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

In [2]:
```
lst = [1, 2, 3, 4, 5]
print(lst)
```

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

In [5]:
```
print('1D Array')
a = np.array([1, 2, 3, 4, 5])
print(a)
```

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

In [7]:
```
print('2D Array')
b = np.array([[1, 2, 3, 4, 5],
              [6, 7, 8, 9, 10]])
print(b)
```

```
2D Array
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

In [8]:
```
print('3D Array')
c = np.array([[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15]]])
print(c)
```

```
3D Array
[[[ 1  2  3  4  5]
  [ 6  7  8  9 10]
  [11 12 13 14 15]]]
```

In [9]:
```
type(a)
```

Out[9]:
```
numpy.ndarray
```

**size=Tells about no. of elements in that array**

In [11]:
```
print(a.size)
print(b.size)
print(c.size)
```

```
5
10
15
```

**shape =(row, cols)**

In [15]:
```
print(a.shape)
print(b.shape)
print(c.shape)
```

```
(5,)
(2, 5)
(1, 3, 5)
```

**dtype = Type of Data Present in the Array**

In [17]:
```
print(a.dtype)
print(b.dtype)
print(c.dtype)
```

```
int32
int32
int32
```

In [18]:
```
d = np.array([[[1, 2, 3, 4.2, 5],
               [6, 7, 8.9, 9, 10],
               [11, 12, 13, 14, 15]]])
print(d.dtype)
```

```
float64
```

**arr.transpose()**

In [20]:
```
d.transpose()
```

Out[20]:
```
array([[[ 1. ],
        [ 6. ],
        [11. ]],

       [[ 2. ],
        [ 7. ],
        [12. ]],

       [[ 3. ],
        [ 8.9],
        [13. ]],

       [[ 4.2],
        [ 9. ],
        [14. ]],

       [[ 5. ],
        [10. ],
        [15. ]]])
```

**Part 2**

np.empty((rows,cols),dtype)

In [23]:
```
np.empty((4,4), dtype = float)
```

Out[23]:
```
array([[4.67296746e-307, 1.69121096e-306, 4.89538318e-307,
        3.44898841e-307],
       [1.11256817e-306, 1.06811422e-306, 1.42417221e-306,
        1.11260619e-306],
       [8.90094053e-307, 1.86919378e-306, 1.06809792e-306,
        1.37962456e-306],
       [1.69111861e-306, 1.78020169e-306, 1.37961777e-306,
        7.56599807e-307]])
```

In [24]:
```
np.empty((4,4), dtype = int)
```

Out[24]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

np.ones((rows,cols), dtype)

In [25]:
```
x=np.ones(6)
x
```

Out[25]:
```
array([1., 1., 1., 1., 1., 1.])
```

In [27]:
```
y=np.ones((3, 5))
y
```

Out[27]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [28]:
```
z=np.ones((3, 5), dtype= int)
z
```

Out[28]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

np.zeros((rows,cols), dtype)

In [29]:
```
x=np.zeros(4)
x
```

Out[29]:
```
array([0., 0., 0., 0.])
```

In [30]:
```
y=np.zeros((4,8))
y
```

Out[30]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [31]:
```
z=np.zeros((4,8), dtype = int)
z
```

Out[31]:
```
array([[0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0]])
```

In [32]:
```
z=np.ones((4,8), dtype = str)
```

```
z
```

Out[32]:
```
array([['1', '1', '1', '1', '1', '1', '1', '1'],
       ['1', '1', '1', '1', '1', '1', '1', '1'],
       ['1', '1', '1', '1', '1', '1', '1', '1'],
       ['1', '1', '1', '1', '1', '1', '1', '1']], dtype='<U1')
```

In [35]:
```
z=np.zeros((4,8), dtype = str)
z
```

Out[35]:
```
array([['', '', '', '', '', '', '', ''],
       ['', '', '', '', '', '', '', ''],
       ['', '', '', '', '', '', '', ''],
       ['', '', '', '', '', '', '', '']], dtype='<U1')
```

In [33]:
```
z=np.ones((4,8), dtype = bool)
z
```

Out[33]:
```
array([[ True,  True,  True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True,  True,  True]])
```

In [34]:
```
z=np.zeros((4,8), dtype = bool)
z
```

Out[34]:
```
array([[False, False, False, False, False, False, False, False],
       [False, False, False, False, False, False, False, False],
       [False, False, False, False, False, False, False, False],
       [False, False, False, False, False, False, False, False]])
```

***Part 3***

np.arange(start,end,step)

In [6]:
```
a = np.arange(1, 20)
print(a)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [7]:
```
b= np.arange(1, 20, 2)
print(b)
```

```
[ 1  3  5  7  9 11 13 15 17 19]
```

In [7]:
```
c=np.arange(2, 20, 2)
print(c)
```

```
[ 2  4  6  8 10 12 14 16 18]
```

arr.reshape((rows, cols))

In [8]:
```
c=c.reshape((3,3))
c
```

Out[8]:
```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

In [4]:
```
b = np.arange(1, 100, 2)
b
```

Out[4]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
       69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

In [5]:
```
b = b.reshape(10,5)
b
```

Out[5]:
```
array([[ 1,  3,  5,  7,  9],
       [11, 13, 15, 17, 19],
       [21, 23, 25, 27, 29],
       [31, 33, 35, 37, 39],
       [41, 43, 45, 47, 49],
       [51, 53, 55, 57, 59],
       [61, 63, 65, 67, 69],
       [71, 73, 75, 77, 79],
       [81, 83, 85, 87, 89],
       [91, 93, 95, 97, 99]])
```

In [6]:
```
b = b.flatten()
b
```

Out[6]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
       69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

In [9]:
```
c = c.ravel()
c
```

Out[9]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

***Part 4***

In [10]:
```
a = np.arange(1, 51)
a = a.reshape(10, 5)
a
```

Out[10]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25],
       [26, 27, 28, 29, 30],
       [31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45],
       [46, 47, 48, 49, 50]])
```

In [12]:
```
a[0]
```

Out[12]:
```
array([1, 2, 3, 4, 5])
```

In [13]:
```
a[2]
```

Out[13]:
```
array([11, 12, 13, 14, 15])
```

In [14]:
```
a[3, 4]
```

Out[14]:
```
20
```

In [15]:
```
a[2:5]
```

Out[15]:
```
array([[11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [16]:
```
a[0:10]
```

Out[16]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25],
       [26, 27, 28, 29, 30],
       [31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45],
       [46, 47, 48, 49, 50]])
```

In [17]:
```
a[:, 2]
```

Out[17]:
```
array([ 3,  8, 13, 18, 23, 28, 33, 38, 43, 48])
```

In [18]:
```
a[2:5, 4]
```

Out[18]:
```
array([15, 20, 25])
```

In [19]:
```
a[:,:]
```

Out[19]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25],
       [26, 27, 28, 29, 30],
       [31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45],
       [46, 47, 48, 49, 50]])
```

In [20]:
```
a[:, 2:5]
```

Out[20]:
```
array([[ 3,  4,  5],
       [ 8,  9, 10],
       [13, 14, 15],
       [18, 19, 20],
       [23, 24, 25],
       [28, 29, 30],
       [33, 34, 35],
       [38, 39, 40],
       [43, 44, 45],
       [48, 49, 50]])
```

In [22]:
```
a[:,2:5]. dtype
```

Out[22]:
```
dtype('int32')
```

***Part 5***

In [23]:
```
a = np.arange(0, 18).reshape((6,3))
b = np.arange(20, 38).reshape((6,3))
print(a)
print(b)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [15 16 17]]
[[20 21 22]
 [23 24 25]
 [26 27 28]
 [29 30 31]
 [32 33 34]
 [35 36 37]]
```

In [24]:
```
a+b
```

Out[24]:
```
array([[20, 22, 24],
       [26, 28, 30],
       [32, 34, 36],
       [38, 40, 42],
       [44, 46, 48],
       [50, 52, 54]])
```

In [25]:
```
np.add(a,b)
```

Out[25]:
```
array([[20, 22, 24],
       [26, 28, 30],
       [32, 34, 36],
       [38, 40, 42],
       [44, 46, 48],
       [50, 52, 54]])
```

In [26]:
```
a-b
```

Out[26]:
```
array([[-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20]])
```

In [27]:
```
np.subtract(a,b)
```

Out[27]:
```
array([[-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20],
       [-20, -20, -20]])
```

In [28]:
```
a*b
```

Out[28]:
```
array([[  0,  21,  44],
       [ 69,  96, 125],
       [156, 189, 224],
       [261, 300, 341],
       [384, 429, 476],
       [525, 576, 629]])
```

In [29]:
```
np.multiply(a,b)
```

Out[29]:
```
array([[  0,  21,  44],
       [ 69,  96, 125],
       [156, 189, 224],
       [261, 300, 341],
       [384, 429, 476],
       [525, 576, 629]])
```

In [30]:
```
a/b
```

Out[30]:
```
array([[0.        , 0.04761905, 0.09090909],
       [0.13043478, 0.16666667, 0.2       ],
       [0.23076923, 0.25925926, 0.28571429],
       [0.31034483, 0.33333333, 0.35483871],
       [0.375     , 0.39393939, 0.41176471],
       [0.42857143, 0.44444444, 0.45945946]])
```

In [31]:
```
np.divide(a,b)
```

Out[31]:
```
array([[0.        , 0.04761905, 0.09090909],
       [0.13043478, 0.16666667, 0.2       ],
       [0.23076923, 0.25925926, 0.28571429],
       [0.31034483, 0.33333333, 0.35483871],
       [0.375     , 0.39393939, 0.41176471],
       [0.42857143, 0.44444444, 0.45945946]])
```

In [37]:
```
print(a.shape)
print(b.shape)
```

```
(6, 3)
(6, 3)
```

In [35]:
```
a.dot(b)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[35], line 1
----> 1 a.dot(b)

ValueError: shapes (6,3) and (6,3) not aligned: 3 (dim 1) != 6 (dim 0)
```

In [38]:
```
b.max()
```

Out[38]:
```
37
```

In [39]:
```
b.min()
```

Out[39]:
```
20
```

In [40]:
```
b.argmax()
```

Out[40]:
```
17
```

In [41]:
```
np.sum(b)
```

Out[41]:
```
513
```

In [43]:
```
np.sum(b, axis =1)
```

Out[43]:
```
array([ 63,  72,  81,  90,  99, 108])
```

In [44]:
```
np.sum(b, axis =0)
```

Out[44]:
```
array([165, 171, 177])
```

In [45]:
```
np.mean(b)
```

Out[45]:
```
28.5
```

In [46]:
```
np.sqrt(b)
```

Out[46]:
```
array([[4.47213595, 4.58257569, 4.69041576],
       [4.79583152, 4.89897949, 5.        ],
       [5.09901951, 5.19615242, 5.29150262],
       [5.38516481, 5.47722558, 5.56776436],
       [5.65685425, 5.74456265, 5.83095189],
       [5.91607978, 6.        , 6.08276253]])
```

In [47]:
```
np.std(b)
```

Out[47]:
```
5.188127472091127
```

In [48]:
```
np.log(b)
```

Out[48]:
```
array([[2.99573227, 3.04452244, 3.09104245],
       [3.13549422, 3.17805383, 3.21887582],
       [3.25809654, 3.29583687, 3.33220451],
       [3.36729583, 3.40119738, 3.4339872 ],
       [3.4657359 , 3.49650756, 3.52636052],
       [3.55534806, 3.58351894, 3.61091791]])
```

***Part 6***

----Numpy Trigonometric Operations

In [2]:
```
import matplotlib.pyplot as plt
plt.style.use('dark_background')
```

In [3]:
```
np.pi
```

Out[3]:
```
3.141592653589793
```

In [4]:
```
np.sin(np.pi/2) # 90 degrees
```

In [4]:
```
np.sin(np.pi/2) # 90 degrees
```

Out[4]:
```
1.0
```

In [5]:
```
np.sin(np.pi/6)
```

Out[5]:
```
0.49999999999999994
```

In [6]:
```
np.cos(np.pi/2)
```

Out[6]:
```
6.123233995736766e-17
```

In [7]:
```
np.tan(np.pi/2)
```

Out[7]:
```
1.633123935319537e+16
```

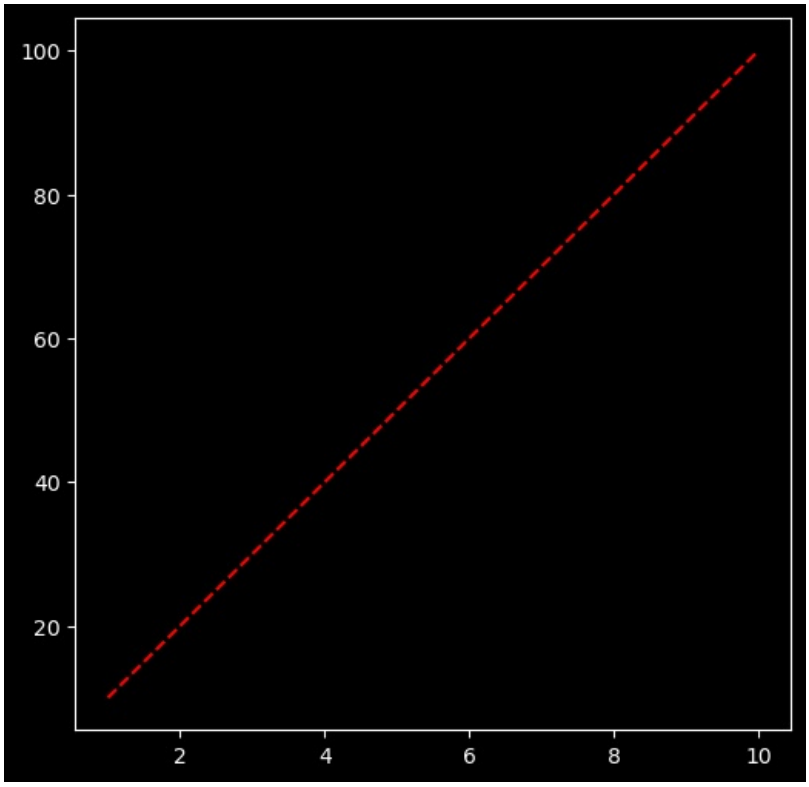
In [8]:
```
np.tan(0)
```

Out[8]:
```
0.0
```

Using Matplotlib with Numpy

In [10]:
```
x=np.arange(1, 11)
y=np.arange(10, 110, 10)

plt.figure(figsize = (6,6))
plt.plot(x,y, 'r--')
plt.show()
```

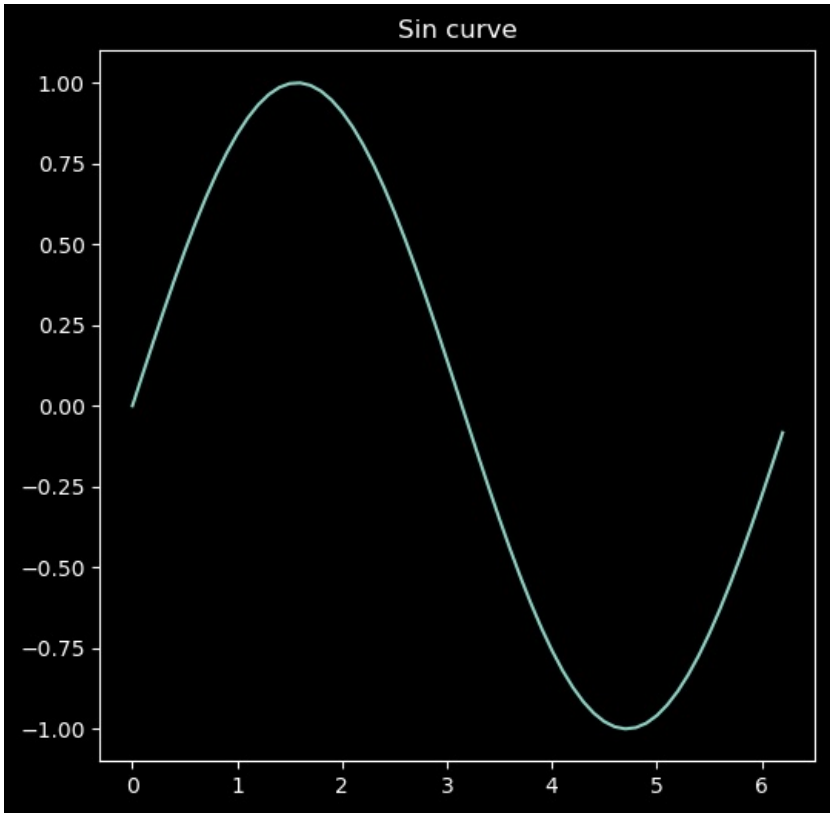



Plotting Trigonometric Curve

In [11]:
```
x_sin = np.arange(0, 2*np.pi, 0.1)
y_sin = np.sin(x_sin)
print(y_sin)
```

```
[ 0.          0.09983342  0.19866933  0.29552021  0.38941834  0.47942554
  0.56464247  0.64421769  0.71735609  0.78332691  0.84147098  0.89120736
  0.93203909  0.96355819  0.98544973  0.99749499  0.9995736   0.99166481
  0.97384763  0.94630009  0.90929743  0.86320937  0.8084964   0.74570521
  0.67546318  0.59847214  0.51550137  0.42737988  0.33498815  0.23924933
  0.14112001  0.04158066 -0.05837414 -0.15774569 -0.2555411  -0.35078323
 -0.44252044 -0.52983614 -0.61185789 -0.68776616 -0.7568025  -0.81827711
 -0.87157577 -0.91616594 -0.95160207 -0.97753012 -0.993691   -0.99992326
 -0.99616461 -0.98245261 -0.95892427 -0.92581468 -0.88345466 -0.83226744
 -0.77276449 -0.70554033 -0.63126664 -0.55068554 -0.46460218 -0.37387666
 -0.2794155  -0.1821625  -0.0830894 ]
```

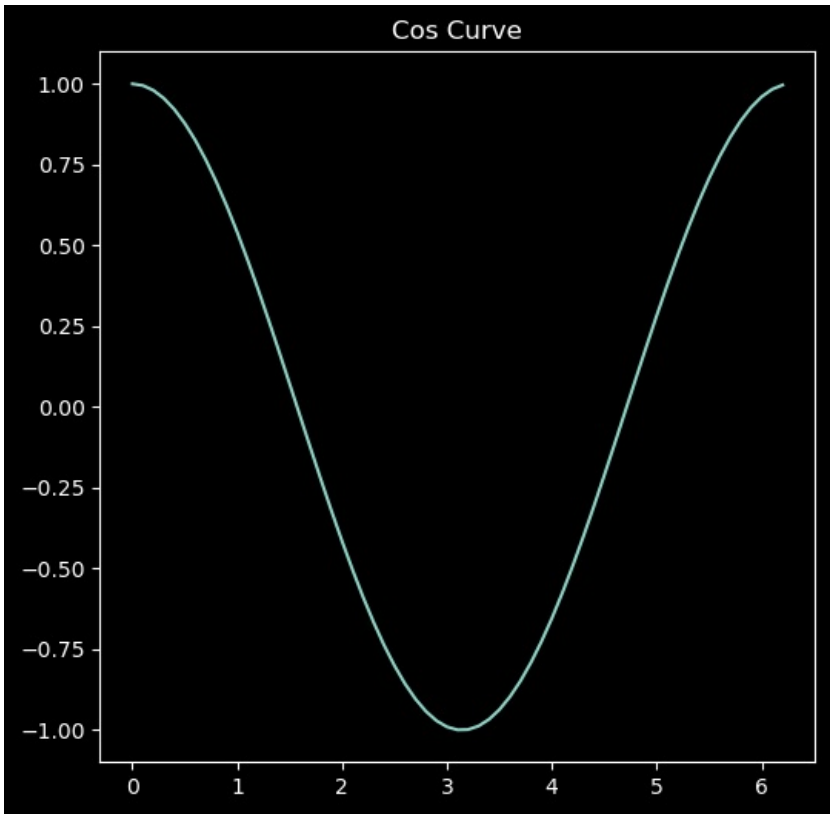
In [12]:
```
plt.figure(figsize = (6, 6))
plt.plot(x_sin, y_sin)
plt.title('Sin curve')
plt.show()
```




In [13]:
```
x_cos = np.arange(0, 2*np.pi, 0.1)
y_cos = np.cos(x_sin)
print(y_cos)
```

```
[ 1.          0.99500417  0.98006658  0.95533649  0.92106099  0.87758256
  0.82533561  0.76484219  0.69670671  0.62160997  0.54030231  0.45359612
  0.36235775  0.26749883  0.16996714  0.0707372  -0.02919952 -0.12884449
 -0.22720209 -0.32328957 -0.41614684 -0.5048461  -0.58850112 -0.66627602
 -0.73739372 -0.80114362 -0.85688875 -0.90407214 -0.94222234 -0.97095817
 -0.9899925  -0.99913515 -0.99829478 -0.98747977 -0.96679819 -0.93645669
 -0.89675842 -0.84810003 -0.79096771 -0.7259323  -0.65364362 -0.57482395
 -0.49026082 -0.40079917 -0.30733287 -0.2107958  -0.11215253 -0.01238866
  0.08749898  0.18651237  0.28366219  0.37797774  0.46851667  0.55437434
  0.63469288  0.70866977  0.77556588  0.83471278  0.88551952  0.92747843
  0.96017029  0.98326844  0.9965421 ]
```

In [14]:
```
plt.figure(figsize = (6, 6))
plt.plot(x_cos, y_cos)
plt.title('Cos Curve')
plt.show()
```

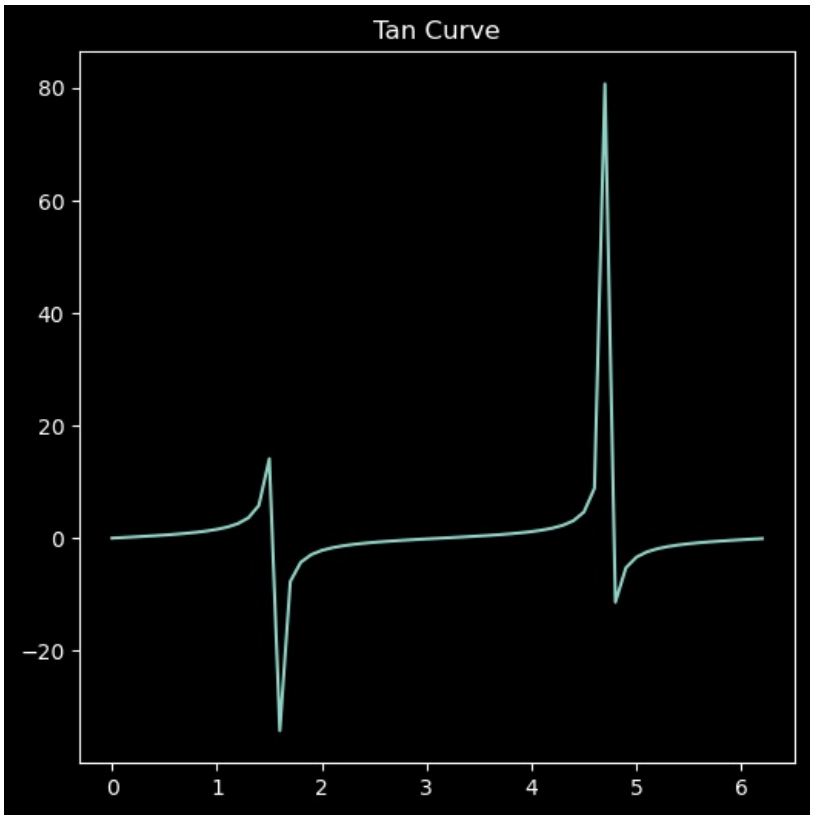



In [15]:
```
x_tan = np.arange(0, 2*np.pi, 0.1)
```

In [15]:
```
x_tan = np.arange(0, 2*np.pi, 0.1)
y_tan = np.tan(x_tan)
print(y_tan)
```

```
[ 0.00000000e+00  1.00334672e-01  2.02710036e-01  3.09336250e-01
  4.22793219e-01  5.46302490e-01  6.84136808e-01  8.42288380e-01
  1.02963856e+00  1.26015822e+00  1.55740772e+00  1.96475966e+00
  2.57215162e+00  3.60210245e+00  5.79788372e+00  1.41014199e+01
 -3.42325327e+01 -7.69660214e+00 -4.28626167e+00 -2.92709751e+00
 -2.18503986e+00 -1.70984654e+00 -1.37382306e+00 -1.11921364e+00
 -9.16014290e-01 -7.47022297e-01 -6.01596613e-01 -4.72727629e-01
 -3.55529832e-01 -2.46405394e-01 -1.42546543e-01 -4.16166546e-02
  5.84738545e-02  1.59745748e-01  2.64316901e-01  3.74585640e-01
  4.93466730e-01  6.24733075e-01  7.73556091e-01  9.47424650e-01
  1.15782128e+00  1.42352648e+00  1.77777977e+00  2.28584788e+00
  3.09632378e+00  4.63733205e+00  8.86017490e+00  8.07127630e+01
 -1.13848707e+01 -5.26749307e+00 -3.38051501e+00 -2.44938942e+00
 -1.88564188e+00 -1.50127340e+00 -1.21754082e+00 -9.95584052e-01
 -8.13943284e-01 -6.59730572e-01 -5.24666222e-01 -4.03110900e-01
 -2.91006191e-01 -1.85262231e-01 -8.33777149e-02]
```

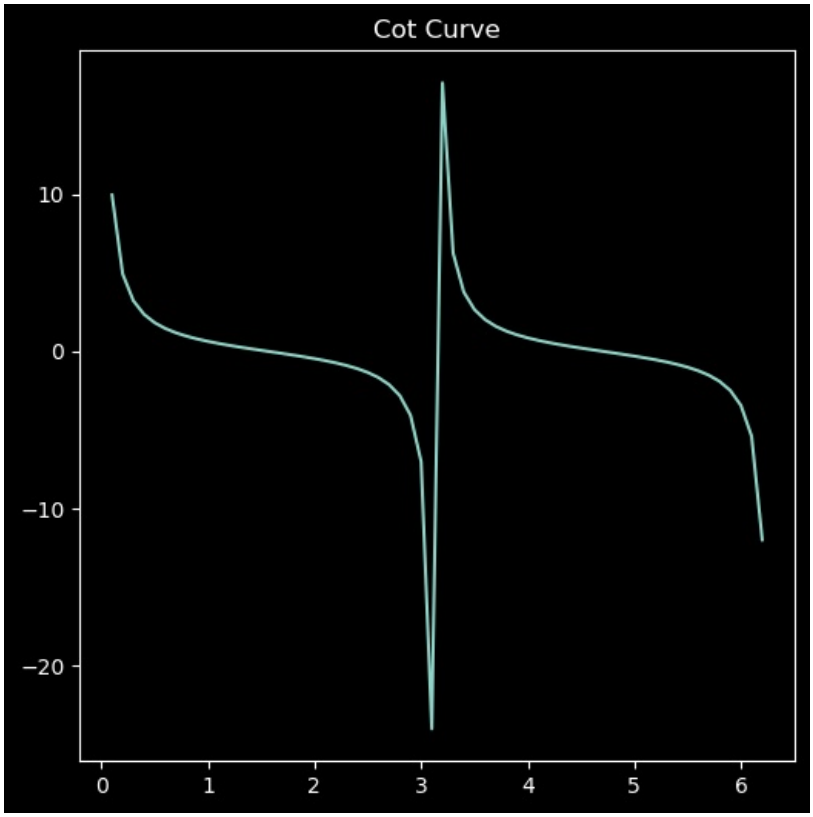
In [17]:
```
plt.figure(figsize = (6, 6))
plt.plot(x_tan, y_tan)
plt.title('Tan Curve')
plt.show()
```




In [19]:
```
x_cot = np.arange(0, 2*np.pi, 0.1)
y_cot = 1/np.tan(x_cot)
print(y_cot)

plt.figure(figsize = (6, 6))
plt.plot(x_cot, y_cot)
plt.title('Cot Curve')
plt.show()
```

```
C:\Users\PAVILION\AppData\Local\Temp\ipykernel_11600\3926713967.py:2: RuntimeWarning: divide by zero encountere
d in divide
  y_cot = 1/np.tan(x_cot)
[            inf  9.96664442e+00  4.93315488e+00  3.23272814e+00
  2.36522242e+00  1.83048772e+00  1.46169595e+00  1.18724183e+00
  9.71214601e-01  7.93551148e-01  6.42092616e-01  5.08968105e-01
  3.88779569e-01  2.77615647e-01  1.72476726e-01  7.09148443e-02
 -2.92119782e-02 -1.29927464e-01 -2.33303535e-01 -3.41635355e-01
 -4.57657554e-01 -5.84847806e-01 -7.27895776e-01 -8.93484463e-01
 -1.09168603e+00 -1.33864813e+00 -1.66224340e+00 -2.11538302e+00
 -2.81270349e+00 -4.05835272e+00 -7.01525255e+00 -2.40288416e+01
  1.71016604e+01  6.25994754e+00  3.78333734e+00  2.66961648e+00
  2.02647907e+00  1.60068362e+00  1.29273108e+00  1.05549291e+00
  8.63691154e-01  7.02480784e-01  5.62499368e-01  4.37474431e-01
  3.22963640e-01  2.15641233e-01  1.12864589e-01  1.23896143e-02
 -8.78358684e-02 -1.89843629e-01 -2.95812916e-01 -4.08265012e-01
 -5.30323394e-01 -6.66101193e-01 -8.21327696e-01 -1.00443553e+00
 -1.22858683e+00 -1.51577029e+00 -1.90597366e+00 -2.48070692e+00
 -3.43635300e+00 -5.39775429e+00 -1.19936125e+01]
```



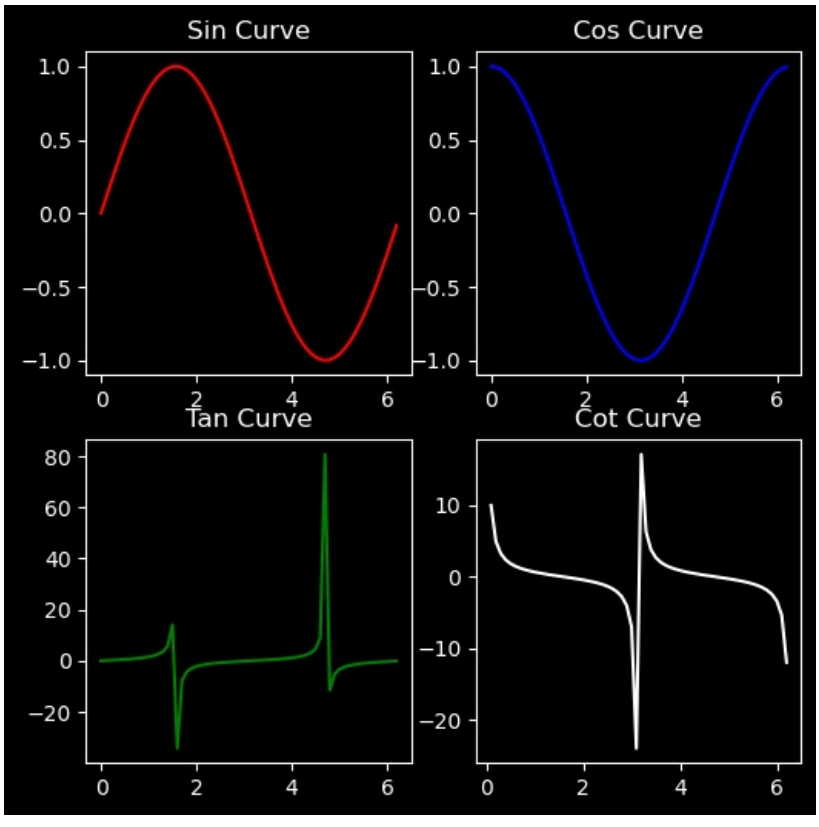

In [21]:
```
plt.figure(figsize = (6, 6))
plt.subplot(2,2,1)
plt.plot(x_sin, y_sin, 'r-')
plt.title('Sin Curve')

plt.subplot(2,2,2)
plt.plot(x_cos, y_cos, 'b-')
plt.title('Cos Curve')

plt.subplot(2,2,3)
plt.plot(x_tan, y_tan, 'g-')
plt.title('Tan Curve')

plt.subplot(2,2,4)
plt.plot(x_cot, y_cot, 'w-')
plt.title('Cot Curve')

plt.show()
```




**Part 7** Numpy Random Operations

***Part 7*** Numpy Random Operations

In [22]:
```
np.random.random(1)
```

Out[22]:
```
array([0.33782977])
```

In [23]:
```
np.random.random(2)
```

Out[23]:
```
array([0.69936354, 0.56259747])
```

In [25]:
```
np.random.random((2,2))
```

Out[25]:
```
array([[0.65522393, 0.41462385],
       [0.6464932 , 0.25499563]])
```

In [26]:
```
np.random.randint(1, 10)
```

Out[26]:
```
7
```

In [27]:
```
np.random.randint(1, 10, (2,2))
```

Out[27]:
```
array([[5, 3],
       [8, 2]])
```

In [28]:
```
np.random.randint(1, 10, (3,4,5))
```

Out[28]:
```
array([[[1, 9, 1, 8, 8],
        [8, 7, 9, 5, 6],
        [4, 5, 1, 2, 5],
        [4, 6, 3, 2, 1]],

       [[9, 7, 2, 3, 8],
        [6, 5, 5, 4, 7],
        [7, 1, 5, 2, 1],
        [9, 5, 8, 3, 6]],

       [[7, 2, 3, 4, 4],
        [8, 9, 6, 6, 5],
        [9, 5, 8, 6, 7],
        [7, 8, 7, 4, 4]]])
```

In [29]:
```
np.random.rand(2,2)
```

Out[29]:
```
array([[0.64445377, 0.49158132],
       [0.58565799, 0.4777468 ]])
```

In [30]:
```
np.random.randn(2,2)
```

Out[30]:
```
array([[-0.45436393, -1.84861965],
       [-1.69562612,  1.13892865]])
```

In [31]:
```
a=np.arange(1,10)
print(a)
```

```
[1 2 3 4 5 6 7 8 9]
```

In [32]:
```
np.random.choice(a)
```

Out[32]:
```
9
```

***Part 8*** Numpy String Operations/ np.char

In [35]:
```
s1 = 'Collins is my name'
s2 = 'I am a Nigeria'
```

In [36]:
```
np.char.add(s1, s2)
```

Out[36]:
```
array('Collins is my nameI am a Nigeria', dtype='<U32')
```

In [37]:
```
np.char.upper(s1)
```

Out[37]:
```
array('COLLINS IS MY NAME', dtype='<U18')
```

In [38]:
```
np.char.lower(s1)
```

Out[38]:
```
array('collins is my name', dtype='<U18')
```

In [39]:
```
np.char.split(s1)
```

Out[39]:
```
array(list(['Collins', 'is', 'my', 'name']), dtype=object)
```

In [40]:
```
s3 = ' Collins is my/name'
np.char.splitlines(s3)
```

Out[40]:
```
array(list([' Collins is my/name']), dtype=object)
```

In [41]:
```
np.char.replace(s1, 'name', 'surname')
```

Out[41]:
```
array('Collins is my surname', dtype='<U21')
```

In [42]:
```
print('********good bye********')
```

```
********good bye********
```

In [43]:
```
print(np.char.center('good bye', 80, '*'))
```

```
**********************************good bye***********************************
```

### Pandas Library

--------Part 1--------

# I mporting Library

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

pd.__version__

## Pandas Series with Python Lists

In [46]:
```
Ist =(1,2,3,4,5,6)
print(Ist)
```

```
(1, 2, 3, 4, 5, 6)
```

In [49]:
```
series = pd.Series(Ist)
print(series)
print(type(series))
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
<class 'pandas.core.series.Series'>
```

In [51]:
```
empty = pd.Series([])
empty
```

Out[51]:
```
Series([], dtype: object)
```

In [52]:
```
a = pd.Series(['p','q','r','s','t'], index = [10,11,12,13,14])
a
```

Out[52]:
```
10    p
11    q
12    r
13    s
14    t
dtype: object
```

In [53]:
```
b = pd.Series(['p','q','r','s','t'], index = [10,11,12,13,14], name ='alphabets')
b
```

Out[53]:
```
10    p
11    q
12    r
13    s
14    t
Name: alphabets, dtype: object
```

In [54]:
```
scalar_series = pd.Series(0.5)
scalar_series
```

Out[54]:
```
0    0.5
dtype: float64
```

### PART 1 Pandas

## Importing Library

In [2]:
```
import pandas as pd
```

## Pandas Series with with Python Lists

In [4]:
```
lst = [1, 2, 3, 4, 5, 6]
print(lst)
```

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

In [12]:
```
series = pd.Series(lst)
print(series)
print(type(series))
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
<class 'pandas.core.series.Series'>
```

In [13]:
```
empty = pd.Series([])
empty
```

Out[13]:
```
Series([], dtype: object)
```

In [15]:
```
b = pd.Series(['p','q','r','s','t'], index = [10,11,12,13,14], name = 'alphabets')
b
```

Out[15]:
```
10    p
11    q
12    r
13    s
14    t
Name: alphabets, dtype: object
```

In [18]:
```
scalar_series = pd.Series(0.5)
scalar_series
```

Out[18]:
```
0    0.5
dtype: float64
```

In [19]:
```
scalar_series = pd.Series(0.5, index = [1,2,3])
scalar_series
```

Out[19]:
```
1    0.5
2    0.5
3    0.5
dtype: float64
```

## Pandas Series with Python Dictionary

In [22]:
```
dict_series = pd.Series({'p':1, 'q':2, 'r':3, 's':4, 't':5})
dict_series
```

Out[22]:
```
p    1
q    2
r    3
s    4
t    5
dtype: int64
```

In [24]:
```
dict_series[0]
```

Out[24]:
```
1
```

In [25]:
```
dict_series[0:3]
```

Out[25]:
```
p    1
q    2
r    3
dtype: int64
```

In [26]:
```
max(dict_series)
```

Out[26]:
```
5
```

In [28]:
```
dict_series = pd.Series({'p':[1,5,6], 'q':[2,6,7],'r':[3,9,0], 's':[4,4,5], 't':[5,1,2]})
dict_series
```

Out[28]:
```
p    [1, 5, 6]
q    [2, 6, 7]
r    [3, 9, 0]
s    [4, 4, 5]
t    [5, 1, 2]
dtype: object
```

## Pandas DataFrame

In [30]:
```
df = pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

## DataFrame Using List

In [31]:
```
lst = [1,2,3,4,5]
df = pd.DataFrame(lst)
df
```

Out[31]:

| | 0 |
|---|---|
| **0** | 1 |
| **1** | 2 |
| **2** | 3 |
| **3** | 4 |
| **4** | 5 |

In [32]:
```
lst = [[1,2,3,4,5], [11,12,13,14,15]]
df = pd.DataFrame(lst)
df
```

Out[32]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 4 | 5 |
| **1** | 11 | 12 | 13 | 14 | 15 |

In [36]:
```
a = [{'a':5, 'b':7, 'c':9, 'd':2},
     {'a':4, 'b':8, 'c':19,'d':12}]    # Dictionary keys represents column names
df = pd.DataFrame(a)
df
```

Out[36]:

| | a | b | c | d |
|---|---|---|---|---|
| **0** | 5 | 7 | 9 | 2 |
| **1** | 4 | 8 | 19 | 12 |

In [2]:
```
b = {'RollNo':pd.Series([1,2,3,4,5]),
     'Maths':pd.Series([67,89,23,90,56]),
     'Physics':pd.Series([12,98,44,98,78])}

df = pd.DataFrame(b)
df
```

Out[2]:

| | RollNo | Maths | Physics |
|---|---|---|---|
| **0** | 1 | 67 | 12 |
| **1** | 2 | 89 | 98 |
| **2** | 3 | 23 | 44 |
| **3** | 4 | 90 | 98 |
| **4** | 5 | 56 | 78 |

## PART 3: Reading CSV FILE

In [4]:
```
df = pd.read_csv('Salary_Data.csv')
df
```

Out[4]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | 32.0 | Male | Bachelor's | Software Engineer | 5.0 | 90000.0 |
| **1** | 28.0 | Female | Master's | Data Analyst | 3.0 | 65000.0 |
| **2** | 45.0 | Male | PhD | Senior Manager | 15.0 | 150000.0 |
| **3** | 36.0 | Female | Bachelor's | Sales Associate | 7.0 | 60000.0 |
| **4** | 52.0 | Male | Master's | Director | 20.0 | 200000.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **6699** | 49.0 | Female | PhD | Director of Marketing | 20.0 | 200000.0 |
| **6700** | 32.0 | Male | High School | Sales Associate | 3.0 | 50000.0 |
| **6701** | 30.0 | Female | Bachelor's Degree | Financial Manager | 4.0 | 55000.0 |
| **6702** | 46.0 | Male | Master's Degree | Marketing Manager | 14.0 | 140000.0 |
| **6703** | 26.0 | Female | High School | Sales Executive | 1.0 | 35000.0 |

6704 rows × 6 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Age', 'Gender', 'Education Level', 'Job Title', 'Years of Experience',
       'Salary'],
      dtype='object')
```

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

Out[6]:
```
(6704, 6)
```

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

Out[7]:
```
40224
```

In [8]:
```
df.head()
```

Out[8]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | 32.0 | Male | Bachelor's | Software Engineer | 5.0 | 90000.0 |
| **1** | 28.0 | Female | Master's | Data Analyst | 3.0 | 65000.0 |
| **2** | 45.0 | Male | PhD | Senior Manager | 15.0 | 150000.0 |
| **3** | 36.0 | Female | Bachelor's | Sales Associate | 7.0 | 60000.0 |
| **4** | 52.0 | Male | Master's | Director | 20.0 | 200000.0 |

In [9]:
```
df.head(3)
```

Out[9]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | 32.0 | Male | Bachelor's | Software Engineer | 5.0 | 90000.0 |
| **1** | 28.0 | Female | Master's | Data Analyst | 3.0 | 65000.0 |
| **2** | 45.0 | Male | PhD | Senior Manager | 15.0 | 150000.0 |

In [10]:
```
df.tail()
```

Out[10]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **6699** | 49.0 | Female | PhD | Director of Marketing | 20.0 | 200000.0 |
| **6700** | 32.0 | Male | High School | Sales Associate | 3.0 | 50000.0 |
| **6701** | 30.0 | Female | Bachelor's Degree | Financial Manager | 4.0 | 55000.0 |
| **6702** | 46.0 | Male | Master's Degree | Marketing Manager | 14.0 | 140000.0 |
| **6703** | 26.0 | Female | High School | Sales Executive | 1.0 | 35000.0 |

In [11]:
```
df.describe()
```

Out[11]:

| | Age | Years of Experience | Salary |
|---|---|---|---|
| **count** | 6702.000000 | 6701.000000 | 6699.000000 |
| **mean** | 33.620859 | 8.094687 | 115326.964771 |
| **std** | 7.614633 | 6.059003 | 52786.183911 |
| **min** | 21.000000 | 0.000000 | 350.000000 |
| **25%** | 28.000000 | 3.000000 | 70000.000000 |
| **50%** | 32.000000 | 7.000000 | 115000.000000 |
| **75%** | 38.000000 | 12.000000 | 160000.000000 |
| **max** | 62.000000 | 34.000000 | 250000.000000 |

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6704 entries, 0 to 6703
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Age                   6702 non-null   float64
 1   Gender                6702 non-null   object 
 2   Education Level       6701 non-null   object 
 3   Job Title             6702 non-null   object 
 4   Years of Experience   6701 non-null   float64
 5   Salary                6699 non-null   float64
dtypes: float64(3), object(3)
memory usage: 314.4+ KB
```

## PART 4: Handling Missing Values

In [13]:
```
df.isnull()
```

Out[13]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... |
| **6699** | False | False | False | False | False | False |
| **6700** | False | False | False | False | False | False |
| **6701** | False | False | False | False | False | False |
| **6702** | False | False | False | False | False | False |
| **6703** | False | False | False | False | False | False |

6704 rows × 6 columns

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Age                   2
Gender                2
Education Level       3
Job Title             2
Years of Experience   3
Salary                5
dtype: int64
```

In [15]:
```
df.isnull().sum().sum()
```

Out[15]:
```
17
```

## Dropping Row with NAN Values

In [16]:
```
df2 = df.dropna()       # default axis = 0
```

In [17]:
```
df2.shape
```

Out[17]:
```
(6698, 6)
```

## Dropping Column with NAN values

In [25]:
```
df3 = df.dropna(axis = 1)          # axis =1 for column
```

In [26]:
```
df3.shape
```

Out[26]:
```
(6698, 6)
```

In [18]:
```
df.dropna(how = 'any')      # if any row value is null then remove that row
```

Out[18]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | 32.0 | Male | Bachelor's | Software Engineer | 5.0 | 90000.0 |
| **1** | 28.0 | Female | Master's | Data Analyst | 3.0 | 65000.0 |
| **2** | 45.0 | Male | PhD | Senior Manager | 15.0 | 150000.0 |
| **3** | 36.0 | Female | Bachelor's | Sales Associate | 7.0 | 60000.0 |
| **4** | 52.0 | Male | Master's | Director | 20.0 | 200000.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **6699** | 49.0 | Female | PhD | Director of Marketing | 20.0 | 200000.0 |
| **6700** | 32.0 | Male | High School | Sales Associate | 3.0 | 50000.0 |
| **6701** | 30.0 | Female | Bachelor's Degree | Financial Manager | 4.0 | 55000.0 |
| **6702** | 46.0 | Male | Master's Degree | Marketing Manager | 14.0 | 140000.0 |
| **6703** | 26.0 | Female | High School | Sales Executive | 1.0 | 35000.0 |

6698 rows × 6 columns

In [19]:
```
df.dropna(how = 'all')       # if all row values is null then remove  that row
```

Out[19]:

| | Age | Gender | Education Level | Job Title | Years of Experience | Salary |
|---|---|---|---|---|---|---|
| **0** | 32.0 | Male | Bachelor's | Software Engineer | 5.0 | 90000.0 |
| **1** | 28.0 | Female | Master's | Data Analyst | 3.0 | 65000.0 |
| **2** | 45.0 | Male | PhD | Senior Manager | 15.0 | 150000.0 |
| **3** | 36.0 | Female | Bachelor's | Sales Associate | 7.0 | 60000.0 |
| **4** | 52.0 | Male | Master's | Director | 20.0 | 200000.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **6699** | 49.0 | Female | PhD | Director of Marketing | 20.0 | 200000.0 |
| **6700** | 32.0 | Male | High School | Sales Associate | 3.0 | 50000.0 |
| **6701** | 30.0 | Female | Bachelor's Degree | Financial Manager | 4.0 | 55000.0 |
| **6702** | 46.0 | Male | Master's Degree | Marketing Manager | 14.0 | 140000.0 |
| **6703** | 26.0 | Female | High School | Sales Executive | 1.0 | 35000.0 |

6702 rows × 6 columns

In [20]:
```
df.dropna(inplace = True)
```

In [21]:
```
df.shape
```

Out[21]:
```
(6698, 6)
```

## Filling in the NULL values

In [ ]: