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

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

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

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

Out[ ]:

	name	marks	city
0	harry	92	rampur
1	rohan	34	kolkata

In [ ]:
```
df.iloc[0, 1]
```

Out[ ]:
```
92
```

In [ ]:
```
df.loc[0, 'name']
```

Out[ ]:
```
'harry'
```

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

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

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

Out[ ]:

	name	marks	city
2	skillf	24	bareilly
3	shubh	17	antartica

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

Out[ ]:

	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 [ ]:
```
harry =pd.read_csv('harry.csv')
harry.head(2)
```

Out[ ]:

	Train No	speed	city
0	12332	23	rampur
1	21312	434	kolkata

In [ ]:
```
harry['speed']
```

Out[ ]:
```
0     23
1    434
2     34
3     54
Name: speed, dtype: int64
```

In [ ]:
```
harry['speed'][0]
```

Out[ ]:
```
23
```

In [ ]:
```
harry['speed'][0] = 50
```

```
/tmp/ipykernel_39255/1373715151.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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  harry['speed'][0] = 50
```

In [ ]:
```
harry
```

Out[ ]:

	Train No	speed	city
0	12332	50	rampur
1	21312	434	kolkata
2	33443	34	bareilly
3	23432	54	antartica

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

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

Out[ ]:

	Train No	speed	city
first	12332	50	rampur
second	21312	434	kolkata
third	33443	34	bareilly
fourth	23432	54	antartica

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

Out[ ]:
```
0     0.893803
1     0.608488
2     0.245382
3     0.611206
4     0.131315
5     0.921133
6     0.646493
7     0.255079
8     0.585562
9     0.752594
10    0.911513
11    0.676373
12    0.321083
13    0.062318
14    0.974582
15    0.086408
16    0.288207
17    0.893103
18    0.755164
19    0.104599
20    0.611153
21    0.248585
22    0.336219
23    0.014721
24    0.633397
25    0.219351
26    0.466968
27    0.475167
28    0.878644
29    0.101099
30    0.761335
31    0.557953
32    0.653512
33    0.781162
dtype: float64
```

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

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

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

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

Out[ ]:

	0	1	2	3	4
0	0.810209	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
newdf.shape
```

Out[ ]:
```
(334, 5)
```

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

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

In [ ]:
```
newdf.describe()
```

Out[ ]:

	0	1	2	3	4
count	334.000000	334.000000	334.000000	334.000000	334.000000
mean	0.489582	0.506816	0.504817	0.513452	0.503031
std	0.297242	0.297143	0.297456	0.291413	0.284580
min	0.001228	0.001901	0.001041	0.002026	0.004776
25%	0.238199	0.236683	0.225152	0.259632	0.262637
50%	0.502921	0.524855	0.527587	0.541478	0.493725
75%	0.739170	0.768658	0.761853	0.751679	0.757483
max	0.998542	0.999944	0.994070	0.990504	0.999202

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

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

In [ ]:
```
newdf[0][0] = 'harry'
newdf.dtypes
```

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

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

Out[ ]:

	0	1	2	3	4
0	harry	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# to get index
newdf.index
```

Out[ ]:
```
Index([  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 [ ]:
```
# to get columns
newdf.columns
```

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

In [ ]:
```
newdf[0][0] = 0.3
newdf.to_numpy()
```

Out[ ]:
```
array([[0.3, 0.6100009967284531, 0.13688140220949918,
        0.14887864136113416, 0.8188091085664424],
       [0.5902026884520216, 0.9129778408770255, 0.37047849487838447,
        0.9608183236742593, 0.7139648412719722],
       [0.6649344974860535, 0.2670538616747524, 0.7709577137320747,
        0.16396576763037052, 0.6713831147983278],
       ...,
       [0.8896649342425758, 0.10692054031645581, 0.6952345595737408,
        0.0020259413821824834, 0.6778705815662249],
       [0.7245428159420287, 0.9348624749141109, 0.2720478762326516,
        0.6118086036573352, 0.11334124368108123],
       [0.4935335583678495, 0.23946000565176062, 0.5128098702731696,
        0.9756068576425894, 0.10453680412375155]], dtype=object)
```

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

Out[ ]:
```
numpy.ndarray
```

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

Out[ ]:

	0	1	2	3	4
0	0.3	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# to get transpose
newdf.T
```

Out[ ]:

	0	1	2	3	4	5	6	7	
0	0.3	0.590203	0.664934	0.119425	0.825468	0.102519	0.96169	0.44362	0.
1	0.610001	0.912978	0.267054	0.904558	0.081574	0.32308	0.18208	0.727806	0.
2	0.136881	0.370478	0.770958	0.570511	0.666546	0.812856	0.197319	0.420511	0.
3	0.148879	0.960818	0.163966	0.505337	0.611949	0.745648	0.202361	0.642496	0.
4	0.818809	0.713965	0.671383	0.21656	0.825693	0.342632	0.121922	0.136511	0.

5 rows × 334 columns

In [ ]:
```
# newdf2 is a view of newdf. Like a pointer
newdf2 = newdf
```

In [ ]:
```
newdf2[0][0] = 978
newdf.head()
```

Out[ ]:

	0	1	2	3	4
0	978	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# to get copy
newdf3 = newdf.copy()
```

In [ ]:
```
newdf3[0][0] = 123
newdf.head()
```

```
/tmp/ipykernel_39255/412867064.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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  newdf3[0][0] = 123
```

Out[ ]:

	0	1	2	3	4
0	978	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# To not get error while setting values
newdf.loc[0, 0] = 654
```

```
newdf.head()
```

Out[ ]:

	0	1	2	3	4
0	654	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# To change columns
newdf.columns = list('ABCDE')
newdf.head()
```

Out[ ]:

	A	B	C	D	E
0	654	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
newdf.loc[0, 0] = 654
newdf.head()
```

Out[ ]:

	A	B	C	D	E	0
0	654	0.610001	0.136881	0.148879	0.818809	654.0
1	0.590203	0.912978	0.370478	0.960818	0.713965	NaN
2	0.664934	0.267054	0.770958	0.163966	0.671383	NaN
3	0.119425	0.904558	0.570511	0.505337	0.216560	NaN
4	0.825468	0.081574	0.666546	0.611949	0.825693	NaN

In [ ]:
```
newdf.loc[0, 'A'] = 655
newdf.head()
```

Out[ ]:

	A	B	C	D	E	0
0	655	0.610001	0.136881	0.148879	0.818809	654.0
1	0.590203	0.912978	0.370478	0.960818	0.713965	NaN
2	0.664934	0.267054	0.770958	0.163966	0.671383	NaN
3	0.119425	0.904558	0.570511	0.505337	0.216560	NaN
4	0.825468	0.081574	0.666546	0.611949	0.825693	NaN

In [ ]:
```
# axis = 0 for rows and axis = 1 for columns
newdf = newdf.drop(0, axis=1)
```

```
newdf.head()
```

Out[ ]:

	A	B	C	D	E
0	655	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
newdf.loc[[1, 2], ['C', 'D']]
# To change
# newdf = newdf.loc[[1, 2], ['C', 'D']]
```

Out[ ]:

	C	D
1	0.370478	0.960818
2	0.770958	0.163966

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

Out[ ]:

	A	B	C	D	E
0	655	0.610001	0.136881	0.148879	0.818809
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383
3	0.119425	0.904558	0.570511	0.505337	0.216560
4	0.825468	0.081574	0.666546	0.611949	0.825693

In [ ]:
```
# Use : to get all rows or columns
newdf.loc[[1, 2], :]
```

Out[ ]:

	A	B	C	D	E
1	0.590203	0.912978	0.370478	0.960818	0.713965
2	0.664934	0.267054	0.770958	0.163966	0.671383

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

Out[ ]:

	A	B	C	D	E
3	0.119425	0.904558	0.570511	0.505337	0.21656

In [ ]:
```
# by using index
newdf.iloc[0, 4]
```

Out[ ]:
```
0.8188091085664424
```

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

Out[ ]:

	B	C
0	0.610001	0.136881
1	0.912978	0.370478

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

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

Out[ ]:

	B	D	E
0	0.610001	0.148879	0.818809
1	0.912978	0.960818	0.713965
2	0.267054	0.163966	0.671383
3	0.904558	0.505337	0.216560
4	0.081574	0.611949	0.825693

In [ ]:
```
# to reset index
newdf.reset_index(drop=True, inplace=True)
newdf.head()
```

Out[ ]:

	B	D	E
0	0.610001	0.148879	0.818809
1	0.912978	0.960818	0.713965
2	0.267054	0.163966	0.671383
3	0.904558	0.505337	0.216560
4	0.081574	0.611949	0.825693

In [ ]:
```
# check if any value is null
newdf['B'].isnull()
```

Out[ ]:
```
0    False
1    False
2    False
3    False
4    False
Name: B, dtype: bool
```

In [ ]:
```
newdf.loc[:, ['B']] = None
newdf['B'].isnull()
```

Out[ ]:
```
0    True
1    True
2    True
3    True
4    True
Name: B, dtype: bool
```

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

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   name    4 non-null      object
 1   marks   4 non-null      int64 
 2   city    4 non-null      object
dtypes: int64(1), object(2)
memory usage: 224.0+ bytes
```

In [ ]:
```
# dropna=False means to count NaN values also
# dropna=True means to not count NaN values
df['name'].value_counts(dropna=False)
```

Out[ ]:
```
name
harry     1
rohan     1
skillf    1
shubh     1
Name: count, dtype: int64
```

In [ ]:
```
df.notnull()
```

Out[ ]:

	name	marks	city
0	True	True	True
1	True	True	True
2	True	True	True
3	True	True	True

In [ ]:
```
df.isnull()
```

Out[ ]:

	name	marks	city
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False

In [ ]:
```
# To read from excel
data = pd.read_excel('data.xlsx', sheet_name='Sheet1')
# install pip3 install xlrd
```