In [ ]:

In [ ]:

Matplotlib

Matplotlib is a Data Visualization Technique in Python.

Matplotlib is a Python library for data visualization, primarily used to create static, animated, and interactive plots. It provides a wide range of plotting functions to visualize data effectively.

****Markers****

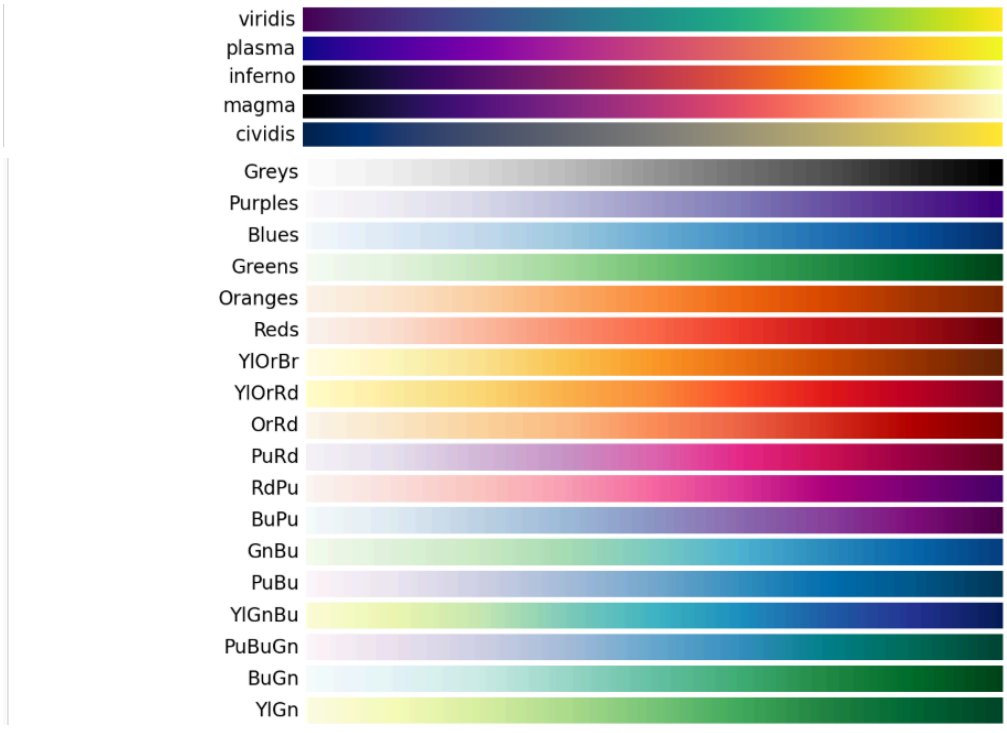
character	description
``'.'``	point marker
``','``	pixel marker
``'o'``	circle marker
``'v'``	triangle_down marker
``'^'``	triangle_up marker
``'<'``	triangle_left marker
``'>'``	triangle_right marker
``'1'``	tri_down marker
``'2'``	tri_up marker
``'3'``	tri_left marker
``'4'``	tri_right marker
``'8'``	octagon marker
``'s'``	square marker
``'p'``	pentagon marker
``'P'``	plus (filled) marker
``'*'``	star marker
``'h'``	hexagon1 marker
``'H'``	hexagon2 marker
``'+'``	plus marker
``'x'``	x marker
``'X'``	x (filled) marker
``'D'``	diamond marker
``'d'``	thin_diamond marker
``'|'``	vline marker
``'_'``	hline marker

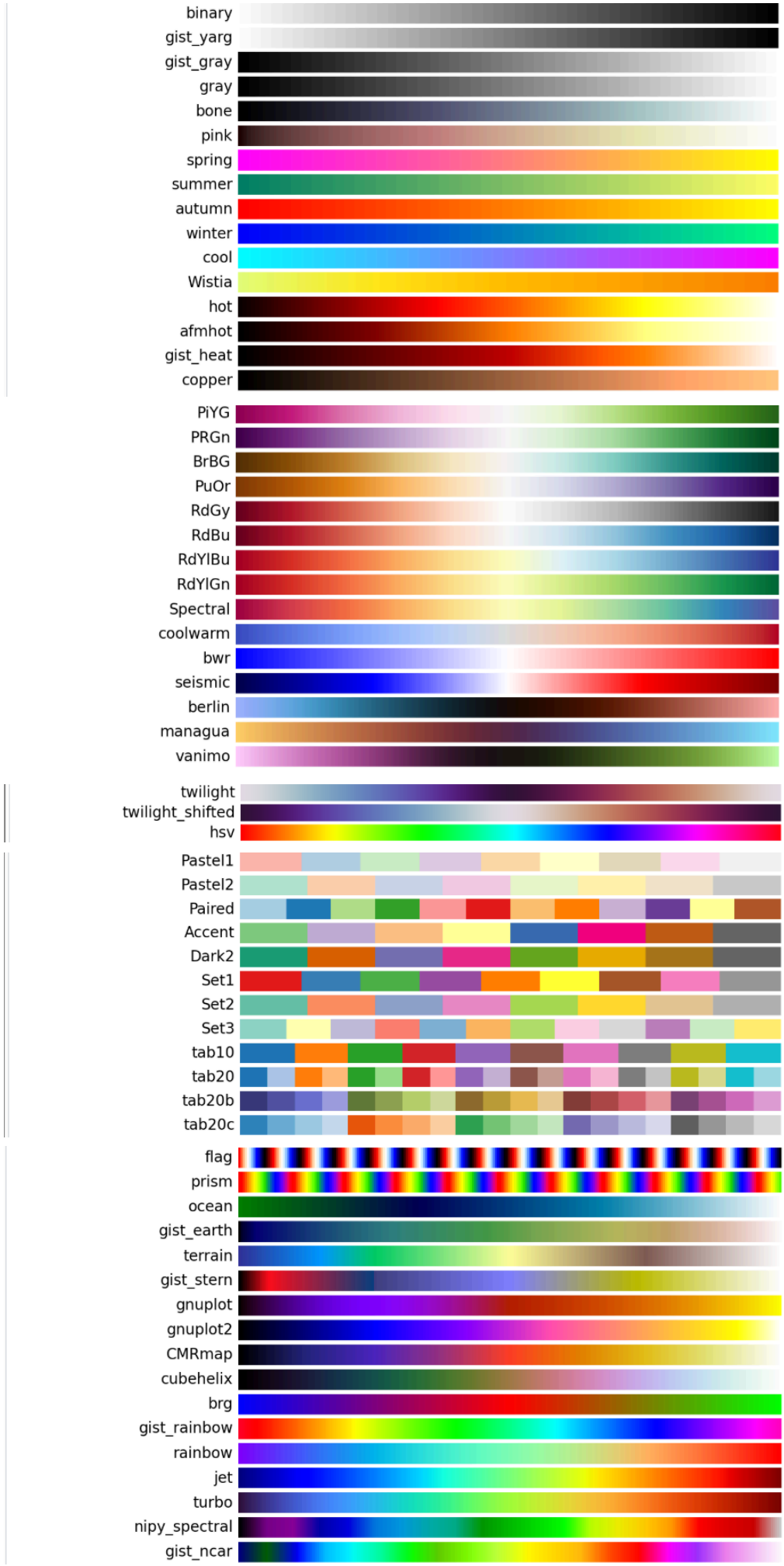
****Line Styles****

character	description
``'-'``	solid line style
``'--'``	dashed line style
``'-.'``	dash-dot line style

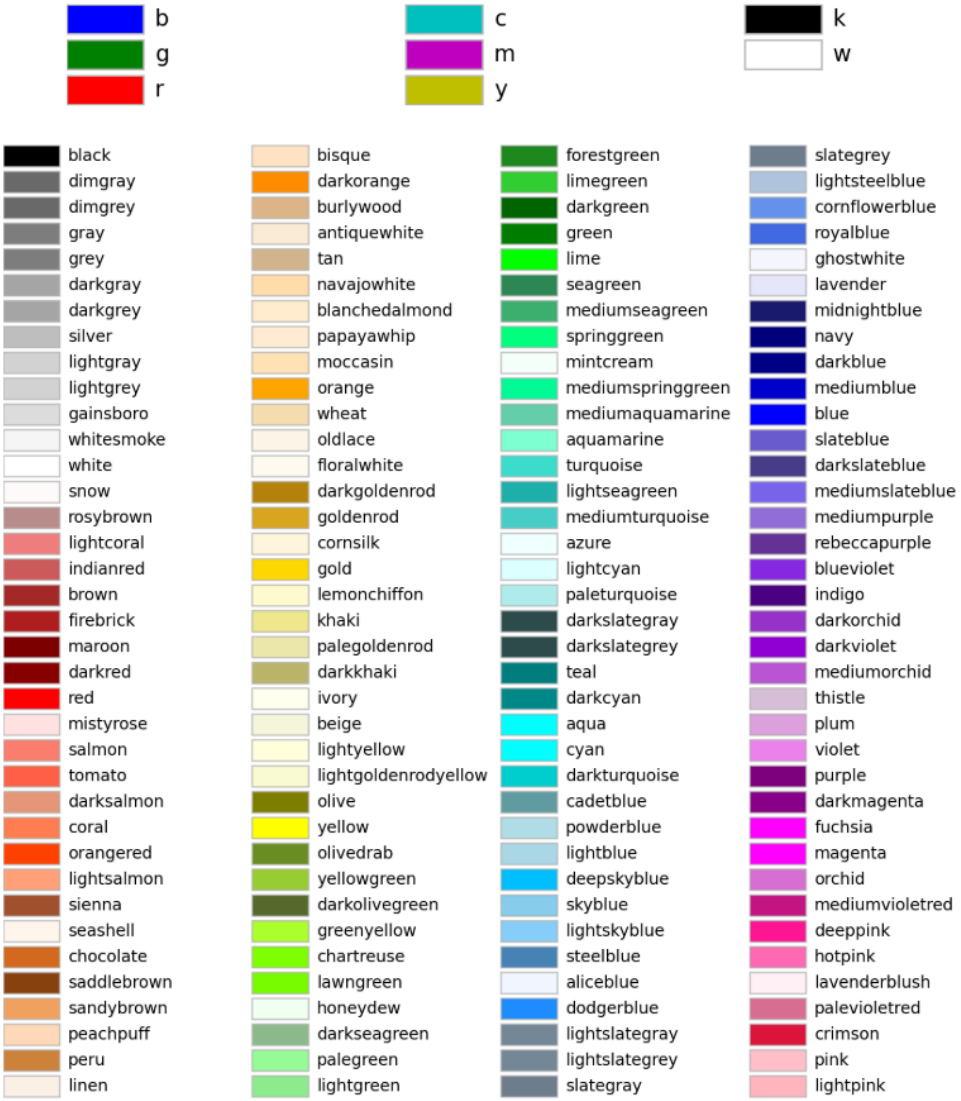
```
``':'``          dotted line style
=============    ===============================

Example format strings::

    'b'    # blue markers with default shape
    'or'   # red circles
    '-g'   # green solid line
    '--'   # dashed line with default color
    '^k:'  # black triangle_up markers connected by a dotted
line

**Colors**

The supported color abbreviations are the single letter codes

=============    ===============================
character        color
=============    ===============================
``'b'``          blue
``'g'``          green
``'r'``          red
``'c'``          cyan
``'m'``          magenta
``'y'``          yellow
``'k'``          black
``'w'``          white
=============    ===============================
```

Color Palettes



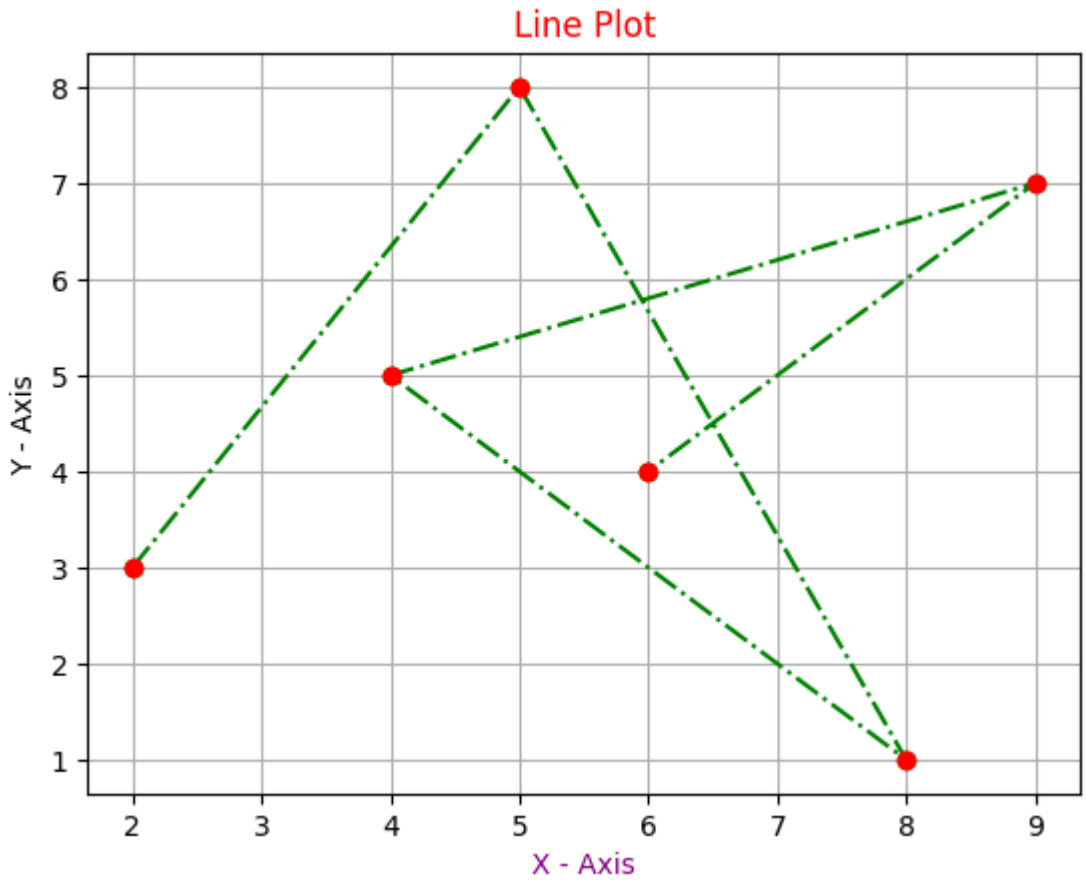

binary
gist_yarg
gist_gray
gray
bone
pink
spring
summer
autumn
winter
cool
Wistia
hot
afmhot
gist_heat
copper
PiYG
PRGn
BrBG
PuOr
RdGy
RdBu
RdYlBu
RdYlGn
Spectral
coolwarm
bwr
seismic
berlin
managua
vanimo
twilight
twilight_shifted
hsv
Pastel1
Pastel2
Paired
Accent
Dark2
Set1
Set2
Set3
tab10
tab20
tab20b
tab20c
flag
prism
ocean
gist_earth
terrain
gist_stern
gnuplot
gnuplot2
CMRmap
cubehelix
brg
gist_rainbow
rainbow
jet
turbo
nipy_spectral
gist_ncar

Colors List



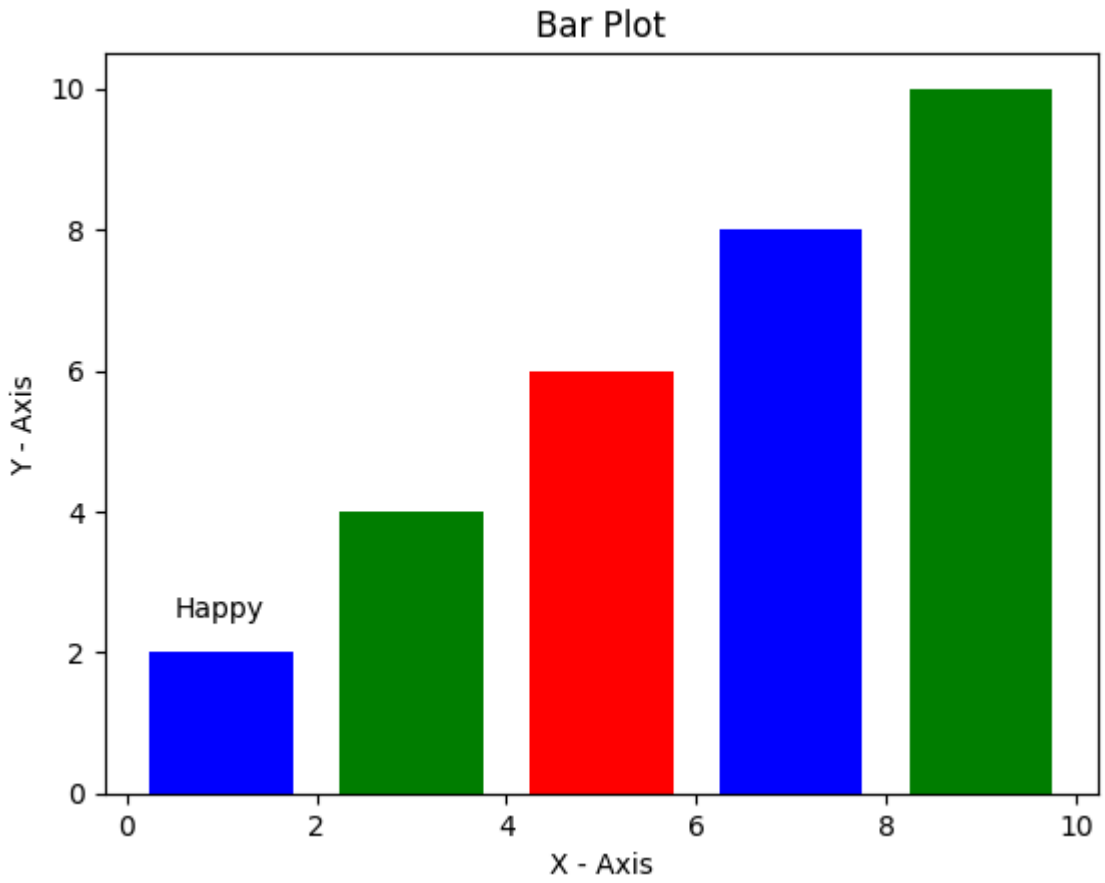
In [5]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
#Line Plot
x = [2,5,8,4,9,6]
y = [3,8,1,5,7,4]

plt.plot(x,y,'og-.')
plt.plot(x,y,'or')
plt.title("Line Plot",color='r')
plt.xlabel("X - Axis",color='purple')
plt.ylabel("Y - Axis")
plt.grid()
plt.show()
```

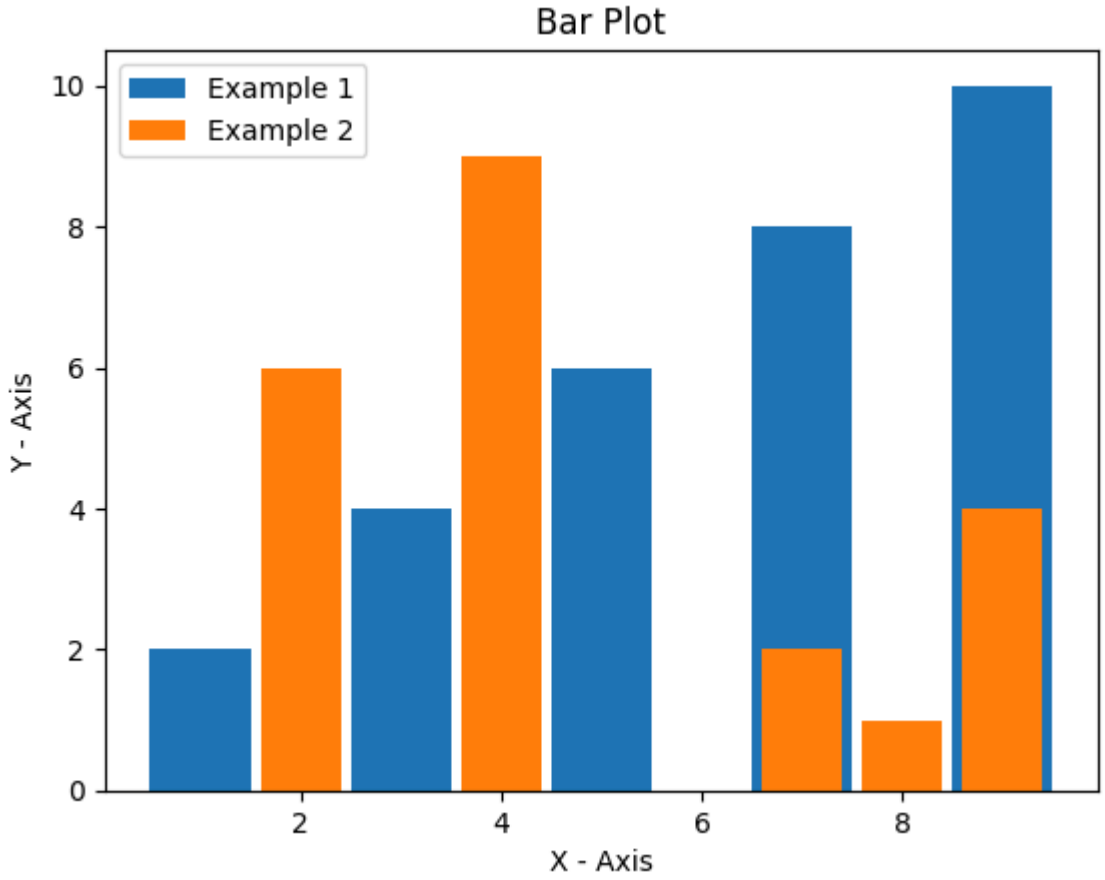


In [10]:
```
#help(plt.plot)
```

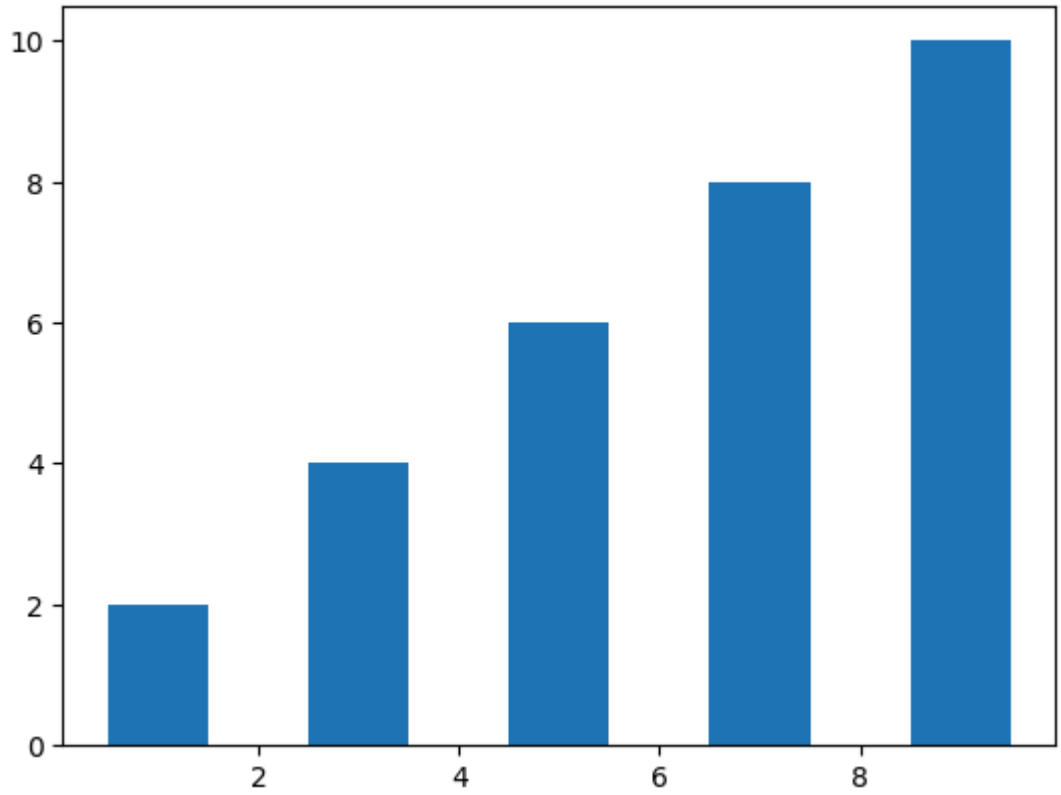
In [60]:
```
#Bar Plot
x = [1,3,5,7,9]
y = [2,4,6,8,10]

plt.bar(x,y,color=['b','g','r'],width=1.5)
plt.title("Bar Plot")
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.annotate("Happy",xy=(0.5,2.5))
plt.show()
```

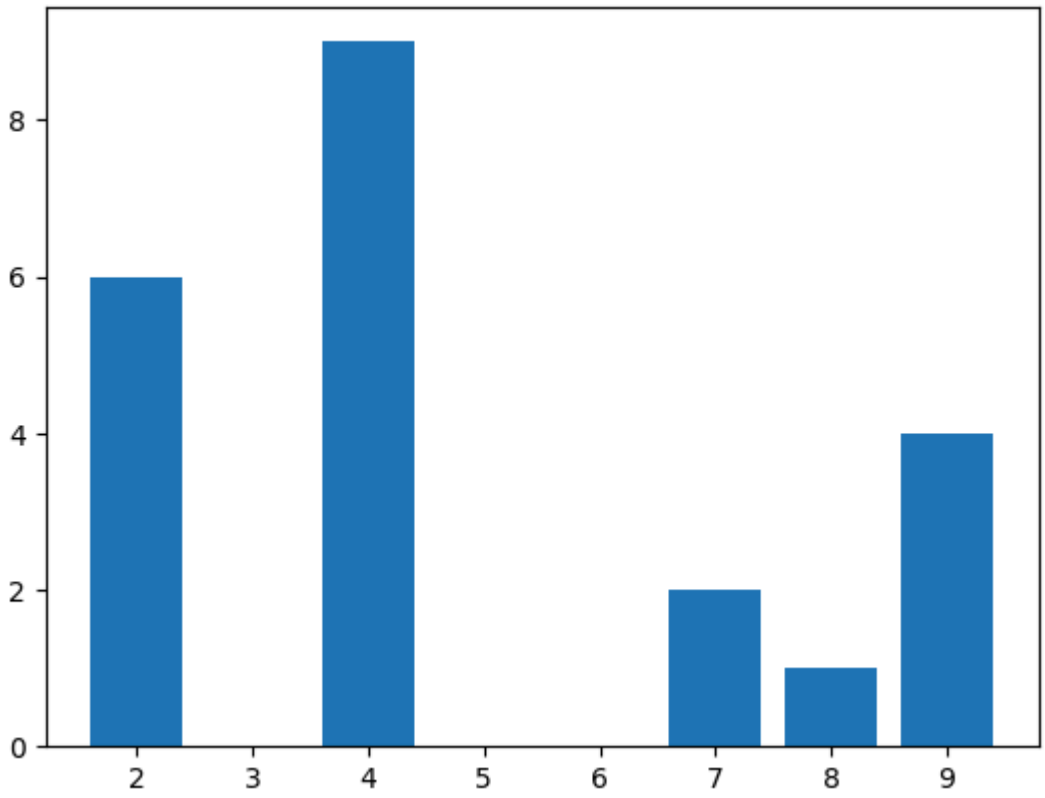


In [65]:
```
#Multiple Bar Chart
x = [1,3,5,7,9]
y = [2,4,6,8,10]
plt.bar(x,y,width=1,label='Example 1')

x1 = [4,7,2,9,8]
y1 = [9,2,6,4,1]
plt.bar(x1,y1,label='Example 2')

plt.title("Bar Plot")
plt.legend()
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```

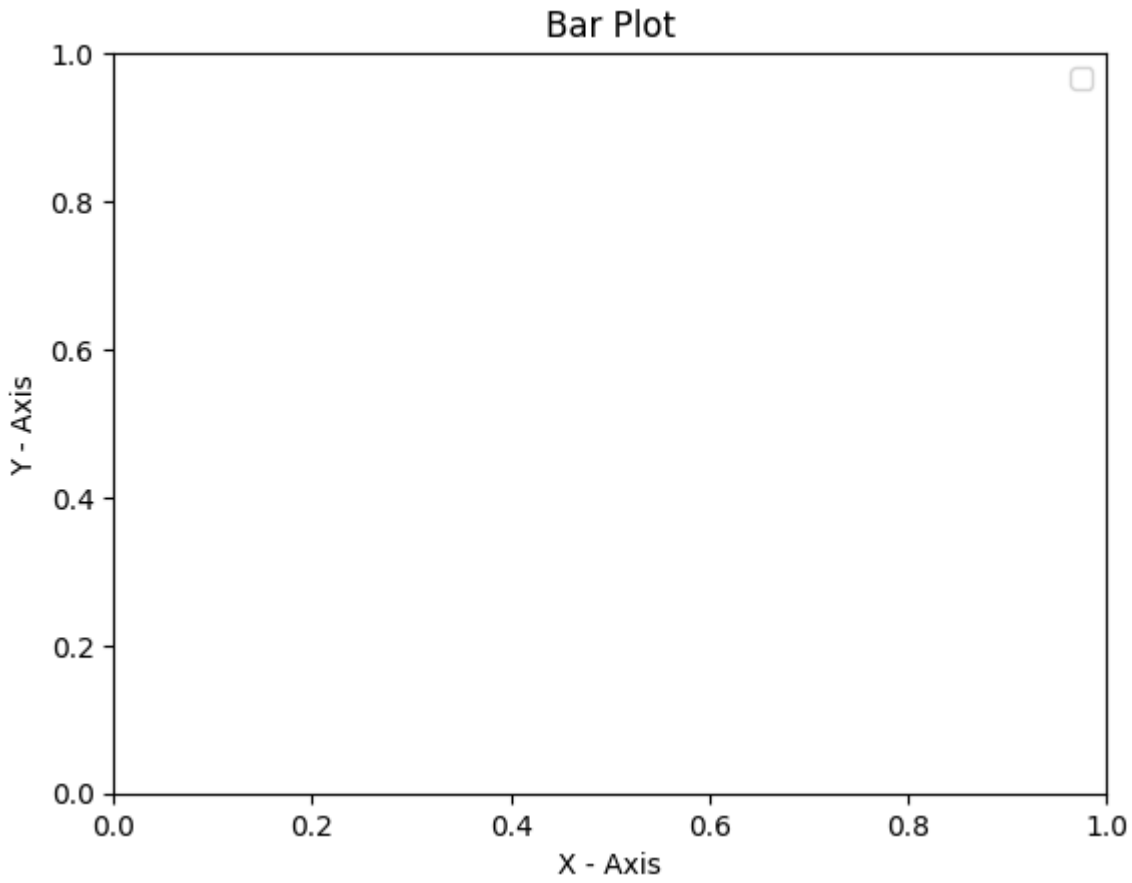


In [66]:
```
#Multiple Bar Chart
x = [1,3,5,7,9]
y = [2,4,6,8,10]
plt.bar(x,y,width=1,label='Example 1')
plt.show()

x1 = [4,7,2,9,8]
y1 = [9,2,6,4,1]
plt.bar(x1,y1,label='Example 2')
plt.show()

plt.title("Bar Plot")
plt.legend()
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```

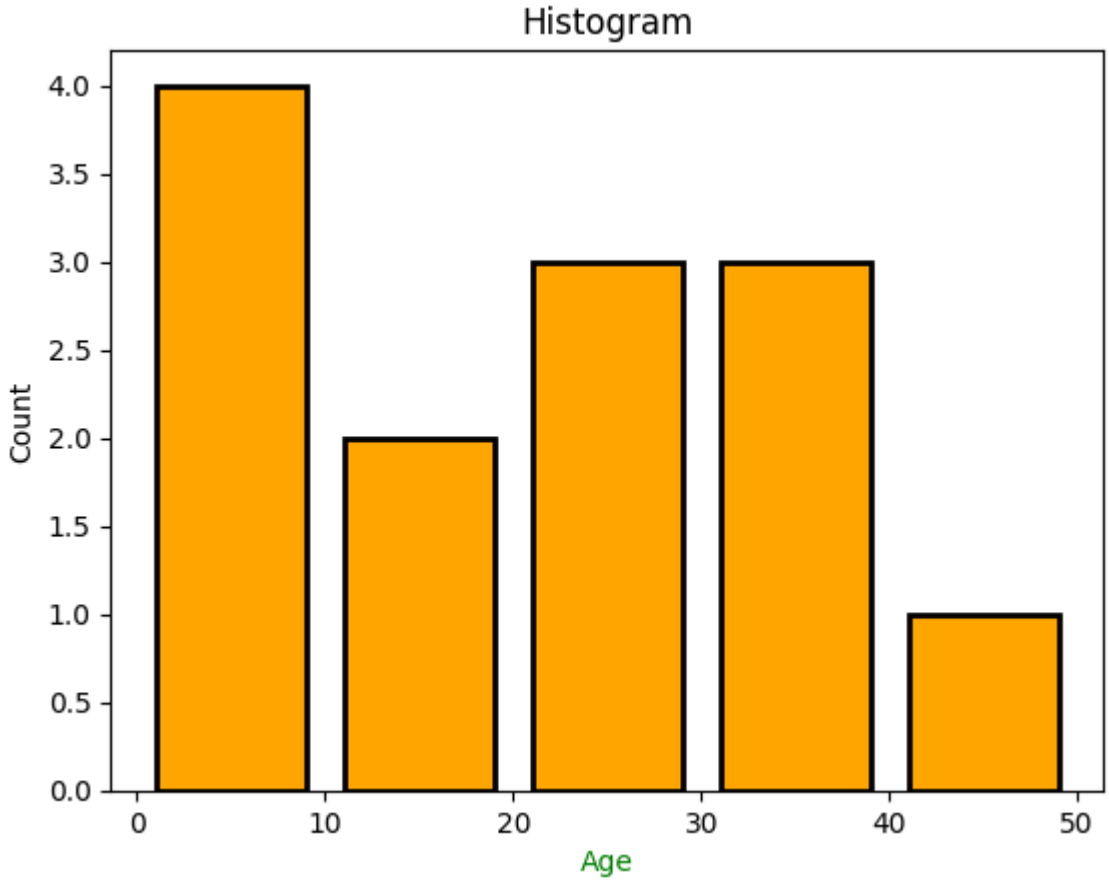





```
C:\Users\VICTUS\AppData\Local\Temp\ipykernel_3388\1623033510.py:13: UserWarning:
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```

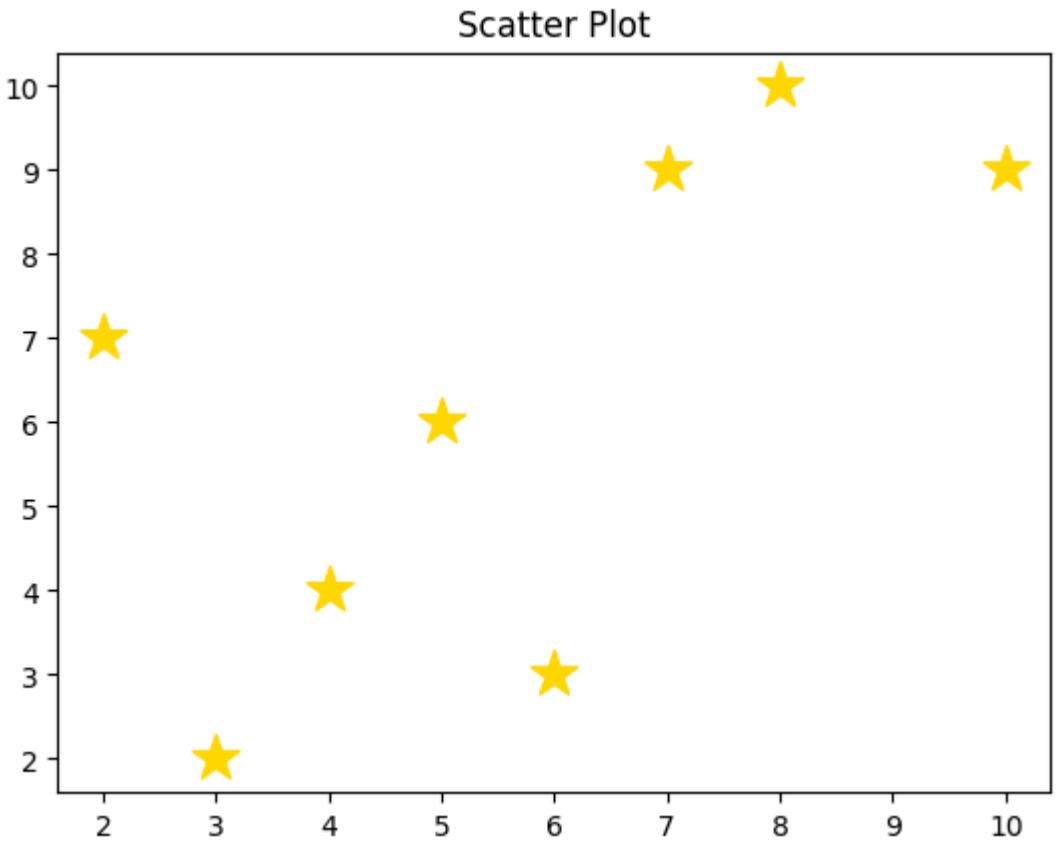


In [78]:
```
#Histogram
ages = [10,5,30,25,18,45,28,20,35,4,7,3,32]
bins = [0,10,20,30,40,50]

plt.hist(ages,bins,rwidth=0.8,edgecolor='k',linewidth=2,color='orange')
plt.title("Histogram")
plt.xlabel("Age",color='g')
plt.ylabel("Count")
plt.show()
```



In [103…

```
#Scatter Plot
x = [3,5,4,7,2,10,6,8]
y = [2,6,4,9,7,9,3,10]

plt.scatter(x,y,s=300,color='gold',marker='*')
plt.title("Scatter Plot")
plt.show()
```

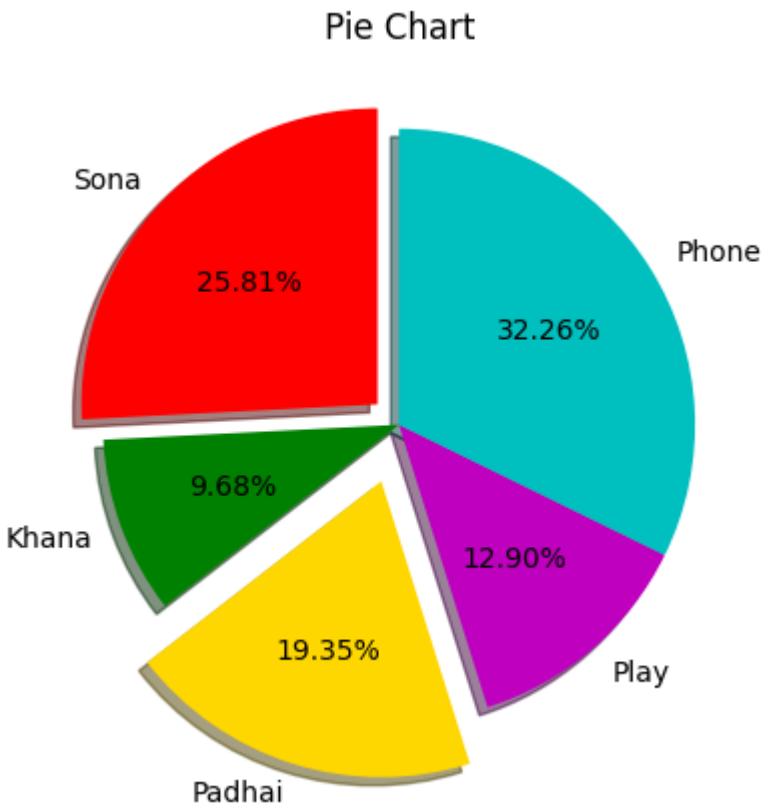


In [115…

```
x = [3,5,4,7,2,10,6,8]
y = [2,6,4,9,7,9,3,10]

plt.bar(x,y,color='g',width=0.05)
plt.scatter(x,y,s=300,color='gold',marker='*',edgecolors='orange',linewidths=10)
plt.scatter(x,y,color='brown')

plt.title("Flower Plot")
plt.show()
```

In [128…

```
#Pie Chart
slices = [8,3,6,4,10]
activities = ['Sona','Khana','Padhai','Play','Phone']
cols = ['r','g','gold','m','c']

plt.pie(slices,labels=activities,colors=cols,startangle=90,
       shadow=True,explode=(0.1,0,0.2,0,0),autopct='%1.2f%%')

plt.title("Pie Chart")
plt.show()
```



In [ ]:

Seaborn

Statistical Data Visualization

Seaborn is a Python data visualization library based on matplotlib.

It provides a high-level interface for drawing attractive and informative statistical graphics.

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

In [3]:
```
iris = pd.read_csv(r"C:\Users\VICTUS\Downloads\iris\iris.data",header=None)
iris.head()
```

Out[3]:

	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

In [4]:
```
iris.columns = ['SL','SW','PL','PW','Flower']
iris.head(2)
```

Out[4]:

	SL	SW	PL	PW	Flower
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

In [5]:
```
#Data Cleaning
iris.isnull().sum()
```

Out[5]:
```
SL        0
SW        0
PL        0
PW        0
Flower    0
dtype: int64
```

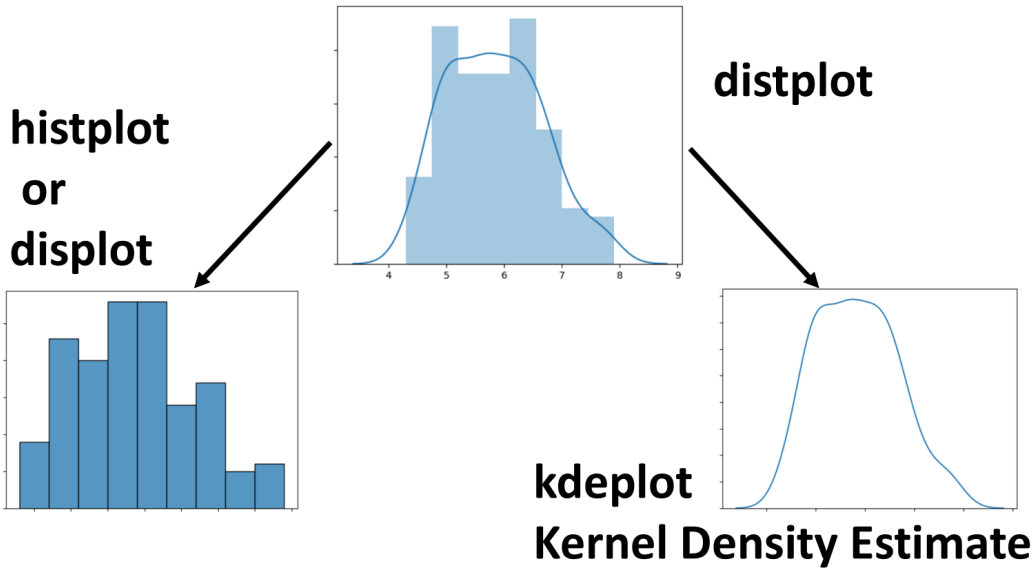
In [6]:
```
iris.dtypes
```

Out[6]:
```
SL         float64
SW         float64
PL         float64
PW         float64
Flower      object
dtype: object
```

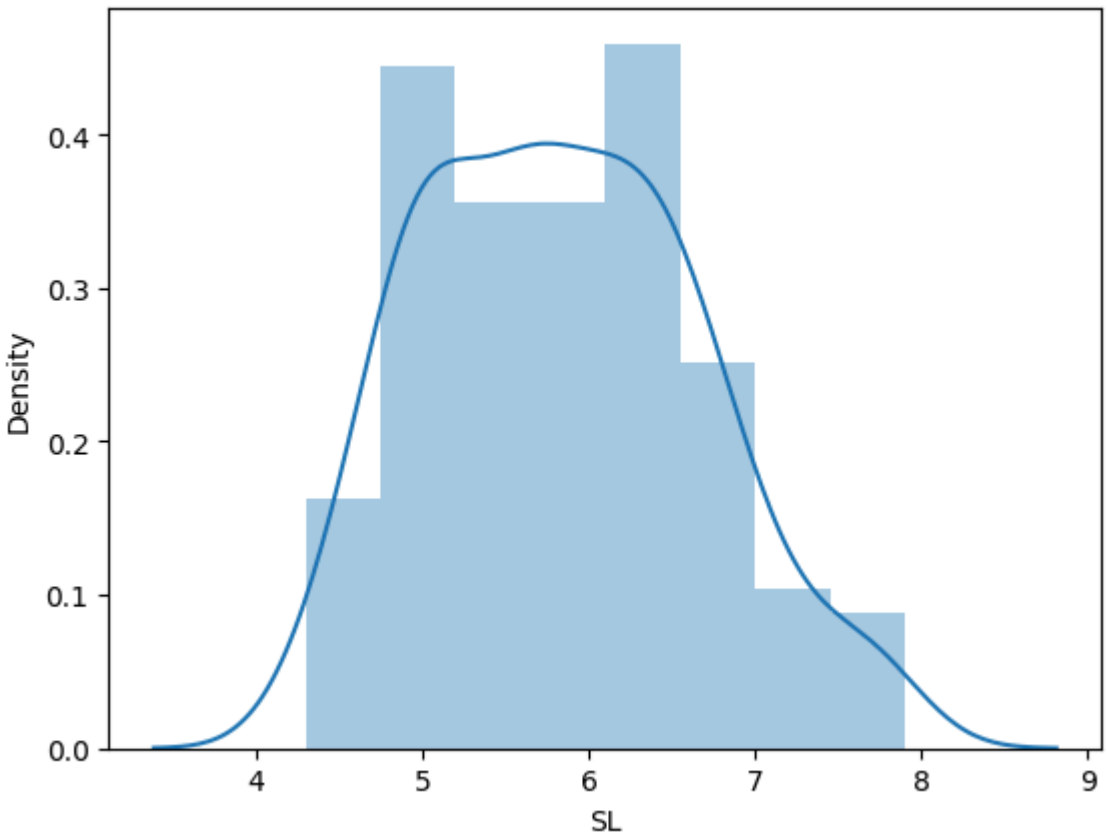
In [140…
```
for i in iris.columns:
    print(i,':','\n',iris[i].unique(),'\n')
```

```
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]

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

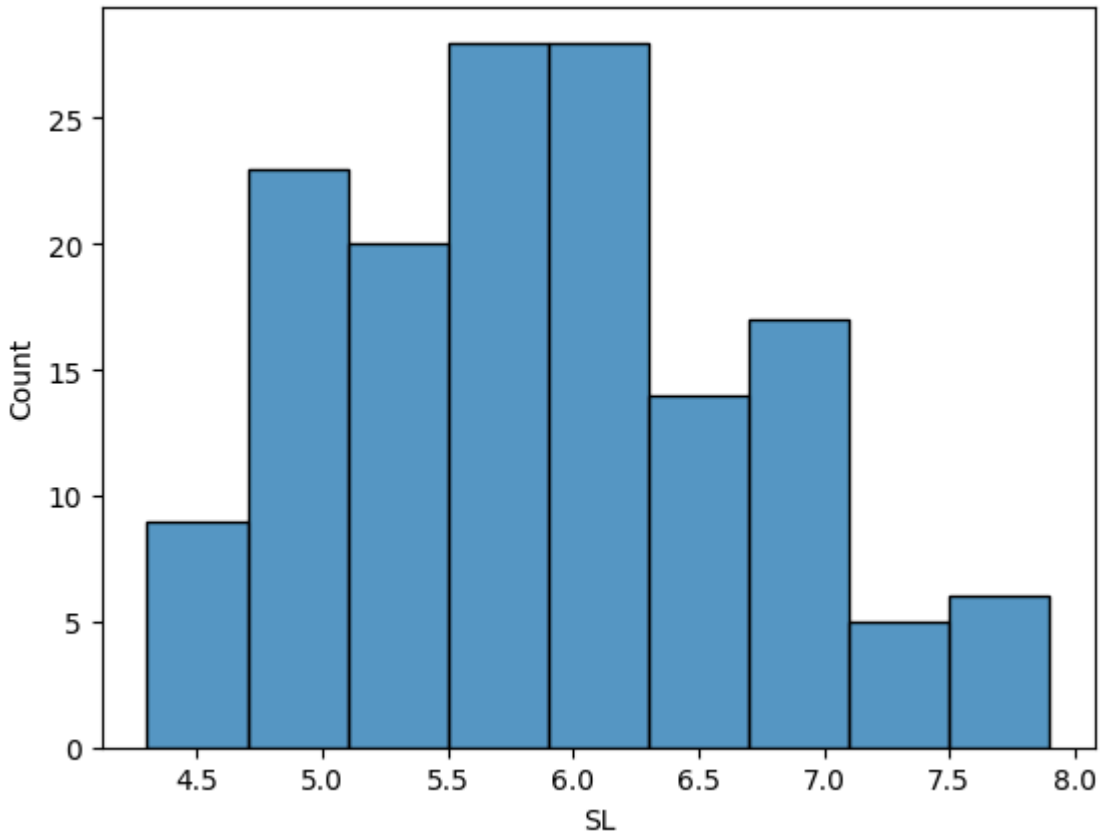
In [141…
```
import warnings
warnings.filterwarnings('ignore')
```



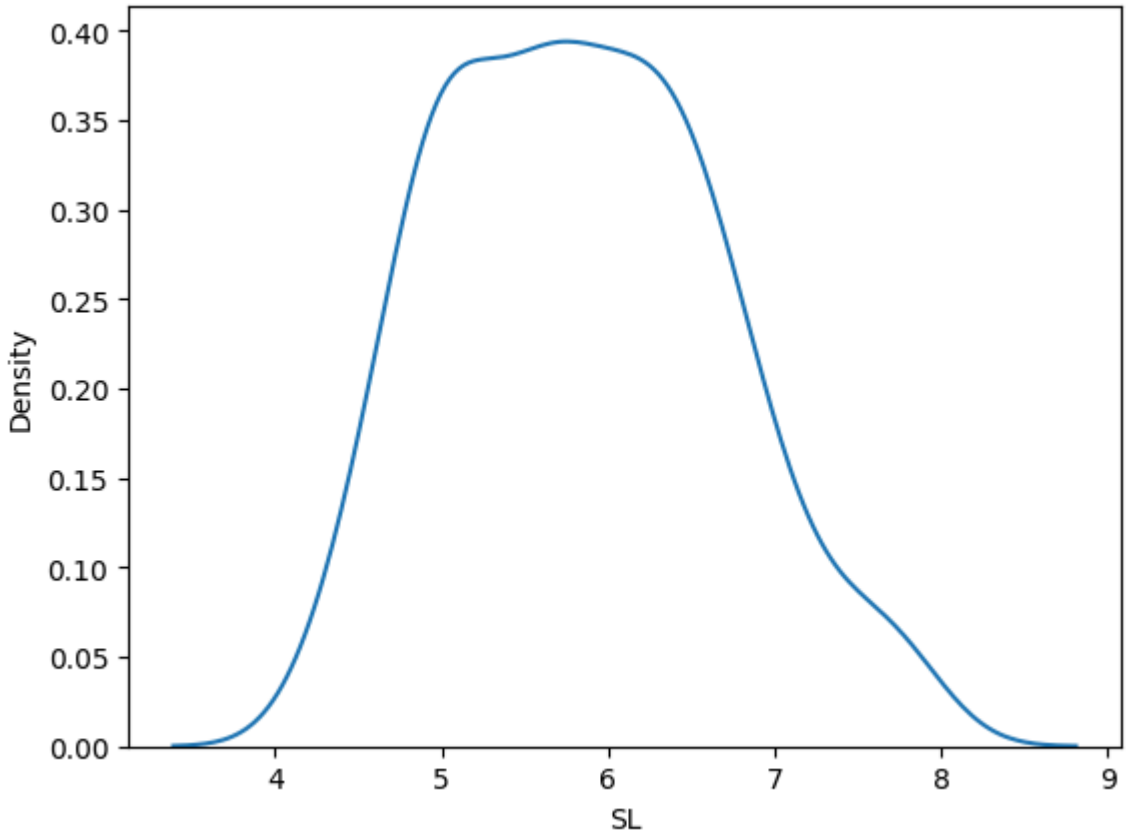
In [142…
```
#Continuous and Categorical
#Distribution Plot
sns.distplot(iris.SL)
plt.show()
```



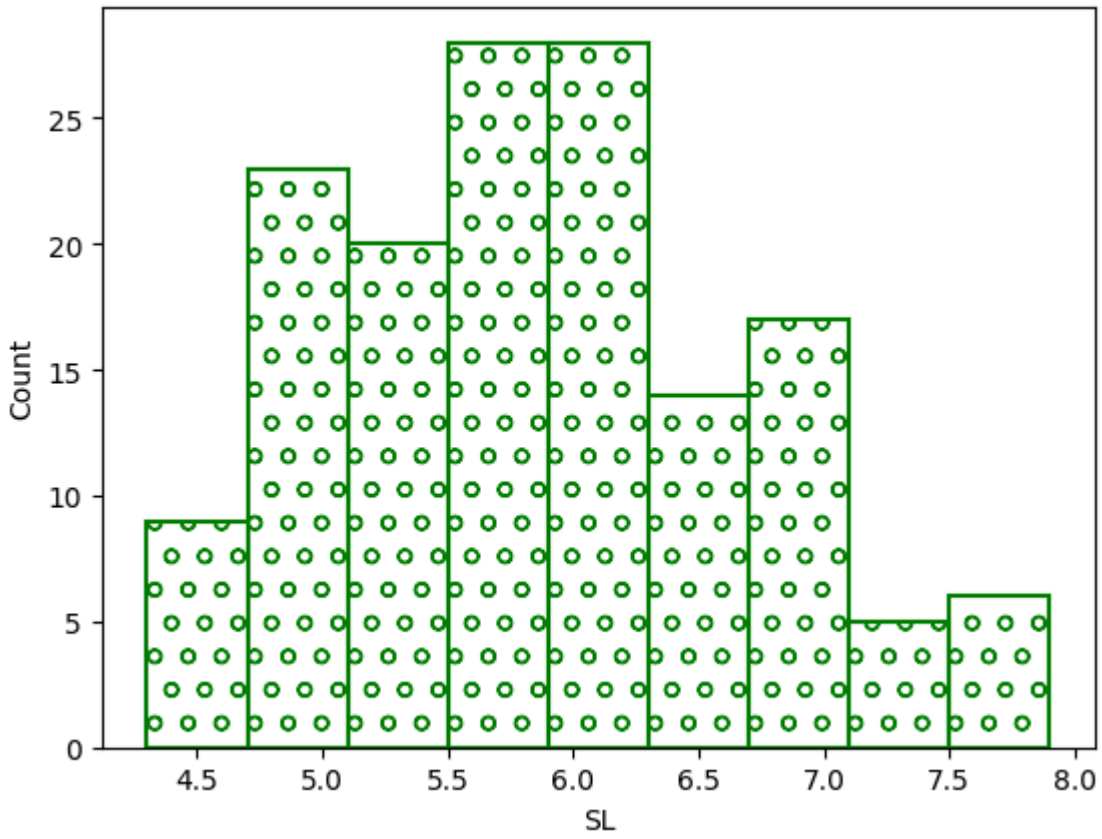
In [143…
```
sns.histplot(iris.SL)
plt.show()
```



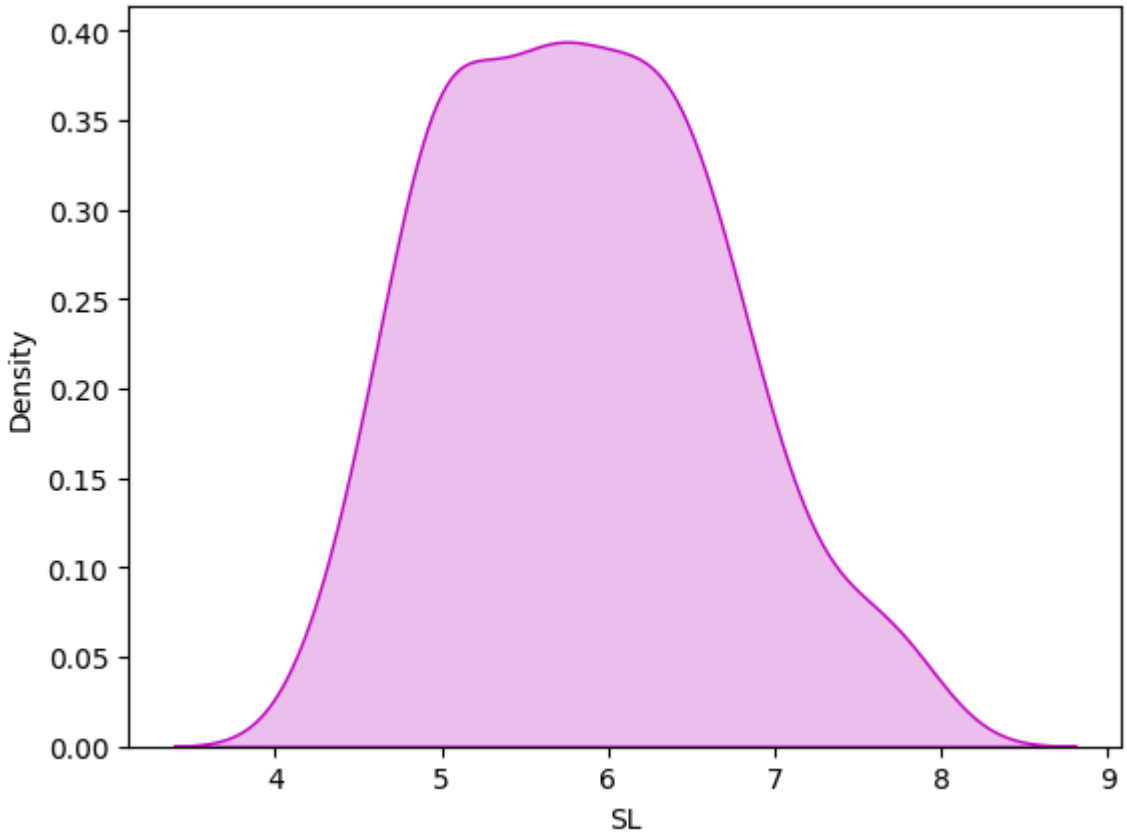
In [144…
```
sns.kdeplot(iris.SL)
plt.show()
```



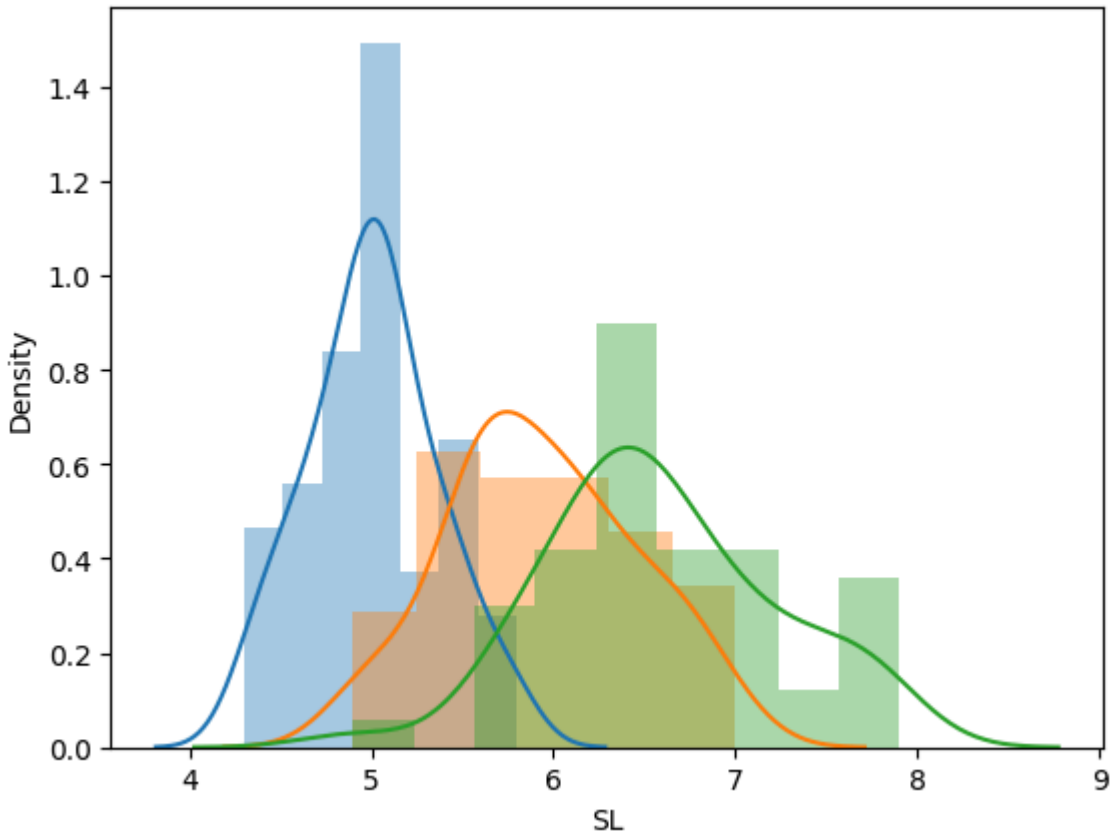
In [150…
```
sns.histplot(iris.SL,fill=False,color='g',hatch='o')
plt.show()
```



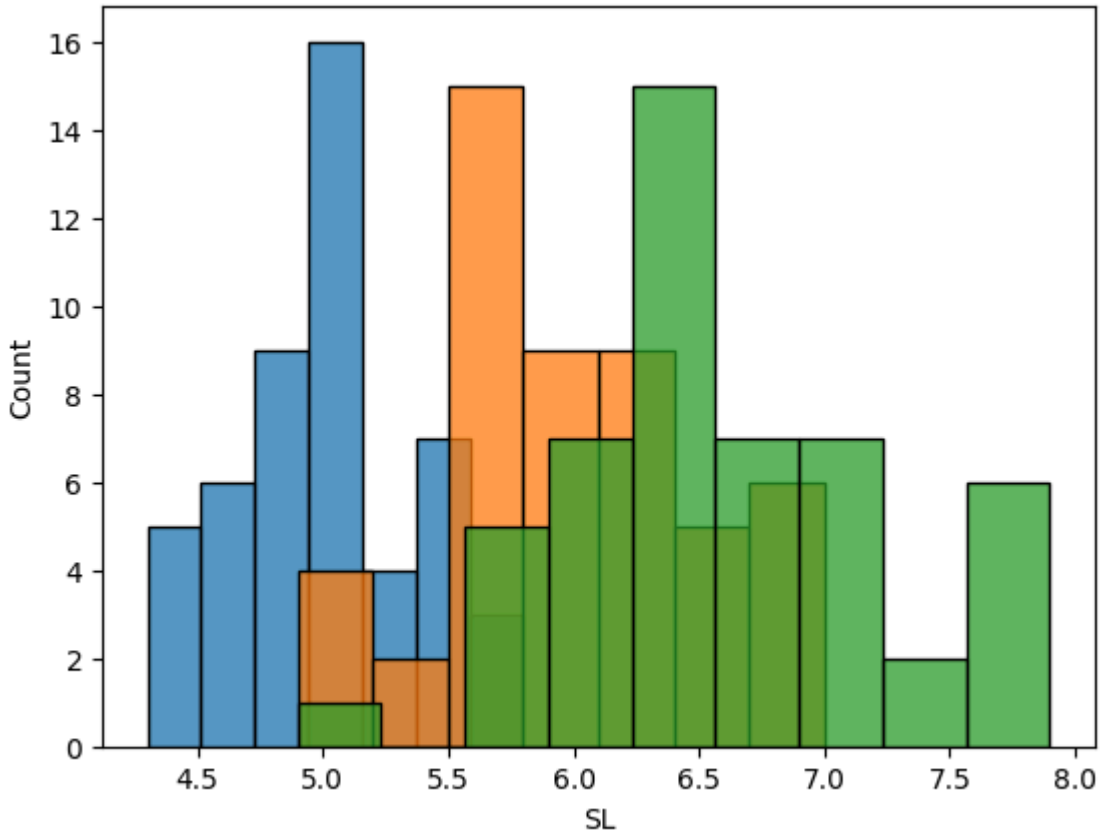
In [152…
```
sns.kdeplot(iris.SL,fill=True,color='m')
plt.show()
```



In [154…
```
sns.distplot(iris.SL[iris['Flower']=='Iris-setosa'])
sns.distplot(iris.SL[iris.Flower=='Iris-versicolor'])
sns.distplot(iris.SL[iris.Flower=='Iris-virginica'])
plt.show()
```

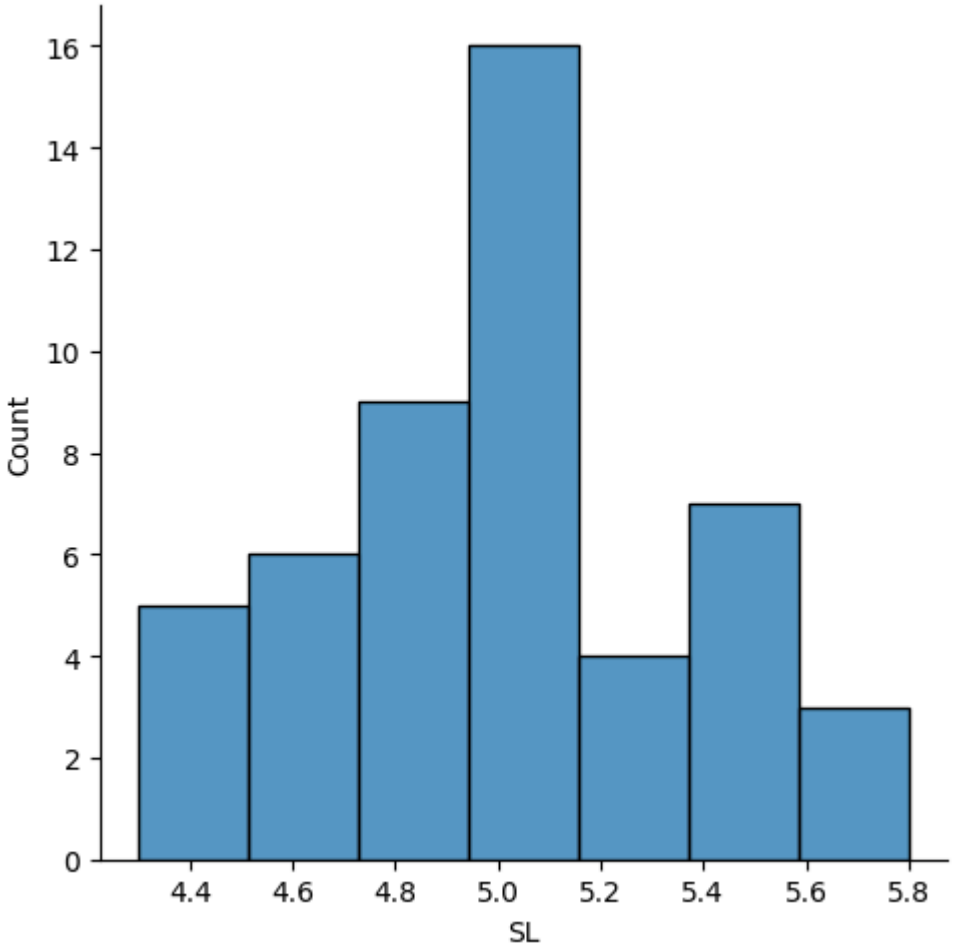


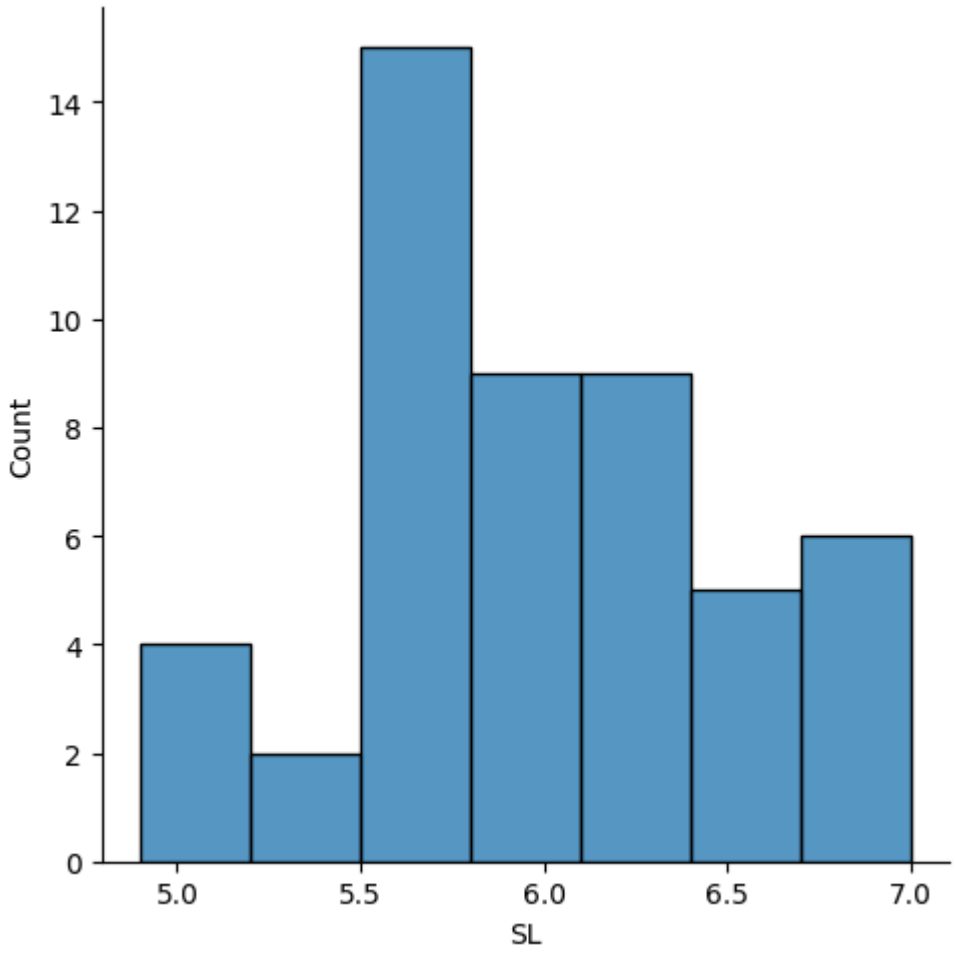
In [155…
```
sns.histplot(iris.SL[iris['Flower']=='Iris-setosa'])
sns.histplot(iris.SL[iris.Flower=='Iris-versicolor'])
sns.histplot(iris.SL[iris.Flower=='Iris-virginica'])
plt.show()
```

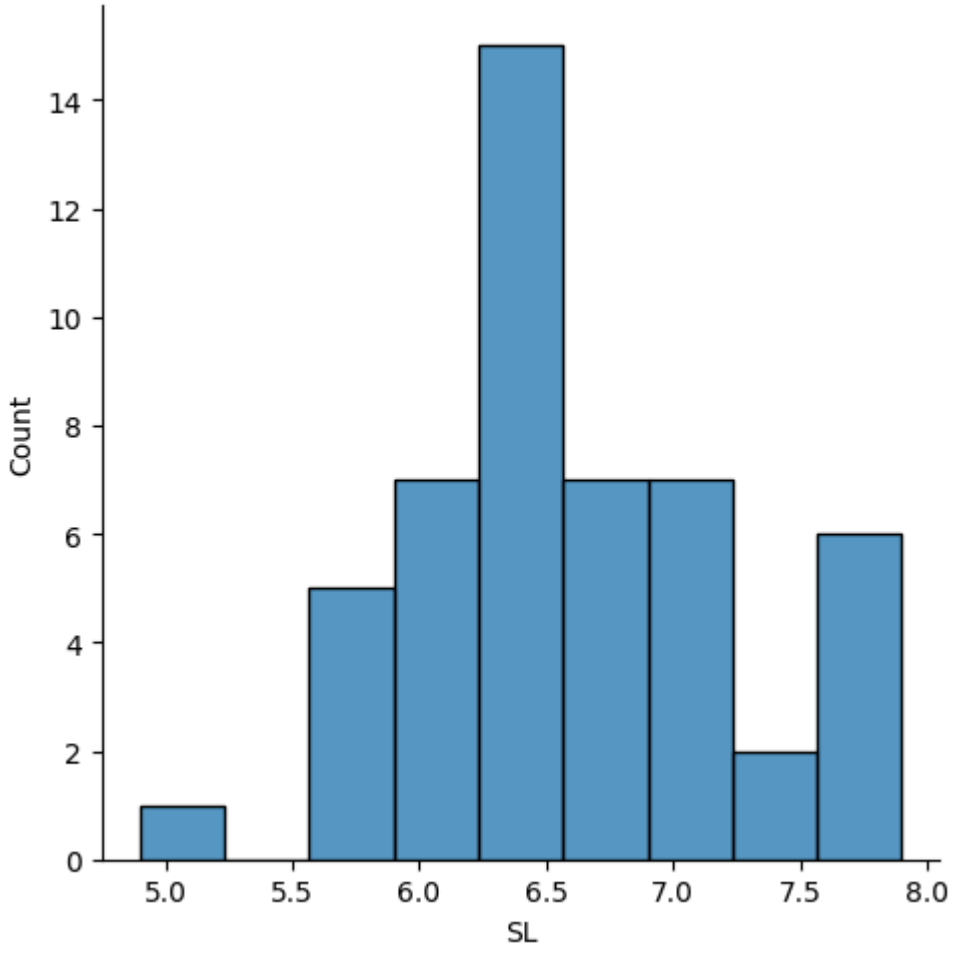


In [156…
```
sns.displot(iris.SL[iris['Flower']=='Iris-setosa'])
sns.displot(iris.SL[iris.Flower=='Iris-versicolor'])
```

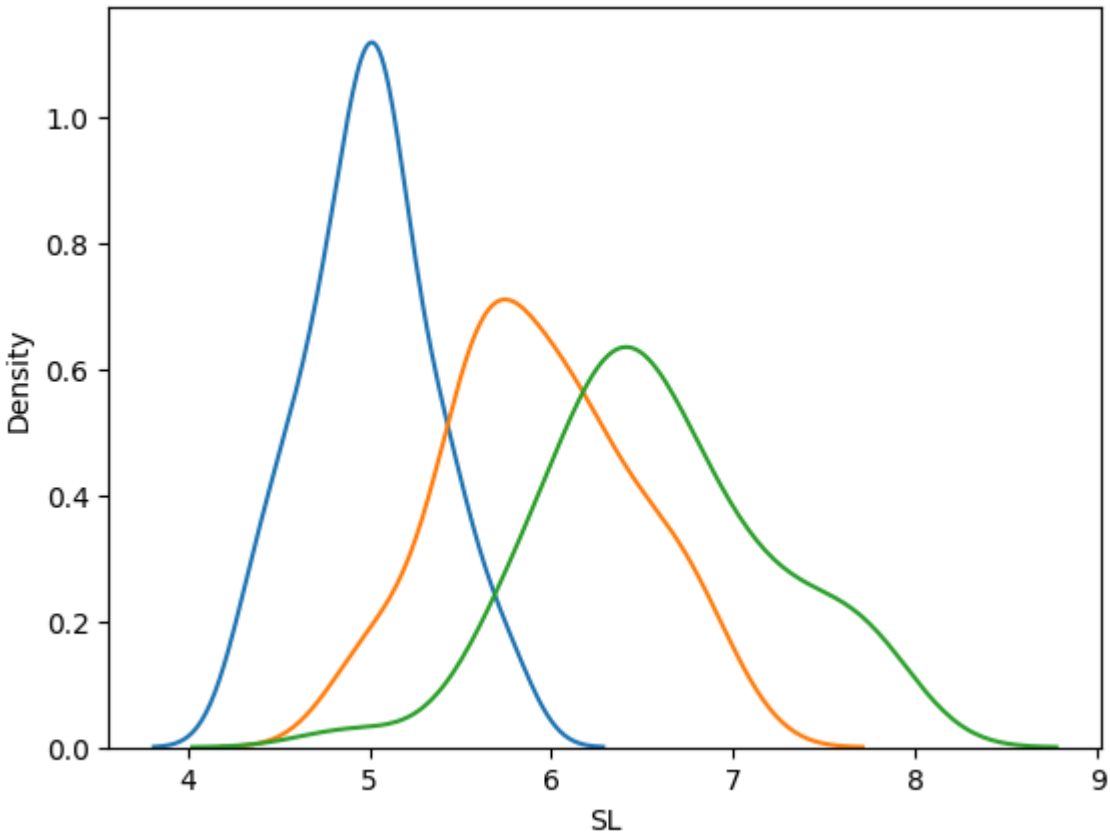
```
sns.displot(iris.SL[iris.Flower=='Iris-virginica'])
plt.show()
```



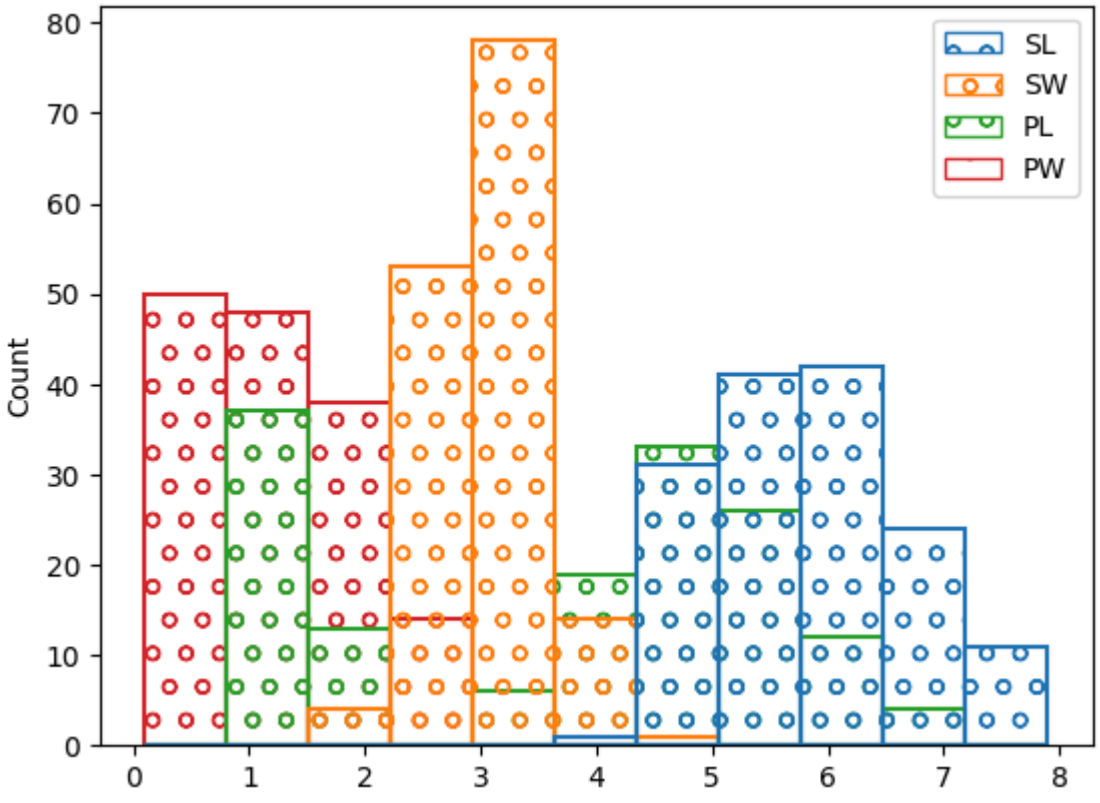

Count
SL
0
2
4
6
8
10
12
14
5.0
5.5
6.0
6.5
7.0


Count
SL
0
2
4
6
8
10
12
14
5.0
5.5
6.0
6.5
7.0
7.5
8.0

In [157…
```
sns.kdeplot(iris.SL[iris['Flower']=='Iris-setosa'])
sns.kdeplot(iris.SL[iris.Flower=='Iris-versicolor'])
sns.kdeplot(iris.SL[iris.Flower=='Iris-virginica'])
plt.show()
```

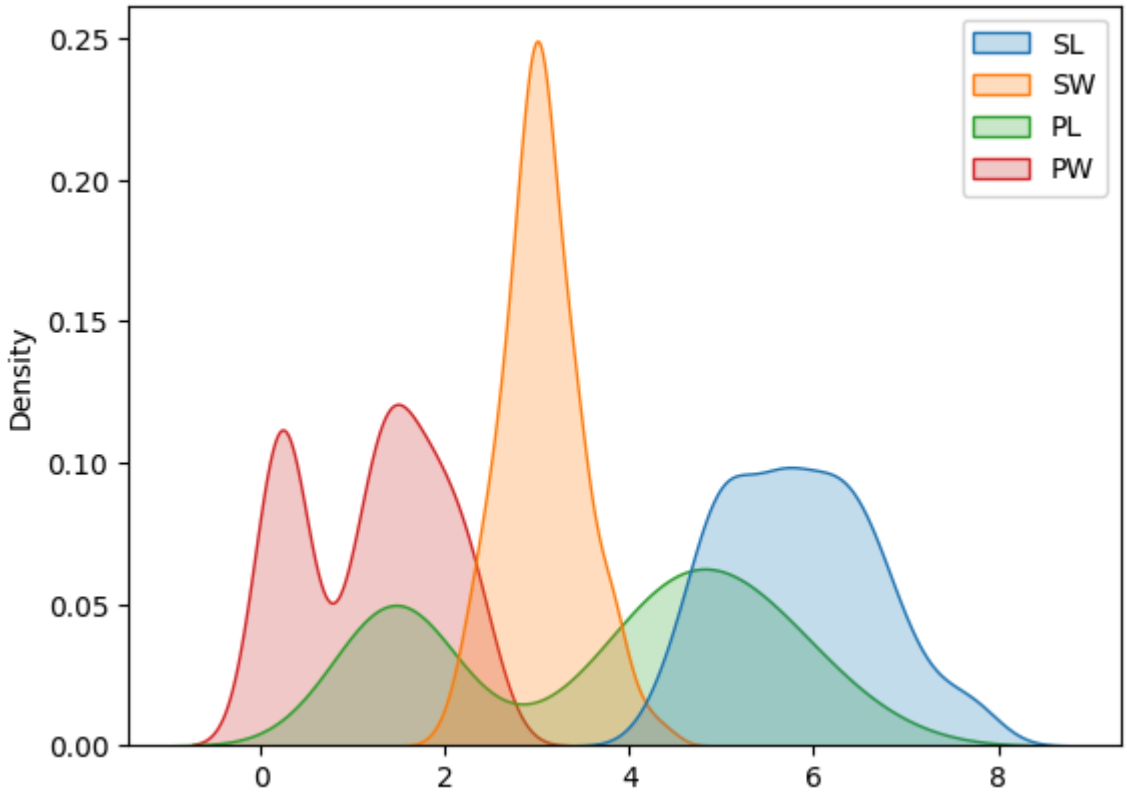


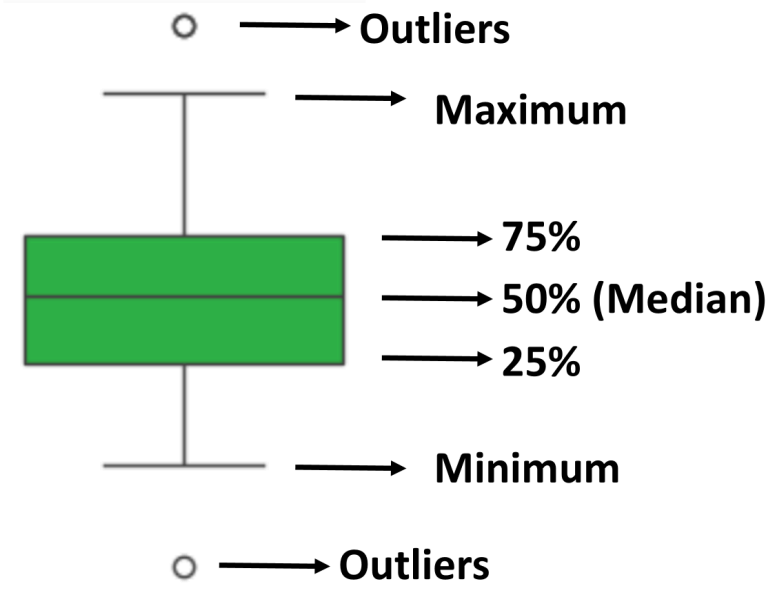
In [162…
```
sns.histplot(data=iris,fill=False,hatch='o')
plt.show()
```

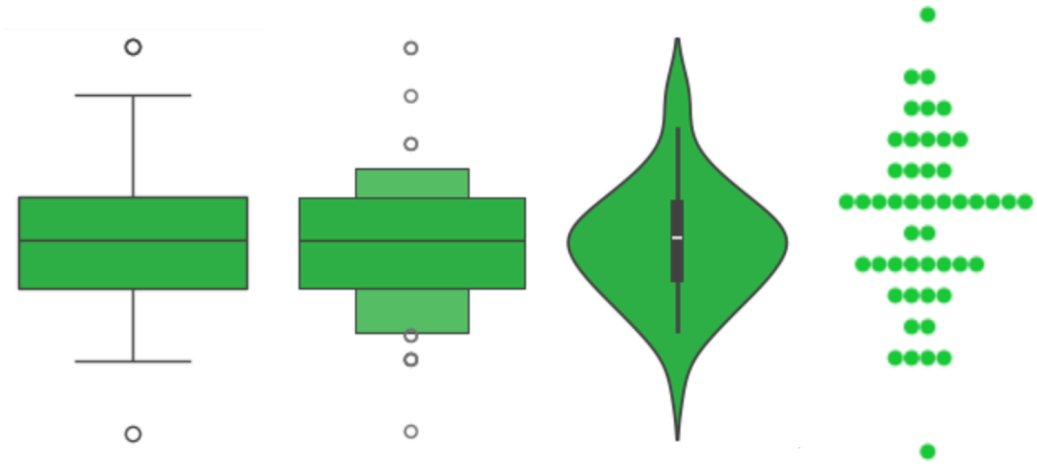


In [163…

```
sns.kdeplot(data=iris,fill=True)
plt.show()
```

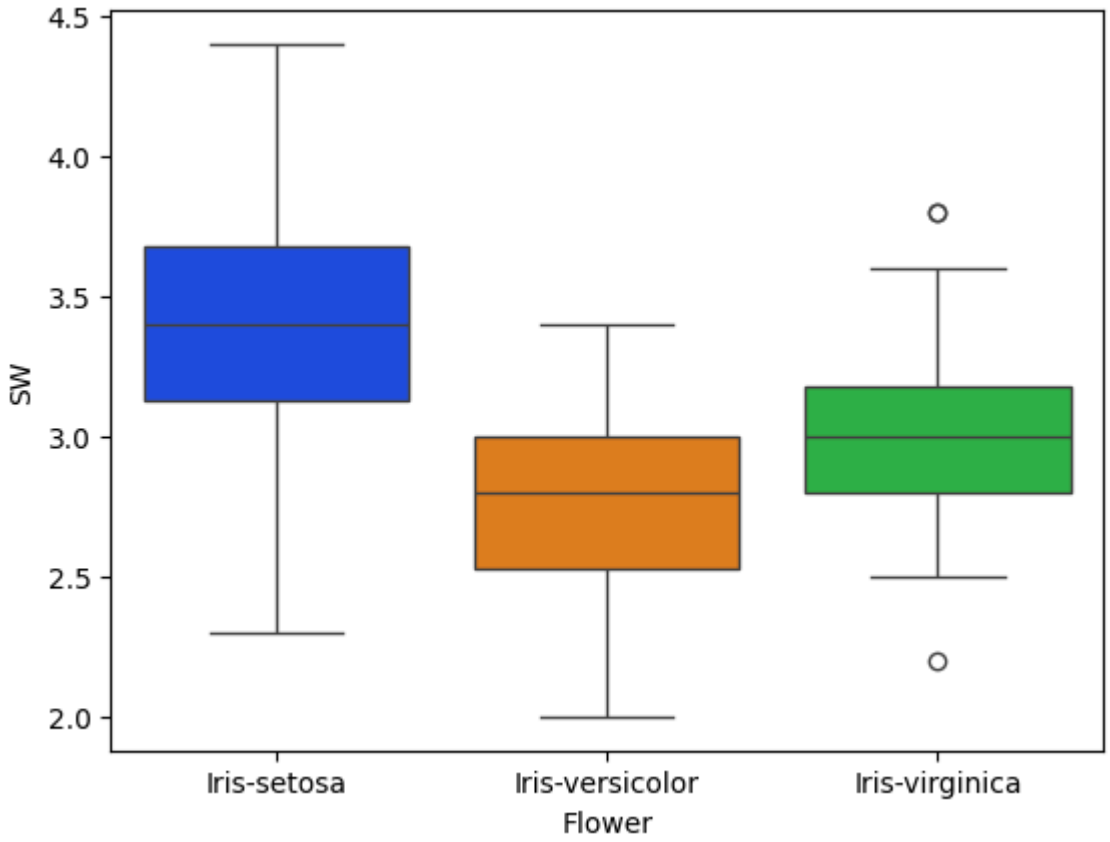






In [165…

```
#Box Plot
sns.boxplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```

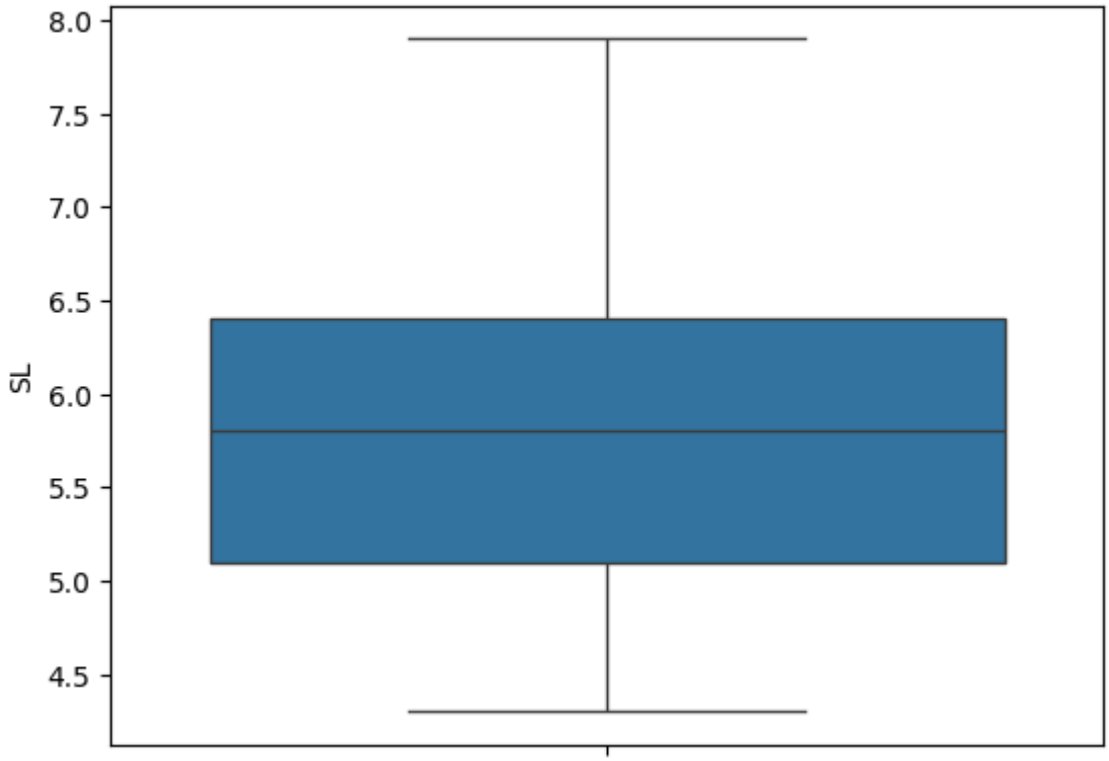


In [166…
```
iris.describe()
```

Out[166…

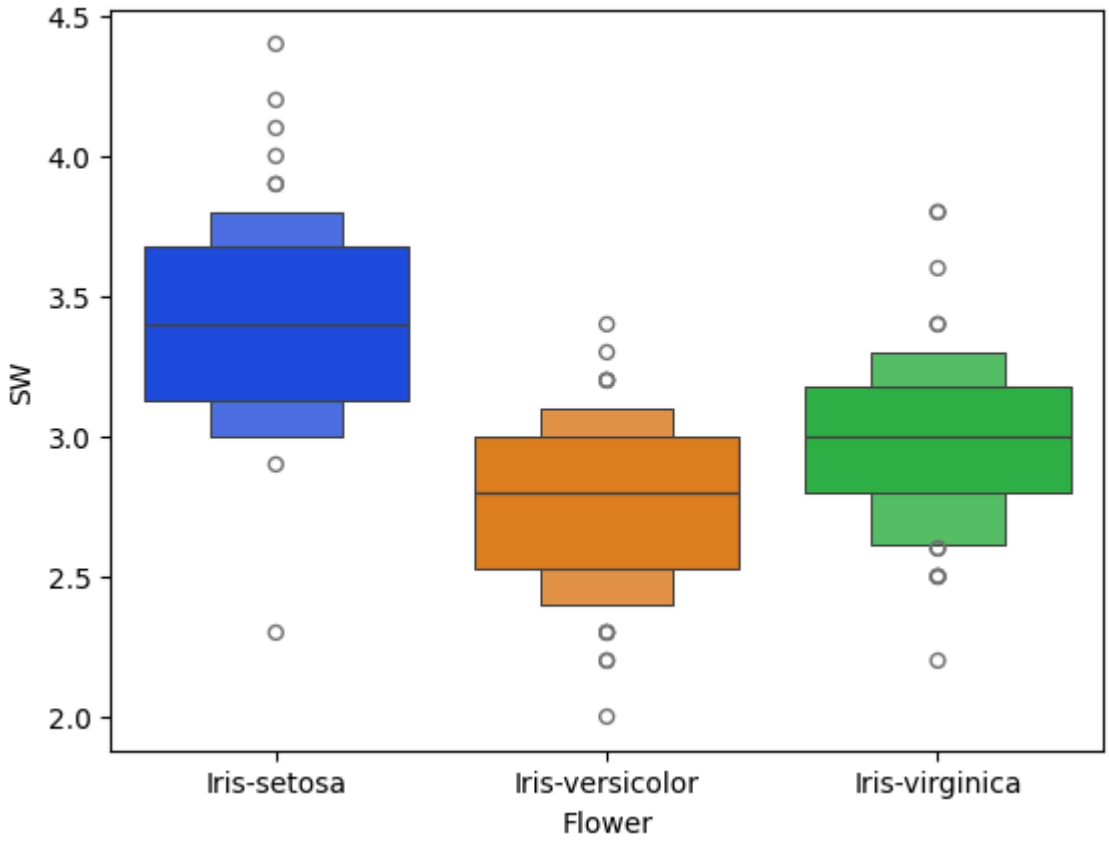
	SL	SW	PL	PW
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

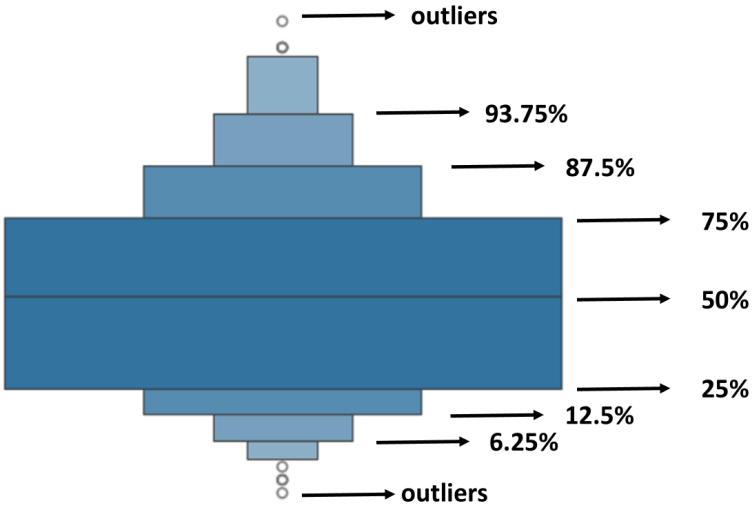
In [167…
```
sns.boxplot(y=iris.SL)
plt.show()
```



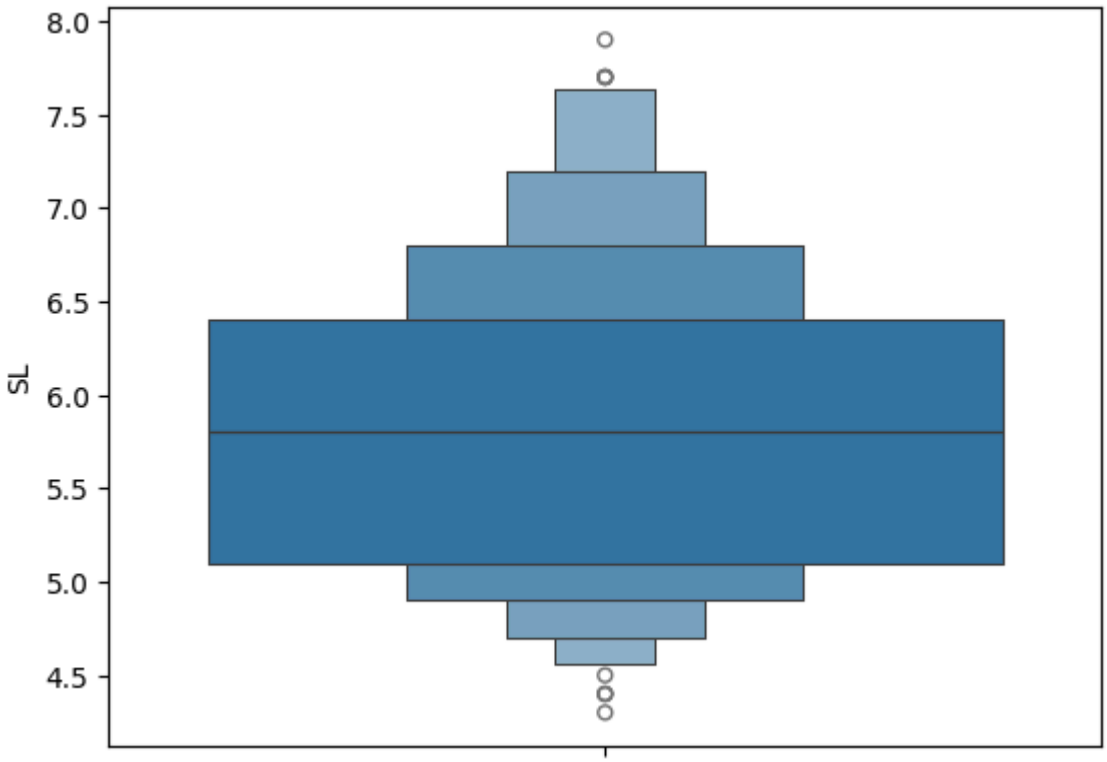
In [168…

```
#Boxen Plot
sns.boxenplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```





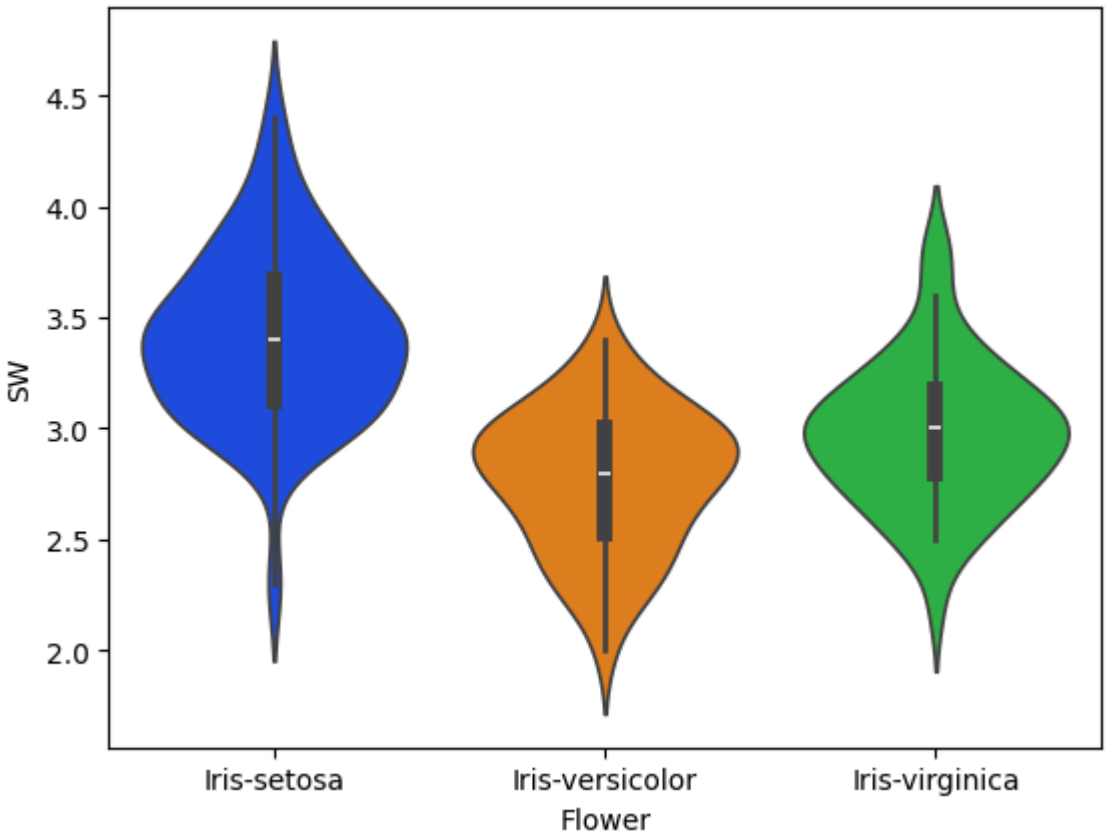
In [169…
```
sns.boxenplot(y=iris.SL)
plt.show()
```



In [172…
```
np.percentile(iris.SL,87.5)
```

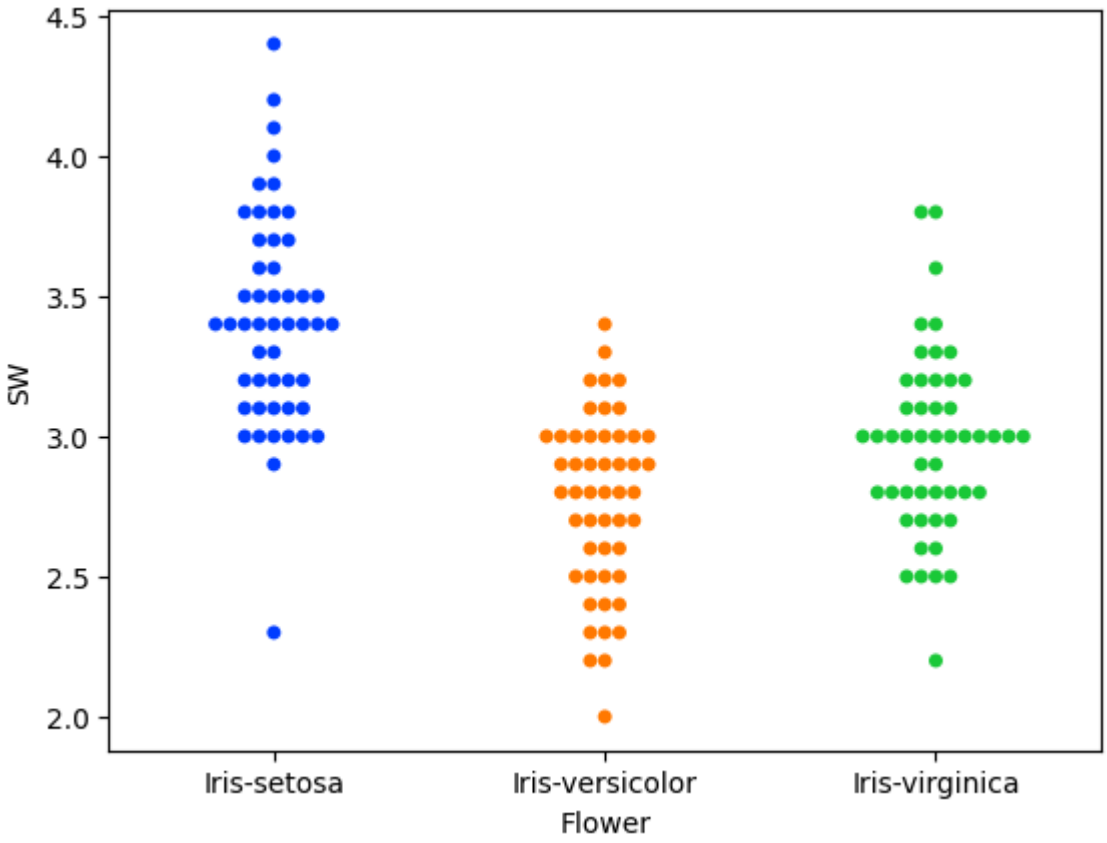
Out[172… 6.8

In [173…
```
#Violin Plot
sns.violinplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```



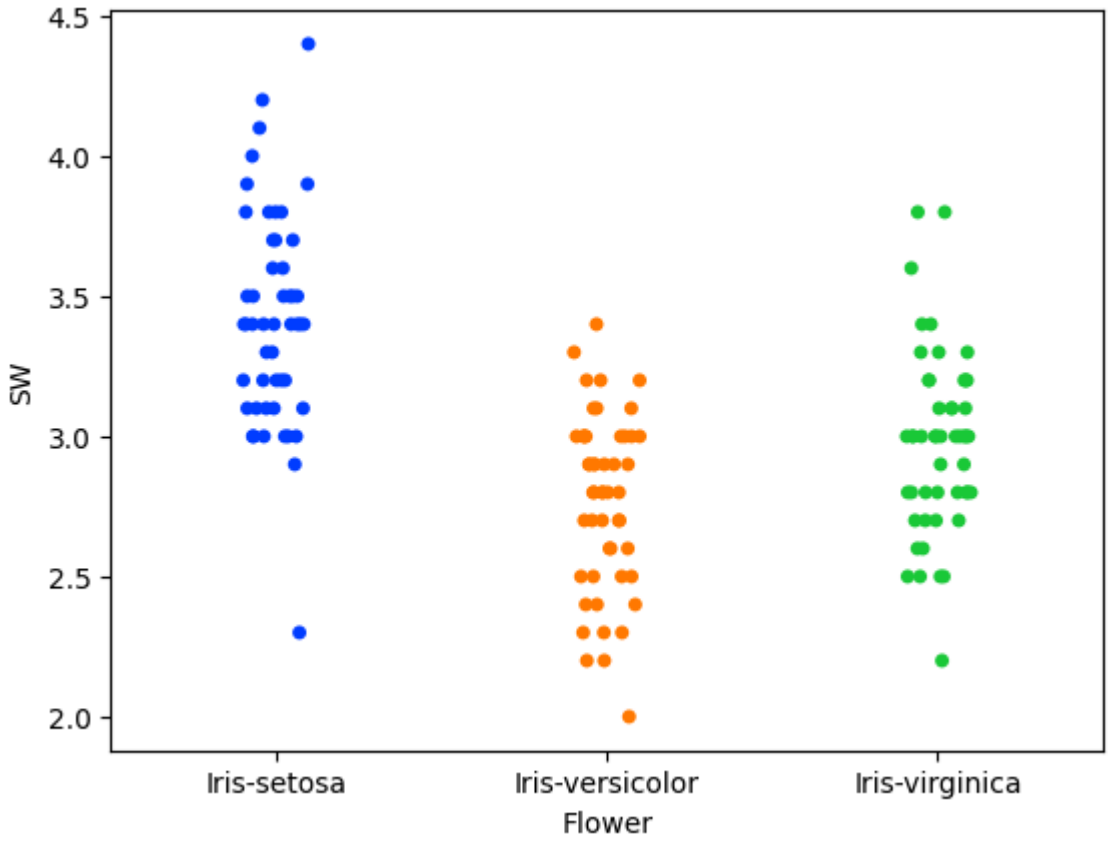
In [174…

```
#Swarm Plot
sns.swarmplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```



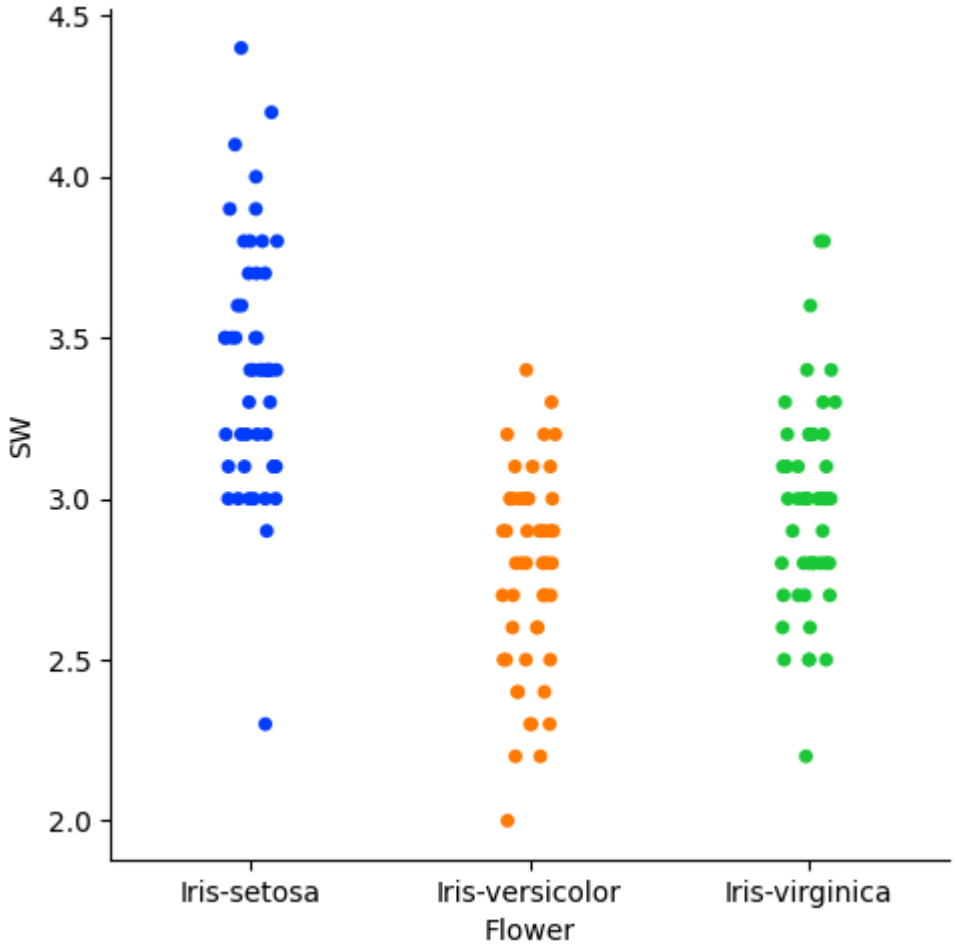
In [175…

```
#Strip Plot
sns.stripplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```

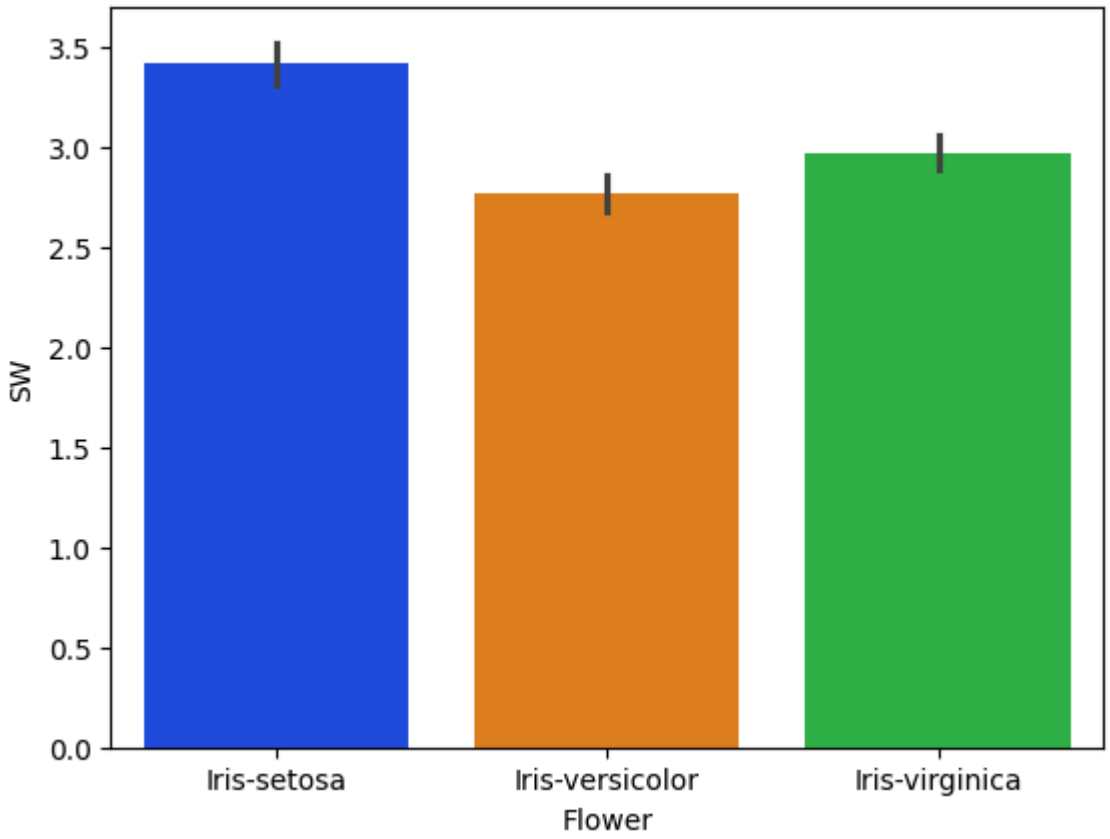


In [176…

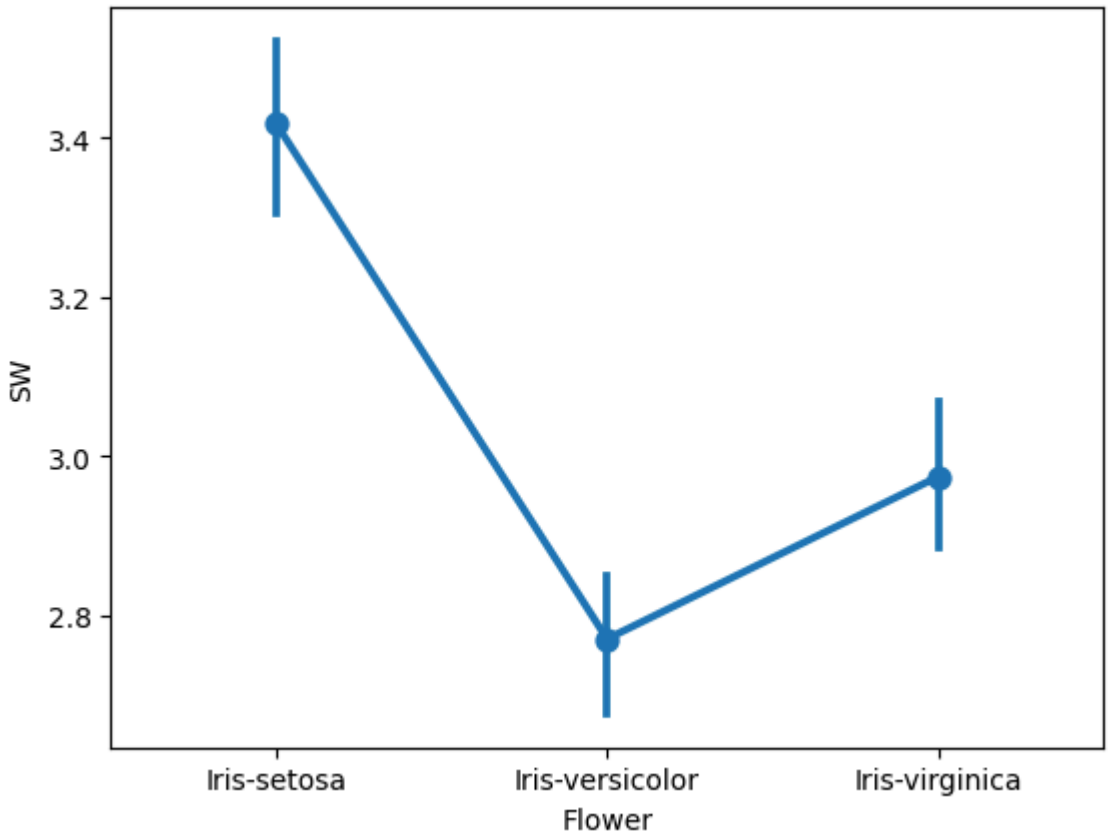
```
#Cat Plot
sns.catplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```



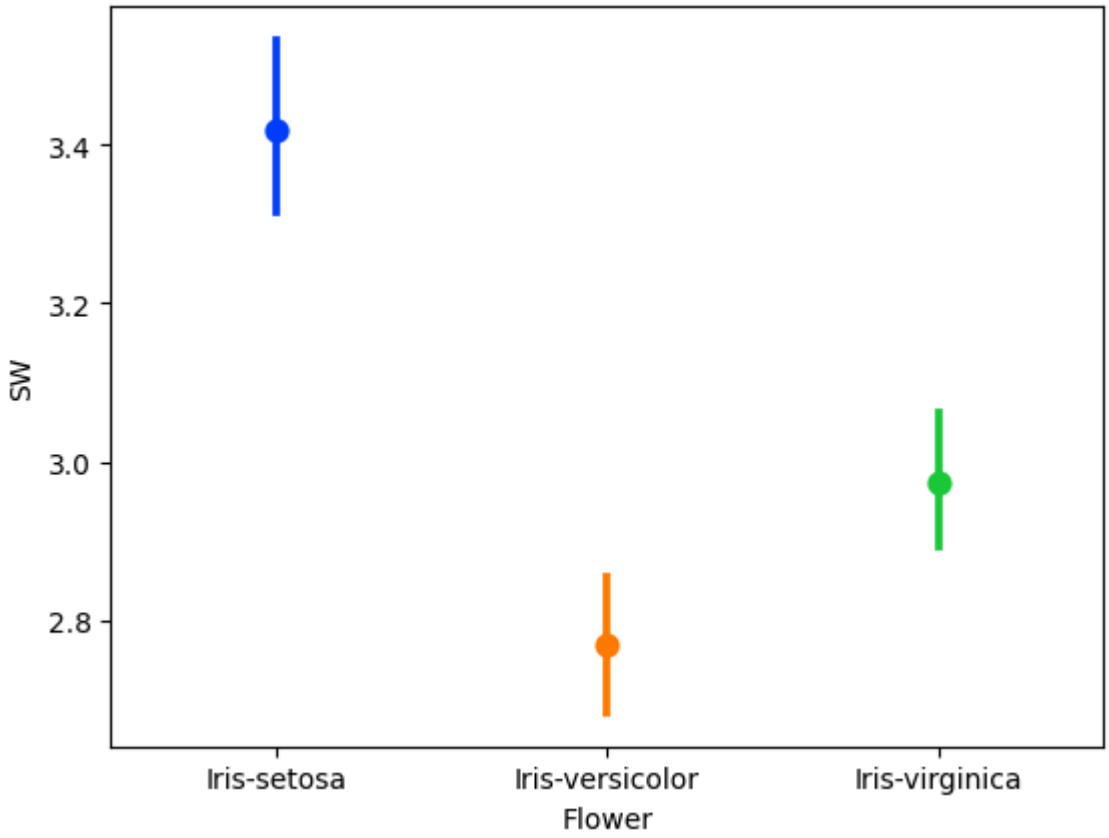
In [177…
```
#Bar Plot
sns.barplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```



In [178…
```
#Point Plot
sns.pointplot(x=iris.Flower,y=iris.SW)
plt.show()
```

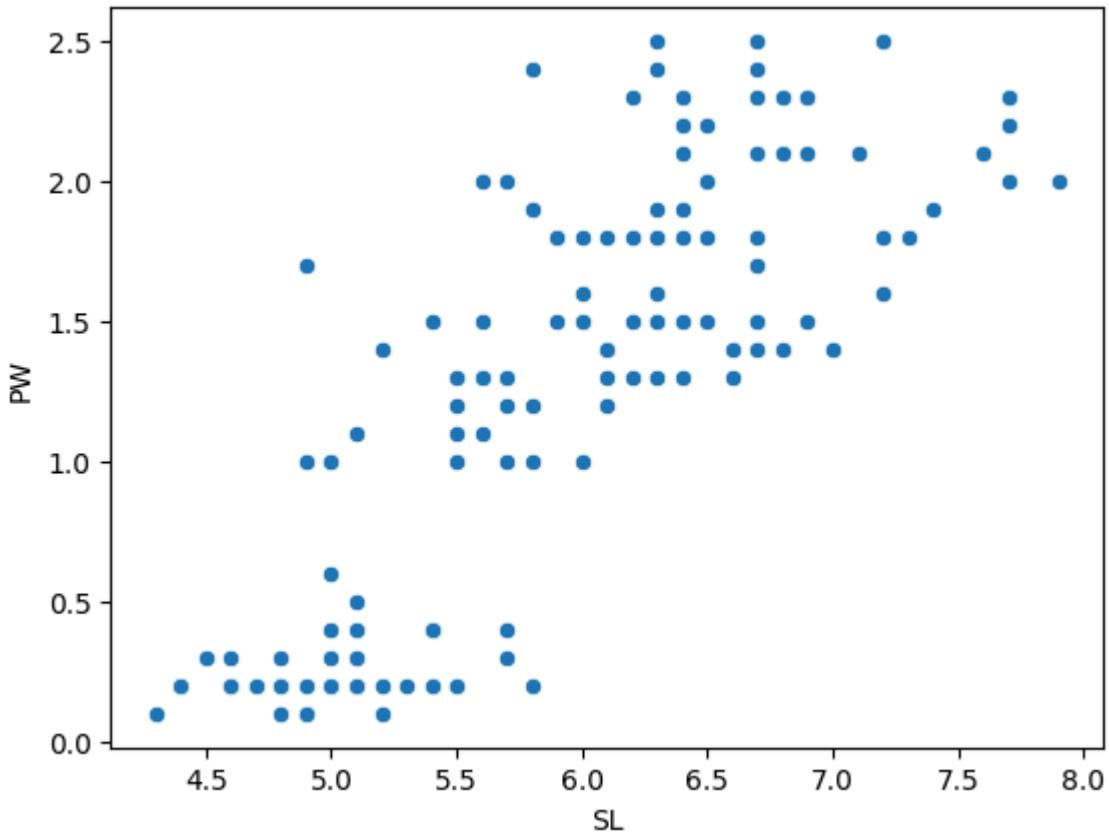


In [179…
```
#Point Plot
sns.pointplot(x=iris.Flower,y=iris.SW,palette='bright')
plt.show()
```



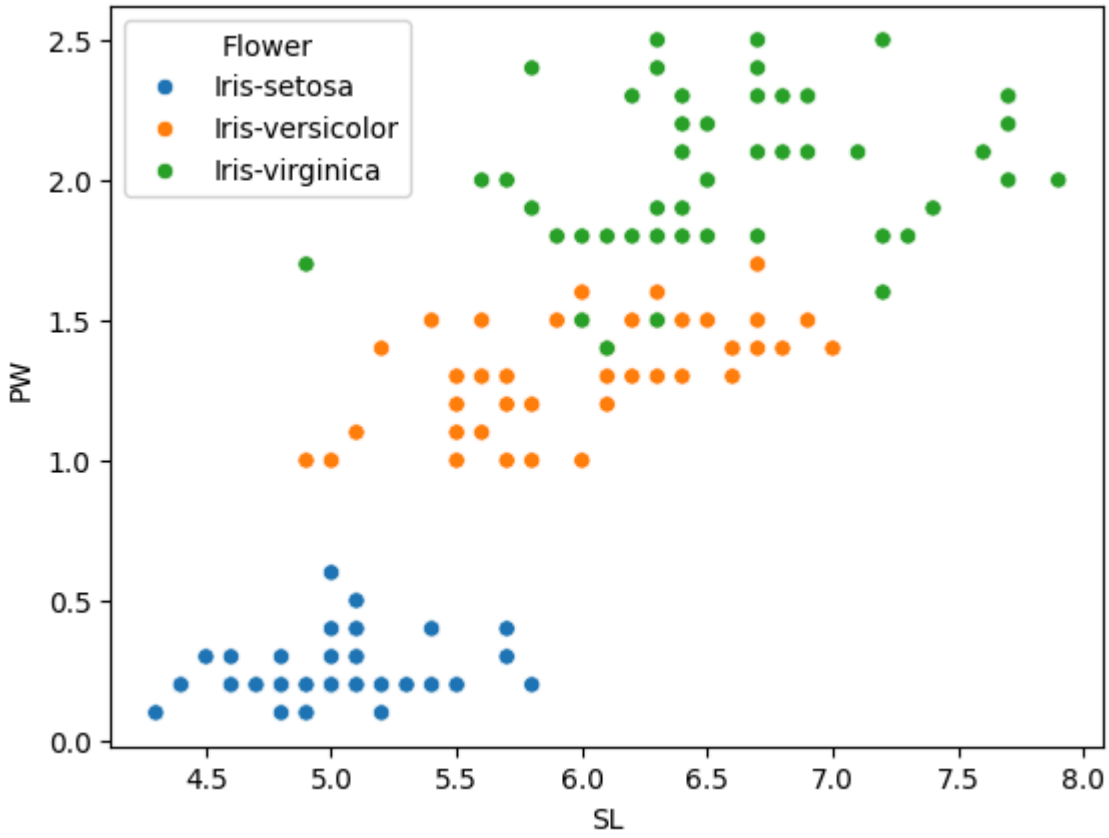
In [180…
```
#Continuous and Continuous
#Scatter Plot
```

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

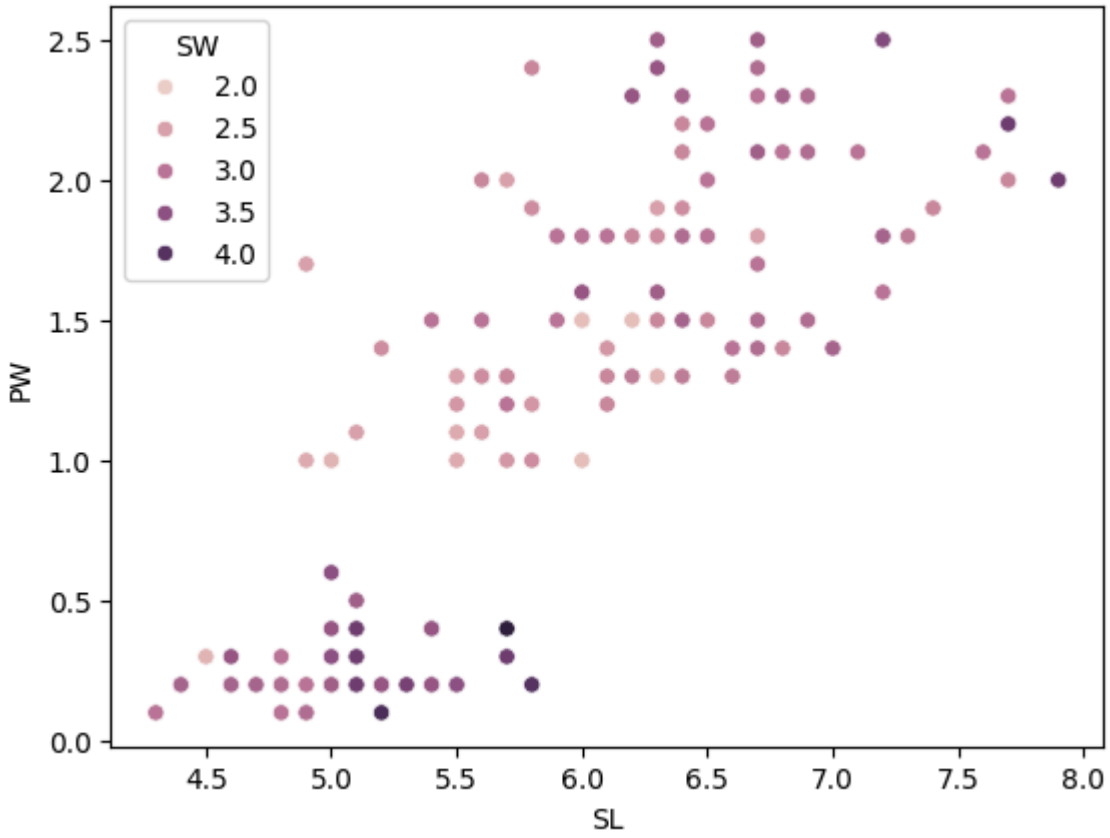


In [181…

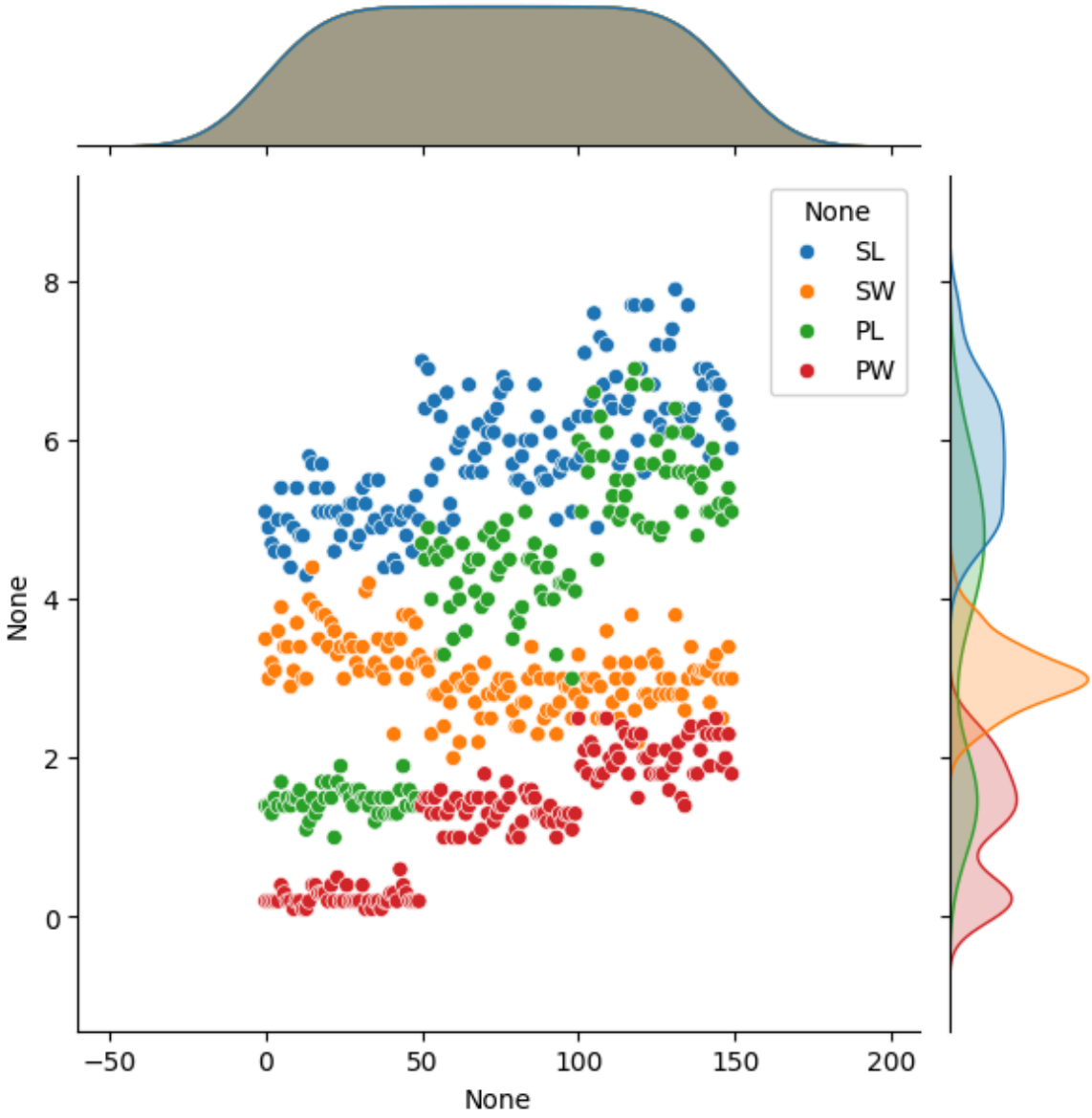
```
sns.scatterplot(x=iris.SL,y=iris.PW,hue=iris.Flower)
plt.show()
```



In [182…
```
sns.scatterplot(x=iris.SL,y=iris.PW,hue=iris.SW)
plt.show()
```

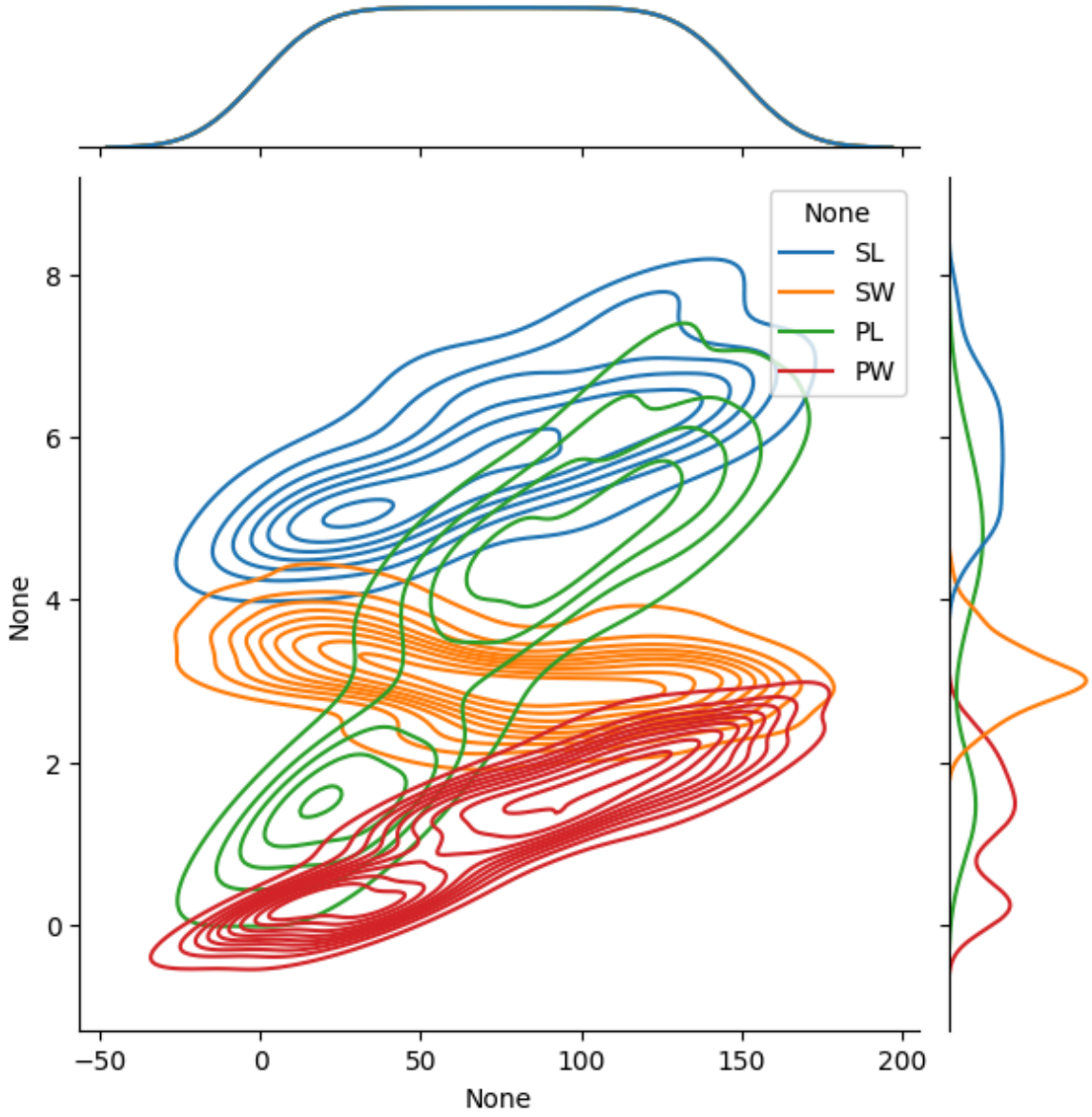


In [183…
```
#Joint Plot
sns.jointplot(data=iris)
plt.show()
```



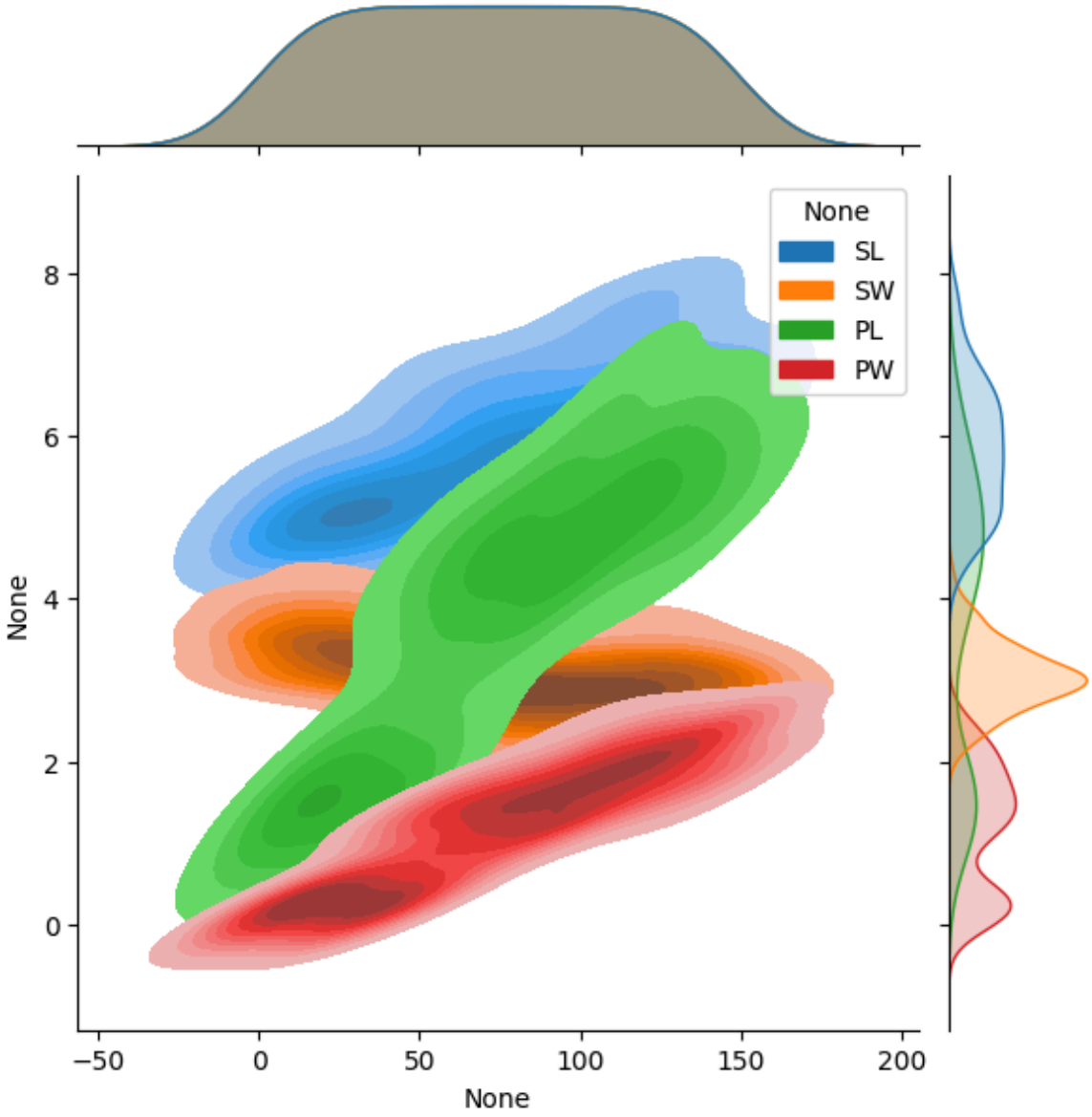
In [184…

```
sns.jointplot(data=iris,kind='kde')
plt.show()
```



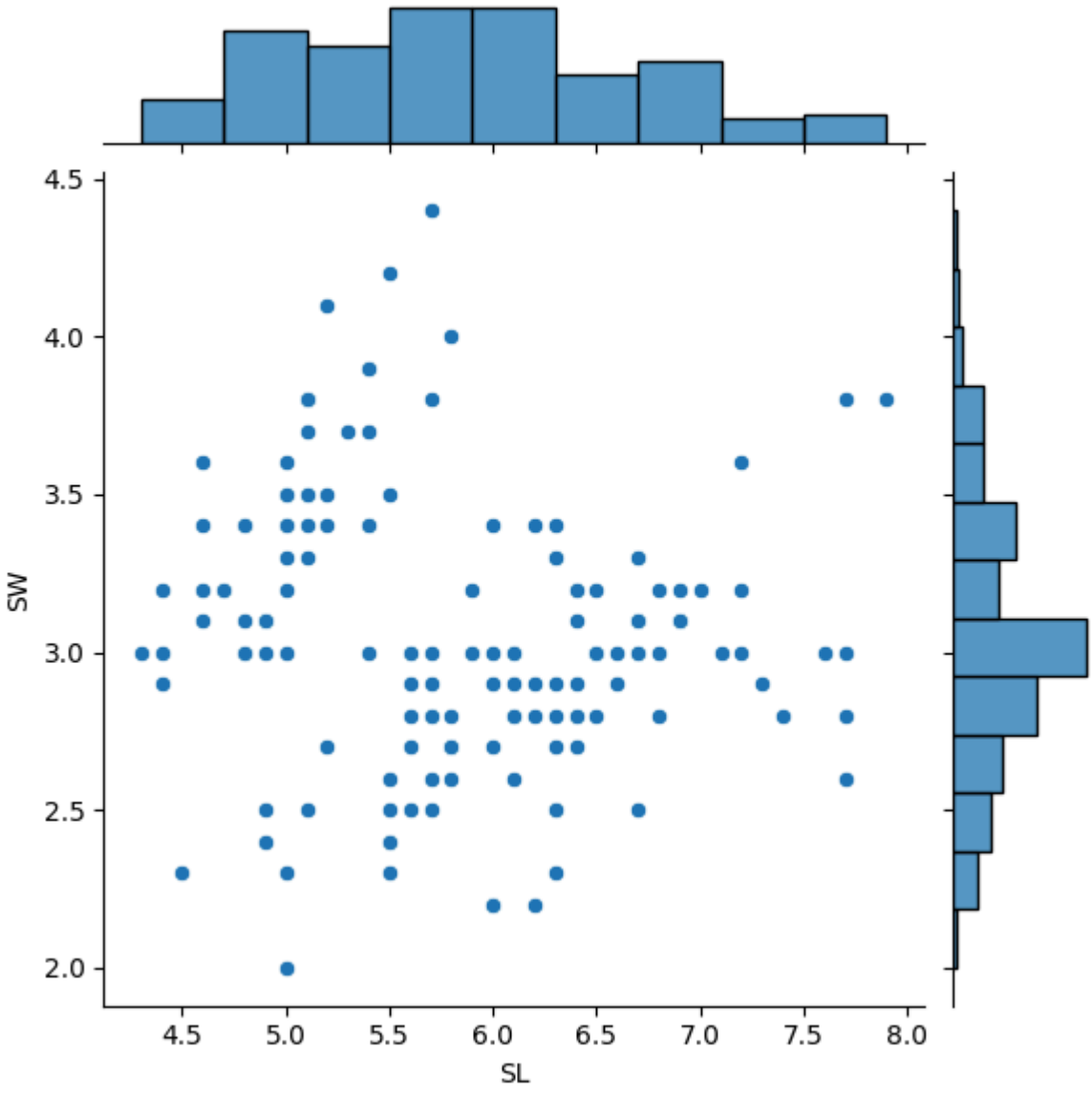
In [185…

```
sns.jointplot(data=iris,kind='kde',fill=True)
plt.show()
```



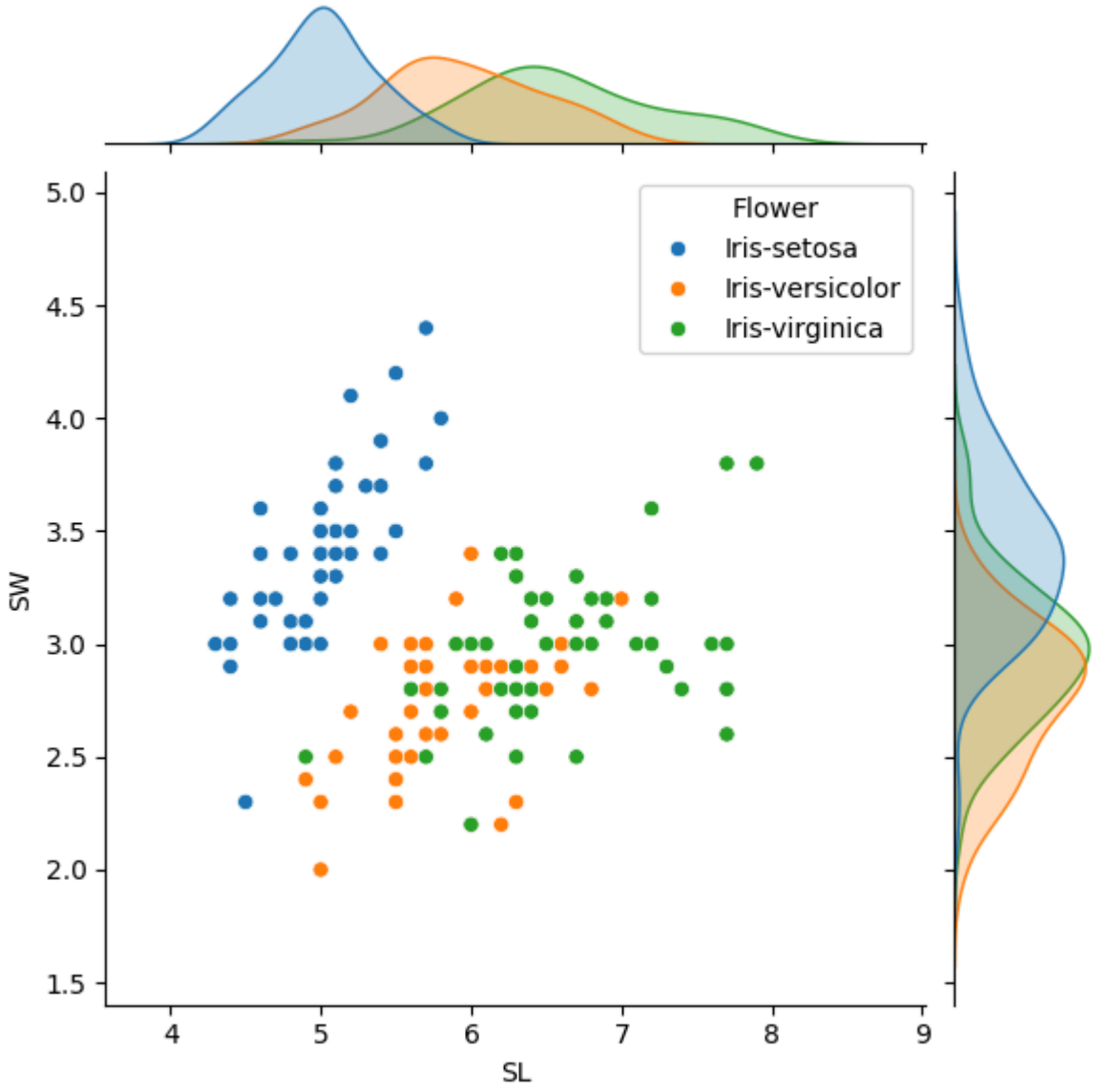
In [186…

```
sns.jointplot(x=iris.SL,y=iris.SW)
plt.show()
```



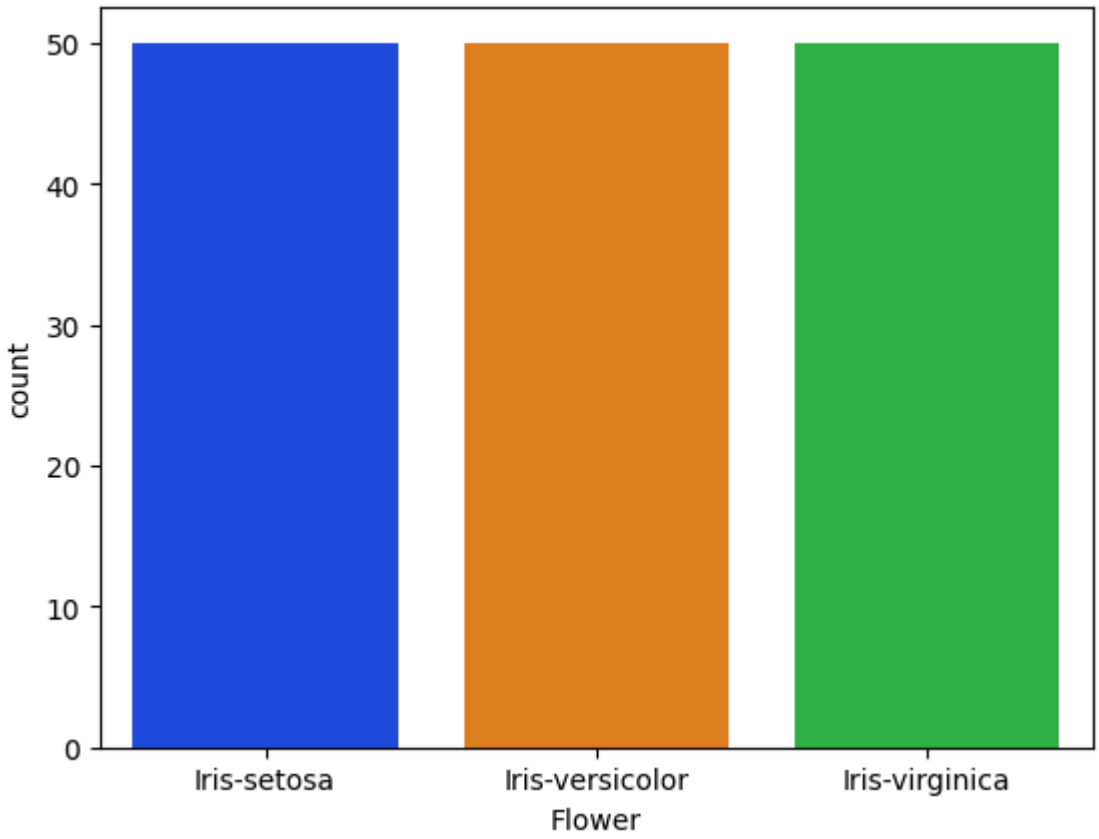
In [187…

```
sns.jointplot(x=iris.SL,y=iris.SW,hue=iris.Flower)
plt.show()
```

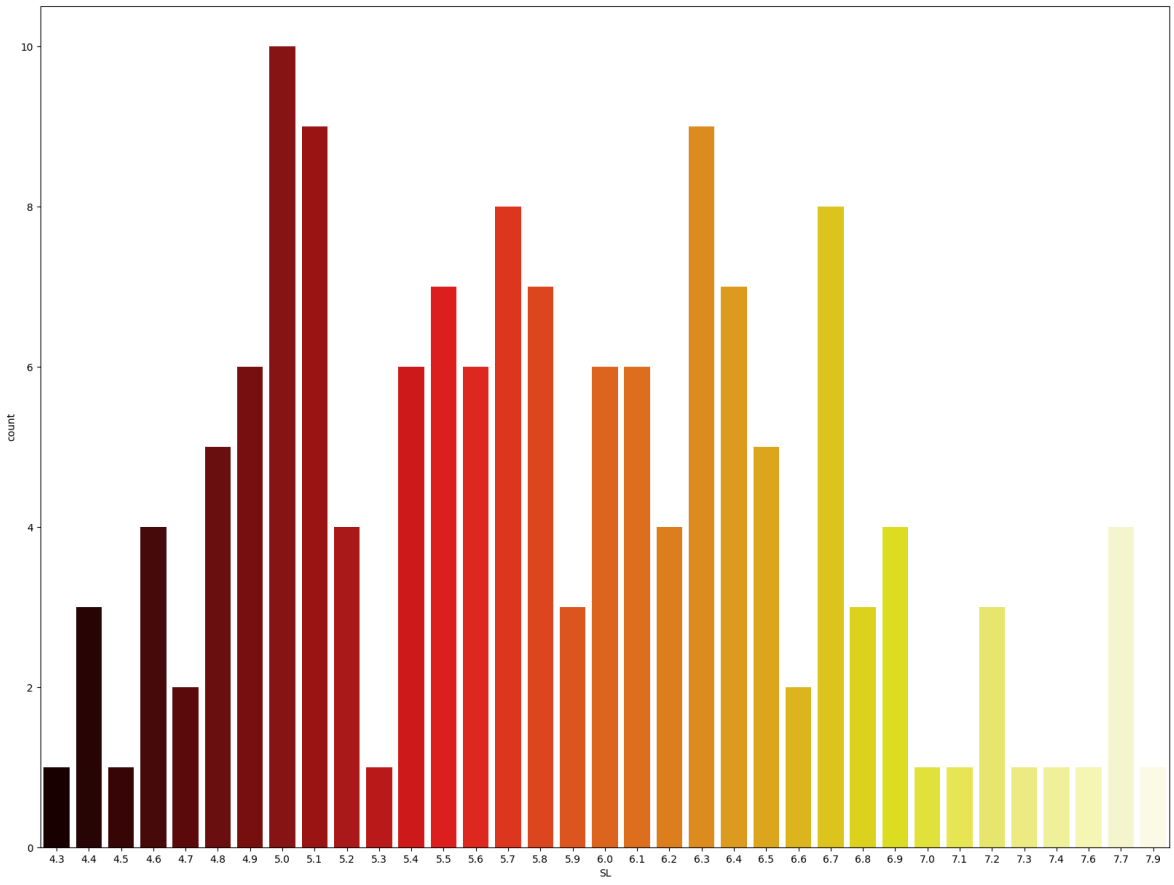


In [190…

```
#Categorical and Categorical
#Count Plot
sns.countplot(x=iris.Flower,palette='bright')
plt.show()
```

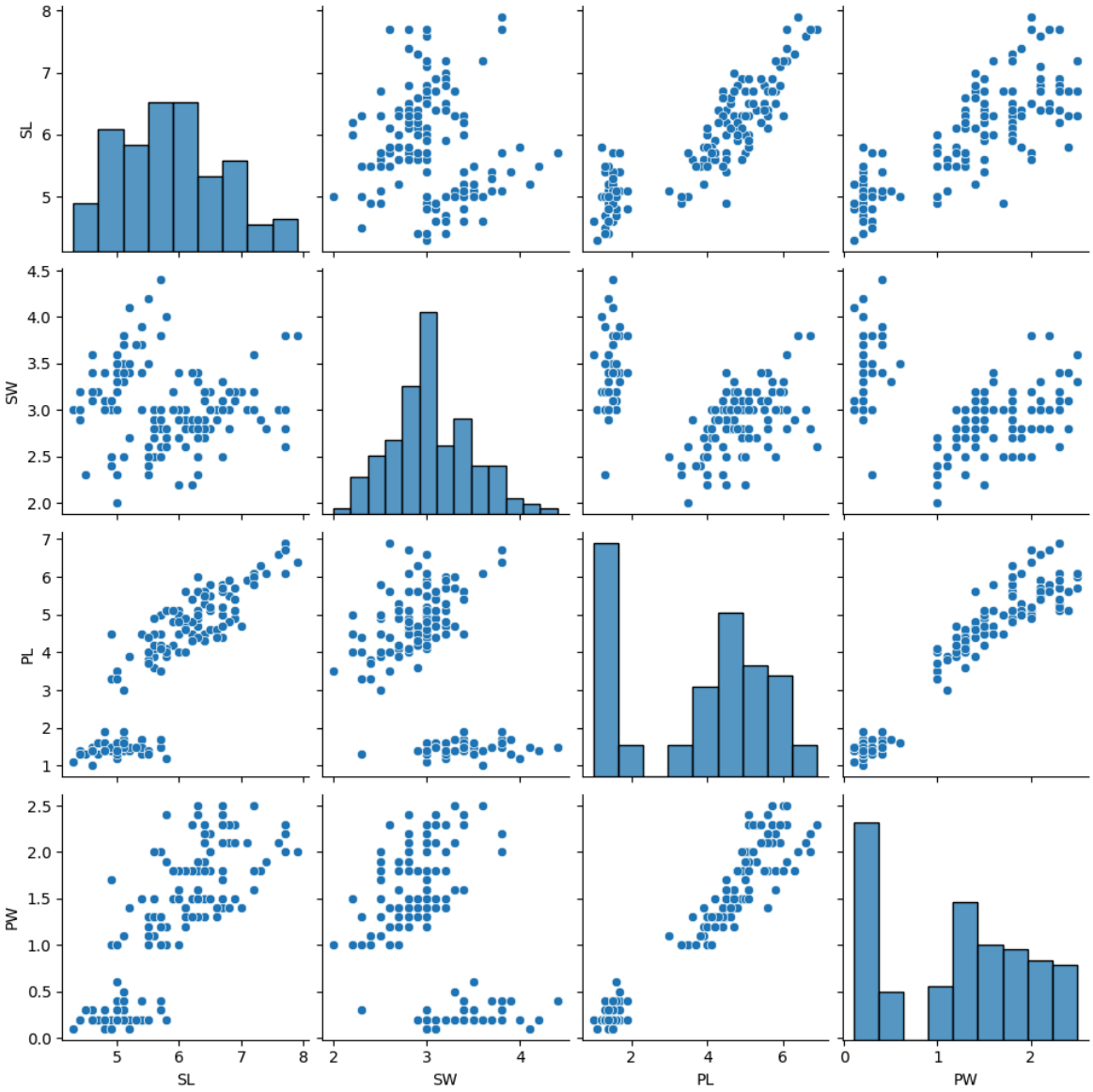


In [198…
```
plt.figure(figsize=(20,15))
sns.countplot(x=iris.SL,palette='hot')
plt.show()
```



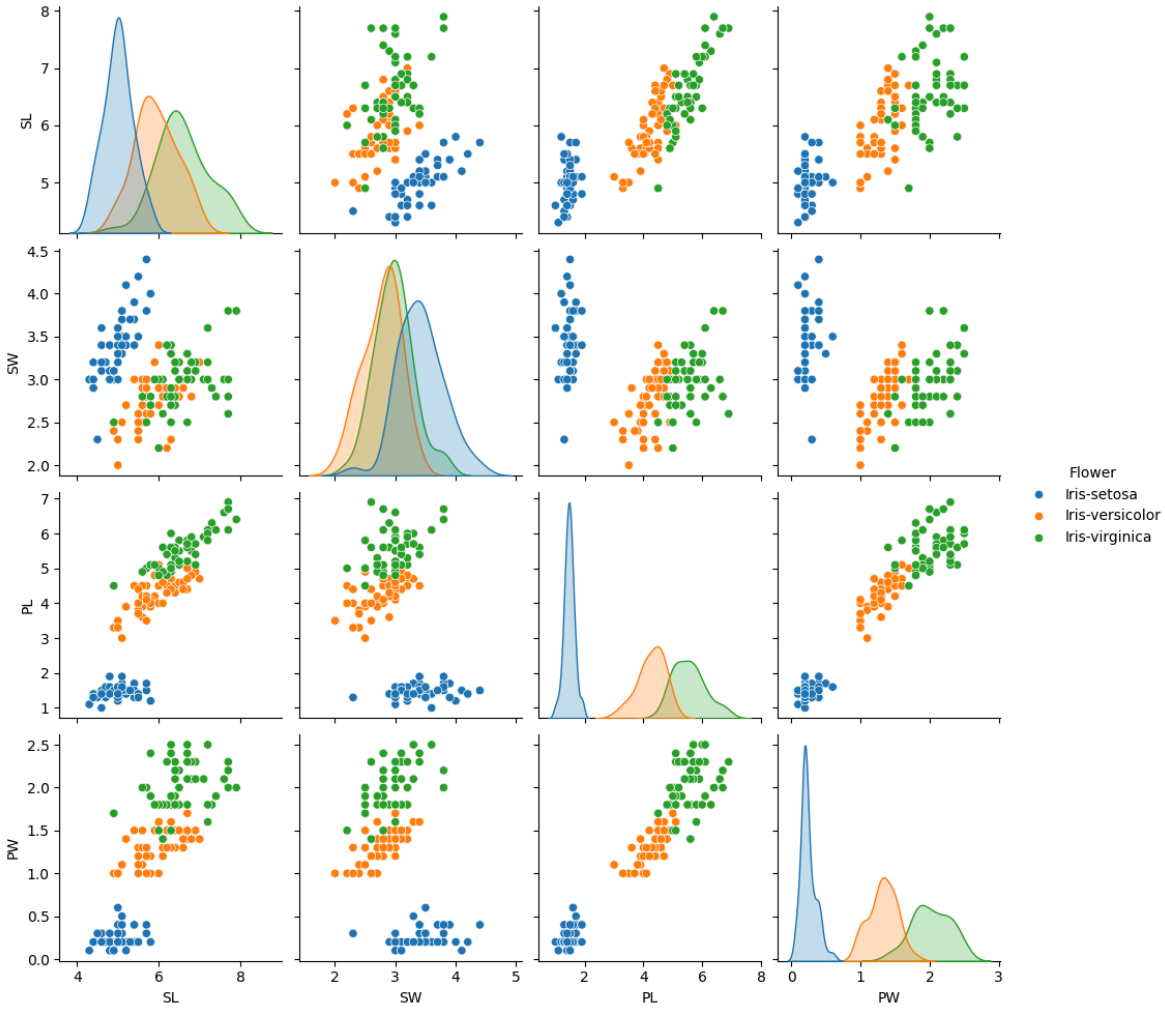
In [199…
```
#Pair Plot
sns.pairplot(iris)
```

```
plt.show()
```



In [200…

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



In [201…

```
#Heat Map
c = iris.corr()
c
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[201], line 2
      1 #Heat Map
----> 2 c = iris.corr()
      3 c

File C:\anaconda\Lib\site-packages\pandas\core\frame.py:11049, in DataFrame.corr
(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File C:\anaconda\Lib\site-packages\pandas\core\frame.py:1993, in DataFrame.to_num
py(self, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File C:\anaconda\Lib\site-packages\pandas\core\internals\managers.py:1694, in Blo
ckManager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File C:\anaconda\Lib\site-packages\pandas\core\internals\managers.py:1753, in Blo
ckManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

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

In [202…

```
iris.drop('Flower',axis=1,inplace=True)
iris.head(3)
```

Out[202…

	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

In [203…

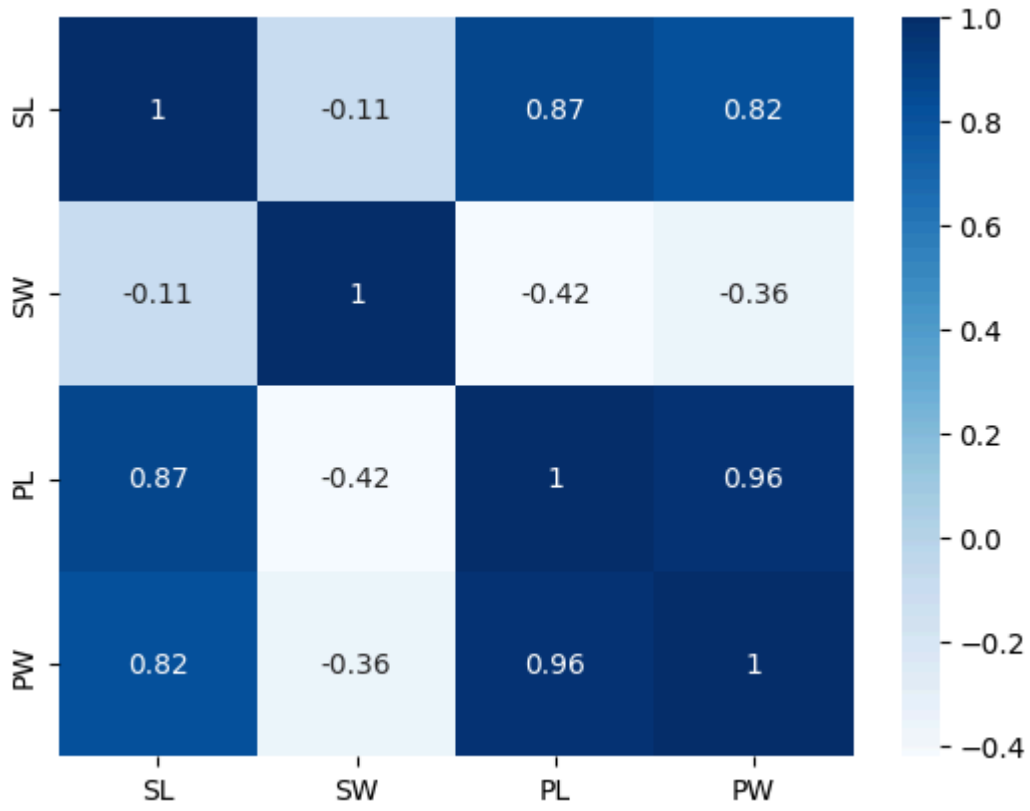
```
c = iris.corr()
c
```

Out[203…

	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 [207…

```
sns.heatmap(c,annot=True,cmap='Blues')
plt.show()
```



In [ ]: