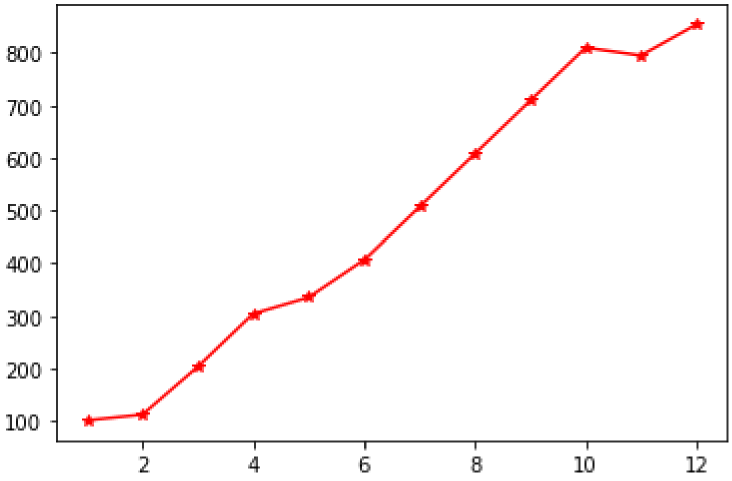
Line Plot

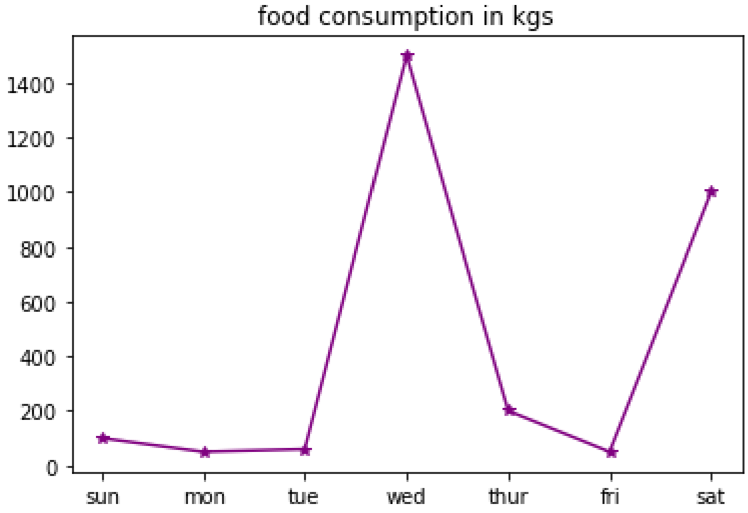
In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
month = np.arange(1,13)
prices = [101,112,203,304,335,406,507,608,709,810,795,854]
# we are plotting prices vs months here
# 'color' assigns the color to line plot
#'marker' assigns the shape of a data point

plt.plot(month,prices,color = 'r',marker = '*')
plt.show()
```

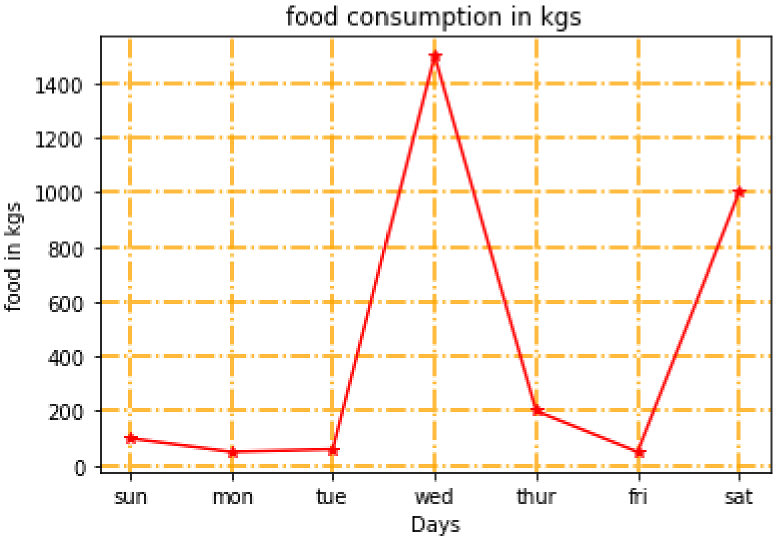


In [3]:

```
weeks = ['sun','mon','tue','wed','thur','fri','sat']
food = [100 , 50 , 60 , 1500 , 200 , 50 , 1000]
plt.plot(weeks,food,color = 'purple',marker = '*')
plt.title('food consumption in kgs')
plt.show()
```

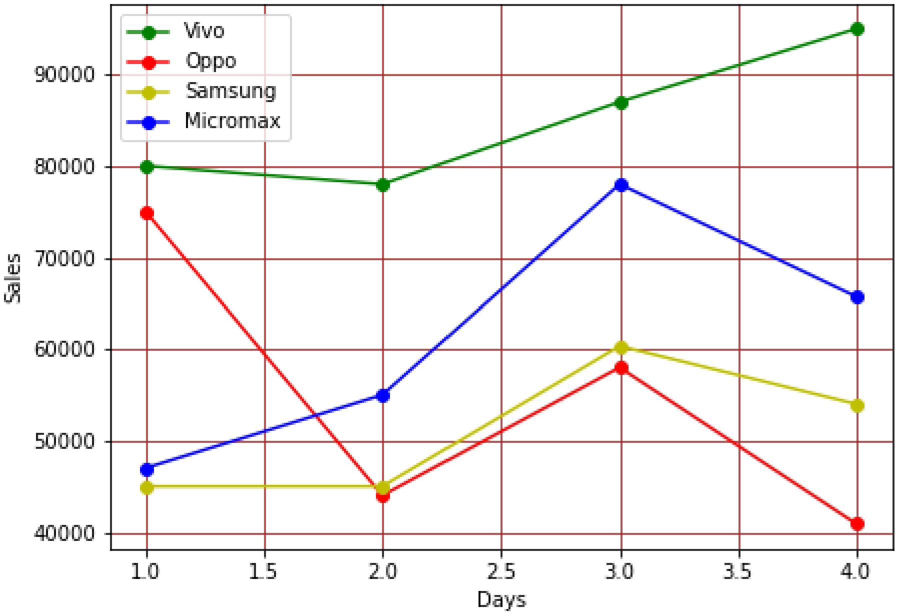


In [4]:
```
weeks = ['sun','mon','tue','wed','thur','fri','sat']
food = [100 , 50 , 60 , 1500 , 200 , 50 , 1000]
plt.plot(weeks,food,color = 'r',marker = '*')
# plt.title is used to label the plot
plt.title('food consumption in kgs')
# add axes labels
plt.xlabel('Days')
plt.ylabel('food in kgs')
# addding grid lines
plt.grid(linestyle = '-.',linewidth = '1.5',color = 'orange')
# plt.show() is used to display the plot
plt.show()
```



Multiple Line Plots

In [5]:
```
plt.figure(figsize=(7,5))  #setting the plot size
day = [1,2,3,4]
vivo_sales = [80000,78000,87000,95000]
oppo_sales = [75000,44000,58000,40888]
samsung_sales = [45000,45000,60333,54000]
micromax_sales = [47000,55000,78000,65700]
# plotting multiple lines
plt.plot(day,vivo_sales,color = 'g',marker = 'o',label = 'Vivo')
plt.plot(day,oppo_sales,color = 'r',marker = 'o',label = 'Oppo')
plt.plot(day,samsung_sales,color = 'y',marker = 'o',label = 'Samsung')
plt.plot(day,micromax_sales,color = 'b',marker = 'o',label = 'Micromax')
plt.xlabel('Days')
plt.ylabel('Sales')
plt.grid(color = 'brown')
plt.legend()

plt.show()
```



Scatter Plot

In [6]:

```
df_iris = pd.read_csv('iris.csv')
df_iris
```

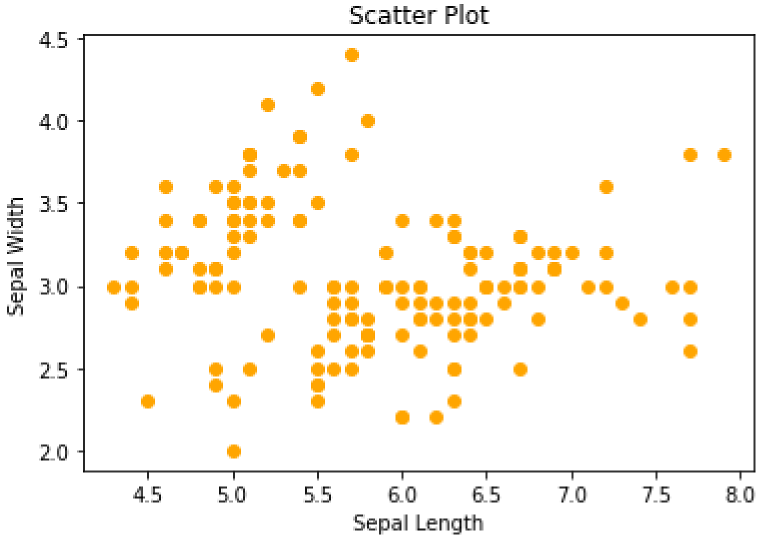
Out[6]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [7]:

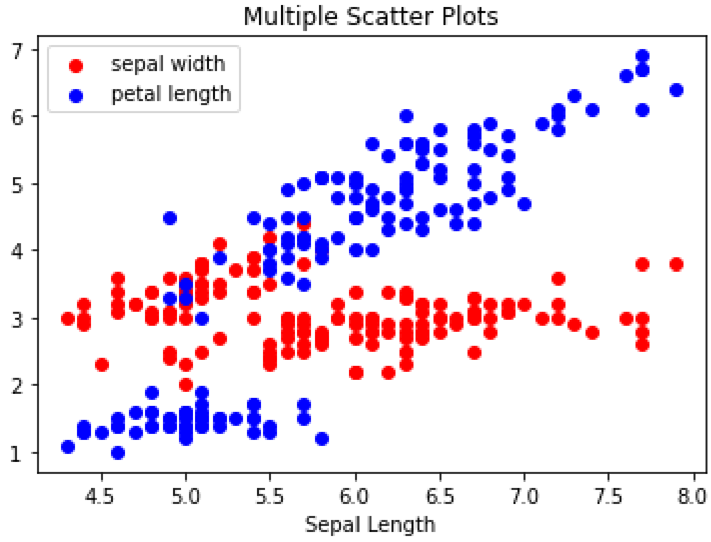
```
plt.scatter(x='sepal_length',y='sepal_width',data=df_iris,color = 'orange')
plt.title('Scatter Plot')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
```



Multiple Scatter Plots

In [8]:

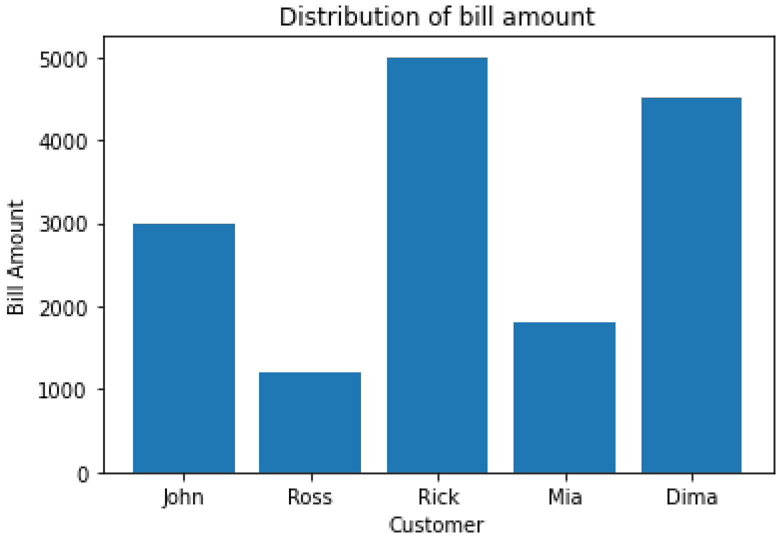
```
plt.scatter(x='sepal_length',y='sepal_width',color = 'red',data=df_iris,label=
'sepal width')
plt.scatter(x='sepal_length',y='petal_length',color='blue',data=df_iris,label=
'petal length')
plt.title('Multiple Scatter Plots')
plt.xlabel('Sepal Length')
plt.legend()
plt.show()
```



Bar Plot

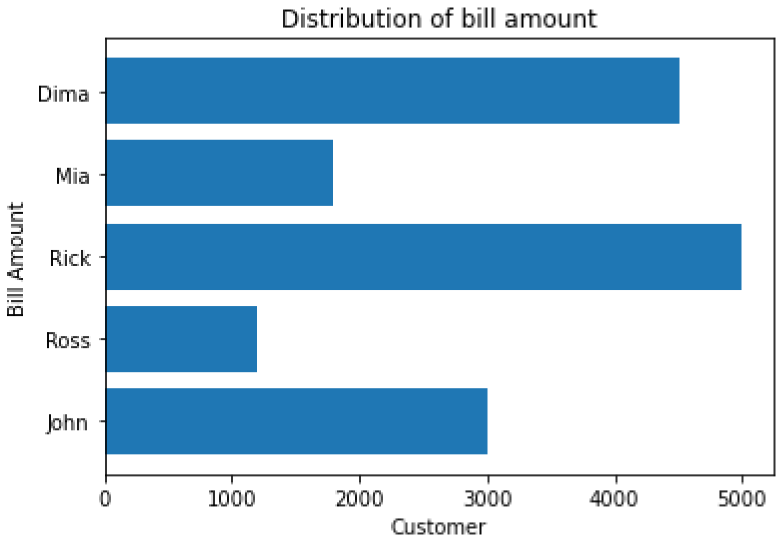
In [9]:

```
amount=[3000,1200,5000,1800,4500]
customer=('John','Ross','Rick','Mia','Dima')
# position of bar
# anything can be written in place of liftpositions
liftpositions=np.arange(len(customer))
plt.bar(x=liftpositions,height = amount)
# add label to each bar
plt.xticks(liftpositions,customer)
plt.title('Distribution of bill amount')
plt.xlabel('Customer')
plt.ylabel('Bill Amount')
plt.show()
```



Horizontal bar Plot

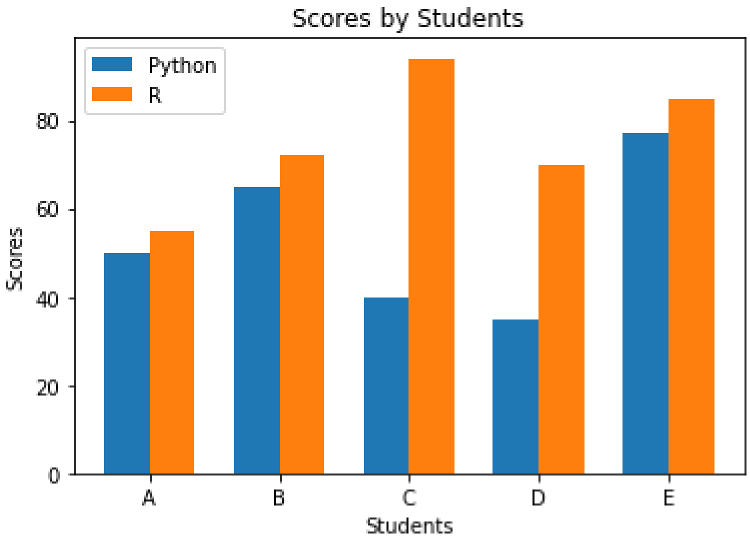
In [10]:
```python
amount=[3000,1200,5000,1800,4500]
customer=('John','Ross','Rick','Mia','Dima')
# position of bar
liftpositions=np.arange(len(customer))
# 'y' represents categorical variable
# 'width' represents value of each bar
plt.barh(y=liftpositions,width = amount)
# add label to each bar
plt.yticks(liftpositions,customer)
plt.title('Distribution of bill amount')
plt.xlabel('Customer')
plt.ylabel('Bill Amount')
plt.show()
```



Grouped Bar Plot

In [11]:

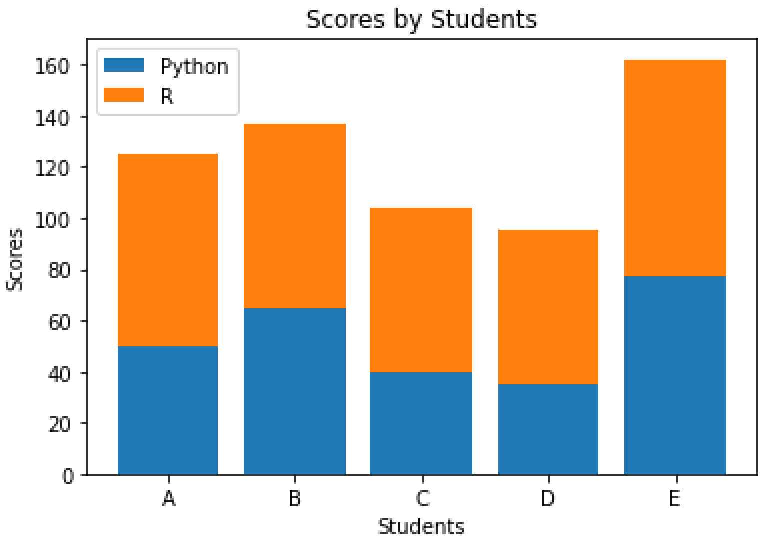
```
Python_marks = (50,65,40,35,77)
R_marks = (55,72,94,70,85)
# set the position of the bar
index = np.arange(5)
# plot a bar plot for each subject
# 'x' represents position of the bar
# 'height' represents value of the bar
# 'width' represents width of the bar
# 'label' assigns label to the bar
plt.bar(x=index,height=Python_marks,width = 0.35 , label = 'Python')
plt.bar(x=index + 0.35,height = R_marks, width = 0.35, label='R')
# add axes and plot label
plt.xlabel('Students')
plt.ylabel('Scores')
plt.title('Scores by Students')
# 'ticks' assigns position of the label
# 'labels' assigns label to each bar
plt.xticks(ticks = index + 0.35 / 2, labels = ('A','B','C','D','E'))
plt.legend()
plt.show()
```



Stacked Bar Plot

In [12]:

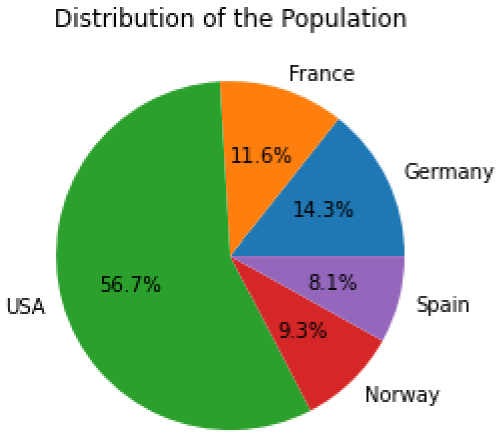
```python
Python_marks = (50,65,40,35,77)
R_marks = (75,72,64,60,85)
# set the position of the bar
index = np.arange(5)
# plot a bar plot for each subject
# 'x' represents position of the bar
# 'height' represents value of the bar
# 'bottom' represents the bar plot at the bottom
# 'label' assigns label to the bar
plt.bar(x = index,height = Python_marks, label='Python')
plt.bar(x = index,height = R_marks, bottom = Python_marks, label='R')
# add axes and plot label
plt.xlabel('Students')
plt.ylabel('Scores')
plt.title('Scores by Students')

# 'ticks' assigns position of label
# 'labels' assigns label to each bar
plt.xticks(ticks = index, labels = ('A','B','C','D','E'))
plt.legend()
plt.show()
```



Pie Plot

In [17]:

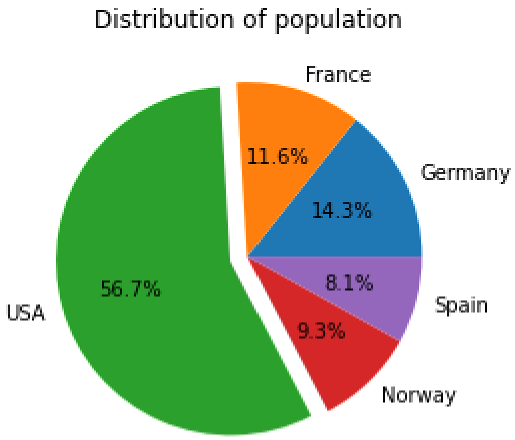
```
countries = ('Germany','France','USA','Norway','Spain')
population = [8.28,6.7,32.72,5.37,4.67]
# 'x' represents values to plot
# 'labels' represent categories
# 'autopct' returns the percentages with one decimal value
plt.pie(x = population, labels = countries, autopct = '%1.1f%%')
# set the plot label
plt.title('Distribution of the Population')
# display the plot
plt.show()
```



Exploded Pie Plot

In [24]:

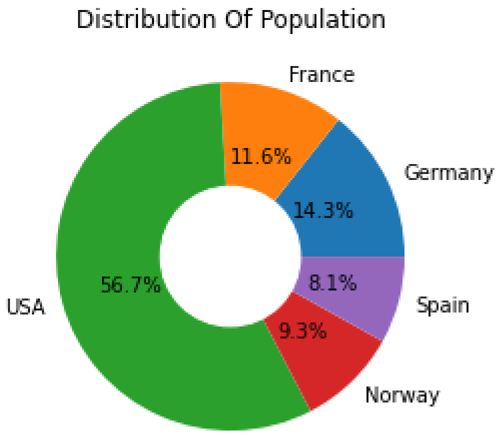
```
countries = ('Germany','France','USA','Norway','Spain')
population = [8.28,6.7,32.72,5.37,4.67]
# to explode the slice with highest population
explode = (0,0,0.1,0,0)
# 'x' represents the values to plot
# 'labels' represent categories
# 'explode' returns the exploded pie plot
# 'autopct' returns the percentage with one decimal value
# set the plot label
plt.pie(x = population, labels = countries, autopct = '%1.1f%%',explode = expl
ode)
plt.title('Distribution of population')
# display the plot
plt.show()
```



Donut Pie Plot

In [33]:

```
countries = ('Germany','France','USA','Norway','Spain')
population = [8.28,6.7,32.72,5.37,4.67]
# 'x' represents the values to plot
# 'labels' represent categories
# 'autopct' returns the percentage with one decimal value
plt.pie(x = population, labels = countries, autopct = '%1.1f%%')
# add a circle at the center of the pie plot
# 'xy' assigns the center of the circle
# 'radius' assigns the radius of the circle
# 'color' assigns the color of the circle
circle = plt.Circle(xy = (0,0), radius = 0.4 , color = 'white')
plt.gcf()
plt.gca().add_artist(circle)
# set the plot label
plt.title('Distribution Of Population')
# display the plot
plt.show()
```



Histogram

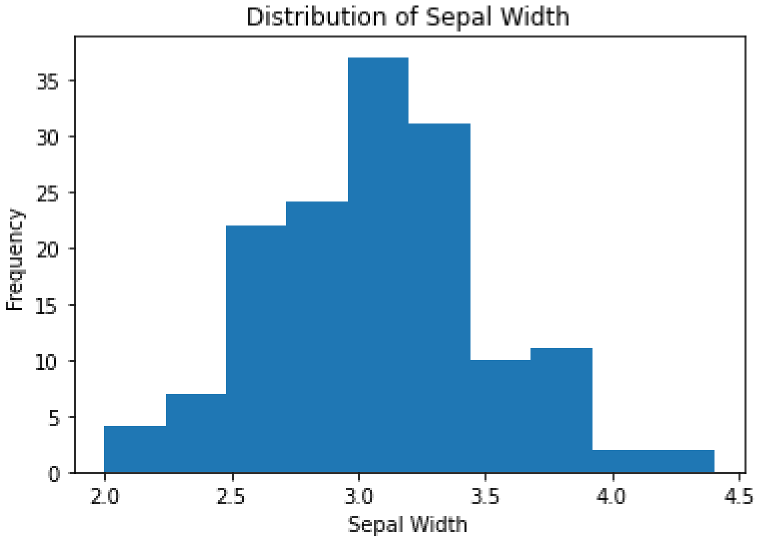
In [34]:

```
df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

Out[34]:

	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

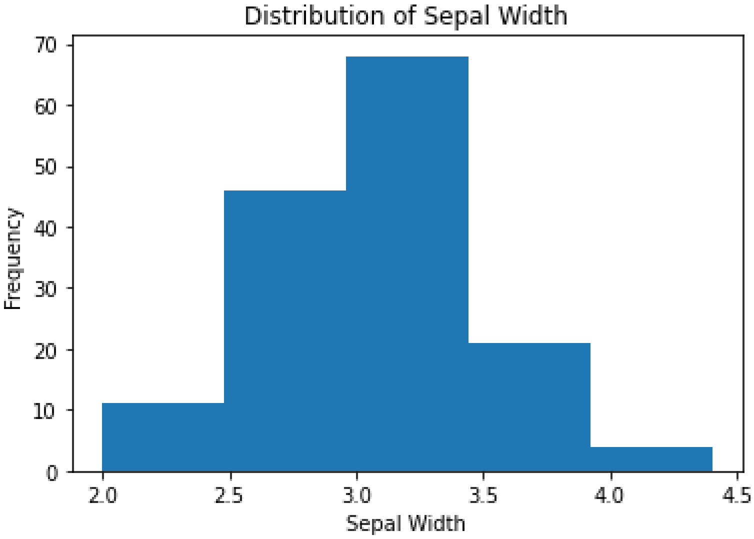
```
In [36]: # plot the histogram
         # 'x' represents the variable to plot the histogram
         # plot the histogram to check the distribution of the variable,'sepal width'
         plt.hist(x = df_iris['sepal_width'])
         # add axes plot labels
         plt.title('Distribution of Sepal Width')
         plt.xlabel('Sepal Width')
         plt.ylabel('Frequency')
         # display the plot
         plt.show()
```



plot a histogram with 5 bins (bars)

In [37]:

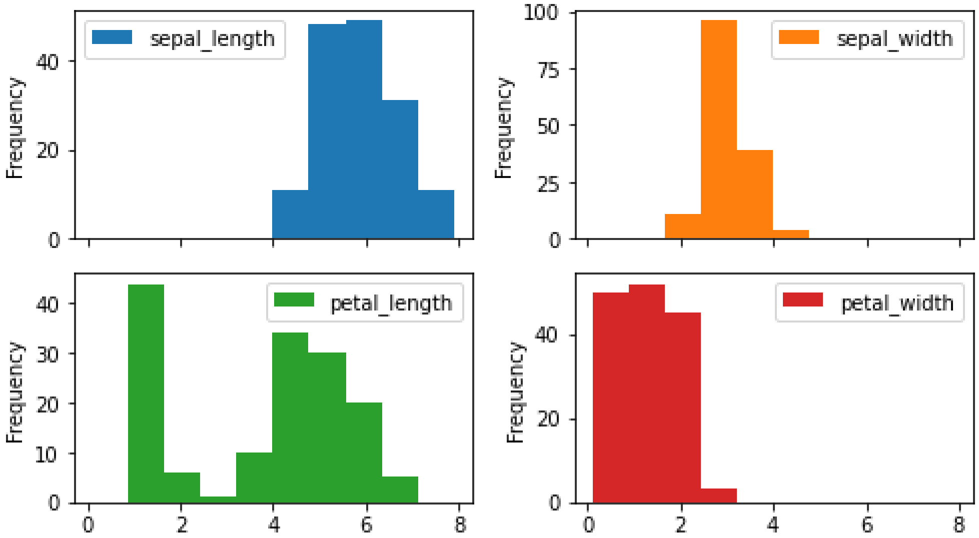
```
# plot the histogram
# 'x' represents the variable to plot the histogram
# plot the histogram to check the distribution of the variable,'sepal width'
# 'bins' return a histogram with specified number of bars
plt.hist(x = df_iris['sepal_width'], bins = 5)
# add axes plot labels
plt.title('Distribution of Sepal Width')
plt.xlabel('Sepal Width')
plt.ylabel('Frequency')
# display the plot
plt.show()
```



Create Multiple Histograms

In [40]:

```
# plot the multiple histograms
# 'subplots = True' returns the multiple plots as subplots
# 'Layout' assigns the Layout for the subplots
# 'figsize' set the figure size
# 'sharex' and 'sharey' controls the properties of x any y axis respectively
df_iris.plot.hist(subplots = True , layout = (2,2), figsize = (7,4), sharex =
True, sharey = False)
# to adjust the subplot
plt.tight_layout()
# display the plot
plt.show()
```



Box Plot

In [41]:

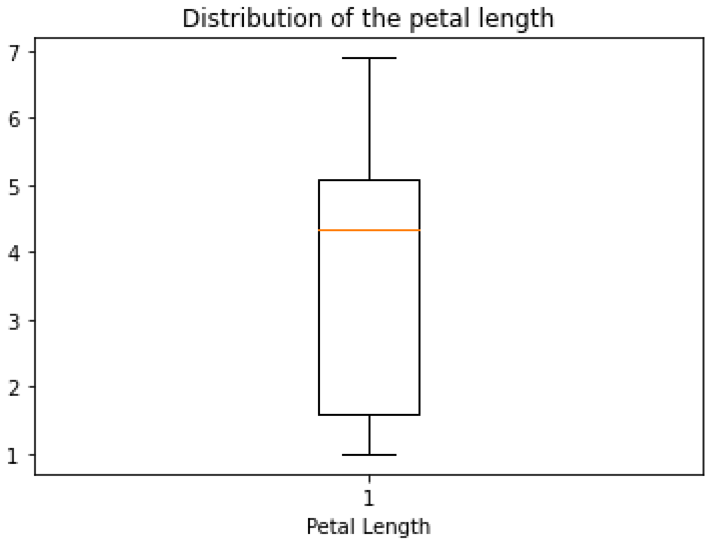
```
df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

Out[41]:

	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 [43]:

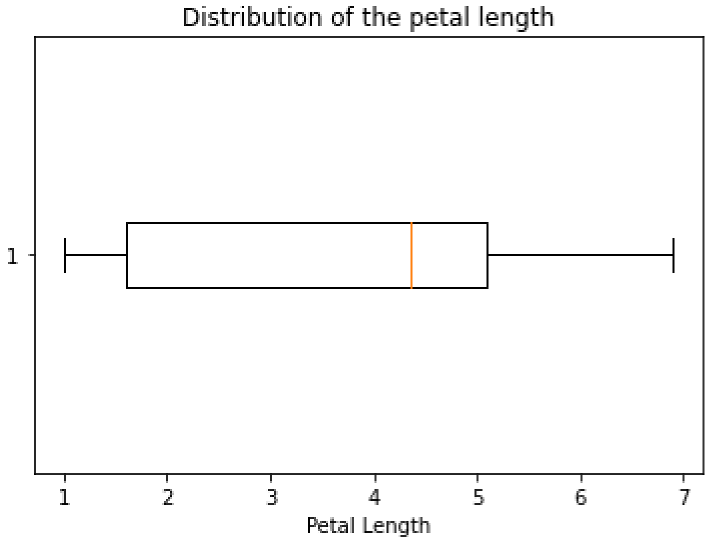
```
# create a boxplot
# 'x' represents the data to plot a boxplot
plt.boxplot(x = df_iris['petal_length'])
# add the axis and plot label
plt.title('Distribution of the petal length')
plt.xlabel('Petal Length')
# display the plot
plt.show()
```



Horizontal Box Plot

In [44]:

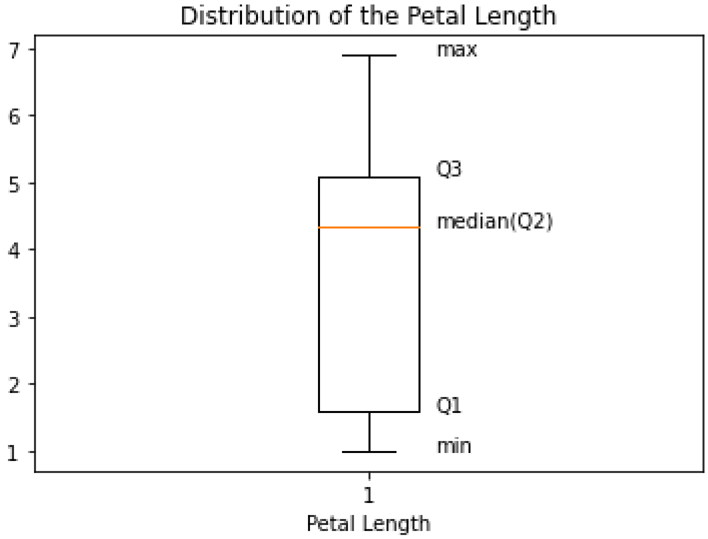
```
# create a boxplot
# 'x' represents the data to plot a boxplot
plt.boxplot(x = df_iris['petal_length'], vert = False)
# add the axis and plot label
plt.title('Distribution of the petal length')
plt.xlabel('Petal Length')
# display the plot
plt.show()
```



Add five number summary to box plot

In [47]:

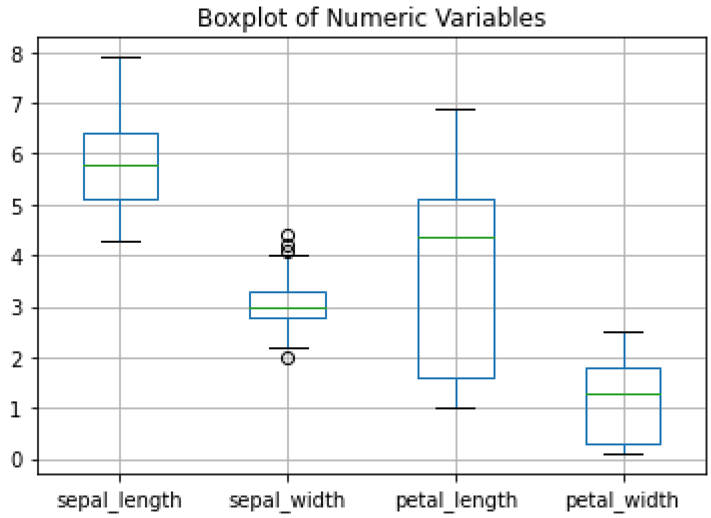
```
# create a boxplot
# 'x' represents the data to plot a box plot
plt.boxplot(x = df_iris['petal_length'])
# add labels for five number summary
# 'x' and 'y' represents the position of the text
# 's' represents the text
plt.text(x = 1.1, y = df_iris['petal_length'].min(),s='min')
plt.text(x = 1.1, y = df_iris['petal_length'].quantile(0.25), s = 'Q1')
plt.text(x = 1.1, y = df_iris['petal_length'].median(), s = 'median(Q2)')
plt.text(x = 1.1, y = df_iris['petal_length'].quantile(0.75), s = 'Q3')
plt.text(x = 1.1, y = df_iris['petal_length'].max(), s = 'max')
# add the axis and plot label
plt.title('Distribution of the Petal Length')
plt.xlabel('Petal Length')
# display the plot
plt.show()
```



Plot the box plot of all the numeric variables in the data

In [48]:

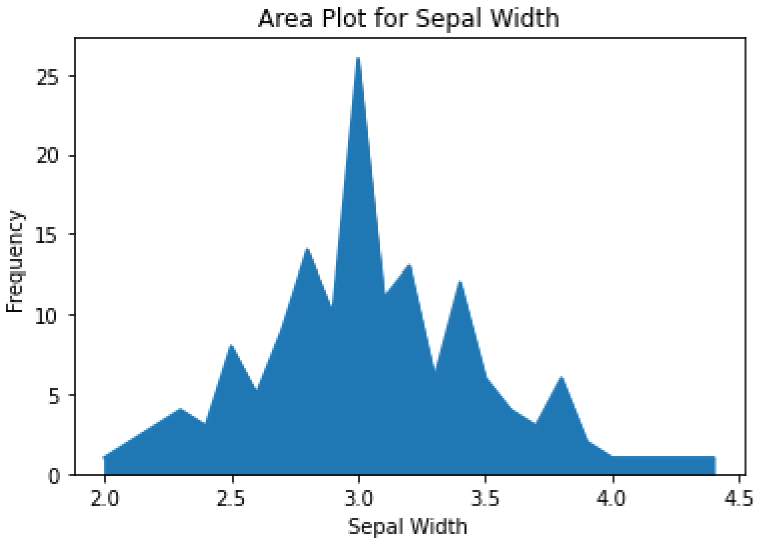
```
# plot box plot of all the numeric variables
df_iris.boxplot()
# add plot label
plt.title('Boxplot of Numeric Variables')
# display the plot
plt.show()
```



Area Plot

In [50]:

```
# create the area plot
# area() returns the area plot
df_iris['sepal_width'].value_counts().sort_index().plot.area()
# add axes and plot labels
plt.title('Area Plot for Sepal Width')
plt.xlabel('Sepal Width')
plt.ylabel('Frequency')
# display the plot
plt.show()
```



In [9]:
```
import seaborn as sns
import plotly
plotly.offline.init_notebook_mode(connected=True)
#import the library
import plotly.express as px
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-9-ff0ddeb06052> in <module>
      1 import seaborn as sns
----> 2 import plotly
      3 plotly.offline.init_notebook_mode(connected=True)
      4 #import the library
      5 import plotly.express as px

ModuleNotFoundError: No module named 'plotly'
```

Visualization Using Seaborn

Strip Plot

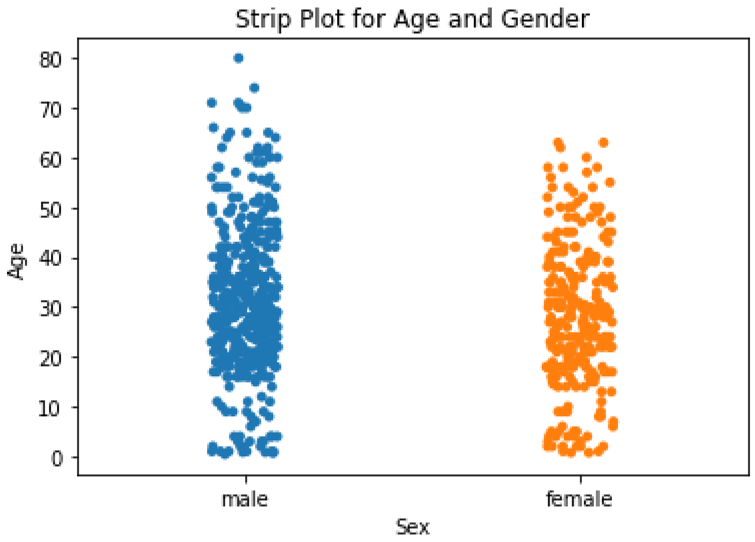
In [7]:
```
df_titanic = pd.read_excel('titanic_info.xlsx')
df_titanic.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

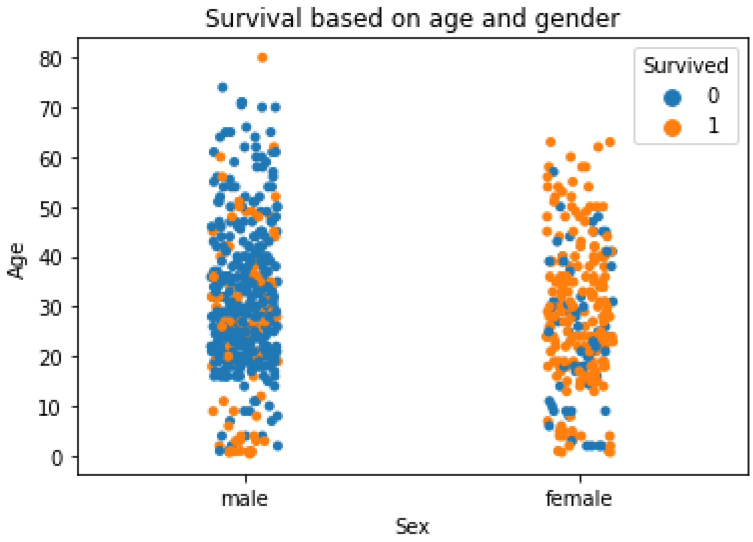
In [10]:

```
# plot a strip plot
# 'x' represents variable on x-axis
# 'y' represents variable on y-axis
# 'data' represents the DataFrame
sns.stripplot(x = 'Sex', y = 'Age' , data = df_titanic)
# add the plot label
plt.title('Strip Plot for Age and Gender')
# display the plot
plt.show()
```



Add the one more categorical variable to strip plot using the parameter , 'hue'

```
In [11]: #plot a strip plot
         #'hue' adds one more variable to the plot
         #'data' represents the DataFrame
         sns.stripplot(x = 'Sex' , y = 'Age' , hue = 'Survived' , data = df_titanic)
         #add the plot label
         plt.title('Survival based on age and gender')
         #display the plot
         plt.show()
```



Violin Plot

In [12]:

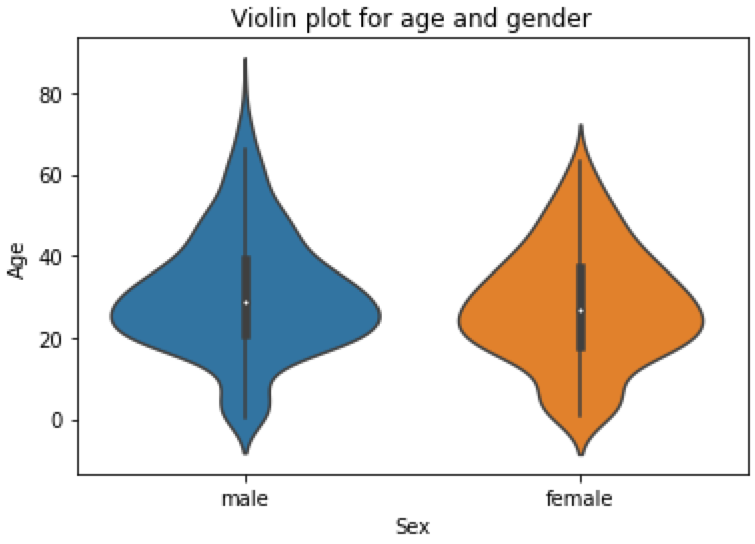
```
df_titanic = pd.read_excel('titanic_info.xlsx')
df_titanic.head()
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

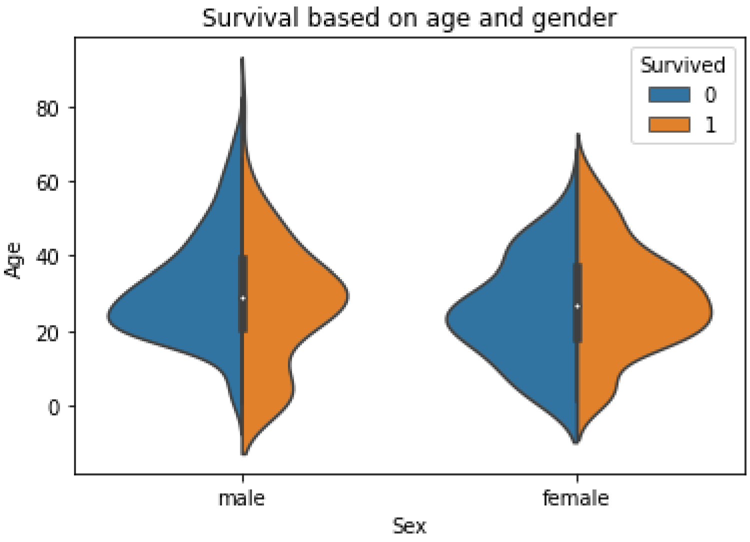
In [14]:

```
# plot a violin plot
sns.violinplot(x = 'Sex',y = 'Age' , data = df_titanic)
plt.title('Violin plot for age and gender')
plt.show()
```



violin plot can be divided into two halfs , surviving and non surviving passengers

In [15]:
```
# 'hue' adds one more variable to the plot
# 'split' returns the plot splitted in two halves
sns.violinplot(x = 'Sex' , y = 'Age' , data = df_titanic , hue = 'Survived' ,
split = True)
plt.title('Survival based on age and gender')
plt.show()
```



Swarm Plot

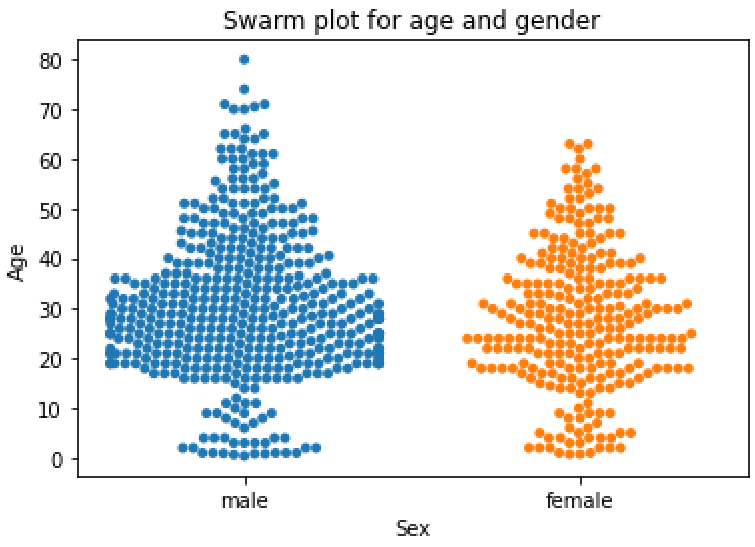
In [16]:
```
df_titanic = pd.read_excel('titanic_info.xlsx')
df_titanic.head()
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

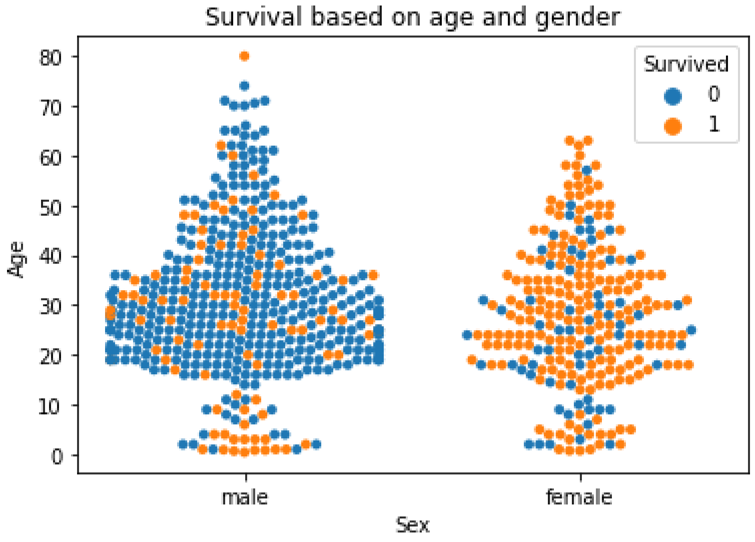
In [17]:
```
sns.swarmplot(x = 'Sex' , y = 'Age' , data = df_titanic)
plt.title('Swarm plot for age and gender')
plt.show()
```

```
C:\Users\Vishal Venkata\anaconda3\lib\site-packages\seaborn\categorical.py:13
11: RuntimeWarning: invalid value encountered in less
  off_low = points < low_gutter
C:\Users\Vishal Venkata\anaconda3\lib\site-packages\seaborn\categorical.py:13
15: RuntimeWarning: invalid value encountered in greater
  off_high = points > high_gutter
```



Add one more categorical variable 'Survived' to the swarm plot

In [18]:
```
sns.swarmplot(x = 'Sex' , y = 'Age' , data = df_titanic , hue = 'Survived')
plt.title('Survival based on age and gender')
plt.show()
```



In [19]:

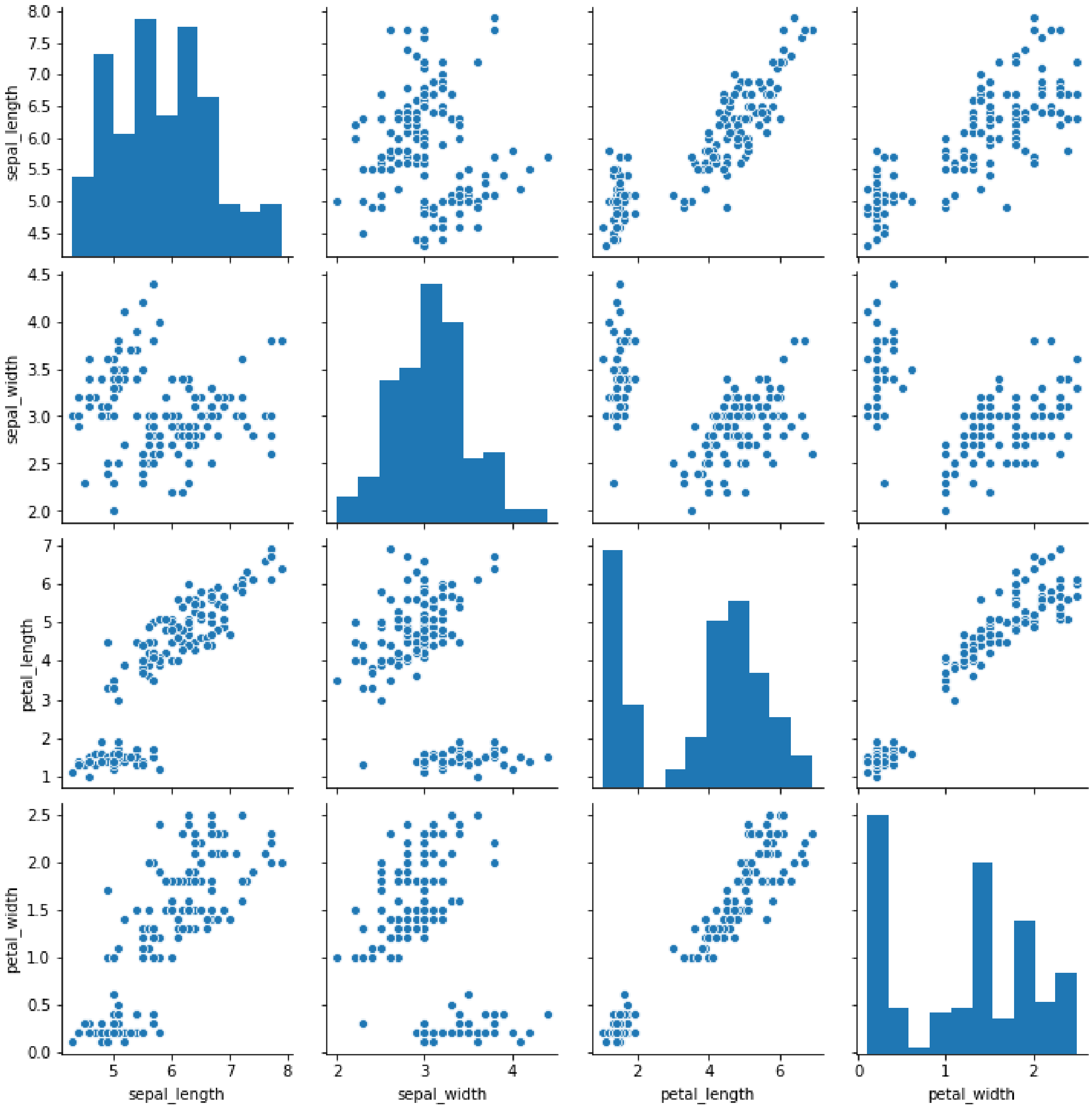
```
df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

Out[19]:

	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 [21]:

```
sns.pairplot(data = df_iris)
plt.show()
```



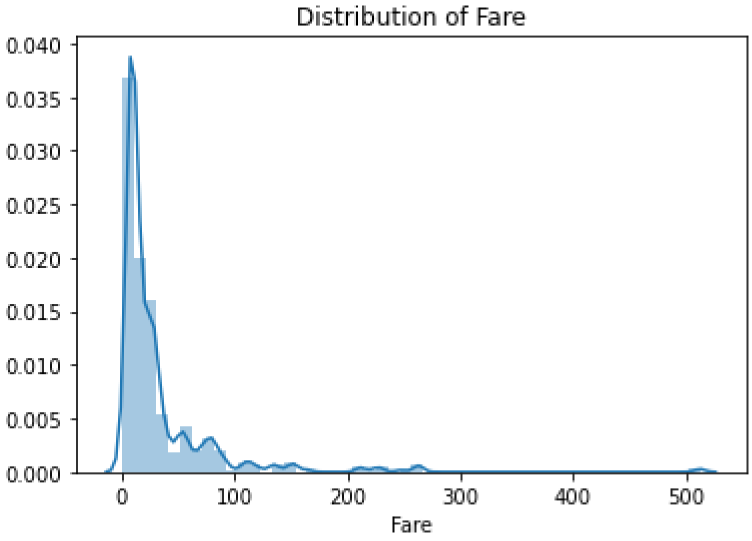
Distribution Plot

In [22]:
```
df_titanic = pd.read_excel('titanic_info.xlsx')
df_titanic.head()
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

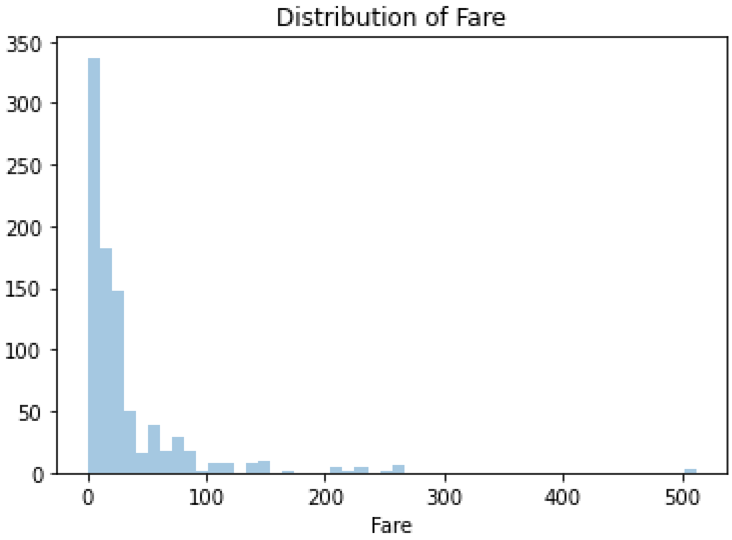
In [23]:
```
sns.distplot(a = df_titanic['Fare'])
plt.title('Distribution of Fare')
plt.show()
```



without KDE

In [24]:

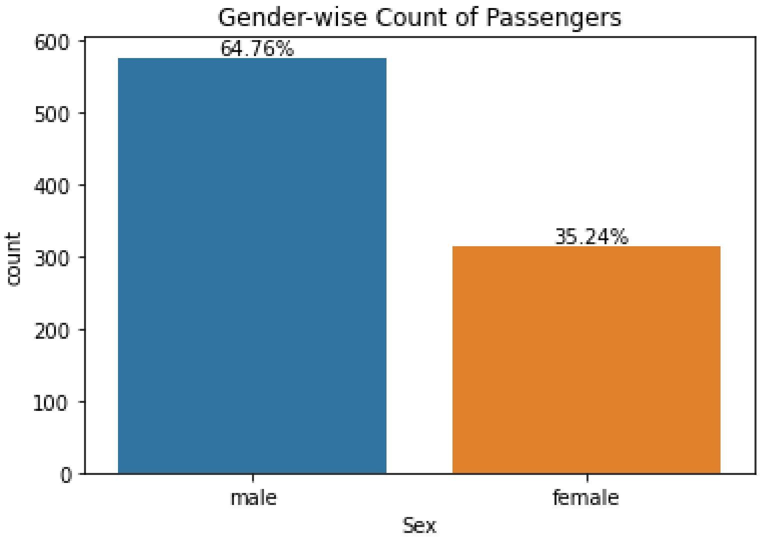
```
sns.distplot(a = df_titanic['Fare'] , kde = False)
plt.title('Distribution of Fare')
plt.show()
```



Count Plot

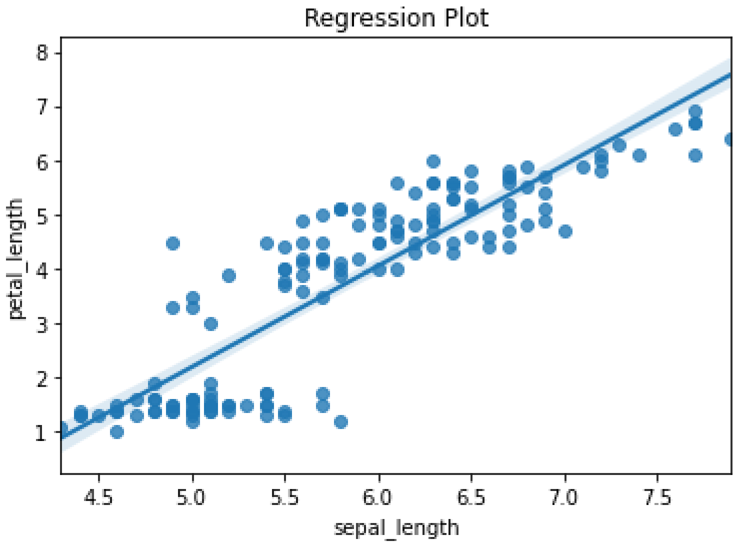
In [25]:

```
sns.countplot(x = 'Sex' , data = df_titanic)
plt.text(x = -0.1,y=580,s= str(round(df_titanic.Sex.value_counts()[0]/len(df_titanic)*100, 2)) + '%')
plt.text(x = 0.9, y = 320 , s = str(round(df_titanic.Sex.value_counts()[1]/len(df_titanic)*100, 2)) + '%')
plt.title('Gender-wise Count of Passengers')
plt.show()
```

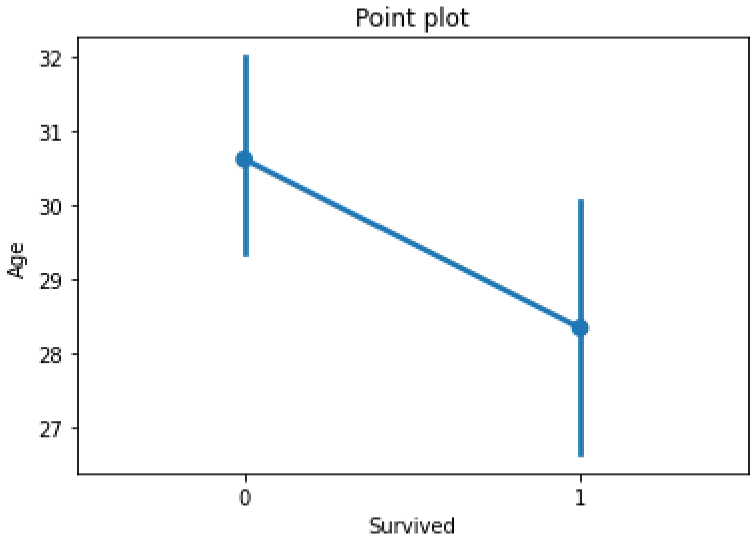


Regression Plot

In [27]:
```
sns.regplot(x = 'sepal_length' , y = 'petal_length' , data = df_iris)
plt.title('Regression Plot')
plt.show()
```

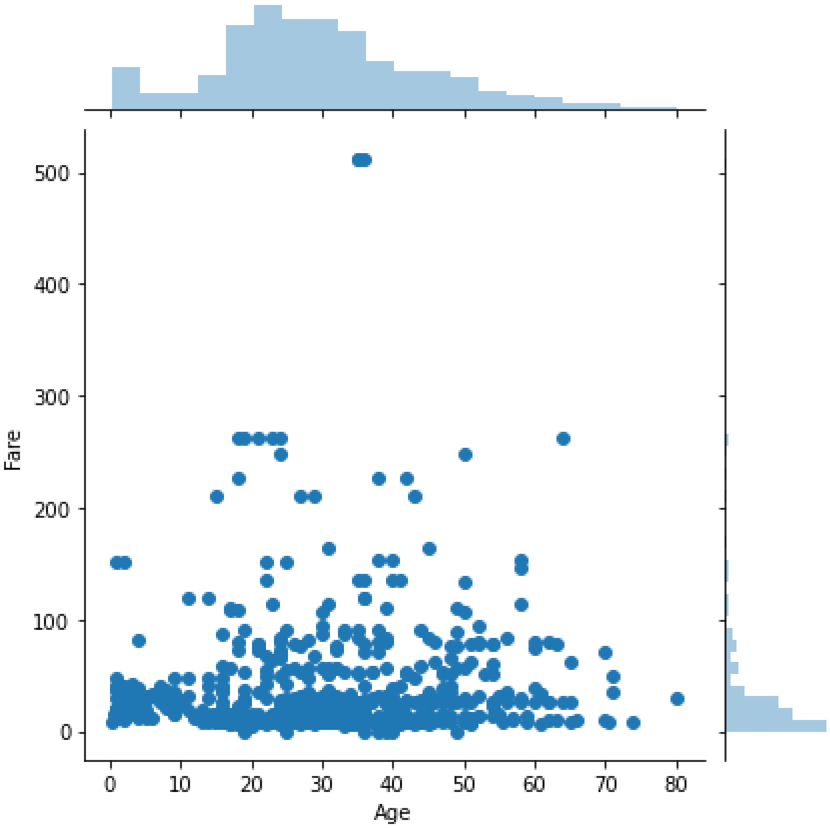


In [28]:
```
sns.pointplot(x = 'Survived' , y = 'Age' , data = df_titanic)
plt.title('Point plot')
plt.show()
```



Joint Plot

In [30]:
```
sns.jointplot(x = 'Age', y = 'Fare' , data = df_titanic)
plt.show()
```



Heatmap

In [31]:
```
df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

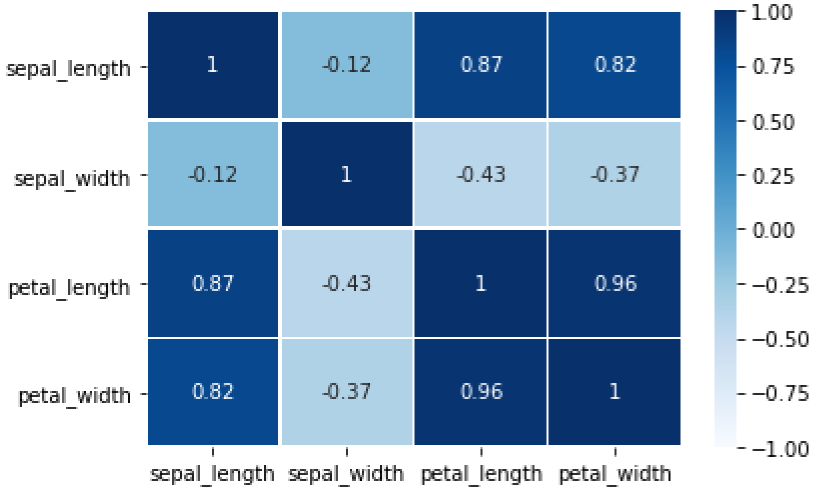
Out[31]:

	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

Heatmap

In [32]:

```
#plot heatmap to study correlation
#'data' returns the data for the heatmap
#'annot' returns the correlation values on heatmap
#'linewidth' add lines between each cell
#'cmap' assigns the color to each cell
#'cbar' returns the color bar beside the heatmap
#'vmin' and 'vmax' assigns the minimum and maximum values to anchor the color bar
sns.heatmap(data = df_iris.corr(), annot = True, linewidth = 0.5,
            cmap = 'Blues', cbar = True , vmin = -1 , vmax = 1)
plt.show()
```



The variables 'petal width' and 'petal length' are highly positively correlated

Visualization using plotly

In [9]:

```
import plotly.express as px
#offline version of plotly
import plotly
plotly.offline.init_notebook_mode(connected = True)
```

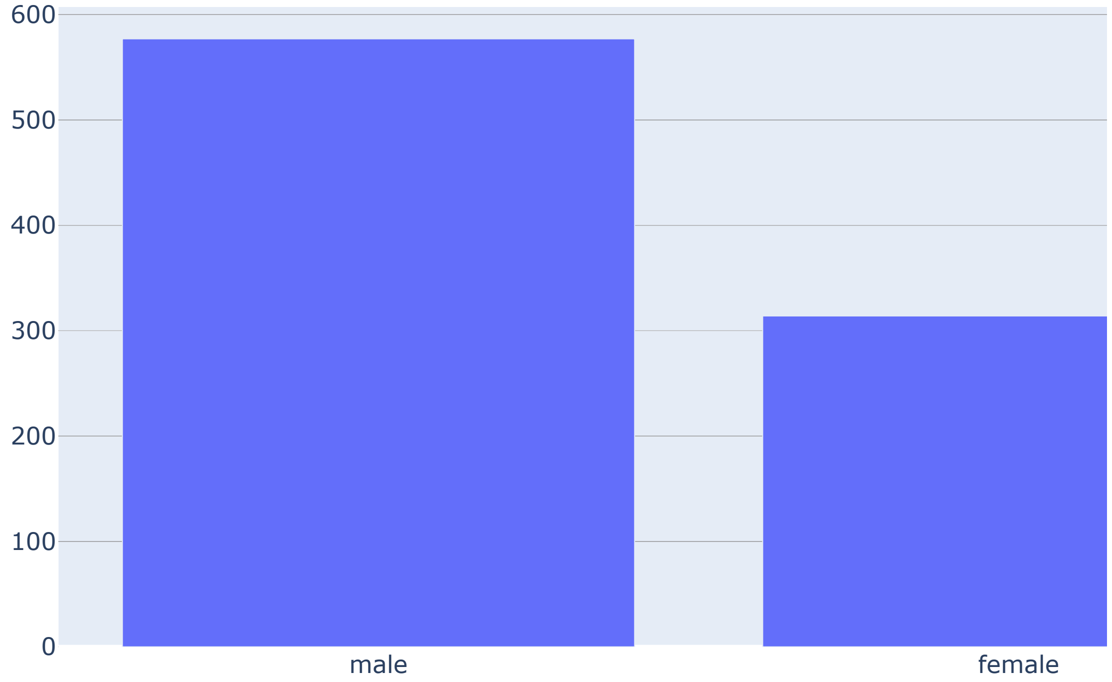
In [5]:
```
df_titanic = pd.read_excel('titanic_info.xlsx')
df_titanic.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

Bar Plot

In [6]:
```
import plotly.graph_objs as go
fig = go.Figure(layout={'autosize':False ,'height':500 , 'width':800})
fig.add_trace(go.Bar(x = df_titanic['Sex'], y = df_titanic.Sex.value_counts(),
name = 'Sex'))
```



Histogram

In [7]:
```
fig = px.histogram(data_frame = df_titanic,x = "Fare")
fig.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-eb9342d1ae83> in <module>
----> 1 fig = px.histogram(data_frame = df_titanic,x = "Fare")
      2 fig.show()

NameError: name 'px' is not defined
```

```
In [11]: df_iris = pd.read_csv('iris.csv')
         df_iris.head()
```

Out[11]:

	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

Boxplot

```
In [ ]: # plot a boxplot
```

```
In [ ]: # 'y' assigns the variable to plot a boxplot
```

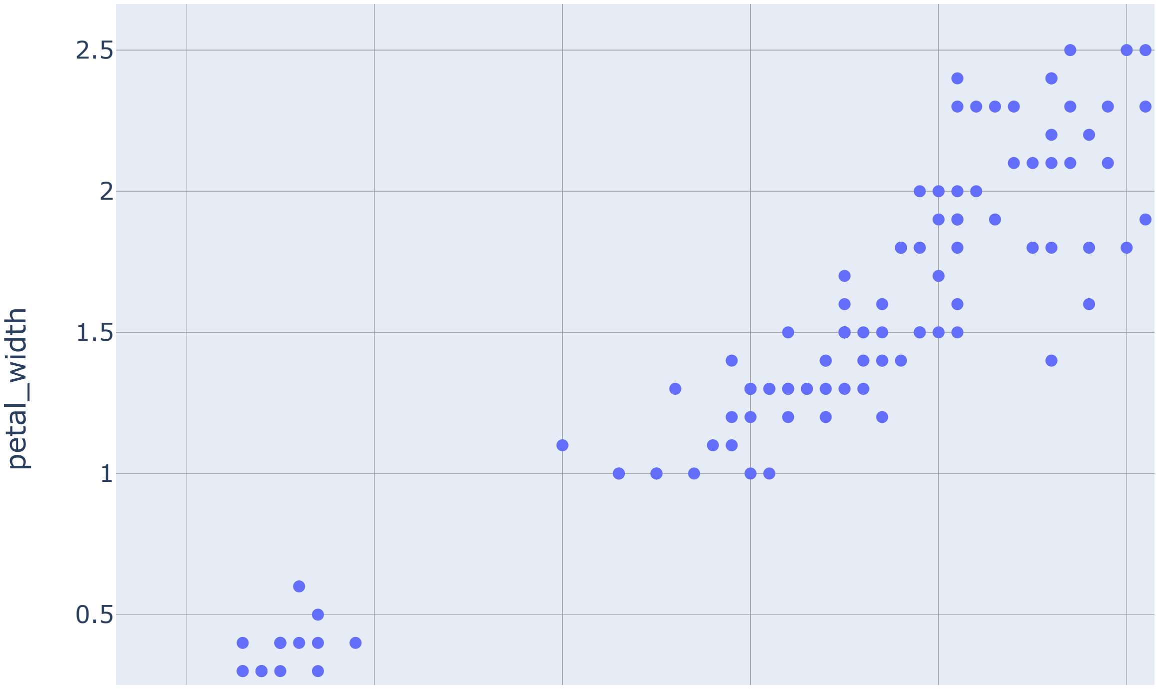
```
In [1]: fig = px.box(df_iris, y = 'sepal_length')
        fig.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a6b97823af34> in <module>
----> 1 fig = px.box(df_iris, y = 'sepal_length')
      2 fig.show()

NameError: name 'px' is not defined
```

The boxplot shows that , the 'sepal length' is not significantly skewed

Scatter Plot

In [15]:
```
fig = px.scatter(df_iris,x = 'petal_length' , y = 'petal_width')
fig.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: