

```
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', 'bareilly', 'antarctica']
}
```

This DataFrame converts the data into excel sheet type for faster indexing.

```
In [3]: df = pd.DataFrame(dict1)
```

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

```
Out[4]:
```

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

exporting this data into cse file---

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

```
In [6]: df.to_csv('friends_index_false.csv', index = False)
```

```
In [7]: df.head(2)
```

```
Out[7]:
```

	name	marks	city
--	------	-------	------

	name	marks	city
0	harry	92	rampur
1	rohan	34	kolkata

In [8]: `df.tail(2)`

Out[8]:

	name	marks	city
2	skillf	24	bareilly
3	shubh	17	antarctica

In [9]: `df.describe()`

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]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Train No	Speed	city
0	first	first	first	first	first	45472	53	rampur
1	second	second	second	second	second	12457	123	kolkata
2	third	third	third	third	third	55214	66	bareilly
3	fourth	fourth	fourth	fourth	fourth	22143	78	antarctica

In [12]: `harry['Speed'][0] = 53`

C:\Users\sinha\AppData\Local\Temp\ipykernel\_3256\358142011.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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

`harry['Speed'][0] = 53`

In [13]: `harry`

Out[13]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Train No	Speed	city
0	first	first	first	first	first	45472	53	rampur
1	second	second	second	second	second	12457	123	kolkata
2	third	third	third	third	third	55214	66	bareilly
3	fourth	fourth	fourth	fourth	fourth	22143	78	antarctica

In [14]: `harry.to_csv('harry.csv')`

In [15]: `harry.index = ['first', 'second', 'third', 'fourth']`

In [16]: `harry`

Out[16]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Train No	Speed	city
<b>first</b>	first	first	first	first	first	45472	53	rampur
<b>second</b>	second	second	second	second	second	12457	123	kolkata
<b>third</b>	third	third	third	third	third	55214	66	bareilly
<b>fourth</b>	fourth	fourth	fourth	fourth	fourth	22143	78	antarctica

```
In [17]: harry.to_csv('harry.csv')
```

## Series:

```
In [18]: ser = pd.Series(np.random.rand(34))
```

```
In [19]: ser
```

```
Out[19]: 0    0.888400
1    0.743659
2    0.560770
3    0.514610
4    0.127471
5    0.349229
6    0.261804
7    0.447980
8    0.855278
9    0.221545
10   0.857607
11   0.670443
12   0.858393
13   0.789814
14   0.655485
15   0.349726
16   0.617645
17   0.139599
18   0.592984
19   0.314106
20   0.503375
```

```
21    0.088006
22    0.938074
23    0.978958
24    0.834993
25    0.717370
26    0.992183
27    0.608254
28    0.110769
29    0.076230
30    0.440298
31    0.500516
32    0.955693
33    0.997272
dtype: float64
```

```
In [20]: type(ser)
```

```
Out[20]: pandas.core.series.Series
```

## DataFrame

```
In [21]: newdf = pd.DataFrame(np.random.rand(334,5), index = np.arange(334))
```

```
In [22]: newdf
```

```
Out[22]:
```

	0	1	2	3	4
0	0.938211	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618

	0	1	2	3	4
330	0.333856	0.739582	0.109828	0.092426	0.290442
331	0.976554	0.446599	0.518686	0.592286	0.495567
332	0.920160	0.033013	0.293166	0.165293	0.697086
333	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

In [23]:

```
type(newdf)
```

Out[23]:

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

In [24]:

```
newdf.describe
```

Out[24]:

```
<bound method NDFrame.describe of
0    0.938211    0.221561    0.378597    0.115729    0.812980
1    0.350938    0.243591    0.001701    0.917582    0.569717
2    0.145707    0.840852    0.770460    0.170001    0.652148
3    0.284041    0.069086    0.611711    0.443285    0.534159
4    0.690714    0.063363    0.509107    0.283771    0.983590
..    ...      ...      ...      ...      ...
329  0.796148    0.659404    0.830879    0.205306    0.228618
330  0.333856    0.739582    0.109828    0.092426    0.290442
331  0.976554    0.446599    0.518686    0.592286    0.495567
332  0.920160    0.033013    0.293166    0.165293    0.697086
333  0.446511    0.157353    0.015194    0.639812    0.216157

[334 rows x 5 columns]>
```

In [25]:

```
newdf.dtypes
```

Out[25]:

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

```
In [26]: newdf.head()
```

```
Out[26]:
```

	0	1	2	3	4
0	0.938211	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590

```
In [27]: newdf[0][0]='harry'
```

```
In [28]: newdf.head()
```

```
Out[28]:
```

	0	1	2	3	4
0	harry	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590

```
In [29]: newdf.dtypes
```

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

```
In [30]: newdf.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]: newdf.columns
```

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

```
In [32]: newdf.to_numpy()
```

```
Out[32]: array([[ 'harry', 0.22156142044355298, 0.3785972525427397,
                  0.11572886650108405, 0.8129803444807175],
                 [0.35093754834242596, 0.2435910201337277, 0.0017013831317554962,
                  0.9175823115601884, 0.5697169870040705],
                 [0.1457070882451783, 0.8408515105846878, 0.770459518706541,
                  0.17000106743828414, 0.6521482107779615],
                 ...,
                 [0.9765536610520034, 0.4465993826344621, 0.5186859958781891,
                  0.5922857911909607, 0.49556739791455273],
                 [0.9201602154726723, 0.03301295797648185, 0.2931662433335356,
                  0.16529262482223772, 0.6970861314514633],
                 [0.44651137244053596, 0.15735289529740104, 0.015194072124751212,
                  0.6398124366225497, 0.21615749841108056]], dtype=object)
```

```
In [33]: newdf[0][0]=0.3
```

```
In [34]: newdf.head()
```

```
Out[34]:
```

	0	1	2	3	4
0	0.3	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159



	0	1	2	3	4
4	0.690714	0.063363	0.509107	0.283771	0.983590

In [35]: `newdf.dtypes`

Out[35]:

```

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

```

In [36]: `newdf.T`

Out[36]:

	0	1	2	3	4	5	6	7	8	9	...	324	325	326	327	328
0	0.3	0.350938	0.145707	0.284041	0.690714	0.452395	0.509202	0.14414	0.563063	0.454412	...	0.993492	0.633888	0.351577	0.237644	0.671256
1	0.221561	0.243591	0.840852	0.069086	0.063363	0.434619	0.459558	0.125766	0.658304	0.487483	...	0.144144	0.770651	0.627598	0.924786	0.329321
2	0.378597	0.001701	0.77046	0.611711	0.509107	0.97073	0.394767	0.88858	0.264246	0.512859	...	0.080921	0.814575	0.066513	0.321506	0.566324
3	0.115729	0.917582	0.170001	0.443285	0.283771	0.263025	0.555282	0.490155	0.95104	0.428134	...	0.642152	0.383002	0.648455	0.03148	0.32697
4	0.81298	0.569717	0.652148	0.534159	0.98359	0.725147	0.57334	0.865127	0.511284	0.908319	...	0.377582	0.839567	0.783015	0.068454	0.275698

5 rows × 334 columns



In [37]: `newdf`

Out[37]:

	0	1	2	3	4
0	0.3	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148

	0	1	2	3	4
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618
330	0.333856	0.739582	0.109828	0.092426	0.290442
331	0.976554	0.446599	0.518686	0.592286	0.495567
332	0.92016	0.033013	0.293166	0.165293	0.697086
333	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

In [38]: `newdf.head()`

Out[38]:

	0	1	2	3	4
0	0.3	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590

In [39]: `newdf.sort_index(axis=0, ascending = False)`

Out[39]:

	0	1	2	3	4
333	0.446511	0.157353	0.015194	0.639812	0.216157
332	0.92016	0.033013	0.293166	0.165293	0.697086
331	0.976554	0.446599	0.518686	0.592286	0.495567

	0	1	2	3	4
<b>330</b>	0.333856	0.739582	0.109828	0.092426	0.290442
<b>329</b>	0.796148	0.659404	0.830879	0.205306	0.228618
...	...	...	...	...	...
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>0</b>	0.3	0.221561	0.378597	0.115729	0.812980

334 rows × 5 columns

In [40]:

```
newdf.sort_index(axis=1,ascending = False)
```

Out[40]:

	4	3	2	1	0
<b>0</b>	0.812980	0.115729	0.378597	0.221561	0.3
<b>1</b>	0.569717	0.917582	0.001701	0.243591	0.350938
<b>2</b>	0.652148	0.170001	0.770460	0.840852	0.145707
<b>3</b>	0.534159	0.443285	0.611711	0.069086	0.284041
<b>4</b>	0.983590	0.283771	0.509107	0.063363	0.690714
...	...	...	...	...	...
<b>329</b>	0.228618	0.205306	0.830879	0.659404	0.796148
<b>330</b>	0.290442	0.092426	0.109828	0.739582	0.333856
<b>331</b>	0.495567	0.592286	0.518686	0.446599	0.976554
<b>332</b>	0.697086	0.165293	0.293166	0.033013	0.92016
<b>333</b>	0.216157	0.639812	0.015194	0.157353	0.446511

334 rows × 5 columns

```
In [41]: newdf.head()
```

```
Out[41]:
```

	0	1	2	3	4
0	0.3	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590

```
In [42]: newdf[0]
```

```
Out[42]:
```

0	0.3
1	0.350938
2	0.145707
3	0.284041
4	0.690714
...	
329	0.796148
330	0.333856
331	0.976554
332	0.92016
333	0.446511

Name: 0, Length: 334, dtype: object

```
In [43]: type(newdf[0])
```

```
Out[43]: pandas.core.series.Series
```

```
In [44]: newdf.head()
```

```
Out[44]:
```

	0	1	2	3	4
--	---	---	---	---	---

	0	1	2	3	4
<b>0</b>	0.3	0.221561	0.378597	0.115729	0.812980
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590

**View:** Here, the newdf2 is reffering to the newdf, if we change the newdf2, then newdf will also change.

In [45]: `newdf2 = newdf`

In [46]: `newdf2[0][0]=9783`

In [47]: `newdf`

Out[47]:

	0	1	2	3	4
<b>0</b>	9783	0.221561	0.378597	0.115729	0.812980
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
<b>329</b>	0.796148	0.659404	0.830879	0.205306	0.228618
<b>330</b>	0.333856	0.739582	0.109828	0.092426	0.290442
<b>331</b>	0.976554	0.446599	0.518686	0.592286	0.495567

	0	1	2	3	4
<b>332</b>	0.92016	0.033013	0.293166	0.165293	0.697086
<b>333</b>	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

**Copy:** Here if we change the copied version, then there will be no effect on the base version.

In [48]: `newdf3 = newdf.copy()`

In [49]: `newdf3[0][0] = 998741`

C:\Users\sinha\AppData\Local\Temp\ipykernel\_3256\3139651283.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\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

`newdf3[0][0] = 998741`

In [50]: `newdf3`

Out[50]:

	0	1	2	3	4
<b>0</b>	998741	0.221561	0.378597	0.115729	0.812980
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
<b>329</b>	0.796148	0.659404	0.830879	0.205306	0.228618

	0	1	2	3	4
<b>330</b>	0.333856	0.739582	0.109828	0.092426	0.290442
<b>331</b>	0.976554	0.446599	0.518686	0.592286	0.495567
<b>332</b>	0.92016	0.033013	0.293166	0.165293	0.697086
<b>333</b>	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

In [51]:

```
newdf
```

Out[51]:

	0	1	2	3	4
<b>0</b>	9783	0.221561	0.378597	0.115729	0.812980
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
<b>329</b>	0.796148	0.659404	0.830879	0.205306	0.228618
<b>330</b>	0.333856	0.739582	0.109828	0.092426	0.290442
<b>331</b>	0.976554	0.446599	0.518686	0.592286	0.495567
<b>332</b>	0.92016	0.033013	0.293166	0.165293	0.697086
<b>333</b>	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

In [52]:

```
newdf.loc[0,0]=7854
```

```
In [53]: newdf
```

```
Out[53]:
```

	0	1	2	3	4
0	7854	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618
330	0.333856	0.739582	0.109828	0.092426	0.290442
331	0.976554	0.446599	0.518686	0.592286	0.495567
332	0.92016	0.033013	0.293166	0.165293	0.697086
333	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

```
In [54]: newdf.columns = list('ABCDE')
```

```
In [55]: newdf.head()
```

```
Out[55]:
```

	A	B	C	D	E
0	7854	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590



```
In [56]: newdf.loc[0,0]=5478
```

```
In [57]: newdf
```

```
Out[57]:
```

	A	B	C	D	E	O
0	7854	0.221561	0.378597	0.115729	0.812980	5478.0
1	0.350938	0.243591	0.001701	0.917582	0.569717	NaN
2	0.145707	0.840852	0.770460	0.170001	0.652148	NaN
3	0.284041	0.069086	0.611711	0.443285	0.534159	NaN
4	0.690714	0.063363	0.509107	0.283771	0.983590	NaN
...	...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618	NaN
330	0.333856	0.739582	0.109828	0.092426	0.290442	NaN
331	0.976554	0.446599	0.518686	0.592286	0.495567	NaN
332	0.92016	0.033013	0.293166	0.165293	0.697086	NaN
333	0.446511	0.157353	0.015194	0.639812	0.216157	NaN

334 rows × 6 columns

We don't want the above thing.

```
In [58]: newdf.loc[0, 'A']=5741226
```

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

```
Out[59]:
```

	A	B	C	D	E	O
0	5741226	0.221561	0.378597	0.115729	0.812980	5478.0

	A	B	C	D	E	O
1	0.350938	0.243591	0.001701	0.917582	0.569717	NaN
2	0.145707	0.840852	0.770460	0.170001	0.652148	NaN
3	0.284041	0.069086	0.611711	0.443285	0.534159	NaN
4	0.690714	0.063363	0.509107	0.283771	0.983590	NaN
...	...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618	NaN
330	0.333856	0.739582	0.109828	0.092426	0.290442	NaN
331	0.976554	0.446599	0.518686	0.592286	0.495567	NaN
332	0.92016	0.033013	0.293166	0.165293	0.697086	NaN
333	0.446511	0.157353	0.015194	0.639812	0.216157	NaN

334 rows × 6 columns

```
In [60]: newdf = newdf.drop(0,axis=1)
```

```
In [61]: newdf.head()
```

```
Out[61]:
```

	A	B	C	D	E
0	5741226	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590

```
In [62]: newdf.loc[[1,2],['C','D']]
```

Out[62]:

	C	D
1	0.001701	0.917582
2	0.770460	0.170001

In [63]:

```
newdf.loc[:,['C','D']]
```

Out[63]:

	C	D
0	0.378597	0.115729
1	0.001701	0.917582
2	0.770460	0.170001
3	0.611711	0.443285
4	0.509107	0.283771
...	...	...
329	0.830879	0.205306
330	0.109828	0.092426
331	0.518686	0.592286
332	0.293166	0.165293
333	0.015194	0.639812

334 rows × 2 columns

In [64]:

```
newdf.loc[[1,2],:]
```

Out[64]:

	A	B	C	D	E
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148

```
In [66]: newdf.loc[(newdf['A']<0.3) & (newdf['C']>0.1)]
```

```
Out[66]:
```

	A	B	C	D	E
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
7	0.14414	0.125766	0.888580	0.490155	0.865127
13	0.058742	0.538232	0.306175	0.821081	0.339722
16	0.213656	0.983737	0.794776	0.952600	0.693306
...	...	...	...	...	...
311	0.270118	0.639630	0.538410	0.289098	0.942373
319	0.257582	0.014462	0.605000	0.448764	0.719262
321	0.082699	0.765306	0.772211	0.498051	0.970876
322	0.264446	0.806987	0.413375	0.308250	0.462475
327	0.237644	0.924786	0.321506	0.031480	0.068454

101 rows × 5 columns

**iloc is used for choosing any values by using the indexes irrespective of index's names. But, in loc you have to specify the names of indexes for choosing.**

```
In [68]: newdf.iloc[0,4]
```

```
Out[68]: 0.8129803444807175
```

```
In [69]: newdf.iloc[[0,5],[1,2]]
```

```
Out[69]:
```

	B	C
0	0.221561	0.378597
5	0.434619	0.970730

```
In [70]: newdf.head(3)
```

```
Out[70]:
```

	A	B	C	D	E
0	5741226	0.221561	0.378597	0.115729	0.812980
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148

```
In [71]: newdf.drop([0])
```

```
Out[71]:
```

	A	B	C	D	E
1	0.350938	0.243591	0.001701	0.917582	0.569717
2	0.145707	0.840852	0.770460	0.170001	0.652148
3	0.284041	0.069086	0.611711	0.443285	0.534159
4	0.690714	0.063363	0.509107	0.283771	0.983590
5	0.452395	0.434619	0.970730	0.263025	0.725147
...	...	...	...	...	...
329	0.796148	0.659404	0.830879	0.205306	0.228618
330	0.333856	0.739582	0.109828	0.092426	0.290442
331	0.976554	0.446599	0.518686	0.592286	0.495567
332	0.92016	0.033013	0.293166	0.165293	0.697086
333	0.446511	0.157353	0.015194	0.639812	0.216157

333 rows × 5 columns

```
In [73]: newdf.drop(['A', 'C'], axis=1)
```

```
Out[73]:
```

	B	D	E
--	---	---	---

	<b>B</b>	<b>D</b>	<b>E</b>
<b>0</b>	0.221561	0.115729	0.812980
<b>1</b>	0.243591	0.917582	0.569717
<b>2</b>	0.840852	0.170001	0.652148
<b>3</b>	0.069086	0.443285	0.534159
<b>4</b>	0.063363	0.283771	0.983590
...	...	...	...
<b>329</b>	0.659404	0.205306	0.228618
<b>330</b>	0.739582	0.092426	0.290442
<b>331</b>	0.446599	0.592286	0.495567
<b>332</b>	0.033013	0.165293	0.697086
<b>333</b>	0.157353	0.639812	0.216157

334 rows × 3 columns

**inplace** is used for placing the changed dataframe into original dataframe.

In [74]:

```
newdf
```

Out[74]:

	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<b>0</b>	5741226	0.221561	0.378597	0.115729	0.812980
<b>1</b>	0.350938	0.243591	0.001701	0.917582	0.569717
<b>2</b>	0.145707	0.840852	0.770460	0.170001	0.652148
<b>3</b>	0.284041	0.069086	0.611711	0.443285	0.534159
<b>4</b>	0.690714	0.063363	0.509107	0.283771	0.983590
...	...	...	...	...	...
<b>329</b>	0.796148	0.659404	0.830879	0.205306	0.228618

	A	B	C	D	E
<b>330</b>	0.333856	0.739582	0.109828	0.092426	0.290442
<b>331</b>	0.976554	0.446599	0.518686	0.592286	0.495567
<b>332</b>	0.92016	0.033013	0.293166	0.165293	0.697086
<b>333</b>	0.446511	0.157353	0.015194	0.639812	0.216157

334 rows × 5 columns

In [76]: `newdf.drop(['A','D'], axis=1, inplace = True)`

In [77]: `newdf`

Out[77]:

	B	C	E
<b>0</b>	0.221561	0.378597	0.812980
<b>1</b>	0.243591	0.001701	0.569717
<b>2</b>	0.840852	0.770460	0.652148
<b>3</b>	0.069086	0.611711	0.534159
<b>4</b>	0.063363	0.509107	0.983590
...	...	...	...
<b>329</b>	0.659404	0.830879	0.228618
<b>330</b>	0.739582	0.109828	0.290442
<b>331</b>	0.446599	0.518686	0.495567
<b>332</b>	0.033013	0.293166	0.697086
<b>333</b>	0.157353	0.015194	0.216157

334 rows × 3 columns

```
In [78]: newdf.drop([1,5], axis=0, inplace = True)
```

```
In [79]: newdf
```

```
Out[79]:
```

	<b>B</b>	<b>C</b>	<b>E</b>
<b>0</b>	0.221561	0.378597	0.812980
<b>2</b>	0.840852	0.770460	0.652148
<b>3</b>	0.069086	0.611711	0.534159
<b>4</b>	0.063363	0.509107	0.983590
<b>6</b>	0.459558	0.394767	0.573340
...	...	...	...
<b>329</b>	0.659404	0.830879	0.228618
<b>330</b>	0.739582	0.109828	0.290442
<b>331</b>	0.446599	0.518686	0.495567
<b>332</b>	0.033013	0.293166	0.697086
<b>333</b>	0.157353	0.015194	0.216157

332 rows × 3 columns

**For resetting the indexing use reset\_index() ----->**

```
In [80]: newdf.reset_index()
```

```
Out[80]:
```

	<b>index</b>	<b>B</b>	<b>C</b>	<b>E</b>
<b>0</b>	0	0.221561	0.378597	0.812980
<b>1</b>	2	0.840852	0.770460	0.652148
<b>2</b>	3	0.069086	0.611711	0.534159
<b>3</b>	4	0.063363	0.509107	0.983590



	index	B	C	E	
	4	6	0.459558	0.394767	0.573340
	...	...	...	...	...
	327	329	0.659404	0.830879	0.228618
	328	330	0.739582	0.109828	0.290442
	329	331	0.446599	0.518686	0.495567
	330	332	0.033013	0.293166	0.697086
	331	333	0.157353	0.015194	0.216157

332 rows × 4 columns

```
In [82]: newdf.reset_index(drop=True,inplace=True)
```

```
In [84]: newdf.head(5)
```

```
Out[84]:
```

	B	C	E
0	0.221561	0.378597	0.812980
1	0.840852	0.770460	0.652148
2	0.069086	0.611711	0.534159
3	0.063363	0.509107	0.983590
4	0.459558	0.394767	0.573340

```
In [87]: newdf['B'].isnull()
```

```
Out[87]:
```

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

```
In [89]: newdf['B'].isnull()
```

```

Out[89]: 0      True
          1      True
          2      True
          3      True
          4      True
          ...
          327    True
          328    True
          329    True
          330    True
          331    True
Name: B, Length: 332, dtype: bool

```

**But don't set any thing by using above method beacuse it sometime change or sometime not, so use loc.**

```
In [90]: newdf.loc[:,['B']] = None
```

```
In [91]: newdf
```

```

Out[91]:
   B      C      E
0  None  0.378597  0.812980
1  None  0.770460  0.652148
2  None  0.611711  0.534159
3  None  0.509107  0.983590

```

	<b>B</b>	<b>C</b>	<b>E</b>
<b>4</b>	None	0.394767	0.573340
...	...	...	...
<b>327</b>	None	0.830879	0.228618
<b>328</b>	None	0.109828	0.290442
<b>329</b>	None	0.518686	0.495567
<b>330</b>	None	0.293166	0.697086
<b>331</b>	None	0.015194	0.216157

332 rows × 3 columns

In [92]: `newdf.loc[:,['B']] = 34`

In [93]: `newdf`

Out[93]:

	<b>B</b>	<b>C</b>	<b>E</b>
<b>0</b>	34	0.378597	0.812980
<b>1</b>	34	0.770460	0.652148
<b>2</b>	34	0.611711	0.534159
<b>3</b>	34	0.509107	0.983590
<b>4</b>	34	0.394767	0.573340
...	...	...	...
<b>327</b>	34	0.830879	0.228618
<b>328</b>	34	0.109828	0.290442
<b>329</b>	34	0.518686	0.495567
<b>330</b>	34	0.293166	0.697086

	B	C	E
331	34	0.015194	0.216157

332 rows × 3 columns

```
In [94]: df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'],
                             "toy": [np.nan, 'Batmobile', 'Bullwhip'],
                             "born": [pd.NaT, pd.Timestamp("1940-04-25"),
                                       pd.NaT]})
```

In [95]: df

```
Out[95]:
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

In [96]: df.dropna()

```
Out[96]:
```

	name	toy	born
1	Batman	Batmobile	1940-04-25

```
In [99]: df2 = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'],
                             "toy": [np.nan, np.nan, np.nan],
                             "born": [pd.NaT, pd.Timestamp("1940-04-25"),
                                       pd.NaT]})
```

In [100... df2

```
Out[100...
```

	name	toy	born
--	------	-----	------

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	NaN	1940-04-25
2	Catwoman	NaN	NaT

In [102... `df2.dropna(how='all',axis=1)`

Out[102...

	name	born
0	Alfred	NaT
1	Batman	1940-04-25
2	Catwoman	NaT

In [103... `df3 = pd.DataFrame({"name": ['Alfred', 'Batman', 'Alfred'],  
"toy": [np.nan, 'Batmobile', 'Bullwhip'],  
"born": [pd.NaT, pd.Timestamp("1940-04-25"),  
pd.NaT]})`

In [104... `df3`

Out[104...

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Alfred	Bullwhip	NaT

In [106... `df3.drop_duplicates(subset=['name'])`

Out[106...

	name	toy	born
0	Alfred	NaN	NaT

	name	toy	born
1	Batman	Batmobile	1940-04-25

```
In [107... df3.drop_duplicates(subset=['name'], keep = 'first')
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25

```
In [108... df3.drop_duplicates(subset=['name'], keep = 'last')
```

	name	toy	born
1	Batman	Batmobile	1940-04-25
2	Alfred	Bullwhip	NaT

```
In [109... df3.drop_duplicates(subset=['name'], keep = False)
```

	name	toy	born
1	Batman	Batmobile	1940-04-25

```
In [110... df
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

```
In [111... df.shape
```

```
Out[111... (3, 3)
```

```
In [113... df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0   name    3 non-null      object
1   toy     2 non-null      object
2   born    1 non-null      datetime64[ns]
dtypes: datetime64[ns](1), object(2)
memory usage: 200.0+ bytes
```

```
In [115... df['toy'].value_counts(dropna=False)
```

```
Out[115... NaN      1
Batmobile  1
Bullwhip   1
Name: toy, dtype: int64
```

```
In [116... df['toy'].value_counts(dropna=True)
```

```
Out[116... Batmobile  1
Bullwhip   1
Name: toy, dtype: int64
```

```
In [117... df.isnull()
```

```
Out[117...   name  toy  born
0  False  True  True
1  False False False
2  False False  True
```

```
In [118... df.notnull()
```

```
Out[118...
   name  toy  born
0   True  False False
1   True   True   True
2   True   True  False
```

## Using the excel sheet:

```
In [119... data_excel = pd.read_excel('Data.xls')
```

```
-----
ImportError                                Traceback (most recent call last)
```

```
~\AppData\Local\Temp\ipykernel_3256\254838327.py in <module>
```

```
----> 1 data_excel = pd.read_excel('Data.xls')
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
```

```
    309         stacklevel=stacklevel,
```

```
    310     )
--> 311     return func(*args, **kwargs)
```

```
    312
```

```
    313     return wrapper
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\excel\_base.py in read_excel(io, sheet_name, header, names, index_col, usecols, squeeze, dtype, engine, converters, true_values, false_values, skiprows, nrows, na_values, keep_default_na, na_filter, verbose, parse_dates, date_parser, thousands, comment, skipfooter, convert_float, mangle_dupe_cols, storage_options)
```

```
    362     if not isinstance(io, ExcelFile):
```

```
    363         should_close = True
```

```
--> 364     io = ExcelFile(io, storage_options=storage_options, engine=engine)
```

```
    365     elif engine and engine != io.engine:
```

```
    366         raise ValueError(
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\excel\_base.py in __init__(self, path_or_buffer, engine, storage_options)
```

```
    1231     self.storage_options = storage_options
```

```
    1232
```

```
-> 1233     self._reader = self._engines[engine](self._io, storage_options=storage_options)
```

```
    1234
```



```

1235     def __fspath__(self):

~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\excel\_xlrd.py in __init__(self, filepath_or_buffer, storage_
options)
    22         """
    23         err_msg = "Install xlrd >= 1.0.0 for Excel support"
--> 24         import_optional_dependency("xlrd", extra=err_msg)
    25         super().__init__(filepath_or_buffer, storage_options=storage_options)
    26

~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\compat\_optional.py in import_optional_dependency(name, extra, e
rrors, min_version)
    116     except ImportError:
    117         if errors == "raise":
--> 118             raise ImportError(msg) from None
    119         else:
    120             return None

```

**ImportError:** Missing optional dependency 'xlrd'. Install xlrd >= 1.0.0 for Excel support Use pip or conda to install xlrd.

**We need xlrd module for reading the excel file**

```
In [122... data_excel = pd.read_excel('Data.xls')
```

```
In [123... data_excel
```

```
Out[123...
   Train No  Speed  city
0      45472     53  rampur
1      12457    123  kolkata
2      55214     66  bareilly
3      22143     78  antarctica
```

**For reading the sheet2 in the excel file use, ----->**

```
In [125... pd.read_excel('Data.xls', sheet_name='Sheet1')
```

```
Out[125...
   Train No  Speed  city
```

	Train No	Speed	city
0	45472	53	rampur
1	12457	123	kolkata
2	55214	66	bareilly
3	22143	78	antarctica

```
In [130...] data_excel=pd.read_excel('Data.xls', sheet_name='Sheet2')
```

```
In [131...] data_excel
```

```
Out[131...] 

|   | Train NoS2 | SpeedS2 | cityS2     |
|---|------------|---------|------------|
| 0 | 45472      | 53      | rampur     |
| 1 | 24577      | 123     | kolkata    |
| 2 | 55214      | 4456    | bareilly   |
| 3 | 22143      | 78      | antarctica |


```

```
In [132...] data_excel.iloc[0,0]=34
```

```
In [133...] data_excel
```

```
Out[133...] 

|   | Train NoS2 | SpeedS2 | cityS2     |
|---|------------|---------|------------|
| 0 | 34         | 53      | rampur     |
| 1 | 24577      | 123     | kolkata    |
| 2 | 55214      | 4456    | bareilly   |
| 3 | 22143      | 78      | antarctica |


```

```
In [134... data_excel.to_excel('Data.xls', sheet_name='Sheet2')
```

C:\Users\sinha\AppData\Local\Temp\ipykernel\_3256\270422046.py:1: FutureWarning: As the xlwt package is no longer maintained, the xlwt engine will be removed in a future version of pandas. This is the only engine in pandas that supports writing in the xls format. Install openpyxl and write to an xlsx file instead. You can set the option io.excel.xls.writer to 'xlwt' to silence this warning. While this option is deprecated and will also raise a warning, it can be globally set and the warning suppressed.

```
data_excel.to_excel('Data.xls', sheet_name='Sheet2')
```

**ModuleNotFoundError** Traceback (most recent call last)

```
~\AppData\Local\Temp\ipykernel_3256\270422046.py in <module>
```

```
----> 1 data_excel.to_excel('Data.xls', sheet_name='Sheet2')
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\generic.py in to_excel(self, excel_writer, sheet_name, na_rep, float_format, columns, header, index, index_label, startrow, startcol, engine, merge_cells, encoding, inf_rep, verbose, freeze_panes, storage_options)
```

```
2282         inf_rep=inf_rep,
2283     )
-> 2284     formatter.write(
2285         excel_writer,
2286         sheet_name=sheet_name,
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\formats\excel.py in write(self, writer, sheet_name, startrow, startcol, freeze_panes, engine, storage_options)
```

```
832         # error: Cannot instantiate abstract class 'ExcelWriter' with abstract
833         # attributes 'engine', 'save', 'supported_extensions' and 'write_cells'
--> 834         writer = ExcelWriter( # type: ignore[abstract]
835             writer, engine=engine, storage_options=storage_options
836         )
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\excel\_xlwt.py in __init__(self, path, engine, date_format, datetime_format, encoding, mode, storage_options, if_sheet_exists, engine_kwargs, **kwargs)
```

```
37     ):
38         # Use the xlwt module as the Excel writer.
---> 39         import xlwt
40
41         engine_kwargs = combine_kwargs(engine_kwargs, kwargs)
```

**ModuleNotFoundError**: No module named 'xlwt'

**We need openpyxl and xlwt module for writing in the excel file**

```
In [135... data_excel.to_excel('Data.xls', sheet_name='Sheet2')
```

```
C:\Users\sinha\AppData\Local\Temp\ipykernel_3256\270422046.py:1: FutureWarning: As the xlwt package is no longer maintained, the xlwt engine will be removed in a future version of pandas. This is the only engine in pandas that supports writing in the xls format. Install openpyxl and write to an xlsx file instead. You can set the option io.excel.xls.writer to 'xlwt' to silence this warning. While this option is deprecated and will also raise a warning, it can be globally set and the warning suppressed.  
    data_excel.to_excel('Data.xls', sheet_name='Sheet2')
```

**Sheet1 is removed that Data.xls file and only update Sheet2 is present there**

In [ ]: