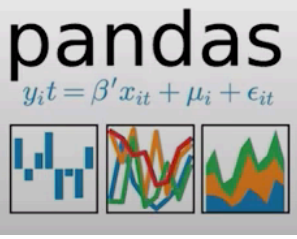
WHAT IS PANDAS

- Pandas is an open source data analysis library written in python
- It leverages the power and speed of numpy to make data analysis and preprocessing easy for data scientists
- It provides rich and highly robust data operations



PANDAS DATA STRUCTURE

- **Pandas has two types of data structures:**
  - **a) Series – It's a one dimensional array with indexes, it stores a single column or row of data in a Dataframe**
  - **b) Dataframe – It's a tabular spreadsheet like structure representing rows each of which contains one or multiple columns**
- A one-dimensional array(labeled) capable of holding any type of data– Series
- A two-dimensional data (labeled) structure with columns of potentially different types of data - DataFrame

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

In [2]:
```
dict1 = {
    "name" : ['anant', 'bobby', 'celina', 'danny'],
    "marks": [21, 34, 45, 33],
    "city" : ["Ahmedabad", "Lko", "knp", "kolkata"]
}
```

In [3]:
```
dict1
```

Out[3]:
```
{'name': ['anant', 'bobby', 'celina', 'danny'],
 'marks': [21, 34, 45, 33],
 'city': ['Ahmedabad', 'Lko', 'knp', 'kolkata']}
```

In [4]:
```
df = pd.DataFrame(dict1)
```

In [5]:
```
df
```

Out[5]:

	name	marks	city
0	anant	21	Ahmedabad
1	bobby	34	Lko
2	celina	45	knp
3	danny	33	kolkata

In [6]:
```
df.to_csv("Friends.csv") #To export data to a new excel file
```

In [7]:
```
df.to_csv("Friends2.csv", index=False) #To export data to file without ind
```

In [8]:
```
df.head()
```

Out[8]:

	name	marks	city
0	anant	21	Ahmedabad
1	bobby	34	Lko
2	celina	45	knp
3	danny	33	kolkata

In [9]:
```
df.head(2)
```

Out[9]:

	name	marks	city
0	anant	21	Ahmedabad
1	bobby	34	Lko

In [10]:
```
df.tail()
```

Out[10]:

	name	marks	city
0	anant	21	Ahmedabad
1	bobby	34	Lko
2	celina	45	knp
3	danny	33	kolkata

In [11]:
```
df.tail(2)
```

Out[11]:

	name	marks	city
2	celina	45	knp
3	danny	33	kolkata

In [12]:
```
df.describe() #Gives statistical analysis of numerical columns
```

Out[12]:

	marks
count	4.000000
mean	33.250000
std	9.810708
min	21.000000
25%	30.000000
50%	33.500000
75%	36.750000
max	45.000000

In [13]:
```
df = pd.read_csv('dataset1.csv')
```

In [14]:
```
df
```

Out[14]:

	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	0	0	0	0	0	12345	100	Ahmedabad
1	1	1	1	1	1	12067	304	Lucknow
2	2	2	2	2	2	12087	45	Kanpur
3	3	3	3	3	3	13290	330	Kolkata
4	4	4	4	4	4	12637	210	Jammu
5	5	5	5	5	5	12845	190	Nagpur
6	6	6	6	6	6	12390	100	Palampur

In [15]:
```
df['Speed']
```

Out[15]:
```
0    100
1    304
2     45
3    330
4    210
5    190
6    100
Name: Speed, dtype: int64
```

In [16]:
```
df['Speed'][0]
```

Out[16]:
```
100
```

In [17]:
```
df['Speed'][0] = 100 #This way to update the value is not recommended. we u
```

```
<ipython-input-17-147559c6ed8f>: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
  df['Speed'][0] = 100 #This way to update the value is not recommended. we
use loc method
```

In [18]:
```
df['Speed'][0]
```

Out[18]:
```
100
```

In [19]:
```
df.to_csv('dataset1.csv') #Updated the value in the dataset1.csv file
```

In [20]:
```
df
```

Out[20]:

	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	0	0	0	0	0	12345	100	Ahmedabad
1	1	1	1	1	1	12067	304	Lucknow
2	2	2	2	2	2	12087	45	Kanpur
3	3	3	3	3	3	13290	330	Kolkata
4	4	4	4	4	4	12637	210	Jammu
5	5	5	5	5	5	12845	190	Nagpur
6	6	6	6	6	6	12390	100	Palampur

In [21]:
```
df.index = ['First', 'Second', 'Third', 'Fourth', 'Fifth', 'Sixth', 'Seven
```

In [22]:
```
df
```

Out[22]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Train No.	Speed	Cit
First	0	0	0	0	0	12345	100	Ahmedaba
Second	1	1	1	1	1	12067	304	Luckno
Third	2	2	2	2	2	12087	45	Kanpu
Fourth	3	3	3	3	3	13290	330	Kolkat
Fifth	4	4	4	4	4	12637	210	Jamm
Sixth	5	5	5	5	5	12845	190	Nagpu
Seventh	6	6	6	6	6	12390	100	Palampu

In [23]:
```
type(df)
```

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

In [24]:
```
type(df['Train No.'])
```

Out[24]:
```
pandas.core.series.Series
```

In [ ]:
```

```

In [25]:
```
list1 = [1,2,3,4,5,6,7,8,9]
```

In [26]:
```
ser = pd.Series(list1)
ser
```

Out[26]:
```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
```

In [27]:
```
ser1 = pd.Series(np.random.rand(11))
ser1
```

Out[27]:
```
0     0.296957
1     0.090852
2     0.830458
3     0.893584
4     0.039890
5     0.042769
6     0.066162
7     0.097233
8     0.448138
9     0.722560
10    0.612566
dtype: float64
```

In [28]:
```
type(ser1)
```

Out[28]:
```
pandas.core.series.Series
```

In [ ]:
```

```

In [29]:
```
newdf = pd.DataFrame(np.random.rand(301, 5), index = np.arange(301))
```

In [30]:
```
newdf
```

Out[30]:

	0	1	2	3	4
0	0.907607	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176
...	...	...	...	...	...
296	0.565956	0.142535	0.352416	0.648648	0.881349
297	0.722087	0.928463	0.823335	0.888897	0.162035
298	0.712653	0.453446	0.622997	0.959046	0.309309
299	0.080318	0.903879	0.030067	0.165670	0.469691
300	0.486524	0.503904	0.563623	0.681858	0.295950

301 rows × 5 columns

In [31]:
```
newdf.head()
```

Out[31]:

	0	1	2	3	4
0	0.907607	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

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

Out[32]:

	0	1	2	3	4
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	0.517538	0.517650	0.497146	0.529472	0.490285
std	0.286387	0.294479	0.282087	0.291290	0.290346
min	0.001572	0.010097	0.003731	0.001188	0.000152
25%	0.287228	0.269409	0.259438	0.292513	0.247710
50%	0.510080	0.527024	0.479553	0.541962	0.479786
75%	0.753036	0.797915	0.738755	0.761343	0.736899
max	0.996921	0.998032	0.985981	0.997328	0.999219

In [33]:
```
newdf.dtypes
```

Out[33]:
```
0    float64
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [34]:
```
newdf[0][0] = 'anant'
newdf.head()
```

Out[34]:

	0	1	2	3	4
0	anant	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [35]:
```
newdf.dtypes
```

Out[35]:
```
0     object
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [36]:
```
newdf.index
```

Out[36]:
```
Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
            ...
            291, 292, 293, 294, 295, 296, 297, 298, 299, 300],
           dtype='int64', length=301)
```

In [37]:
```
newdf.columns
```

Out[37]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [38]:
```
newdf[0][0] = 0.3 #Not a recommended wat to update value
```

```
<ipython-input-38-b6977304486d>: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
  newdf[0][0] = 0.3 #Not a recommended wat to update value
```

In [39]:
```
newdf.head()
```

Out[39]:

	0	1	2	3	4
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [40]:
```
newdf.to_numpy()
```

Out[40]:
```
array([[0.3, 0.2701587118056906, 0.4283526072575028, 0.5010411966711856,
        0.5065569506892139],
       [0.7647307985404265, 0.15880712636102268, 0.1203809118391087,
        0.06374850354608397, 0.9210386931889576],
       [0.9397272993066019, 0.7659373267792087, 0.04503312900121581,
        0.5217993435440882, 0.4564107118416326],
       ...,
       [0.7126531987152469, 0.4534457238500833, 0.6229971840196203,
        0.959046468926998, 0.3093088083661535],
       [0.08031785425355553, 0.9038794715560224, 0.030066873666714677,
        0.1656701187256664, 0.46969126663397776],
       [0.4865235151447531, 0.5039042474800163, 0.5636233614336897,
        0.6818584287476473, 0.29595019223080077]], dtype=object)
```

In [41]:
```
newdf.T #Transpose
```

Out[41]:

	0	1	2	3	4	5	6	7	
0	0.3	0.764731	0.939727	0.574922	0.335585	0.857899	0.971334	0.586492	0.73
1	0.270159	0.158807	0.765937	0.104767	0.852265	0.322121	0.382831	0.525181	0.35
2	0.428353	0.120381	0.045033	0.823047	0.30647	0.645693	0.98358	0.814233	0.8
3	0.501041	0.063749	0.521799	0.167228	0.845981	0.307572	0.095674	0.752958	0.41
4	0.506557	0.921039	0.456411	0.022926	0.046176	0.508184	0.641932	0.936798	0.398

5 rows × 301 columns

In [42]:

```
newdf.head()
```

Out[42]:

	0	1	2	3	4
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [43]:

```
newdf.sort_index(axis = 0) #axis = 0 is for row
```

Out[43]:

	0	1	2	3	4
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176
...	...	...	...	...	...
296	0.565956	0.142535	0.352416	0.648648	0.881349
297	0.722087	0.928463	0.823335	0.888897	0.162035
298	0.712653	0.453446	0.622997	0.959046	0.309309
299	0.080318	0.903879	0.030067	0.165670	0.469691
300	0.486524	0.503904	0.563623	0.681858	0.295950

301 rows × 5 columns

In [44]:
```
newdf.sort_index(axis = 1) #axis = 1 is for column
```

Out[44]:

	0	1	2	3	4
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176
...	...	...	...	...	...
296	0.565956	0.142535	0.352416	0.648648	0.881349
297	0.722087	0.928463	0.823335	0.888897	0.162035
298	0.712653	0.453446	0.622997	0.959046	0.309309
299	0.080318	0.903879	0.030067	0.165670	0.469691
300	0.486524	0.503904	0.563623	0.681858	0.295950

301 rows × 5 columns

In [45]:
```
newdf.sort_index(axis = 0, ascending = False)
```

Out[45]:

	0	1	2	3	4
300	0.486524	0.503904	0.563623	0.681858	0.295950
299	0.080318	0.903879	0.030067	0.165670	0.469691
298	0.712653	0.453446	0.622997	0.959046	0.309309
297	0.722087	0.928463	0.823335	0.888897	0.162035
296	0.565956	0.142535	0.352416	0.648648	0.881349
...	...	...	...	...	...
4	0.335585	0.852265	0.306470	0.845981	0.046176
3	0.574922	0.104767	0.823047	0.167228	0.022926
2	0.939727	0.765937	0.045033	0.521799	0.456411
1	0.764731	0.158807	0.120381	0.063749	0.921039
0	0.3	0.270159	0.428353	0.501041	0.506557

301 rows × 5 columns

In [46]:
```
newdf.sort_index(axis = 1, ascending = False)
```

Out[46]:

	4	3	2	1	0
0	0.506557	0.501041	0.428353	0.270159	0.3
1	0.921039	0.063749	0.120381	0.158807	0.764731
2	0.456411	0.521799	0.045033	0.765937	0.939727
3	0.022926	0.167228	0.823047	0.104767	0.574922
4	0.046176	0.845981	0.306470	0.852265	0.335585
...	...	...	...	...	...
296	0.881349	0.648648	0.352416	0.142535	0.565956
297	0.162035	0.888897	0.823335	0.928463	0.722087
298	0.309309	0.959046	0.622997	0.453446	0.712653
299	0.469691	0.165670	0.030067	0.903879	0.080318
300	0.295950	0.681858	0.563623	0.503904	0.486524

301 rows × 5 columns

In [47]:
```
newdf[0]
```

Out[47]:
```
0           0.3
1      0.764731
2      0.939727
3      0.574922
4      0.335585
         ...
296    0.565956
297    0.722087
298    0.712653
299    0.080318
300    0.486524
Name: 0, Length: 301, dtype: object
```

In [48]:
```
type(newdf[0])
```

Out[48]:
```
pandas.core.series.Series
```

In [49]:
```
type(newdf)
```

Out[49]:
```
pandas.core.frame.DataFrame
```

In [ ]:
```

```

In [50]:
```
newdf2 = newdf
```

In [51]:
```
newdf2[0][0] = 1234 #We changed the value in newdf2 but the value also git
                    #because newdf2 is a view of newdf
```

```
<ipython-input-51-a23717a8ec25>: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
  newdf2[0][0] = 1234 #We changed the value in newdf2 but the value also gi
t changed in newdf,
```

In [52]:
```
print(newdf[0][0])
print(newdf2[0][0])
```

```
1234
1234
```

In [53]:
```
newdf3 = newdf.copy()
```

In [54]:
```
newdf3[0][0] = 9999 #We changed the value in newdf3 and the value did not c
                    #because newdf3 is a copy(not a view) of newdf
```

```
<ipython-input-54-493d9df10842>: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
  newdf3[0][0] = 9999 #We changed the value in newdf3 and the value did not
changed in newdf,
```

In [55]:
```
print(newdf[0][0])
print(newdf3[0][0])
```

```
1234
9999
```

In [ ]:
```

```

In [56]:
```
#Right way to update value, using the loc function
newdf.loc[0,0] = 0.3
```

In [57]:
```
newdf.head()
```

Out[57]:

	0	1	2	3	4
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [58]:

```
newdf.columns = ['a', 'b', 'c', 'd', 'e']
```

In [59]:

```
newdf.head()
```

Out[59]:

	a	b	c	d	e
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [60]:

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

In [61]:

```
newdf.head()
```

Out[61]:

	A	B	C	D	E
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [62]:

```
#To create a new column
newdf.loc[:,'F'] = None
```

In [63]:

```
newdf.head()
```

Out[63]:

	A	B	C	D	E	F
0	0.3	0.270159	0.428353	0.501041	0.506557	None
1	0.764731	0.158807	0.120381	0.063749	0.921039	None
2	0.939727	0.765937	0.045033	0.521799	0.456411	None
3	0.574922	0.104767	0.823047	0.167228	0.022926	None
4	0.335585	0.852265	0.306470	0.845981	0.046176	None

In [64]:

```
newdf.tail()
```

Out[64]:

	A	B	C	D	E	F
296	0.565956	0.142535	0.352416	0.648648	0.881349	None
297	0.722087	0.928463	0.823335	0.888897	0.162035	None
298	0.712653	0.453446	0.622997	0.959046	0.309309	None
299	0.080318	0.903879	0.030067	0.165670	0.469691	None
300	0.486524	0.503904	0.563623	0.681858	0.295950	None

In [65]:

```
#To remove a column
newdf = newdf.drop('F', axis = 1) # axis = 1 for column
newdf.head()
```

Out[65]:

	A	B	C	D	E
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [66]:

```
newdf = newdf.drop(300, axis = 0)
newdf.tail()
```

Out[66]:

	A	B	C	D	E
295	0.907414	0.511383	0.817263	0.015517	0.419369
296	0.565956	0.142535	0.352416	0.648648	0.881349
297	0.722087	0.928463	0.823335	0.888897	0.162035
298	0.712653	0.453446	0.622997	0.959046	0.309309
299	0.080318	0.903879	0.030067	0.165670	0.469691

In [67]:

```
newdf.loc[[0,1], :]
```

Out[67]:

	A	B	C	D	E
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039

In [68]:

```
newdf.loc[:, ['A', 'B', 'C']]
```

Out[68]:

	A	B	C
0	0.3	0.270159	0.428353
1	0.764731	0.158807	0.120381
2	0.939727	0.765937	0.045033
3	0.574922	0.104767	0.823047
4	0.335585	0.852265	0.306470
...	...	...	...
295	0.907414	0.511383	0.817263
296	0.565956	0.142535	0.352416
297	0.722087	0.928463	0.823335
298	0.712653	0.453446	0.622997
299	0.080318	0.903879	0.030067

300 rows × 3 columns

In [69]:

```
newdf.loc[(newdf['A'] < 0.1) & (newdf['C'] > 0.9)]
```

Out[69]:

A	B	C	D	E

In [70]:

```
newdf.head()
```

Out[70]:

	A	B	C	D	E
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [71]:
```
newdf.iloc[0,4]
```

Out[71]: 0.5065569506892139

In [72]:
```
newdf.iloc[0,1]
```

Out[72]: 0.2701587118056906

In [73]:
```
newdf.iloc[0,0]
```

Out[73]: 0.3

In [74]:
```
newdf.iloc[[0,3],[1,2]]
```

Out[74]:

	B	C
0	0.270159	0.428353
3	0.104767	0.823047

In [75]:
```
newdf.iloc[[0,1,2,3],[0,1,2,3]]
```

Out[75]:

	A	B	C	D
0	0.3	0.270159	0.428353	0.501041
1	0.764731	0.158807	0.120381	0.063749
2	0.939727	0.765937	0.045033	0.521799
3	0.574922	0.104767	0.823047	0.167228

In [ ]:
```

```

In [76]:
```
newdf.head()
```

Out[76]:

	A	B	C	D	E
0	0.3	0.270159	0.428353	0.501041	0.506557
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176

In [77]:

```
newdf = newdf.drop([0], axis = 0)
newdf.head()
```

Out[77]:

	A	B	C	D	E
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176
5	0.857899	0.322121	0.645693	0.307572	0.508184

In [78]:

```
newdf.drop(['E'], axis = 1) #Here i have not done newdf = newdf.drop(['E'],
```

Out[78]:

	A	B	C	D
1	0.764731	0.158807	0.120381	0.063749
2	0.939727	0.765937	0.045033	0.521799
3	0.574922	0.104767	0.823047	0.167228
4	0.335585	0.852265	0.306470	0.845981
5	0.857899	0.322121	0.645693	0.307572
...	...	...	...	...
295	0.907414	0.511383	0.817263	0.015517
296	0.565956	0.142535	0.352416	0.648648
297	0.722087	0.928463	0.823335	0.888897
298	0.712653	0.453446	0.622997	0.959046
299	0.080318	0.903879	0.030067	0.165670

299 rows × 4 columns

In [79]:

```
newdf.head()
```

Out[79]:

	A	B	C	D	E
1	0.764731	0.158807	0.120381	0.063749	0.921039
2	0.939727	0.765937	0.045033	0.521799	0.456411
3	0.574922	0.104767	0.823047	0.167228	0.022926
4	0.335585	0.852265	0.306470	0.845981	0.046176
5	0.857899	0.322121	0.645693	0.307572	0.508184

In [80]:
```
newdf.drop(['E'], axis = 1, inplace = True) # Here i wrote inplace = True,
```

In [81]:
```
newdf.head()
```

Out[81]:

	A	B	C	D
1	0.764731	0.158807	0.120381	0.063749
2	0.939727	0.765937	0.045033	0.521799
3	0.574922	0.104767	0.823047	0.167228
4	0.335585	0.852265	0.306470	0.845981
5	0.857899	0.322121	0.645693	0.307572

In [82]:
```
newdf.drop([1,3,5], axis = 0, inplace = True)
```

In [83]:
```
newdf.head() #the index 1,3,5 got deleted
```

Out[83]:

	A	B	C	D
2	0.939727	0.765937	0.045033	0.521799
4	0.335585	0.852265	0.306470	0.845981
6	0.971334	0.382831	0.983580	0.095674
7	0.586492	0.525181	0.814233	0.752958
8	0.739272	0.350422	0.803500	0.419294

In [84]:
```
newdf.reset_index(drop = True, inplace = True)
```

In [85]:
```
newdf.head()
```

Out[85]:

	A	B	C	D
0	0.939727	0.765937	0.045033	0.521799
1	0.335585	0.852265	0.306470	0.845981
2	0.971334	0.382831	0.983580	0.095674
3	0.586492	0.525181	0.814233	0.752958
4	0.739272	0.350422	0.803500	0.419294

In [86]:
```
newdf.size
```

Out[86]: 1184

In [87]:
```
newdf.shape
```

Out[87]: (296, 4)

In [88]:
```
newdf.isnull()
```

Out[88]:

	A	B	C	D
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
291	False	False	False	False
292	False	False	False	False
293	False	False	False	False
294	False	False	False	False
295	False	False	False	False

296 rows × 4 columns

In [89]:
```
newdf.notnull()
```

Out[89]:

	A	B	C	D
0	True	True	True	True
1	True	True	True	True
2	True	True	True	True
3	True	True	True	True
4	True	True	True	True
...	...	...	...	...
291	True	True	True	True
292	True	True	True	True
293	True	True	True	True
294	True	True	True	True
295	True	True	True	True

296 rows × 4 columns

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

In [91]:
```
newdf.head()
```

Out[91]:

	A	B	C	D
0	0.939727	None	0.045033	0.521799
1	0.335585	None	0.306470	0.845981
2	0.971334	None	0.983580	0.095674
3	0.586492	None	0.814233	0.752958
4	0.739272	None	0.803500	0.419294

In [92]:
```
newdf['B'].isnull()
```

Out[92]:
```
0      True
1      True
2      True
3      True
4      True
       ...
291    True
292    True
293    True
294    True
295    True
Name: B, Length: 296, dtype: bool
```

In [ ]:
```
```

In [ ]:
```
```

In [93]:
```
df1 = pd.DataFrame({
    'name': ['Alfred', 'Batman', 'Alfred'],
    'toy' : [np.nan, np.nan, np.nan],
    'born': [pd.NaT, pd.Timestamp('2002-02-04'), pd.NaT]
})
```

In [94]:
```
df1
```

Out[94]:

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	NaN	2002-02-04
2	Alfred	NaN	NaT

In [95]:
```
df1.head()
```

Out[95]:

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	NaN	2002-02-04
2	Alfred	NaN	NaT

In [96]:
```
df1.dropna(how = 'all', axis = 1)
```

Out[96]:

	name	born
0	Alfred	NaT
1	Batman	2002-02-04
2	Alfred	NaT

In [97]:
```
df1.head()
```

Out[97]:

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	NaN	2002-02-04
2	Alfred	NaN	NaT

In [98]:
```
df1.drop_duplicates(subset = ['name'], keep = 'first')
```

Out[98]:

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	NaN	2002-02-04

In [99]:
```
df1.drop_duplicates(subset = ['name'], keep = 'last')
```

Out[99]:

	name	toy	born
1	Batman	NaN	2002-02-04
2	Alfred	NaN	NaT

In [100…
```
df1.drop_duplicates(subset = ['name'], keep = False)
```

Out[100…

	name	toy	born
1	Batman	NaN	2002-02-04

In [101…
```
df1.shape
```

Out[101… (3, 3)

In [102…
```
df1.size
```

Out[102… 9

In [103…
```
df1.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     0 non-null      float64
 2   born    1 non-null      datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 200.0+ bytes
```

In [104…

```
df1 = pd.DataFrame({
    'name': ['Alfred', 'Batman', 'Alfred'],
    'toy' : [np.nan, 'Batmobile', 'Bullwhip'],
    'born': [pd.NaT, pd.Timestamp('2002-02-04'), pd.NaT]
})
df1.head()
```

Out[104…

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	2002-02-04
2	Alfred	Bullwhip	NaT

In [105…

```
df1['name'].value_counts()
```

Out[105…

```
Alfred    2
Batman    1
Name: name, dtype: int64
```

In [106…

```
df1['toy'].value_counts()
```

Out[106…

```
Bullwhip     1
Batmobile    1
Name: toy, dtype: int64
```

In [107…

```
df1['toy'].value_counts(dropna = False)
```

Out[107…

```
NaN          1
Bullwhip     1
Batmobile    1
Name: toy, dtype: int64
```

In [108…

```
df1['toy'].value_counts(dropna = True)
```

Out[108…

```
Bullwhip     1
Batmobile    1
Name: toy, dtype: int64
```

In [109…

```
df1.isnull()
```

Out[109…

	name	toy	born
0	False	True	True
1	False	False	False
2	False	False	True

In [110…

```
df1.notnull()
```

Out[110…

	name	toy	born
0	True	False	False
1	True	True	True
2	True	True	False

In [111…

```
np_arr = np.array([[1,2],
                   [8,5],
                   [12,90]])
```

In [112…

```
df_quiz = pd.DataFrame(np_arr)
```

In [113…

```
df_quiz
```

Out[113…

	0	1
0	1	2
1	8	5
2	12	90

In [114…

```
df_quiz.columns = list('AB')
```

In [115…

```
df_quiz
```

Out[115…

	A	B
0	1	2
1	8	5
2	12	90

In [116…

```
df_quiz.describe()
```

Out[116…

	A	B
count	3.000000	3.000000
mean	7.000000	32.333333
std	5.567764	49.963320
min	1.000000	2.000000
25%	4.500000	3.500000
50%	8.000000	5.000000
75%	10.000000	47.500000
max	12.000000	90.000000

In [117…

```
#Pandas dataframe.corr() is used to find the pairwise correlation of all c
#Any na values are automatically excluded.
#For any non-numeric data type columns in the dataframe it is ignored.
df_quiz.corr()
```

Out[117…

	A	B
A	1.000000	0.796236
B	0.796236	1.000000

In [118…

```
df_quiz.count()
```

Out[118…

```
A    3
B    3
dtype: int64
```

In [119…

```
df_quiz.corr(method = 'pearson') #Explore it when required
```

Out[119…

	A	B
A	1.000000	0.796236
B	0.796236	1.000000

In [120…

```
df_quiz.corr(method = 'kendall') #Explore it when required
```

Out[120…

	A	B
A	1.0	1.0
B	1.0	1.0