In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
dict1 = {
    "name":['harry','rohan','skillf','shubh'],
    "marks":[92,34,24,17],
    "city" : ['rampur','kolkata','sendhwa','warla']
}
```

In [3]:

```
df = pd.DataFrame(dict1)     # excel sheet of python
```

In [4]:

```
df
```

Out[4]:

	name	marks	city
0	harry	92	rampur
1	rohan	34	kolkata
2	skillf	24	sendhwa
3	shubh	17	warla

In [5]:

```
#df.to_csv('friends.csv')    #dictionary ko csv file mai export kar diya humne
```

In [6]:

```
df.to_csv('friends_index_false.csv',index=False)    #friends ki csv file mai se humne inde
x hata diye
```

In [7]:

```
df.head(2)    #big data ki starting ke 2 rows dikhayega
```

Out[7]:

	name	marks	city
0	harry	92	rampur
1	rohan	34	kolkata

In [8]:

```
df.tail(2)     #big data ki starting ke 2 rows dikhayega
```

Out[8]:

	name	marks	city
2	skillf	24	sendhwa
3	shubh	17	warla

In [9]:

```
df.describe()    #statical analysis karega
df.describe()    #statical analysis karega
```

Out[9]:

	marks
count	4.00000
mean	41.75000
std	34.21866
min	17.00000
25%	22.25000
50%	29.00000
75%	48.50000
max	92.00000

In [10]:

```
harry = pd.read_csv('harry.csv')
```

In [11]:

```
harry
```

Out[11]:

	train no.	marks	city
0	12345	92	rampur
1	6789	134	kolkata
2	9845	224	sendhwa
3	88787	17	warla

In [12]:

```
harry['marks']
```

Out[12]:

```
0     92
1    134
2    224
3     17
Name: marks, dtype: int64
```

In [13]:

```
harry['marks'][0]
```

Out[13]:

```
92
```

In [14]:

```
harry['marks'][0]=50
```

```
c:\users\todays\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launch
er.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [15]:

```
harry  #50 aagya hai
```

Out[15]:

	train no.	marks	city
0	12345	50	rampur
1	6789	134	kolkata
2	9845	224	sendhwa
3	88787	17	warla

In [16]:

```
harry.index = ['first','second','third','fourth']
```

In [17]:

```
harry
```

Out[17]:

	train no.	marks	city
first	12345	50	rampur
second	6789	134	kolkata
third	9845	224	sendhwa
fourth	88787	17	warla

CREATING DATA-FRAMES AND SERIES

In [18]:

```
ser = pd.Series(np.random.rand)
```

In [19]:

```
ser = pd.Series(np.random.rand(34))
```

In [20]:

```
ser
```

Out[20]:

```
0      0.049421
1      0.666874
2      0.181108
3      0.344739
4      0.954977
5      0.255420
6      0.870149
7      0.912351
8      0.228373
9      0.334605
10     0.994039
11     0.229756
12     0.780152
13     0.514914
14     0.978002
15     0.142244
16     0.577486
17     0.913164
18     0.096533
19     0.313820
20     0.802105
21     0.664694
```

```
22    0.097568
23    0.860976
24    0.197913
25    0.266642
26    0.785491
27    0.874523
28    0.429349
29    0.423502
30    0.594706
31    0.069077
32    0.753873
33    0.352135
dtype: float64
```

In [21]:

```
type(ser)
```

Out[21]:

```
pandas.core.series.Series
```

In [22]:

```
newdf = pd.DataFrame(np.random.rand(334,5),index = np.arange(334))
```

In [23]:

```
newdf.head()
```

Out[23]:

	0	1	2	3	4
0	0.583561	0.597810	0.034069	0.044353	0.946134
1	0.120110	0.550272	0.276535	0.425229	0.667197
2	0.764540	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724

In [24]:

```
newdf
```

Out[24]:

	0	1	2	3	4
0	0.583561	0.597810	0.034069	0.044353	0.946134
1	0.120110	0.550272	0.276535	0.425229	0.667197
2	0.764540	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724
...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625
330	0.630727	0.310683	0.334484	0.347565	0.459219
331	0.858608	0.891282	0.637447	0.972274	0.287337
332	0.565196	0.420689	0.636493	0.601230	0.309745
333	0.818161	0.219872	0.286213	0.167307	0.592274

334 rows × 5 columns

In [25]:

```
type(newdf)
```

Out[25]:

```
pandas.core.frame.DataFrame
```

In [26]:

```
newdf.describe()
```

Out[26]:

	0	1	2	3	4
count	334.000000	334.000000	334.000000	334.000000	334.000000
mean	0.503215	0.534662	0.500954	0.513117	0.513716
std	0.285055	0.287220	0.283772	0.285647	0.300588
min	0.000681	0.000059	0.008177	0.006676	0.006027
25%	0.251291	0.291770	0.259765	0.275139	0.232355
50%	0.543155	0.557049	0.510503	0.517158	0.532784
75%	0.747184	0.778473	0.735858	0.750233	0.774546
max	0.998958	0.998992	0.993837	0.998323	0.996663

In [27]:

```
newdf.dtypes
```

Out[27]:

```
0    float64
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [28]:

```
newdf[0][0]= "harry"
```

In [29]:

```
newdf.dtypes
```

Out[29]:

```
0     object
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [30]:

```
newdf.head()
```

Out[30]:

	0	1	2	3	4
0	harry	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328

	0	1	2	3	4
4	0.367675	0.277244	0.668145	0.960508	0.673724

In [31]:

```
newdf.index
```

Out[31]:

```
Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
            ...
            324, 325, 326, 327, 328, 329, 330, 331, 332, 333],
           dtype='int64', length=334)
```

In [32]:

```
newdf.columns  # 0 se 5 taq column hai
```

Out[32]:

```
RangeIndex(start=0, stop=5, step=1)
```

In [33]:

```
newdf[0][0] = 0.3
```

```
c:\users\todays\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launch
er.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [34]:

```
newdf.head()
```

Out[34]:

	0	1	2	3	4
0	0.3	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724

In [35]:

```
type(newdf)
```

Out[35]:

```
pandas.core.frame.DataFrame
```

In [36]:

```
newdf.to_numpy()   # dtype object hai
```

Out[36]:

```
array([[0.3, 0.597809886787921, 0.03406892751487245,
        0.044353218054538956, 0.9461340338894026],
       [0.12011014130891651, 0.5502721689344098, 0.27653514103489285,
        0.4252292782388245, 0.6671970123307828],
       [0.7645400317491767, 0.9096119896222608, 0.04047640395761043,
        0.8888501280036447, 0.2804346894169234],
       ...,
       [0.8586082246044735, 0.8912817277099832, 0.6374467698783677,
  0.9722743441074144, 0.28733748960318806],
 [0.5651955930081936, 0.42068908508530733, 0.6364933220278365,
  0.6012299426669835, 0.30974472797887576],
 [0.8181607761975885, 0.2198715049462997, 0.2862132591943768,
  0.16730740276400025, 0.5922738178922999]], dtype=object)
```

In [37]:

```
type(newdf.to_numpy())
```

Out[37]:

```
numpy.ndarray
```

In [38]:

```
newdf.T  # transpose ho gya hai , ab 325 column hai ;-)
```

Out[38]:

	0	1	2	3	4	5	6	7	8	9	...	324	325
0	0.3	0.12011	0.76454	0.869661	0.367675	0.982131	0.40556	0.320733	0.734534	0.355476	...	0.209045	0.471841
1	0.59781	0.550272	0.909612	0.220616	0.277244	0.503123	0.048589	0.192657	0.090477	0.889605	...	0.499615	0.108502
2	0.034069	0.276535	0.040476	0.668346	0.668145	0.342183	0.247162	0.841446	0.509514	0.412508	...	0.57578	0.261289
3	0.044353	0.425229	0.88885	0.890203	0.960508	0.104475	0.707146	0.357538	0.802024	0.24041	...	0.975334	0.753495
4	0.946134	0.667197	0.280435	0.902328	0.673724	0.535577	0.959192	0.348969	0.066753	0.295104	...	0.893473	0.058559

5 rows × 334 columns

In [39]:

```
newdf.head()
```

Out[39]:

	0	1	2	3	4
0	0.3	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724

In [40]:

```
newdf.sort_index(axis=0, ascending=False)   #rows mai sorting kar di jaayegi
                                             # ascending default true hi hota hai.
```

Out[40]:

	0	1	2	3	4
333	0.818161	0.219872	0.286213	0.167307	0.592274
332	0.565196	0.420689	0.636493	0.601230	0.309745
331	0.858608	0.891282	0.637447	0.972274	0.287337
330	0.630727	0.310683	0.334484	0.347565	0.459219
329	0.769477	0.794713	0.824400	0.034039	0.148625
...	...	...	...	...	...
4	0.367675	0.277244	0.668145	0.960508	0.673724
3	0.869661	0.220616	0.668346	0.890203	0.902328

2	0.76454	0.909612	0.040476	0.888850	0.280435
1	0.12011	0.550272	0.276535	0.425229	0.667197
0	0.3	0.597810	0.034069	0.044353	0.946134

334 rows × 5 columns

```
In [41]:
```

```
newdf.sort_index(axis=0)
```

```
Out[41]:
```

	0	1	2	3	4
0	0.3	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724
...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625
330	0.630727	0.310683	0.334484	0.347565	0.459219
331	0.858608	0.891282	0.637447	0.972274	0.287337
332	0.565196	0.420689	0.636493	0.601230	0.309745
333	0.818161	0.219872	0.286213	0.167307	0.592274

334 rows × 5 columns

```
In [42]:
```

```
newdf.sort_index(axis=1, ascending=False) # column ko sort kr dega
```

```
Out[42]:
```

	4	3	2	1	0
0	0.946134	0.044353	0.034069	0.597810	0.3
1	0.667197	0.425229	0.276535	0.550272	0.12011
2	0.280435	0.888850	0.040476	0.909612	0.76454
3	0.902328	0.890203	0.668346	0.220616	0.869661
4	0.673724	0.960508	0.668145	0.277244	0.367675
...	...	...	...	...	...
329	0.148625	0.034039	0.824400	0.794713	0.769477
330	0.459219	0.347565	0.334484	0.310683	0.630727
331	0.287337	0.972274	0.637447	0.891282	0.858608
332	0.309745	0.601230	0.636493	0.420689	0.565196
333	0.592274	0.167307	0.286213	0.219872	0.818161

334 rows × 5 columns

2-d numpy array ke liye axis = 0, row hoti hai axis = 1, column hoti hai

```
In [43]:
```

```
newdf[0]
```

```
Out[43]:
```

```
0          0.3
```

```
0         0.3
1       0.12011
2       0.76454
3      0.869661
4      0.367675
         ...
329    0.769477
330    0.630727
331    0.858608
332    0.565196
333    0.818161
Name: 0, Length: 334, dtype: object
```

In [44]:

```
type(newdf[0])
```

Out[44]:

```
pandas.core.series.Series
```

copy and view

In [45]:

```
newdf.head()
```

Out[45]:

	0	1	2	3	4
0	0.3	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724

In [46]:

```
newdf2 = newdf
```

In [47]:

```
newdf2[0][0] = 9783
```

```
c:\users\todays\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launch
er.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [48]:

```
newdf  # newdf2 mai change kiya toh newdf mai bhi change ho gya hai.  isko "view" bolte h
ai
```

Out[48]:

	0	1	2	3	4
0	9783	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328

	0	1	2	3	4
4	0.367675	0.277244	0.668145	0.960508	0.673724
...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625
330	0.630727	0.310683	0.334484	0.347565	0.459219
331	0.858608	0.891282	0.637447	0.972274	0.287337
332	0.565196	0.420689	0.636493	0.601230	0.309745
333	0.818161	0.219872	0.286213	0.167307	0.592274

334 rows × 5 columns

In [49]:

```
newdf2 = newdf.copy()
```

In [50]:

```
newdf2[0][0] = 123459
```

```
c:\users\todays\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launch
er.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [51]:

```
newdf   #copy kar ke change karenge toh original file mai changes nhi ate hai
```

Out[51]:

	0	1	2	3	4
0	9783	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724
...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625
330	0.630727	0.310683	0.334484	0.347565	0.459219
331	0.858608	0.891282	0.637447	0.972274	0.287337
332	0.565196	0.420689	0.636493	0.601230	0.309745
333	0.818161	0.219872	0.286213	0.167307	0.592274

334 rows × 5 columns

KABHI KAABHI YEH BAHUT DIFICULT HOTA HAI PANDAS KE LIYE KI VIEW RETURN KARE YA COPY RETURN KARE, TOH HUM KISI VALUE KO SET LOC FUNCTION SE SET KARTE HAI

In [52]:

```
newdf.loc[0,0] = 654
```

In [53]:

```
newdf.head(2)
```

Out[53]:

	0	1	2	3	4
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197

In [54]:

```
newdf.columns= list("ABCDE")
```

In [55]:

```
newdf.head(2)
```

Out[55]:

	A	B	C	D	E
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197

In [56]:

```
newdf.loc[0,0] = 654 # now changing the value again, since we have not written the cordin
ates properly
                     # so it will create a new zero column, but it will not replace (0,0
)
```

In [57]:

```
newdf.head()
```

Out[57]:

	A	B	C	D	E	0
0	654	0.597810	0.034069	0.044353	0.946134	654.0
1	0.12011	0.550272	0.276535	0.425229	0.667197	NaN
2	0.76454	0.909612	0.040476	0.888850	0.280435	NaN
3	0.869661	0.220616	0.668346	0.890203	0.902328	NaN
4	0.367675	0.277244	0.668145	0.960508	0.673724	NaN

In [58]:

```
newdf.drop(0) #without axis yeh row mai se hatayega, or axis=0, lagane se bhi yeh same op
eration karega
```

Out[58]:

	A	B	C	D	E	0
1	0.12011	0.550272	0.276535	0.425229	0.667197	NaN
2	0.76454	0.909612	0.040476	0.888850	0.280435	NaN
3	0.869661	0.220616	0.668346	0.890203	0.902328	NaN
4	0.367675	0.277244	0.668145	0.960508	0.673724	NaN
5	0.982131	0.503123	0.342183	0.104475	0.535577	NaN
...	...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625	NaN
330	0.630727	0.310683	0.334484	0.347565	0.459219	NaN
331	0.858608	0.891282	0.637447	0.972274	0.287337	NaN
332	0.565196	0.420689	0.636493	0.601230	0.309745	NaN

	A	B	C	D	E	0
333	0.818161	0.219872	0.286213	0.167307	0.592274	NaN

333 rows × 6 columns

In [59]:

```
newdf.drop(0, axis=1).head(2)  #ab yeh column mai se hatayega (axis=1)
                               # alternate code can be:-
                               # newdf = newdf.drop(0, axis=1)
                               # newdf.head()
```

Out[59]:

	A	B	C	D	E
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197

In [60]:

```
newdf = newdf.drop(0, axis=1)
```

In [61]:

```
newdf.head(2)
```

Out[61]:

	A	B	C	D	E
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197

In [62]:

```
newdf.loc[[1,2],['C','D']]     # to get set of column by loc function
```

Out[62]:

	C	D
1	0.276535	0.425229
2	0.040476	0.888850

newdf abhi taq, change nhi hua hai, isko change karne ke liye humko newdf=newdf.loc[[1,2],['C','D']], karna parga

In [63]:

```
newdf.loc[:,['C','D']]   #saare ke saare rows aa jayenge
```

Out[63]:

	C	D
0	0.034069	0.044353
1	0.276535	0.425229
2	0.040476	0.888850
3	0.668346	0.890203
4	0.668145	0.960508
...	...	...
329	0.824400	0.034039

	C	D
330	0.334484	0.347565
331	0.637447	0.972274
332	0.636493	0.601230
333	0.286213	0.167307

334 rows × 2 columns

In [64]:

```
newdf.loc[[1,2],:] # saare k saare column aa jaayege of row 1,2
```

Out[64]:

	A	B	C	D	E
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435

complex queries in pandas

In [65]:

```
newdf.loc[(newdf['A']<0.3)]   # 'A' column mai jaha jaha value 0.3 se kam aarhi h print kr
r do values.
```

Out[65]:

	A	B	C	D	E
1	0.12011	0.550272	0.276535	0.425229	0.667197
10	0.022695	0.951864	0.293573	0.753393	0.708440
12	0.241449	0.072632	0.636529	0.031855	0.360401
18	0.066118	0.543551	0.966136	0.711149	0.570929
19	0.222766	0.560364	0.044611	0.392123	0.255730
...	...	...	...	...	...
306	0.244188	0.833432	0.684494	0.894486	0.187685
312	0.164498	0.776250	0.068919	0.614533	0.631257
315	0.043375	0.622662	0.592407	0.716021	0.813493
324	0.209045	0.499615	0.575780	0.975334	0.893473
328	0.248103	0.652167	0.548285	0.795469	0.132718

96 rows × 5 columns

In [66]:

```
newdf.loc[(newdf['A']<0.3) & (newdf['C']>0.1)]      # &'C' bada ho 0.1 se..print it
```

Out[66]:

	A	B	C	D	E
1	0.12011	0.550272	0.276535	0.425229	0.667197
10	0.022695	0.951864	0.293573	0.753393	0.708440
12	0.241449	0.072632	0.636529	0.031855	0.360401
18	0.066118	0.543551	0.966136	0.711149	0.570929
22	0.004815	0.083171	0.971300	0.584355	0.552460
...	...	...	...	...	...

	A	B	C	D	E
303	0.105457	0.471974	0.473231	0.857508	0.561681
306	0.244188	0.833432	0.684494	0.894486	0.187685
315	0.043375	0.622662	0.592407	0.716021	0.813493
324	0.209045	0.499615	0.575780	0.975334	0.893473
328	0.248103	0.652167	0.548285	0.795469	0.132718

85 rows × 5 columns

iloc

In [67]:

```
newdf.head(2)
```

Out[67]:

	A	B	C	D	E
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197

In [68]:

```
newdf.iloc[0,4]   # iloc ka use jb krte h, jb humko index se search krna ho naa ki row,col
umn k naam se
```

Out[68]:

```
0.9461340338894026
```

In [69]:

```
newdf.iloc[[0,1],[1,2]]
```

Out[69]:

	B	C
0	0.597810	0.034069
1	0.550272	0.276535

In [70]:

```
newdf.iloc[[0,5],[1,2]]
```

Out[70]:

	B	C
0	0.597810	0.034069
5	0.503123	0.342183

In [71]:

```
newdf.head(3)
```

Out[71]:

	A	B	C	D	E
0	654	0.597810	0.034069	0.044353	0.946134
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435

In [72]:

```
newdf.drop([0])
```

Out[72]:

	A	B	C	D	E
1	0.12011	0.550272	0.276535	0.425229	0.667197
2	0.76454	0.909612	0.040476	0.888850	0.280435
3	0.869661	0.220616	0.668346	0.890203	0.902328
4	0.367675	0.277244	0.668145	0.960508	0.673724
5	0.982131	0.503123	0.342183	0.104475	0.535577
...	...	...	...	...	...
329	0.769477	0.794713	0.824400	0.034039	0.148625
330	0.630727	0.310683	0.334484	0.347565	0.459219
331	0.858608	0.891282	0.637447	0.972274	0.287337
332	0.565196	0.420689	0.636493	0.601230	0.309745
333	0.818161	0.219872	0.286213	0.167307	0.592274

333 rows × 5 columns

In [73]:

```
newdf.drop(['A'],axis=1)
```

Out[73]:

	B	C	D	E
0	0.597810	0.034069	0.044353	0.946134
1	0.550272	0.276535	0.425229	0.667197
2	0.909612	0.040476	0.888850	0.280435
3	0.220616	0.668346	0.890203	0.902328
4	0.277244	0.668145	0.960508	0.673724
...	...	...	...	...
329	0.794713	0.824400	0.034039	0.148625
330	0.310683	0.334484	0.347565	0.459219
331	0.891282	0.637447	0.972274	0.287337
332	0.420689	0.636493	0.601230	0.309745
333	0.219872	0.286213	0.167307	0.592274

334 rows × 4 columns

In [74]:

```
newdf.drop(['A','C'],axis=1)
```

Out[74]:

	B	D	E
0	0.597810	0.044353	0.946134
1	0.550272	0.425229	0.667197
2	0.909612	0.888850	0.280435
3	0.220616	0.890203	0.902328

	B	D	E
4	0.277244	0.960508	0.673724
...	...	...	...
329	0.794713	0.034039	0.148625
330	0.310683	0.347565	0.459219
331	0.891282	0.972274	0.287337
332	0.420689	0.601230	0.309745
333	0.219872	0.167307	0.592274

334 rows × 3 columns

In [75]:

```
newdf.drop(['A','D'],axis=1)   #yeh humne changes original files m nhi kiya h, it is just
a copy.
                               # isiliye c udane k baad bhi wapis aa chuka h
```

Out[75]:

	B	C	E
0	0.597810	0.034069	0.946134
1	0.550272	0.276535	0.667197
2	0.909612	0.040476	0.280435
3	0.220616	0.668346	0.902328
4	0.277244	0.668145	0.673724
...	...	...	...
329	0.794713	0.824400	0.148625
330	0.310683	0.334484	0.459219
331	0.891282	0.637447	0.287337
332	0.420689	0.636493	0.309745
333	0.219872	0.286213	0.592274

334 rows × 3 columns

Inplace = True

**inplace=True, krne se yeh sidhe original file m changes kr dega,
no neeed to write newdf is eql to operation name**

In [76]:

```
newdf.drop(['A','D'],axis=1,inplace=True)
```

In [77]:

```
newdf
```

Out[77]:

	B	C	E
0	0.597810	0.034069	0.946134
1	0.550272	0.276535	0.667197
2	0.909612	0.040476	0.280435
3	0.220616	0.668346	0.902328
4	0.277244	0.668145	0.673724

	B	C	E
...	...	...	...
329	0.794713	0.824400	0.148625
330	0.310683	0.334484	0.459219
331	0.891282	0.637447	0.287337
332	0.420689	0.636493	0.309745
333	0.219872	0.286213	0.592274

334 rows × 3 columns

In [ ]:

```

```

In [78]:

```
newdf.drop([1,5],axis=0,inplace=True)
```

In [79]:

```
newdf.head(6)
```

Out[79]:

	B	C	E
0	0.597810	0.034069	0.946134
2	0.909612	0.040476	0.280435
3	0.220616	0.668346	0.902328
4	0.277244	0.668145	0.673724
6	0.048589	0.247162	0.959192
7	0.192657	0.841446	0.348969

In [80]:

```
newdf.reset_index()
```

Out[80]:

	index	B	C	E
0	0	0.597810	0.034069	0.946134
1	2	0.909612	0.040476	0.280435
2	3	0.220616	0.668346	0.902328
3	4	0.277244	0.668145	0.673724
4	6	0.048589	0.247162	0.959192
...	...	...	...	...
327	329	0.794713	0.824400	0.148625
328	330	0.310683	0.334484	0.459219
329	331	0.891282	0.637447	0.287337
330	332	0.420689	0.636493	0.309745
331	333	0.219872	0.286213	0.592274

332 rows × 4 columns

In [81]:

```
newdf.reset_index().head(5)
```

Out[81]:

	index	B	C	E
0	0	0.597810	0.034069	0.946134
1	2	0.909612	0.040476	0.280435
2	3	0.220616	0.668346	0.902328
3	4	0.277244	0.668145	0.673724
4	6	0.048589	0.247162	0.959192

In [82]:

```
newdf.reset_index(drop=True,inplace=True)
```

In [83]:

```
newdf.head(3)
```

Out[83]:

	B	C	E
0	0.597810	0.034069	0.946134
1	0.909612	0.040476	0.280435
2	0.220616	0.668346	0.902328

In [84]:

```
newdf['B'].isnull()    # jaha pr true h, wha value zero h.
```

Out[84]:

```
0       False
1       False
2       False
3       False
4       False
        ...
327     False
328     False
329     False
330     False
331     False
Name: B, Length: 332, dtype: bool
```

In [85]:

```
newdf['B'] = None
```

In [86]:

```
newdf
```

Out[86]:

	B	C	E
0	None	0.034069	0.946134
1	None	0.040476	0.280435
2	None	0.668346	0.902328
3	None	0.668145	0.673724
4	None	0.247162	0.959192
...	...	...	...

	B	C	E
327	None	0.824400	0.148625
328	None	0.334484	0.459219
329	None	0.637447	0.287337
330	None	0.636493	0.309745
331	None	0.286213	0.592274

332 rows × 3 columns

In [87]:

```
newdf.loc[:,['B']] = None   # correct method to set values
```

In [88]:

```
newdf
```

Out[88]:

	B	C	E
0	None	0.034069	0.946134
1	None	0.040476	0.280435
2	None	0.668346	0.902328
3	None	0.668145	0.673724
4	None	0.247162	0.959192
...	...	...	...
327	None	0.824400	0.148625
328	None	0.334484	0.459219
329	None	0.637447	0.287337
330	None	0.636493	0.309745
331	None	0.286213	0.592274

332 rows × 3 columns

In [89]:

```
newdf.loc[:,['B']] = 56
```

In [90]:

```
newdf
```

Out[90]:

	B	C	E
0	56	0.034069	0.946134
1	56	0.040476	0.280435
2	56	0.668346	0.902328
3	56	0.668145	0.673724
4	56	0.247162	0.959192
...	...	...	...
327	56	0.824400	0.148625
328	56	0.334484	0.459219
329	56	0.637447	0.287337
330	56	0.636493	0.309745

	B	C	E
331	56	0.286213	0.592274

332 rows × 3 columns

In [91]:

```
newdf.head()
```

Out[91]:

	B	C	E
0	56	0.034069	0.946134
1	56	0.040476	0.280435
2	56	0.668346	0.902328
3	56	0.668145	0.673724
4	56	0.247162	0.959192

In [92]:

```
df = pd.DataFrame({'col1': [1, 2,'NaT'],
                   'col2': [3, 4,0],
                   'xcol3': ['napiun',5,6]
                  })
```

In [93]:

```
df
```

Out[93]:

	col1	col2	xcol3
0	1	3	napiun
1	2	4	5
2	NaT	0	6

In [94]:

```
df.dropna()
```

Out[94]:

	col1	col2	xcol3
0	1	3	napiun
1	2	4	5
2	NaT	0	6

DROPNA ,DROP_DUPLICATE and VALUE_COUNTS... WE HAVEN'T UNDERSTOOD TILL NOW

learning some more functions,methods and attributes

In [95]:

```
df
```

Out[95]:

	col1	col2	xcol3
0	1	3	napiun

	col1	col2	xcol3
0	1	3	naplun
1	2	4	5
2	NaT	0	6

In [96]:

```
df.shape
```

Out[96]:

```
(3, 3)
```

In [97]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   col1    3 non-null      object
 1   col2    3 non-null      int64 
 2   xcol3   3 non-null      object
dtypes: int64(1), object(2)
memory usage: 200.0+ bytes
```

In [99]:

```
df.notnull() #jaha null hai waha false dikhayega
```

Out[99]:

	col1	col2	xcol3
0	True	True	True
1	True	True	True
2	True	True	True

In [100]:

```
df.isnull()  # jaha null hai waha true dikhayega
```

Out[100]:

	col1	col2	xcol3
0	False	False	False
1	False	False	False
2	False	False	False

'''Assignment''' run the following method df.describe() df.corr() df.mean() df.count() df.max() df.min() df.median() df.std()

In [ ]:

```

```