In [ ]:

EDA --> Exploratory Data analysis

Parts of EDA

1. Univirate Analysis --> Analysis of single independent column
2. Bivariate Analysis --> Analysis of two columns
3. Multivariate Analysis --> Analysis of more than one column

Data Types

1. Numerical Data --> continous data --> age(year,date,month), height, weight
2. Categorical Data --> Descrete data --> total no of employees

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt # visualizatio liberary
import seaborn as sns  # matplotlib updated version
```

In [3]:
```
df=pd.read_csv("F:\\New folder\\ML\\CSV files\\titanic.csv")
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [6]:
```
df.shape
```

Out[6]:
```
(418, 12)
```

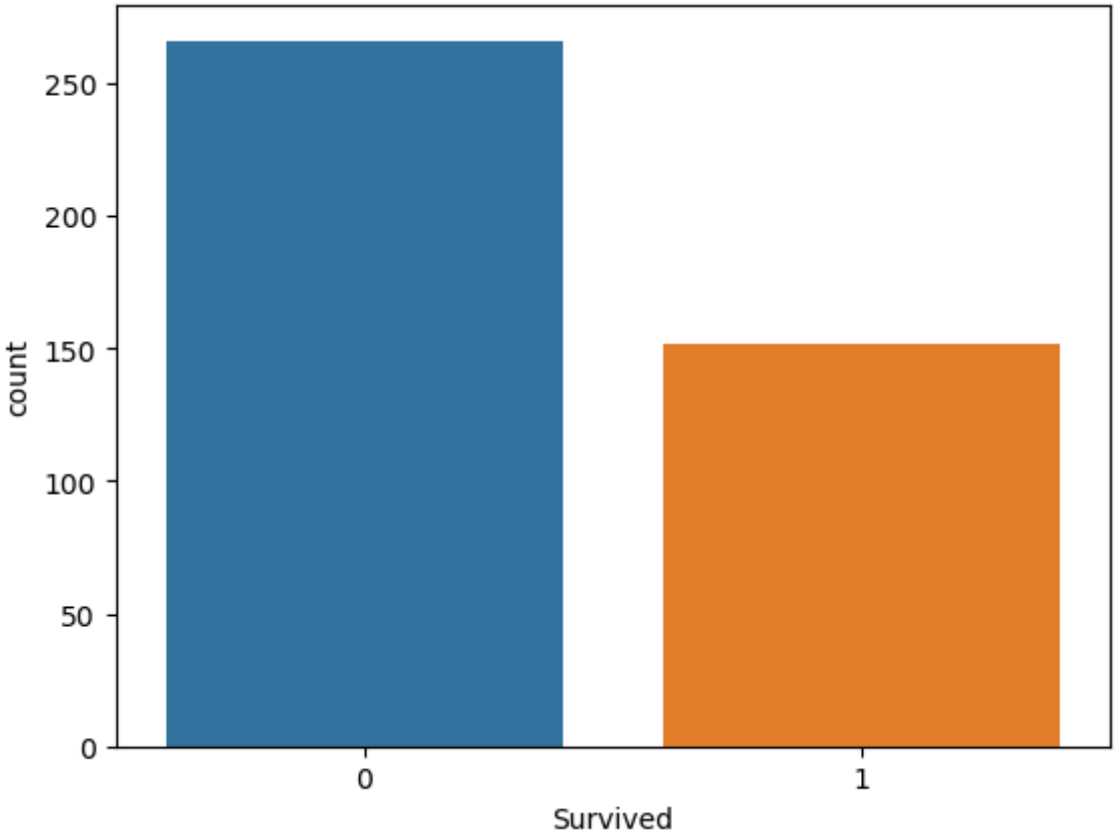
1) Univeriate Analysis

In [7]:
```
df.columns
```

Out[7]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [8]:
```
sns.countplot(x=df['Survived'])
```

Out[8]:
```
<Axes: xlabel='Survived', ylabel='count'>
```

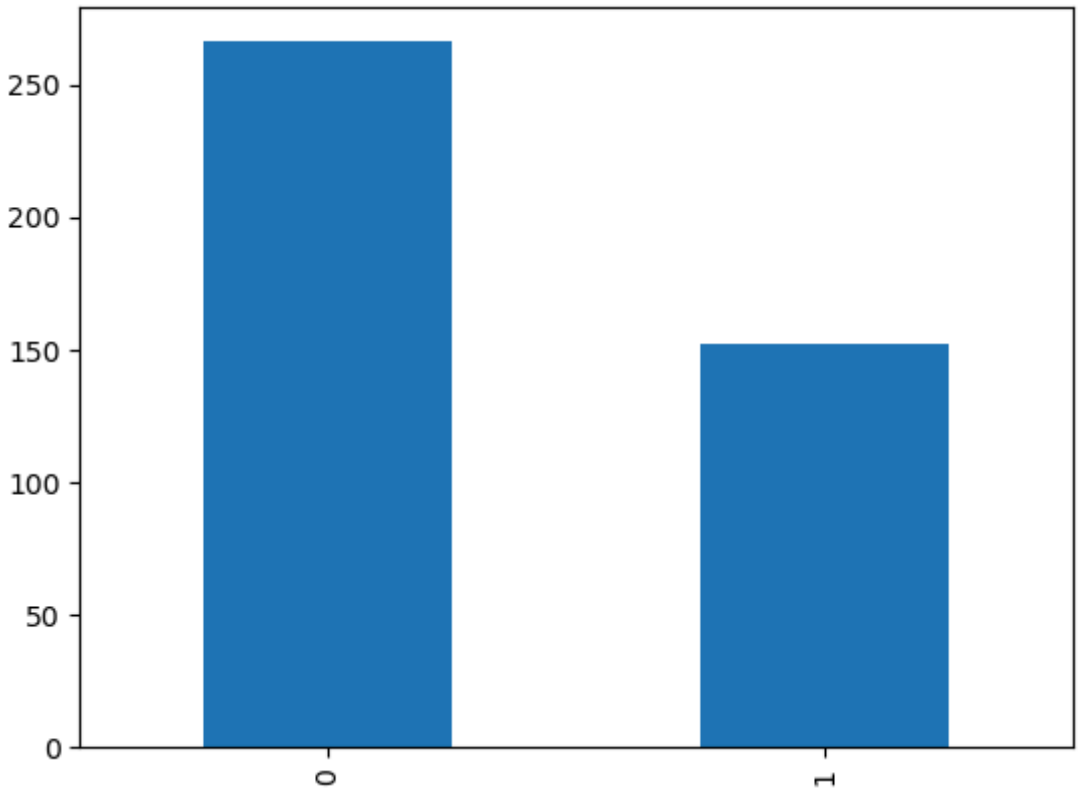


In [10]:
```
df['Survived'].value_counts()   #value count is only applicabel on catogeric
```

Out[10]:
```
0    266
1    152
Name: Survived, dtype: int64
```

In [11]:
```
df['Survived'].value_counts().plot(kind='bar')
```

Out[11]: <Axes: >

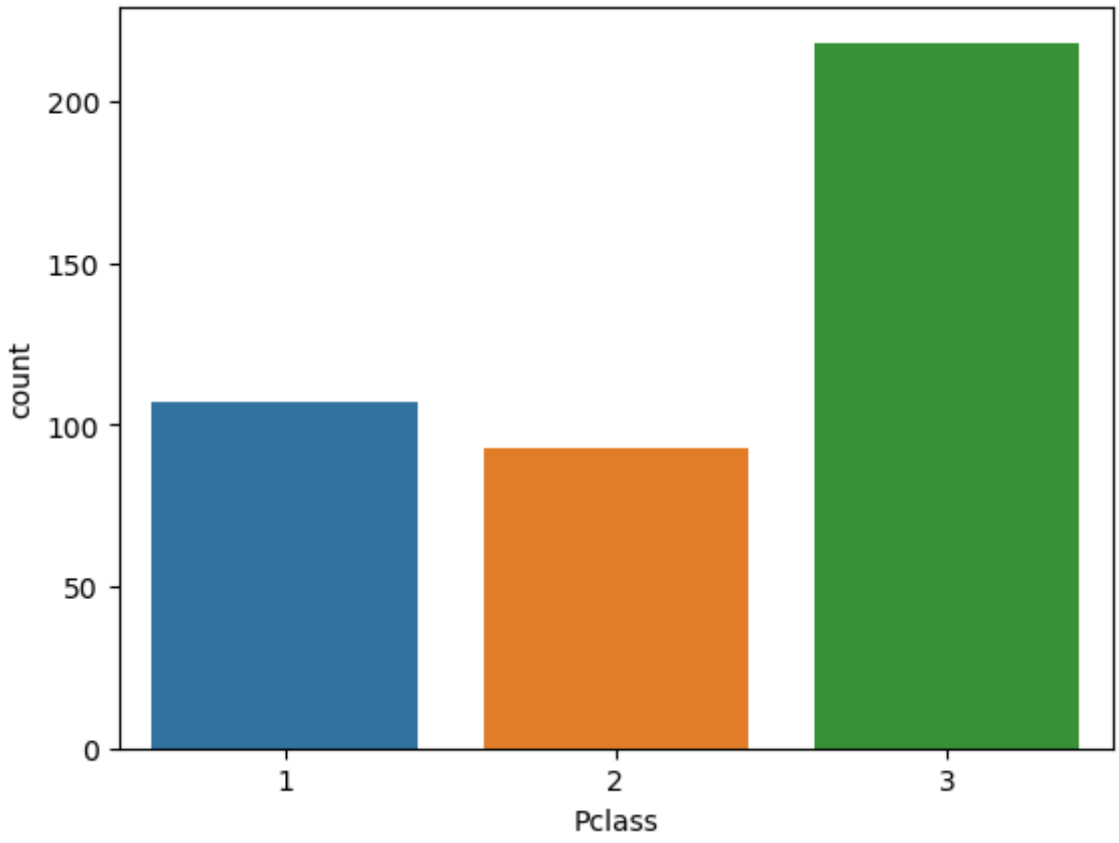


In [12]:
```
df['Pclass'].value_counts()
```

Out[12]:
```
3    218
1    107
2     93
Name: Pclass, dtype: int64
```

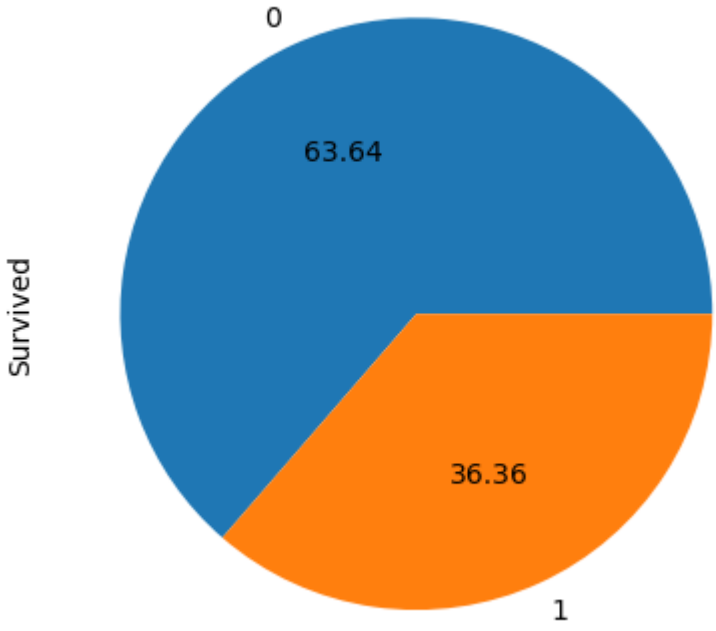
In [14]:
```
sns.countplot(x=df['Pclass'])
```

Out[14]:
```
<Axes: xlabel='Pclass', ylabel='count'>
```



In [19]:
```
df['Survived'].value_counts().plot(kind='pie',autopct='%.2f')   # autopct is
```

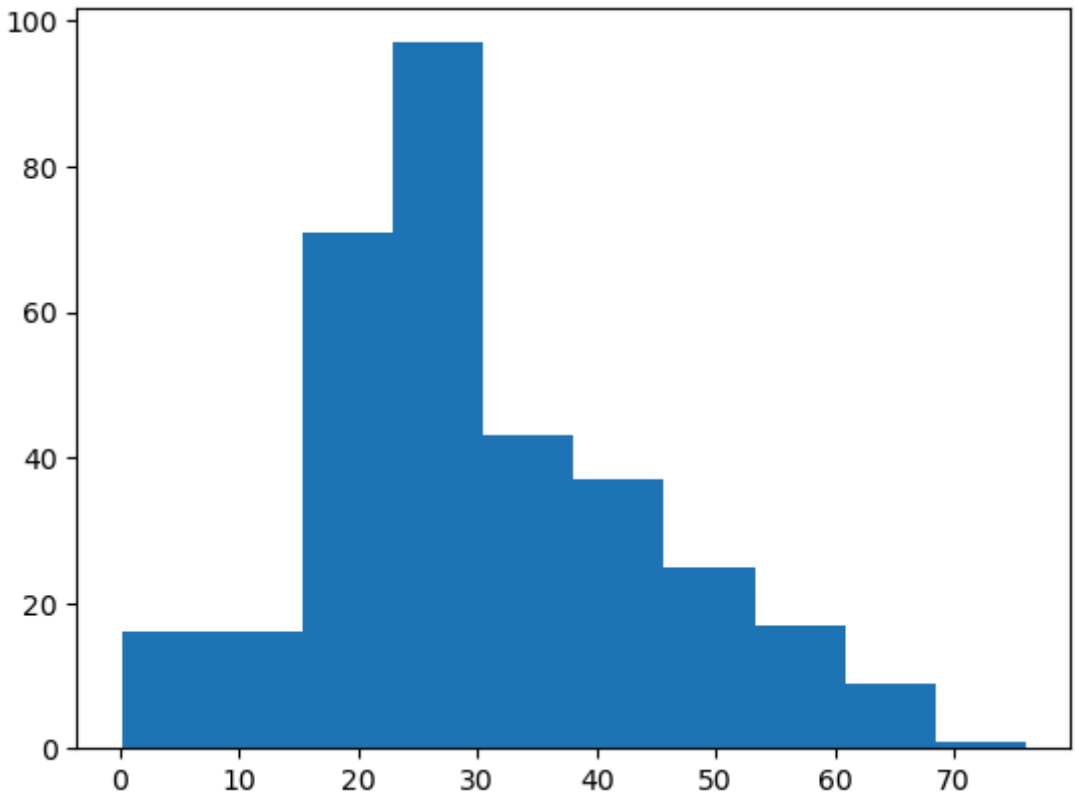
Out[19]:
```
<Axes: ylabel='Survived'>
```



If we have numerical data we then we use histogram because it finds the destribution

In [20]:
```
plt.hist(df['Age'])
```

Out[20]:
```
(array([16., 16., 71., 97., 43., 37., 25., 17.,  9.,  1.]),
 array([ 0.17 ,  7.753, 15.336, 22.919, 30.502, 38.085, 45.668, 53.251,
        60.834, 68.417, 76.   ]),
 <BarContainer object of 10 artists>)
```



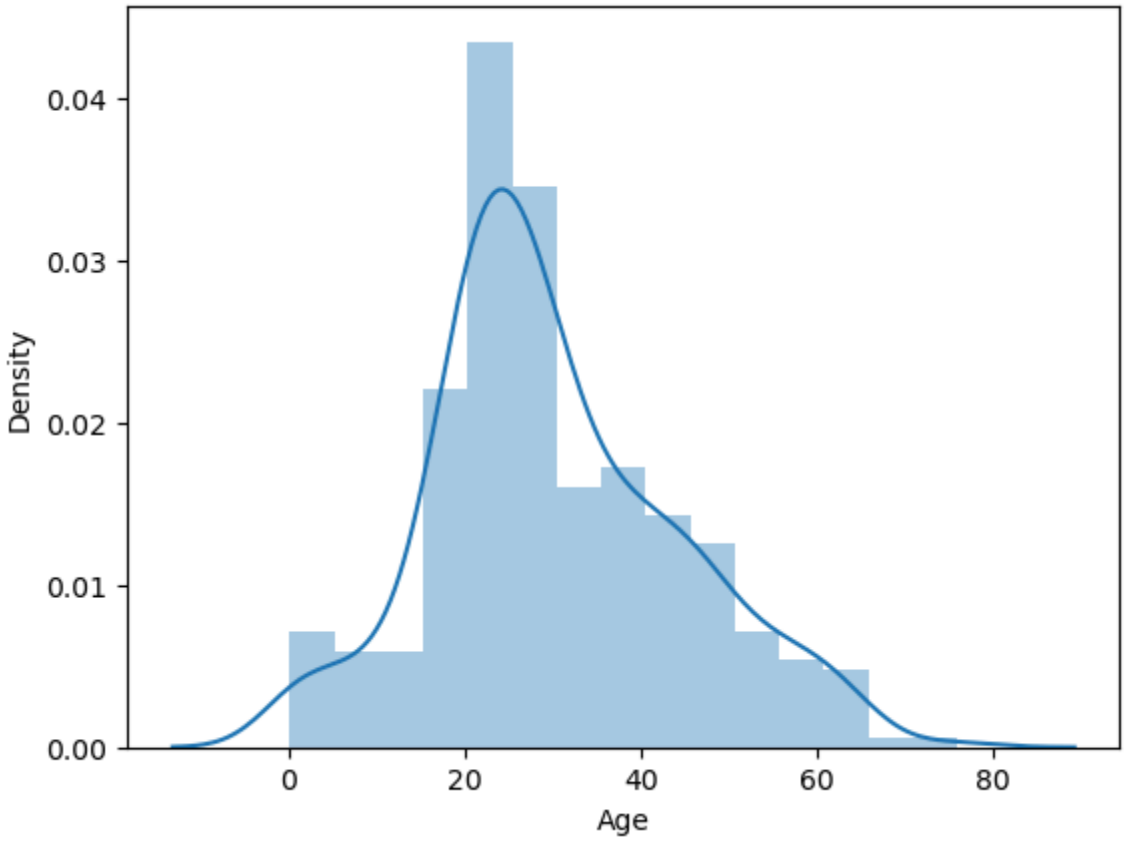
Distplot

curve --> KDE(Kurnel Density Extraction) used to find probability

In [21]:
```
sns.distplot(df['Age']) # to fing the peak value
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_18956\3255828239.py:1: UserWar
ning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['Age'])
```

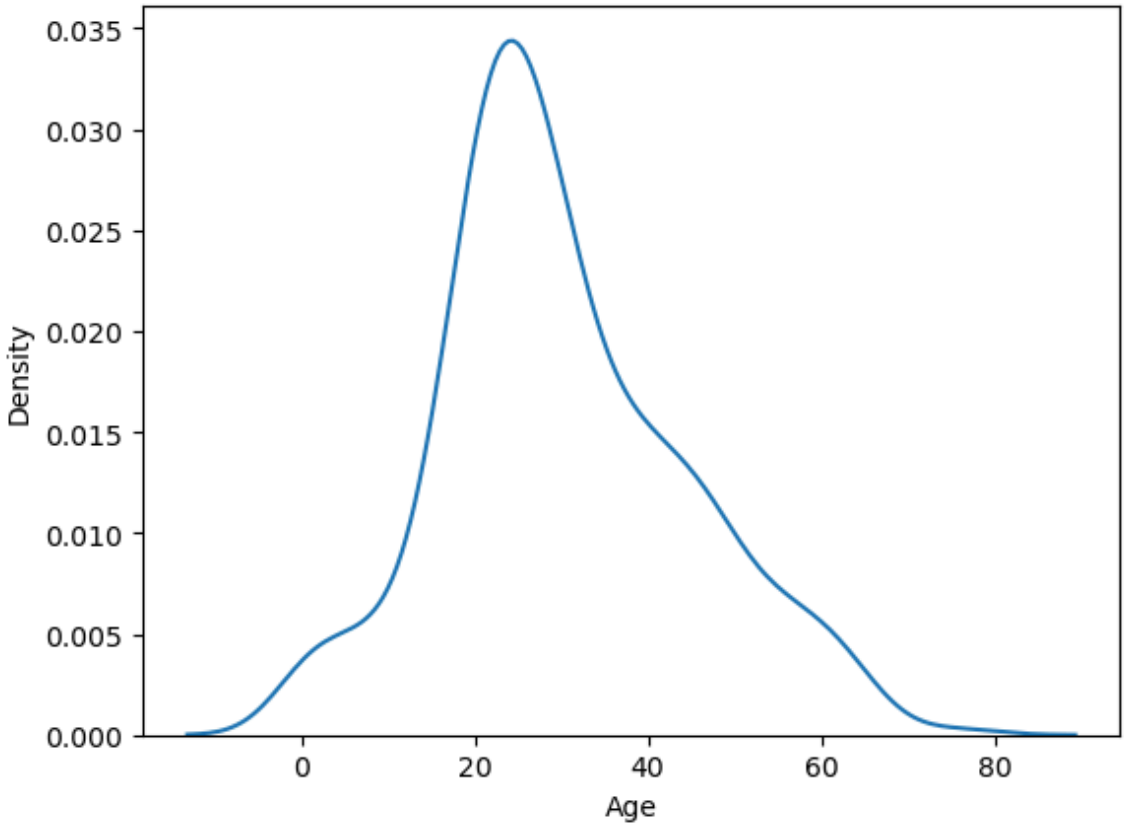
Out[21]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



In [22]:
```
sns.distplot(df['Age'], hist=False)
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_18956\1565444543.py:1: UserWar
ning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `kdeplot` (an axes-level function for kernel densi
ty plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['Age'], hist=False)
```

Out[22]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



BoxPlot

Outliers will be present below the lower fence and upper fence

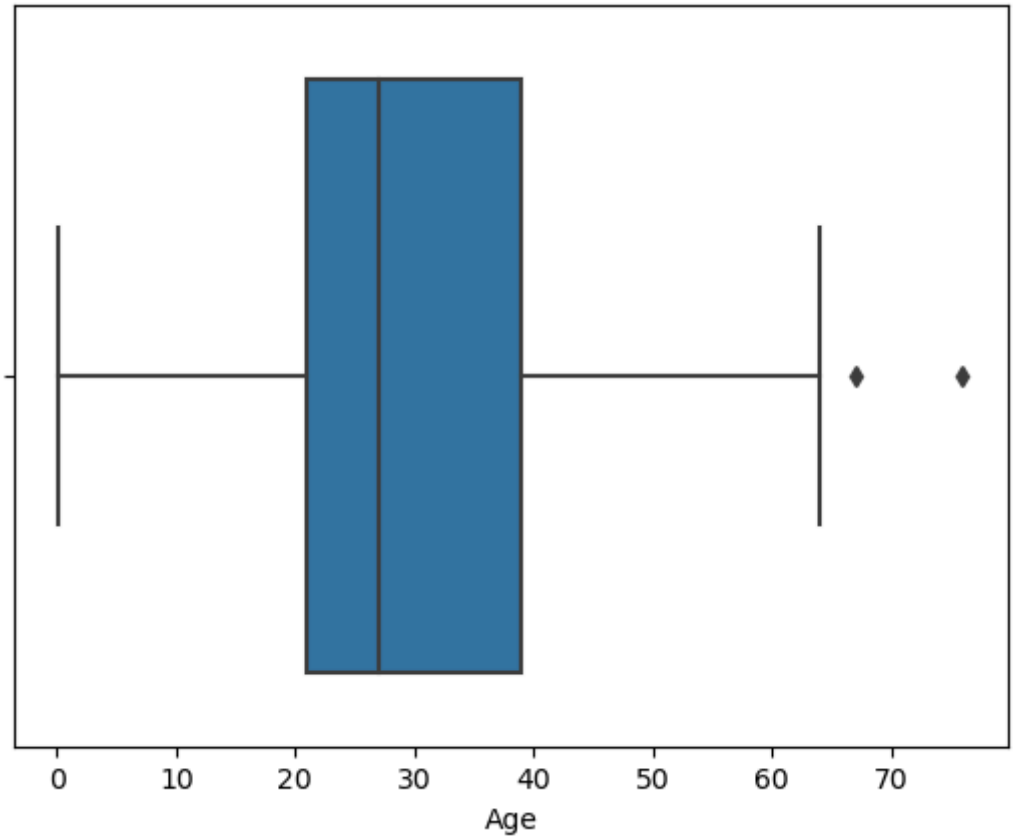
BoxPlot --> it is used to find the outliers

1. lower fence
2. 25% data
3. IOR (Inter Quarantil range)(75%-25%)
4. 75% data

5. upper fence

In [24]:
```
sns.boxplot(x=df['Age'])
```

Out[24]:
```
<Axes: xlabel='Age'>
```



In [26]:
```
sns.boxplot(x=df['Fare'])
```

Out[26]:
```
<Axes: xlabel='Fare'>
```

In [27]:
```
tips=pd.read_csv("F:\\New folder\\ML\\CSV files\\tips.csv")
```

In [28]:
```
tips
```

Out[28]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [29]:
```
tips.head()
```

Out[29]:

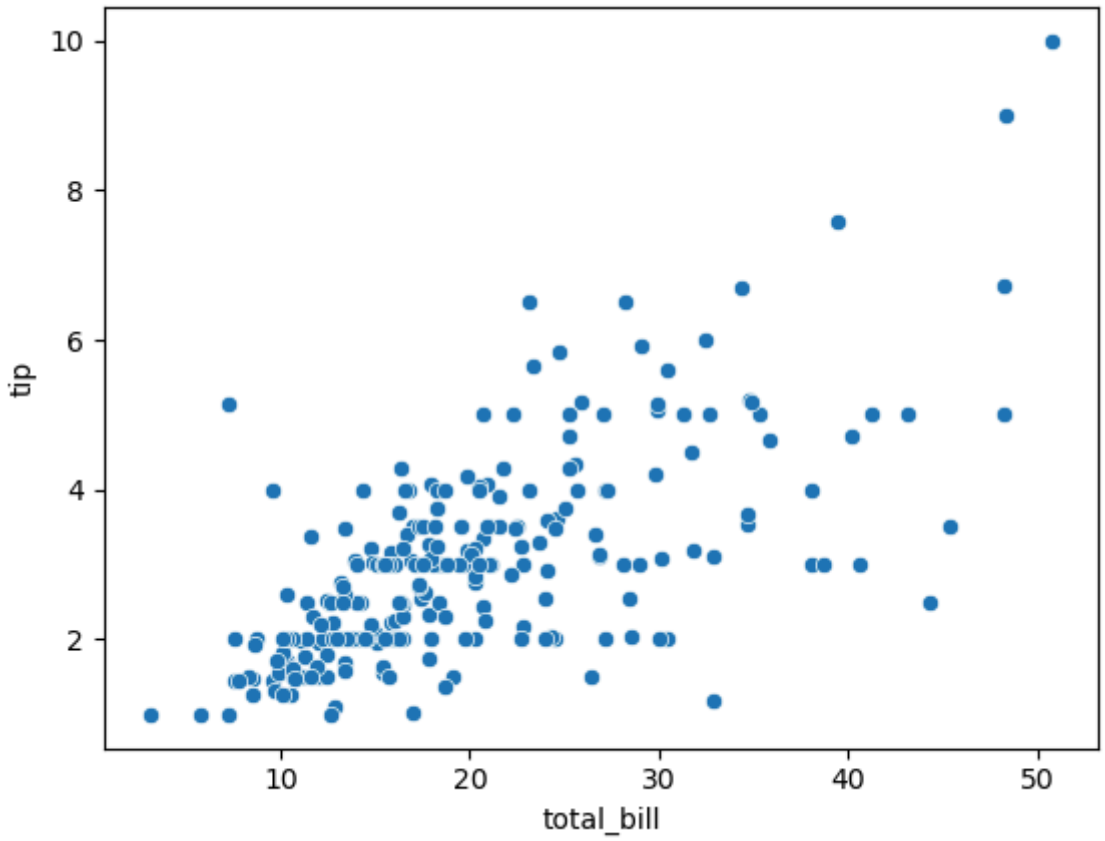
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Bivariate Analysis

1. ScatterPlot(/numerical column - Numerical Column)

In [30]:
```
sns.scatterplot(x=tips['total_bill'],y=tips['tip'])
```

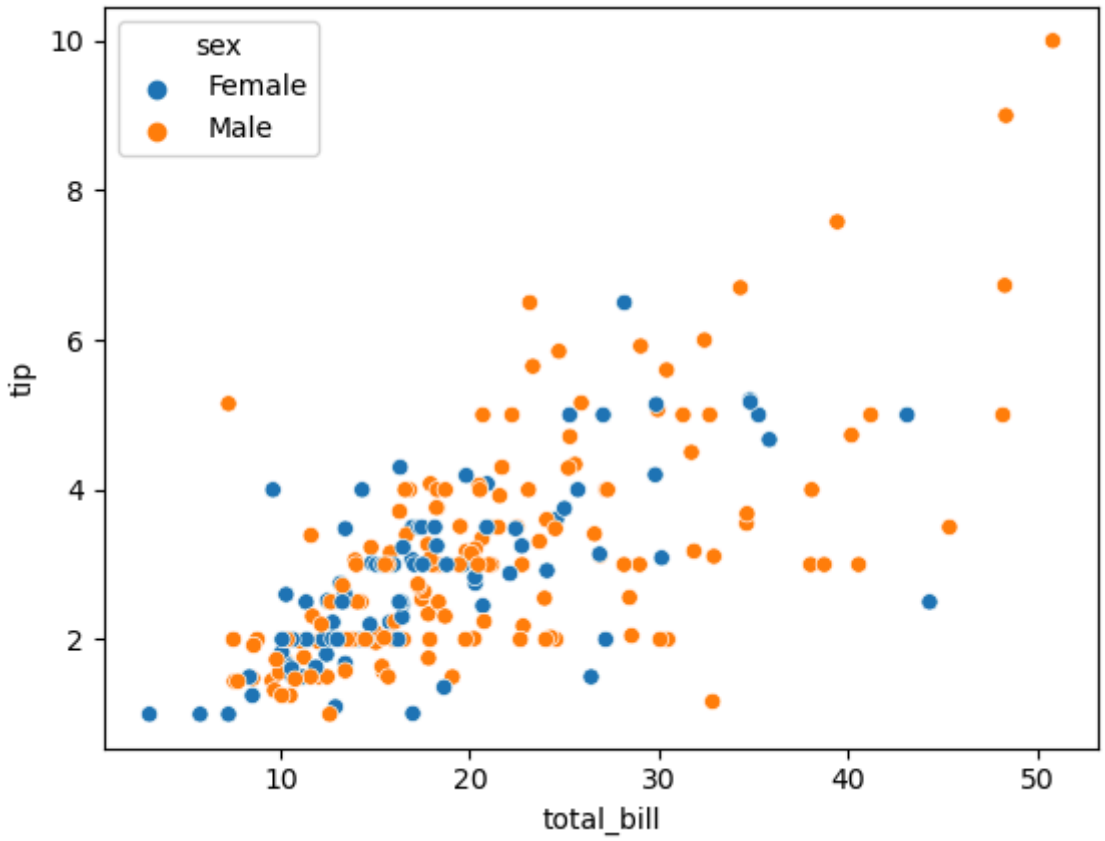
Out[30]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



Hue , style , color , legend are hyper paramater from which we check the relation between columns

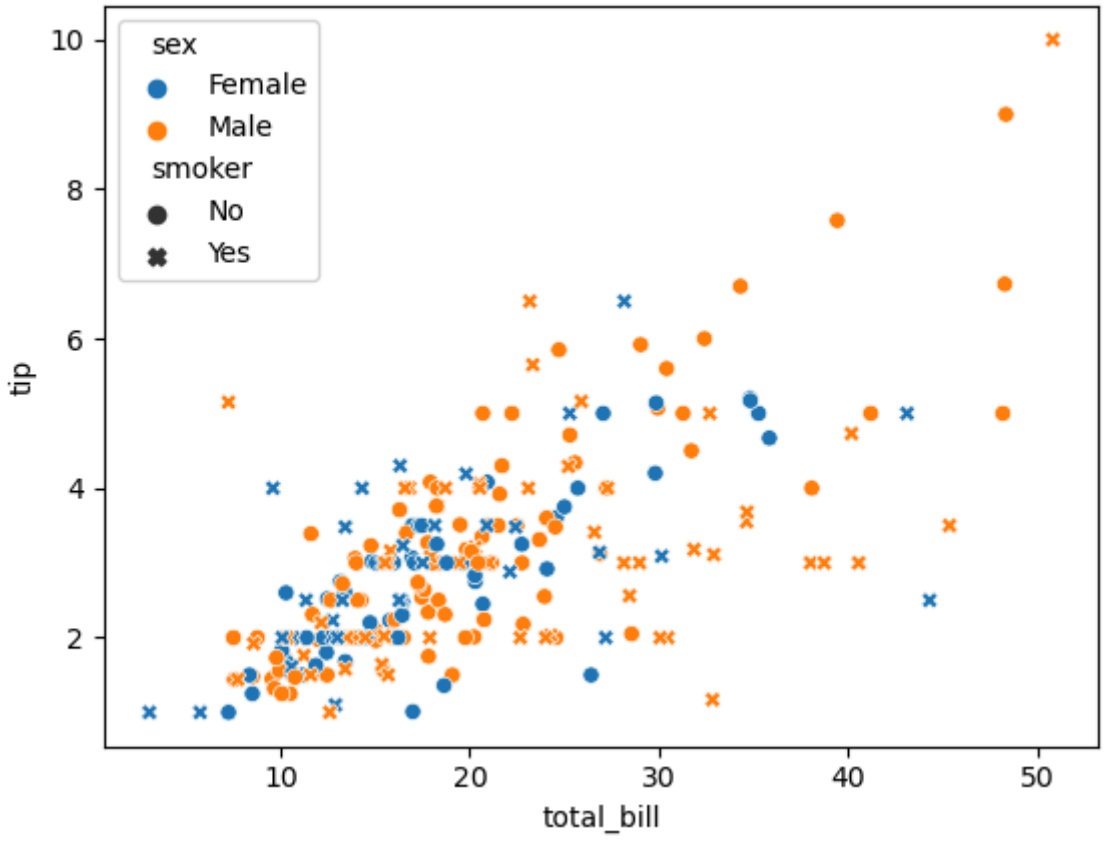
In [34]:
```
sns.scatterplot(x='total_bill',y='tip',data=tips,hue=tips['sex']) # to take
```

Out[34]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



In [35]:
```
sns.scatterplot(x='total_bill',y='tip',data=tips,hue=tips['sex'], style=tips
```

Out[35]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



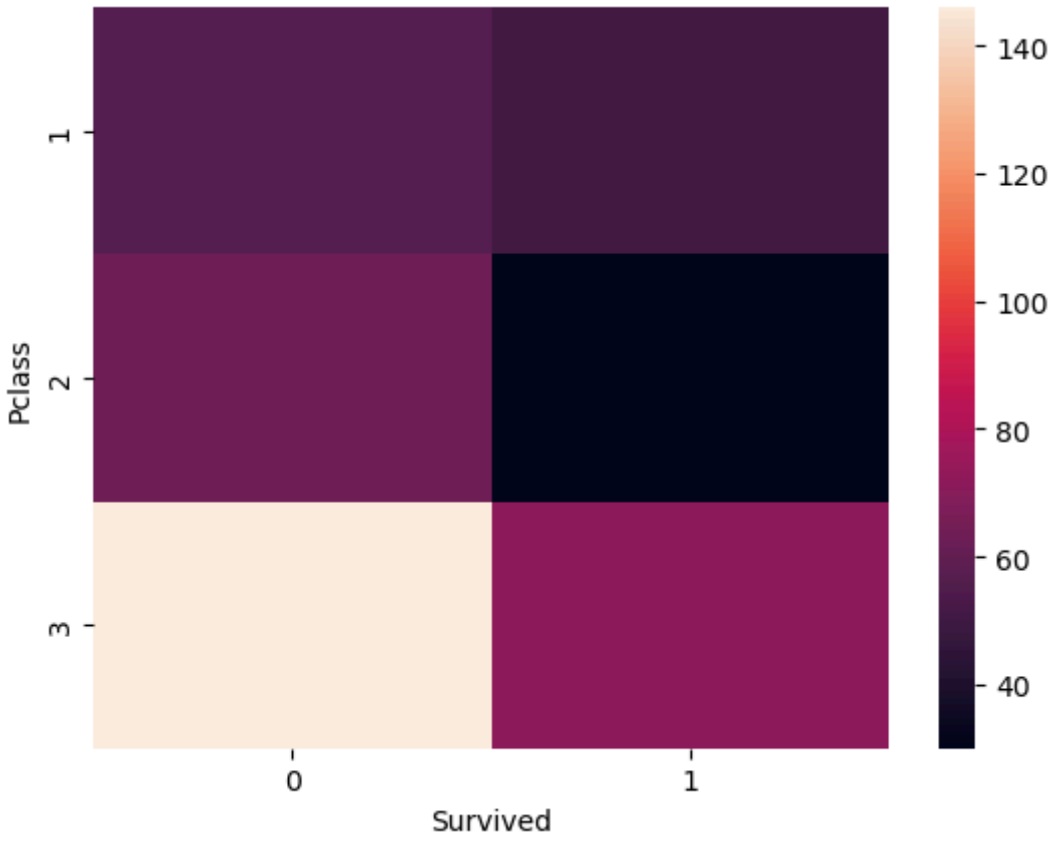
In [39]:
```
a=pd.crosstab(df['Pclass'],df['Survived'])
a
```

Out[39]:

Survived	0	1
Pclass		
1	57	50
2	63	30
3	146	72

In [42]:
```
sns.heatmap(a) # heat map is used to classify dark color denotes less data c
# heatmap is used on catogericaldata
```

Out[42]: <Axes: xlabel='Survived', ylabel='Pclass'>

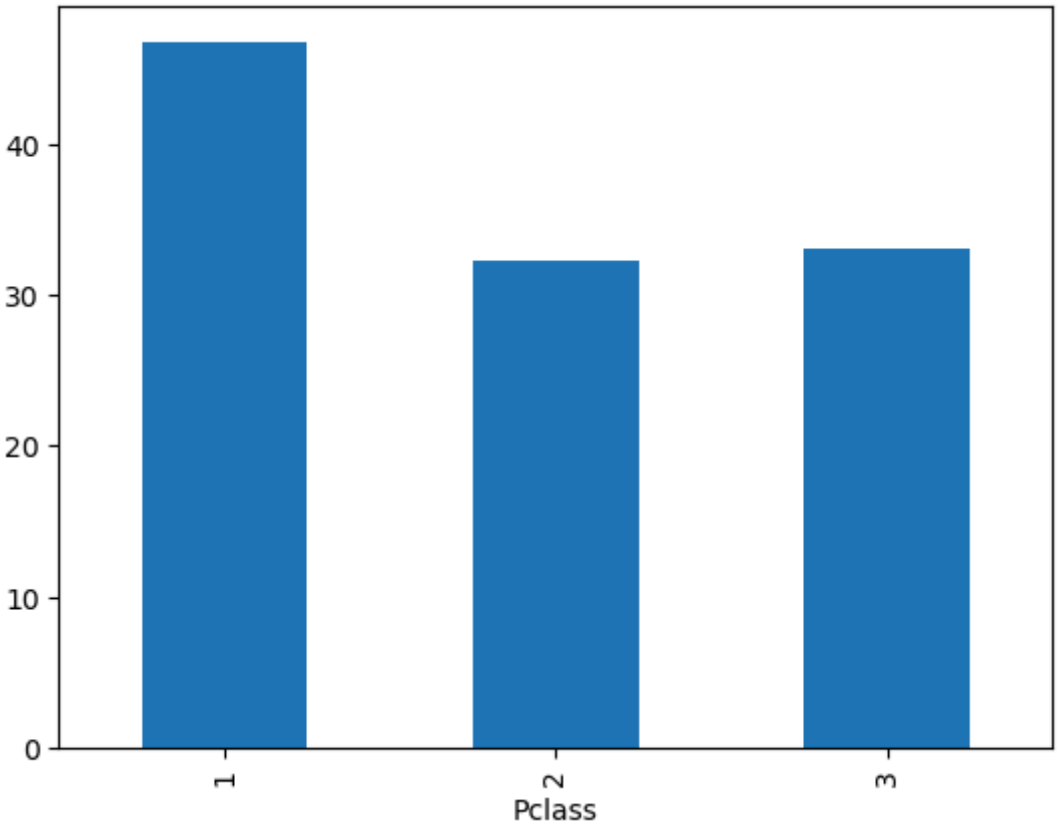


Aggeregation Function

In [44]:
```
((df.groupby('Pclass').mean()['Survived'])*100).plot(kind='bar')
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_18956\3568284408.py:1: FutureW
arning: The default value of numeric_only in DataFrameGroupBy.mean is depr
ecated. In a future version, numeric_only will default to False. Either sp
ecify numeric_only or select only columns which should be valid for the fu
nction.
  ((df.groupby('Pclass').mean()['Survived'])*100).plot(kind='bar')
```

Out[44]:
```
<Axes: xlabel='Pclass'>
```



In [45]:
```
df.groupby('Pclass').mean()
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_18956\3811989730.py:1: FutureW
arning: The default value of numeric_only in DataFrameGroupBy.mean is depr
ecated. In a future version, numeric_only will default to False. Either sp
ecify numeric_only or select only columns which should be valid for the fu
nction.
  df.groupby('Pclass').mean()
```

Out[45]:

Pclass	PassengerId	Survived	Age	SibSp	Parch	Fare
1	1098.224299	0.467290	40.918367	0.476636	0.383178	94.280297
2	1117.935484	0.322581	28.777500	0.376344	0.344086	22.202104
3	1094.178899	0.330275	24.027945	0.463303	0.417431	12.459678

In [ ]: