
plt.xlabel('Teams')
plt.ylabel('No.of Members')
plt.show()
```

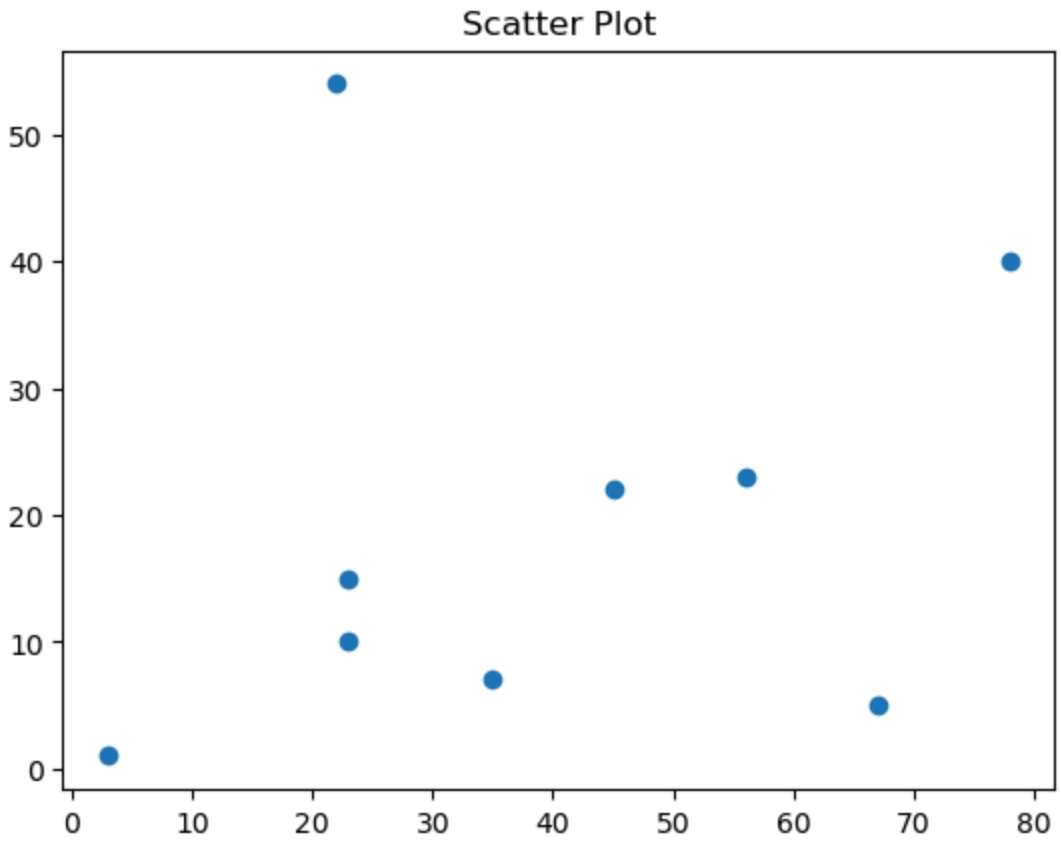


In [63]:
```
x = ['Team A','Team B','Team C']
y = [288,567,345]
colours = ['yellow','green','orange']
plt.bar(x,y,color = colours ) # plot a bar graph
plt.title('Bar Plot')
plt.xlabel('Teams')
plt.ylabel('No.of Members')
plt.show()
```

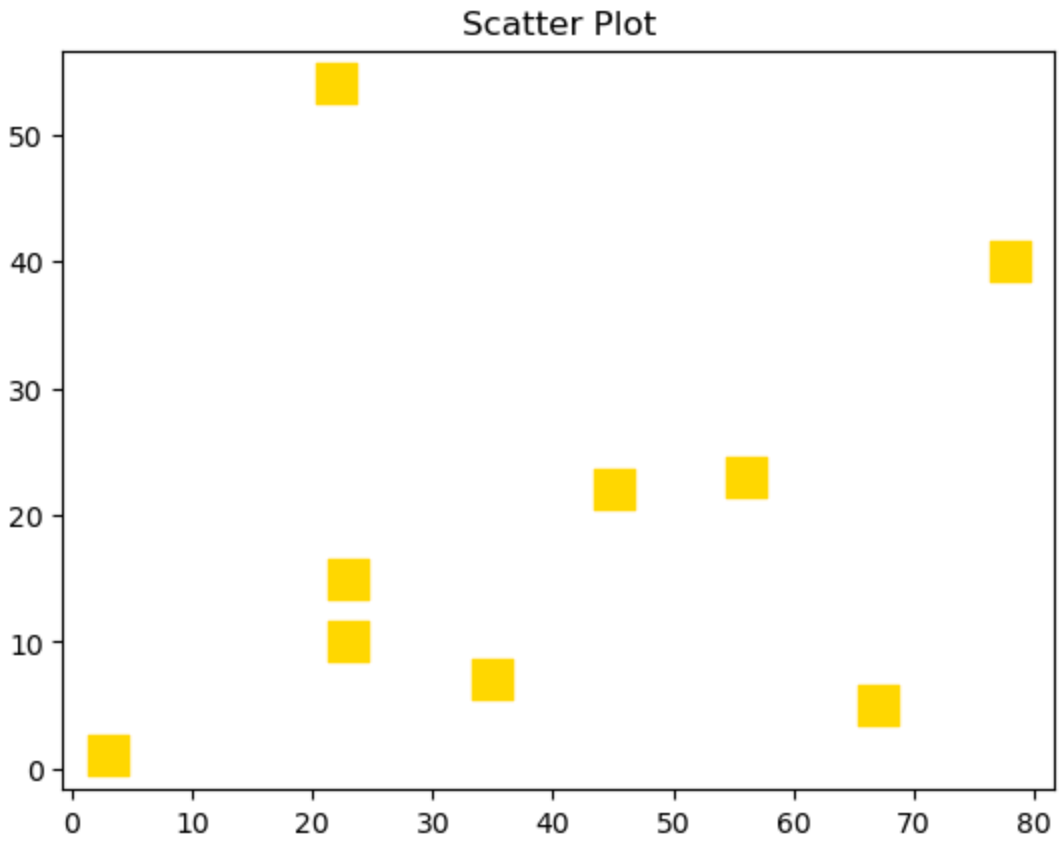


Scatter Plot:- Used to display the relationship between 2 variables.

In [83]:
```
x = [3,67,35,23,45,23,78,56,22]
y = [1,5,7,10,22,15,40,23,54]
plt.scatter(x,y)
plt.title('Scatter Plot')
plt.show()
```

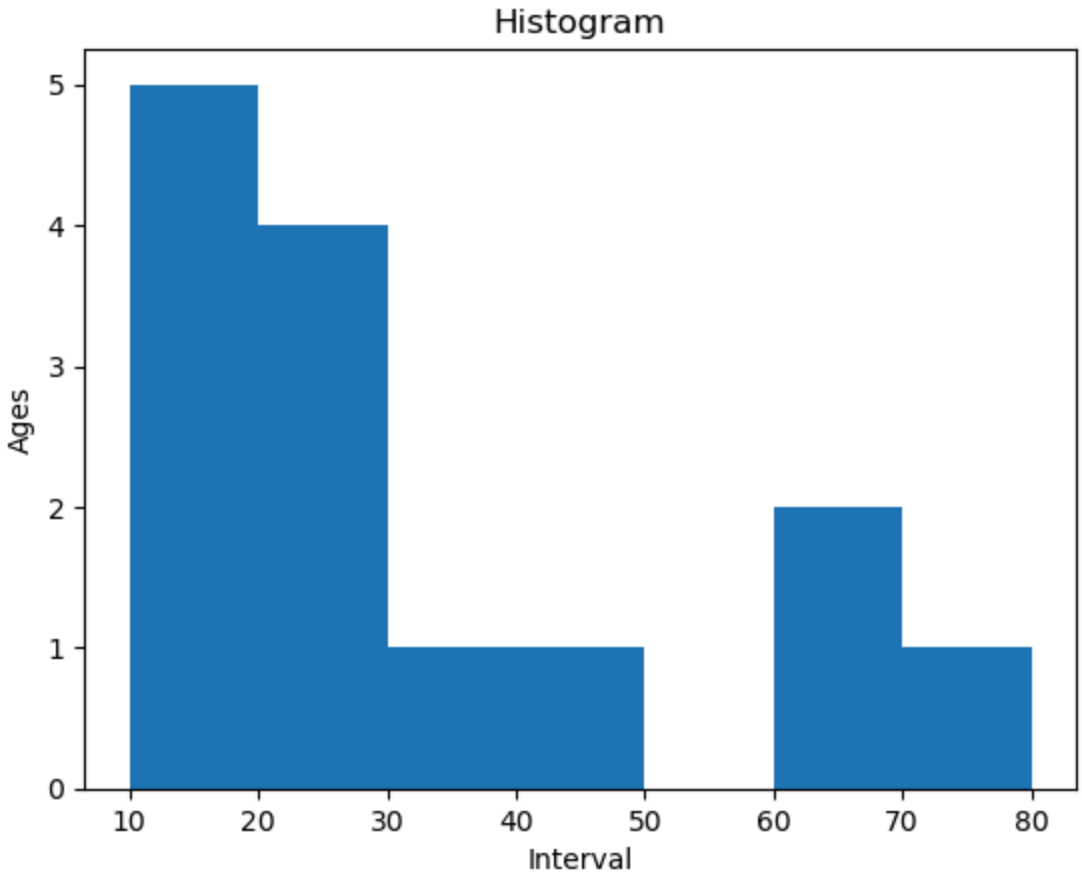


In [84]:
```
x = [3,67,35,23,45,23,78,56,22]
y = [1,5,7,10,22,15,40,23,54]
plt.scatter(x,y,marker='s',s=200,color='gold')
plt.title('Scatter Plot')
plt.show()
```

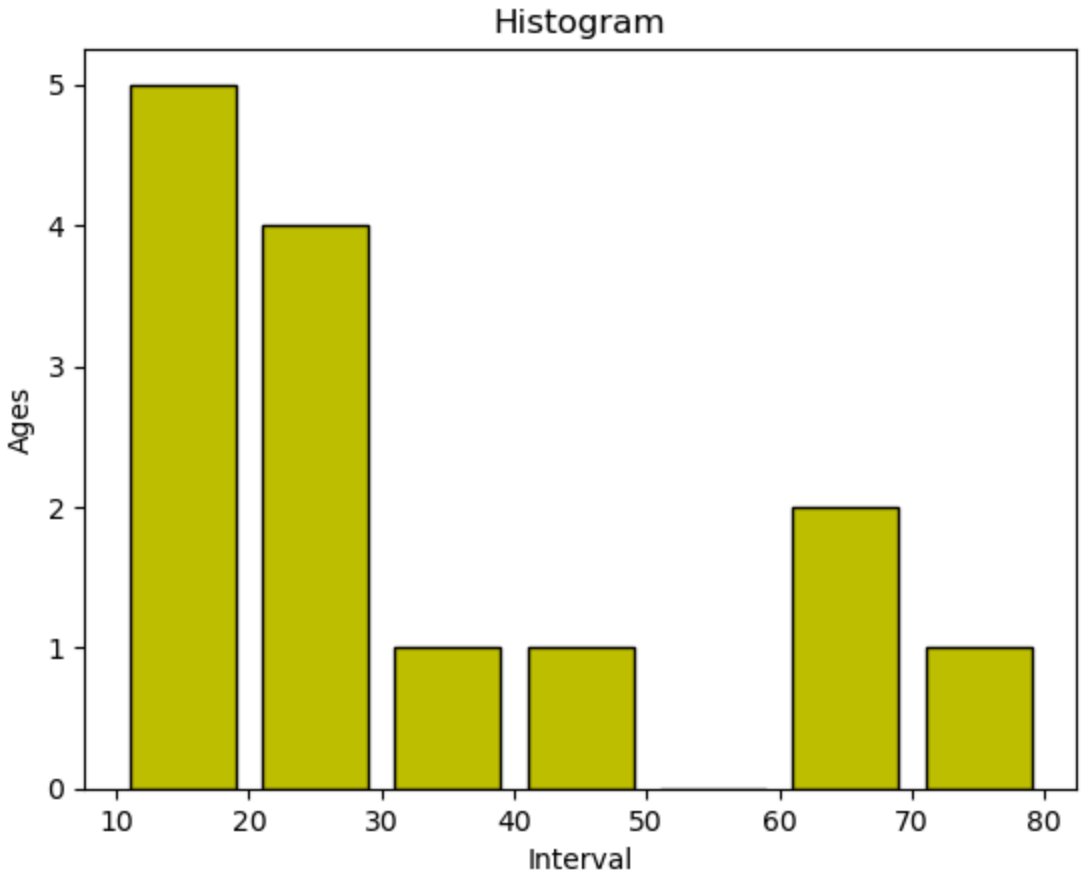


Histogram :- Used to display the frequency of the data distribution.

In [85]:
```
age = [15,26,14,18,16,75,26,29,60,30,25,49,64,11]
bins = [10,20,30,40,50,60,70,80]
plt.hist(age,bins)
plt.title('Histogram')
plt.xlabel('Interval')
plt.ylabel('Ages')
plt.show()
```

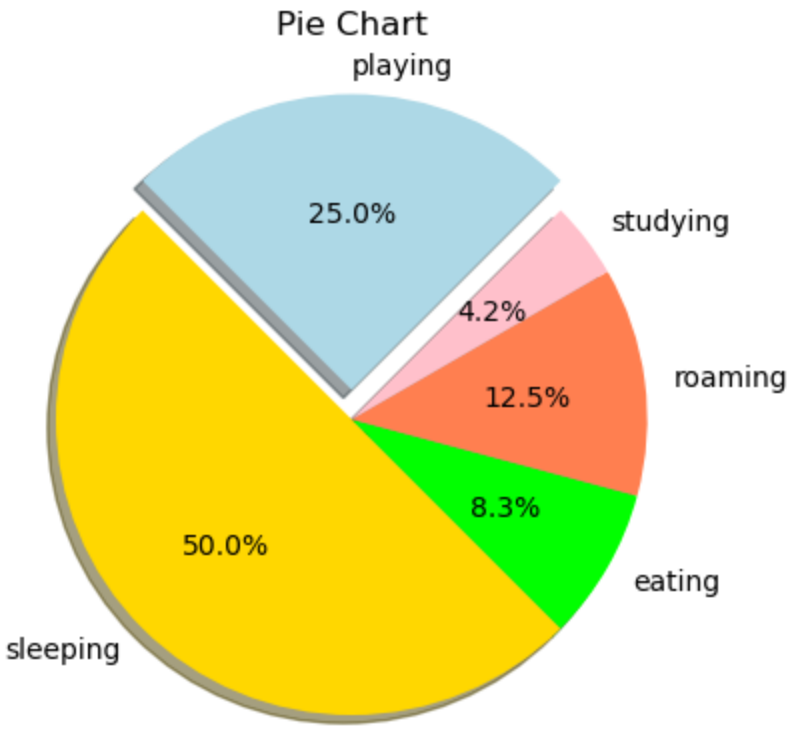


In [86]:
```
age = [15,26,14,18,16,75,26,29,60,30,25,49,64,11]
bins = [10,20,30,40,50,60,70,80]
plt.hist(age,bins,histtype='bar',rwidth=0.8,color='y',
        edgecolor='k')
plt.title('Histogram')
plt.xlabel('Interval')
plt.ylabel('Ages')
plt.show()
```



Pie Plot/ Pie Chart:- Used to display data as a proportion of any whole.

In [87]:
```
activities = ['playing','sleeping','eating','roaming',
              'studying']
slices = [6,12,2,3,1]
cl = ['lightblue','gold','lime','coral','pink']
plt.pie(slices,labels=activities,colors = cl,
        startangle=45,shadow=True,explode=(0.1,0,0,0,0),
       autopct='%1.1f%%')
plt.title('Pie Chart')
plt.show()
```



Seaborn: Data Visualization Tool

Statistical Data Visualization

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

In [2]:
```
# Reading a dataset:-
iris = pd.read_csv(r"C:\Users\CTTC\Downloads\iris\iris.data",
                   header=None)
iris
```

Out[2]:

	0	1	2	3	4
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [3]:
```
iris.columns = ['SL','SW','PL','PW','FlowerType']
```

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

Out[4]:

	SL	SW	PL	PW	FlowerType
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 [92]:
```
# Data Cleaning:-
iris.isnull().sum()
```

Out[92]:
```
SL            0
SW            0
PL            0
PW            0
FlowerType    0
dtype: int64
```

In [93]:
```
iris.dtypes
```

Out[93]:
```
SL            float64
SW            float64
PL            float64
PW            float64
FlowerType     object
dtype: object
```

In [94]:
```
# to find the missing values
for i in iris.columns:
    print(f"{i}:\n {iris[i].unique()}\n")
```

```
SL:
 [5.1 4.9 4.7 4.6 5.  5.4 4.4 4.8 4.3 5.8 5.7 5.2 5.5 4.5 5.3 7.  6.4 6.9
 6.5 6.3 6.6 5.9 6.  6.1 5.6 6.7 6.2 6.8 7.1 7.6 7.3 7.2 7.7 7.4 7.9]

SW:
 [3.5 3.  3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.  4.4 3.8 3.3 4.1 4.2 2.3 2.8 2.4
 2.7 2.  2.2 2.5 2.6]

PL:
 [1.4 1.3 1.5 1.7 1.6 1.1 1.2 1.  1.9 4.7 4.5 4.9 4.  4.6 3.3 3.9 3.5 4.2
 3.6 4.4 4.1 4.8 4.3 5.  3.8 3.7 5.1 3.  6.  5.9 5.6 5.8 6.6 6.3 6.1 5.3
 5.5 6.7 6.9 5.7 6.4 5.4 5.2]

PW:
 [0.2 0.4 0.3 0.1 0.5 0.6 1.4 1.5 1.3 1.6 1.  1.1 1.8 1.2 1.7 2.5 1.9 2.1
 2.2 2.  2.4 2.3]

FlowerType:
 ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

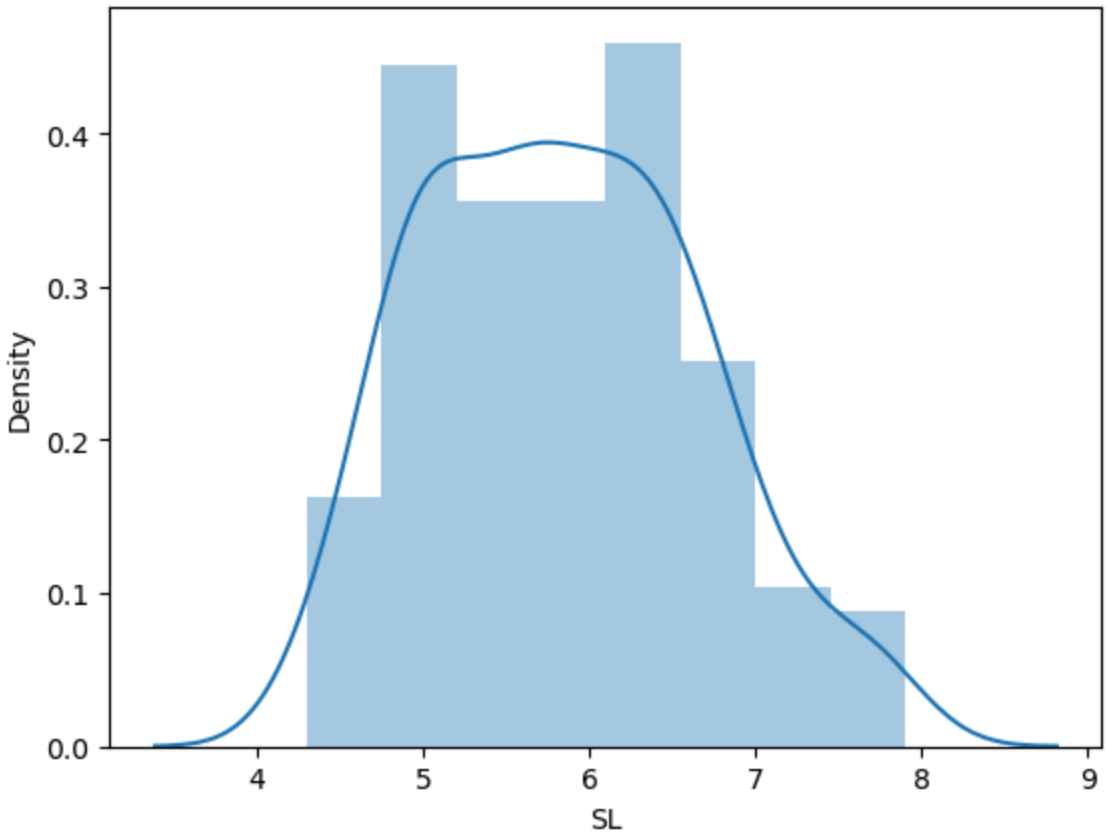
In [95]:
```
# to remove the warnings
import warnings
warnings.filterwarnings('ignore')
```

Data Visualization:-

Distribution Plot:- used to show the distribution of any single variable.

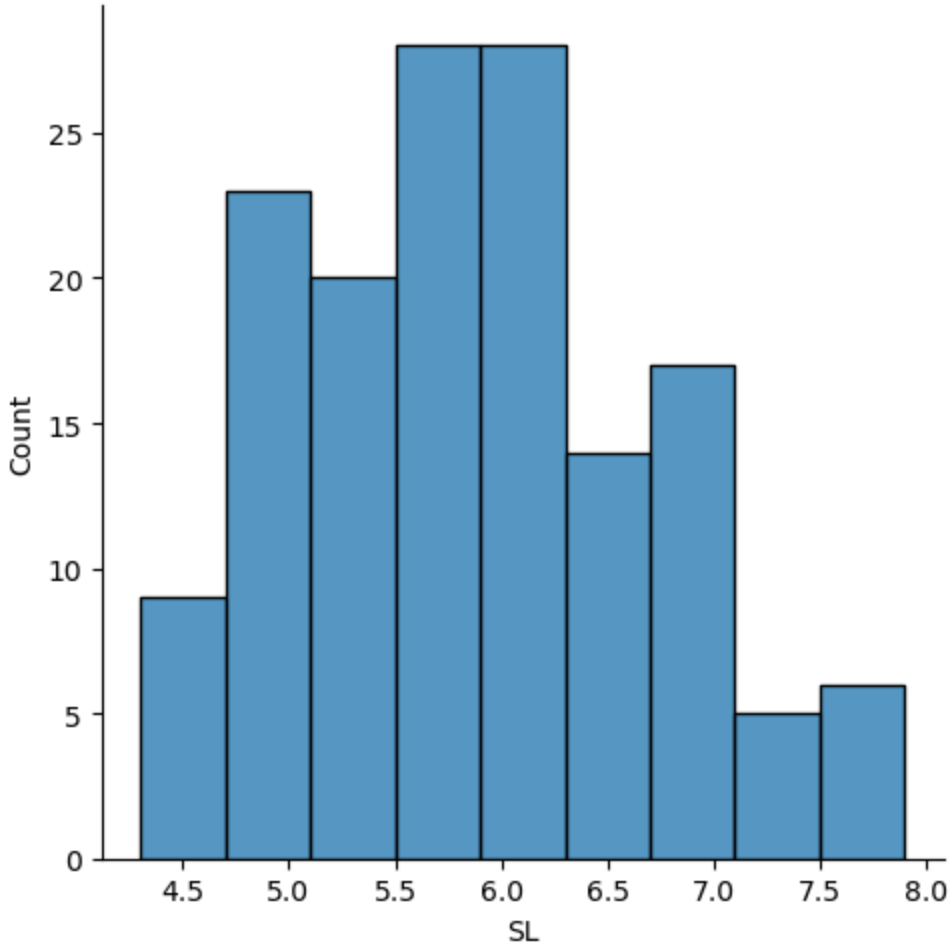
combines the histogram and kernal density estimate plot (KDE).

In [96]:
```
sns.distplot(iris.SL)
plt.show()
```

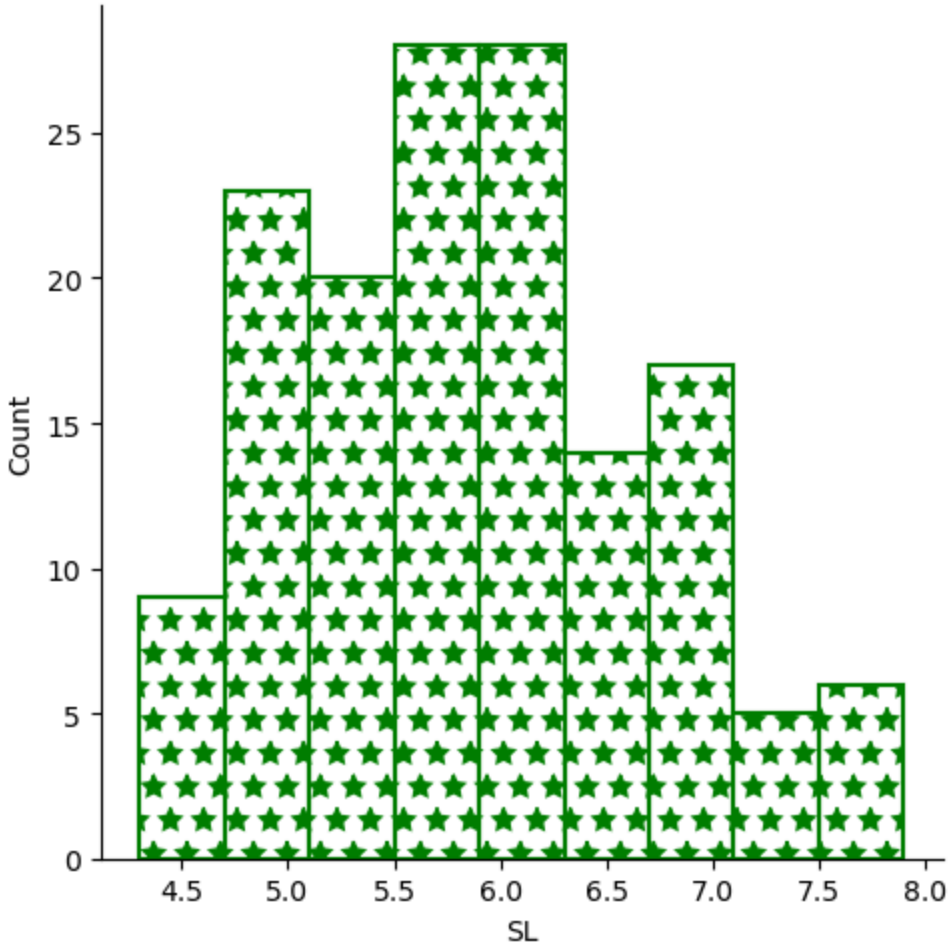


Displot: shows the distribution of a single variable using histogram.

In [97]:
```
sns.displot(iris.SL)
plt.show()
```

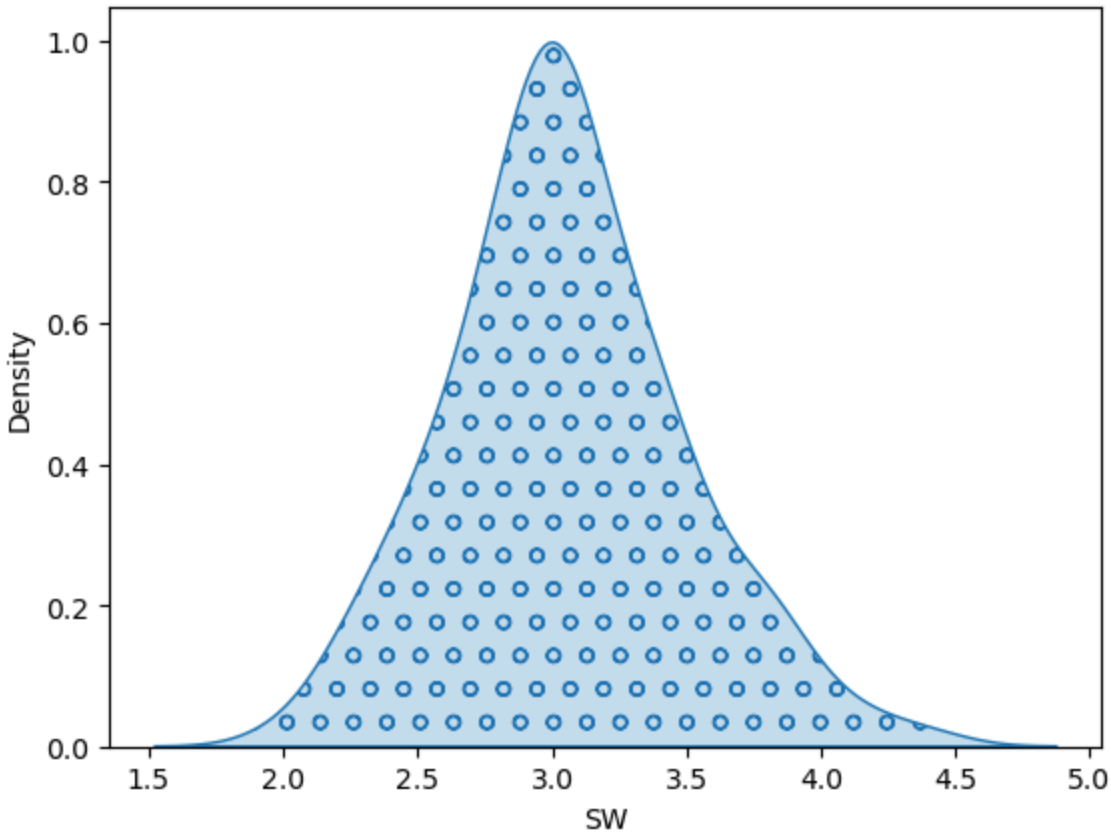


In [98]:
```
sns.displot(iris.SL,fill=False,hatch='*',color='g')
plt.show()
```



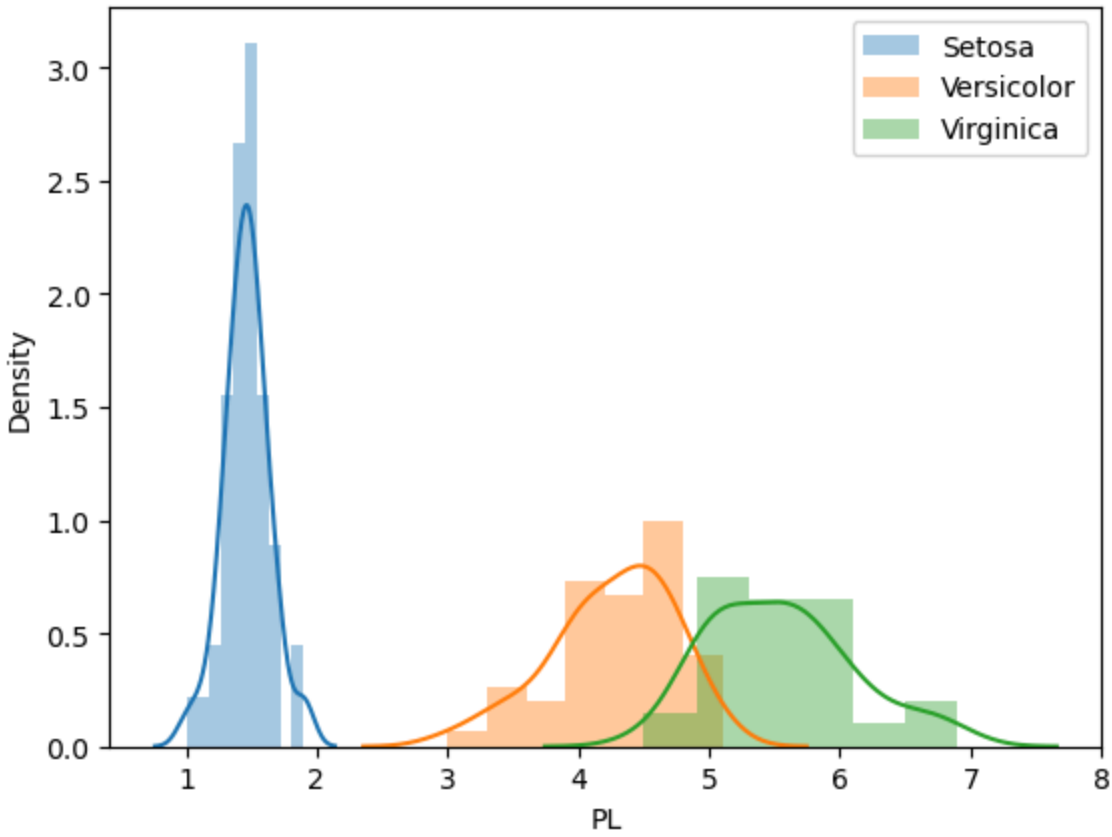
KDE plot: Kernal Density Estimate plot is used to display the frequency distribution of any single variable using curve line.

In [99]:
```
sns.kdeplot(iris.SW,fill=True,hatch='o')
plt.show()
```



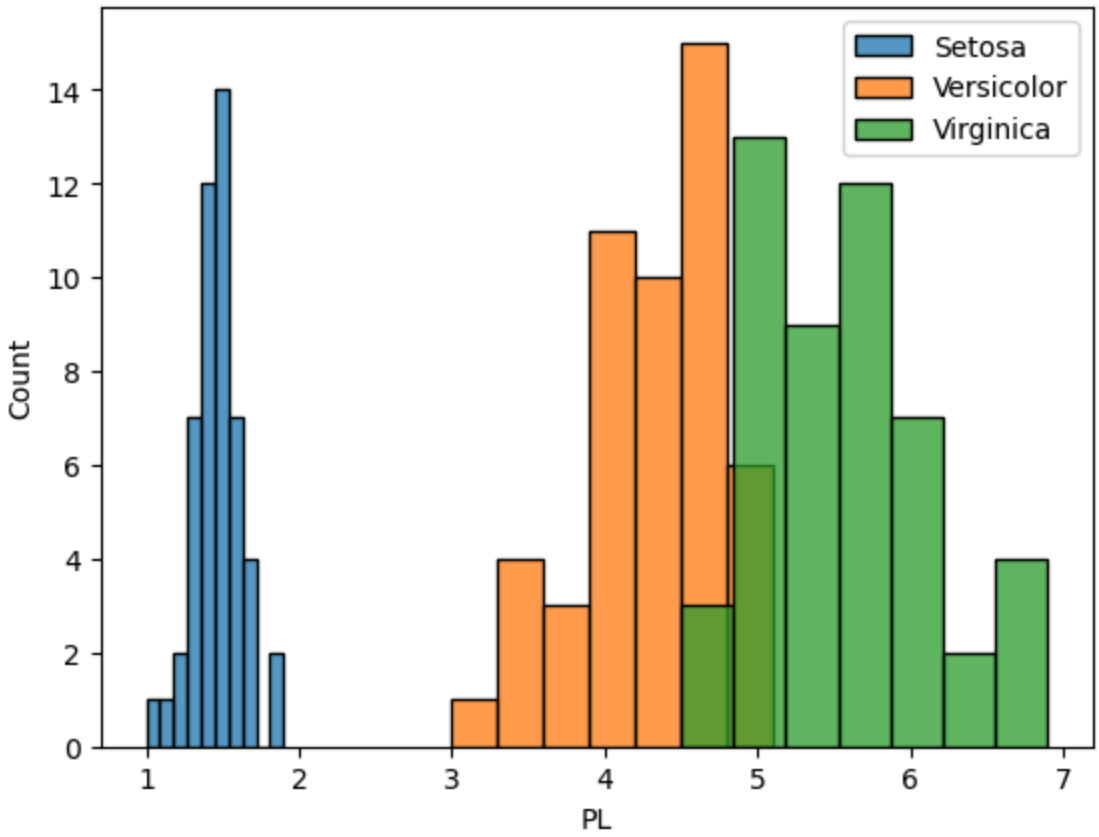
In [100…

```
sns.distplot(iris.PL[iris.FlowerType=='Iris-setosa'],label='Setosa')
sns.distplot(iris.PL[iris.FlowerType=='Iris-versicolor'],label='Versicolor')
sns.distplot(iris.PL[iris.FlowerType=='Iris-virginica'],label='Virginica')
plt.legend()
plt.show()
```



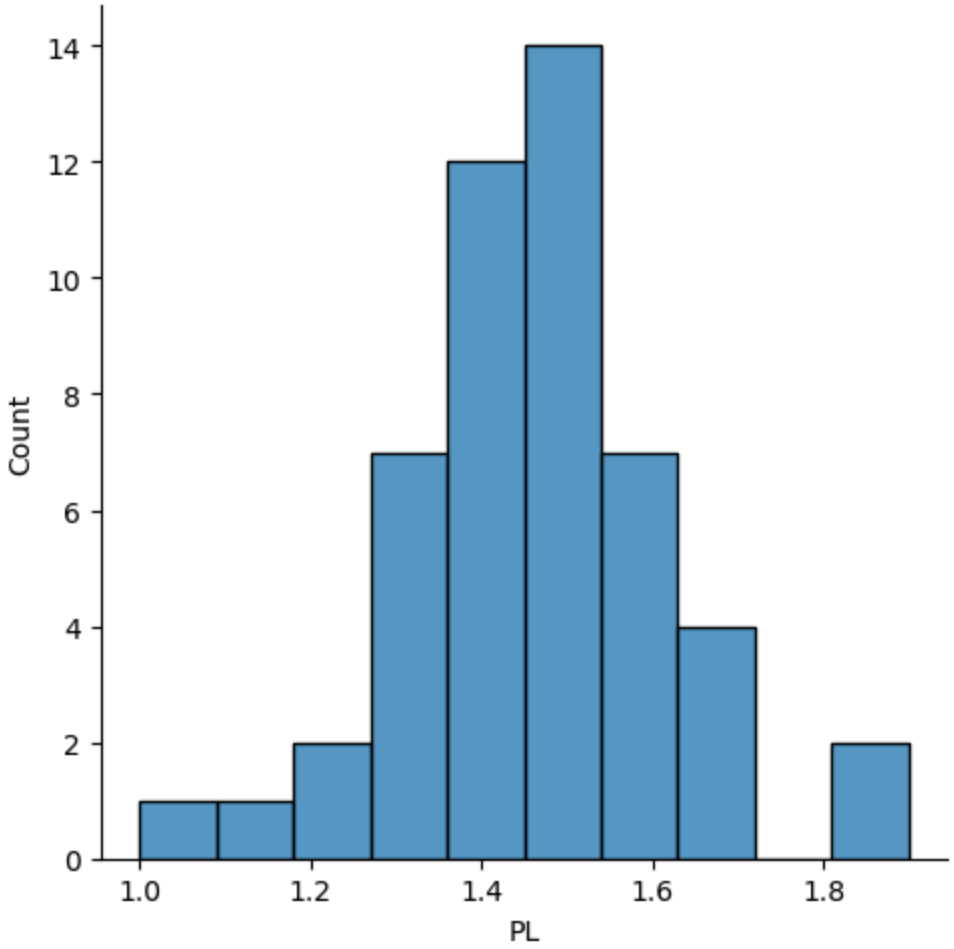
In [101…

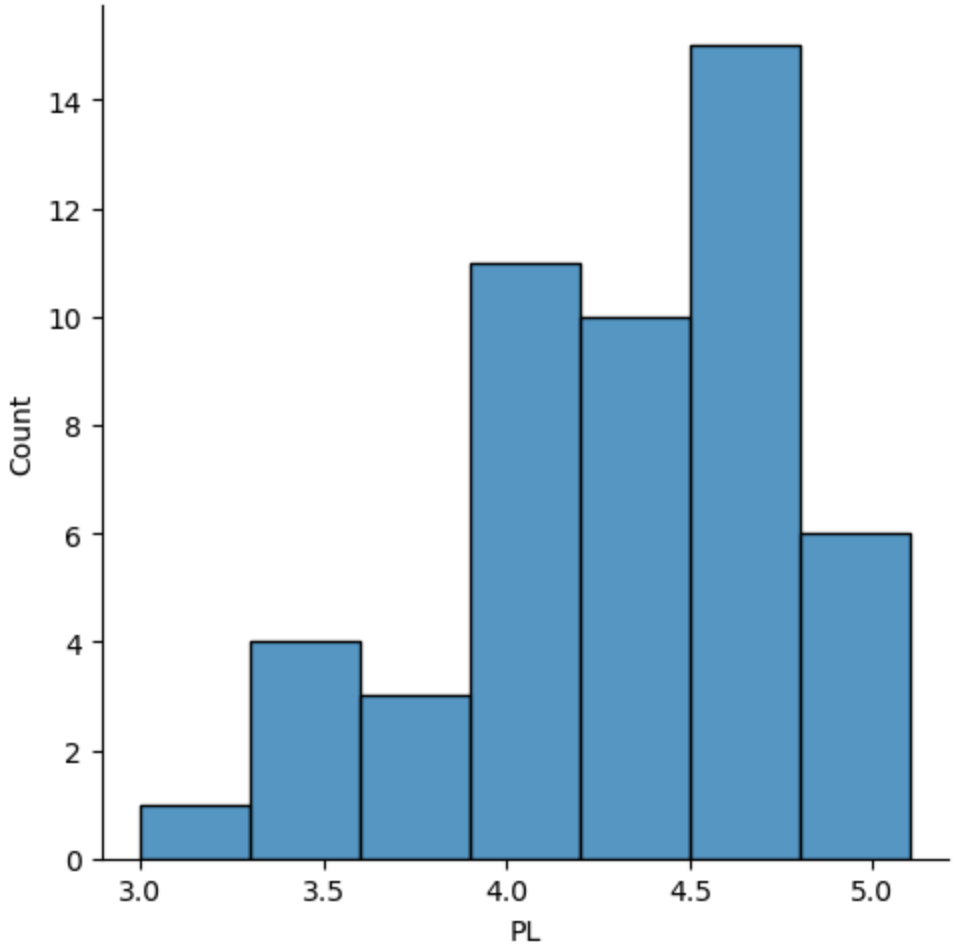
```
sns.histplot(iris.PL[iris.FlowerType=='Iris-setosa']
             ,label='Setosa')
sns.histplot(iris.PL[iris.FlowerType=='Iris-versicolor']
             ,label='Versicolor')
sns.histplot(iris.PL[iris.FlowerType=='Iris-virginica']
             ,label='Virginica')
plt.legend()
plt.show()
```

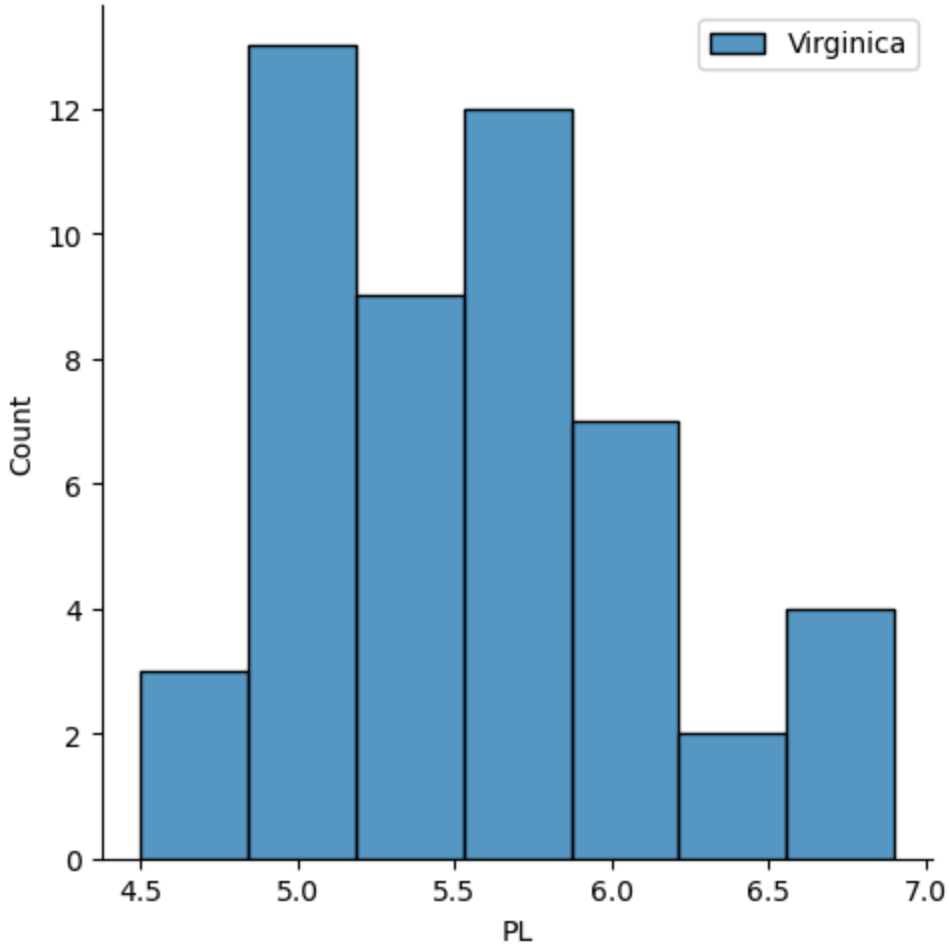


In [102…

```
sns.displot(iris.PL[iris.FlowerType=='Iris-setosa']
            ,label='Setosa')
sns.displot(iris.PL[iris.FlowerType=='Iris-versicolor']
            ,label='Versicolor')
sns.displot(iris.PL[iris.FlowerType=='Iris-virginica']
            ,label='Virginica')
plt.legend()
plt.show()
```

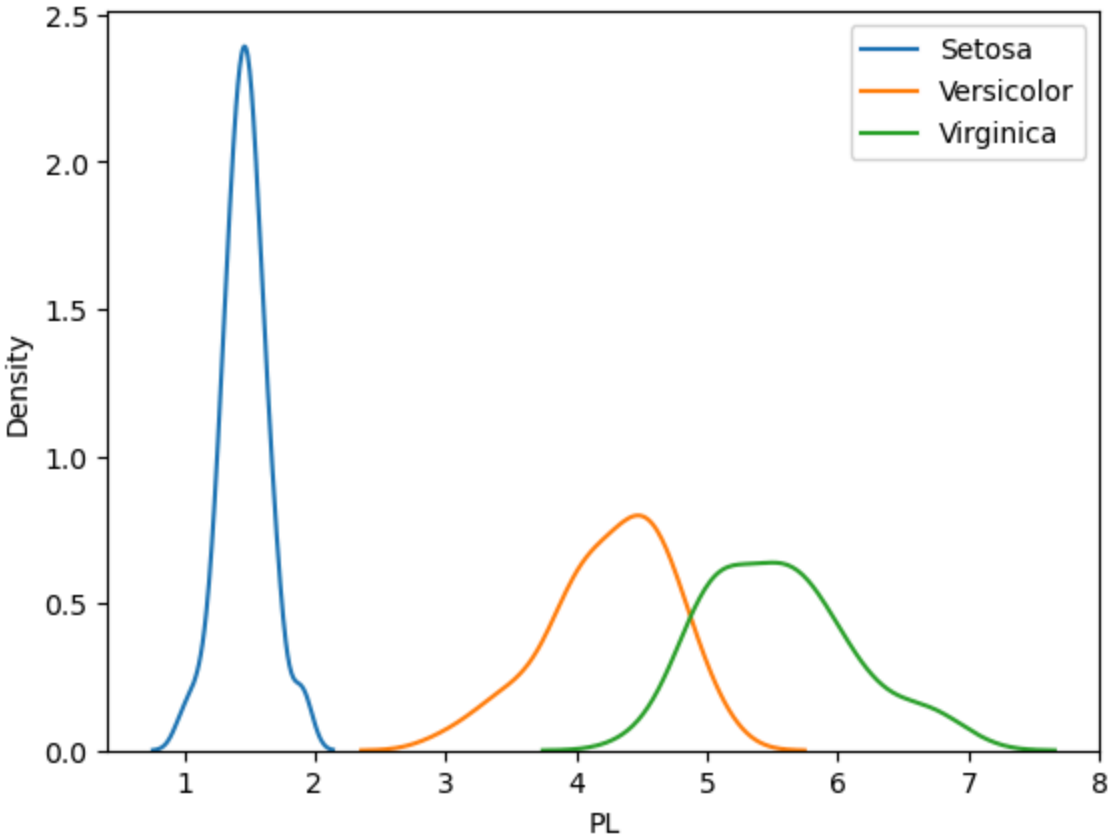

14
12
10
8
6
4
2
0
Count
1.0
1.2
1.4
1.6
1.8
PL


Count
PL
0
2
4
6
8
10
12
14
3.0
3.5
4.0
4.5
5.0



In [103…

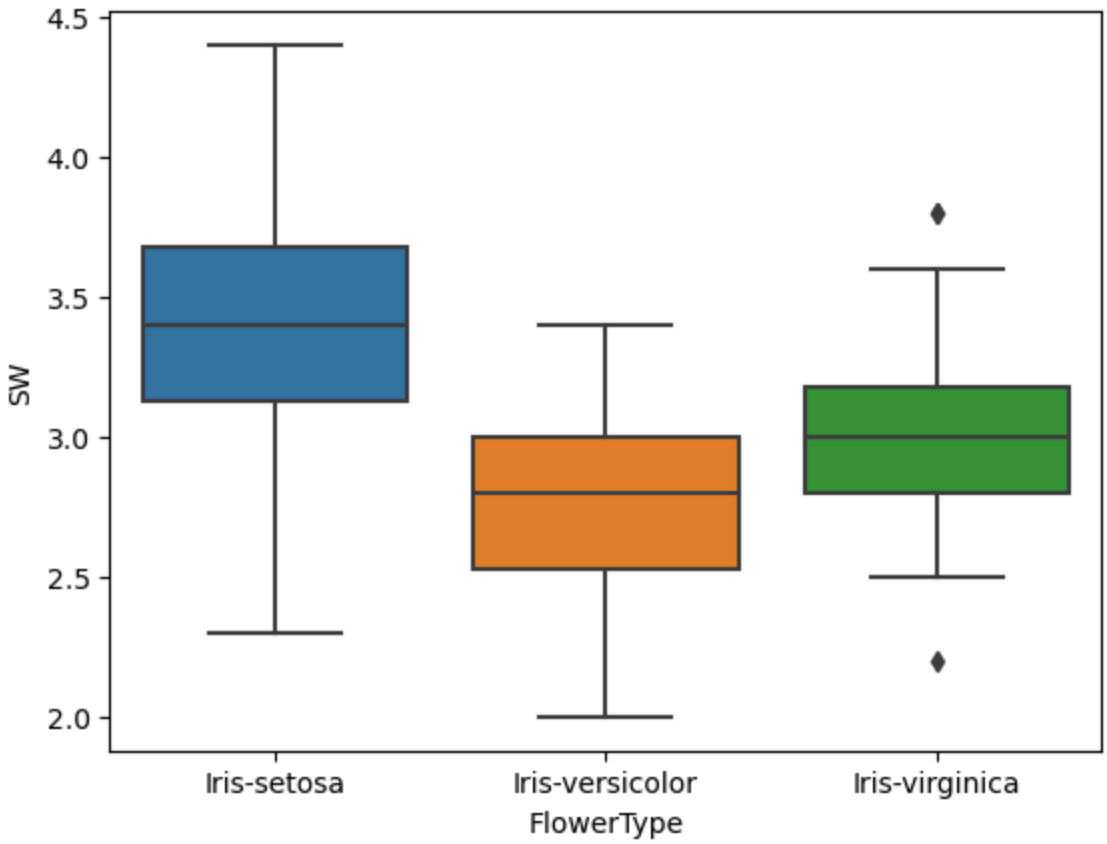
```
sns.kdeplot(iris.PL[iris.FlowerType=='Iris-setosa']
            ,label='Setosa')
sns.kdeplot(iris.PL[iris.FlowerType=='Iris-versicolor']
            ,label='Versicolor')
sns.kdeplot(iris.PL[iris.FlowerType=='Iris-virginica']
            ,label='Virginica')
plt.legend()
plt.show()
```



Box Plot:- Used to display the statistical data of the variable.

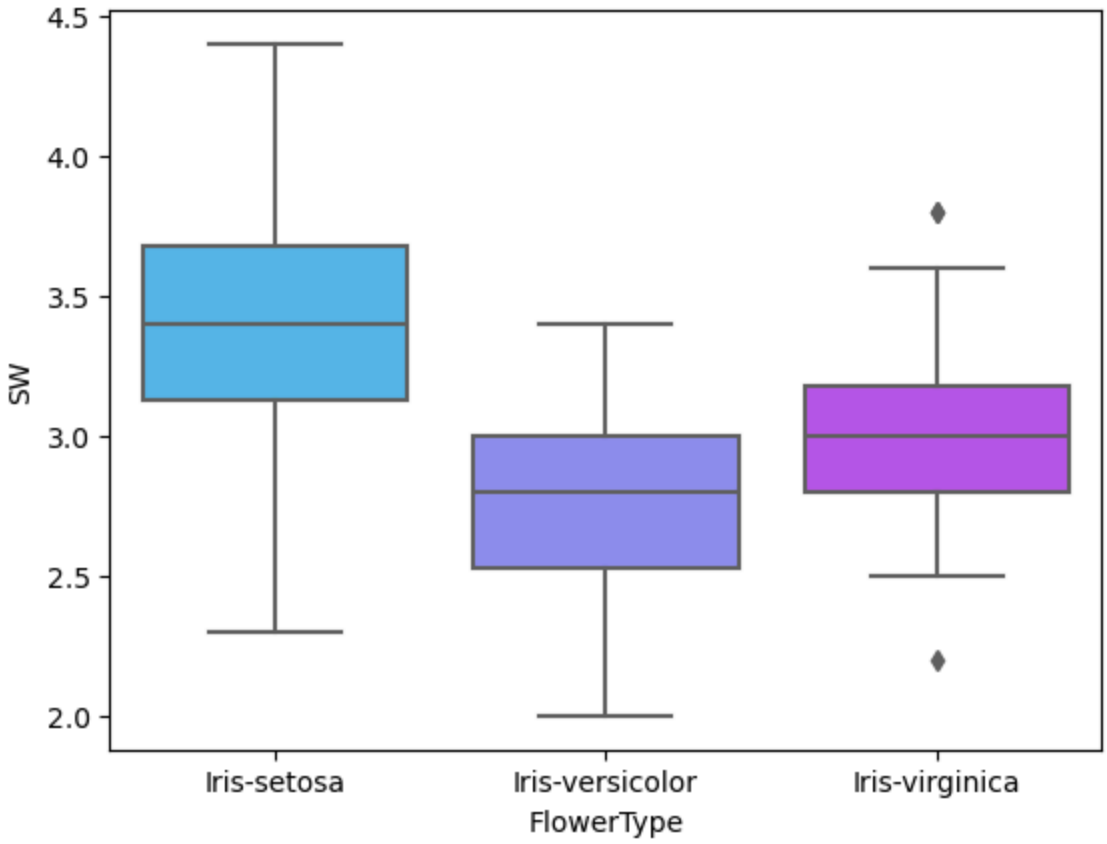
In [104…

```
sns.boxplot(x=iris.FlowerType,y=iris.SW)
plt.show()
```



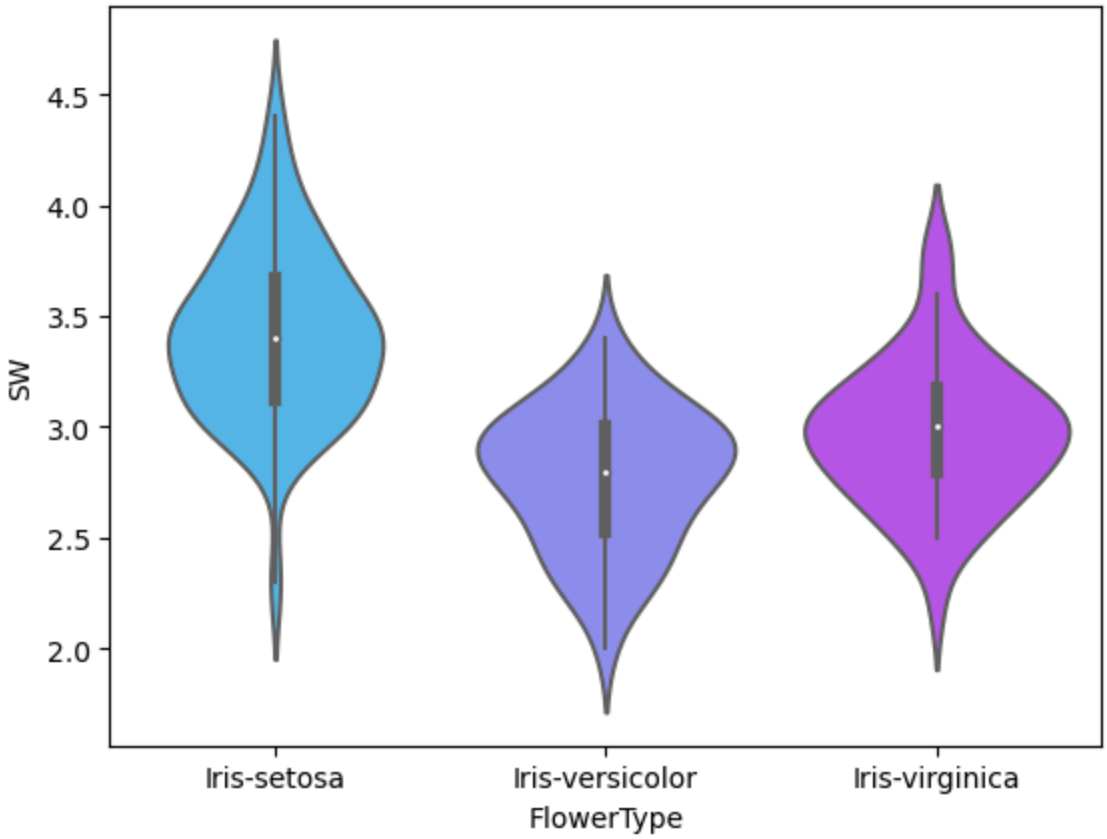
In [105…

```
sns.boxplot(x=iris.FlowerType,y=iris.SW,palette='cool')
plt.show()
```



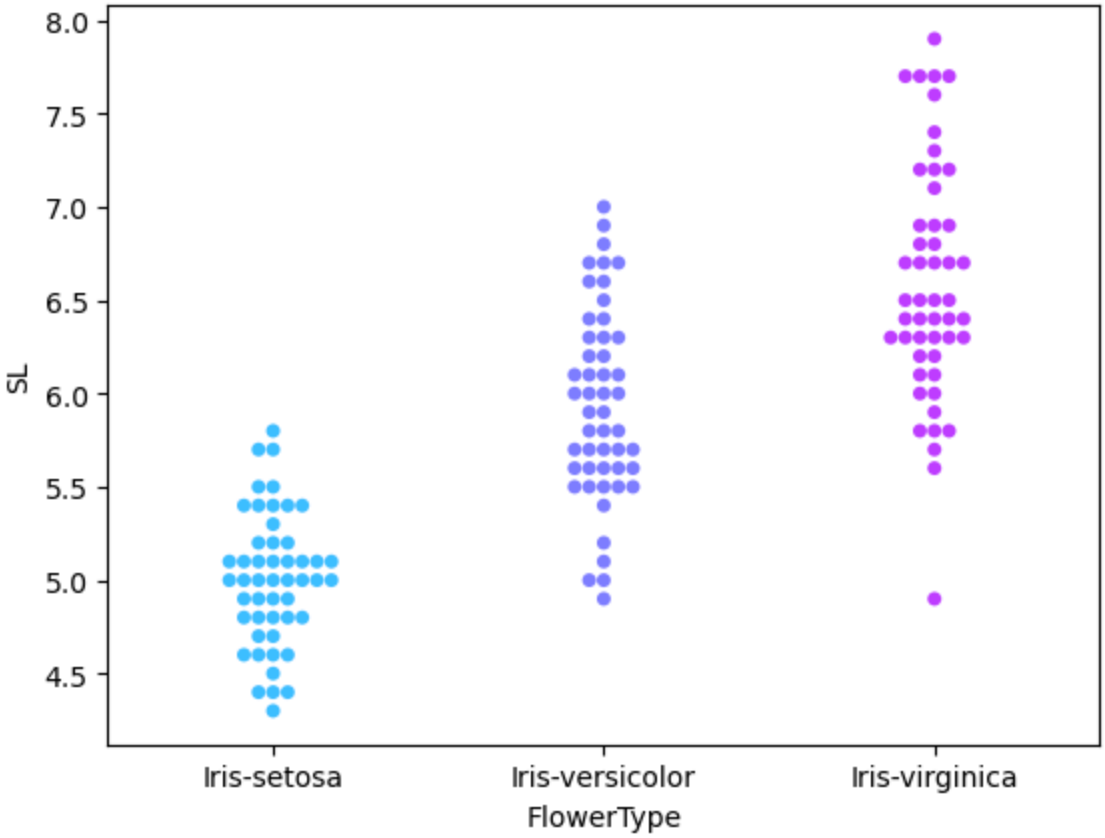
Violin plot:- used to display the distribution, median and density of the datas across the categories.

In [106…
```
sns.violinplot(x=iris.FlowerType,y=iris.SW,palette='cool')
plt.show()
```



Swarm plot:- Used to display the statistical datas of each categories using all data points.

In [107…
```
sns.swarmplot(x=iris.FlowerType,y=iris.SL,palette='cool')
plt.show()
```

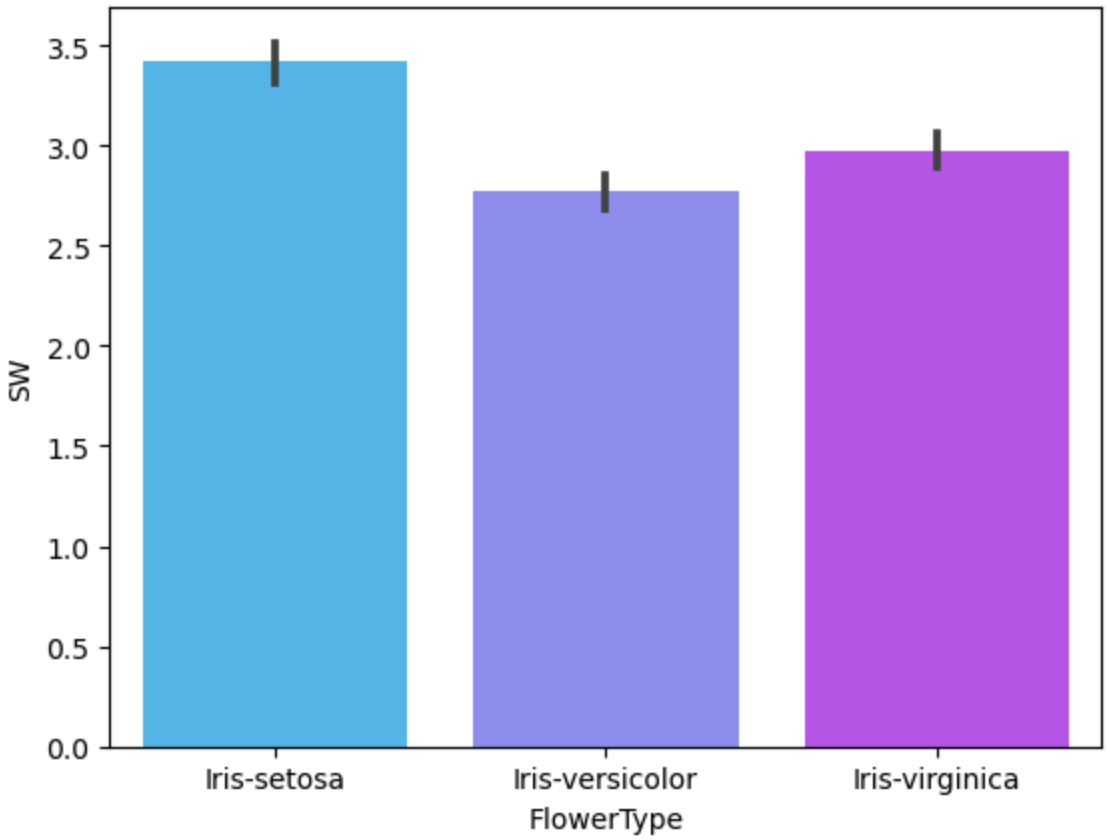


Bar plot:- Used to display the summary statistic(mean) of a variable along with a categorical data.

It automatically represents the Confidence Interval (CI) around the mean.
CI :- range of values that are actually true.

In [108…

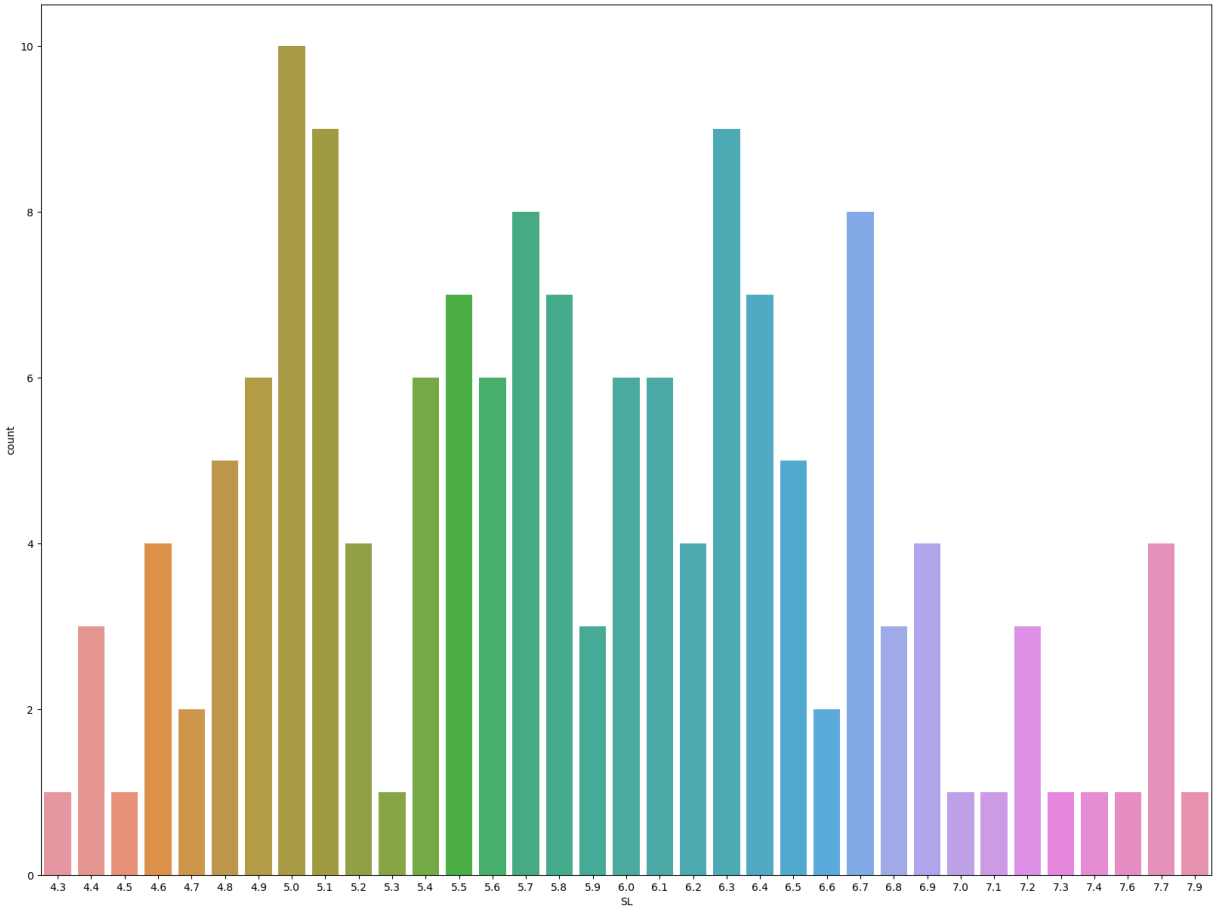
```
sns.barplot(x=iris.FlowerType,y=iris.SW,palette='cool')
plt.show()
```



Countplot:- Show the count of each categories.

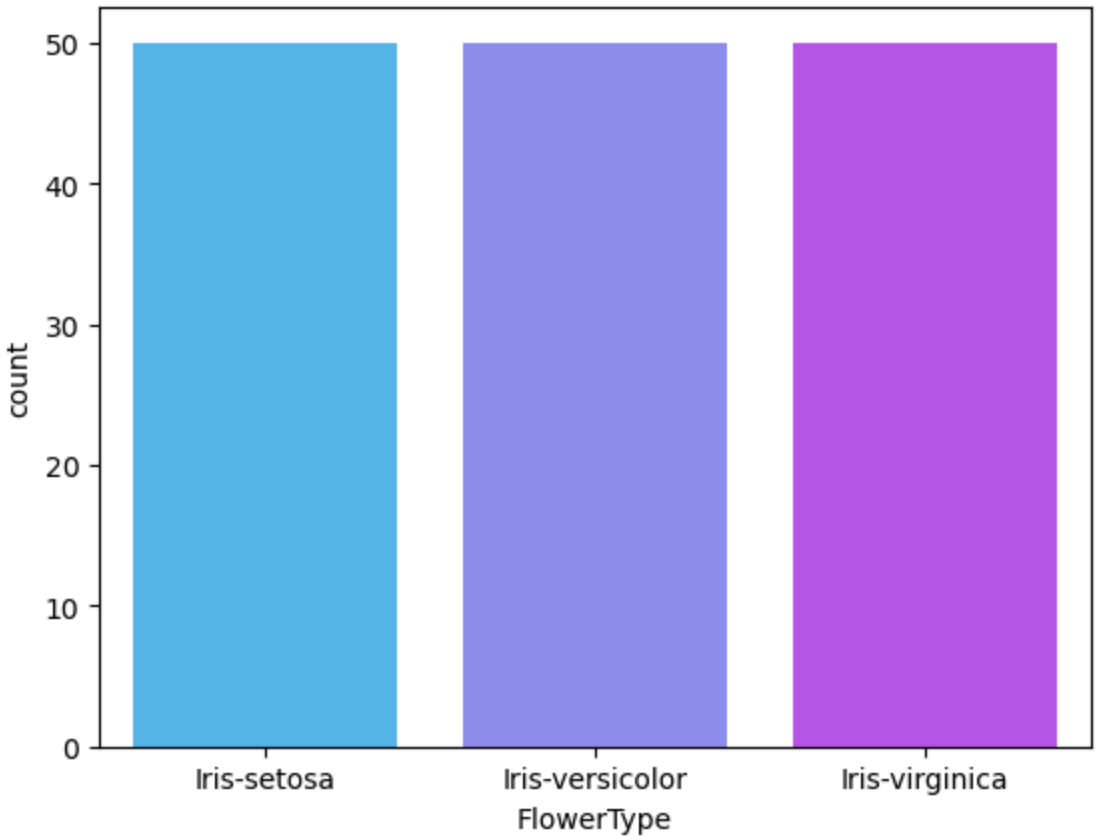
In [109…

```
plt.figure(figsize=(20,15))
sns.countplot(x=iris.SL)
plt.show()
```



In [110…

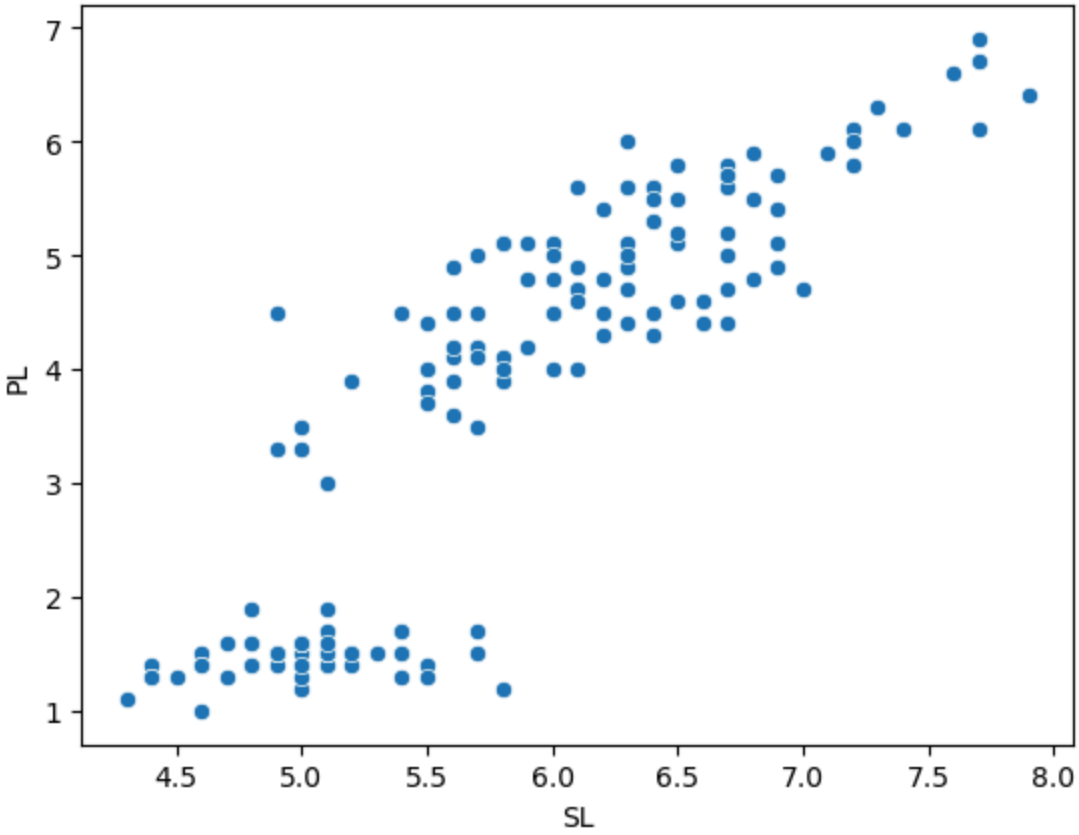
```
sns.countplot(x=iris.FlowerType,data=iris,palette='cool')
plt.show()
```



Scatter plot:- used to display the relationship between 2 numerical variables.

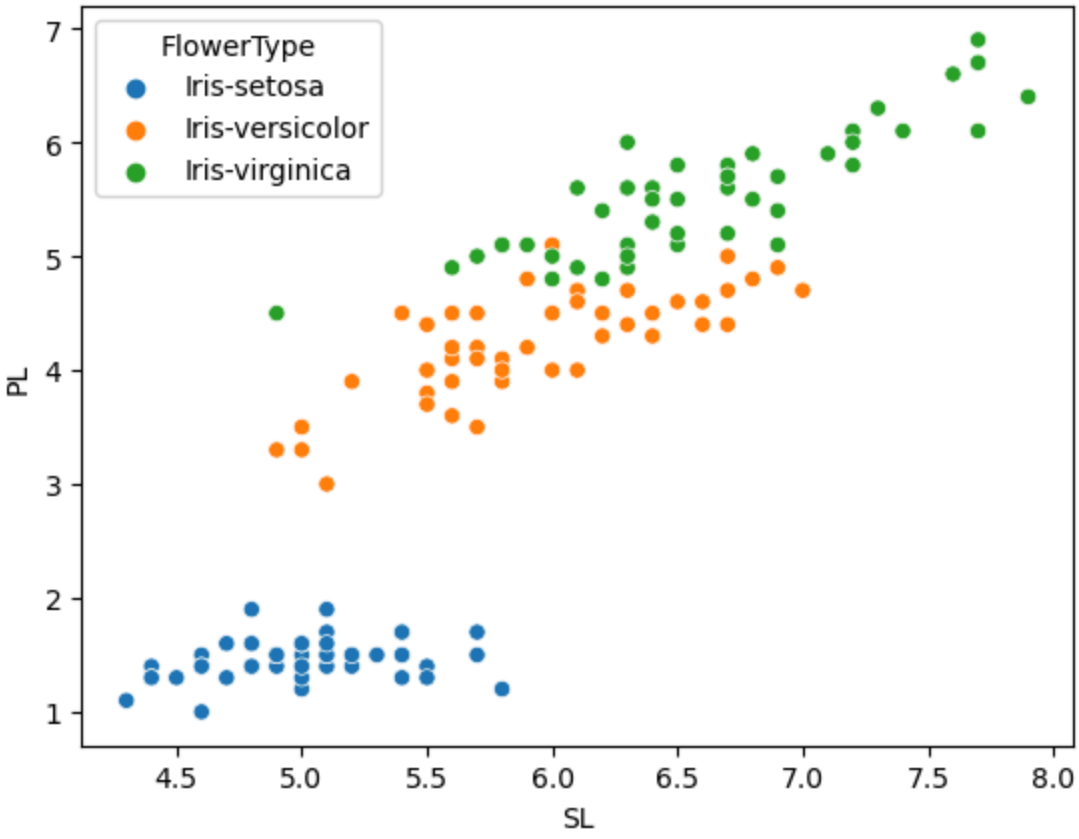
In [111…

```
sns.scatterplot(x=iris.SL,y=iris.PL)
plt.show()
```



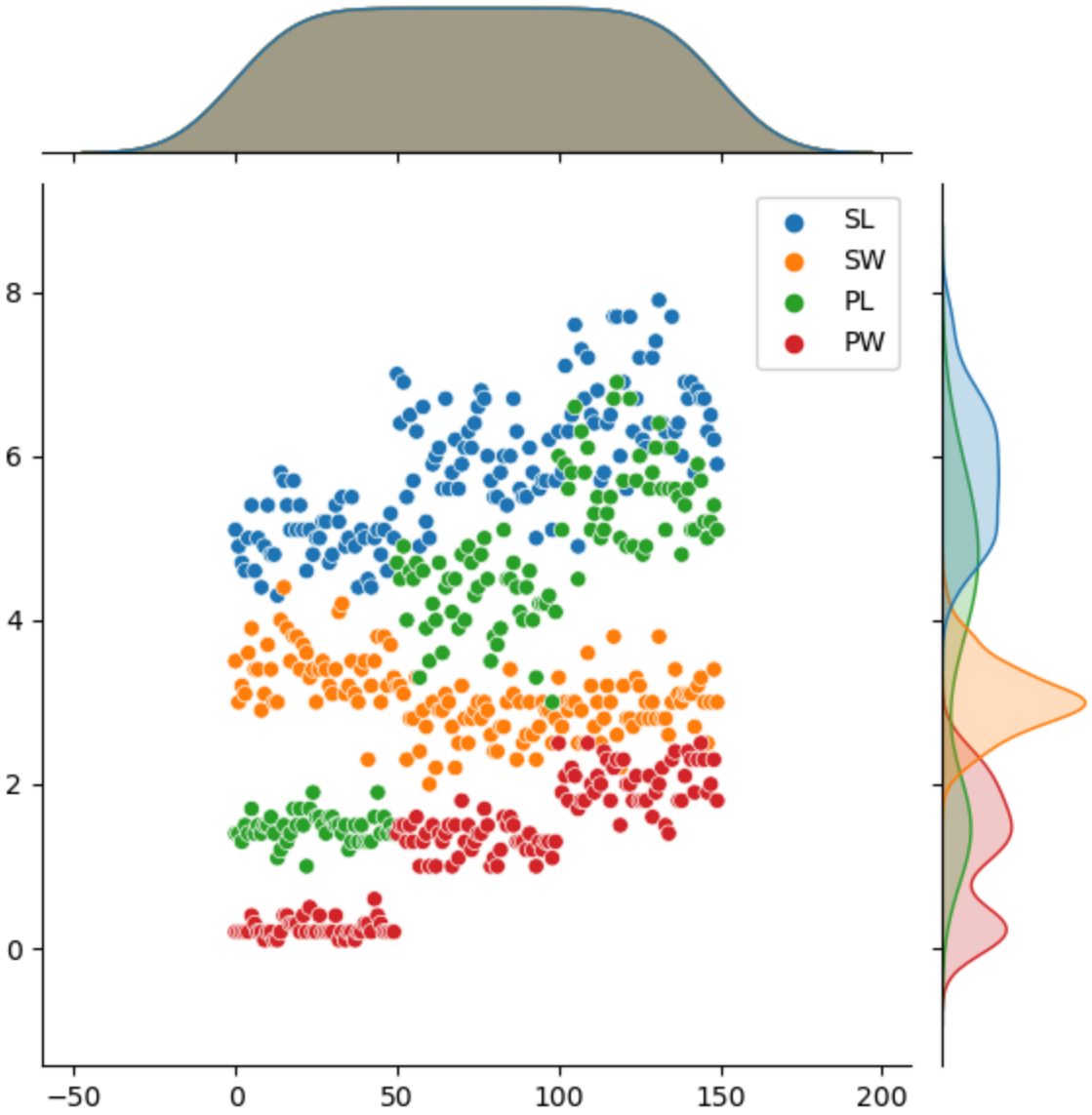
In [112…

```
sns.scatterplot(x=iris.SL,y=iris.PL,hue=iris.FlowerType)
plt.show()
```

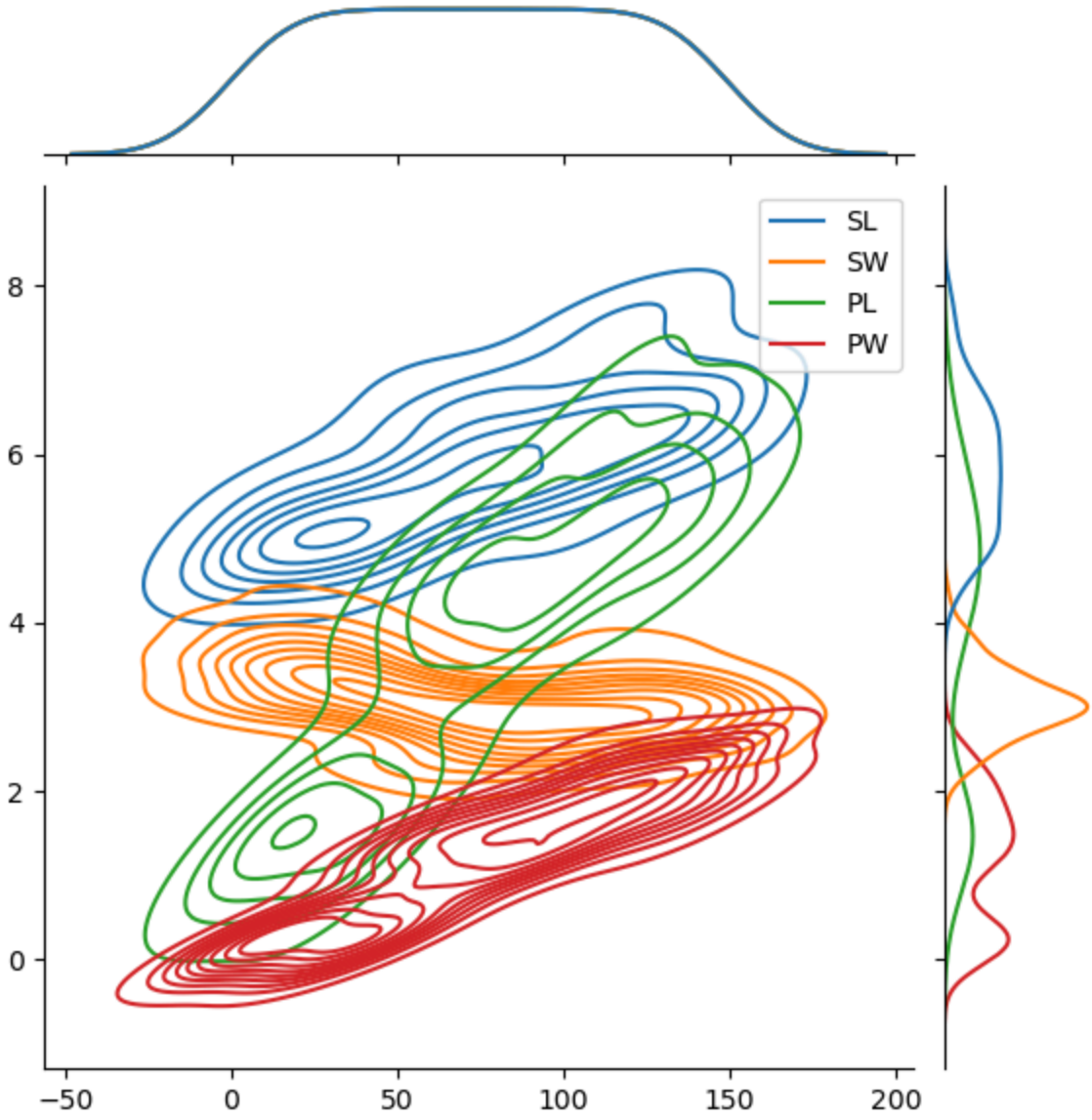


Joint plot:- used to display the bivariate scatter plot of each variable.

In [113…
```
sns.jointplot(data=iris)
plt.show()
```



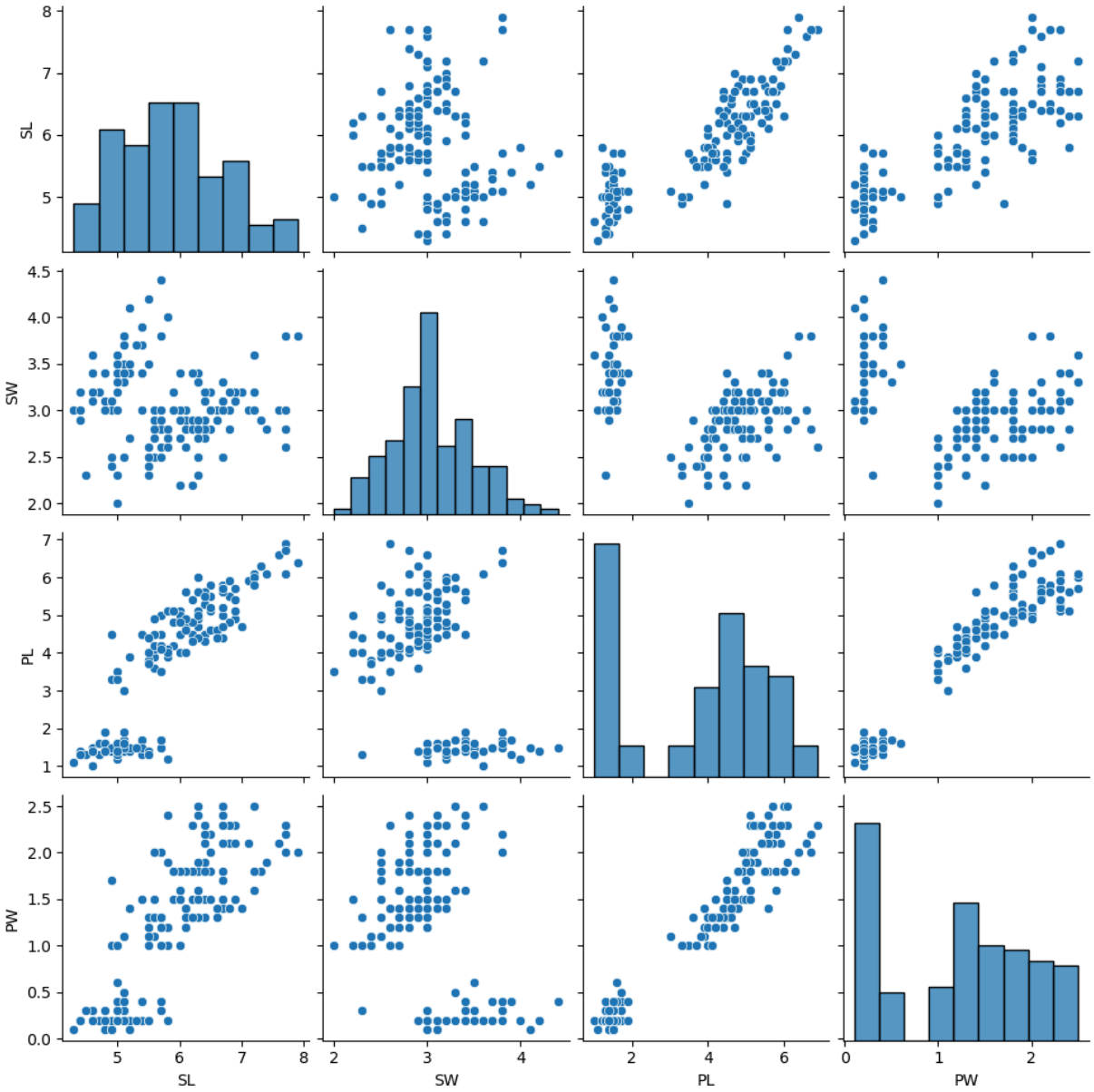
In [114…
```
sns.jointplot(data=iris,kind='kde')
plt.show()
```



Pair plot:- creates a grid of plots that displays the pairwise relationship between mulitple numerical columns in the dataset.

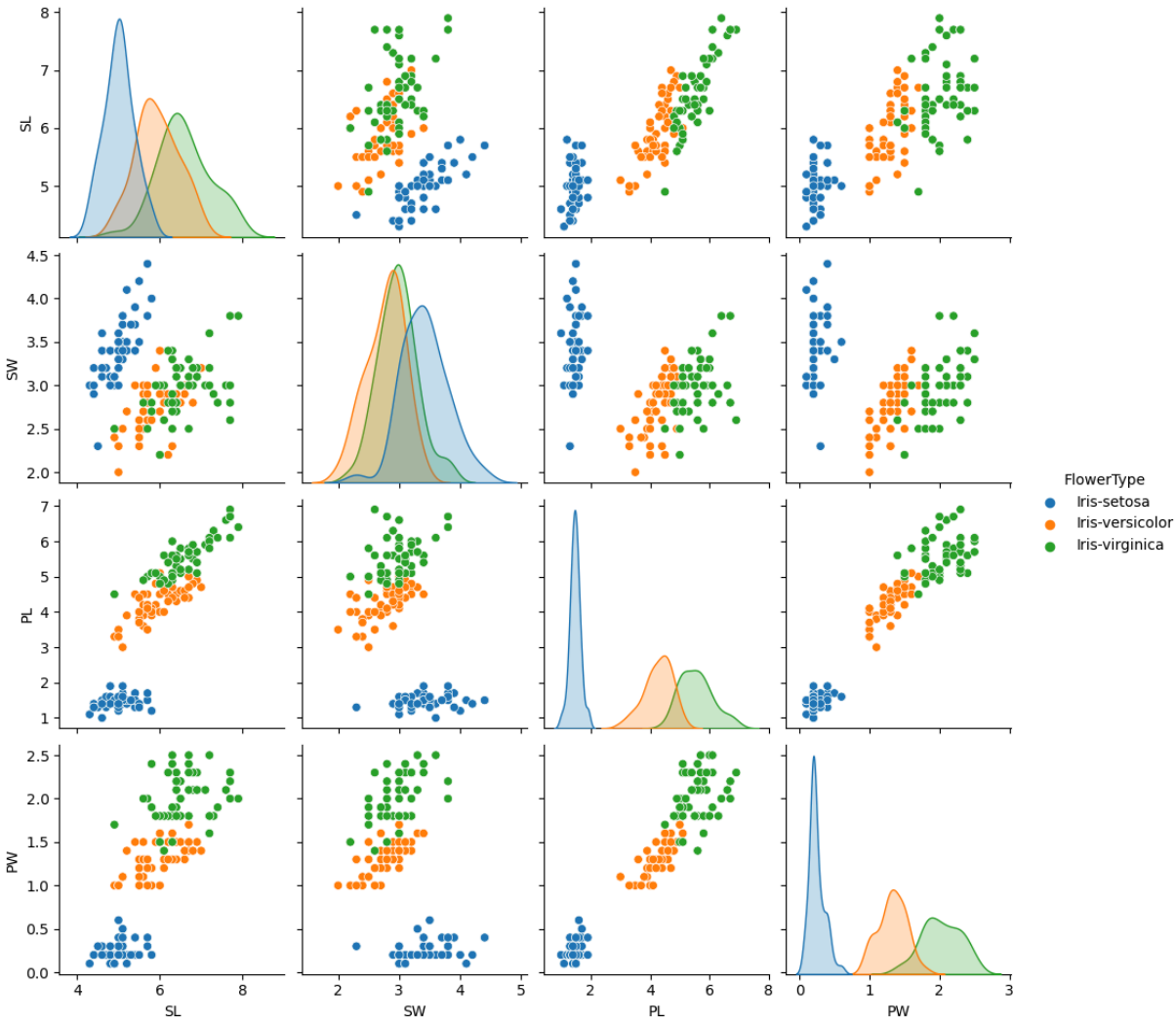
In [115…

```
sns.pairplot(iris)
plt.show()
```



In [116…

```
sns.pairplot(iris,hue='FlowerType')
plt.show()
```



Heat Map:- shows the correlation of the datas.

In [117…

```
c = iris.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[117], line 1
----> 1 c = iris.corr()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:10704, in DataF
rame.corr(self, method, min_periods, numeric_only)
  10702 cols = data.columns
  10703 idx = cols.copy()
> 10704 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10706 if method == "pearson":
  10707     correl = libalgos.nancorr(mat, minp=min_periods)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:1889, in DataFr
ame.to_numpy(self, dtype, copy, na_value)
   1887 if dtype is not None:
   1888     dtype = np.dtype(dtype)
-> 1889 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1890 if result.dtype is not dtype:
   1891     result = np.array(result, dtype=dtype, copy=False)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:16
56, in BlockManager.as_array(self, dtype, copy, na_value)
   1654         arr.flags.writeable = False
   1655 else:
-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)
   1657     # The underlying data was copied within _interleave, so no need
   1658     # to further copy if copy=True or setting na_value
   1660 if na_value is lib.no_default:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:17
15, in BlockManager._interleave(self, dtype, na_value)
   1713     else:
   1714         arr = blk.get_values(dtype)
-> 1715     result[rl.indexer] = arr
   1716     itemmask[rl.indexer] = 1
   1718 if not itemmask.all():

ValueError: could not convert string to float: 'Iris-setosa'
```

In [5]:
```
# eliminate the string(object) column 'FlowerType'.
iris.drop('FlowerType',axis=1,inplace=True)
iris.head()
```

Out[5]:

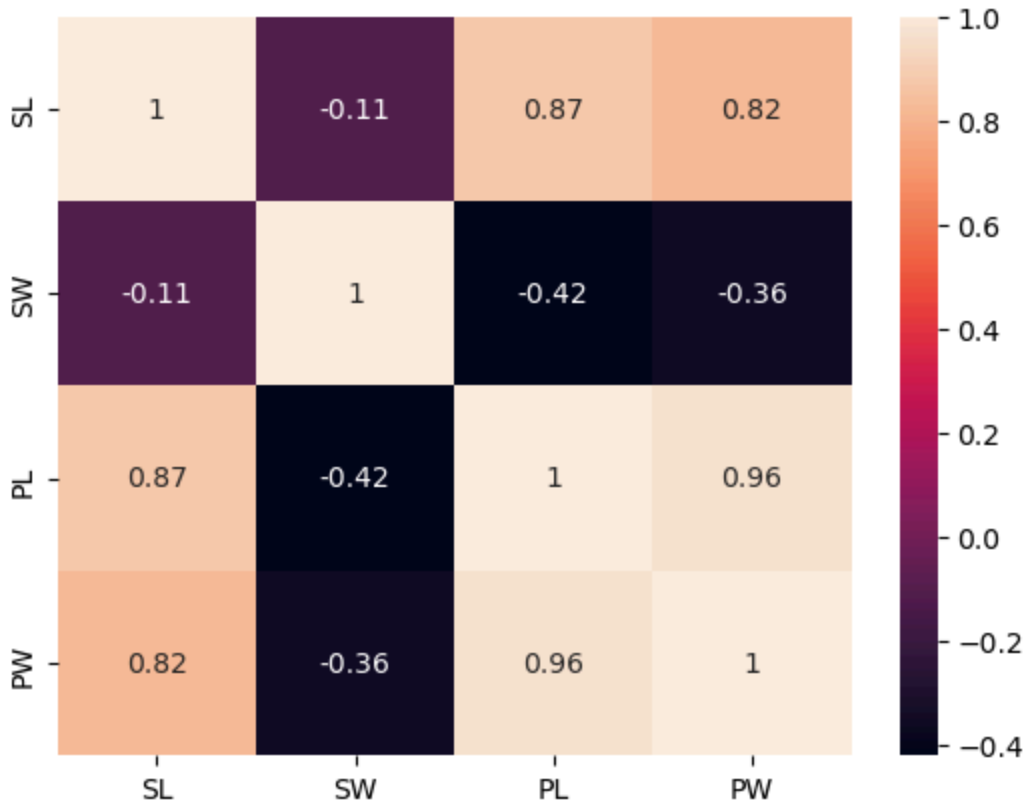
	SL	SW	PL	PW
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [6]:
```
c = iris.corr()
c
```

Out[6]:

	SL	SW	PL	PW
SL	1.000000	-0.109369	0.871754	0.817954
SW	-0.109369	1.000000	-0.420516	-0.356544
PL	0.871754	-0.420516	1.000000	0.962757
PW	0.817954	-0.356544	0.962757	1.000000

In [7]:
```
sns.heatmap(c,annot=True)
plt.show()
```



In [ ]:
```
!pip install seaborn --upgrade
```

In [ ]:
```

```