PYTHON TIPS AND TRICKS

11- Copy data from clipboard

In [ ]:
```
import seaborn as sns
import pandas as pd

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

Out[ ]:

	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

In [ ]:
```
df.to_excel('tips.xlsx')
```

In [ ]:
```
pip install openpyxl
```

```
Collecting openpyxlNote: you may need to restart the kernel to use updated package
s.
  Downloading openpyxl-3.0.10-py2.py3-none-any.whl (242 kB)
     ------------------------------------ 242.1/242.1 KB 780.8 kB/s eta 0:00:00
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.10

WARNING: You are using pip version 22.0.4; however, version 22.2.1 is available.
You should consider upgrading via the 'c:\Users\Azka\AppData\Local\Programs\Python
\Python310\python.exe -m pip install --upgrade pip' command.
```

In [ ]:
```
df = pd.read_clipboard()
df
```

Out[ ]:

							Unnamed: 7
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
5	**25.29**	**4.71**	**Male**	**No**	**Sun**	**Dinner**	4
6	**8.77**	**2.00**	**Male**	**No**	**Sun**	**Dinner**	2
7	**26.88**	**3.12**	**Male**	**No**	**Sun**	**Dinner**	4
8	**15.04**	**1.96**	**Male**	**No**	**Sun**	**Dinner**	2
9	**14.78**	**3.23**	**Male**	**No**	**Sun**	**Dinner**	2
10	**10.27**	**1.71**	**Male**	**No**	**Sun**	**Dinner**	2
11	**35.26**	**5.00**	**Female**	**No**	**Sun**	**Dinner**	4
12	**15.42**	**1.57**	**Male**	**No**	**Sun**	**Dinner**	2
13	**18.43**	**3.00**	**Male**	**No**	**Sun**	**Dinner**	4
14	**14.83**	**3.02**	**Female**	**No**	**Sun**	**Dinner**	2
15	**21.58**	**3.92**	**Male**	**No**	**Sun**	**Dinner**	2
16	**10.33**	**1.67**	**Female**	**No**	**Sun**	**Dinner**	3
17	**16.29**	**3.71**	**Male**	**No**	**Sun**	**Dinner**	3
18	**16.97**	**3.50**	**Female**	**No**	**Sun**	**Dinner**	3

12- Split dataframe into two subsets

In [ ]:
```
import seaborn as sns
import pandas as pd

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

Out[ ]:

	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

In [ ]:
```
len(df)
```

Out[ ]:
```
244
```

In [ ]:
```
df.shape
```

Out[ ]: (244, 7)

In [ ]:
```
from random import Random
tips1= df.sample(frac=0.50, random_state=1)
tips1.shape
```

Out[ ]: (122, 7)

In [ ]:
```
tips2= df.sample(frac=0.50, random_state=2)
tips2.shape
```

Out[ ]: (122, 7)

In [ ]:
```
len(tips1)+len(tips2)
```

Out[ ]: 244

13- Joing two datasets

In [ ]:
```
tips3= tips1.append(tips2)
tips3.head()
```

```
C:\Users\Azka\AppData\Local\Temp\ipykernel_9996\4064519846.py:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  tips3= tips1.append(tips2)
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
67	3.07	1.00	Female	Yes	Sat	Dinner	1
243	18.78	3.00	Female	No	Thur	Dinner	2
206	26.59	3.41	Male	Yes	Sat	Dinner	3
122	14.26	2.50	Male	No	Thur	Lunch	2
89	21.16	3.00	Male	No	Thur	Lunch	2

In [ ]:
```
tips3.shape
```

Out[ ]: (244, 7)

14- Flitering a datasets

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

Out[ ]:

	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

In [ ]:
```
df.sex.unique()
```

Out[ ]:
```
['Female', 'Male']
Categories (2, object): ['Male', 'Female']
```

In [ ]:
```
df[(df.sex=="Female")]
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
11	35.26	5.00	Female	No	Sun	Dinner	4
14	14.83	3.02	Female	No	Sun	Dinner	2
16	10.33	1.67	Female	No	Sun	Dinner	3
...	...	...	...	...	...	...	...
226	10.09	2.00	Female	Yes	Fri	Lunch	2
229	22.12	2.88	Female	Yes	Sat	Dinner	2
238	35.83	4.67	Female	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

87 rows × 7 columns

In [ ]:
```
df.day.unique()
```

Out[ ]:
```
['Sun', 'Sat', 'Thur', 'Fri']
Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']
```

In [ ]:
```
df[(df.day=="Sun")]
```

Out[ ]:

	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
...	...	...	...	...	...	...	...
186	20.90	3.50	Female	Yes	Sun	Dinner	3
187	30.46	2.00	Male	Yes	Sun	Dinner	5
188	18.15	3.50	Female	Yes	Sun	Dinner	3
189	23.10	4.00	Male	Yes	Sun	Dinner	3
190	15.69	1.50	Male	Yes	Sun	Dinner	2

76 rows × 7 columns

In [ ]:
```
df[(df.day=="Sun")].shape
```

Out[ ]:
```
(76, 7)
```

In [ ]:
```
df[(df.day=="Sun") &
    (df.sex=="Female")]
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
11	35.26	5.00	Female	No	Sun	Dinner	4
14	14.83	3.02	Female	No	Sun	Dinner	2
16	10.33	1.67	Female	No	Sun	Dinner	3
18	16.97	3.50	Female	No	Sun	Dinner	3
51	10.29	2.60	Female	No	Sun	Dinner	2
52	34.81	5.20	Female	No	Sun	Dinner	4
114	25.71	4.00	Female	No	Sun	Dinner	3
115	17.31	3.50	Female	No	Sun	Dinner	2
155	29.85	5.14	Female	No	Sun	Dinner	5
157	25.00	3.75	Female	No	Sun	Dinner	4
158	13.39	2.61	Female	No	Sun	Dinner	2
162	16.21	2.00	Female	No	Sun	Dinner	3
164	17.51	3.00	Female	Yes	Sun	Dinner	2
178	9.60	4.00	Female	Yes	Sun	Dinner	2
186	20.90	3.50	Female	Yes	Sun	Dinner	3
188	18.15	3.50	Female	Yes	Sun	Dinner	3

In [ ]:
```
df[df.day.isin(['Fri'])].head()
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
90	28.97	3.00	Male	Yes	Fri	Dinner	2
91	22.49	3.50	Male	No	Fri	Dinner	2
92	5.75	1.00	Female	Yes	Fri	Dinner	2
93	16.32	4.30	Female	Yes	Fri	Dinner	2
94	22.75	3.25	Female	No	Fri	Dinner	2

In [ ]:
```
df[df.day.isin(['Sat','Fri'])].head()
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
19	20.65	3.35	Male	No	Sat	Dinner	3
20	17.92	4.08	Male	No	Sat	Dinner	2
21	20.29	2.75	Female	No	Sat	Dinner	2
22	15.77	2.23	Female	No	Sat	Dinner	2
23	39.42	7.58	Male	No	Sat	Dinner	4

In [ ]:
```
df[df.tip < 3 ].shape
```

Out[ ]:
```
(123, 7)
```

In [ ]:
```
df[df.tip < 1.5 ].shape
```

Out[ ]:
```
(17, 7)
```

In [ ]:
```
df[df.tip > 5 ].shape
```

Out[ ]:
```
(18, 7)
```

In [ ]:
```
df[df.tip > 9 ].shape
```

Out[ ]:
```
(1, 7)
```

15- Filtering by large categories

In [ ]:
```
df.tip.value_counts().nlargest(5)
```

Out[ ]:
```
2.0    33
3.0    23
4.0    12
5.0    10
2.5    10
Name: tip, dtype: int64
```

In [ ]:
```
counts = df.tip.value_counts()
counts.nlargest(3).index
```

Out[ ]:
```
Float64Index([2.0, 3.0, 4.0], dtype='float64')
```

In [ ]:
```
df[df.sex.isna(counts.nlargest(2).index)].head()
```

16- Splitting a string into multiple columns

In [ ]:
```
import pandas as pd

df= pd.DataFrame({'name':['azka saleem','junaid latif','taeeda atvat','roulin yang
                  'location':['fsd,pk','kml,pk','kml,pk','xian,ch']})

df
```

Out[ ]:

	name	location
0	azka saleem	fsd,pk
1	junaid latif	kml,pk
2	taeeda atvat	kml,pk
3	roulin yang	xian,ch

In [ ]:
```
df.name.str.split(' ',expand=True).head()
```

Out[ ]:

	0	1
0	azka	saleem
1	junaid	latif
2	taeeda	atvat
3	roulin	yang

In [ ]:
```
df
```

Out[ ]:

	name	location
0	azka saleem	fsd,pk
1	junaid latif	kml,pk
2	taeeda atvat	kml,pk
3	roulin yang	xian,ch

In [ ]:
```
df[['first name','last name']]= df.name.str.split(' ',expand=True)
df
```

Out[ ]:

	name	location	first name	last name
0	azka saleem	fsd,pk	azka	saleem
1	junaid latif	kml,pk	junaid	latif
2	taeeda atvat	kml,pk	taeeda	atvat
3	roulin yang	xian,ch	roulin	yang

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

Out[ ]:

	name	location	first name	last name	city	country
0	azka saleem	fsd,pk	azka	saleem	fsd	pk
1	junaid latif	kml,pk	junaid	latif	kml	pk
2	taeeda atvat	kml,pk	taeeda	atvat	kml	pk
3	roulin yang	xian,ch	roulin	yang	xian	ch

In [ ]:
```
df= df[['first name','last name','city','country']]
df
```

Out[ ]:

	first name	last name	city	country
0	azka	saleem	fsd	pk
1	junaid	latif	kml	pk
2	taeeda	atvat	kml	pk
3	roulin	yang	xian	ch

17- Aggregate by multiple groups/function

In [ ]:
```
df= sns.load_dataset('tips')
df.head()
```

Out[ ]:

	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

In [ ]:
```
df.groupby('day').mean()
```

Out[ ]:

day	total_bill	tip	size
Thur	17.682742	2.771452	2.451613
Fri	17.151579	2.734737	2.105263
Sat	20.441379	2.993103	2.517241
Sun	21.410000	3.255132	2.842105

In [ ]:
```
df.groupby('day').count()
```

Out[ ]:

day	total_bill	tip	sex	smoker	time	size
Thur	62	62	62	62	62	62
Fri	19	19	19	19	19	19
Sat	87	87	87	87	87	87
Sun	76	76	76	76	76	76

In [ ]:
```
df.groupby('smoker').count()
```

Out[ ]:

smoker	total_bill	tip	sex	day	time	size
Yes	93	93	93	93	93	93
No	151	151	151	151	151	151

In [ ]:
```
len(df.groupby('size'))
```

Out[ ]:
```
6
```

In [ ]:
```
df.groupby(['sex','time','smoker']).count()
```

Out[ ]:

sex	time	smoker	total_bill	tip	day	size
Male	**Lunch**	**Yes**	13	13	13	13
		No	20	20	20	20
	Dinner	**Yes**	47	47	47	47
		No	77	77	77	77
Female	**Lunch**	**Yes**	10	10	10	10
		No	25	25	25	25
	Dinner	**Yes**	23	23	23	23
		No	29	29	29	29

In [ ]:
```
df.describe().loc[['min','50%','max']]
```

Out[ ]:

	total_bill	tip	size
min	3.070	1.0	1.0
50%	17.795	2.9	2.0
max	50.810	10.0	6.0

In [ ]:
```
df.describe().loc['min':'max']
```

Out[ ]:

	total_bill	tip	size
min	3.0700	1.0000	1.0
25%	13.3475	2.0000	2.0
50%	17.7950	2.9000	2.0
75%	24.1275	3.5625	3.0
max	50.8100	10.0000	6.0

In [ ]:
```
df.describe().loc['min':'max','day':'size']
```

19- Reshaping multi index series

In [ ]:
```
df= sns.load_dataset('tips')
df.head()
```

Out[ ]:

	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

In [ ]:
```
df.tip.mean()
```

Out[ ]:
```
2.99827868852459
```

In [ ]:
```
df.groupby('sex').time.mean()
```

20- Continous to catgorical data conversion

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

Out[ ]:

	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

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

Out[ ]:
```
0    Sun
1    Sun
2    Sun
3    Sun
4    Sun
Name: day, dtype: category
Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']
```

In [ ]:
```
#creating bins
pd.cut(df.tip, bins=[0,2,4,10], labels=['averge','nice','rich']).head()
df['tip range']=pd.cut(df.tip, bins=[0,2,4,10], labels=['averge','nice','rich'])
df.head()
```

Out[ ]:

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

21- Convert one set of values into another

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

Out[ ]:
```
0    Female
1      Male
2      Male
3      Male
4    Female
Name: sex, dtype: category
Categories (2, object): ['Male', 'Female']
```

In [ ]:
```
df.sex.map({'Male':1,'Female':0})
```

Out[ ]:
```
0      0
1      1
2      1
3      1
4      0
      ..
239    1
240    0
241    1
242    1
243    0
Name: sex, Length: 244, dtype: category
Categories (2, int64): [1, 0]
```

In [ ]:
```
df['sex encoding']= df.sex.map({'Male':1,'Female':0})
df.head()
```

Out[ ]:

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

In [ ]:
```
df.time.unique()
```

Out[ ]:
```
['Dinner', 'Lunch']
Categories (2, object): ['Lunch', 'Dinner']
```

In [ ]:
```
df.time.factorize()[0]
```

Out[ ]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0], dtype=int64)
```

In [ ]:
```
df['time encoding']= df.time.factorize()[0]
df.head()
```

Out[ ]:

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

22- Transpose a wide dataframe

In [ ]:
```
#new df
import numpy as np
df= pd.DataFrame(np.random.rand(200,25),columns=list('abcdefghijklmnopqrstuvwxy'))
df.head()
```

Out[ ]:

	a	b	c	d	e	f	g	h	i	
0	0.040200	0.662400	0.193667	0.115706	0.464606	0.463900	0.510713	0.091619	0.066790	0.750
1	0.879247	0.205009	0.639448	0.939976	0.386922	0.198709	0.746150	0.522167	0.475508	0.220
2	0.819767	0.444312	0.774072	0.259454	0.634342	0.965464	0.546735	0.459999	0.177064	0.529
3	0.119328	0.037929	0.152667	0.030718	0.107276	0.378213	0.259030	0.420966	0.602106	0.921
4	0.792890	0.582513	0.283510	0.901785	0.371716	0.625500	0.071014	0.780386	0.559033	0.118

5 rows × 25 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(200, 25)
```

In [ ]:
```
df.head(10).T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	
a	0.040200	0.879247	0.819767	0.119328	0.792890	0.666045	0.565655	0.724479	0.951474	0.125
b	0.662400	0.205009	0.444312	0.037929	0.582513	0.640011	0.966298	0.610773	0.975794	0.491
c	0.193667	0.639448	0.774072	0.152667	0.283510	0.264461	0.000641	0.794116	0.571500	0.958
d	0.115706	0.939976	0.259454	0.030718	0.901785	0.925314	0.970741	0.538795	0.955987	0.535
e	0.464606	0.386922	0.634342	0.107276	0.371716	0.470272	0.385502	0.537402	0.878169	0.531
f	0.463900	0.198709	0.965464	0.378213	0.625500	0.749554	0.032413	0.684072	0.927081	0.180
g	0.510713	0.746150	0.546735	0.259030	0.071014	0.846415	0.084109	0.881723	0.664216	0.017
h	0.091619	0.522167	0.459999	0.420966	0.780386	0.580729	0.475067	0.423808	0.955163	0.494
i	0.066790	0.475508	0.177064	0.602106	0.559033	0.362385	0.540819	0.990690	0.483380	0.431
j	0.750462	0.220063	0.529229	0.921909	0.118099	0.185682	0.980668	0.026911	0.558162	0.288
k	0.180714	0.056353	0.649276	0.618791	0.835383	0.589950	0.173958	0.091643	0.357750	0.410
l	0.191809	0.293806	0.313903	0.684704	0.705414	0.002003	0.823743	0.779024	0.550220	0.646
m	0.660099	0.960118	0.225310	0.423570	0.345825	0.545123	0.262089	0.051201	0.611014	0.073
n	0.734458	0.826604	0.927383	0.891901	0.325104	0.465219	0.602961	0.624221	0.596214	0.095
o	0.185254	0.703781	0.103486	0.751526	0.757770	0.509833	0.923436	0.601906	0.824269	0.843
p	0.654876	0.657200	0.828575	0.945887	0.588032	0.692529	0.795002	0.209367	0.352602	0.904
q	0.427132	0.068845	0.998392	0.008860	0.530685	0.393986	0.257699	0.837845	0.812928	0.762
r	0.414806	0.837776	0.042311	0.461197	0.426814	0.610844	0.385906	0.041258	0.282960	0.254
s	0.818090	0.749180	0.890365	0.184975	0.176653	0.043568	0.645988	0.766126	0.743329	0.369
t	0.876932	0.076460	0.940383	0.821505	0.568899	0.796224	0.815845	0.662239	0.821372	0.844
u	0.109762	0.777611	0.263156	0.561197	0.670038	0.305857	0.677512	0.686790	0.911549	0.400
v	0.990007	0.612672	0.160234	0.424134	0.085717	0.343616	0.548509	0.698848	0.808898	0.758
w	0.299006	0.772212	0.926544	0.411521	0.582910	0.478985	0.161046	0.452236	0.240349	0.187
x	0.468402	0.822991	0.468279	0.202851	0.036298	0.093720	0.327963	0.429233	0.405187	0.130
y	0.163646	0.840579	0.398673	0.168067	0.541187	0.671957	0.897179	0.546963	0.663708	0.049

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

Out[ ]:

	a	b	c	d	e	f	g	
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.00
mean	0.527227	0.511712	0.490443	0.487241	0.485703	0.536524	0.491463	0.52
std	0.315872	0.288950	0.279313	0.301794	0.288364	0.285580	0.281105	0.27
min	0.000253	0.007629	0.000641	0.002167	0.000802	0.000337	0.004859	0.00
25%	0.238274	0.241038	0.247393	0.231465	0.267780	0.298812	0.249042	0.33
50%	0.562193	0.535853	0.501089	0.468905	0.464587	0.568378	0.487296	0.53
75%	0.818924	0.773655	0.730656	0.748524	0.732354	0.763474	0.733592	0.75
max	0.998937	0.997211	0.991280	0.992641	0.987486	0.998412	0.989316	0.98

8 rows × 25 columns

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

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
a	200.0	0.527227	0.315872	0.000253	0.238274	0.562193	0.818924	0.998937
b	200.0	0.511712	0.288950	0.007629	0.241038	0.535853	0.773655	0.997211
c	200.0	0.490443	0.279313	0.000641	0.247393	0.501089	0.730656	0.991280
d	200.0	0.487241	0.301794	0.002167	0.231465	0.468905	0.748524	0.992641
e	200.0	0.485703	0.288364	0.000802	0.267780	0.464587	0.732354	0.987486
f	200.0	0.536524	0.285580	0.000337	0.298812	0.568378	0.763474	0.998412
g	200.0	0.491463	0.281105	0.004859	0.249042	0.487296	0.733592	0.989316
h	200.0	0.528545	0.275173	0.005270	0.335845	0.530817	0.759797	0.984693
i	200.0	0.523234	0.282183	0.001091	0.295680	0.532039	0.764455	0.999205
j	200.0	0.497012	0.285645	0.002765	0.261737	0.485004	0.736215	0.981812
k	200.0	0.498412	0.280041	0.001623	0.295675	0.458113	0.726785	0.997760
l	200.0	0.517496	0.286690	0.002003	0.300638	0.542755	0.772960	0.989491
m	200.0	0.496075	0.285072	0.000094	0.258846	0.494103	0.739179	0.994393
n	200.0	0.518691	0.282526	0.008674	0.267471	0.565140	0.756714	0.982802
o	200.0	0.547754	0.266619	0.002280	0.349343	0.572038	0.766448	0.987904
p	200.0	0.521484	0.283320	0.007888	0.290994	0.539882	0.768281	0.989960
q	200.0	0.468453	0.292080	0.003849	0.224240	0.438724	0.696652	0.999100
r	200.0	0.470012	0.296884	0.002296	0.195157	0.449290	0.751364	0.978567
s	200.0	0.497735	0.289659	0.009601	0.247182	0.473700	0.753417	0.989432
t	200.0	0.509781	0.292938	0.000508	0.268053	0.529537	0.766644	0.995903
u	200.0	0.495032	0.273628	0.000980	0.289148	0.475823	0.723524	0.995263
v	200.0	0.492513	0.294693	0.009677	0.229495	0.478818	0.773548	0.993003
w	200.0	0.490996	0.288310	0.012999	0.242688	0.474075	0.755823	0.989614
x	200.0	0.476073	0.286428	0.001471	0.239019	0.468340	0.727011	0.981379
y	200.0	0.503904	0.284473	0.001290	0.292954	0.510115	0.730137	0.998940

23- Reshaping a dataframe

In [ ]:
```
fasla= pd.DataFrame([['1234',100,200,300],['5678',400,500,600],['9012',700,800,900
                        columns=['zip','factory','warehouse','retail'])
fasla.head()
```

Out[ ]:

	zip	factory	warehouse	retail
0	1234	100	200	300
1	5678	400	500	600
2	9012	700	800	900

In [ ]:
```
fasla.T
```

Out[ ]:

	0	1	2
zip	1234	5678	9012
factory	100	400	700
warehouse	200	500	800
retail	300	600	900

In [ ]:
```
fasla2= pd.DataFrame([[1,'1234','factory'],[2,'5678','warehouse'],
                      [3,'9012','retail']],columns=['id','zip','location'])
fasla2.head()
```

Out[ ]:

	id	zip	location
0	1	1234	factory
1	2	5678	warehouse
2	3	9012	retail

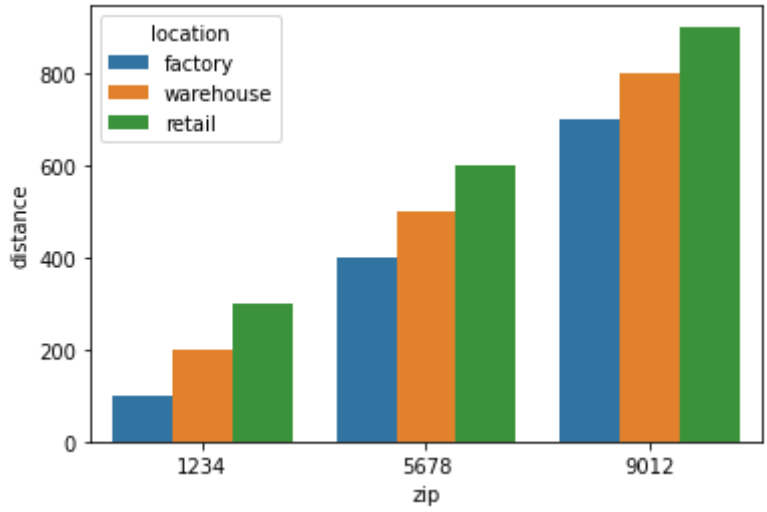
In [ ]:
```
faslanew= fasla.melt(id_vars=['zip'],value_name='distance',var_name='location')
faslanew.head()
```

Out[ ]:

	zip	location	distance
0	1234	factory	100
1	5678	factory	400
2	9012	factory	700
3	1234	warehouse	200
4	5678	warehouse	500

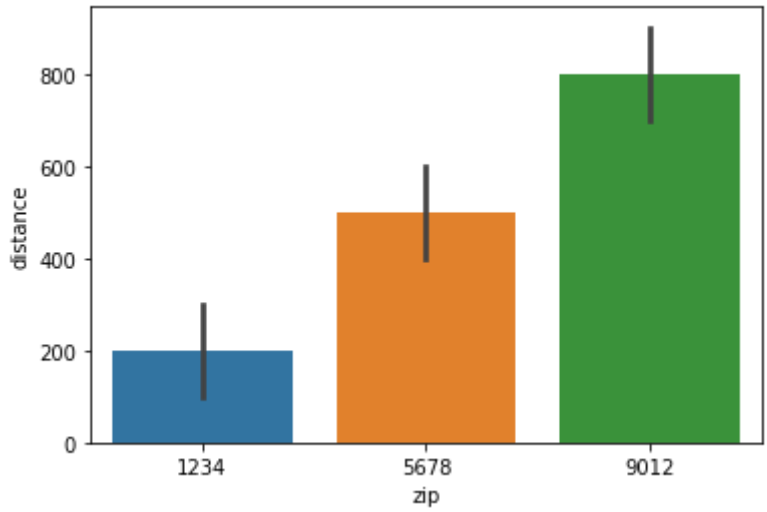
In [ ]:
```
import seaborn as sns
sns.barplot(x='zip',y='distance',hue='location', data=faslanew)
```

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



In [ ]:
```
sns.barplot(x='zip',y='distance', data=faslanew)
```

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



In [ ]: