Pandas (Python) tips and tricks

Written by: M.Danish Azeem Date: 15.11.2023 Email:m danishazeem365@gmail.com

01-How to find the version

```
import pandas as pd
pd.__version__
```

```
'2.1.3'
```

First i activate pandas libraries .Then i import pandas version.

```
# another way
pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit                : 2a953cf80b77e4348bf50ed724f8abc0d814d9dd
python                : 3.12.0.final.0
python-bits           : 64
OS                    : Windows
OS-release            : 10
Version               : 10.0.19045
machine               : AMD64
processor             : Intel64 Family 6 Model 42 Stepping 7,
GenuineIntel
byteorder             : little
LC_ALL                : None
LANG                  : None
LOCALE                : English_United States.1252

pandas                : 2.1.3
numpy                 : 1.26.1
pytz                  : 2023.3.post1
dateutil              : 2.8.2
setuptools            : 69.0.2
pip                   : 23.3.1
Cython                : None
pytest                : None
hypothesis            : None
sphinx                : None
blosc                 : None
feather               : None
xlsxwriter            : 3.1.9
```

```
lxml.etree           : None
html5lib             : None
pymysql              : None
psycopg2             : None
jinja2               : 3.1.2
IPython              : 8.18.1
pandas_datareader    : None
bs4                  : 4.12.2
bottleneck           : None
dataframe-api-compat: None
fastparquet          : None
fsspec               : None
gcsfs                : None
matplotlib           : 3.8.1
numba                : None
numexpr              : None
odfpy                : None
openpyxl             : 3.1.2
pandas_gbq           : None
pyarrow              : None
pyreadstat           : None
pyxlsb               : None
s3fs                 : None
scipy                : None
sqlalchemy           : None
tables               : None
tabulate             : None
xarray               : None
xlrd                 : None
zstandard            : None
tzdata               : 2023.3
qtpy                 : None
pyqt5                : None
```

Then i type pd.show_versions() to get all information about pandas

02- Make a dataframe

```
df = pd.DataFrame({'A col': [1,2,3,6,7,8,9,22], 'B col':
[4,5,6,7,8,9,10,11]})

df.head()
```

```
   A col  B col
0      1      4
1      2      5
2      3      6
```

```
3      6      7
4      7      8
```

Secondly I use this command (f = pd.DataFrame({'A col': [1,2,3,6,7,8,9,22], 'B col': [4,5,6,7,8,9,10,11]}))to creat datafram and give the name df and run as df.head()

```
# numpy array use to creat dataframe
import numpy as np
arr = np.array([[1,2,3], [4,5,6], [7,8,9]])
pd.DataFrame(arr)
```

```
   0  1  2
0  1  2  3
1  4  5  6
2  7  8  9
```

Then i import numpy libraries after this i creat a data frame with numpy and called it arr veriable by using (arr = np.array([[1,2,3], [4,5,6], [7,8,9]]) pd.DataFrame(arr)) then run the pd.DataFrame(arr) command for executing and got the result.

```
# numpy array dataframe
pd.DataFrame(np.random.rand(4,8))
```

```
          0         1         2         3         4         5
6  \
0  0.015423  0.753582  0.735567  0.614379  0.643070  0.260783
0.629006
1  0.828775  0.953675  0.355998  0.691969  0.352886  0.511967
0.081660
2  0.547072  0.873657  0.177309  0.400991  0.344104  0.623305
0.103849
3  0.691643  0.730221  0.446508  0.982548  0.142146  0.476878
0.188509

          7
0  0.281331
1  0.672790
2  0.758074
3  0.642856
```

After this i creat numpy randem data by using this command (pd.DataFrame(np.random.rand(4,8)))

```
pd.DataFrame(np.random.rand(4,8), columns=list('ABCDEFGH'))
```

```
          A         B         C         D         E         F
G  \
0  0.799503  0.054151  0.672972  0.900114  0.406323  0.452236
0.991435
```

```
1  0.300219  0.399405  0.775128  0.566473  0.601738  0.793357
0.784099
2  0.021397  0.142346  0.303766  0.201643  0.912564  0.451718
0.175032
3  0.166071  0.591816  0.001913  0.615207  0.825256  0.754079
0.024260

          H
0  0.364483
1  0.329612
2  0.893666
3  0.643891
```

After this i gave them the columns names A to H

```
pd.DataFrame(np.random.rand(12,12), columns=list('ABCDEFGHIJkl'))
```

```
           A         B         C         D         E         F
G  \
0   0.224451  0.279393  0.700657  0.297379  0.953304  0.059169
0.128473
1   0.436424  0.402002  0.992561  0.411057  0.044430  0.617181
0.463158
2   0.008602  0.783771  0.755995  0.478794  0.158651  0.783269
0.092821
3   0.458654  0.630011  0.910797  0.631525  0.340917  0.804172
0.375946
4   0.183630  0.929450  0.380698  0.245292  0.303352  0.076697
0.007698
5   0.227443  0.603990  0.003279  0.845764  0.855238  0.270000
0.705923
6   0.484610  0.362661  0.590745  0.125925  0.625649  0.522931
0.404418
7   0.098684  0.739038  0.356347  0.258675  0.574251  0.650143
0.989048
8   0.002744  0.193246  0.104326  0.086131  0.308746  0.590508
0.010634
9   0.972185  0.744513  0.599353  0.160428  0.554060  0.891110
0.526075
10  0.359167  0.504568  0.117822  0.070028  0.858392  0.933971
0.066429
11  0.029457  0.771432  0.592714  0.406565  0.076624  0.877779
0.625654

           H         I         J         k         l
0   0.393869  0.130054  0.660988  0.752216  0.462130
1   0.588608  0.114299  0.896735  0.466171  0.104740
2   0.440202  0.386247  0.648628  0.832276  0.447197
3   0.633147  0.098400  0.293153  0.514551  0.721657
```

```
4   0.027481  0.597750  0.274986  0.650281  0.290965
5   0.273979  0.969118  0.357245  0.955498  0.026553
6   0.375179  0.800547  0.066520  0.199027  0.300844
7   0.216588  0.524565  0.894849  0.876771  0.356899
8   0.234550  0.558645  0.893774  0.221157  0.685990
9   0.074780  0.421753  0.402027  0.252822  0.495635
10  0.066921  0.212568  0.794781  0.553791  0.670749
11  0.135556  0.597930  0.957107  0.877430  0.823167
```

3- How to rename columns?

```
import pandas as pd
df.rename(columns={'A col': 'col_aa', 'B col': 'col_bb'},
inplace=True)
df
```

```
   col_aa  col_bb
0       1       4
1       2       5
2       3       6
3       6       7
4       7       8
5       8       9
6       9      10
7      22      11
```

Thirdly again I activate pandas librarie and rename columns by using function and type df function to get result

```
# rename columns
df.columns=['col_a', 'col_b']
df
```

```
   col_a  col_b
0      1      4
1      2      5
2      3      6
3      6      7
4      7      8
5      8      9
6      9     10
7     22     11
```

We also rename columns by using this function.

```
# to replace any character, string
```

```
df.columns=df.columns.str.replace('_' , '  ')
df
```

```
   col  a  col  b
0       1       4
1       2       5
2       3       6
3       6       7
4       7       8
5       8       9
6       9      10
7      22      11
```

To replace any character and string we use this command df.columns=df.columns.str.replace('_' , ' ')df and run df function

```
df.columns=df.columns.str.replace(' ' , '_')
df
```

```
   col__a  col__b
0       1       4
1       2       5
2       3       6
3       6       7
4       7       8
5       8       9
6       9      10
7      22      11
```

```
# Adding Prefix to coiumns
df = df.add_prefix('baba_')
df
```

```
   baba_col__a  baba_col__b
0            1            4
1            2            5
2            3            6
3            6            7
4            7            8
5            8            9
6            9           10
7           22           11
```

We can add prefix by using this command df = df.add_prefix('baba_') df and run the function by using df

```
# Adding suffix to coiumns
df = df.add_suffix('_baba')
df
```

```
   baba_col__a_baba  baba_col__b_baba
0                 1                 4
1                 2                 5
2                 3                 6
3                 6                 7
4                 7                 8
5                 8                 9
6                 9                10
7                22                11
```

To add suffix to columns we use df = df.add_suffix('_baba') df function and run the df

```
df.columns = ['col_a', 'col_b']
df
```

```
   col_a  col_b
0      1      4
1      2      5
2      3      6
3      6      7
4      7      8
5      8      9
6      9     10
7     22     11
```

After this we convert it its normal coulmns name

4- Using template data

```
import pandas as pd
import numpy as np
import seaborn as sns

df = sns.load_dataset('tips')
df.head()
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

Fourthly i import pandas,numpy and seaborn libraries.After this i load dataset of seaborn by usin function and call it as a df and run the df.head() and got this dataset.

```
# Summery
df.describe()
```

```
# columns names
df.columns
```

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'],
dtype='object')
```

We can summarize data by using df.describe() function and column names as df,columns

```
# saving a dataset
df.to_csv('tips_save.csv')
#pip install openpyxl
df.to_excel('tips_save.xlsx')
```

After this we convert seaborn dataset into csv and excel formet by using these functions or commands

5- Using your own data

```
import pandas as pd
df = pd.read_csv('tips_save.csv')
df.head()
#df = pd.read_excel('tips_save.xlsx')
```

```
   Unnamed: 0  total_bill   tip     sex smoker  day    time  size
0           0       16.99  1.01  Female     No  Sun  Dinner     2
1           1       10.34  1.66    Male     No  Sun  Dinner     3
2           2       21.01  3.50    Male     No  Sun  Dinner     3
3           3       23.68  3.31    Male     No  Sun  Dinner     2
4           4       24.59  3.61  Female     No  Sun  Dinner     4
```

Lastly i import pandas librarie and import csv and excel file from folder by using df = pd.read_csv('tips_save.csv') df.head()

#df = pd.read_excel('tips_save.xlsx') functions and run them for use.

6- Reverse Row Order

```
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')
df.head(-6)
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0           0       3    male  22.0      1      0   7.2500        S
```

```
Third
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
4           0       3    male  35.0      0      0   8.0500        S
Third
..        ...     ...     ...   ...    ...    ...      ...      ...
...
880         1       2  female  25.0      0      1  26.0000        S
Second
881         0       3    male  33.0      0      0   7.8958        S
Third
882         0       3  female  22.0      0      0  10.5167        S
Third
883         0       2    male  28.0      0      0  10.5000        S
Second
884         0       3    male  25.0      0      0   7.0500        S
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
880  woman       False  NaN  Southampton   yes  False
881    man        True  NaN  Southampton    no   True
882  woman       False  NaN  Southampton    no   True
883    man        True  NaN  Southampton    no   True
884    man        True  NaN  Southampton    no   True

[885 rows x 15 columns]
```

```
df.loc[::-1].head()
```

```
     survived  pclass     sex   age  sibsp  parch   fare embarked
class  \
890         0       3    male  32.0      0      0   7.75        Q
Third
889         1       1    male  26.0      0      0  30.00        C
First
888         0       3  female   NaN      1      2  23.45        S
Third
887         1       1  female  19.0      0      0  30.00        S
First
886         0       2    male  27.0      0      0  13.00        S
```

```
Second

       who  adult_male deck  embark_town alive  alone
890    man        True  NaN   Queenstown    no   True
889    man        True    C    Cherbourg   yes   True
888  woman       False  NaN  Southampton    no  False
887  woman       False    B  Southampton   yes   True
886    man        True  NaN  Southampton    no   True

df.loc[::-1].reset_index(drop=True).head()

   survived  pclass     sex   age  sibsp  parch   fare embarked
class  \
0         0       3    male  32.0      0      0   7.75        Q
Third
1         1       1    male  26.0      0      0  30.00        C
First
2         0       3  female   NaN      1      2  23.45        S
Third
3         1       1  female  19.0      0      0  30.00        S
First
4         0       2    male  27.0      0      0  13.00        S
Second

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN   Queenstown    no   True
1    man        True    C    Cherbourg   yes   True
2  woman       False  NaN  Southampton    no  False
3  woman       False    B  Southampton   yes   True
4    man        True  NaN  Southampton    no   True
```

7-Reverse Column order

```
df.loc[:,::-1].head()

   alone alive  embark_town deck  adult_male    who  class embarked
fare  \
0  False    no  Southampton  NaN        True    man  Third        S
7.2500
1  False   yes    Cherbourg    C       False  woman  First        C
71.2833
2   True   yes  Southampton  NaN       False  woman  Third        S
7.9250
3  False   yes  Southampton    C       False  woman  First        S
53.1000
4   True    no  Southampton  NaN        True    man  Third        S
8.0500
```

```
   parch  sibsp   age     sex  pclass  survived
0      0      1  22.0    male       3         0
1      0      1  38.0  female       1         1
2      0      0  26.0  female       3         1
3      0      1  35.0  female       1         1
4      0      0  35.0    male       3         0
```

8-select a column by dtype

```
df.dtypes
```

```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

```
# only select those have numeric type
df.select_dtypes(include=['number']).head()
```

```
   survived  pclass   age  sibsp  parch     fare
0         0       3  22.0      1      0   7.2500
1         1       1  38.0      1      0  71.2833
2         1       3  26.0      0      0   7.9250
3         1       1  35.0      1      0  53.1000
4         0       3  35.0      0      0   8.0500
```

```
# only select those have object type
df.select_dtypes(include=['object']).head()
```

```
      sex embarked    who  embark_town alive
0    male        S    man  Southampton    no
1  female        C  woman    Cherbourg   yes
2  female        S  woman  Southampton   yes
3  female        S  woman  Southampton   yes
4    male        S    man  Southampton    no
```

```
# only select those have multiple type
df.select_dtypes(include=['object', 'number', 'category']).head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who deck  embark_town alive
0    man  NaN  Southampton    no
1  woman    C    Cherbourg   yes
2  woman  NaN  Southampton   yes
3  woman    C  Southampton   yes
4    man  NaN  Southampton    no
```

```
df.select_dtypes(exclude=['object']).head()
```

```
   survived  pclass   age  sibsp  parch     fare  class  adult_male
deck  \
0         0       3  22.0      1      0   7.2500  Third        True
NaN
1         1       1  38.0      1      0  71.2833  First       False
C
2         1       3  26.0      0      0   7.9250  Third       False
NaN
3         1       1  35.0      1      0  53.1000  First       False
C
4         0       3  35.0      0      0   8.0500  Third        True
NaN

   alone
0  False
1  False
2   True
3  False
4   True
```

9- Convert strings to number

```
df = pd.DataFrame({'col_A':
['1.2','2','3','6','7','8','9','22'],'col_B':
['4','5','6','7','8','9','10','11']})
df
```

```
  col_A col_B
0   1.2     4
1     2     5
2     3     6
3     6     7
4     7     8
5     8     9
6     9    10
7    22    11
```

```
df.dtypes
```

```
col_A    object
col_B    object
dtype: object
```

```
df.astype({'col_A': 'float64', 'col_B': 'int64'}).dtypes
```

```
col_A    float64
col_B      int64
dtype: object
```

```
pd.to_numeric(df['col_A'], errors='coerce')
pd.to_numeric(df['col_B'], errors='coerce')
```

```
0     4
1     5
2     6
3     7
4     8
5     9
6    10
7    11
Name: col_B, dtype: int64
```

```
df.dtypes
```

```
col_A    object
col_B    object
dtype: object
```

10- Reduce dataframe size

```
df = sns.load_dataset('titanic')
df.shape
```

```
(891, 15)
```

```
df.sample(frac=0.1).shape
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

11-copy data from clip board

```
# dataset download
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')
df.to_excel('kashti.xlsx')

# read clipboard in python
df= pd.read_clipboard()
df.head()
df.to_csv('execfile.csv')
```

12- split data frame into two subsets

```
# dataset download
import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
df.head
```

```
<bound method NDFrame.head of      survived  pclass     sex   age
sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S
Third
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
4           0       3    male  35.0      0      0   8.0500        S
Third
..        ...     ...     ...   ...    ...    ...      ...      ...
...
886         0       2    male  27.0      0      0  13.0000        S
Second
887         1       1  female  19.0      0      0  30.0000        S
First
888         0       3  female   NaN      1      2  23.4500        S
Third
889         1       1    male  26.0      0      0  30.0000        C
First
890         0       3    male  32.0      0      0   7.7500        Q
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]>
```

```
len(df)
```

```
891
```

```
df.shape
```

```
(891, 15)
```

```
from random import random
kashti_1 = df.sample(frac=0.50, random_state=1)
kashti_1.shape
```

```
(446, 15)
```

```
kashti_2 = df.drop(kashti_1.index)
kashti_2.shape
```

```
(445, 15)
```

```
kashti_1.head()
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
862         1       1  female  48.0      0      0  25.9292        S
First
223         0       3    male   NaN      0      0   7.8958        S
Third
84          1       2  female  17.0      0      0  10.5000        S
Second
680         0       3  female   NaN      0      0   8.1375        Q
Third
535         1       2  female   7.0      0      2  26.2500        S
Second

       who  adult_male deck  embark_town alive  alone
862  woman       False    D  Southampton   yes   True
223    man        True  NaN  Southampton    no   True
84   woman       False  NaN  Southampton   yes   True
680  woman       False  NaN   Queenstown    no   True
535  child       False  NaN  Southampton   yes  False
```

```
kashti_2.head()
```

```
    survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
1          1       1  female  38.0      1      0  71.2833        C
First
7          0       3    male   2.0      3      1  21.0750        S
Third
10         1       3  female   4.0      1      1  16.7000        S
Third
15         1       2  female  55.0      0      0  16.0000        S
Second
18         0       3  female  31.0      1      0  18.0000        S
```

```
Third

      who  adult_male deck  embark_town alive  alone
1   woman       False    C    Cherbourg   yes  False
7   child       False  NaN  Southampton    no  False
10  child       False    G  Southampton   yes  False
15  woman       False  NaN  Southampton   yes   True
18  woman       False  NaN  Southampton    no  False

df.shape

(891, 15)

len(kashti_1) + len(kashti_2)

891
```

13- join two datasets

```
import pandas as pd

# assuming kashti_1 and kashti_2 are dataframes
df1 = pd.concat([kashti_1, kashti_2], ignore_index=True)

# Now you can print the shape of the new dataframe
df1.shape

(891, 15)
```

14- Filtering a dataset

```
df.head()

   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
```

```
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
df.sex.unique()
```

```
array(['male', 'female'], dtype=object)
```

```
df[(df.sex=="female")]
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
8           1       3  female  27.0      0      2  11.1333        S
Third
9           1       2  female  14.0      1      0  30.0708        C
Second
..        ...     ...     ...   ...    ...    ...      ...      ...
...
880         1       2  female  25.0      0      1  26.0000        S
Second
882         0       3  female  22.0      0      0  10.5167        S
Third
885         0       3  female  39.0      0      5  29.1250        Q
Third
887         1       1  female  19.0      0      0  30.0000        S
First
888         0       3  female   NaN      1      2  23.4500        S
Third

       who  adult_male deck  embark_town alive  alone
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
8    woman       False  NaN  Southampton   yes  False
9    child       False  NaN    Cherbourg   yes  False
..     ...         ...  ...          ...   ...    ...
880  woman       False  NaN  Southampton   yes  False
882  woman       False  NaN  Southampton    no   True
885  woman       False  NaN   Queenstown    no  False
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
```

```
[314 rows x 15 columns]
```

```
print(df.embark_town.unique())
df[(df.embark_town=='southampton')].shape
```

```
['Southampton' 'Cherbourg' 'Queenstown' nan]
```

```
(0, 15)
```

```
df[((df.embark_town=='southampton')|
    (df.embark_town=='Queenstown')) &
    (df.sex=="female")]
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
22          1       3  female  15.0      0      0   8.0292        Q
Third
28          1       3  female   NaN      0      0   7.8792        Q
Third
32          1       3  female   NaN      0      0   7.7500        Q
Third
44          1       3  female  19.0      0      0   7.8792        Q
Third
47          1       3  female   NaN      0      0   7.7500        Q
Third
82          1       3  female   NaN      0      0   7.7875        Q
Third
109         1       3  female   NaN      1      0  24.1500        Q
Third
156         1       3  female  16.0      0      0   7.7333        Q
Third
186         1       3  female   NaN      1      0  15.5000        Q
Third
198         1       3  female   NaN      0      0   7.7500        Q
Third
208         1       3  female  16.0      0      0   7.7500        Q
Third
241         1       3  female   NaN      1      0  15.5000        Q
Third
264         0       3  female   NaN      0      0   7.7500        Q
Third
274         1       3  female   NaN      0      0   7.7500        Q
Third
289         1       3  female  22.0      0      0   7.7500        Q
Third
300         1       3  female   NaN      0      0   7.7500        Q
Third
303         1       2  female   NaN      0      0  12.3500        Q
Second
```

```
322          1       2  female  30.0      0      0  12.3500         Q
Second
330          1       3  female   NaN      2      0  23.2500         Q
Third
358          1       3  female   NaN      0      0   7.8792         Q
Third
359          1       3  female   NaN      0      0   7.8792         Q
Third
368          1       3  female   NaN      0      0   7.7500         Q
Third
412          1       1  female  33.0      1      0  90.0000         Q
First
501          0       3  female  21.0      0      0   7.7500         Q
Third
502          0       3  female   NaN      0      0   7.6292         Q
Third
573          1       3  female   NaN      0      0   7.7500         Q
Third
593          0       3  female   NaN      0      2   7.7500         Q
Third
612          1       3  female   NaN      1      0  15.5000         Q
Third
653          1       3  female   NaN      0      0   7.8292         Q
Third
654          0       3  female  18.0      0      0   6.7500         Q
Third
657          0       3  female  32.0      1      1  15.5000         Q
Third
680          0       3  female   NaN      0      0   8.1375         Q
Third
697          1       3  female   NaN      0      0   7.7333         Q
Third
727          1       3  female   NaN      0      0   7.7375         Q
Third
767          0       3  female  30.5      0      0   7.7500         Q
Third
885          0       3  female  39.0      0      5  29.1250         Q
Third

       who  adult_male deck embark_town alive  alone
22   child       False  NaN  Queenstown   yes   True
28   woman       False  NaN  Queenstown   yes   True
32   woman       False  NaN  Queenstown   yes   True
44   woman       False  NaN  Queenstown   yes   True
47   woman       False  NaN  Queenstown   yes   True
82   woman       False  NaN  Queenstown   yes   True
109  woman       False  NaN  Queenstown   yes  False
156  woman       False  NaN  Queenstown   yes   True
186  woman       False  NaN  Queenstown   yes  False
```

```
198  woman       False  NaN  Queenstown   yes   True
208  woman       False  NaN  Queenstown   yes   True
241  woman       False  NaN  Queenstown   yes  False
264  woman       False  NaN  Queenstown    no   True
274  woman       False  NaN  Queenstown   yes   True
289  woman       False  NaN  Queenstown   yes   True
300  woman       False  NaN  Queenstown   yes   True
303  woman       False    E  Queenstown   yes   True
322  woman       False  NaN  Queenstown   yes   True
330  woman       False  NaN  Queenstown   yes  False
358  woman       False  NaN  Queenstown   yes   True
359  woman       False  NaN  Queenstown   yes   True
368  woman       False  NaN  Queenstown   yes   True
412  woman       False    C  Queenstown   yes  False
501  woman       False  NaN  Queenstown    no   True
502  woman       False  NaN  Queenstown    no   True
573  woman       False  NaN  Queenstown   yes   True
593  woman       False  NaN  Queenstown    no  False
612  woman       False  NaN  Queenstown   yes  False
653  woman       False  NaN  Queenstown   yes   True
654  woman       False  NaN  Queenstown    no   True
657  woman       False  NaN  Queenstown    no  False
680  woman       False  NaN  Queenstown    no   True
697  woman       False  NaN  Queenstown   yes   True
727  woman       False  NaN  Queenstown   yes   True
767  woman       False  NaN  Queenstown    no   True
885  woman       False  NaN  Queenstown    no  False
```

```
df[df.embark_town.isin(['Queenstown', 'Southampton'])].head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third
5         0       3    male   NaN      0      0   8.4583        Q
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
5    man        True  NaN   Queenstown    no   True
```

```
df[df.age < 18].shape
```

```
(113, 15)
```

15- Filtering by large categories

```
df.age.value_counts().nlargest(3)
```

```
age
24.0    30
22.0    27
18.0    26
Name: count, dtype: int64
```

```
counts = df.who.value_counts()
counts.nlargest(3)
```

```
who
man      537
woman    271
child     83
Name: count, dtype: int64
```

```
df[df.who.isin(counts.nlargest(2).index)].head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

16- Splitting a string into multiple coulumns

```
# import libraries
import pandas as pd
df = pd.DataFrame({'Name':['Ahmad Raza', 'Ali Afzal', 'Sajjad Ali',
'Abu Bakkar'],
                   'location':['Lahore, pakistan','Sargodah,
Pakistan', 'Karachi, Pakistan','Hamburg, Germany']})
df
```

```
         Name             location
0  Ahmad Raza    Lahore, pakistan
1   Ali Afzal  Sargodah, Pakistan
2  Sajjad Ali   Karachi, Pakistan
3  Abu Bakkar    Hamburg, Germany
```

```
# spliting a columns into two columns
df.Name.str.split(' ', expand=True)
```

```
        0       1
0   Ahmad    Raza
1     Ali   Afzal
2  Sajjad     Ali
3     Abu  Bakkar
```

```
# addding those splits into new columns
df[["first_Name", "last_Name"]] = df.Name.str.split(' ', expand=True)
df
```

```
         Name             location first_Name last_Name
0  Ahmad Raza    Lahore, pakistan      Ahmad      Raza
1   Ali Afzal  Sargodah, Pakistan        Ali     Afzal
2  Sajjad Ali   Karachi, Pakistan     Sajjad       Ali
3  Abu Bakkar    Hamburg, Germany        Abu    Bakkar
```

```
df[["city", "country"]] = df.location.str.split(', ', expand=True)
df
```

```
         Name             location first_Name last_Name      city
country
0  Ahmad Raza    Lahore, pakistan      Ahmad      Raza    Lahore
pakistan
1   Ali Afzal  Sargodah, Pakistan        Ali     Afzal  Sargodah
Pakistan
2  Sajjad Ali   Karachi, Pakistan     Sajjad       Ali   Karachi
Pakistan
3  Abu Bakkar    Hamburg, Germany        Abu    Bakkar   Hamburg
Germany
```

```
# Refine data manipulation
df = df[['first_Name', 'last_Name', 'city', 'country']]
df
```

```
  first_Name last_Name      city   country
0      Ahmad      Raza    Lahore  pakistan
1        Ali     Afzal  Sargodah  Pakistan
2     Sajjad       Ali   Karachi  Pakistan
3        Abu    Bakkar   Hamburg   Germany
```

17- Aggregate by multiple groups/function

```
# libraries
import pandas as pd
import seaborn as sns\

# import data base
df = sns.load_dataset('titanic')
df.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
df.groupby('sex').count()
```

```
        survived  pclass  age  sibsp  parch  fare  embarked  class
who  \
sex

female       314     314  261    314    314   314       312    314
314
male         577     577  453    577    577   577       577    577
```

```
577
```

```
        adult_male  deck  embark_town  alive  alone
sex
female         314    97          312    314    314
male           577   106          577    577    577
```

```
df.groupby('who').count()
```

```
       survived  pclass  sex  age  sibsp  parch  fare  embarked  class
\
who

child        83      83   83   83     83     83    83        83     83

man         537     537  537  413    537    537   537       537    537

woman       271     271  271  218    271    271   271       269    271


       adult_male  deck  embark_town  alive  alone
who
child          83    13           83     83     83
man           537    99          537    537    537
woman         271    91          269    271    271
```

```
len(df.groupby('pclass'))
```

```
3
```

```
df.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third
```

```
     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
df.groupby(['sex','pclass','embarked']).count()
```

sex	pclass	embarked	survived	age	sibsp	parch	fare	class	who
female	1	C	43	38	43	43	43	43	43
		Q	1	1	1	1	1	1	1
		S	48	44	48	48	48	48	48
	2	C	7	7	7	7	7	7	7
		Q	2	1	2	2	2	2	2
		S	67	66	67	67	67	67	67
	3	C	23	16	23	23	23	23	23
		Q	33	10	33	33	33	33	33
		S	88	76	88	88	88	88	88
male	1	C	42	36	42	42	42	42	42
		Q	1	1	1	1	1	1	1
		S	79	64	79	79	79	79	79
	2	C	10	8	10	10	10	10	10
		Q	1	1	1	1	1	1	1
		S	97	90	97	97	97	97	97
	3	C	43	25	43	43	43	43	43
		Q	39	14	39	39	39	39	39
		S	265	214	265	265	265	265	265

sex	pclass	embarked	adult_male	deck	embark_town	alive	alone
female	1	C	43	35	43	43	43
		Q	1	1	1	1	1
		S	48	43	48	48	48
	2	C	7	1	7	7	7
		Q	2	1	2	2	2
		S	67	8	67	67	67

```
       3      C                23     1           23     23     23
              Q                33     0           33     33     33
              S                88     5           88     88     88
male   1      C                42    31           42     42     42
              Q                 1     1            1      1      1
              S                79    62           79     79     79
       2      C                10     1           10     10     10
              Q                 1     0            1      1      1
              S                97     5           97     97     97
       3      C                43     0           43     43     43
              Q                39     1           39     39     39
              S               265     5          265    265    265
```

18- select specific rows or column

```
df

     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0           0       3    male  22.0      1      0   7.2500        S
Third
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
4           0       3    male  35.0      0      0   8.0500        S
Third
..        ...     ...     ...   ...    ...    ...      ...      ...
...
886         0       2    male  27.0      0      0  13.0000        S
Second
887         1       1  female  19.0      0      0  30.0000        S
First
888         0       3  female   NaN      1      2  23.4500        S
Third
889         1       1    male  26.0      0      0  30.0000        C
First
890         0       3    male  32.0      0      0   7.7500        Q
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
```

```
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

```
df.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
# select coilumns
df[['sex', 'class', 'deck']]
```

```
        sex   class deck
0      male   Third  NaN
1    female   First    C
2    female   Third  NaN
3    female   First    C
4      male   Third  NaN
..      ...     ...  ...
886    male  Second  NaN
887  female   First    B
888  female   Third  NaN
889    male   First    C
890    male   Third  NaN

[891 rows x 3 columns]
```

```
df.describe()
```

```
         survived      pclass         age       sibsp       parch
fare
count  891.000000  891.000000  714.000000  891.000000  891.000000
891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594
32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057
49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000
0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000
7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000
14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000
31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000
512.329200
```

```
# df.describe().loc[['min', '25%', '50%', '75%', 'max']]
# or
df.describe().loc['min': 'max']
```

```
     survived  pclass     age  sibsp  parch      fare
min       0.0     1.0   0.420    0.0    0.0    0.0000
25%       0.0     2.0  20.125    0.0    0.0    7.9104
50%       0.0     3.0  28.000    0.0    0.0   14.4542
75%       1.0     3.0  38.000    1.0    0.0   31.0000
max       1.0     3.0  80.000    8.0    6.0  512.3292
```

```
df.describe().loc['min': 'max', 'survived':'age']
```

```
     survived  pclass     age
min       0.0     1.0   0.420
25%       0.0     2.0  20.125
50%       0.0     3.0  28.000
75%       1.0     3.0  38.000
max       1.0     3.0  80.000
```

19- Reshape multi index series

```
df.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
```

```
2          1       3  female  26.0      0      0   7.9250        S
Third
3          1       1  female  35.0      1      0  53.1000        S
First
4          0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
df.survived.mean()
```

```
0.3838383838383838
```

```
df.groupby('sex').survived.mean()
```

```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

```
# df.groupby(['sex', 'class']).survived.mean()
df.groupby(['sex', 'pclass']).survived.mean()
```

```
sex     pclass
female  1         0.968085
        2         0.921053
        3         0.500000
male    1         0.368852
        2         0.157407
        3         0.135447
Name: survived, dtype: float64
```

20- Continuous to catogirical data conversion

```
df.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
```

```
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True

df.age.head()

0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64

pd.cut(df.age, bins = [0, 18, 25, 99], labels=['child' ,'young_adult',
'adult']).head()
df['new_age'] = pd.cut(df.age, bins = [0, 18, 25, 99], labels=['child'
,'young_adult', 'adult'])
df.head()

   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone      new_age
0    man        True  NaN  Southampton    no  False  young_adult
1  woman       False    C    Cherbourg   yes  False        adult
2  woman       False  NaN  Southampton   yes   True        adult
3  woman       False    C  Southampton   yes  False        adult
4    man        True  NaN  Southampton    no   True        adult
```

21- convert one set of values into anotherrone

```
df.sex.head()

0      male
1    female
2    female
3    female
4      male
Name: sex, dtype: object

df['sex_num'] = df.sex.map({'male':0, 'female':1})
df.head()

   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

     who  adult_male deck  embark_town alive  alone      new_age
sex_num
0    man        True  NaN  Southampton    no  False  young_adult
0
1  woman       False    C    Cherbourg   yes  False        adult
1
2  woman       False  NaN  Southampton   yes   True        adult
1
3  woman       False    C  Southampton   yes  False        adult
1
4    man        True  NaN  Southampton    no   True        adult
0

df.embarked.unique()

array(['S', 'C', 'Q', nan], dtype=object)

df['embarked'] = df.embarked.factorize()[0]
df.head(15)

    survived  pclass     sex   age  sibsp  parch     fare  embarked
class  \
0          0       3    male  22.0      1      0   7.2500         0
Third
```

```
1          1       1  female  38.0      1      0  71.2833          1
First
2          1       3  female  26.0      0      0   7.9250          0
Third
3          1       1  female  35.0      1      0  53.1000          0
First
4          0       3    male  35.0      0      0   8.0500          0
Third
5          0       3    male   NaN      0      0   8.4583          2
Third
6          0       1    male  54.0      0      0  51.8625          0
First
7          0       3    male   2.0      3      1  21.0750          0
Third
8          1       3  female  27.0      0      2  11.1333          0
Third
9          1       2  female  14.0      1      0  30.0708          1
Second
10         1       3  female   4.0      1      1  16.7000          0
Third
11         1       1  female  58.0      0      0  26.5500          0
First
12         0       3    male  20.0      0      0   8.0500          0
Third
13         0       3    male  39.0      1      5  31.2750          0
Third
14         0       3  female  14.0      0      0   7.8542          0
Third

      who  adult_male deck  embark_town alive  alone      new_age
sex_num
0     man        True  NaN  Southampton    no  False  young_adult
0
1   woman       False    C    Cherbourg   yes  False        adult
1
2   woman       False  NaN  Southampton   yes   True        adult
1
3   woman       False    C  Southampton   yes  False        adult
1
4     man        True  NaN  Southampton    no   True        adult
0
5     man        True  NaN   Queenstown    no   True          NaN
0
6     man        True    E  Southampton    no   True        adult
0
7   child       False  NaN  Southampton    no  False        child
0
8   woman       False  NaN  Southampton   yes  False        adult
1
```

```
9    child       False  NaN    Cherbourg   yes  False        child
1
10   child       False    G   Southampton   yes  False        child
1
11   woman       False    C   Southampton   yes   True        adult
1
12     man        True  NaN   Southampton    no   True  young_adult
0
13     man        True  NaN   Southampton    no  False        adult
0
14   child       False  NaN   Southampton    no   True        child
1
```

22- transpose a wide dataframe

```
import numpy as np
import pandas as pd

# creating a new df
df = pd.DataFrame(np.random.rand(200,25),
columns=list('abcdefghijklmnopqrstuvwxy'))
df.head(10)
```

```
          a         b         c         d         e         f
g  \
0  0.132827  0.861478  0.381068  0.199260  0.361266  0.858788
0.795668
1  0.409010  0.523257  0.108362  0.449044  0.598061  0.917210
0.171974
2  0.914070  0.093799  0.415149  0.109303  0.827890  0.648840
0.005322
3  0.583846  0.436292  0.056225  0.445281  0.957982  0.557661
0.744001
4  0.082551  0.540671  0.175764  0.419771  0.592846  0.767741
0.798678
5  0.731650  0.391901  0.487670  0.932065  0.978401  0.374425
0.358040
6  0.059058  0.077855  0.965495  0.945744  0.167510  0.930586
0.639985
7  0.672971  0.653301  0.412649  0.559806  0.338267  0.711914
0.528979
8  0.599614  0.948672  0.677091  0.360083  0.674268  0.840227
0.413527
9  0.720797  0.851270  0.737823  0.876542  0.534041  0.661559
0.216963

          h         i         j  ...         p         q         r
s  \
```

```
0  0.554288  0.910321  0.087478  ...  0.304814  0.731939  0.329750
0.964195
1  0.804450  0.516903  0.322671  ...  0.935043  0.710343  0.240080
0.250194
2  0.075459  0.214318  0.074475  ...  0.560155  0.470067  0.994082
0.367344
3  0.234530  0.510315  0.912674  ...  0.569000  0.231142  0.940988
0.613934
4  0.282036  0.148481  0.125331  ...  0.110869  0.000713  0.721072
0.549921
5  0.549278  0.743621  0.791742  ...  0.226542  0.607266  0.959127
0.695051
6  0.738865  0.264602  0.979091  ...  0.174893  0.844764  0.637718
0.831202
7  0.956678  0.069897  0.988219  ...  0.381732  0.734731  0.690840
0.254625
8  0.465039  0.693752  0.618672  ...  0.025943  0.998545  0.266659
0.920411
9  0.122911  0.044974  0.970713  ...  0.310744  0.084840  0.673100
0.594431

          t         u         v         w         x         y
0  0.404717  0.596945  0.889682  0.964098  0.179332  0.375344
1  0.882369  0.082566  0.253448  0.698653  0.863154  0.617480
2  0.857296  0.436519  0.640322  0.124675  0.314416  0.140331
3  0.961064  0.232827  0.789785  0.209710  0.964641  0.685779
4  0.459573  0.180333  0.196675  0.839519  0.922636  0.931810
5  0.794907  0.277105  0.429832  0.257540  0.019205  0.309159
6  0.221402  0.069310  0.227090  0.556741  0.332199  0.826710
7  0.229858  0.789363  0.936128  0.655419  0.977878  0.388748
8  0.356931  0.467131  0.890091  0.854352  0.989689  0.232079
9  0.905446  0.324299  0.520878  0.093108  0.039993  0.486274

[10 rows x 25 columns]

df.head(10).T

          0         1         2         3         4         5
6  \
a  0.132827  0.409010  0.914070  0.583846  0.082551  0.731650
0.059058
b  0.861478  0.523257  0.093799  0.436292  0.540671  0.391901
0.077855
c  0.381068  0.108362  0.415149  0.056225  0.175764  0.487670
0.965495
d  0.199260  0.449044  0.109303  0.445281  0.419771  0.932065
0.945744
e  0.361266  0.598061  0.827890  0.957982  0.592846  0.978401
0.167510
f  0.858788  0.917210  0.648840  0.557661  0.767741  0.374425
```

```
0.930586
g  0.795668  0.171974  0.005322  0.744001  0.798678  0.358040
0.639985
h  0.554288  0.804450  0.075459  0.234530  0.282036  0.549278
0.738865
i  0.910321  0.516903  0.214318  0.510315  0.148481  0.743621
0.264602
j  0.087478  0.322671  0.074475  0.912674  0.125331  0.791742
0.979091
k  0.776532  0.128775  0.725980  0.547118  0.075080  0.405607
0.436187
l  0.563153  0.350413  0.411525  0.100533  0.643376  0.924056
0.757701
m  0.506581  0.370257  0.638452  0.659163  0.593176  0.577983
0.985460
n  0.738594  0.437178  0.131410  0.372446  0.813174  0.068731
0.069332
o  0.924236  0.838442  0.265618  0.351531  0.607700  0.580748
0.594271
p  0.304814  0.935043  0.560155  0.569000  0.110869  0.226542
0.174893
q  0.731939  0.710343  0.470067  0.231142  0.000713  0.607266
0.844764
r  0.329750  0.240080  0.994082  0.940988  0.721072  0.959127
0.637718
s  0.964195  0.250194  0.367344  0.613934  0.549921  0.695051
0.831202
t  0.404717  0.882369  0.857296  0.961064  0.459573  0.794907
0.221402
u  0.596945  0.082566  0.436519  0.232827  0.180333  0.277105
0.069310
v  0.889682  0.253448  0.640322  0.789785  0.196675  0.429832
0.227090
w  0.964098  0.698653  0.124675  0.209710  0.839519  0.257540
0.556741
x  0.179332  0.863154  0.314416  0.964641  0.922636  0.019205
0.332199
y  0.375344  0.617480  0.140331  0.685779  0.931810  0.309159
0.826710

          7         8         9
a  0.672971  0.599614  0.720797
b  0.653301  0.948672  0.851270
c  0.412649  0.677091  0.737823
d  0.559806  0.360083  0.876542
e  0.338267  0.674268  0.534041
f  0.711914  0.840227  0.661559
g  0.528979  0.413527  0.216963
h  0.956678  0.465039  0.122911
```

```
i  0.069897  0.693752  0.044974
j  0.988219  0.618672  0.970713
k  0.074679  0.160476  0.314950
l  0.048775  0.580361  0.738518
m  0.043590  0.916966  0.011569
n  0.087198  0.969044  0.415028
o  0.813047  0.268501  0.892454
p  0.381732  0.025943  0.310744
q  0.734731  0.998545  0.084840
r  0.690840  0.266659  0.673100
s  0.254625  0.920411  0.594431
t  0.229858  0.356931  0.905446
u  0.789363  0.467131  0.324299
v  0.936128  0.890091  0.520878
w  0.655419  0.854352  0.093108
x  0.977878  0.989689  0.039993
y  0.388748  0.232079  0.486274
```

```
df.describe()
```

```
                a           b           c           d           e
f  \
count  200.000000  200.000000  200.000000  200.000000  200.000000
200.000000
mean     0.460594    0.492878    0.483272    0.474102    0.500020
0.518502
std      0.293608    0.295228    0.287744    0.288826    0.290930
0.288769
min      0.002107    0.000694    0.009867    0.003563    0.001078
0.005289
25%      0.191533    0.260476    0.226651    0.235435    0.254638
0.272457
50%      0.444254    0.496476    0.473327    0.448856    0.500464
0.551443
75%      0.689634    0.740939    0.715815    0.735535    0.753976
0.794599
max      0.989680    0.999116    0.994806    0.993701    0.997255
0.996676

                g           h           i           j  ...           p
\
count  200.000000  200.000000  200.000000  200.000000  ...  200.000000

mean     0.497501    0.468524    0.514731    0.544683  ...    0.501548

std      0.303009    0.299626    0.302171    0.283796  ...    0.298784

min      0.000366    0.001855    0.003624    0.005732  ...    0.003446

25%      0.214446    0.203493    0.243256    0.319732  ...    0.235802
```

```
50%      0.491303    0.444485    0.526063    0.578165  ...    0.492860
75%      0.760920    0.747500    0.763558    0.791900  ...    0.778172
max      0.997577    0.995240    0.996599    0.998771  ...    0.994695

                q           r           s           t           u
v  \
count  200.000000  200.000000  200.000000  200.000000  200.000000
200.000000
mean     0.498249    0.513066    0.483161    0.515969    0.508895
0.490933
std      0.285130    0.286913    0.279778    0.290108    0.294794
0.294278
min      0.000713    0.005135    0.000628    0.013732    0.014359
0.009953
25%      0.249483    0.274911    0.281488    0.276470    0.251345
0.214264
50%      0.519747    0.522715    0.468762    0.524207    0.514692
0.489064
75%      0.732495    0.756561    0.722363    0.795148    0.791521
0.745907
max      0.998545    0.997848    0.998479    0.986085    0.997727
0.993958

                w           x           y
count  200.000000  200.000000  200.000000
mean     0.501581    0.513423    0.491502
std      0.304189    0.279110    0.283336
min      0.009421    0.007967    0.001944
25%      0.216251    0.286159    0.247849
50%      0.524404    0.520898    0.463880
75%      0.780817    0.749686    0.743113
max      0.992957    0.997189    0.993264

[8 rows x 25 columns]

df.describe().T

   count      mean       std       min       25%       50%       75%
max
a  200.0  0.460594  0.293608  0.002107  0.191533  0.444254  0.689634
0.989680
b  200.0  0.492878  0.295228  0.000694  0.260476  0.496476  0.740939
0.999116
c  200.0  0.483272  0.287744  0.009867  0.226651  0.473327  0.715815
0.994806
d  200.0  0.474102  0.288826  0.003563  0.235435  0.448856  0.735535
```

```
0.993701
e  200.0  0.500020  0.290930  0.001078  0.254638  0.500464  0.753976
0.997255
f  200.0  0.518502  0.288769  0.005289  0.272457  0.551443  0.794599
0.996676
g  200.0  0.497501  0.303009  0.000366  0.214446  0.491303  0.760920
0.997577
h  200.0  0.468524  0.299626  0.001855  0.203493  0.444485  0.747500
0.995240
i  200.0  0.514731  0.302171  0.003624  0.243256  0.526063  0.763558
0.996599
j  200.0  0.544683  0.283796  0.005732  0.319732  0.578165  0.791900
0.998771
k  200.0  0.479959  0.279592  0.004412  0.222466  0.461285  0.710133
0.982542
l  200.0  0.495865  0.288744  0.004735  0.239849  0.485594  0.749203
0.998213
m  200.0  0.545320  0.289421  0.004494  0.313906  0.576571  0.807897
0.994288
n  200.0  0.476944  0.298765  0.004423  0.238985  0.445518  0.753176
0.995580
o  200.0  0.508036  0.281658  0.003724  0.267780  0.513060  0.746842
0.996107
p  200.0  0.501548  0.298784  0.003446  0.235802  0.492860  0.778172
0.994695
q  200.0  0.498249  0.285130  0.000713  0.249483  0.519747  0.732495
0.998545
r  200.0  0.513066  0.286913  0.005135  0.274911  0.522715  0.756561
0.997848
s  200.0  0.483161  0.279778  0.000628  0.281488  0.468762  0.722363
0.998479
t  200.0  0.515969  0.290108  0.013732  0.276470  0.524207  0.795148
0.986085
u  200.0  0.508895  0.294794  0.014359  0.251345  0.514692  0.791521
0.997727
v  200.0  0.490933  0.294278  0.009953  0.214264  0.489064  0.745907
0.993958
w  200.0  0.501581  0.304189  0.009421  0.216251  0.524404  0.780817
0.992957
x  200.0  0.513423  0.279110  0.007967  0.286159  0.520898  0.749686
0.997189
y  200.0  0.491502  0.283336  0.001944  0.247849  0.463880  0.743113
0.993264
```

23- Reshaping a dataframe

```
fasla = pd.DataFrame([['12345', 100,200,300], ['34567', 400,500,600],
['67890',700,800,900]],
```

```
                      columns=['zip', 'factory','warehouse', 'retail'])
fasla.head()
```

```
     zip  factory  warehouse  retail
0  12345      100        200     300
1  34567      400        500     600
2  67890      700        800     900
```

```
#fasla.head().T

fasla2 = pd.DataFrame([[1, '12345','factory'], [2, '34567',
'warehouse']],
                      columns=['user_id', 'zip', 'location_type'])
fasla2.head()
```

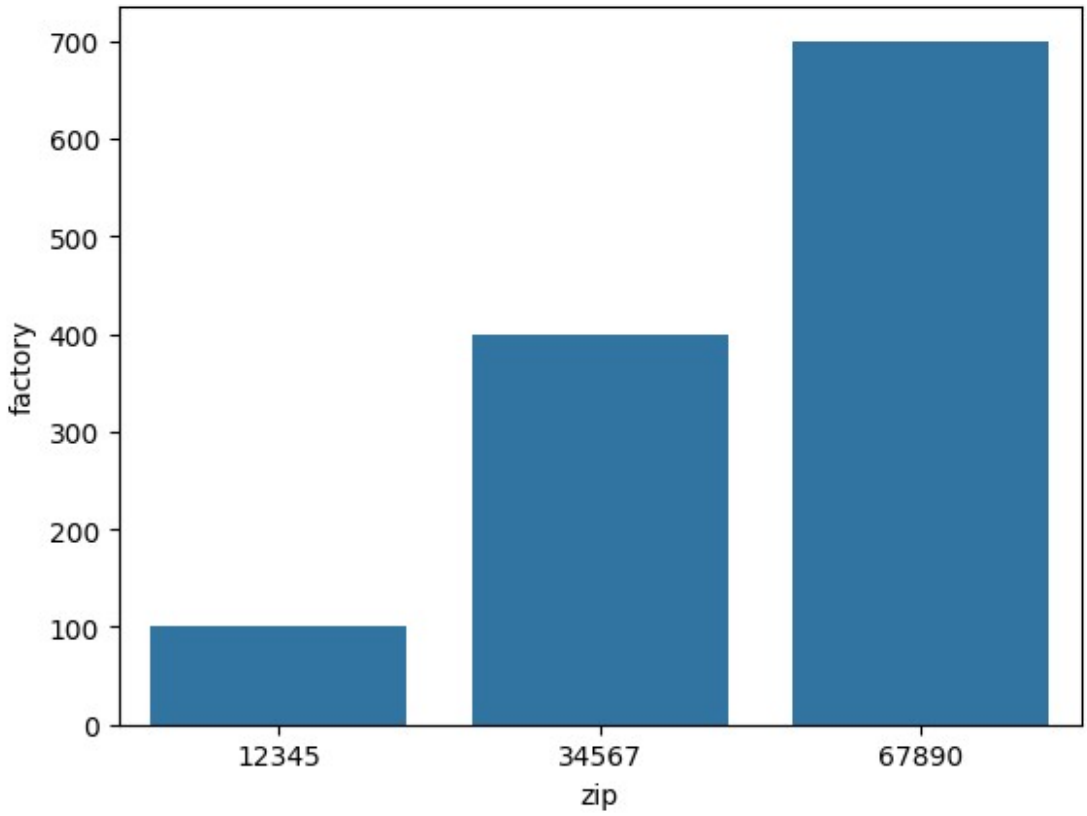
```
   user_id    zip location_type
0        1  12345       factory
1        2  34567     warehouse
```

```
fasla
```

```
     zip  factory  warehouse  retail
0  12345      100        200     300
1  34567      400        500     600
2  67890      700        800     900
```

```
sns.barplot(x='zip', y='factory', data=fasla)
```

```
<Axes: xlabel='zip', ylabel='factory'>
```




```
fasla_long = fasla.melt(id_vars='zip', var_name='location_type',
value_name='distance')
fasla_long.head()
```

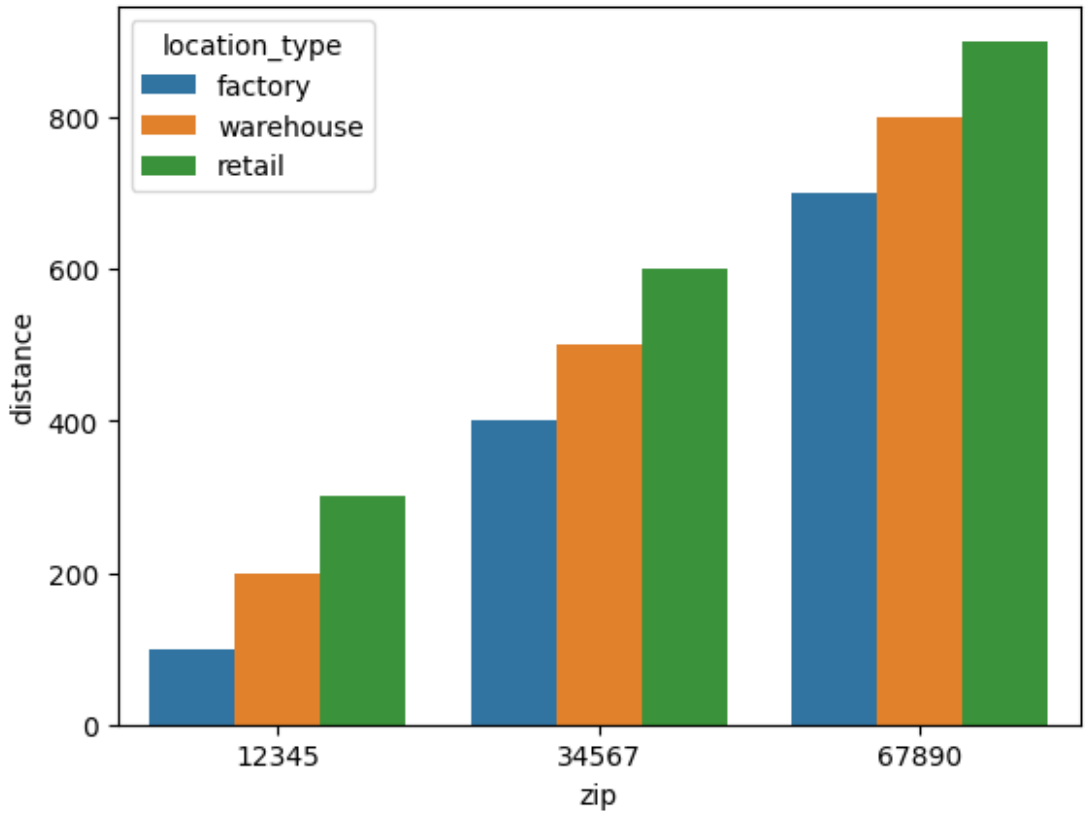
```
     zip location_type  distance
0  12345       factory       100
1  34567       factory       400
2  67890       factory       700
3  12345     warehouse       200
4  34567     warehouse       500
```

```
fasla_long.dtypes
```

```
zip              object
location_type    object
distance          int64
dtype: object
```

```
import seaborn as sns
sns.barplot(x='zip', y='distance', hue='location_type',
data=fasla_long)
```

```
<Axes: xlabel='zip', ylabel='distance'>
```


location_type
factory
warehouse
retail
distance
0
200
400
600
800
12345
34567
67890
zip