1.a)Write python programs to implement indexing, slicing and splitting, iterating a list.

**import** numpy **as** np

array1d **=** np**.**array([1, 2, 3, 4, 5, 6,7,8])

array2d **=** np**.**array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

array3d **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(array1d)

print("---------Indexing operation ----------")

print("---------Indexing on 1D----------")

print(array1d[0]) *# Get first value*

print(array1d[**-**1]) *# Get last value*

print(array1d[3]) *# Get 4th value from first*

print(array1d[**-**5]) *# Get 5th value from last*

print(array1d[[0,2,1]]) *#Get multiple values from array*

print("---------Indexing on 2D----------")

print(array2d)

print(array2d[0, 0]) *# Get first row first col*

print(array2d[0, 1]) *# Get first row second col*

print(array2d[0, 2]) *# Get first row third col*

print(array2d[0, 1]) *# Get first row second col*

print(array2d[**-**1, 1]) *# Get second row second col*

print(array2d[2, **-**2]) *# Get third row second col*

print(array2d[[0,1,2],[1,0,2]])*#Get multiple values from array*

print(array2d[[0,1,2],[**-**1,0,**-**3]])*#Get multiple values from array*

print(array2d[0,2]**+**array2d[1,1])*#adding the elements on array*

print("---------Indexing on 3D----------")

print(array3d)

print(array3d[0, 1, 2])

print(array3d[0, **-**2, 1])

print(array3d[0, 0, 2])

print(array3d[[0,1,0],[1,0,1],[1,0,2]])

print(array3d[[0,1,0,1],[1,0,1,**-**1],[1,0,2,**-**3]])

print(array3d[1,**-**1,2]**+**array3d[0, **-**1, **-**3])

print("---------Slicing operation----------")

print("---------Slicing on 1D----------")

print(array1d)

print(array1d[4:]) *# From index 4 to last index*

print(array1d[:4]) *# From index 0 to 4 index*

print(array1d[4:7]) *# From index 4(included) up to index 7(excluded)*

print(array1d[:**-**1]) *# Excluded last element*

print(array1d[:**-**2]) *# Up to second last index(negative index)*

print(array1d[::**-**1]) *# From last to first in reverse order(negative step)*

print(array1d[::**-**2]) *# All odd numbers in reversed order*

print(array1d[**-**2::**-**2]) *# All even numbers in reversed order*

print(array1d[::]) *# All elements*

array1d[0]**=**10 *#modifying the element in array*

print(array1d)

print("---------Slicing on 2D----------")

array2d **=** np**.**array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(array2d)

print(array2d[:, 0:2]) *# 2nd and 3rd col*

print(array2d[1:3, 0:3]) *# 2nd and 3rd row*

print(array2d[**-**1::**-**1, **-**1::**-**1]) *# Reverse an array*

print(array2d[0:2, 0:2:2]) *#start,stop,step*

print(array2d[1,0:2:2])

print("---------Slicing on 3D----------")

array3d **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(array3d)

print(array3d[0,1,2])

print(array3d[1,**-**1,2])

print(array3d[1,0:1,0:2])

print(array3d[0:2,0:2,0:2])

print(array3d[0:2,1,**-**3])

print("---------Splitting operations----------")

print("---------Splitting on 1D----------")

print(array1d)

print("-------splitting on Horizontal---------")

print(np**.**hsplit(array1d,2))*#split divides into equal division*

*#or*

np**.**split(array1d,2)

*#np.split(array1d,3) error*

print(np**.**array\_split(array1d,3)

*#splitting on vertical on 1D not possible it requires 2D or More Dimensional*

print("---------Splitting on 2D----------")

print(array2d)

print("---------splitting on horizontal on 2D (column-wise)-----")

print(np**.**hsplit(array2d,3))*#split divides into equal division*

*#or*

print(np**.**split(array2d,3,axis**=**1))

*#np.split(array2d,2) error use array\_split()*

print(np**.**array\_split(array2d,2,axis**=**1))

print(np**.**array\_split(array2d,4,axis**=**1))

print("-"**\***10)

print("---------splitting on vertical on 2D (row-wise)---------")

print(np**.**vsplit(array2d,3))*#split divides into equal division*

*#or*

print(np**.**split(array2d,3,axis**=**0))

print(np**.**array\_split(array2d,2,axis**=**0))

print(np**.**array\_split(array2d,4,axis**=**0))

print("--------------splitting on 3D--------------")

print(array3d)

print("-------------split on horizontal----------------")

print(np**.**hsplit(array3d,2))*#split divides into equal division*

*#or*

print(np**.**split(array3d,2,axis**=**1))

*#print(np.split(array3d,3,axis=1)) error not in equal division so go for array\_split()*

print(np**.**array\_split(array3d,3,axis**=**1))

print(np**.**array\_split(array3d,4,axis**=**1))

print("---------split on vertical------------ ")

print(np**.**vsplit(array3d,2))*#split divides into equal division*

*#or*

print(np**.**split(array3d,2,axis**=**0))

*#print(np.split(array3d,3,axis=1)) error not in equal division so go for array\_split()*

print(np**.**array\_split(array3d,3,axis**=**0))

print(np**.**array\_split(array3d,4,axis**=**0))

print("-----Iterating operations------")

print("-----Iterating Arrays Using nditer()--------")

print("-----Iterating operations on 1D------")

print(array1d)

**for** x **in** np**.**nditer(array1d,flags**=**['buffered'],op\_dtypes**=**['S']):

print(x)

print("-----Iterating operations on 2D------")

print(array2d)

**for** x **in** np**.**nditer(array2d[:,::2]):

print(x)

print("-----Iterating operations on 3D------")

print(array3d)

**for** x **in** np**.**nditer(array3d[:,::2,2]):

print(x)

print("---------modifying array values----------")

**for** x **in** np**.**nditer(array3d):

x**=**5**\***x

print(x)

print("----------using ndenumerate()--------")

for idx,x in np.ndenumerate(array3d):

print(idx,x)

Output:

[1 2 3 4 5 6 7 8]

---------Indexing operation ----------

---------Indexing on 1D----------

1

8

4

4

[1 3 2]

---------Indexing on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

1

2

3

2

8

8

[2 4 9]

[3 4 7]

8

---------Indexing on 3D----------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

6

2

3

[5 7 6]

[ 5 7 6 10]

16

---------Slicing operation----------

---------Slicing on 1D----------

[1 2 3 4 5 6 7 8]

[5 6 7 8]

[1 2 3 4]

[5 6 7]

[1 2 3 4 5 6 7]

[1 2 3 4 5 6]

[8 7 6 5 4 3 2 1]

[8 6 4 2]

[7 5 3 1]

[1 2 3 4 5 6 7 8]

[10 2 3 4 5 6 7 8]

---------Slicing on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

[[1 2]

[4 5]

[7 8]]

[[4 5 6]

[7 8 9]]

[[9 8 7]

[6 5 4]

[3 2 1]]

[[1]

[4]]

[4]

---------Slicing on 3D----------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

6

12

[[7 8]]

[[[ 1 2]

[ 4 5]]

[[ 7 8]

[10 11]]]

[ 4 10]

---------Splitting operations----------

---------Splitting on 1D----------

[10 2 3 4 5 6 7 8]

-------splitting on Horizontal---------

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

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

---------Splitting on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

---------splitting on horizontal on 2D (column-wise)-----

[array([[1],

[4],

[7]]), array([[2],

[5],

[8]]), array([[3],

[6],

[9]])]

[array([[1],

[4],

[7]]), array([[2],

[5],

[8]]), array([[3],

[6],

[9]])]

----------

---------splitting on vertical on 2D (row-wise)---------

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

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

[array([[1, 2, 3],

[4, 5, 6]]), array([[7, 8, 9]])]

[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]]), array([], shape=(0, 3), dtype=int32)]

--------------splitting on 3D--------------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

-------------split on horizontal----------------

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]])]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]])]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]]), array([], shape=(2, 0, 3), dtype=int32)]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]]), array([], shape=(2, 0, 3), dtype=int32), array([], shape=(2, 0, 3), dtype=int32)]

---------split on vertical------------

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]])]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]])]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]]), array([], shape=(0, 2, 3), dtype=int32)]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]]), array([], shape=(0, 2, 3), dtype=int32), array([], shape=(0, 2, 3), dtype=int32)]

-----Iterating operations------

-----Iterating Arrays Using nditer()--------

-----Iterating operations on 1D------

[10 2 3 4 5 6 7 8]

b'10'

b'2'

b'3'

b'4'

b'5'

b'6'

b'7'

b'8'

-----Iterating operations on 2D------

[[1 2 3]

[4 5 6]

[7 8 9]]

1

3

4

6

7

9

-----Iterating operations on 3D------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

3

9

---------modifying array values----------

5

10

15

20

25

30

35

40

45

50

55

60

----------using ndenumerate()--------

(0, 0, 0) 1

(0, 0, 1) 2

(0, 0, 2) 3

(0, 1, 0) 4

(0, 1, 1) 5

(0, 1, 2) 6

(1, 0, 0) 7

(1, 0, 1) 8

(1, 0, 2) 9

(1, 1, 0) 10

(1, 1, 1) 11

(1, 1, 2) 12

1.b) Write a python program on Indexing, Slicing, Splitting & Iterating on Dataset.

DATASET.csv file

65.78331 ,112.9925 ,345.89 ,234.23

71.51521 ,136.4873 ,346.89, 235.23

69.39874 ,153.0269 ,347.89 ,236.23

68.2166 ,142.3354 ,348.89, 237.23

67.78781 ,144.2971 ,349.89 ,238.23

68.69784 ,123.3024, 350.89, 239.23

69.80204 ,141.4947 ,351.89 ,240.23

70.01472, 136.4623 ,352.89, 241.23

67.90265 ,112.3723 ,353.89, 242.23

66.78236, 120.6672, 354.89 ,243.23

import numpy as np

dataset =np.genfromtxt('C:/Users/JOSHITH/Desktop/DATASET1.csv',delimiter=',')

print(dataset)

print("The rows & columns on dataset:", dataset.shape)

print("---------Perform Indexing operation on Dataset------ ")

print(dataset[0]) # index single element in outermost dimension

print(dataset[-1]) # index in reversed order in outermost dimension

print(dataset[1, 1]) # index single element in two-dimensional data

print(dataset[-1, -1]) # index in reversed order in two-dimensional data

print(dataset[8,3])

print(dataset[[0,7,3,9],[1,-1,2,3]])

print(dataset[0,3]+dataset[2,-1])

print(dataset[dataset>70])#booloean operation

print("---------Perform Slicing operation on Dataset------ ")

print(dataset[1:3]) # rows 1 and 2

print(dataset[:2, :2]) # 2x2 subset of the data

print(dataset[-1, ::-1]) # last row with elements reversed

print(dataset[-5:-1, :6:2])# last 4 rows,every other element up to index 6

print("---------Perform Splitting operation on Dataset------ ")

print("-----------Horizontal splitting-----------")

print(np.hsplit(dataset,2)) # split horizontally in 3 equal lists

print("-----------vertical splitting-----------")

print(np.vsplit(dataset, 2)) # split vertically in 2 equal lists

print("---------Perform Iterating operation on Dataset------ ")

print("----------using nditer()--------")

for x in np.nditer(dataset):

print(x)

print("----------using ndenumerate()--------")

for idx,x in np.ndenumerate(dataset):

print(idx,x)

Output:

[[ 65.78331 112.9925 345.89 234.23 ]

[ 71.51521 136.4873 346.89 235.23 ]

[ 69.39874 153.0269 347.89 236.23 ]

[ 68.2166 142.3354 348.89 237.23 ]

[ 67.78781 144.2971 349.89 238.23 ]

[ 68.69784 123.3024 350.89 239.23 ]

[ 69.80204 141.4947 351.89 240.23 ]

[ 70.01472 136.4623 352.89 241.23 ]

[ 67.90265 112.3723 353.89 242.23 ]

[ 66.78236 120.6672 354.89 243.23 ]]

The rows & columns on dataset: (10, 4)

---------Perform Indexing operation on Dataset------

[ 65.78331 112.9925 345.89 234.23 ]

[ 66.78236 120.6672 354.89 243.23 ]

136.4873

243.23

242.23

[112.9925 241.23 348.89 243.23 ]

470.46

[112.9925 345.89 234.23 71.51521 136.4873 346.89 235.23

153.0269 347.89 236.23 142.3354 348.89 237.23 144.2971

349.89 238.23 123.3024 350.89 239.23 141.4947 351.89

240.23 70.01472 136.4623 352.89 241.23 112.3723 353.89

242.23 120.6672 354.89 243.23 ]

---------Perform Slicing operation on Dataset------

[[ 71.51521 136.4873 346.89 235.23 ]

[ 69.39874 153.0269 347.89 236.23 ]]

[[ 65.78331 112.9925 ]

[ 71.51521 136.4873 ]]

[243.23 354.89 120.6672 66.78236]

[[ 68.69784 350.89 ]

[ 69.80204 351.89 ]

[ 70.01472 352.89 ]

[ 67.90265 353.89 ]]

---------Perform Splitting operation on Dataset------

-----------Horizontal splitting-----------

[array([[ 65.78331, 112.9925 ],

[ 71.51521, 136.4873 ],

[ 69.39874, 153.0269 ],

[ 68.2166 , 142.3354 ],

[ 67.78781, 144.2971 ],

[ 68.69784, 123.3024 ],

[ 69.80204, 141.4947 ],

[ 70.01472, 136.4623 ],

[ 67.90265, 112.3723 ],

[ 66.78236, 120.6672 ]]), array([[345.89, 234.23],

[346.89, 235.23],

[347.89, 236.23],

[348.89, 237.23],

[349.89, 238.23],

[350.89, 239.23],

[351.89, 240.23],

[352.89, 241.23],

[353.89, 242.23],

[354.89, 243.23]])]

-----------vertical splitting-----------

[array([[ 65.78331, 112.9925 , 345.89 , 234.23 ],

[ 71.51521, 136.4873 , 346.89 , 235.23 ],

[ 69.39874, 153.0269 , 347.89 , 236.23 ],

[ 68.2166 , 142.3354 , 348.89 , 237.23 ],

[ 67.78781, 144.2971 , 349.89 , 238.23 ]]), array([[ 68.69784, 123.3024 , 350.89 , 239.23 ],

[ 69.80204, 141.4947 , 351.89 , 240.23 ],

[ 70.01472, 136.4623 , 352.89 , 241.23 ],

[ 67.90265, 112.3723 , 353.89 , 242.23 ],

[ 66.78236, 120.6672 , 354.89 , 243.23 ]])]

---------Perform Iterating operation on Dataset------

----------using nditer()--------

65.78331

112.9925

345.89

234.23

71.51521

136.4873

346.89

235.23

69.39874

153.0269

347.89

236.23

68.2166

142.3354

348.89

237.23

67.78781

144.2971

349.89

238.23

68.69784

123.3024

350.89

239.23

69.80204

141.4947

351.89

240.23

70.01472

136.4623

352.89

241.23

67.90265

112.3723

353.89

242.23

66.78236

120.6672

354.89

243.23

----------using ndenumerate()--------

(0, 0) 65.78331

(0, 1) 112.9925

(0, 2) 345.89

(0, 3) 234.23

(1, 0) 71.51521

(1, 1) 136.4873

(1, 2) 346.89

(1, 3) 235.23

(2, 0) 69.39874

(2, 1) 153.0269

(2, 2) 347.89

(2, 3) 236.23

(3, 0) 68.2166

(8, 3) 242.23

(9, 0) 66.78236

(9, 1) 120.6672

(9, 2) 354.89

(9, 3) 243.23

2.a) Write python programs to implement statistical functions like Mean, Median, Variance, and Standard Deviation using numpy.

Program:

import numpy as np

a = np.array([[2,10,20],[80,43,31],[22,43,10]])

print("The original array:\n")

print(a)

print("\nThe minimum element among the array:",np.amin(a))

print("The maximum element among the array:",np.amax(a))

print("\nThe minimum elements with axis=0 ",np.amin(a,0))

print("The maximum elements with axis=0 ",np.amax(a,0))

print("\nThe minimum elements with axis=1 ",np.amin(a,1))

print("The maximum element with axis=1 ",np.amax(a,1))

print("Array:\n",a)

print("Mean of array :",np.mean(a))

print("Mean of array along axis 0:",np.mean(a,0))

print("Mean of array along axis 1:",np.mean(a,1))

print("\nMedian of array:",np.median(a))

print("\nMedian of array along axis 0:",np.median(a,0))

print("\nMedian of array along axis 1:",np.median(a,1))

print("Average of array along axis 1:",np.average(a,1))

print("Variance of array :",np.var(a))

print("Variance of array with axis=0 :",np.var(a,0))

print("Variance of array with axis=1 :",np.var(a,1))

print("Standard Deviation of array :",np.std(a))

print("Standard Deviation of array with axis=0 :",np.std(a,0))

print("Standard Deviation of array with axis=1 :",np.std(a,1))

Output:

The original array:

[[ 2 10 20]

[80 43 31]

[22 43 10]]

The minimum element among the array: 2

The maximum element among the array: 80

The minimum elements with axis=0 [ 2 10 10]

The maximum elements with axis=0 [80 43 31]

The minimum elements with axis=1 [ 2 31 10]

The maximum element with axis=1 [20 80 43]

Array:

[[ 2 10 20]

[80 43 31]

[22 43 10]]

Mean of array : 29.0

Mean of array along axis 0: [34.66666667 32. 20.33333333]

Mean of array along axis 1: [10.66666667 51.33333333 25. ]

Median of array: 22.0

Median of array along axis 0: [22. 43. 20.]

Median of array along axis 1: [10. 43. 22.]

Average of array along axis 1: [10.66666667 51.33333333 25. ]

Variance of array : 508.6666666666667

Variance of array with axis=0 : [1094.22222222 242. 73.55555556]

Variance of array with axis=1 : [ 54.22222222 434.88888889 186. ]

Standard Deviation of array : 22.55363976538303

Standard Deviation of array with axis=0 : [33.07902995 15.55634919 8.57645355]

Standard Deviation of array with axis=1 : [ 7.36357401 20.85398976 13.6381817 ]

2.b) Write python programs to implement statistical functions like Mean, Median, Variance, and Standard Deviation using numpy on dataset.

Program:

import numpy as np

dataset =np.genfromtxt('C:/Users/pc/Desktop/dataset.csv',delimiter=',')

print("The original array:\n")

print(dataset)

print("The rows & columns on dataset:", dataset.shape)

print("\nThe minimum element among the array:",np.amin(dataset))

print("The maximum element among the array:",np.amax(dataset))

print("\nThe minimum elements with axis=0 ",np.amin(dataset,0))

print("The maximum elements with axis=0 ",np.amax(dataset,0))

print("\nThe minimum elements with axis=1 ",np.amin(dataset,1))

print("The maximum element with axis=1 ",np.amax(dataset,1))

print("Mean of array :",np.mean(dataset))

print("Mean of array along axis 0:",np.mean(dataset,0))

print("Mean of array along axis 1:",np.mean(dataset,1))

print("\nMedian of array:",np.median(dataset))

print("\nMedian of array along axis 0:",np.median(dataset,0))

print("\nMedian of array along axis 1:",np.median(dataset,1))

print("Average of array along axis 1:",np.average(dataset,1))

print("Variance of array :",np.var(dataset))

print("Variance of array with axis=0 :",np.var(dataset,0))

print("Variance of array with axis=1 :",np.var(dataset,1))

print("Standard Deviation of array :",np.std(dataset))

print("Standard Deviation of array with axis=0 :",np.std(dataset,0))

print("Standard Deviation of array with axis=1 :",np.std(dataset,1))

Output:

The original array:

[[ 65.78331 112.9925 345.89 234.23 ]

[ 71.51521 136.4873 346.89 235.23 ]

[ 69.39874 153.0269 347.89 236.23 ]

[ 68.2166 142.3354 348.89 237.23 ]

[ 67.78781 144.2971 349.89 238.23 ]

[ 68.69784 123.3024 350.89 239.23 ]

[ 69.80204 141.4947 351.89 240.23 ]

[ 70.01472 136.4623 352.89 241.23 ]

[ 67.90265 112.3723 353.89 242.23 ]

[ 66.78236 120.6672 354.89 243.23 ]]

The rows & columns on dataset: (10, 4)

The minimum element among the array: 65.78331

The maximum element among the array: 354.89

The minimum elements with axis=0 [ 65.78331 112.3723 345.89 234.23 ]

The maximum elements with axis=0 [ 71.51521 153.0269 354.89 243.23 ]

The minimum elements with axis=1 [65.78331 71.51521 69.39874 68.2166 67.78781 68.69784 69.80204 70.01472

67.90265 66.78236]

The maximum element with axis=1 [345.89 346.89 347.89 348.89 349.89 350.89 351.89 352.89 353.89 354.89]

Mean of array : 197.51348449999995

Mean of array along axis 0: [ 68.590128 132.34381 350.39 238.73 ]

Mean of array along axis 1: [189.7239525 197.5306275 201.63641 199.168 200.0512275 195.53006

200.854185 200.149255 194.0987375 196.39239 ]

Median of array: 193.62845

Median of array along axis 0: [ 68.45722 136.4748 350.39 238.73 ]

Median of array along axis 1: [173.61125 185.85865 194.62845 189.7827 191.26355 181.2662 190.86235

188.84615 177.30115 181.9486 ]

Average of array along axis 1: [189.7239525 197.5306275 201.63641 199.168 200.0512275 195.53006

200.854185 200.149255 194.0987375 196.39239 ]

Variance of array : 11533.83509367655

Variance of array with axis=0 : [ 2.51218688 177.97972078 8.25 8.25 ]

Variance of array with axis=1 : [11904.40533048 10833.90953011 10609.12872161 11060.90167818

11127.85268899 11837.410653 11265.12134705 11502.09705744

12613.63227198 12462.07959084]

Standard Deviation of array : 107.3956940183197

Standard Deviation of array with axis=0 : [ 1.58498797 13.34090405 2.87228132 2.87228132]

Standard Deviation of array with axis=1 : [109.10731108 104.0860679 103.00062486 105.17082142 105.48863772

108.79986513 106.13727595 107.24783008 112.31042815 111.63368484]

3. Write python programs to implement Filtering, Sorting, Combining (vstack), Reshaping operations using numpy.

**import** numpy **as** np

arr**=**np**.**array([[1,2,3],[5,2,3],[6,4,9],[9,11,5]])

print('Our array is:')

print(arr)

print('\n')

print("-----1.Performing Filtering operations in Numpy-------")

print("----a)Based on Boolean index list-----------")

x**=**[**False**,**True**,**True**,**False**]

print("The boolean index list:",arr[x])

print('\n')

print("-----b)Based on Conditions,filtering the array-------")

print("arr[arr>5]:",arr[arr**>**5])

print("arr[arr==2]:",arr[arr**==**2])

print("arr[(arr>2)&(arr<10)]:",arr[(arr**>**2)**&**(arr**<**10)])

print("arr[(arr>3|arr<10)]:",arr[(arr**>**3)**|**(arr**<**10)])

print('\n')

print("----c)Using where(),filtering the array-------")

b**=**np**.**where(arr**>**2)

print("The array is:",b)

print('\n')

c**=**np**.**where((arr**<**4)**|**(arr**>**3))

print("where((arr<4)|(arr>3)):",c)

print('\n')

d**=**np**.**where((arr**>**2)**&**(arr**<**5))

print("where((arr>2)&(arr<5)):",d)

print('\n')

print("----d)Using extract(),filtering the array-------")

a1**=**np**.**extract((arr**>**5),arr)

print("extract((arr>5),arr)):",a1)

print('\n')

a2**=**np**.**extract((arr**<**4)**|**(arr**>**3),arr)

print("extract((arr<4)|(arr>3),arr):",a2)

print('\n')

a3**=**np**.**extract((arr**<**4)**&**(arr**>**3),arr)

print("extract((arr<4)&(arr>3),arr):",a2)

print('\n')

print("-----2.Performing Sorting operations in Numpy-------")

b1**=**np**.**sort(arr)

print("The sorted array is:\n",b1)

b2**=**np**.**sort(arr,axis**=**0)

print("Sort along axis=0:\n",b2)

b3**=**np**.**sort(arr,axis**=**1)

print("Sort along axis=1:\n",b3)

b4**=**np**.**sort(arr,axis**=None**)

print("Sort along axis=None:\n",b4)

print('\n')

print("-----Applying argmax() function--------")

print(" The index value of maximum element:",np**.**argmax(arr))

print("Indices of maximum along axis=0:",np**.**argmax(arr,axis**=**0))

print("Indices of maximum along axis=1:",np**.**argmax(arr,axis**=**1))

print('\n')

print("-----Applying argmin() function--------")

print(" The index value of minimum element:",np**.**argmin(arr))

print("Indices of minimum along axis=0:",np**.**argmin(arr,axis**=**0))

print("Indices of minimum along axis=1:",np**.**argmin(arr,axis**=**1))

print('\n')

print("-----Applying argsort() function--------")

print("Indices of sorted elements:\n",np**.**argsort(arr))

print("Indices of sorted elements along axis=0:\n",np**.**argsort(arr,axis**=**0))

print("Indices of sorted elements along axis=1:\n",np**.**argsort(arr,axis**=**1))

print('\n')

print("-----3.Applying vstack,hstack function--------")

arr1**=**np**.**array([[0,1],[2,3]])

print("First array:\n",arr1)

arr2**=**np**.**array([[4,5],[6,7]])

print("Second array:\n",arr2)

print("The Horizontal Stack:\n",np**.**hstack((arr1,arr2)))

print("The Vertical Stack:\n",np**.**vstack((arr1,arr2)))

print('\n')

print("----4.Performing Reshaping operation on the array------")

print("The array is :\n",arr)

res **=** np**.**reshape(arr, (2, 6))

print("Reshaping the original array with 2 rows, 6 columns:\n",res)

res1 **=** np**.**reshape(arr, (2,3,2))

print("Reshaping the original array with 2 matrix,3 rows, 2 columns:\n ",res1)

Output:

Our array is:

[[ 1 2 3]

[ 5 2 3]

[ 6 4 9]

[ 9 11 5]]

-----1.Performing Filtering operations in Numpy-------

----Based on Boolean index list-----------

The boolean index list: [[5 2 3]

[6 4 9]]

-----Based on Conditions,filtering the array-------

arr[arr>5]: [ 6 9 9 11]

arr[arr==2]: [2 2]

arr[(arr>2)&(arr<10)]: [3 5 3 6 4 9 9 5]

arr[(arr>3|arr<10)]: [ 1 2 3 5 2 3 6 4 9 9 11 5]

----Using where(),filtering the array-------

The array is: (array([0, 1, 1, 2, 2, 2, 3, 3, 3], dtype=int64), array([2, 0, 2, 0, 1, 2, 0, 1, 2], dtype=int64))

where((arr<4)|(arr>3)): (array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3], dtype=int64), array([0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2], dtype=int64))

where((arr>2)&(arr<5)): (array([0, 1, 2], dtype=int64), array([2, 2, 1], dtype=int64))

----Using extract(),filtering the array-------

extract((arr>5),arr)): [ 6 9 9 11]

extract((arr<4)|(arr>3),arr): [ 1 2 3 5 2 3 6 4 9 9 11 5]

extract((arr<4)&(arr>3),arr): [ 1 2 3 5 2 3 6 4 9 9 11 5]

-----2.Performing Sorting operations in Numpy-------

The sorted array is:

[[ 1 2 3]

[ 2 3 5]

[ 4 6 9]

[ 5 9 11]]

Sort along axis=0:

[[ 1 2 3]

[ 5 2 3]

[ 6 4 5]

[ 9 11 9]]

Sort along axis=1:

[[ 1 2 3]

[ 2 3 5]

[ 4 6 9]

[ 5 9 11]]

Sort along axis=None:

[ 1 2 2 3 3 4 5 5 6 9 9 11]

-----Applying argmax() function--------

The index value of maximum element: 10

Indices of maximum along axis=0: [3 3 2]

Indices of maximum along axis=1: [2 0 2 1]

-----Applying argmin() function--------

The index value of minimum element: 0

Indices of minimum along axis=0: [0 0 0]

Indices of minimum along axis=1: [0 1 1 2]

-----Applying argsort() function--------

Indices of sorted elements:

[[0 1 2]

[1 2 0]

[1 0 2]

[2 0 1]]

Indices of sorted elements along axis=0:

[[0 0 0]

[1 1 1]

[2 2 3]

[3 3 2]]

Indices of sorted elements along axis=1:

[[0 1 2]

[1 2 0]

[1 0 2]

[2 0 1]]

-----3.Applying vstack,hstack function--------

First array:

[[0 1]

[2 3]]

Second array:

[[4 5]

[6 7]]

The Horizontal Stack:

[[0 1 4 5]

[2 3 6 7]]

The Vertical Stack:

[[0 1]

[2 3]

[4 5]

[6 7]]

----4.Performing Reshaping operation on the array------

The array is :

[[ 1 2 3]

[ 5 2 3]

[ 6 4 9]

[ 9 11 5]]

Reshaping the original array with 2 rows, 6 columns:

[[ 1 2 3 5 2 3]

[ 6 4 9 9 11 5]]

Reshaping the original array with 2 matrix,3 rows, 2 columns:

[[[ 1 2]

[ 3 5]

[ 2 3]]

[[ 6 4]

[ 9 9]

[11 5]]]

4. Write python programs to implement Indexing, Slicing, Iterating using Pandas.

**nda.csv**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Name | Team | Number | Position | Age | Height | Weight | College | Salary | | Avery Bradley | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 | | Jae Crowder | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 | | John Holland | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 | | R.J. Hunter | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 | | Jonas Jerebko | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 | | Amir Johnson | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 | | Jordan Mickey | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 | | Kelly Olynyk | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 | | Terry Rozier | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 | | Marcus Smart | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**Program:**

import pandas as pd

d=pd.read\_csv("C:\\Users\\pc\\Desktop\\nda.csv",index\_col ="Name")

data=pd.DataFrame(d)

print(data)

Team Number Position Age Height Weight \

Name

Avery Bradley Boston Celtics 0 PG 25 06-Feb 180

Jae Crowder Boston Celtics 99 SF 25 06-Jun 235

John Holland Boston Celtics 30 SG 27 06-May 205

R.J. Hunter Boston Celtics 28 SG 22 06-May 185

Jonas Jerebko Boston Celtics 8 PF 29 06-Oct 231

Amir Johnson Boston Celtics 90 PF 29 06-Sep 240

Jordan Mickey Boston Celtics 55 PF 21 06-Aug 235

Kelly Olynyk Boston Celtics 41 C 25 06-08-2022 238

Terry Rozier Boston Celtics 12 PG 22 06-Feb 190

Marcus Smart Boston Celtics 36 PG 22 06-Apr 220

College Salary

Name

Avery Bradley Texas 7730337

Jae Crowder Marquette 6796117

John Holland Boston University 54545

R.J. Hunter Georgia State 1148640

Jonas Jerebko Texas 5000000

Amir Johnson LSU 12000000

Jordan Mickey LSU 1170960

Kelly Olynyk Gonzaga 2165160

Terry Rozier Louisville 1824360

Marcus Smart Oklahoma State 3431040

**data.head()**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |

**data.tail()**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
|  | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**data.describe()**

| **Number** | **Age** | **Weight** | **Salary** |
| --- | --- | --- | --- |
| **count** | 10.000000 | 10.000000 | 10.000000 | 1.000000e+01 |
| **mean** | 39.900000 | 24.700000 | 215.900000 | 4.132116e+06 |
| **std** | 33.124177 | 2.945807 | 23.750789 | 3.751788e+06 |
| **min** | 0.000000 | 21.000000 | 180.000000 | 5.454500e+04 |
| **25%** | 16.000000 | 22.000000 | 193.750000 | 1.334310e+06 |
| **50%** | 33.000000 | 25.000000 | 225.500000 | 2.798100e+06 |
| **75%** | 51.500000 | 26.500000 | 235.000000 | 6.347088e+06 |
| **max** | 99.000000 | 29.000000 | 240.000000 | 1.200000e+0 |

**data.shape**

(10, 8)

**data.columns**

Index(['Team', 'Number', 'Position', 'Age', 'Height', 'Weight', 'College','Salary'],dtype='object')

**data.index**

Index(['Avery Bradley', 'Jae Crowder', 'John Holland', 'R.J. Hunter',

'Jonas Jerebko', 'Amir Johnson', 'Jordan Mickey', 'Kelly Olynyk','Terry Rozier', 'Marcus Smart'],dtype='object', name='Name')

**data.dtypes**

Team object

Number int64

Position object

Age int64

Height object

Weight int64

College object

Salary int64

dtype: object

**#Indexing Operations**

**print(data['Team'])**

Name

Avery Bradley Boston Celtics

Jae Crowder Boston Celtics

John Holland Boston Celtics

R.J. Hunter Boston Celtics

Jonas Jerebko Boston Celtics

Amir Johnson Boston Celtics

Jordan Mickey Boston Celtics

Kelly Olynyk Boston Celtics

Terry Rozier Boston Celtics

Marcus Smart Boston Celtics

Name: Team, dtype: object

**print(data['Number'])**

Name

Avery Bradley 0

Jae Crowder 99

John Holland 30

R.J. Hunter 28

Jonas Jerebko 8

Amir Johnson 90

Jordan Mickey 55

Kelly Olynyk 41

Terry Rozier 12

Marcus Smart 36

Name: Number, dtype: int64

**print(data['Number']>15)**

Name

Avery Bradley False

Jae Crowder True

John Holland True

R.J. Hunter True

Jonas Jerebko False

Amir Johnson True

Jordan Mickey True

Kelly Olynyk True

Terry Rozier False

Marcus Smart True

Name: Number, dtype: bool

**print(data[["Team","Number"]].head())**

Team Number

Name

Avery Bradley Boston Celtics 0

Jae Crowder Boston Celtics 99

John Holland Boston Celtics 30

R.J. Hunter Boston Celtics 28

Jonas Jerebko Boston Celtics 8

**print(data[["Team","Number"]].tail())**

Team Number

Name

Amir Johnson Boston Celtics 90

Jordan Mickey Boston Celtics 55

Kelly Olynyk Boston Celtics 41

Terry Rozier Boston Celtics 12

Marcus Smart Boston Celtics 36

**print(data.loc["Amir Johnson"])**

Team Boston Celtics

Number 90

Position PF

Age 29

Height 06-Sep

Weight 240

College LSU

Salary 12000000

Name: Amir Johnson, dtype: object

**data[["Position","Age"]].loc[["Amir Johnson"]]**

| **Position** | **Age** |
| --- | --- |
| **Name** |  |  |
| **Amir Johnson** | PF | 29 |

**data.loc["Jordan Mickey"]**

Team Boston Celtics

Number 55

Position PF

Age 21

Height 06-Aug

Weight 235

College LSU

Salary 1170960

Name: Jordan Mickey, dtype: object

**print(data[["Age","Team","Number","Position"]])**

Age Team Number Position

Name

Avery Bradley 25 Boston Celtics 0 PG

Jae Crowder 25 Boston Celtics 99 SF

John Holland 27 Boston Celtics 30 SG

R.J. Hunter 22 Boston Celtics 28 SG

Jonas Jerebko 29 Boston Celtics 8 PF

Amir Johnson 29 Boston Celtics 90 PF

Jordan Mickey 21 Boston Celtics 55 PF

Kelly Olynyk 25 Boston Celtics 41 C

Terry Rozier 22 Boston Celtics 12 PG

Marcus Smart 22 Boston Celtics 36 PG

**data.iloc[-1] # index the last row**

Team Boston Celtics

Number 36

Position PG

Age 22

Height 06-Apr

Weight 220

College Oklahoma State

Salary 3431040

Name: Marcus Smart, dtype: object

**data.iloc[1]**

Team Boston Celtics

Number 99

Position SF

Age 25

Height 06-Jun

Weight 235

College Marquette

Salary 6796117

Name: Jae Crowder, dtype: object

**data.iloc[3]**

Team Boston Celtics

Number 28

Position SG

Age 22

Height 06-May

Weight 185

College Georgia State

Salary 1148640

Name: R.J. Hunter, dtype: object

**data.iloc[[3,5,7]]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |

**data.iloc[[3,4],[1,2]]**

| **Number** | **Position** |
| --- | --- |
| **Name** |  |  |
| **R.J. Hunter** | 28 | SG |
| **Jonas Jerebko** | 8 | PF |

**data.loc[data["Number"]==28]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |

**data.loc[data['Position']=='SG']**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 114864 |

**print(data[["Age"]].loc[["Jae Crowder"]])**

Age

Name

Jae Crowder 25

**#Slicing operations**

**data.iloc[:,1:4]**

| **Number** | **Position** | **Age** |
| --- | --- | --- |
| **Name** |  |  |  |
| **Avery Bradley** | 0 | PG | 25 |
| **Jae Crowder** | 99 | SF | 25 |
| **John Holland** | 30 | SG | 27 |
| **R.J. Hunter** | 28 | SG | 22 |
| **Jonas Jerebko** | 8 | PF | 29 |
| **Amir Johnson** | 90 | PF | 29 |
| **Jordan Mickey** | 55 | PF | 21 |
| **Kelly Olynyk** | 41 | C | 25 |
| **Terry Rozier** | 12 | PG | 22 |
| **Marcus Smart** | 36 | PG | 22 |

**data[1:3]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |

**data[1:5:2]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |

**data.loc[["Jae Crowder", "R.J. Hunter"]]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |

**print(data[["Age","Team","Number","Position"]][1:6])**

Age Team Number Position

Name

Jae Crowder 25 Boston Celtics 99 SF

John Holland 27 Boston Celtics 30 SG

R.J. Hunter 22 Boston Celtics 28 SG

Jonas Jerebko 29 Boston Celtics 8 PF

Amir Johnson 29 Boston Celtics 90 PF

**print(data[["Age","Team","Number","Position"]][5:])**

Age Team Number Position

Name

Amir Johnson 29 Boston Celtics 90 PF

Jordan Mickey 21 Boston Celtics 55 PF

Kelly Olynyk 25 Boston Celtics 41 C

Terry Rozier 22 Boston Celtics 12 PG

Marcus Smart 22 Boston Celtics 36 PG

**data.iloc[1:10:2]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**data.loc[:,['Team','Number','Position']]**

| **Team** | **Number** | **Position** |
| --- | --- | --- |
| **Name** |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG |
| **Jae Crowder** | Boston Celtics | 99 | SF |
| **John Holland** | Boston Celtics | 30 | SG |
| **R.J. Hunter** | Boston Celtics | 28 | SG |
| **Jonas Jerebko** | Boston Celtics | 8 | PF |
| **Amir Johnson** | Boston Celtics | 90 | PF |
| **Jordan Mickey** | Boston Celtics | 55 | PF |
| **Kelly Olynyk** | Boston Celtics | 41 | C |
| **Terry Rozier** | Boston Celtics | 12 | PG |
| **Marcus Smart** | Boston Celtics | 36 | PG |

**data.iloc[5:10, [0, 1]]**

| **Team** | **Number** |
| --- | --- |
| **Name** |  |  |
| **Amir Johnson** | Boston Celtics | 90 |
| **Jordan Mickey** | Boston Celtics | 55 |
| **Kelly Olynyk** | Boston Celtics | 41 |
| **Terry Rozier** | Boston Celtics | 12 |
| **Marcus Smart** | Boston Celtics | 36 |

**data.iloc[0:10]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**data.loc[["Amir Johnson", "Avery Bradley"]]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |

**data.loc[["Avery Bradley", "Amir Johnson"]][["College", "Salary"]]**

| **College** | **Salary** |
| --- | --- |
| **Name** |  |  |
| **Avery Bradley** | Texas | 7730337 |
| **Amir Johnson** | LSU | 1200000 |

**#Iterating operations**

**# iterating the whole dataset**

**for index, row in data.iterrows():**

**print(index, row)**

Avery Bradley Team Boston Celtics

Number 0

Position PG

Age 25

Height 06-Feb

Weight 180

College Texas

Salary 7730337

Name: Avery Bradley, dtype: object

Jae Crowder Team Boston Celtics

Number 99

Position SF

Age 25

Height 06-Jun

Weight 235

College Marquette

Salary 6796117

Name: Jae Crowder, dtype: object

John Holland Team Boston Celtics

Number 30

Position SG

Age 27

Height 06-May

Weight 205

College Boston University

Salary 54545

Name: John Holland, dtype: object

R.J. Hunter Team Boston Celtics

Number 28

Position SG

Age 22

Height 06-May

Weight 185

College Georgia State

Salary 1148640

Name: R.J. Hunter, dtype: object

Jonas Jerebko Team Boston Celtics

Number 8

Position PF

Age 29

Height 06-Oct

Weight 231

College Texas

Salary 5000000

Name: Jonas Jerebko, dtype: object

Amir Johnson Team Boston Celtics

Number 90

Position PF

Age 29

Height 06-Sep

Weight 240

College LSU

Salary 12000000

Name: Amir Johnson, dtype: object

Jordan Mickey Team Boston Celtics

Number 55

Position PF

Age 21

Height 06-Aug

Weight 235

College LSU

Salary 1170960

Name: Jordan Mickey, dtype: object

Kelly Olynyk Team Boston Celtics

Number 41

Position C

Age 25

Height 06-08-2022

Weight 238

College Gonzaga

Salary 2165160

Name: Kelly Olynyk, dtype: object

Terry Rozier Team Boston Celtics

Number 12

Position PG

Age 22

Height 06-Feb

Weight 190

College Louisville

Salary 1824360

Name: Terry Rozier, dtype: object

Marcus Smart Team Boston Celtics

Number 36

Position PG

Age 22

Height 06-Apr

Weight 220

College Oklahoma State

Salary 3431040

Name: Marcus Smart, dtype: object

**for row in data.loc[:, ['Number', 'Age', 'Position']].itertuples():print(row)**

Pandas(Index='Avery Bradley', Number=0, Age=25, Position='PG')

Pandas(Index='Jae Crowder', Number=99, Age=25, Position='SF')

Pandas(Index='John Holland', Number=30, Age=27, Position='SG')

Pandas(Index='R.J. Hunter', Number=28, Age=22, Position='SG')

Pandas(Index='Jonas Jerebko', Number=8, Age=29, Position='PF')

Pandas(Index='Amir Johnson', Number=90, Age=29, Position='PF')

Pandas(Index='Jordan Mickey', Number=55, Age=21, Position='PF')

Pandas(Index='Kelly Olynyk', Number=41, Age=25, Position='C')

Pandas(Index='Terry Rozier', Number=12, Age=22, Position='PG')

Pandas(Index='Marcus Smart', Number=36, Age=22, Position='PG')

**5. Write python programs to compute statistics using pandas(use Dictionary method).**

**Program**

**import pandas as pd**

**df = pd.DataFrame({'player': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'],**

**'points': [25, 34, 15, 14, 19, 23, 25, 29],**

**'assists': [5, 7, 7, 9, 12, 9, 9, 4],**

**'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})**

**#view DataFrame**

**df**

| **player** | **points** | **assists** | **rebounds** |
| --- | --- | --- | --- |
| **0** | A | 25 | 5 | 11 |
| **1** | B | 34 | 7 | 8 |
| **2** | C | 15 | 7 | 10 |
| **3** | D | 14 | 9 | 6 |
| **4** | E | 19 | 12 | 6 |
| **5** | F | 23 | 9 | 5 |
| **6** | G | 25 | 9 | 9 |
| **7** | H | 29 | 4 | 12 |

**#calculate the mean**

**df.mean()**

points 23.000

assists 7.750

rebounds 8.375

dtype: float64

**df.mean(axis=0)**

points 23.000

assists 7.750

rebounds 8.375

dtype: float64

**df.mean(axis=1)**

0 13.666667

1 16.333333

2 10.666667

3 9.666667

4 12.333333

5 12.333333

6 14.333333

7 15.000000

dtype: float64

**df["points"].mean()**

23.0

**df["points"].mean(axis=0)**

23.0

**# df["points"].mean(axis=1) –-error(not possible to calculate mean with one value)**

**df[["points","rebounds"]].mean()**

points 23.000

rebounds 8.375

dtype: float64

**df[["points",'rebounds']].mean(axis=1)**

0 18.0

1 21.0

2 12.5

3 10.0

4 12.5

5 14.0

6 17.0

7 20.5

dtype: float64

**df[["points",'rebounds']].mean(axis=0)**

points 23.000

rebounds 8.375

dtype: float64

**#calculate the median**

**df.median()**

points 24.0

assists 8.0

rebounds 8.5

dtype: float64

**df.median(axis=0)**

points 24.0

assists 8.0

rebounds 8.5

dtype: float64

**df.median(axis=1)**

0 11.0

1 8.0

2 10.0

3 9.0

4 12.0

5 9.0

6 9.0

7 12.0

dtype: float64

**df["points"].median()**

24.0

**df["points"].median(axis=0)**

24.0

**# df["points"].median(axis=1)-----error**

**df[["points","rebounds"]].median(axis=0)**

points 24.0

rebounds 8.5

dtype: float64

**df[["points","rebounds"]].median(axis=1)**

0 18.0

1 21.0

2 12.5

3 10.0

4 12.5

5 14.0

6 17.0

7 20.5

dtype: float64

**#Calculate min & max**

**df.min()**

player A

points 14

assists 4

rebounds 5

dtype: object

**df.max()**

player H

points 34

assists 12

rebounds 12

dtype: object

**df["points"].min()**

14

**df["points"].max()**

34

**df["points"].min(axis=0)**

14

**#df["points"].min(axis=1)---error**

**df[["points","rebounds"]].min(axis=0)**

points 14

rebounds 5

dtype: int64

**df[["points","rebounds"]].min(axis=1)**

0 11

1 8

2 10

3 6

4 6

5 5

6 9

7 12

dtype: int64

**#calculate the variance**

**df.var()**

points 46.571429

assists 6.500000

rebounds 6.553571

dtype: float64

**df["points"].var()**

46.57142857142857

**df["points"].var(axis=0)**

46.57142857142857

**#df["points"].var(axis=1)—error**

**df[["points","rebounds"]].var(axis=0)**

points 46.571429

rebounds 6.553571

dtype: float64

**df[["points","rebounds"]].var(axis=1)**

0 98.0

1 338.0

2 12.5

3 32.0

4 84.5

5 162.0

6 128.0

7 144.5

dtype: float64

**#calculate Standard Deviation**

**df.std()**

points 6.824326

assists 2.549510

rebounds 2.559994

dtype: float64

**df["points"].std()**

6.824326235712107

**df["points"].std(axis=0)**

6.824326235712107

**#df["points"].std(axis=1)---error**

**df[["points","rebounds"]].var(axis=0)**

points 46.571429

rebounds 6.553571

dtype: float64

**df[["points","rebounds"]].var(axis=1)**

0 98.0

1 338.0

2 12.5

3 32.0

4 84.5

5 162.0

6 128.0

7 144.5

dtype: float64

6.Write python programs to implement Filtering, Sorting, Reshaping operations using Pandas.

**nda.csv**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
| Avery Bradley | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| Jae Crowder | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| John Holland | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| R.J. Hunter | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| Jonas Jerebko | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| Amir Johnson | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| Jordan Mickey | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| Kelly Olynyk | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| Terry Rozier | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| Marcus Smart | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**Program:**

**import pandas as pd**

**d=pd.read\_csv("C:\\Users\\pc\\Desktop\\nda.csv",index\_col="Name")**

**data=pd.DataFrame(d)**

**print(data)**

Team Number Position Age Height Weight \

Name

Avery Bradley Boston Celtics 0 PG 25 06-Feb 180

Jae Crowder Boston Celtics 99 SF 25 06-Jun 235

John Holland Boston Celtics 30 SG 27 06-May 205

R.J. Hunter Boston Celtics 28 SG 22 06-May 185

Jonas Jerebko Boston Celtics 8 PF 29 06-Oct 231

Amir Johnson Boston Celtics 90 PF 29 06-Sep 240

Jordan Mickey Boston Celtics 55 PF 21 06-Aug 235

Kelly Olynyk Boston Celtics 41 C 25 06-08-2022 238

Terry Rozier Boston Celtics 12 PG 22 06-Feb 190

Marcus Smart Boston Celtics 36 PG 22 06-Apr 220

College Salary

Name

Avery Bradley Texas 7730337

Jae Crowder Marquette 6796117

John Holland Boston University 54545

R.J. Hunter Georgia State 1148640

Jonas Jerebko Texas 5000000

Amir Johnson LSU 12000000

Jordan Mickey LSU 1170960

Kelly Olynyk Gonzaga 2165160

Terry Rozier Louisville 1824360

Marcus Smart Oklahoma State 3431040

**data.head()**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |

**data.tail()**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**#filtering operations**

**data.filter(items=["Number"]) # # select columns by name**

| **Number** |
| --- |
| **Name** |  |
| **Avery Bradley** | 0 |
| **Jae Crowder** | 99 |
| **John Holland** | 30 |
| **R.J. Hunter** | 28 |
| **Jonas Jerebko** | 8 |
| **Amir Johnson** | 90 |
| **Jordan Mickey** | 55 |
| **Kelly Olynyk** | 41 |
| **Terry Rozier** | 12 |
| **Marcus Smart** | 36 |

**data.filter(items=["Number","Age"]) # only column**

| **Number** | **Age** |
| --- | --- |
| **Name** |  |  |
| **Avery Bradley** | 0 | 25 |
| **Jae Crowder** | 99 | 25 |
| **John Holland** | 30 | 27 |
| **R.J. Hunter** | 28 | 22 |
| **Jonas Jerebko** | 8 | 29 |
| **Amir Johnson** | 90 | 29 |
| **Jordan Mickey** | 55 | 21 |
| **Kelly Olynyk** | 41 | 25 |
| **Terry Rozier** | 12 | 22 |
| **Marcus Smart** | 36 | 22 |

**data.filter(items=["Number"] ,axis=0)**

TeamNumber Position Age Height Weight College Salary

Name

**data.filter(items=["Number"] ,axis=1)**

| **Number** |
| --- |
| **Name** |  |
| **Avery Bradley** | 0 |
| **Jae Crowder** | 99 |
| **John Holland** | 30 |
| **R.J. Hunter** | 28 |
| **Jonas Jerebko** | 8 |
| **Amir Johnson** | 90 |
| **Jordan Mickey** | 55 |
| **Kelly Olynyk** | 41 |
| **Terry Rozier** | 12 |
| **Marcus Smart** | 36 |

**data.filter(items=["Position"])**

| **Position** |
| --- |
| **Name** |  |
| **Avery Bradley** | PG |
| **Jae Crowder** | SF |
| **John Holland** | SG |
| **R.J. Hunter** | SG |
| **Jonas Jerebko** | PF |
| **Amir Johnson** | PF |
| **Jordan Mickey** | PF |
| **Kelly Olynyk** | C |
| **Terry Rozier** | PG |
| **Marcus Smart** | PG |

**# applying filter function**

**data.filter([ "College", "Salary"])**

| **College** | **Salary** |
| --- | --- |
| **Name** |  |  |
| **Avery Bradley** | Texas | 7730337 |
| **Jae Crowder** | Marquette | 6796117 |
| **John Holland** | Boston University | 54545 |
| **R.J. Hunter** | Georgia State | 1148640 |
| **Jonas Jerebko** | Texas | 5000000 |
| **Amir Johnson** | LSU | 12000000 |
| **Jordan Mickey** | LSU | 1170960 |
| **Kelly Olynyk** | Gonzaga | 2165160 |
| **Terry Rozier** | Louisville | 1824360 |
| **Marcus Smart** | Oklahoma State | 3431040 |

**data.filter(regex ='[aA]')# columns which has letter 'a' or 'A' in its name.**

| **Team** | **Age** | **Salary** |
| --- | --- | --- |
| **Name** |  |  |  |
| **Avery Bradley** | Boston Celtics | 25 | 7730337 |
| **Jae Crowder** | Boston Celtics | 25 | 6796117 |
| **John Holland** | Boston Celtics | 27 | 54545 |
| **R.J. Hunter** | Boston Celtics | 22 | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 29 | 5000000 |
| **Amir Johnson** | Boston Celtics | 29 | 12000000 |
| **Jordan Mickey** | Boston Celtics | 21 | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 25 | 2165160 |
| **Terry Rozier** | Boston Celtics | 22 | 1824360 |
| **Marcus Smart** | Boston Celtics | 22 | 3431040 |

**#Filter a DataFrame for a single column value with a given condition**

**data[data['Number'] > 30]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**#Filter a DataFrame with multiple conditions**

**data[(data['Number'] > 10) & (data["Weight"] > 20)]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**#Filtering DataFrame based on Date value**

**data[data['Height'] >'06-Feb']**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |

**#Filtering DataFrame based on a specific string**

**data[data['Team'] == 'Boston Celtics']**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**#Filtering the DataFrame rows using regular expressions(REGEX)**

**data[data['Team'].str.contains('Celtics$')]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |

**data[(data["Number"] < 10)]**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |

**# select columns by regular expression**

**data.filter(regex='o$', axis=0)**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 500000 |

**data.filter(regex="e$", axis=1)#regex for columns**

| **Age** | **College** |
| --- | --- |
| **Name** |  |  |
| **Avery Bradley** | 25 | Texas |
| **Jae Crowder** | 25 | Marquette |
| **John Holland** | 27 | Boston University |
| **R.J. Hunter** | 22 | Georgia State |
| **Jonas Jerebko** | 29 | Texas |
| **Amir Johnson** | 29 | LSU |
| **Jordan Mickey** | 21 | LSU |
| **Kelly Olynyk** | 25 | Gonzaga |
| **Terry Rozier** | 22 | Louisville |
| **Marcus Smart** | 22 | Oklahoma State |

**# select rows containing 'Amir Johnson'**

**data.filter(like='Amir Johnson', axis=0)**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |

**#Sorting operations# Sorting by column 'Number'**

**data.sort\_values(by=["Number"])**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |

**# values sorted by descending**

**data.sort\_values(by=["Number"],ascending=False)**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |

**# Sorting by columns "Number" and then "Age"**

**data.sort\_values(by=['Number', 'Age'])**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |

**data.sort\_values(by ='Number', ascending = 0)**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |

**data.sort\_values(by = ['Number', 'Age'], ascending = [True, False])**

| **Team** | **Number** | **Position** | **Age** | **Height** | **Weight** | **College** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** |  |  |  |  |  |  |  |  |
| **Avery Bradley** | Boston Celtics | 0 | PG | 25 | 06-Feb | 180 | Texas | 7730337 |
| **Jonas Jerebko** | Boston Celtics | 8 | PF | 29 | 06-Oct | 231 | Texas | 5000000 |
| **Terry Rozier** | Boston Celtics | 12 | PG | 22 | 06-Feb | 190 | Louisville | 1824360 |
| **R.J. Hunter** | Boston Celtics | 28 | SG | 22 | 06-May | 185 | Georgia State | 1148640 |
| **John Holland** | Boston Celtics | 30 | SG | 27 | 06-May | 205 | Boston University | 54545 |
| **Marcus Smart** | Boston Celtics | 36 | PG | 22 | 06-Apr | 220 | Oklahoma State | 3431040 |
| **Kelly Olynyk** | Boston Celtics | 41 | C | 25 | 06-08-2022 | 238 | Gonzaga | 2165160 |
| **Jordan Mickey** | Boston Celtics | 55 | PF | 21 | 06-Aug | 235 | LSU | 1170960 |
| **Amir Johnson** | Boston Celtics | 90 | PF | 29 | 06-Sep | 240 | LSU | 12000000 |
| **Jae Crowder** | Boston Celtics | 99 | SF | 25 | 06-Jun | 235 | Marquette | 6796117 |

**#reshaping operation**

**a=pd.pivot\_table(data,index='Team', columns='Position')**

**a**

| **Age** | | | | | **Number** | | | | | **Salary** | | | | | **Weight** | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Position** | **C** | **PF** | **PG** | **SF** | **SG** | **C** | **PF** | **PG** | **SF** | **SG** | **C** | **PF** | **PG** | **SF** | **SG** | **C** | **PF** | **PG** | **SF** | **SG** |
| **Team** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Boston Celtics** | 25.0 | 26.333333 | 23.0 | 25.0 | 24.5 | 41 | 51 | 16 | 99 | 29 | 2165160.0 | 6.056987e+06 | 4328579.0 | 6796117.0 | 601592.5 | 238.0 | 235.333333 | 196.666667 | 235.0 | 195. |

**c=pd.pivot\_table(data,index=["Team"],columns="Age",values="Number")**

**c**

| **Age** | **21** | **22** | **25** | **27** | **29** |
| --- | --- | --- | --- | --- | --- |
| **Team** |  |  |  |  |  |
| **Boston Celtics** | 55.0 | 25.333333 | 46.666667 | 30.0 | 49.0 |

**7.** Write a python program for the following Data visualization techniques.

a) Line Chart

b) Bar chart

c) Pie chart

d)Scatter plot

e) Histogram

f) Box plot

g) Stacked Area Chart

**a) Line chart:**

#LINE GRAPH

import matplotlib.pyplot as plt

x=[10,20,30,40,50]

y=[65,98,170,220,310]

plt.plot(x,y,color='r',marker="D",mec="k",mfc="w",linestyle="dashed",

linewidth="2")

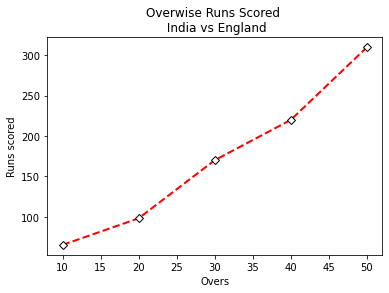
plt.xlabel("Overs")

plt.ylabel("Runs scored")

plt.title("Overwise Runs Scored \n India vs England")

plt.show()

**Output:**



**b)Bar Graph**

#Bar Graph

import matplotlib.pyplot as plt

import numpy as np

Overs=["1-10","11-20","21-30","31-40","41-50"]

Runs=[65,55,70,50,80]

plt.bar(Overs,Runs,color=["r","g","b","c","k"],width=[0.1,0.2,0.3,0.4,0.5])

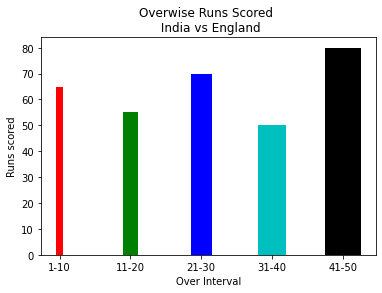
plt.xlabel("Over Interval")

plt.ylabel("Runs scored")

plt.title("Overwise Runs Scored \n India vs England")

plt.show()

**Output:**



**c) Pie chart**

#pie chart

import matplotlib.pyplot as plt

import numpy as np

branches=['cse','it','eee','ece','csebs']

data=[90,40,15,45,21]

fig=plt.figure(figsize=(9,10))

mycolors=['red','green','cyan','black','hotpink']

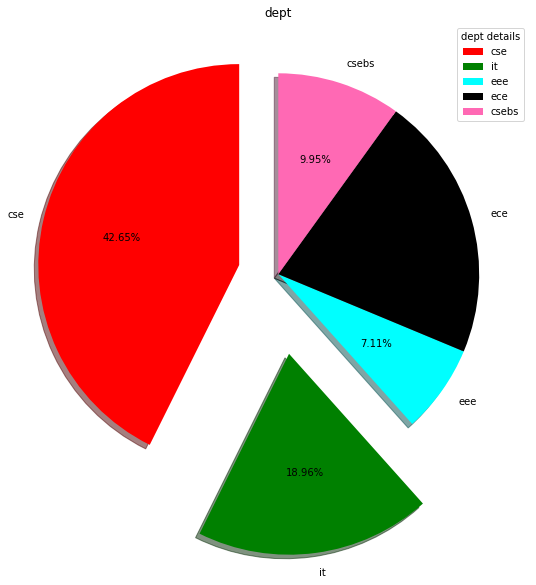
plt.pie(data,labels=branches,startangle=90,explode=[0.2,0.4,0,0,0],shadow=True,colors=mycolors,autopct="%0.2f%%")

plt.title("dept")

plt.legend(title="dept details",loc="upper right")

plt.show()

Output:



**d) Scatter plot**

#Scatter plot

import matplotlib.pyplot as plt

import numpy as np

x=[2,2.5,3,3.5,4.5,4.7,5.0]

y=[7.5,8,8.5,9,9.5,10,10.5]

x1=[9,8.5,9,9.5,10,10.5,12]

y1=[3,3.5,4.7,4,4.5,5,5.2]

plt.scatter(x,y,label="low savings high income",color='g',marker="D",linewidth=4,edgecolor="k")

plt.scatter(x1,y1,label="high savings low income",color='r',marker="s",linewidth=2,edgecolor="g")

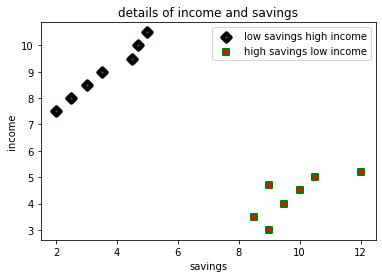
plt.xlabel("savings")

plt.ylabel("income")

plt.title("details of income and savings")

plt.legend()

plt.show()



**e) Histogram**

#Histogram

import matplotlib.pyplot as plt

age=[22,32,35,45,55,14,26,19,56,44,48,33,38,28]

years=[0,10,20,30,40,50,60]

plt.hist(age,bins=years,color='magenta',histtype='bar',edgecolor='black',rwidth=0.6)

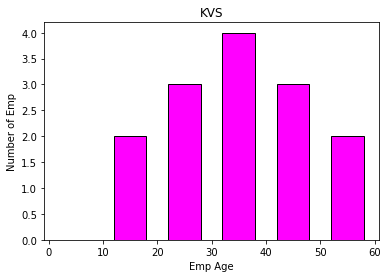
plt.xlabel("Emp Age")

plt.ylabel("Number of Emp")

plt.title("KVS")

plt.show()

Output:



**f)Box plot:**

#Box plot

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

value1 = [82,76,24,40,67,62,75,78,71,32,98,89,78,67,72,82,87,66,56,52]

value2=[62,5,91,25,36,32,96,95,3,90,95,32,27,55,100,15,71,11,37,21]

value3=[23,89,12,78,72,89,25,69,68,86,19,49,15,16,16,75,65,31,25,52]

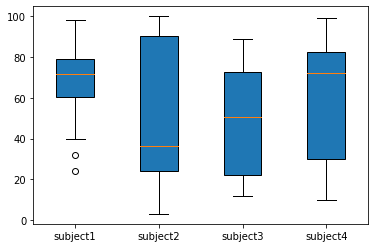
value4=[59,73,70,16,81,61,88,98,10,87,29,72,16,23,72,88,78,99,75,30]

box\_plot\_data=[value1,value2,value3,value4]

plt.boxplot(box\_plot\_data,patch\_artist=True,labels=['subject1','subject2','subject3','subject4'])

plt.show()

Output:



**g) Stacked Area Chart**

import numpy as np

import matplotlib.pyplot as plt

days = [1, 2, 3, 4, 5]

# No of Study Hours

Studying = [7, 8, 6, 11, 7]

# No of Playing Hours

playing = [8, 5, 7, 8, 13]

# Stackplot with X, Y, colors value

plt.stackplot(days, Studying, playing,

colors =['r', 'c'])

# Days

plt.xlabel('Days')

# No of hours

plt.ylabel('No of Hours')

# Title of Graph

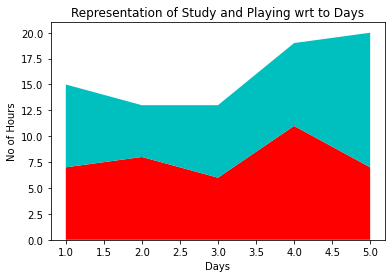
plt.title('Representation of Study and \

Playing wrt to Days')

# Displaying Graph

plt.show()

Output:



8.Write python programs to Create Layout and Subplots using matplotlib.

import matplotlib.pyplot as plt

import numpy as np

fig,a=plt.subplots(2,2,sharex=True,sharey=True)

x=np.arange(1,5)

a[0,0].plot(x,x\*x,color="black")

a[0,0].set\_title("Fig1")

a[0,0].set\_xlabel("X Label")

a[0,0].set\_ylabel("Y Label")

a[0,0].set\_facecolor("pink")

a[0,1].plot(x,x+x,color="black")

a[0,1].set\_title("Fig2")

a[0,1].set\_xlabel("X Label")

a[0,1].set\_ylabel("Y Label")

a[0,1].set\_facecolor("yellow")

a[1,0].plot(x,x+x+x,color="black")

a[1,0].set\_title("Fig3")

a[1,0].set\_xlabel("X Label")

a[1,0].set\_ylabel("Y Label")

a[1,0].set\_facecolor("magenta")

a[1,1].plot(x,x+x,color="black")

a[1,1].set\_title("Fig4")

a[1,1].set\_xlabel("X Label")

a[1,1].set\_ylabel("Y Label")

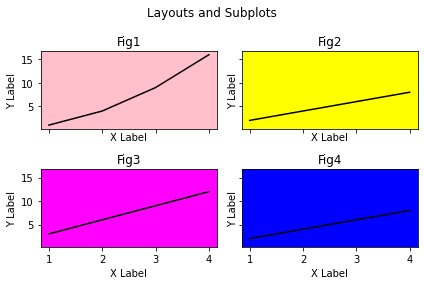
a[1,1].set\_facecolor("blue")

fig.suptitle("Layouts and Subplots")

plt.tight\_layout()

plt.show()

**Output:**



9.Write a Python programs to create Figure Aesthetics, Contexts, Color Palettes, using Pandas DataFrame in Seaborn (reading .csv file in Pandas DataFrame) for the following plots:

i)Line plot

ii)Scatter plot

**team.csv**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Team | Number | Position | Age | salary |
| Avery Bradley | A | 12 | PG | 25 | 2500 |
| Jae Crowder | B | 34 | SG | 25 | 4545 |
| John Holland | C | 12 | SG | 27 | 2323 |
| R.J. Hunter | D | 56 | SG | 22 | 2324 |
| Jonas Jerebko | E | 67 | PF | 29 | 1456 |
| Amir Johnson | F | 56 | PF | 29 | 2324 |
| Jordan Mickey | G | 78 | PF | 21 | 6754 |
| Kelly Olynyk | H | 34 | PG | 25 | 3421 |
| Terry Rozier | I | 67 | PG | 22 | 890 |
| Marcus Smart | J | 45 | PG | 22 | 5678 |

i)**Line plot**

**Program:**

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv("C:\\Users\\JOSHITH\\Desktop\\team.csv")

print(data)

# plotting a single line graph

sns.lineplot(x="Team", y="Number", data=data, hue="Position", palette="bright")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Line plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

Name Team Number Position Age salary

0 Avery Bradley A 12 PG 25 2500

1 Jae Crowder B 34 SG 25 4545

2 John Holland C 12 SG 27 2323

3 R.J. Hunter D 56 SG 22 2324

4 Jonas Jerebko E 67 PF 29 1456

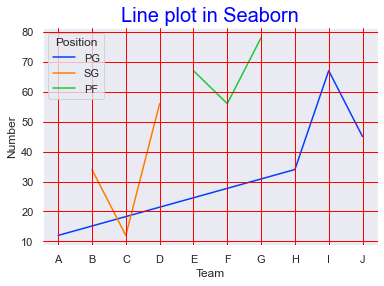
5 Amir Johnson F 56 PF 29 2324

6 Jordan Mickey G 78 PF 21 6754

7 Kelly Olynyk H 34 PG 25 3421

8 Terry Rozier I 67 PG 22 890

9 Marcus Smart J 45 PG 22 5678

****

****

**ii)Scatter Plot:**

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv("C:\\Users\\JOSHITH\\Desktop\\team.csv")

print(data)

sns.scatterplot(x="Team", y="Number", data=data, hue="Position", palette="bright")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Scatter plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

Name Team Number Position Age salary

0 Avery Bradley A 12 PG 25 2500

1 Jae Crowder B 34 SG 25 4545

2 John Holland C 12 SG 27 2323

3 R.J. Hunter D 56 SG 22 2324

4 Jonas Jerebko E 67 PF 29 1456

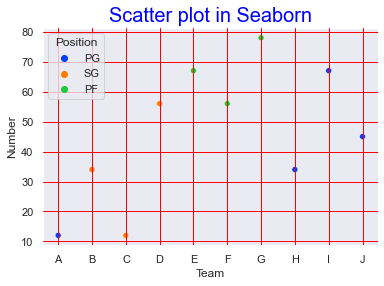
5 Amir Johnson F 56 PF 29 2324

6 Jordan Mickey G 78 PF 21 6754

7 Kelly Olynyk H 34 PG 25 3421

8 Terry Rozier I 67 PG 22 890

9 Marcus Smart J 45 PG 22 5678





10. Write a python programs to create Figure Aesthetics, Contexts, Color Palettes using Dictionaries in Pandas in Seaborn for the following plots:

i)Histogram

ii)Bar plot

**i)Histogram:**

**Program**

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

data = pd.DataFrame({'Name': ['Avery Bradley', 'Jae Crowder', 'John Holland', 'R.J. Hunter', 'Jonas Jerebko', 'Amir Johnson',

'Jordan Mickey', 'Kelly Olynyk','Terry Rozier','Marcus Smart'],

'Team': ['A','B','C','D','E','F','G','H','I','J'],

'Number':[12,34,12,56,67,56,78,34,67,45],

'Position':['PG','SG','SG','SG','PF','PF','PF','PG','PG','PG'],

'Age':[25,25,27,22,29,29,21,25,22,22],

'salary':[2500,4545,2323,2324,1456,2324,6754,3421,890,5698]

})

print(data)

sns.histplot(data = data, x = "Number", kde = True, hue = "Position", palette="Greens")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Histogram plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

Name Team Number Position Age salary

0 Avery Bradley A 12 PG 25 2500

1 Jae Crowder B 34 SG 25 4545

2 John Holland C 12 SG 27 2323

3 R.J. Hunter D 56 SG 22 2324

4 Jonas Jerebko E 67 PF 29 1456

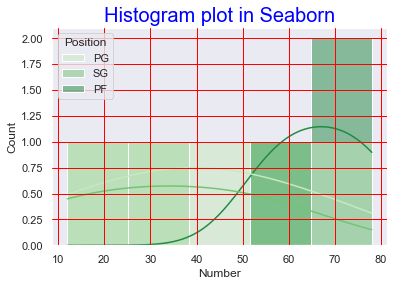
5 Amir Johnson F 56 PF 29 2324

6 Jordan Mickey G 78 PF 21 6754

7 Kelly Olynyk H 34 PG 25 3421

8 Terry Rozier I 67 PG 22 890

9 Marcus Smart J 45 PG 22 5678





**ii)Bar plot:**

**Program:**

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

data = pd.DataFrame({'Name': ['Avery Bradley', 'Jae Crowder', 'John Holland', 'R.J. Hunter', 'Jonas Jerebko', 'Amir Johnson',

'Jordan Mickey', 'Kelly Olynyk','Terry Rozier','Marcus Smart'],

'Team': ['A','B','C','D','E','F','G','H','I','J'],

'Number':[12,34,12,56,67,56,78,34,67,45],

'Position':['PG','SG','SG','SG','PF','PF','PF','PG','PG','PG'],

'Age':[25,25,27,22,29,29,21,25,22,22],

'salary':[2500,4545,2323,2324,1456,2324,6754,3421,890,5698]

})

print(data)

sns.barplot(x="Team", y="Number", data=data, hue="Position", palette="bright")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Bar plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

Name Team Number Position Age salary

0 Avery Bradley A 12 PG 25 2500

1 Jae Crowder B 34 SG 25 4545

2 John Holland C 12 SG 27 2323

3 R.J. Hunter D 56 SG 22 2324

4 Jonas Jerebko E 67 PF 29 1456

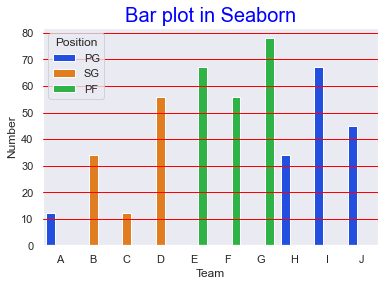
5 Amir Johnson F 56 PF 29 2324

6 Jordan Mickey G 78 PF 21 6754

7 Kelly Olynyk H 34 PG 25 3421

8 Terry Rozier I 67 PG 22 890

9 Marcus Smart J 45 PG 22 5698

****

****

11. Write python programs to create Figure Aesthetics, Contexts, Color Palettes, using Seaborn (loading dataset available in seaborn) for the following plots:

i)Box plot

ii)Histogram

**i)Box Plot:**

**Program:**

import numpy as np

import pandas as pd

import seaborn as sns

# Load dataset

data = sns.load\_dataset("fmri")

print(data)

sns.boxplot(x='timepoint', y='subject', data=data, palette="Greens")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Box plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

subject timepoint event region signal

0 s13 18 stim parietal -0.017552

1 s5 14 stim parietal -0.080883

2 s12 18 stim parietal -0.081033

3 s11 18 stim parietal -0.046134

4 s10 18 stim parietal -0.037970

... ... ... ... ... ...

1059 s0 8 cue frontal 0.018165

1060 s13 7 cue frontal -0.029130

1061 s12 7 cue frontal -0.004939

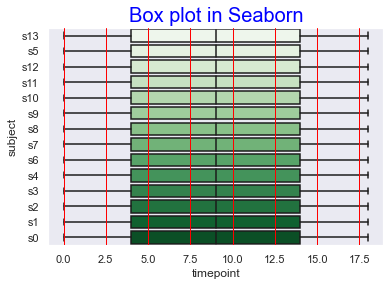
1062 s11 7 cue frontal -0.025367

1063 s0 0 cue parietal -0.006899

[1064 rows x 5 columns]

Out[94]:

<AxesSubplot:xlabel='timepoint', ylabel='subject'>

​ 



**ii)Histogram**

**Program:**

import numpy as np

import pandas as pd

import seaborn as sns

# Load dataset

dataset = sns.load\_dataset("fmri")

print(dataset)

sns.histplot(data = dataset, x = "timepoint", kde = True, hue = "region", palette="Greens")

sns.set\_style("dark",{'axes.axisbelow':False,'axes.grid':True,'grid.color':'red'})

sns.set\_context("notebook",font\_scale=1,rc={'figure.figsize':(8,5)})

plt.title("Histogram plot in Seaborn",color="blue",fontsize=20)

p=sns.color\_palette("deep")#deep,colorblind,dark,pastel,bright

sns.palplot(p)

plt.show()

**Output:**

subject timepoint event region signal

0 s13 18 stim parietal -0.017552

1 s5 14 stim parietal -0.080883

2 s12 18 stim parietal -0.081033

3 s11 18 stim parietal -0.046134

4 s10 18 stim parietal -0.037970

... ... ... ... ... ...

1059 s0 8 cue frontal 0.018165

1060 s13 7 cue frontal -0.029130

1061 s12 7 cue frontal -0.004939

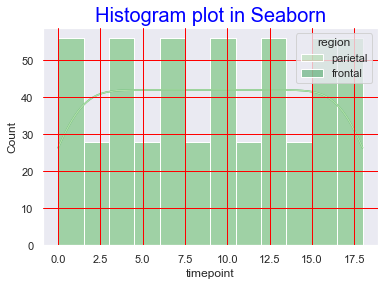
1062 s11 7 cue frontal -0.025367

1063 s0 0 cue parietal -0.006899

[1064 rows x 5 columns]

Out[87]:

<AxesSubplot:xlabel='timepoint', ylabel='Count'>



****

12.Write a Python program to implement the following Kernel Density Estimation

i. Univariate Distribution

ii. Bivariate Distribution.

1. **Univariate Distribution**

**Program:**

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

from warnings import filterwarnings

#sns.get\_dataset\_names()

df = sns.load\_dataset('fmri')

print(df)

sns.distplot(df['timepoint'],

kde = True,

color ='red',

bins = 30)

plt.show()

**Output:**

subject timepoint event region signal

0 s13 18 stim parietal -0.017552

1 s5 14 stim parietal -0.080883

2 s12 18 stim parietal -0.081033

3 s11 18 stim parietal -0.046134

4 s10 18 stim parietal -0.037970

... ... ... ... ... ...

1059 s0 8 cue frontal 0.018165

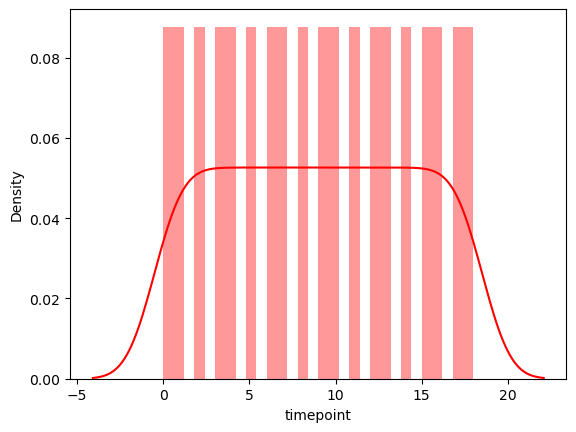
1060 s13 7 cue frontal -0.029130

1061 s12 7 cue frontal -0.004939

1062 s11 7 cue frontal -0.025367

1063 s0 0 cue parietal -0.006899

[1064 rows x 5 columns]



1. **Bivariate Distribution.**

**Program:**

import pandas as pd

import seaborn as sb

from matplotlib import pyplot as plt

df = sb.load\_dataset('iris')

print(df)

sb.jointplot(x = 'petal\_length',

y = 'petal\_width',

data = df,

kind="scatter",

height=6,

ratio=5,

space=0.2,

color='red')

plt.show()

Output:

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

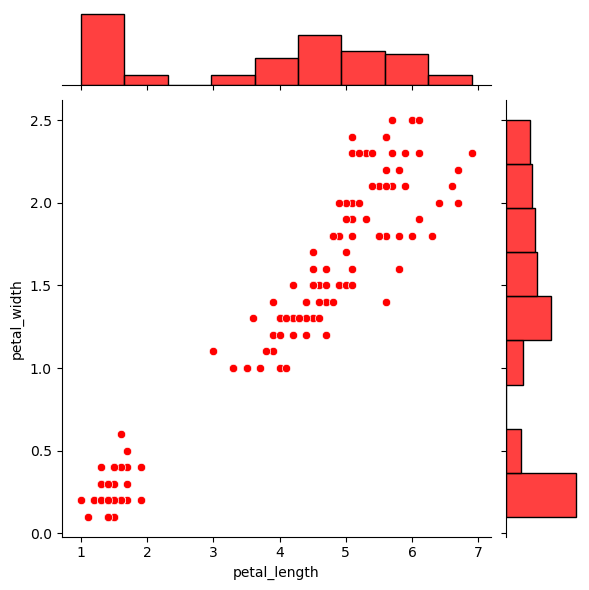
146 6.3 2.5 5.0 1.9 virginica

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 x 5 columns]



13.Write a python program to implement visualization on Pairwise relationship

**Program:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

mydata =pd.read\_csv("C:\\Users\\pc\\Desktop\\new.csv")

print(mydata)

sns.set(style="whitegrid", color\_codes=True)

sns.pairplot(data=mydata,

diag\_kind = "kde",

kind = "scatter",

markers = "s",

height = 2.5,

aspect = 1 )

plt.show()

**Output:**

Name Team Number Position Age salary

0 Avery Bradley A 12 PG 25 2500

1 Jae Crowder B 34 SF 25 4545

2 John Holland C 12 SG 27 2323

3 R.J. Hunter D 56 SG 22 2324

4 Jonas Jerebko E 67 PF 29 1456

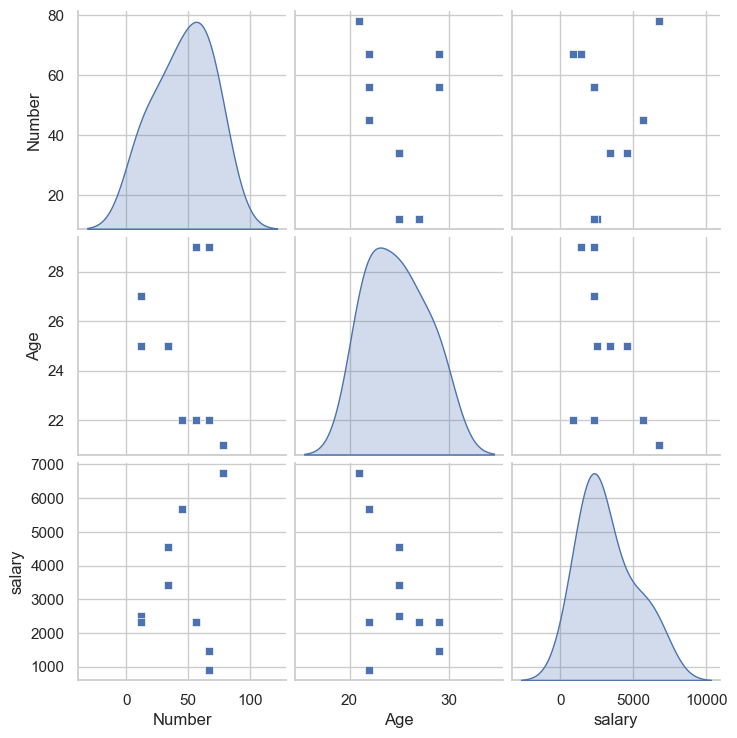
5 Amir Johnson F 56 PF 29 2324

6 Jordan Mickey G 78 PF 21 6754

7 Kelly Olynyk H 34 C 25 3421

8 Terry Rozier I 67 PG 22 890

9 Marcus Smart J 45 PG 22 5678



​