In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
dict3 = {
"name":['ss','dd','rr','cc','ff'] , "sub":[2,4,5,7,8] , "age":[23,45,65,25,87] , "weight":[45,56,76,100,459]
 }
```

In [3]:
```python
df = pd.DataFrame(dict3)
```

In [4]:
```python
df
```

Out[4]:

	name	sub	age	weight
0	ss	2	23	45
1	dd	4	45	56
2	rr	5	65	76
3	cc	7	25	100
4	ff	8	87	459

In [5]:
```python
df.to_csv('name.csv')
```

In [6]:
```python
 df.head(3)
```

Out[6]:

	name	sub	age	weight
0	ss	2	23	45
1	dd	4	45	56
2	rr	5	65	76

In [7]:
```python
df.tail(3)
```

Out[7]:

	name	sub	age	weight
2	rr	5	65	76
3	cc	7	25	100
4	ff	8	87	459

In [8]:
```python
df.describe()
```

Out[8]:

	sub	age	weight
count	5.000000	5.000000	5.000000
mean	5.200000	49.000000	147.200000
std	2.387467	27.239677	175.555404
min	2.000000	23.000000	45.000000
25%	4.000000	25.000000	56.000000
50%	5.000000	45.000000	76.000000
75%	7.000000	65.000000	100.000000
max	8.000000	87.000000	459.000000

In [9]:
```python
dict3['sub'][1]=17
```

In [10]:
```python
dict3['sub'][2]
```

Out[10]:
```
5
```

In [11]:
```python
dict3.index = ['5','ff','4','7','dd']
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-f6c7ce4223f3> in <module>
----> 1 dict3.index = ['5','ff','4','7','dd']

AttributeError: 'dict' object has no attribute 'index'
```

In [12]:
```python
shivam = pd.read_csv('marks.csv') ###excel file ko notebook me bulna ho to ye fun ka use karte###
```

In [13]:
```python
shivam ## deko a gaya##
```

Out[13]:

	Unnamed: 0	name	marks	city
0	0	shivam	90	raipur
1	1	shubham	78	jabalpur
2	2	suraj	87	bhopal
3	3	shymul	67	shahdol

In [14]:
```python
shivam.index = ['5','ff','4','7'] # index ko chnage kar rahe #
```

In [15]:
```python
shivam # deko index vale change ho gye hai op #
```

Out[15]:

	Unnamed: 0	name	marks	city
5	0	shivam	90	raipur
ff	1	shubham	78	jabalpur
4	2	suraj	87	bhopal
7	3	shymul	67	shahdol

In [16]:
```python
shivam['city'][2] # coloum ke value ko bula rahe hai#
```

Out[16]:
```
'bhopal'
```

In [17]:
```python
shivam ['city'][2] ='indore' # coloum ke vale ko change kar rahe hai index kke maded se #
```

```
<ipython-input-17-3548167da00b>: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_guide/indexing.html#returning-a-view-versus-a-copy
  shivam ['city'][2] ='indore' # coloum ke vale ko change kar rahe hai index kke maded se #
```

In [18]:
```python
shivam # deko ho gaya channge #
```

Out[18]:

	Unnamed: 0	name	marks	city
5	0	shivam	90	raipur
ff	1	shubham	78	jabalpur
4	2	suraj	87	indore
7	3	shymul	67	shahdol

In [19]:
```python
shivam.to_csv('shivam2_csv') # csv file wapas se save  ho gaya hai #
```

In [20]:
```python
## chaliye shuru karte hai ##
```

In [21]:
```python
newdfff  = pd.DataFrame(np.random.rand(334,5) , index = np.arange(334))## creation of dataframe ##
```

In [22]:
```python
newdfff # calling new dataframe#
```

Out[22]:

	0	1	2	3	4
0	0.781981	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.092922	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.312270	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [23]:
```python
type(newdfff) # bata raha ki ye dataframe hai #
```

Out[23]:
```
pandas.core.frame.DataFrame
```

In [24]:
```python
newdfff.describe()
```

Out[24]:

	0	1	2	3	4
count	334.000000	334.000000	334.000000	334.000000	334.000000
mean	0.492469	0.476737	0.507156	0.482960	0.526656
std	0.284950	0.290641	0.289953	0.284042	0.273981
min	0.000334	0.000021	0.002552	0.007948	0.002052
25%	0.246542	0.216117	0.260031	0.237201	0.321162
50%	0.497922	0.459141	0.522007	0.492353	0.536820
75%	0.730725	0.744300	0.761693	0.708372	0.734478
max	0.999539	0.998868	0.998139	0.997852	0.998499

In [25]:
```python
newdfff.dtypes
```

Out[25]:
```
0    float64
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [26]:
```python
newdfff.head(10)
```

Out[26]:

	0	1	2	3	4
0	0.781981	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.092922	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
5	0.245853	0.725398	0.840809	0.026238	0.895827
6	0.711601	0.520287	0.547394	0.698721	0.618776
7	0.754726	0.745486	0.389523	0.512943	0.874342
8	0.899062	0.542402	0.582715	0.965121	0.500594
9	0.699753	0.239546	0.955863	0.484494	0.592837

In [27]:
```python
newdfff[0][0]='shivam patel'
```

In [28]:
```python
newdfff
```

Out[28]:

	0	1	2	3	4
0	shivam patel	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [29]:
```python
newdfff.dtypes # first row data types changes to object##
```

Out[29]:
```
0     object
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [30]:
```python
newdfff.index
```

Out[30]:
```
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 [31]:
```python
newdfff.columns
```

Out[31]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [32]:
```python
newdfff.to_numpy() # datatypes converting in to numpy arry#
```

Out[32]:
```
array([['shivam patel', 0.16065673709301487, 0.30692659284693935,
        0.5972397973500868, 0.8174182312622135],
       [0.33768207911931425, 0.03248971190293992, 0.5784102906470643,
        0.19644629680681125, 0.45919430561115704],
       [0.9995392699434426, 0.5533142668440891, 0.687495615152021,
        0.7160247552726826, 0.39786514231437353],
       ...,
       [0.26776257289656713, 0.44371140024431166, 0.34210377351550014,
        0.7960200907114762, 0.879930068879511],
       [0.9295957702894861, 0.7951142178287649, 0.602520754163023,
        0.8427772992785751, 0.8802486322948865],
       [0.22159515130889895, 0.7600039998877186, 0.9096688636965189,
        0.10503494605817776, 0.6891668779946094]], dtype=object)
```

In [33]:
```python
newdfff.T # transpose of matrix#
```

Out[33]:

	0	1	2	3	4	5	6	7	8	9	...	324	325	326	327	328	329	330	331	332	333
0	shivam patel	0.337682	0.999539	0.0929223	0.658756	0.245853	0.711601	0.754726	0.899062	0.699753	...	0.0448656	0.354321	0.688913	0.320678	0.0959358	0.31227	0.264547	0.267763	0.929596	0.221595
1	0.160657	0.0324897	0.553314	0.507143	0.129097	0.725398	0.520287	0.745486	0.542402	0.239546	...	0.56699	0.353269	0.633718	0.478718	0.136939	0.918388	0.479814	0.443711	0.795114	0.760004
2	0.306927	0.57841	0.687496	0.230202	0.0906149	0.840809	0.547394	0.389523	0.582715	0.955863	...	0.0709526	0.444389	0.613868	0.846737	0.122672	0.377515	0.647478	0.342104	0.602521	0.909669
3	0.59724	0.196446	0.716025	0.51295	0.637531	0.0262376	0.698721	0.512943	0.965121	0.484494	...	0.0187626	0.0943753	0.506235	0.397915	0.114118	0.63153	0.813936	0.79602	0.842777	0.105035
4	0.817418	0.459194	0.397865	0.298372	0.647706	0.895827	0.618776	0.874342	0.500594	0.592837	...	0.47963	0.694159	0.344469	0.0398603	0.911283	0.823016	0.374694	0.87993	0.880249	0.689167

5 rows × 334 columns

In [34]:
```python
newdfff.head()
```

Out[34]:

	0	1	2	3	4
0	shivam patel	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706

In [35]:
```python
newdfff.sort_index(axis=1,ascending= False) # axis0 means row and axis 1 means coloum , and here we can using sort_index function#
```

Out[35]:

	4	3	2	1	0
0	0.817418	0.597240	0.306927	0.160657	shivam patel
1	0.459194	0.196446	0.578410	0.032490	0.337682
2	0.397865	0.716025	0.687496	0.553314	0.999539
3	0.298372	0.512950	0.230202	0.507143	0.0929223
4	0.647706	0.637531	0.090615	0.129097	0.658756
...	...	...	...	...	...
329	0.823016	0.631530	0.377515	0.918388	0.31227
330	0.374694	0.813936	0.647478	0.479814	0.264547
331	0.879930	0.796020	0.342104	0.443711	0.267763
332	0.880249	0.842777	0.602521	0.795114	0.929596
333	0.689167	0.105035	0.909669	0.760004	0.221595

334 rows × 5 columns

In [36]:
```python
newdfff.sort_index(axis=0,ascending= False) # sort from 333 to 0#
```

Out[36]:

	0	1	2	3	4
333	0.221595	0.760004	0.909669	0.105035	0.689167
332	0.929596	0.795114	0.602521	0.842777	0.880249
331	0.267763	0.443711	0.342104	0.796020	0.879930
330	0.264547	0.479814	0.647478	0.813936	0.374694
329	0.31227	0.918388	0.377515	0.631530	0.823016
...	...	...	...	...	...
4	0.658756	0.129097	0.090615	0.637531	0.647706
3	0.0929223	0.507143	0.230202	0.512950	0.298372
2	0.999539	0.553314	0.687496	0.716025	0.397865
1	0.337682	0.032490	0.578410	0.196446	0.459194
0	shivam patel	0.160657	0.306927	0.597240	0.817418

334 rows × 5 columns

In [37]:
```python
type(newdfff[0])
```

Out[37]:
```
pandas.core.series.Series
```

In [38]:
```python
newdfff
```

Out[38]:

	0	1	2	3	4
0	shivam patel	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [39]:
```python
newdf = newdfff
```

In [40]:
```python
newdf
```

Out[40]:

	0	1	2	3	4
0	shivam patel	0.160657	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [41]:
```python
newdf.loc[0,1] = 4
```

In [42]:
```python
newdf
```

Out[42]:

	0	1	2	3	4
0	shivam patel	4.000000	0.306927	0.597240	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	0.512950	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [46]:
```python
newdf.loc[3,3] = 8
```

In [47]:
```python
newdf
```

Out[47]:

	0	1	2	3	4
0	shivam patel	4.000000	0.306927	8.000000	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	8.000000	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [48]:
```python
newdf.columns = list('abcde')
```

In [49]:
```python
newdf
```

Out[49]:

	a	b	c	d	e
0	shivam patel	4.000000	0.306927	8.000000	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	8.000000	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [50]:
```python
newdf.loc[0,0] = 78
```

In [51]:
```python
newdf
```

Out[51]:

	a	b	c	d	e	0
0	shivam patel	4.000000	0.306927	8.000000	0.817418	78.0
1	0.337682	0.032490	0.578410	0.196446	0.459194	NaN
2	0.999539	0.553314	0.687496	0.716025	0.397865	NaN
3	0.0929223	0.507143	0.230202	8.000000	0.298372	NaN
4	0.658756	0.129097	0.090615	0.637531	0.647706	NaN
...	...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016	NaN
330	0.264547	0.479814	0.647478	0.813936	0.374694	NaN
331	0.267763	0.443711	0.342104	0.796020	0.879930	NaN
332	0.929596	0.795114	0.602521	0.842777	0.880249	NaN
333	0.221595	0.760004	0.909669	0.105035	0.689167	NaN

334 rows × 6 columns

In [58]:
```python
newdf = newdf.drop(0,axis=1)
```

In [59]:
```python
newdf
```

Out[59]:

	a	b	c	d	e
0	shivam patel	4.000000	0.306927	8.000000	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	8.000000	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [61]:
```python
newdf.iloc[0,4]
```

Out[61]:
```
0.8174182312622135
```

In [64]:
```python
newdf.iloc[[0,1,4,5,55,48,234],[2,4]]
```

Out[64]:

	c	e
0	0.306927	0.817418
1	0.578410	0.459194
4	0.090615	0.647706
5	0.840809	0.895827
55	0.793632	0.662192
48	0.249413	0.474844
234	0.199555	0.588891

In [67]:
```python
newdf.loc[[1,2,3],['a','c']]#selecting perticuler row and coloum#
```

Out[67]:

	a	c
1	0.337682	0.578410
2	0.999539	0.687496
3	0.0929223	0.230202

In [68]:
```python
newdf
```

Out[68]:

	a	b	c	d	e
0	shivam patel	4.000000	0.306927	8.000000	0.817418
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
3	0.0929223	0.507143	0.230202	8.000000	0.298372
4	0.658756	0.129097	0.090615	0.637531	0.647706
...	...	...	...	...	...
329	0.31227	0.918388	0.377515	0.631530	0.823016
330	0.264547	0.479814	0.647478	0.813936	0.374694
331	0.267763	0.443711	0.342104	0.796020	0.879930
332	0.929596	0.795114	0.602521	0.842777	0.880249
333	0.221595	0.760004	0.909669	0.105035	0.689167

334 rows × 5 columns

In [80]:
```python
newdf.loc[(newdf['a']<0.3) and (newdf['b']>0.2)]##this function is use to give condition ##
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-80-422fa5aaa16a> in <module>
----> 1 newdf.loc[(newdf['a']<0.3) and (newdf['b']>0.2)]##this function is use to give condition ##

~\anaconda\lib\site-packages\pandas\core\ops\common.py in new_method(self, other)
     63         other = item_from_zerodim(other)
     64 
---> 65         return method(self, other)
     66 
     67     return new_method

~\anaconda\lib\site-packages\pandas\core\ops\__init__.py in wrapper(self, other)
    368         rvalues = extract_array(other, extract_numpy=True)
    369 
--> 370         res_values = comparison_op(lvalues, rvalues, op)
    371 
    372         return self._construct_result(res_values, name=res_name)

~\anaconda\lib\site-packages\pandas\core\ops\array_ops.py in comparison_op(left, right, op)
    242 
    243     elif is_object_dtype(lvalues.dtype):
--> 244         res_values = comp_method_OBJECT_ARRAY(op, lvalues, rvalues)
    245 
    246     else:

~\anaconda\lib\site-packages\pandas\core\ops\array_ops.py in comp_method_OBJECT_ARRAY(op, x, y)
     54         result = libops.vec_compare(x.ravel(), y.ravel(), op)
     55     else:
---> 56         result = libops.scalar_compare(x.ravel(), y, op)
     57     return result.reshape(x.shape)
     58 

pandas\_libs\ops.pyx in pandas._libs.ops.scalar_compare()

TypeError: '<' not supported between instances of 'str' and 'float'
```

In [81]:
```python
## inplace=true is used to change in to dataframe #
```

In [82]:
```python
newdf['b'].isnull()
```

Out[82]:
```
0      False
1      False
2      False
3      False
4      False
       ...  
329    False
330    False
331    False
332    False
333    False
Name: b, Length: 334, dtype: bool
```

In [87]:
```python
newdf.loc[[1,2,45,68,79,89],:]
```

Out[87]:

	a	b	c	d	e
1	0.337682	0.032490	0.578410	0.196446	0.459194
2	0.999539	0.553314	0.687496	0.716025	0.397865
45	0.541209	0.917080	0.211663	0.263021	0.447161
68	0.838151	0.466380	0.120295	0.043531	0.776046
79	0.874318	0.439572	0.550404	0.131665	0.231317
89	0.342314	0.804204	0.169982	0.868972	0.864078

In [ ]:
```python

```