NumPy Library

```
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
import numpy as np
print(np.__version__)
```

```
2.0.2
```

```
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr[0])
```

```
1
```

```
import numpy as np
arr =np.array([1,2,3,4,5])
print(arr[2] + arr[3])
```

```
7
```

```
import numpy as np
arr = np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(arr)
print('2nd element on 1st row: ', arr[0,1])
print('5th element on 2nd row:',arr[1,4])
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
2nd element on 1st row:  2
5th element on 2nd row: 10
```

Array Slicing

```
import numpy as np
arr = np.array([1,2,3,4,5,6,7])
print(arr[1:5])
print(arr[::-1])
print(arr[4:])
print(arr[:4])
```

```
[2 3 4 5]
[7 6 5 4 3 2 1]
[5 6 7]
[1 2 3 4]
```

```
import numpy as np
arr = np.array([1,2,3,4])
print(arr.dtype)
```

```
int64
```

```
import numpy as np
arr = np.array(['apple','banana','cherry'])
print(arr.dtype)
```

```
<U6
```

DATA TYPES IN NUMPY

i - integer

b - boolean

u - unsigned integer

f - float

c - complex float

m - timedelta

M - datetime

O - object

S - string

U - unicode string

V - fixed chunk of memory for other type(void)

```
#Make a copy, change the original array, and display both arrays

import numpy as np
arr = np.array([1,2,3,4,5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

```
[42  2  3  4  5]
[1 2 3 4 5]
```

```
# Make a view, change the original array, and display both arrays :
import numpy as np
arr = np.array([1,2,3,4,5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

```
[42  2  3  4  5]
[42  2  3  4  5]
```

```
import numpy as np
arr = np.array([[1,2,3,4],[5,6,7,8]])
print(arr.shape)
```

```
(2, 4)
```

```
import numpy as np
arr = np.array([1,2,3,4],ndmin = 5)
print(arr)
print('shape of array:',arr.shape)
```

```
[[[[[1 2 3 4]]]]]
shape of array: (1, 1, 1, 1, 4)
```

```
import numpy as np
a = np.array([(1,2,3),(3,4,4)])
print(a)
print(type(a))
```

```
[[1 2 3]
 [3 4 4]]
<class 'numpy.ndarray'>
```

```
b = np.array([1,2,3])
print(b)
print(b.dtype) #tells us the data types of the elements in an array
print(type(b))
```

```
[1 2 3]
int64
<class 'numpy.ndarray'>
```

```
print(a.ndim) #get dimensions(number dimension)
print(b.ndim)
```

```
2
1
```

```
print(a.shape)
print(b.shape)
```

```
(2, 3)
(3,)
```

```
a = np.array([(1,2,3),(3,4,4)], dtype = 'int16') #if we want to specify the size of it,we can while declaring the size integer
print(a.dtype)
print(a.itemsize)
```

```
int16
2
```

```
a = np.array([(1,2,3),(3,4,4)],dtype = 'int32')
print(a.itemsize) #One element will have how many bytes 1byte = 8 bits
print(a.size) #Total number of elements in the array
print(a.shape) #Will  give Row,Column
```

```
4
6
(2, 3)
```

```
print(a.size * a.itemsize) # another
print(a.nbytes) #total number of bytes in the array or total size
```

```
24
24
```

Accessing / Changing access

```
w = np.arange(start = 1, stop = 10, step = 2) #
print(w)
```

```
[1 3 5 7 9]
```

```
a = np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(a)
print(a.shape)
```

```
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
(2, 7)
```

```
print(a[1,4])
a[0,-1]
```

```
12
np.int64(7)
```

```
print(a[1, : ])
print(a[:,5])
```

```
[ 8  9 10 11 12 13 14]
[ 6 13]
```

```
print(a[0,1:6:2])
```

```
[2 4 6]
```

```
a[1,6] = 20 #directly change the element
print(a)
```

```
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 20]]
```

```
a[:,2] = 5 #All rows, with 3rd column changed to 5
print(a)
```

```
[[ 1  2  5  4  5  6  7]
 [ 8  9  5 11 12 13 20]]
```

```
q = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
#3D : vector of Matrix #2d : Matrix # 1D : Vector
print(q)
```

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

```
print(q[0,1,1]) #1 Matrix, 2row, 2 column
print(q[:,1,:]) # All matrix, 2nd roow, all columns
```

```
4
[[3 4]
 [7 8]]
```

```
q[:,1,:] = [9,9],[9,9] #Replacing All matrix, 2nd row, all column
```

```
q
```

```
array([[[1, 2],
        [9, 9]],

       [[5, 6],
        [9, 9]]])
```

Initializing DIfferent types of arrays

```
a = np.zeros((2,2)) #All 0s matrix, specify shape
print(a)
```

```
[[0. 0.]
 [0. 0.]]
```

```
print(np.zeros((2,2,3)))
```

```
[[[0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]]]
```

```
print(np.ones((4,2,2),dtype = 'int32')) #All 1s matrix
```

```
[[[1 1]
  [1 1]]

 [[1 1]
  [1 1]]

 [[1 1]
  [1 1]]

 [[1 1]
  [1 1]]]
```

```
print(np.full((2,2),99,dtype = 'float32')) #Matrix with 2,2 size and is filled with 99 element with datatye float 32
```

```
[[99. 99.]
 [99. 99.]]
```

```
print(np.full_like(q,7)) #reusing previous matrix (q) and filling it with 7
```

```
[[[7 7]
  [7 7]]

 [[7 7]
  [7 7]]]
```

Creating random numbers

```
print(np.random.rand(2,3)) #create array of size 2,3 with random numbers between [0,1] 0(inclusive) and 1(exclusive)
```

```
[[0.8543366  0.65982771 0.3921741 ]
 [0.71299444 0.61243102 0.60853466]]
```

```
print(np.random.random_sample(q.shape))
```

```
[[[0.86930236 0.25944951]
  [0.09518374 0.50292777]]

 [[0.2793612  0.92274539]
  [0.49009226 0.23578582]]]
```

```
print(np.random.randint(7,size = (3,3))) #randint = integer and generate value between 0 (Include) and 7 (Exclude)
```

```
[[2 4 5]
 [1 2 4]
 [3 4 2]]
```

```
print(np.random.randint(-6,6,size = (2,2))) # between -6(Inclusive) and 6[excluded]
```

```
[[ 5  1]
 [-2  1]]
```

```
print(np.random.randint(-6,6,2)) #2 is the size, i.e, only two values between -6(Inclusive) and 6[excluded]
```

```
[ 3 -1]
```

```
print(np.identity(7,dtype = 'int16')) #To create an identity matrix and 7 is the number of rows
```

```
[[1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 0 1 0 0 0 0]
 [0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0]
 [0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1]]
```

```
arr = np.array([1,2,3])
z = np.repeat(arr,3,axis = 0)
print(z)
```

```
[1 1 1 2 2 2 3 3 3]
```

Copying Arrays

```
a = np.array([1,2,3]) #a is created and then b is also pointing to same object, a copy is not getting created (it's the same ob
b = a
print(b)
b
b[0] = 100
print(b)
print(a)
```

```
[1 2 3]
[100   2   3]
[100   2   3]
```

```
b = a.copy() #Here the contents of a is copied in b
print(a)
b[2] = 90
print(b)
print(a)
```

```
[100   2   3]
[100   2  90]
[100   2   3]
```

Mathematics/BroadCasting

```
a = np.array([1,2,3,4])
print(a)

a += 2
print(a)
print(a+2)
print(a - 2)
print(a * 2)
print(a/2)
```

```
[1 2 3 4]
[3 4 5 6]
[5 6 7 8]
[1 2 3 4]
[ 6  8 10 12]
[1.5 2.  2.5 3. ]
```

```
b = np.array([1,0,1,0])
print(a+b)
print(a ** 2)
```

```
[4 4 6 6]
[ 9 16 25 36]
```

Linear Algebra

```
a = np.ones((2,3))
print(a)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
b = np.full((3,2),2)
print(b)
```

```
[[2 2]
 [2 2]
 [2 2]]
```

```
np.matmul(a,b) #Matrix multiplication : output would be 2 by2 matrix
```

```
array([[6., 6.],
       [6., 6.]])
```

```
c = np.identity(3)
np.linalg.det(c)
```

```
np.float64(1.0)
```

Statistics

```
stats = np.array([[5,2,3],[4,5,6]])
stats
```

```
array([[5, 2, 3],
       [4, 5, 6]])
```

```
print("Maximum :",np.max(stats, axis = 1)) #It returns the new array with max values from each row and axis = 1 means row
```

```
Maximum : [5 6]
```

```
print("Minimum :",np.min(stats, axis = 0)) #It returns the new array with min values from each column and axis = 0 means column
```

```
Minimum : [4 2 3]
```

```
print(np.min(stats))
```

```
2
```

```
print(np.sum(stats))
```

```
25
```

```
sum(sum(stats))
```

```
np.int64(25)
```

Reorganizing Arrays

```
before = np.array([[1,2,3,4],[5,6,7,8]])
print(before)
```

```
[[1 2 3 4]
 [5 6 7 8]]
```

```
after = before.reshape(8,1) #Reshaping array as 8 rows and 1 column
print(after)
```

```
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]]
```

```
#Vertically Staking vectors

v1 = np.array([1,2,3,4]) #vector
v2 = np.array([5,6,7,8]) #vector

np.vstack([v1,v2,v2])  #stacking vectors together as matrices
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [5, 6, 7, 8]])
```

```
#Horizontal Stacking

h1 = np.ones((1,2)) #matrix
h2 = np.zeros((1,2)) #matrix
print(h1)
print(h2)

np.hstack((h1,h2))
```

```
[[1. 1.]]
[[0. 0.]]
array([[1., 1., 0., 0.]])
```

PANDAS

- Pandas are more flexible and can work on big data '
- They are open source built on top of numPy lib
- It is

```
from google.colab import files
uploaded = files.upload()   #1st way to load the data in pandas #2nd way via folder
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import pandas as pd
df = pd.read_csv('pokemon_data.csv')
df
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

800 rows × 12 columns

```
print(df.columns) #reading headers (column names)
```

```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

```
print(df.head(3)) #reading starting 3 rows(By default : it gives 5 rows of starting)
```

```
   #       Name Type 1  Type 2  HP  Attack  Defense  Sp. Atk  Sp. Def  Speed  \
0  1  Bulbasaur  Grass  Poison  45      49       49       65       65     45
1  2    Ivysaur  Grass  Poison  60      62       63       80       80     60
2  3   Venusaur  Grass  Poison  80      82       83      100      100     80

   Generation  Legendary
0           1      False
1           1      False
2           1      False
```

```
print(df["Name"])
```

```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                  Charmander
              ...
795                  Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object
```

```
print(df["Name"][0:10]) #First 10 rows of the specific column using slicing
```

```
0                    Bulbasaur
1                      Ivysaur
2                     Venusaur
3        VenusaurMega Venusaur
4                    Charmander
5                    Charmeleon
6                     Charizard
7    CharizardMega Charizard X
8    CharizardMega Charizard Y
9                     Squirtle
Name: Name, dtype: object
```

```
df.Name  #Simply the name of the column
```

	Name
0	Bulbasaur
1	Ivysaur
2	Venusaur
3	VenusaurMega Venusaur
4	Charmander
...	...
795	Diancie
796	DiancieMega Diancie
797	HoopaHoopa Confined
798	HoopaHoopa Unbound
799	Volcanion

800 rows × 1 columns

dtype: object

```
print(df[["Name","Type 1","HP"]])
```

```
                     Name   Type 1  HP
0               Bulbasaur    Grass  45
1                 Ivysaur    Grass  60
2                Venusaur    Grass  80
3    VenusaurMega Venusaur    Grass  80
4              Charmander     Fire  39
..                    ...      ...  ..
795               Diancie     Rock  50
796    DiancieMega Diancie     Rock  50
797    HoopaHoopa Confined  Psychic  80
798     HoopaHoopa Unbound  Psychic  80
799              Volcanion     Fire  80

[800 rows x 3 columns]
```

```
print(df.iloc[1])
```

```
#                   2
Name          Ivysaur
Type 1          Grass
Type 2         Poison
HP                 60
Attack             62
Defense            63
Sp. Atk            80
Sp. Def            80
Speed              60
Generation          1
Legendary       False
Name: 1, dtype: object
```

```
print(df.iloc[2,1])
```

```
Venusaur
```

```
from google.colab import drive #3rd via uploading file into drive and then mounting the drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
file_path = '/content/drive/My Drive/AIML Lab/pokemon_data.csv'
```

```
import pandas as pd
df = pd.read_csv(file_path) #dataframe is the reference to the file
```

```
df
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

800 rows × 12 columns

Reading Data in Pandas

```
#reading headers
print(df.columns)
```

```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

```
print(df.head) #default
```

```
<bound method NDFrame.head of        #                    Name   Type 1  Type 2  HP  Attack  Defense  \
0      1               Bulbasaur    Grass  Poison  45      49       49
1      2                 Ivysaur    Grass  Poison  60      62       63
2      3                Venusaur    Grass  Poison  80      82       83
3      3  VenusaurMega Venusaur    Grass  Poison  80     100      123
4      4              Charmander     Fire     NaN  39      52       43
..   ...                     ...      ...     ...  ..     ...      ...
795  719                 Diancie     Rock   Fairy  50     100      150
796  719    DiancieMega Diancie     Rock   Fairy  50     160      110
797  720    HoopaHoopa Confined  Psychic   Ghost  80     110       60
798  720     HoopaHoopa Unbound  Psychic    Dark  80     160       60
799  721              Volcanion     Fire   Water  80     110      120

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
0         65       65     45           1      False
1         80       80     60           1      False
2        100      100     80           1      False
3        122      120     80           1      False
4         60       50     65           1      False
..       ...      ...    ...         ...        ...
795      100      150     50           6       True
796      160      110    110           6       True
797      150      130     70           6       True
798      170      130     80           6       True
799      130       90     70           6       True

[800 rows x 12 columns]>
```

```
print(df.head(3)) #1st three rows of each column
```

```
   #       Name Type 1  Type 2  HP  Attack  Defense  Sp. Atk  Sp. Def  Speed  \
0  1  Bulbasaur  Grass  Poison  45      49       49       65       65     45
1  2    Ivysaur  Grass  Poison  60      62       63       80       80     60
2  3   Venusaur  Grass  Poison  80      82       83      100      100     80

   Generation  Legendary
0           1      False
1           1      False
```

```
2          1      False
```

```
print(df['Name'])
```

```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                 Charmander
               ...
795                  Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object
```

```
print(df['Name'][0:10]) #1st 10 names
```

```
0                    Bulbasaur
1                      Ivysaur
2                     Venusaur
3         VenusaurMega Venusaur
4                    Charmander
5                    Charmeleon
6                     Charizard
7    CharizardMega Charizard X
8    CharizardMega Charizard Y
9                     Squirtle
Name: Name, dtype: object
```

```
df.Name  #simply the name of the column
```

	Name
0	Bulbasaur
1	Ivysaur
2	Venusaur
3	VenusaurMega Venusaur
4	Charmander
...	...
795	Diancie
796	DiancieMega Diancie
797	HoopaHoopa Confined
798	HoopaHoopa Unbound
799	Volcanion

800 rows × 1 columns

dtype: object

```
print(df[["Name", "Type 1", "HP"]]) #print 3 column details
```

```
                     Name   Type 1  HP
0                Bulbasaur    Grass  45
1                  Ivysaur    Grass  60
2                 Venusaur    Grass  80
3    VenusaurMega Venusaur    Grass  80
4                Charmander     Fire  39
..                      ...      ...  ..
795                Diancie     Rock  50
796    DiancieMega Diancie     Rock  50
797    HoopaHoopa Confined  Psychic  80
798     HoopaHoopa Unbound  Psychic  80
799              Volcanion     Fire  80

[800 rows x 3 columns]
```

```
print(df.iloc[1])  #Read each row
```

```
#                  2
Name           Ivysaur
Type 1           Grass
Type 2          Poison
HP                  60
Attack              62
Defense             63
Sp. Atk             80
Sp. Def             80
Speed               60
Generation           1
Legendary        False
Name: 1, dtype: object
```

```
print(df.shape) #Total number of (rows, column)
```

```
(800, 12)
```

```
print(df.info)
```

```
<bound method DataFrame.info of        #                   Name   Type 1  Type 2  HP  Attack  Defense  \
0      1              Bulbasaur    Grass  Poison  45      49       49
1      2                Ivysaur    Grass  Poison  60      62       63
2      3               Venusaur    Grass  Poison  80      82       83
3      3  VenusaurMega Venusaur    Grass  Poison  80     100      123
4      4             Charmander     Fire     NaN  39      52       43
..   ...                    ...      ...     ...  ..     ...      ...
795  719                Diancie     Rock   Fairy  50     100      150
796  719    DiancieMega Diancie     Rock   Fairy  50     160      110
797  720    HoopaHoopa Confined  Psychic   Ghost  80     110       60
798  720     HoopaHoopa Unbound  Psychic    Dark  80     160       60
799  721              Volcanion     Fire   Water  80     110      120

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
0         65       65     45           1      False
1         80       80     60           1      False
2        100      100     80           1      False
3        122      120     80           1      False
4         60       50     65           1      False
..       ...      ...    ...         ...        ...
795      100      150     50           6       True
796      160      110    110           6       True
797      150      130     70           6       True
798      170      130     80           6       True
799      130       90     70           6       True

[800 rows x 12 columns]>
```

```
print(df.describe())  #This provides all the stats like mean, standard deviation etc
```

```
                #          HP      Attack     Defense     Sp. Atk     Sp. Def  \
count  800.000000  800.000000  800.000000  800.000000  800.000000  800.000000
mean   362.813750   69.258750   79.001250   73.842500   72.820000   71.902500
std    208.343798   25.534669   32.457366   31.183501   32.722294   27.828916
min      1.000000    1.000000    5.000000    5.000000   10.000000   20.000000
25%    184.750000   50.000000   55.000000   50.000000   49.750000   50.000000
50%    364.500000   65.000000   75.000000   70.000000   65.000000   70.000000
75%    539.250000   80.000000  100.000000   90.000000   95.000000   90.000000
max    721.000000  255.000000  190.000000  230.000000  194.000000  230.000000

            Speed  Generation
count  800.000000   800.00000
mean    68.277500     3.32375
std     29.060474     1.66129
min      5.000000     1.00000
25%     45.000000     2.00000
50%     65.000000     3.00000
75%     90.000000     5.00000
max    180.000000     6.00000
```

```
df.sort_values("Name")
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
510	460	Abomasnow	Grass	Ice	90	92	75	92	85	60	4	False
511	460	AbomasnowMega Abomasnow	Grass	Ice	90	132	105	132	105	30	4	False
68	63	Abra	Psychic	NaN	25	20	15	105	55	90	1	False
392	359	Absol	Dark	NaN	65	130	60	75	60	75	3	False
393	359	AbsolMega Absol	Dark	NaN	65	150	60	115	60	115	3	False
...	...	...	...	...	...	...	...	...	...	...	...	...
632	571	Zoroark	Dark	NaN	60	105	60	120	60	105	5	False
631	570	Zorua	Dark	NaN	40	65	40	80	40	65	5	False
46	41	Zubat	Poison	Flying	40	45	35	30	40	55	1	False
695	634	Zweilous	Dark	Dragon	72	85	70	65	70	58	5	False
794	718	Zygarde50% Forme	Dragon	Ground	108	100	121	81	95	95	6	True

800 rows × 12 columns

```
df.sort_values("Name",ascending = False)
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
794	718	Zygarde50% Forme	Dragon	Ground	108	100	121	81	95	95	6	True
695	634	Zweilous	Dark	Dragon	72	85	70	65	70	58	5	False
46	41	Zubat	Poison	Flying	40	45	35	30	40	55	1	False
631	570	Zorua	Dark	NaN	40	65	40	80	40	65	5	False
632	571	Zoroark	Dark	NaN	60	105	60	120	60	105	5	False
...	...	...	...	...	...	...	...	...	...	...	...	...
393	359	AbsolMega Absol	Dark	NaN	65	150	60	115	60	115	3	False
392	359	Absol	Dark	NaN	65	130	60	75	60	75	3	False
68	63	Abra	Psychic	NaN	25	20	15	105	55	90	1	False
511	460	AbomasnowMega Abomasnow	Grass	Ice	90	132	105	132	105	30	4	False
510	460	Abomasnow	Grass	Ice	90	92	75	92	85	60	4	False

800 rows × 12 columns

```
print(df.sort_values(["Type 1","HP"],ascending = False))
```

```
       #       Name Type 1  Type 2   HP  Attack  Defense  Sp. Atk  Sp. Def  \
351  321    Wailord  Water     NaN  170      90       45       90       45
655  594  Alomomola  Water     NaN  165      75       80       40       45
142  131     Lapras  Water     Ice  130      85       80       85       95
145  134   Vaporeon  Water     NaN  130      65       60      110       95
350  320    Wailmer  Water     NaN  130      70       35       70       35
..   ...        ...    ...     ...  ...     ...      ...      ...      ...
314  290    Nincada    Bug  Ground   31      45       90       30       30
462  415     Combee    Bug  Flying   30      30       42       30       42
603  543   Venipede    Bug  Poison   30      45       59       30       39
230  213    Shuckle    Bug    Rock   20      10      230       10      230
316  292   Shedinja    Bug   Ghost    1      90       45       30       30

     Speed  Generation  Legendary
351     60           3      False
655     65           5      False
142     60           1      False
145     65           1      False
350     60           3      False
..     ...         ...        ...
314     40           3      False
```

```
462     70           4      False
603     57           5      False
230      5           2      False
316     40           3      False

[800 rows x 12 columns]
```

```
print(df.sort_values(["Type 1","HP"],ascending = [1,0]))
```

```
       #                   Name Type 1    Type 2  HP  Attack  Defense  \
520  469                Yanmega    Bug    Flying  86      76       86
698  637               Volcarona    Bug      Fire  85      60       65
231  214               Heracross    Bug  Fighting  80     125       75
232  214  HeracrossMega Heracross    Bug  Fighting  80     185      115
678  617                Accelgor    Bug       NaN  80      70       40
..   ...                     ...    ...       ...  ..     ...      ...
106   98                 Krabby  Water       NaN  30     105       90
125  116                 Horsea  Water       NaN  30      40       70
129  120                 Staryu  Water       NaN  30      45       55
139  129               Magikarp  Water       NaN  20      10       55
381  349                 Feebas  Water       NaN  20      15       20

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
520      116       56     95           4      False
698      135      105    100           5      False
231       40       95     85           2      False
232       40      105     75           2      False
678      100       60    145           5      False
..       ...      ...    ...         ...        ...
106       25       25     50           1      False
125       70       25     60           1      False
129       70       55     85           1      False
139       15       20     80           1      False
381       10       55     80           3      False

[800 rows x 12 columns]
```

Practical 3 EDA (EXPLORATORY DATA ANALYSIS) : To understand the data

```
from google.colab import drive #EDA : Ex
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
file_path = '/content/drive/My Drive/AIML Lab/train.csv'
```

```
import pandas as pd
df = pd.read_csv(file_path)
```

Data Analysis means asking yourself the questions

```
#1. What is the size of data
df.shape
```

```
(891, 12)
```

```
#2. How does the data looks like ?
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath	female	35.0	1	0	113803	53.1000	C123	S

```
#3. What is the datatype of the columns ?
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
#4.Are there any missing values?
df.isnull()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

```
df.isnull().sum() #each attributes have this many NULL values
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2

dtype: int64

```
#5.Is there any duplicate values ?

df.duplicated().sum()
```

```
np.int64(0)
```

```
#6. Is there any corelation ?
df.corr(numeric_only = True) #Any attribute is connected with (or related to) another attribute
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

```
df.corr(numeric_only = True)['Survived']
```

	Survived
PassengerId	-0.005007
Survived	1.000000
Pclass	-0.338481
Age	-0.077221
SibSp	-0.035322
Parch	0.081629
Fare	0.257307

dtype: float64

```
df.corr(numeric_only = True)['Pclass']
```

	Pclass
PassengerId	-0.035144
Survived	-0.338481
Pclass	1.000000
Age	-0.369226
SibSp	0.083081
Parch	0.018443
Fare	-0.549500

dtype: float64

```
df.corr(numeric_only = True)["Age"]
```

	Age
PassengerId	0.036847
Survived	-0.077221
Pclass	-0.369226
Age	1.000000
SibSp	-0.308247
Parch	-0.189119
Fare	0.096067

dtype: float64

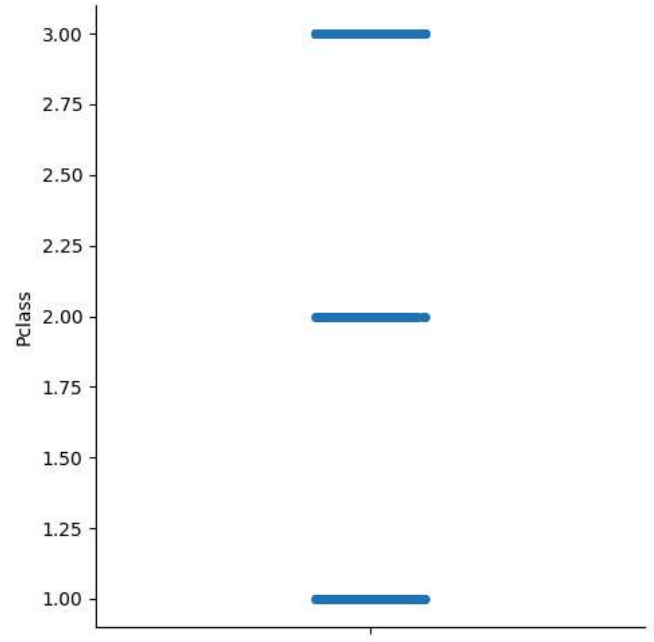
SEABORN MODULE

It has inbuilt dataset

```
import seaborn as sns
```

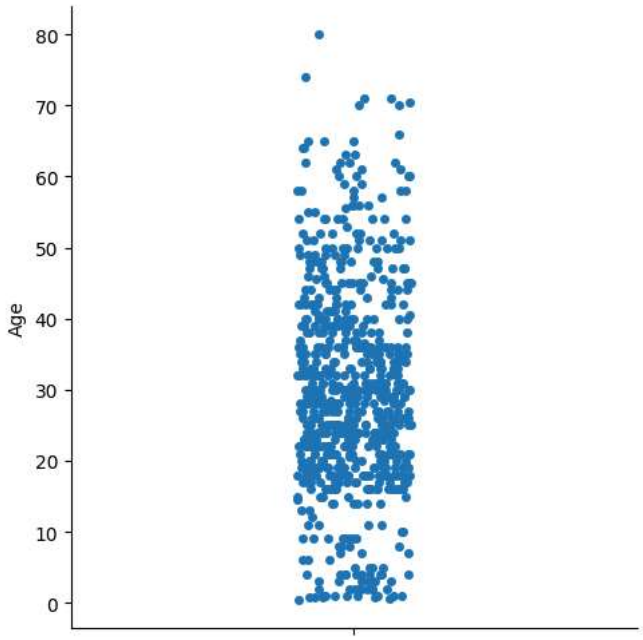
```
sns.catplot(df["Pclass"])
```

```
<seaborn.axisgrid.FacetGrid at 0x780f3c5cd760>
```



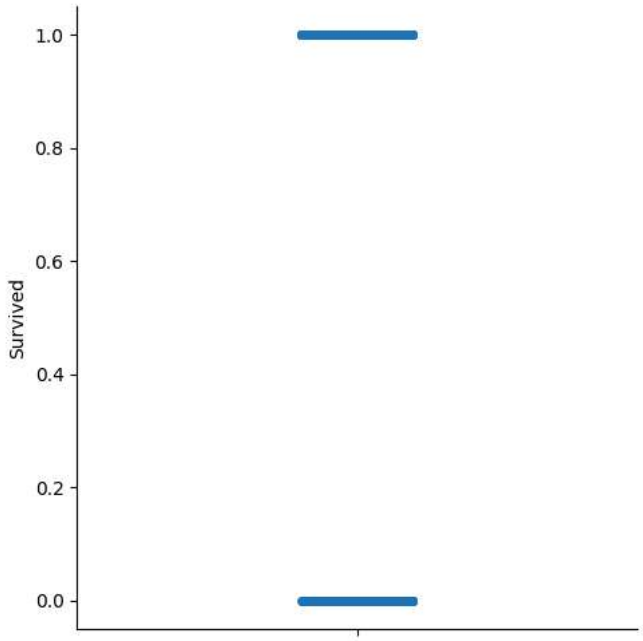
```
sns.catplot(df["Age"])
```

```
<seaborn.axisgrid.FacetGrid at 0x780f3e836870>
```



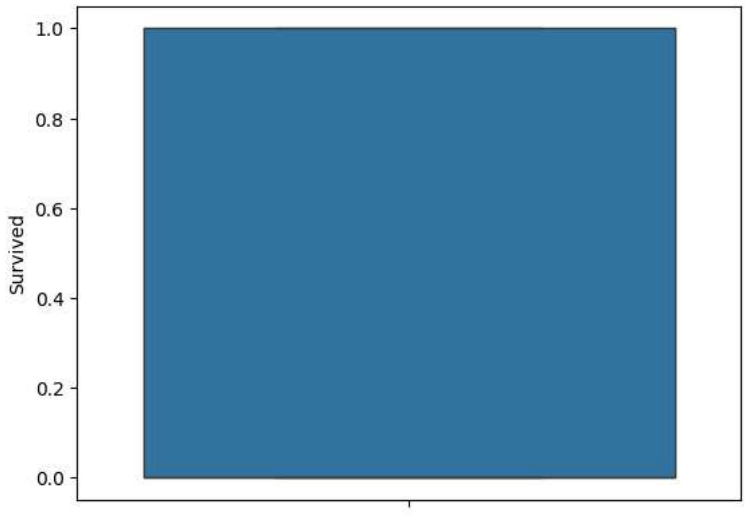

```
sns.catplot(df["Survived"])
```

```
<seaborn.axisgrid.FacetGrid at 0x780f5aabd7f0>
```



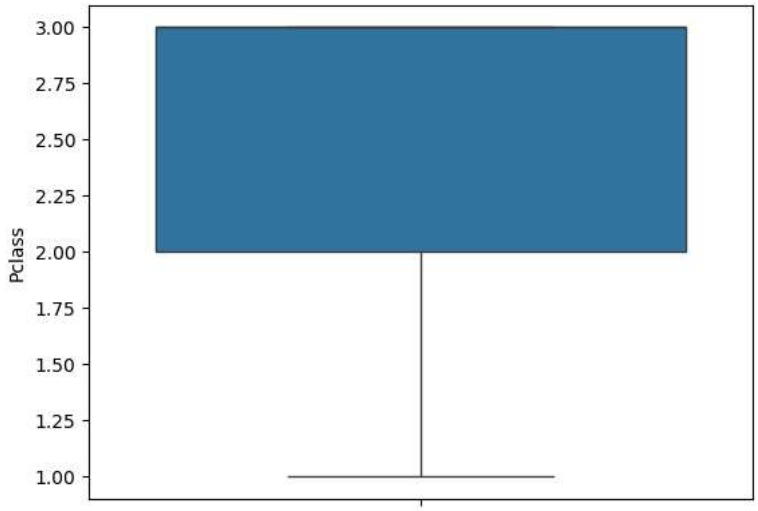

```
sns.boxplot(df["Survived"])
```

```
<Axes: ylabel='Survived'>
```



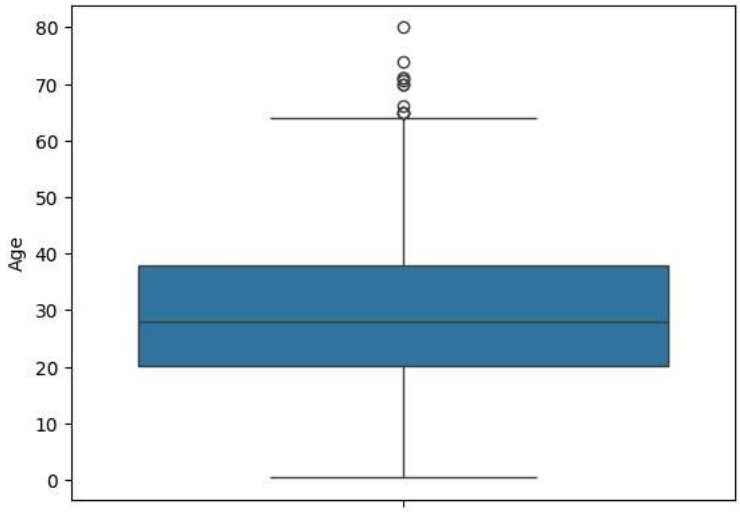

```
sns.boxplot(df["Pclass"])
```

```
<Axes: ylabel='Pclass'>
```



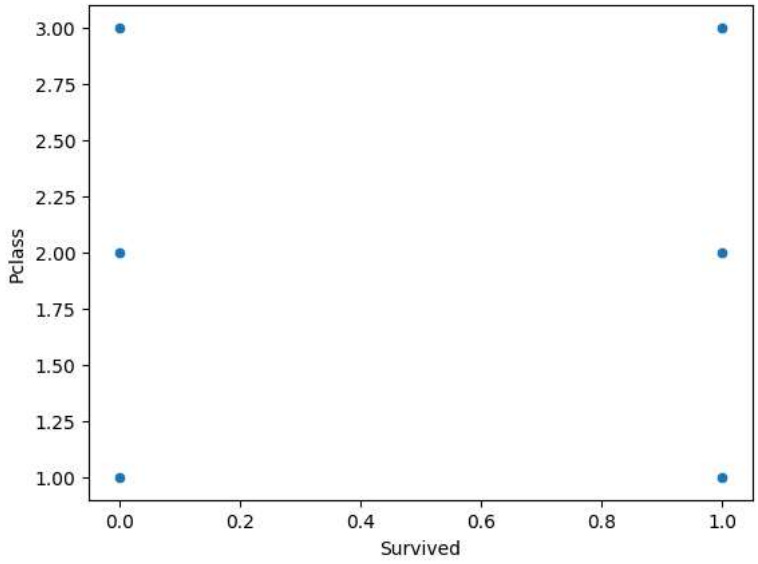

```
sns.boxplot(df['Age'])
```

```
<Axes: ylabel='Age'>
```




```
sns.scatterplot(x="Survived",y="Pclass",data = df)
```

```
<Axes: xlabel='Survived', ylabel='Pclass'>
```



One-variate : A variable is realted to another one variable

It shows relationship between 2 variables

MultiVariate EDA

Multi-variate Data Analyis : As one variable may be related to more than one variable or attribute

It shows relationship between 2 or more variables

```
import seaborn as sns #because seaborn has multiple visualisation and graphs like plots, etc ,matplot lib has all plots, graphs
```

```
df = sns.load_dataset('tips') #df = dataframe referring to tips dataset  and tips dataset is already available in seaborn libra
```

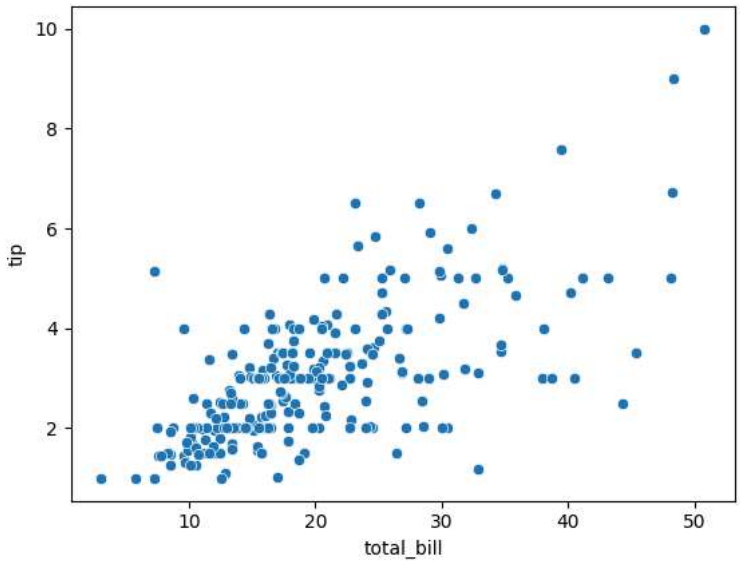
```
titanic = pd.read_csv(file_path)
```

```
df.head()
```

	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

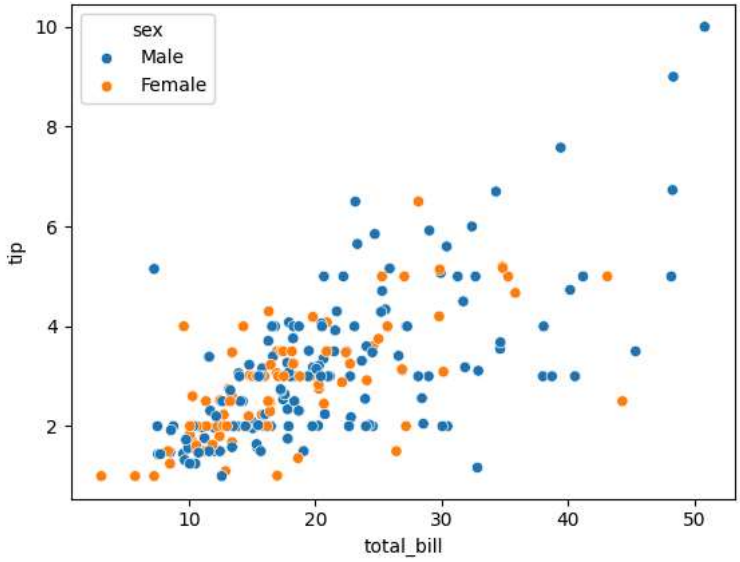
```
sns.scatterplot(x = 'total_bill', y = 'tip', data = df)
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



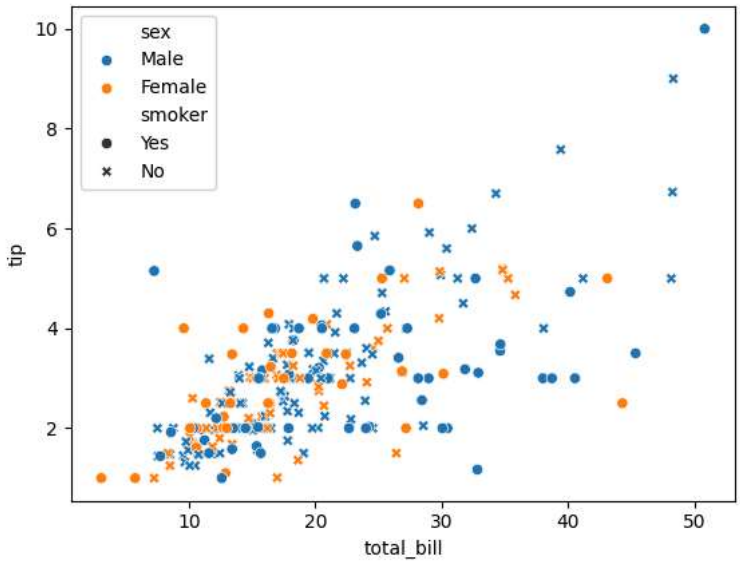

```
sns.scatterplot(x='total_bill', y='tip', hue=df['sex'], data=df)  #hue is used to add a third dimension by color-coding the poi
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



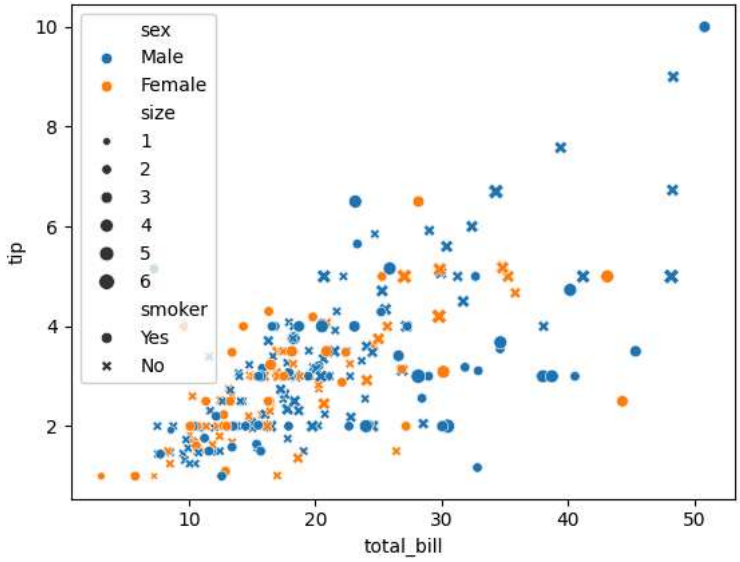

```
sns.scatterplot(x='total_bill', y='tip', hue=df['sex'], style = df['smoker'], data=df) #style is a standard parameter is used t
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



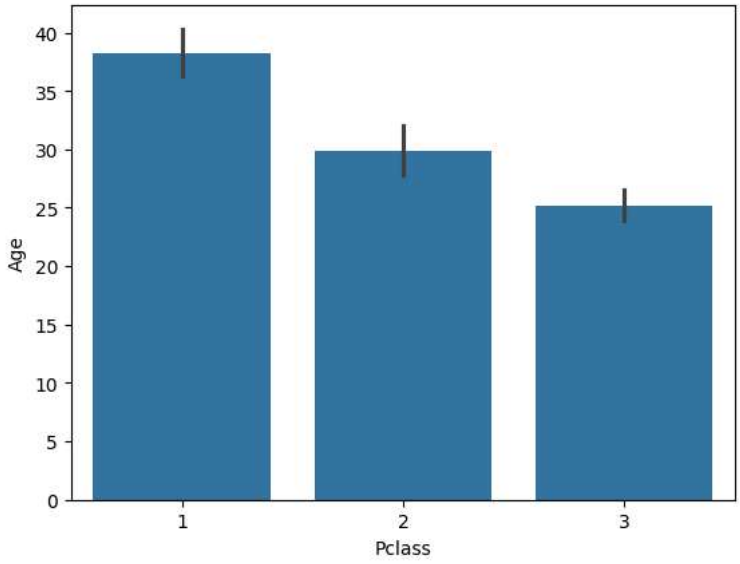

```
sns.scatterplot(x='total_bill', y='tip', hue=df['sex'], style = df['smoker'], size = df["size"] ,data=df) #The size parameter i
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



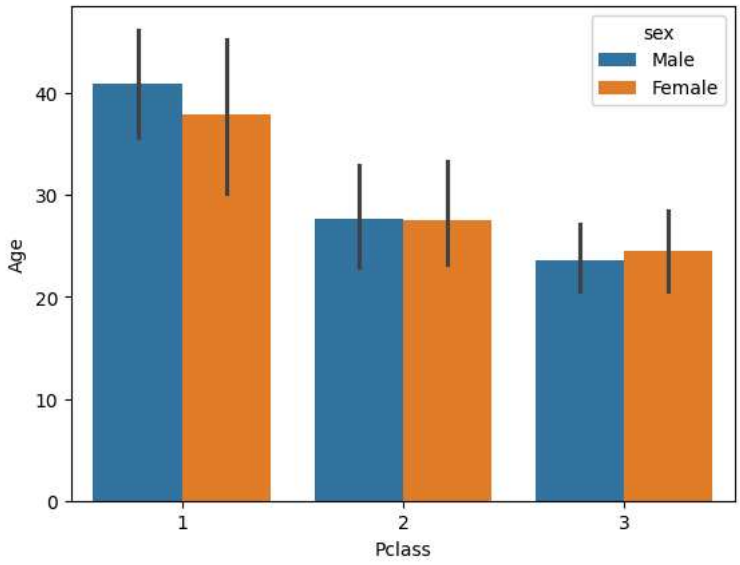

```
sns.barplot(x = "Pclass" , y = "Age", data = titanic)
```

```
<Axes: xlabel='Pclass', ylabel='Age'>
```



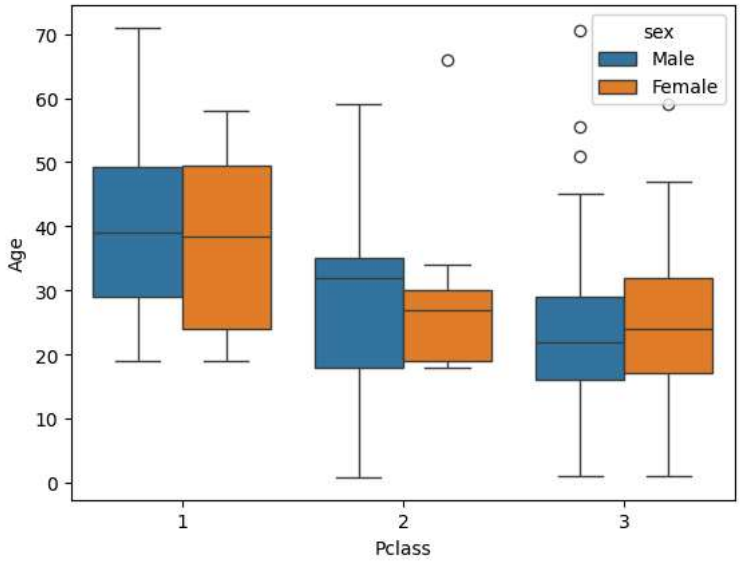

```
sns.barplot(x = "Pclass" , y = "Age", hue = df["sex"],data = titanic)  #(Outliers : are not present)
```

```
<Axes: xlabel='Pclass', ylabel='Age'>
```



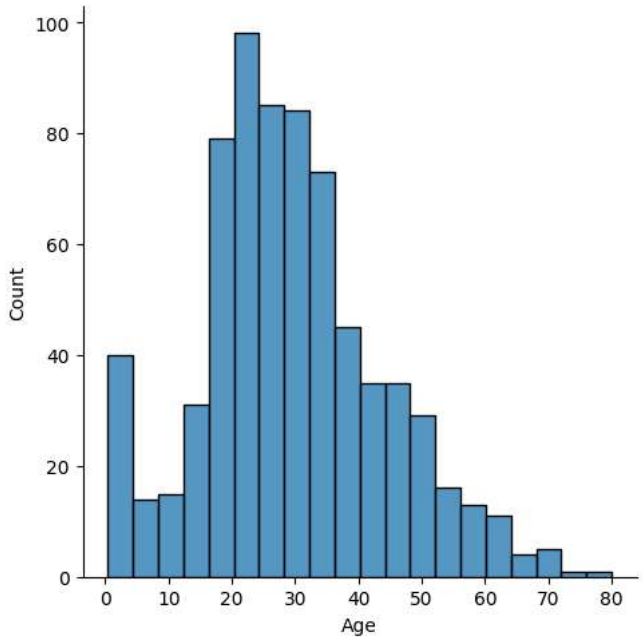

```
sns.boxplot(x = "Pclass" , y = "Age", hue = df['sex'], data = titanic) #Outliers are present (They are the values that are very
```

```
<Axes: xlabel='Pclass', ylabel='Age'>
```




```
sns.displot(titanic['Age']) #This is a way to get avergae values of particular categories (It is used to visualize the distribu
```

```
<seaborn.axisgrid.FacetGrid at 0x780f2f74be00>
```

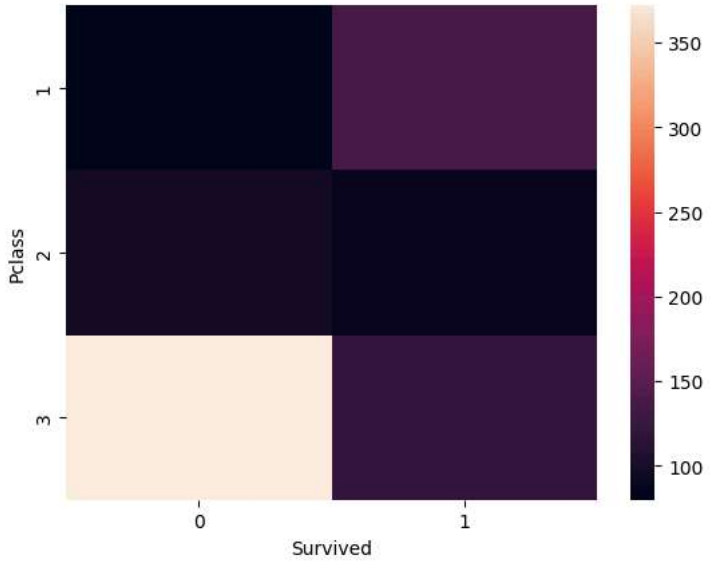



```
pd.crosstab(titanic['Pclass'],titanic['Survived']) #crosstab : it gives ius the summarize of particular tabs and It helps to: i
```

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

```
sns.heatmap(pd.crosstab(titanic['Pclass'],titanic['Survived']))
```

```
<Axes: xlabel='Survived', ylabel='Pclass'>
```




```
pd.crosstab(titanic['SibSp'],titanic['Survived'])
```

Survived	0	1
SibSp		
0	398	210
1	97	112
2	15	13
3	12	4
4	15	3
5	5	0
8	7	0

```
sns.clustermap(pd.crosstab(titanic['SibSp'],titanic['Survived']))
```

```
<seaborn.matrix.ClusterGrid at 0x780f2fc499a0>
```

```
df1 = sns.load_dataset("iris")
```

```
df1.head()
```

	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