```
f1 = open("/testmarks1 (1).csv",'r')
f2 = open("/testmarks2.csv",'r')
import numpy as np
array=np.loadtxt('/testmarks2.csv',dtype=str,delimiter='

print(array)
import numpy as np
array1=np.loadtxt('/testmarks2.csv',dtype=float,delimite



print(array1)
array2 = array1.astype(float)
print(array1)
RollNo = array1[:,0]
EDS = array1[:,1]
SON = array1[:,2]
DT = array1[:,3]
ET = array1[:,4]
print(RollNo)
print(EDS)
print(SON)
print(DT)
print(ET)
#mean marks of students in EDS course
mean_EDS = np.mean(EDS)
print(mean_EDS)
#standard deviation of marks in SON course
std_deviation_SON = np.std(SON)
print(std_deviation_SON)
#correlation between two courses marks {DT and ET}
corr_DT_ET = np.corrcoef(DT,ET)
print(corr_DT_ET)
#to print the sum of marks in the row
print(np.sum(SON,axis = 0))
#to print maximum marks scored in DT
print(max(DT))
#to print minimum marks scored in EDS
print(min(EDS))
#to stack the two arrays vertically
Vstack = np.vstack((EDS,DT))
print(Vstack)
#to copy the transpose of an array in another array
sample_array = np.fastCopyAndTranspose(Vstack)
print(sample_array)
#to check whether the performance of a student is better
for i in range(0,10):
  performance_checker = np.greater_equal(ET[i],SON[i])
print(RollNo[i])
if(performance_checker == False):
  print("not good performance than SON")
else:
  print("good performance than SON")
Increasing_marks_SON = np.sort(SON)
```

testmarks1 (1).csv × testmarl •••

1 to 10 of 10 entries Filter

RollNo	EDS	SON	DT	ET
801	43.05	27.79	28.7	27.79
802	43.47	28.52	28.98	27.89
803	42.24	28.16	28.16	25.63
804	39.24	26.16	26.16	26.16
805	40.9	26.03	27.27	25.65
806	39.47	26.31	26.31	25.21
807	41.68	25.63	27.79	25.46
808	42.19	27.61	28.13	26.21
809	44.75	28.35	29.83	28.21
810	46.95	28.88	31.3	28.53

Show 10 per page

```
[['RollNo' 'SIC' 'AM' 'FE' 'IC']
 ['801' '28.48' '34.18' '30.56' '22.23']
 ['802' '28.1' '33.72' '30.68' '22.82']
 ['803' '26.16' '31.39' '28.2' '22.53']
 ['804' '26.16' '31.39' '28.78' '20.93']
 ['805' '26.1' '31.32' '28.22' '20.82']
 ['806' '25.45' '30.54' '27.73' '21.05']
 ['807' '26.16' '31.39' '28.01' '20.51']
 ['808' '27.44' '32.93' '28.83' '22.08']
 ['809' '28.63' '34.35' '31.03' '22.68']
 ['810' '30.35' '36.42' '31.38' '23.1']]
[[801.    28.48  34.18  30.56  22.23]
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
[[801.    28.48  34.18  30.56  22.23]
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
[801. 802. 803. 804. 805. 806. 807. 808. 809. 8
[28.48 28.1  26.16 26.16 26.1  25.45 26.16 27.4
[34.18 33.72 31.39 31.39 31.32 30.54 31.39 32.9
[30.56 30.68 28.2  28.78 28.22 27.73 28.01 28.8
[22.23 22.82 22.53 20.93 20.82 21.05 20.51 22.0
27.303000000000004
1.7754776822027365
[[1.         0.79223634]
 [0.79223634 1.        ]]
327.63000000000005
31.38
25.45
[[28.48 28.1  26.16 26.16 26.1  25.45 26.16 27.
 [30.56 30.68 28.2  28.78 28.22 27.73 28.01 28.
[[28.48 30.56]
 [28.1  30.68]
 [26.16 28.2 ]
 [26.16 28.78]
 [26.1  28.22]
 [25.45 27.73]
 [26.16 28.01]
 [27.44 28.83]
 [28.63 31.03]
 [30.35 31.38]]
810.0
not good performance than SON
```

Colab paid products - Cancel contracts here

✓ 0s completed at 12:38 PM