pandas

February 12, 2023

[255]:
```
# Subject : Pandas Tips & Tricks
# Written By: Abu Usama
# Email: usamamunawarr@gmail.com
# Tweeter : @Usama_Munawar
# GitHub : UsamaMunawarr
```

0.1 01-How to find the version

[256]:
```
import pandas as pd
pd.__version__
```

[256]: '1.5.3'

[257]:
```
# another way
pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit           : 2e218d10984e9919f0296931d92ea851c6a6faf5
python           : 3.10.9.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
Version          : 10.0.19045
machine          : AMD64
processor        : AMD64 Family 18 Model 1 Stepping 0, AuthenticAMD
byteorder        : little
LC_ALL           : None
LANG             : None
LOCALE           : English_United States.1252

pandas           : 1.5.3
numpy            : 1.24.1
pytz             : 2022.7.1
dateutil         : 2.8.2
setuptools       : 65.6.3
pip              : 22.3.1
```

```
Cython           : None
pytest           : None
hypothesis       : None
sphinx           : None
blosc            : None
feather          : None
xlsxwriter       : None
lxml.etree       : None
html5lib         : None
pymysql          : None
psycopg2         : None
jinja2           : None
IPython          : 8.10.0
pandas_datareader: None
bs4              : None
bottleneck       : None
brotli           : None
fastparquet      : None
fsspec           : None
gcsfs            : None
matplotlib       : 3.6.3
numba            : None
numexpr          : None
odfpy            : None
openpyxl         : 3.1.0
pandas_gbq       : None
pyarrow          : None
pyreadstat       : None
pyxlsb           : None
s3fs             : None
scipy            : 1.10.0
snappy           : None
sqlalchemy       : None
tables           : None
tabulate         : None
xarray           : None
xlrd             : None
xlwt             : None
zstandard        : None
tzdata           : None
```

0.2 02-Make a Dataframe

[258]:
```
df = pd.DataFrame({'A col':[1,2,3],'B col:':[4,5,6]}) # Length OF Both Columns
 ↪Should Be Same
df.head()
```

[258]:

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

[259]:
```
# numpy array use to create dataframe
import numpy as np
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])
pd.DataFrame(arr)
```

[259]:

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

[260]:
```
# Random numpy array for dataframe
pd.DataFrame(np.random.rand(5,3))
```

[260]:

	0	1	2
0	0.659146	0.735215	0.293929
1	0.060365	0.110600	0.561594
2	0.535433	0.669091	0.139903
3	0.205714	0.451061	0.872111
4	0.979244	0.631604	0.601155

[261]:
```
# Random numpy array for dataframe
pd.DataFrame(np.random.rand(5,3),columns=list('ABC'))
```

[261]:

	A	B	C
0	0.831506	0.780607	0.334162
1	0.191650	0.429066	0.977022
2	0.092844	0.408186	0.648925
3	0.352642	0.834463	0.274990
4	0.078514	0.048492	0.986781

0.3 3-How to Rename Columns

[262]:
```
df = pd.DataFrame({'A col':[1,2,3],'B col:':[4,5,6]})
df.head()
```

[262]:

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

[263]:
```
df.rename(columns={'A col':'A','B col:':'B'},inplace=True) #inplace=True Means␣
↪agree to change in column
```

```
df
```

[263]:

	A	B
0	1	4
1	2	5
2	3	6

[264]:
```
#another way of renaming
df.columns = ['col_aa','Col_bb']
df
```

[264]:

	col_aa	Col_bb
0	1	4
1	2	5
2	3	6

[265]:
```
# to replace any Character ('_'),string
df.columns = df.columns.str.replace('_',' ') #
df
```

[265]:

	col aa	Col bb
0	1	4
1	2	5
2	3	6

[266]:
```
# Add Prefix to column
df.add_prefix('baba_')
```

[266]:

	baba_col aa	baba_Col bb
0	1	4
1	2	5
2	3	6

[267]:
```
# Add Sufix to column
df = df.add_suffix('_baba')
df
```

[267]:

	col aa_baba	Col bb_baba
0	1	4
1	2	5
2	3	6

0.4 4-Using Template Data

[268]:
```
import pandas as pd
import numpy as np
import seaborn as sns

# Load Dataset
kashti = sns.load_dataset('titanic') # import from sns
kashti.head()
```

[268]:
```
   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
```

[269]:
```
# Summary
kashti.describe()
```

[269]:
```
         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
```

[270]:
```
# to see columns names
kashti.columns
```

[270]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

[271]:
```
#saving a dataset
kashti.to_csv('titanic_.csv')
```

```python
# pip install openpyxl
kashti.to_excel('kashti.xlsx')
```

0.5 5-Using Your Own Data

[272]:
```python
import pandas as pd
df = pd.read_csv('titanic_.csv')
df.head()
```

[272]:
```
   Unnamed: 0  survived  pclass     sex   age  sibsp  parch     fare embarked  \
0           0         0       3    male  22.0      1      0   7.2500        S
1           1         1       1  female  38.0      1      0  71.2833        C
2           2         1       3  female  26.0      0      0   7.9250        S
3           3         1       1  female  35.0      1      0  53.1000        S
4           4         0       3    male  35.0      0      0   8.0500        S

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

1 6-Reverse Row Order

[273]:
```python
import seaborn as sns
import pandas as pd

# Load Dataset
kashti = sns.load_dataset('titanic') # import from sns
kashti.head()
```

[273]:
```
   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
```

[274]:
```
kashti.loc[::-1].head()
```

[274]:
```
     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
```

[275]:
```
# Here We Just Reset The Indexs
kashti.loc[::-1].reset_index(drop=True).head()
```

[275]:
```
   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
```

2 7-Reverse Column Order

[276]:
```
kashti.loc[:,::-1].head()
```

[276]:
```
   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
```

3 8-Select a Column by dtype

[277]:
```
kashti.dtypes
```

[277]:
```
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
```

[278]:
```
# only select those having numaric types
kashti.select_dtypes(include=['number']).head()
```

[278]:

	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

[279]:
```
# only select those having object types
kashti.select_dtypes(include=['object']).head()
```

[279]:

	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

[280]:
```
# only select those who have category type
kashti.select_dtypes(include=['category']).head()
```

[280]:
```
   class deck
0  Third  NaN
1  First    C
2  Third  NaN
3  First    C
4  Third  NaN
```

[281]:
```
# only select those who have bool type
kashti.select_dtypes(include=['bool']).head()
```

[281]:
```
   adult_male  alone
0        True  False
1       False  False
2       False   True
3       False  False
4        True   True
```

[282]:
```
# only select those having multiple types
kashti.select_dtypes(include=['object','category','number']).head()
```

[282]:
```
   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
```

[283]:
```
# only exclude those having number types
kashti.select_dtypes(exclude=['number']).head()
```

[283]:
```
      sex embarked  class    who  adult_male deck  embark_town alive  alone
0    male        S  Third    man        True  NaN  Southampton    no  False
1  female        C  First  woman       False    C    Cherbourg   yes  False
2  female        S  Third  woman       False  NaN  Southampton   yes   True
3  female        S  First  woman       False    C  Southampton   yes  False
4    male        S  Third    man        True  NaN  Southampton    no   True
```

4 9-convert string to numbers

[284]:
```
df = pd.DataFrame({'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]})
df
```

[284]:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

[285]:
```
df.dtypes
```

[285]:
```
A    int64
B    int64
C    int64
dtype: object
```

[286]:
```
df = pd.DataFrame({'A':['1','2','3'],'B':['4','5','6'],'C':['7','8','9']})
df
```

[286]:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

[287]:
```
df.dtypes
```

[287]:
```
A    object
B    object
C    object
dtype: object
```

[288]:
```
df.astype({'A':'float64','B':'float64','C':'float64'}).dtypes
```

[288]:
```
A    float64
B    float64
C    float64
dtype: object
```

5 10-Reduce Dataframe size

[289]:
```
df_1 =sns.load_dataset('titanic')
df_1.shape
```

[289]:
```
(891, 15)
```

[290]:
```
df_1.sample(frac=0.1).shape #its means 10% of data we used
```

[290]:
```
(89, 15)
```

6 11- Copy Data from Clip board

any data which you copied from any source you can past it in your clipbord command

[291]:
```
# load dataset
import pandas as pd
import seaborn as sns
df_2 = sns.load_dataset('titanic')
df_2.to_excel('kashti.xlsx')
```

[292]:
```
# read clioboard in python
df_2 = pd.read_clipboard()
df_2
```

[292]:
```
Empty DataFrame
Columns: [len(kashti.groupby('embark_town'))]
Index: []
```

6.1 12-slip dataframe into two subsets

[293]:
```
# load dataset
import pandas as pd
import seaborn as sns
df_2 = sns.load_dataset('titanic')
```

[294]:
```
len(df_2)
```

[294]:
```
891
```

[295]:
```
df_2.shape
```

[295]:
```
(891, 15)
```

[296]:
```
from random import random
kashti_1 = df_2.sample(frac=0.50,random_state=1)
kashti_1.shape
```

[296]:
```
(446, 15)
```

[297]:
```
kashti_2 = df_2.drop(kashti_1.index)
kashti_2.shape
```

[297]: (445, 15)

[298]:
```
len(kashti_1) + len(kashti_2)
```

[298]: 891

6.2 13-Joint to datasets

[299]:
```
data= kashti_1.append(kashti_2)
data.shape
```

```
C:\Users\Usama Munawar\AppData\Local\Temp\ipykernel_7980\1096773453.py:1:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  data= kashti_1.append(kashti_2)
```

[299]: (891, 15)

6.3 14-Filtering a dataset

[300]:
```
df_1.head()
```

[300]:
```
   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
```

[301]:
```
# to see unique values
df_1.sex.unique()
```

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

[302]:
```
#To get data only related to females
df_1[(df_1.sex=='female')].shape
```

[302]: (314, 15)

[303]:
```
df_1[((df_1.embark_town=='Southampton')|
(df_1.embark_town=='Queenstown'))&
(df_1.sex=='male')]
# uper command means the passengers who traverled from Southamption or␣
↪Queenstown shoul be male
# and we can see it in table in sex column there are all males
# (|) or
# (&) and
```

[303]:
```
     survived  pclass   sex   age  sibsp  parch     fare embarked   class  \
0           0       3  male  22.0      1      0   7.2500        S   Third
4           0       3  male  35.0      0      0   8.0500        S   Third
5           0       3  male   NaN      0      0   8.4583        Q   Third
6           0       1  male  54.0      0      0  51.8625        S   First
7           0       3  male   2.0      3      1  21.0750        S   Third
..        …     …   …   …    …    …      …       …
881         0       3  male  33.0      0      0   7.8958        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
886         0       2  male  27.0      0      0  13.0000        S  Second
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
4      man        True  NaN  Southampton    no   True
5      man        True  NaN   Queenstown    no   True
6      man        True    E  Southampton    no   True
7    child       False  NaN  Southampton    no  False
..     …         …  …          …   …    …
881    man        True  NaN  Southampton    no   True
883    man        True  NaN  Southampton    no   True
884    man        True  NaN  Southampton    no   True
886    man        True  NaN  Southampton    no   True
890    man        True  NaN   Queenstown    no   True

[482 rows x 15 columns]
```

[304]:
```
# To see specific type in whole column
df_1[df_1.embark_town.isin(['Queenstown'])].head()
```

[304]:
```
    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
5          0       3    male   NaN      0      0   8.4583        Q  Third
16         0       3    male   2.0      4      1  29.1250        Q  Third
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
```

```
      who  adult_male deck embark_town alive  alone
5     man        True  NaN  Queenstown    no   True
16  child       False  NaN  Queenstown    no  False
22  child       False  NaN  Queenstown   yes   True
28  woman       False  NaN  Queenstown   yes   True
32  woman       False  NaN  Queenstown   yes   True
```

[305]:
```python
#To see age more then 30
df_1[df_1.age>30].head()
```

[305]:
```
    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
1          1       1  female  38.0      1      0  71.2833        C  First
3          1       1  female  35.0      1      0  53.1000        S  First
4          0       3    male  35.0      0      0   8.0500        S  Third
6          0       1    male  54.0      0      0  51.8625        S  First
11         1       1  female  58.0      0      0  26.5500        S  First

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

6.4 15-Filtering by large categories

[306]:
```python
# Full detail of specific Column
df_1.embark_town.value_counts()
```

[306]:
```
Southampton    644
Cherbourg      168
Queenstown      77
Name: embark_town, dtype: int64
```

[307]:
```python
# Means 3 largest numbers in count
df_1.age.value_counts().nlargest(3)
```

[307]:
```
24.0    30
22.0    27
18.0    26
Name: age, dtype: int64
```

[308]:
```python
#same like uper command but in index
counts = df_1.age.value_counts()
counts.nlargest(3).index
```

[308]:
```
Float64Index([24.0, 22.0, 18.0], dtype='float64')
```

[309]:
```python
#means show top 1 category in who
counts = df_1.who.value_counts()
counts.nlargest(3).index
df_1[df_1.who.isin(counts.nlargest(1).index)].head()
```

[309]:
```
    survived  pclass   sex   age  sibsp  parch     fare embarked  class  who  \
0          0       3  male  22.0      1      0   7.2500        S  Third  man
4          0       3  male  35.0      0      0   8.0500        S  Third  man
5          0       3  male   NaN      0      0   8.4583        Q  Third  man
6          0       1  male  54.0      0      0  51.8625        S  First  man
12         0       3  male  20.0      0      0   8.0500        S  Third  man

    adult_male deck  embark_town alive  alone
0         True  NaN  Southampton    no  False
4         True  NaN  Southampton    no   True
5         True  NaN   Queenstown    no   True
6         True    E  Southampton    no   True
12        True  NaN  Southampton    no   True
```

6.5 16-Splitting a string into multiple columns

[310]:
```python
#import libraries
import pandas as pd
df = pd.DataFrame({'name':['Abu Usama','Ahmed Rasheed','Umar Twise','Hafiz␣
↪Haroon','Zulquarnain Ali'],
                   'location':
↪['Lhr_Pak','Grw_Pak','Fsd_Pak','IsL_Pak','Karaachi_Pak']})
df.head()
```

[310]:
```
              name      location
0        Abu Usama       Lhr_Pak
1    Ahmed Rasheed       Grw_Pak
2       Umar Twise       Fsd_Pak
3     Hafiz Haroon       IsL_Pak
4  Zulquarnain Ali  Karaachi_Pak
```

[311]:
```python
df[["first_name","second_name"]] = df.name.str.split(' ',expand=True)
df
```

[311]:
```
              name      location   first_name second_name
0        Abu Usama       Lhr_Pak          Abu       Usama
1    Ahmed Rasheed       Grw_Pak        Ahmed     Rasheed
2       Umar Twise       Fsd_Pak         Umar       Twise
3     Hafiz Haroon       IsL_Pak        Hafiz      Haroon
4  Zulquarnain Ali  Karaachi_Pak  Zulquarnain         Ali
```

[312]:
```
df[["first_name","second_name"]] = df.name.str.split(' ',expand=True)
df[["City","Country"]] = df.location.str.split('_',expand=True)
df
```

[312]:
```
              name     location   first_name second_name      City Country
0        Abu Usama      Lhr_Pak          Abu       Usama       Lhr     Pak
1    Ahmed Rasheed      Grw_Pak        Ahmed     Rasheed       Grw     Pak
2       Umar Twise      Fsd_Pak         Umar       Twise       Fsd     Pak
3     Hafiz Haroon      IsL_Pak        Hafiz      Haroon       IsL     Pak
4  Zulquarnain Ali  Karaachi_Pak  Zulquarnain         Ali  Karaachi     Pak
```

[313]:
```
#Refine Data manipulation
df = df[['first_name','second_name','City','City']]
df
```

[313]:
```
    first_name second_name      City      City
0          Abu       Usama       Lhr       Lhr
1        Ahmed     Rasheed       Grw       Grw
2         Umar       Twise       Fsd       Fsd
3        Hafiz      Haroon       IsL       IsL
4  Zulquarnain         Ali  Karaachi  Karaachi
```

6.6 17-Aggrigate by multiple groups/functions

[314]:
```
# Libraries
import pandas as pd
import seaborn as sns

# Load Dataset
kashti = sns.load_dataset('titanic') # import from sns
kashti.head()
```

[314]:
```
   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
```

[315]:
```python
# We Will Get Count ("Mostly For categorigals/Objects")
kashti.groupby('embark_town').count()
```

[315]:

embark_town	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	alive	alone
Cherbourg	168	168	168	130	168	168	168	168	168	168	168	69	168	168
Queenstown	77	77	77	28	77	77	77	77	77	77	77	4	77	77
Southampton	644	644	644	554	644	644	644	644	644	644	644	128	644	644

[316]:
```python
#To check num of catagories
len(kashti.groupby('embark_town'))
```

[316]: 3

[317]:
```python
kashti.groupby(['sex','pclass','embarked']).count()
```

[317]:

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
```

6.7 18-select specific rows and columns

[321]:
```
kashti.head()
```

[321]:
```
   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
```

[323]:
```
#select columns
kashti[['sex','class']]
```

[323]:
```
        sex   class
0      male   Third
1    female   First
2    female   Third
3    female   First
4      male   Third
..      …       …
886    male  Second
887  female   First
```

```
888  female   Third
889    male   First
890    male   Third

[891 rows x 2 columns]
```

[324]:
```
kashti.describe()
```

[324]:
```
         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
```

[326]:
```
#select specific rows
kashti.describe().loc[['min','25%','50%','75%','max']]
```

[326]:
```
     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
```

[327]:
```
#another way
kashti.describe().loc['min':'max']
```

[327]:
```
     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
```

[328]:
```
# select specifi row and column
kashti.describe().loc['min':'max','survived':'age']
```

[328]:
```
     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
```

6.8 19-Reshape multiindex series

[330]:
```
kashti.head()
```

[330]:
```
   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
```

[331]:
```
kashti.survived.mean()
```

[331]:
```
0.3838383838383838
```

[332]:
```
kashti.groupby('sex').survived.mean()
```

[332]:
```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

[334]:
```
kashti.groupby(['sex','pclass']).survived.mean()
```

[334]:
```
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
```

[335]:
```
#another way
kashti.groupby(['sex','pclass']).survived.mean().unstack()
```

[335]:
```
pclass         1         2         3
sex
female  0.968085  0.921053  0.500000
male    0.368852  0.157407  0.135447
```

7 20- Continous to Categorigal

[336]:
```
kashti.head()
```

[336]:
```
   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
```

[338]:
```
kashti.age.head()
```

[338]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

[339]:
```
#creating bins
pd.cut(kashti.age,bins =[0,18,25,99],labels=['child','young_adult','adult']).
 ↪head()
```

[339]:
```
0    young_adult
1          adult
2          adult
3          adult
4          adult
Name: age, dtype: category
Categories (3, object): ['child' < 'young_adult' < 'adult']
```

[340]:
```
#add the column of upercode code in dataset
kashti['new_age'] =pd.cut(kashti.age,bins␣
 ↪=[0,18,25,99],labels=['child','young_adult','adult'])
kashti.head()
```

[340]:
```
   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
```

7.1 21-convert one sr=et of values into another

[341]:
```python
kashti.sex.head()
```

[341]:
```
0      male
1    female
2    female
3    female
4      male
Name: sex, dtype: object
```

[343]:
```python
#convert label into number
kashti.sex.map({'male':0,'female':1})
```

[343]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    1
889    0
890    0
Name: sex, Length: 891, dtype: int64
```

[344]:
```python
#add the column of upercode code in dataset
kashti['sex_num']=kashti.sex.map({'male':0,'female':1})
```

[345]:
```python
#fatest way to convert label into column
kashti.embarked.head()
```

[345]:
```
0    S
1    C
2    S
```

```
3    S
4    S
Name: embarked, dtype: object
```

[346]:

```
#just given the first num then next label authomaticlly converted and then add␣
↪into column
kashti['embark_num'] = kashti.embarked.factorize()[0]
kashti.head(10)
```

[346]:

```
   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
5         0       3    male   NaN      0      0   8.4583        Q   Third
6         0       1    male  54.0      0      0  51.8625        S   First
7         0       3    male   2.0      3      1  21.0750        S   Third
8         1       3  female  27.0      0      2  11.1333        S   Third
9         1       2  female  14.0      1      0  30.0708        C  Second

     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

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

8 22-Transpose a wide data frame

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

[350]:
```
#create a new df
df = pd.DataFrame(np.random.
 ↪rand(200,25),columns=list('abcdefghijklmnopqrstuvwxy'))
df.head(10)
```

[350]:
```
          a         b         c         d         e         f         g  \
0  0.739196  0.962636  0.737367  0.291707  0.122214  0.224559  0.663998
1  0.232421  0.215086  0.590795  0.406559  0.610684  0.553781  0.243950
2  0.665215  0.648586  0.312740  0.443512  0.905017  0.868130  0.426649
3  0.142840  0.429359  0.563898  0.862646  0.264531  0.210176  0.149603
4  0.801768  0.746660  0.001561  0.532474  0.958956  0.839600  0.179217
5  0.013729  0.542512  0.808070  0.572649  0.525837  0.086288  0.624211
6  0.923711  0.253492  0.613705  0.068075  0.072722  0.221727  0.606249
7  0.452430  0.078244  0.578688  0.209122  0.227828  0.751744  0.969986
8  0.663642  0.569777  0.786706  0.563265  0.068768  0.551390  0.870425
9  0.387862  0.504110  0.315317  0.185458  0.947473  0.538324  0.245226

          h         i         j  …         p         q         r         s  \
0  0.064207  0.213898  0.836857  …  0.217524  0.168498  0.705596  0.749959
1  0.450695  0.329926  0.759475  …  0.884608  0.719100  0.925425  0.919831
2  0.740419  0.043397  0.215018  …  0.938395  0.121647  0.718086  0.170395
3  0.631138  0.513374  0.917746  …  0.385982  0.214295  0.150429  0.695390
4  0.622274  0.544334  0.551291  …  0.570401  0.729224  0.193407  0.723316
5  0.957857  0.977494  0.906287  …  0.888793  0.793164  0.961809  0.550445
6  0.860227  0.980341  0.247998  …  0.999231  0.776602  0.968618  0.378250
7  0.751224  0.416843  0.677364  …  0.625257  0.532759  0.954078  0.259447
8  0.227798  0.560430  0.632005  …  0.650527  0.968638  0.314015  0.912056
9  0.937275  0.454881  0.878657  …  0.609825  0.437658  0.728497  0.748051

          t         u         v         w         x         y
0  0.787467  0.980431  0.275195  0.290778  0.911030  0.770289
1  0.998348  0.024582  0.584413  0.135891  0.495662  0.463865
2  0.961862  0.982932  0.978669  0.772768  0.035872  0.563612
3  0.613643  0.135445  0.270248  0.467506  0.578882  0.920145
4  0.975004  0.727155  0.384549  0.714319  0.605021  0.300078
5  0.580103  0.306358  0.232771  0.346197  0.031116  0.809410
6  0.693896  0.986064  0.199369  0.493149  0.279916  0.115562
7  0.408916  0.189607  0.849203  0.744128  0.630082  0.021019
8  0.298195  0.408717  0.513357  0.153976  0.258666  0.485332
9  0.820611  0.032401  0.819617  0.062317  0.907981  0.117922

[10 rows x 25 columns]
```

[349]:
```python
#Transpose
df.head(10).T
```

[349]:

	0	1	2	3	4	5	6 \
a	0.403151	0.005485	0.371091	0.522225	0.898107	0.113655	0.649852
b	0.433746	0.046650	0.175314	0.271713	0.973633	0.155110	0.236052
c	0.346742	0.905293	0.394103	0.243966	0.492044	0.680064	0.171855
d	0.537939	0.636666	0.216641	0.874777	0.143204	0.365620	0.156062
e	0.671371	0.085891	0.927090	0.858404	0.588290	0.484703	0.955130
f	0.914849	0.679959	0.490461	0.153758	0.790131	0.787364	0.863811
g	0.084738	0.033324	0.093540	0.417620	0.751919	0.449590	0.192868
h	0.268385	0.721092	0.601130	0.676869	0.692968	0.722866	0.626020
i	0.585407	0.122552	0.461428	0.541701	0.541313	0.837093	0.939986
j	0.183223	0.800647	0.064688	0.726981	0.713276	0.421647	0.286302
k	0.948050	0.908088	0.878819	0.172419	0.399863	0.387093	0.080049
l	0.772748	0.555684	0.760872	0.172386	0.783389	0.397901	0.132427
m	0.759060	0.510954	0.708210	0.708711	0.849232	0.956611	0.749485
n	0.752523	0.099644	0.834178	0.751464	0.478252	0.853491	0.596394
o	0.153784	0.115084	0.986266	0.622421	0.911904	0.343024	0.317738
p	0.594405	0.559291	0.794257	0.232836	0.332528	0.246794	0.732232
q	0.179016	0.216450	0.473295	0.757731	0.342752	0.476437	0.136844
r	0.092351	0.902225	0.762000	0.475439	0.917878	0.633166	0.344740
s	0.324591	0.529870	0.701361	0.545066	0.218073	0.842947	0.582585
t	0.695916	0.161716	0.905177	0.669382	0.056478	0.281448	0.951033
u	0.142737	0.294936	0.611811	0.699378	0.363384	0.095476	0.965178
v	0.892500	0.138870	0.719540	0.002448	0.178116	0.688048	0.271281
w	0.541977	0.759994	0.093163	0.674784	0.955926	0.609029	0.049496
x	0.745920	0.615171	0.386623	0.834423	0.996243	0.002779	0.826982
y	0.372698	0.781042	0.500660	0.536941	0.771642	0.333967	0.502117

	7	8	9
a	0.403798	0.614342	0.349405
b	0.465393	0.999251	0.755097
c	0.322793	0.788514	0.273547
d	0.068219	0.391249	0.306634
e	0.155782	0.680394	0.598401
f	0.016972	0.702949	0.363921
g	0.413577	0.514398	0.895716
h	0.101072	0.391496	0.221959
i	0.061983	0.807292	0.632210
j	0.101547	0.617419	0.349676
k	0.003343	0.920444	0.746356
l	0.195866	0.320274	0.005334
m	0.483260	0.001952	0.509984
n	0.715800	0.730869	0.440651
o	0.317238	0.197676	0.024749
p	0.532385	0.146513	0.911119

q	0.983349	0.878671	0.153275
r	0.199740	0.007547	0.542156
s	0.162830	0.541392	0.117421
t	0.491911	0.101020	0.522967
u	0.630948	0.495889	0.105146
v	0.121932	0.791049	0.071710
w	0.125476	0.115890	0.763729
x	0.621710	0.357706	0.167487
y	0.868054	0.540057	0.066875

[351]:
```
df.describe()
```

[351]:

	a	b	c	d	e	f	\
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	
mean	0.509741	0.502531	0.478258	0.499917	0.507382	0.453380	
std	0.280697	0.288297	0.298002	0.291728	0.295057	0.288389	
min	0.006893	0.002047	0.000872	0.006327	0.006447	0.000481	
25%	0.290493	0.255893	0.207971	0.240891	0.241291	0.194151	
50%	0.529798	0.499417	0.468535	0.504113	0.510522	0.462914	
75%	0.733226	0.747878	0.736891	0.727739	0.758098	0.698043	
max	0.988997	0.995720	0.996229	0.994406	0.993366	0.996264	

	g	h	i	j	…	p	\
count	200.000000	200.000000	200.000000	200.000000	…	200.000000	
mean	0.481783	0.561572	0.473985	0.512149	…	0.510702	
std	0.260735	0.293626	0.277504	0.299860	…	0.277054	
min	0.007838	0.000409	0.001478	0.011683	…	0.009513	
25%	0.277447	0.285070	0.249229	0.251383	…	0.289890	
50%	0.482986	0.621481	0.474671	0.536905	…	0.473597	
75%	0.661451	0.809941	0.675637	0.775078	…	0.756065	
max	0.999388	0.998380	0.994963	0.995363	…	0.999231	

	q	r	s	t	u	v	\
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	
mean	0.490304	0.514317	0.493567	0.522967	0.525198	0.504498	
std	0.299396	0.291170	0.293931	0.294248	0.284028	0.279113	
min	0.010566	0.007233	0.007842	0.003690	0.001932	0.001069	
25%	0.226732	0.292042	0.243159	0.257528	0.299589	0.268413	
50%	0.474370	0.550533	0.474271	0.535045	0.536089	0.479842	
75%	0.758487	0.729092	0.749591	0.784954	0.744811	0.756795	
max	0.998717	0.995169	0.998975	0.998348	0.995991	0.998179	

	w	x	y
count	200.000000	200.000000	200.000000
mean	0.484840	0.462156	0.517449
std	0.280869	0.282265	0.284616
min	0.000353	0.001473	0.018100

```
25%     0.261798   0.220279   0.297047
50%     0.456859   0.451523   0.523841
75%     0.705225   0.694039   0.765322
max     0.996893   0.991400   0.994256

[8 rows x 25 columns]
```

[352]:
```
#Transpose
df.describe().T
```

[352]:

	count	mean	std	min	25%	50%	75%	max
a	200.0	0.509741	0.280697	0.006893	0.290493	0.529798	0.733226	0.988997
b	200.0	0.502531	0.288297	0.002047	0.255893	0.499417	0.747878	0.995720
c	200.0	0.478258	0.298002	0.000872	0.207971	0.468535	0.736891	0.996229
d	200.0	0.499917	0.291728	0.006327	0.240891	0.504113	0.727739	0.994406
e	200.0	0.507382	0.295057	0.006447	0.241291	0.510522	0.758098	0.993366
f	200.0	0.453380	0.288389	0.000481	0.194151	0.462914	0.698043	0.996264
g	200.0	0.481783	0.260735	0.007838	0.277447	0.482986	0.661451	0.999388
h	200.0	0.561572	0.293626	0.000409	0.285070	0.621481	0.809941	0.998380
i	200.0	0.473985	0.277504	0.001478	0.249229	0.474671	0.675637	0.994963
j	200.0	0.512149	0.299860	0.011683	0.251383	0.536905	0.775078	0.995363
k	200.0	0.516904	0.294169	0.001498	0.232460	0.566196	0.749517	0.999194
l	200.0	0.528727	0.279524	0.003930	0.290884	0.548796	0.774853	0.989074
m	200.0	0.510830	0.292694	0.007852	0.259526	0.523639	0.746737	0.993845
n	200.0	0.518842	0.303445	0.004810	0.237871	0.498810	0.807128	0.994948
o	200.0	0.541733	0.285824	0.000641	0.310253	0.533651	0.790817	0.995443
p	200.0	0.510702	0.277054	0.009513	0.289890	0.473597	0.756065	0.999231
q	200.0	0.490304	0.299396	0.010566	0.226732	0.474370	0.758487	0.998717
r	200.0	0.514317	0.291170	0.007233	0.292042	0.550533	0.729092	0.995169
s	200.0	0.493567	0.293931	0.007842	0.243159	0.474271	0.749591	0.998975
t	200.0	0.522967	0.294248	0.003690	0.257528	0.535045	0.784954	0.998348
u	200.0	0.525198	0.284028	0.001932	0.299589	0.536089	0.744811	0.995991
v	200.0	0.504498	0.279113	0.001069	0.268413	0.479842	0.756795	0.998179
w	200.0	0.484840	0.280869	0.000353	0.261798	0.456859	0.705225	0.996893
x	200.0	0.462156	0.282265	0.001473	0.220279	0.451523	0.694039	0.991400
y	200.0	0.517449	0.284616	0.018100	0.297047	0.523841	0.765322	0.994256

8.1 23-Reshaping a dataftame

[353]:
```
fasla = pd.
 ↪DataFrame([['12345',100,200,300],['34567',400,500,600],['67890',700,800,900]],
                        columns=['zip','factory','warehouse','retail'])
fasla.head()
```

[353]:

	zip	factory	warehouse	retail
0	12345	100	200	300
1	34567	400	500	600

```
2  67890      700         800      900
```

[356]:
```
fasla2 = pd.DataFrame([[1,'12345','factory'],[2,'34567','warehouse']],
                      columns=['user_id','zip','location_type'])
fasla2.head()
```

[356]:
```
   user_id    zip location_type
0        1  12345       factory
1        2  34567     warehouse
```

[358]:
```
fasla
```

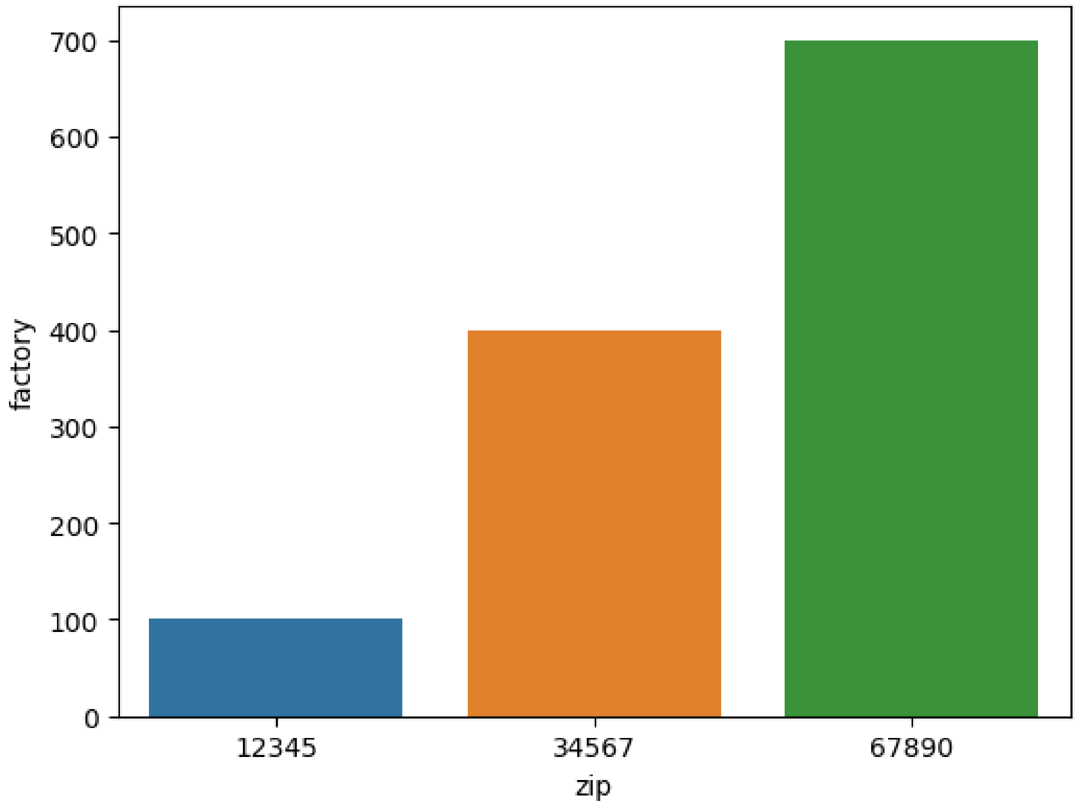
[358]:
```
     zip  factory  warehouse  retail
0  12345      100        200     300
1  34567      400        500     600
2  67890      700        800     900
```

[367]:
```
fasla.dtypes
```

[367]:
```
zip          object
factory       int64
warehouse     int64
retail        int64
dtype: object
```

[366]:
```
sns.barplot(x='zip',y='factory',data=fasla)
```

[366]:
```
<AxesSubplot: xlabel='zip', ylabel='factory'>
```



[363]:
```
fasla_long = fasla.melt(id_vars='zip',var_name='location_type',value_name␣
↪='distance')
fasla_long.head()
```

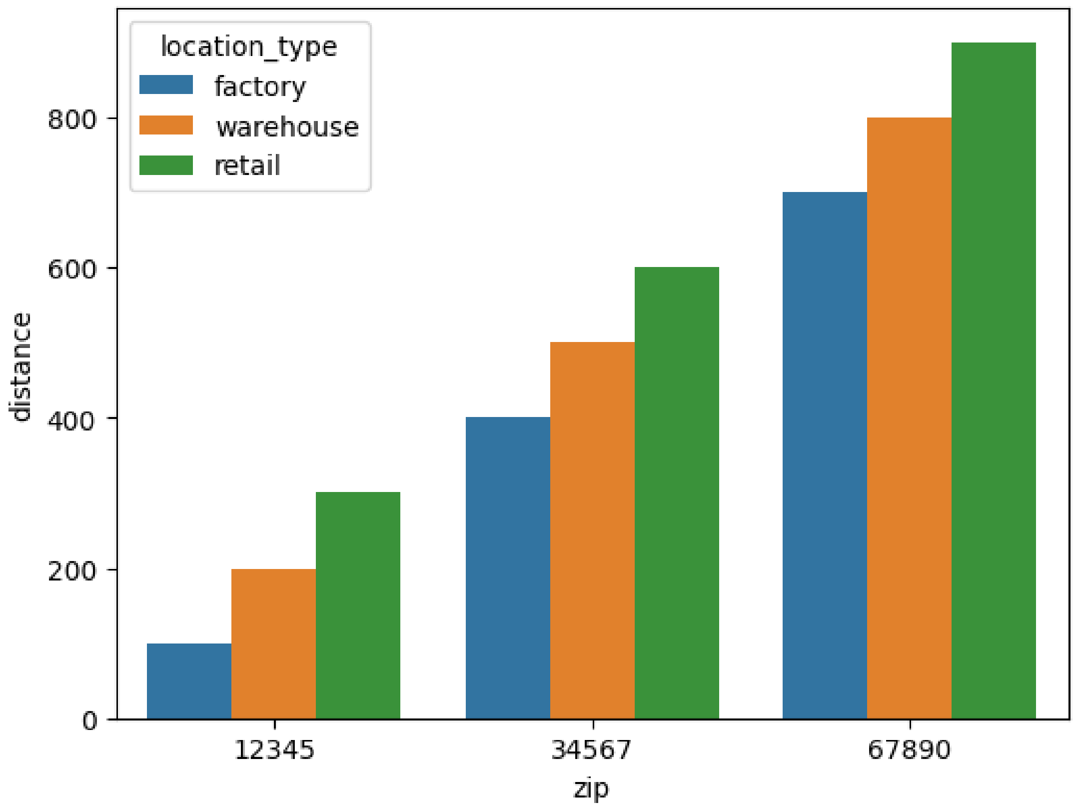
[363]:
```
     zip location_type  distance
0  12345       factory       100
1  34567       factory       400
2  67890       factory       700
3  12345     warehouse       200
4  34567     warehouse       500
```

[365]:
```
fasla_long.dtypes
```

[365]:
```
zip              object
location_type    object
distance          int64
dtype: object
```

[364]:
```
sns.barplot(x='zip',y='distance',hue='location_type',data=fasla_long)
```

[364]:
```
<AxesSubplot: xlabel='zip', ylabel='distance'>
```


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