Muhammad Arham Adeel

Pandas Assignemnt 4

16- Splitting a string into multiple columns

In [1]:
```
#import libraries

import pandas as pd

df = pd.DataFrame({"name":["Ahmad Raza","Ali Raza","Sajjad Ali","Abu Bakar"],
                   "Location":["Lahore, Pakistan","Sargodha, Pakistan","Karachi, Pakista
df
```

Out[1]:

	name	Location
0	Ahmad Raza	Lahore, Pakistan
1	Ali Raza	Sargodha, Pakistan
2	Sajjad Ali	Karachi, Pakistan
3	Abu Bakar	Hamburg, Germany

In [2]:
```
## Splitting into two columns

df.name.str.split(" ")
```

Out[2]:
```
0    [Ahmad, Raza]
1      [Ali, Raza]
2    [Sajjad, Ali]
3     [Abu, Bakar]
Name: name, dtype: object
```

In [3]:
```
## Adding those splits into new columns

df[["first_name","last_name"]] = df.name.str.split(" ",expand=True)
```

In [4]:
```
df
```

Out[4]:

	name	Location	first_name	last_name
0	Ahmad Raza	Lahore, Pakistan	Ahmad	Raza
1	Ali Raza	Sargodha, Pakistan	Ali	Raza
2	Sajjad Ali	Karachi, Pakistan	Sajjad	Ali
3	Abu Bakar	Hamburg, Germany	Abu	Bakar

In [5]:

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

In [6]:
```
df
```

Out[6]:

	name	Location	first_name	last_name	city	country
0	Ahmad Raza	Lahore, Pakistan	Ahmad	Raza	Lahore	Pakistan
1	Ali Raza	Sargodha, Pakistan	Ali	Raza	Sargodha	Pakistan
2	Sajjad Ali	Karachi, Pakistan	Sajjad	Ali	Karachi	Pakistan
3	Abu Bakar	Hamburg, Germany	Abu	Bakar	Hamburg	Germany

In [7]:
```
df = df[["first_name","last_name","city","country"]]
df
```

Out[7]:

	first_name	last_name	city	country
0	Ahmad	Raza	Lahore	Pakistan
1	Ali	Raza	Sargodha	Pakistan
2	Sajjad	Ali	Karachi	Pakistan
3	Abu	Bakar	Hamburg	Germany

17 Aggregating by multiple groups/functions

In [8]:
```
#Libraries

import pandas as pd
import seaborn as sns

# import data set

df = sns.load_dataset("titanic")
df.head()
```

Out[8]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [9]:
```
df.groupby("who").count()
```

Out[9]:

who	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	adult_male	deck	embark_to
child	83	83	83	83	83	83	83	83	83	83	13	
man	537	537	537	413	537	537	537	537	537	537	99	5
woman	271	271	271	218	271	271	271	269	271	271	91	2

In [10]:
```
df.groupby("sex").count()
```

Out[10]:

sex	survived	pclass	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_to
female	314	314	261	314	314	314	312	314	314	314	97	
male	577	577	453	577	577	577	577	577	577	577	106	

In [11]:
```
df.groupby("who").head()
```

Out[11]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	NaN
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	NaN
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	C
12	0	3	male	20.0	0	0	8.0500	S	Third	man	True	NaN
14	0	3	female	14.0	0	0	7.8542	S	Third	child	False	NaN
16	0	3	male	2.0	4	1	29.1250	Q	Third	child	False	NaN

In [12]:
```
df.groupby("who").sum()
```

Out[12]:

who	survived	pclass	age	sibsp	parch	fare	adult_male	alone
child	49	218	528.67	144	105	2721.2210	0	6
man	88	1274	13700.50	159	82	13352.0656	537	410
woman	205	565	6976.00	163	153	12620.6627	0	121

In [13]:
```
len(df.sex)
```

Out[13]:
```
891
```

In [14]:
```
len(df.groupby("who"))
```

Out[14]:
```
3
```

In [15]:
```
len(df.groupby("sex"))
```

Out[15]:
```
2
```

In [16]:
```
len(df.groupby("fare"))
```

Out[16]:
```
248
```

In [17]:
```
len(df.groupby("pclass"))
```

Out[17]:
```
3
```

In [18]:
```
df.head()
```

Out[18]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [19]:
```
df.groupby(["sex",'pclass',"who"])
```

Out[19]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000012367CC8280>
```

In [20]:

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

Out[20]:

sex	pclass	who	survived	age	sibsp	parch	fare	embarked	class	adult_male	deck	embark
female	1	child	3	3	3	3	3	3	3	3	3	
		woman	91	82	91	91	91	89	91	91	78	
	2	child	10	10	10	10	10	10	10	10	1	
		woman	66	64	66	66	66	66	66	66	9	
	3	child	30	30	30	30	30	30	30	30	2	
		woman	114	72	114	114	114	114	114	114	4	
male	1	child	3	3	3	3	3	3	3	3	3	
		man	119	98	119	119	119	119	119	119	91	
	2	child	9	9	9	9	9	9	9	9	3	
		man	99	90	99	99	99	99	99	99	3	
	3	child	28	28	28	28	28	28	28	28	1	
		man	319	225	319	319	319	319	319	319	5	

In [22]:

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

Out[22]:

sex	pclass	adult_male	survived	age	sibsp	parch	fare	embarked	class	who	deck	embark_to
female	1	False	94	85	94	94	94	92	94	94	81	
	2	False	76	74	76	76	76	76	76	76	10	
	3	False	144	102	144	144	144	144	144	144	6	
male	1	False	3	3	3	3	3	3	3	3	3	
		True	119	98	119	119	119	119	119	119	91	
	2	False	9	9	9	9	9	9	9	9	3	
		True	99	90	99	99	99	99	99	99	3	
	3	False	28	28	28	28	28	28	28	28	1	
		True	319	225	319	319	319	319	319	319	5	

In [23]:

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

Out[23]:

sex	pclass	embarked	survived	age	sibsp	parch	fare	class	who	adult_male	deck	embark_to

sex	pclass	embarked	survived	age	sibsp	parch	fare	class	who	adult_male	deck	embark_to
female	1	C	43	38	43	43	43	43	43	43	35	
		Q	1	1	1	1	1	1	1	1	1	
		S	48	44	48	48	48	48	48	48	43	
	2	C	7	7	7	7	7	7	7	7	1	
		Q	2	1	2	2	2	2	2	2	1	
		S	67	66	67	67	67	67	67	67	8	
	3	C	23	16	23	23	23	23	23	23	1	
		Q	33	10	33	33	33	33	33	33	0	
		S	88	76	88	88	88	88	88	88	5	
male	1	C	42	36	42	42	42	42	42	42	31	
		Q	1	1	1	1	1	1	1	1	1	
		S	79	64	79	79	79	79	79	79	62	
	2	C	10	8	10	10	10	10	10	10	1	
		Q	1	1	1	1	1	1	1	1	0	
		S	97	90	97	97	97	97	97	97	5	
	3	C	43	25	43	43	43	43	43	43	0	
		Q	39	14	39	39	39	39	39	39	1	
		S	265	214	265	265	265	265	265	265	5	

18- Selct specific rows or columns

In [24]:
```
df.head()
```

Out[24]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [26]:
```
#Select multiple columns

df[["sex","survived","class"]]
```

Out[26]:

	sex	survived	class
0	male	0	Third
1	female	1	First
2	female	1	Third
3	female	1	First
4	male	0	Third
...	...	...	...
886	male	0	Second
887	female	1	First
888	female	0	Third
889	male	1	First
890	male	0	Third

891 rows × 3 columns

In [27]:
```
df.describe()
```

Out[27]:

	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

Select specific rows

In [29]:
```
df.describe().loc[["min","25%","50%","75%","max"]]
```

Out[29]:

	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

	survived	pclass	age	sibsp	parch	fare
max	1.0	3.0	80.000	8.0	6.0	512.3292

In [32]:
```
df.describe().loc["mean":"max"]
```

Out[32]:

	survived	pclass	age	sibsp	parch	fare
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

In [33]:
```
df.describe().loc["mean":"max", : ]
```

Out[33]:

	survived	pclass	age	sibsp	parch	fare
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

In [34]:
```
df.describe().loc["mean":"max" , "survived"]
```

Out[34]:
```
mean    0.383838
std     0.486592
min     0.000000
25%     0.000000
50%     0.000000
75%     1.000000
max     1.000000
Name: survived, dtype: float64
```

In [35]:
```
df.describe().loc["mean":"max" , ["survived","age"]]
```

Out[35]:

	survived	age
mean	0.383838	29.699118

	survived	age
std	0.486592	14.526497
min	0.000000	0.420000
25%	0.000000	20.125000
50%	0.000000	28.000000
75%	1.000000	38.000000
max	1.000000	80.000000

In [39]:
```
df.describe().loc["min":"max" , "survived":"age"]
```

Out[39]:

	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 multiindex series

In [40]:
```
df.head()
```

Out[40]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [41]:
```
df.survived.mean()
```

Out[41]:
```
0.3838383838383838
```

In [42]:
```
df.groupby("sex").survived.mean()
```

Out[42]:
```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

In [46]:
```
df.groupby(["sex",'class']).survived.mean()
```

Out[46]:
```
sex     class
female  First     0.968085
        Second    0.921053
        Third     0.500000
male    First     0.368852
        Second    0.157407
        Third     0.135447
Name: survived, dtype: float64
```

In [47]:
```
df.groupby(["sex",'class']).survived.mean().unstack()
```

Out[47]:

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

20- Continous to categorical data conversion

In [50]:
```
df.head()
```

Out[50]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [51]:
```
df.age.head()
```

Out[51]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

Creating bins

In [52]:
```
pd.cut(df.age , bins = [0,18,25,99], labels = ["chuld","young_adult","adult"])
```

Out[52]:
```
0      young_adult
1            adult
```

```
2            adult
3            adult
4            adult
          ...
886          adult
887    young_adult
888            NaN
889          adult
890          adult
Name: age, Length: 891, dtype: category
Categories (3, object): ['chuld' < 'young_adult' < 'adult']
```

In [53]:
```
df["new_age"] = pd.cut(df.age , bins = [0,18,25,99], labels = ["chuld","young_adult","a
```

In [55]:
```
df.head()
```

Out[55]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

21- Convert One set of value into another one

In [56]:
```
df.sex
```

Out[56]:
```
0        male
1      female
2      female
3      female
4        male
        ...
886      male
887    female
888    female
889      male
890      male
Name: sex, Length: 891, dtype: object
```

In [57]:
```
df.sex.map({"male":0,"female":1})
```

Out[57]:
```
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
```

In [58]:
```
df.head()
```

Out[58]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [59]:
```
df["sex_number"] =  df.sex.map({"male":0,"female":1})
df.head()
```

Out[59]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [60]:
```
df.embarked.head()
```

Out[60]:
```
0    S
1    C
2    S
3    S
4    S
Name: embarked, dtype: object
```

In [61]:
```
df.embarked.count()
```

Out[61]:
```
889
```

In [62]:
```
df.embarked.unique()
```

Out[62]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [63]:

```
df.embarked.factorize()[0]
```

Out[63]:

```
array([ 0,  1,  0,  0,  0,  2,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  2,
        0,  0,  1,  0,  0,  2,  0,  0,  0,  1,  0,  2,  0,  1,  1,  2,  0,
        1,  0,  1,  0,  0,  1,  0,  0,  1,  1,  2,  0,  2,  2,  1,  0,  0,
        0,  1,  0,  1,  0,  0,  1,  0,  0,  1, -1,  0,  0,  1,  1,  0,  0,
        0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  2,  0,  1,  0,  0,  1,  0,  2,  0,  1,
        0,  0,  0,  1,  0,  0,  1,  2,  0,  1,  0,  1,  0,  0,  0,  0,  1,
        0,  0,  0,  1,  1,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  1,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  2,  0,  0,  1,  0,  0,  1,  0,  0,  0,  1,  0,  0,  0,  0,  2,
        0,  2,  0,  0,  0,  0,  0,  1,  1,  2,  0,  2,  0,  0,  0,  0,  1,
        0,  0,  0,  1,  2,  1,  0,  0,  0,  0,  2,  1,  0,  0,  1,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  1,  2,  0,  0,  1,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        1,  1,  0,  1,  0,  2,  0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,
        0,  1,  2,  0,  0,  0,  2,  0,  2,  0,  0,  0,  0,  1,  0,  0,  0,
        2,  0,  1,  1,  0,  0,  1,  1,  0,  0,  1,  2,  2,  0,  2,  0,  0,
        1,  1,  1,  1,  1,  1,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  2,
        0,  0,  1,  0,  0,  0,  1,  2,  0,  0,  0,  0,  0,  0,  1,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0,  0,
        0,  2,  2,  0,  1,  1,  0,  2,  0,  1,  1,  2,  1,  1,  0,  0,  1,
        0,  1,  0,  1,  1,  0,  1,  1,  0,  0,  0,  0,  0,  0,  2,  1,  0,
        0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  2,  2,  0,  0,  0,  0,  0,  0,  0,  1,  2,  0,  0,  0,
        0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  1,  1,  0,  1,  0,  0,  0,
        2,  0,  0,  0,  0,  0,  0,  0,  0,  2,  1,  0,  0,  0,  1,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  1,  0,  0,  0,  0,  0,
        1,  0,  1,  1,  0,  0,  0,  0,  2,  2,  0,  0,  1,  0,  0,  0,  0,
        2,  0,  0,  1,  0,  0,  0,  2,  0,  0,  0,  0,  1,  1,  1,  2,  0,
        0,  0,  0,  0,  1,  1,  1,  0,  0,  0,  1,  0,  1,  0,  0,  0,  0,
        1,  0,  0,  1,  0,  0,  1,  0,  2,  1,  0,  0,  1,  1,  0,  0,  2,
        0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  2,  0,  0,  0,  0,
        1,  0,  0,  1,  0,  1,  1,  0,  0,  1,  0,  0,  0,  1,  0,  2,  0,
        0,  0,  0,  1,  1,  0,  0,  0,  0,  1,  0,  0,  0,  1,  0,  0,  0,
        2,  2,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0,  0,  0,  2,  0,  0,
        2,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  1,  1,
        0,  1,  0,  0,  0,  0,  0,  2,  2,  0,  0,  2,  0,  1,  0,  1,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,
        2,  1,  0,  0,  0,  1,  0,  0,  0,  0,  0,  1,  0,  1,  0,  0,  0,
        2,  1,  0,  1,  0,  1,  2,  0,  0,  0,  0,  0,  1,  1,  0,  0,  0,
        0,  0,  1,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0,
        1,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,
        0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,
        0,  1,  2,  2,  0,  0,  0,  0,  1,  0,  0,  2,  0,  2,  0,  1,  0,
        0,  0,  0,  0,  0,  2,  0,  1,  2,  0,  0,  1,  0,  0,  0,  0,  1,
        0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  1,  0,  0,  0,  0,  0,  0,  0,  2,  0,  1,  2, -1,  1,  0,  1,
        0,  0,  1,  0,  0,  0,  1,  0,  0,  1,  1,  0,  0,  0,  1,  0,  1,
        0,  0,  1,  0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  0,  0,  1,
        0,  0,  0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  1,  0,  0,  0,  0,
        0,  2,  0,  0,  0,  1,  2], dtype=int64)
```

In [66]:

```
df["embarked_num"] = df.embarked.factorize()[0]
df.sample(10)
```

Out[66]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	N
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	N
315	1	3	female	26.0	0	0	7.8542	S	Third	woman	False	N
340	1	2	male	2.0	1	1	26.0000	S	Second	child	False	
60	0	3	male	22.0	0	0	7.2292	C	Third	man	True	N
503	0	3	female	37.0	0	0	9.5875	S	Third	woman	False	N
541	0	3	female	9.0	4	2	31.2750	S	Third	child	False	N
613	0	3	male	NaN	0	0	7.7500	Q	Third	man	True	N
557	0	1	male	NaN	0	0	227.5250	C	First	man	True	N
365	0	3	male	30.0	0	0	7.2500	S	Third	man	True	N

22- Transpose a wide dataframe

In [67]:
```
import numpy as np
import pandas as pd
```

In [72]:
```
## Creating a new data df

df = pd.DataFrame(np.random.randn(200,26) , columns = list("qwertyuiopasdfghjklzxcvbnm"
```

In [73]:
```
df
```

Out[73]:

	q	w	e	r	t	y	u	i	o	
0	0.860929	-0.422272	0.020497	0.448266	0.679817	1.221083	0.249258	-0.634501	0.211465	-0
1	0.829879	-1.873474	1.158072	-1.659237	-1.129361	-0.149909	-1.767549	-1.625943	-0.606103	-0
2	-0.325454	0.811925	-0.479030	0.525105	-0.196639	0.978271	1.355286	0.218035	0.017941	0
3	0.148398	-0.075814	1.203942	1.438677	-0.580375	-0.051865	0.404960	-1.399807	1.314730	-1
4	0.843806	0.797779	-3.446544	0.013469	0.933809	1.373676	0.951747	0.285639	0.185692	-1
...	...	...	...	...	...	...	...	...	...	
195	0.341151	0.535581	-1.114357	-0.907603	-0.195833	-0.668891	1.665566	0.768435	0.819483	-1
196	-2.649286	-0.766690	-1.475859	-0.442724	1.119541	1.306879	0.228675	-0.075692	0.469634	1
197	-0.665984	1.754397	1.144136	-0.008428	-0.913257	0.137203	-0.533819	-1.485715	-0.273236	1
198	-0.674781	-1.623320	0.645890	-0.448712	1.728253	1.513538	-0.051213	0.536655	-0.103581	-0
199	-0.716666	0.311927	-0.984772	3.241708	-1.186067	-0.651337	0.819708	0.430265	1.207023	1

200 rows × 26 columns

In [74]:
```
df.head(10).T
```

Out[74]:

	0	1	2	3	4	5	6	7	8	
q	0.860929	0.829879	-0.325454	0.148398	0.843806	-0.836928	0.857955	-1.090673	0.311932	-1.1
w	-0.422272	-1.873474	0.811925	-0.075814	0.797779	1.039998	0.047331	-0.334624	0.271452	-1.0
e	0.020497	1.158072	-0.479030	1.203942	-3.446544	-0.331430	-0.209746	0.233499	0.527284	1.3
r	0.448266	-1.659237	0.525105	1.438677	0.013469	-1.261569	0.862893	2.399032	0.915732	-0.6
t	0.679817	-1.129361	-0.196639	-0.580375	0.933809	0.280298	-0.209187	-0.046094	0.595018	0.6
y	1.221083	-0.149909	0.978271	-0.051865	1.373676	0.006581	2.438159	1.084575	0.078342	-1.3
u	0.249258	-1.767549	1.355286	0.404960	0.951747	-0.828917	-0.438498	-0.130721	-1.772710	0.1
i	-0.634501	-1.625943	0.218035	-1.399807	0.285639	0.090071	-1.841023	-0.689875	-0.595815	0.3
o	0.211465	-0.606103	0.017941	1.314730	0.185692	-0.282810	0.108063	0.719983	0.950226	-1.2
p	-0.291075	-0.264948	0.030741	-1.380900	-1.141884	2.352868	-2.569475	-0.661953	0.273885	-0.6
a	-0.536779	0.926657	-0.124238	-0.064331	-0.075152	-0.888094	-0.322093	-0.648510	-0.784191	0.1
s	0.971423	-0.583847	1.230515	-1.161845	0.587095	-1.152437	-0.631133	0.828238	-1.770911	0.0
d	0.721577	1.546513	-0.511505	-0.008140	-0.169255	0.990197	1.921775	0.267067	-1.572601	0.3
f	0.221969	-0.536495	-1.295137	-1.166666	-1.577924	0.516364	0.448035	-1.209538	-0.318994	0.5
g	0.178946	2.180106	2.371504	-0.012298	-1.149992	-1.424037	0.955897	0.254755	-0.556101	0.0
h	-0.779989	0.142212	-2.092758	0.870573	-0.768324	-0.076274	-0.871504	-0.639562	-0.697339	1.3
j	-0.050413	0.063337	-2.416455	-1.065507	-0.146258	1.315216	-0.890170	-1.008540	-0.593135	1.1
k	-0.071923	-1.178189	1.713087	0.888191	-0.429759	0.696972	-0.079652	0.016803	-0.034073	-1.7
l	-0.188674	0.577061	0.137685	-0.196177	1.287033	-1.838855	-0.374660	-1.111223	-2.132965	0.0
z	0.357784	-1.761496	0.669390	2.400347	0.136786	-0.529634	-0.032354	-0.331198	0.607133	-0.7
x	-1.513362	0.962438	-0.816145	0.182792	-1.537805	-0.163511	1.329852	-1.692490	2.097384	0.3
c	-0.368595	1.308206	-0.423777	-1.102555	0.589601	1.099634	-1.260645	0.562229	1.328989	-1.0
v	-2.048889	0.394793	-1.177218	0.274522	-0.341386	-1.686048	-1.512479	-0.813889	0.006994	-0.0
b	-0.535700	0.686961	0.456387	-0.126636	1.214722	-1.160689	1.038504	-0.223616	-1.110659	-0.3
n	-0.925391	1.067398	-0.032551	-0.273617	-0.541988	0.460961	1.619251	0.929207	-0.694546	0.2
m	-2.042446	-1.961124	0.420746	-0.801125	0.803773	-0.549820	-0.196084	-0.695931	0.098124	0.0

In [75]:
```
df.describe()
```

Out[75]:

	q	w	e	r	t	y	u	i
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	0.004673	0.018772	0.039674	0.012617	0.082254	0.084839	0.164849	-0.061911
std	0.917580	1.052176	0.921840	1.059716	1.025368	0.982898	0.965199	0.999364
min	-2.649286	-2.892928	-3.446544	-2.540224	-2.532110	-2.559431	-2.444974	-2.557536
25%	-0.604102	-0.730489	-0.580921	-0.651669	-0.586947	-0.563023	-0.468418	-0.774389
50%	0.016575	0.050422	0.163725	-0.065551	0.036348	0.145551	0.134240	-0.036811
75%	0.679821	0.834223	0.665808	0.721915	0.734985	0.731491	0.860467	0.632254
max	2.502143	2.639051	2.071430	3.241708	2.638099	2.764959	2.766000	2.522643

8 rows × 26 columns

In [76]:

```
df.describe().T
```

Out[76]:

	count	mean	std	min	25%	50%	75%	max
q	200.0	0.004673	0.917580	-2.649286	-0.604102	0.016575	0.679821	2.502143
w	200.0	0.018772	1.052176	-2.892928	-0.730489	0.050422	0.834223	2.639051
e	200.0	0.039674	0.921840	-3.446544	-0.580921	0.163725	0.665808	2.071430
r	200.0	0.012617	1.059716	-2.540224	-0.651669	-0.065551	0.721915	3.241708
t	200.0	0.082254	1.025368	-2.532110	-0.586947	0.036348	0.734985	2.638099
y	200.0	0.084839	0.982898	-2.559431	-0.563023	0.145551	0.731491	2.764959
u	200.0	0.164849	0.965199	-2.444974	-0.468418	0.134240	0.860467	2.766000
i	200.0	-0.061911	0.999364	-2.557536	-0.774389	-0.036811	0.632254	2.522643
o	200.0	0.020212	1.044556	-3.274365	-0.648145	0.055269	0.696263	3.360045
p	200.0	0.074786	1.081455	-2.569475	-0.675112	-0.000076	0.860074	3.051083
a	200.0	-0.076507	0.927486	-2.777126	-0.609905	-0.109748	0.525588	2.172009
s	200.0	-0.052316	0.952234	-2.815088	-0.716762	-0.026054	0.582993	2.471215
d	200.0	-0.002748	1.030542	-2.769680	-0.801527	0.044616	0.647295	3.352627
f	200.0	-0.010466	0.993916	-2.171820	-0.781791	-0.042527	0.702504	2.458576
g	200.0	-0.004620	0.989076	-2.477516	-0.692486	0.008751	0.667866	2.371504
h	200.0	-0.077738	1.045218	-2.810544	-0.874251	-0.070706	0.719065	2.400783
j	200.0	-0.028413	1.024442	-2.969543	-0.676920	-0.063984	0.597996	2.640043
k	200.0	-0.019557	0.962635	-2.477311	-0.768106	-0.044676	0.698053	2.614434
l	200.0	-0.051063	0.960152	-2.754323	-0.661846	-0.111238	0.650405	2.135564
z	200.0	0.038291	0.908874	-2.186552	-0.567127	0.020414	0.667316	2.635373

	count	mean	std	min	25%	50%	75%	max
x	200.0	-0.011170	1.070028	-3.061036	-0.783286	0.070088	0.693684	2.671640
c	200.0	0.046909	1.018258	-2.376317	-0.650051	0.114652	0.760701	2.782484
v	200.0	-0.121012	0.992510	-3.235516	-0.812055	-0.100058	0.482884	2.982865
b	200.0	-0.080741	0.925061	-2.548192	-0.764804	-0.028755	0.688738	2.160829
n	200.0	0.028857	0.932377	-3.835888	-0.542490	0.021039	0.691894	2.369067
m	200.0	-0.084792	0.981641	-3.107319	-0.777946	-0.052062	0.587974	2.251279

here i can save the transpose data

In [77]:
```
df.describe().T.to_csv("transpose.csv")
```

23- Reshaping a dataframe

In [89]:
```
fasla = pd.DataFrame([["12345",100,200,300],["34567",400,500,600],['67890',700,800,900]]
                     columns = ["zip","factory","warehouse","retail"])
fasla
```

Out[89]:

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

In [90]:
```
fasla.T
```

Out[90]:

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

Another method to transpose

In [91]:
```
fasla2 = pd.DataFrame([[1,"12345","factory"],[2,"34567","warehosue"]],
                     columns = ["user_id","zip","location_type"])
fasla2
```

Out[91]:

	user_id	zip	location_type
0	1	12345	factory

	user_id	zip	location_type
1	2	34567	warehosue

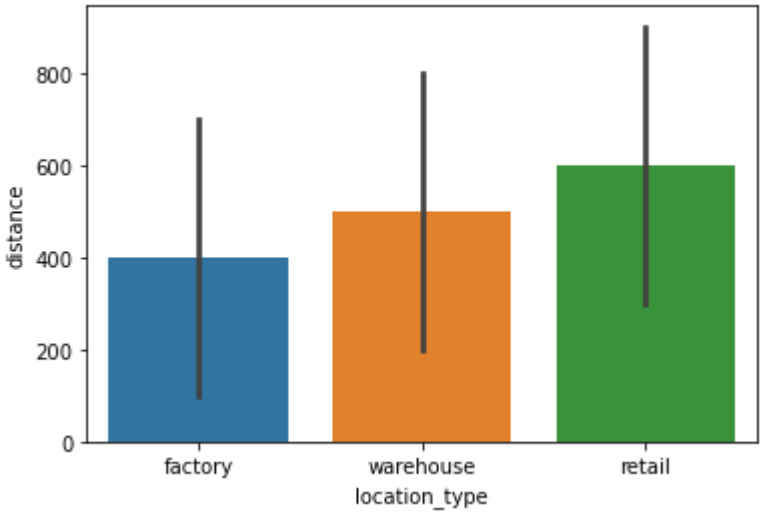
In [92]:
```
fasla_long= fasla.melt(id_vars = "zip",var_name = "location_type", value_name = "distan
fasla_long
```

Out[92]:

	zip	location_type	distance
0	12345	factory	100
1	34567	factory	400
2	67890	factory	700
3	12345	warehouse	200
4	34567	warehouse	500
5	67890	warehouse	800
6	12345	retail	300
7	34567	retail	600
8	67890	retail	900

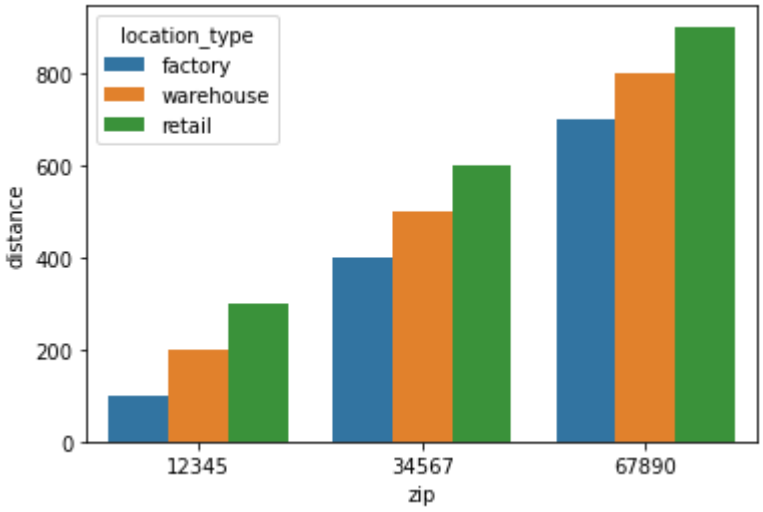
In [93]:
```
sns.barplot(x="location_type",y = "distance", data= fasla_long)
```

Out[93]:
```
<AxesSubplot:xlabel='location_type', ylabel='distance'>
```



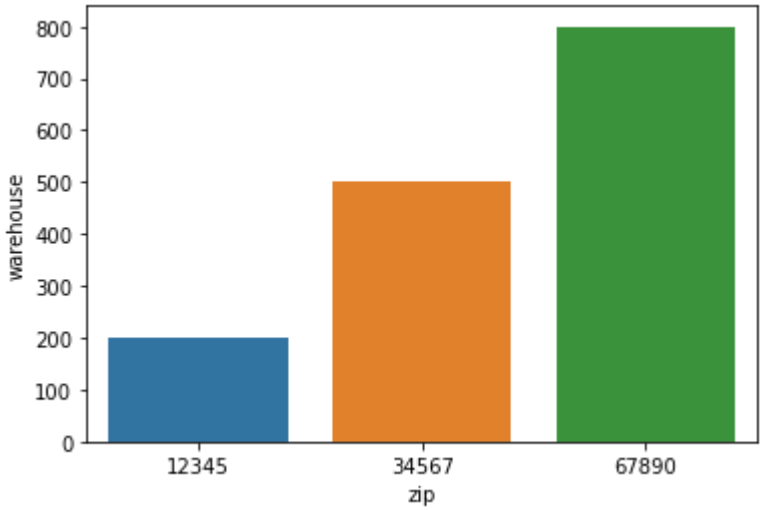
In [94]:
```
sns.barplot(x="zip",y = "distance", hue="location_type" ,data= fasla_long)
```

Out[94]:
```
<AxesSubplot:xlabel='zip', ylabel='distance'>
```



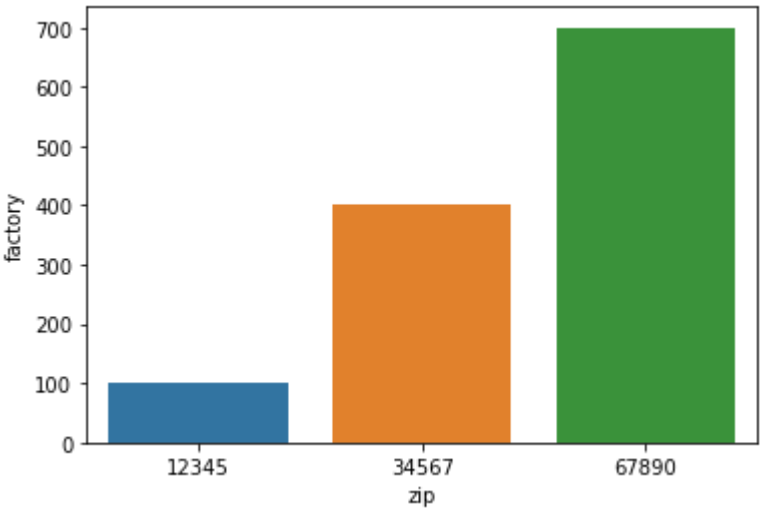
In [95]:
```
sns.barplot(x="zip",y = "warehouse", data= fasla)
```

Out[95]:
```
<AxesSubplot:xlabel='zip', ylabel='warehouse'>
```



In [96]:
```
sns.barplot(x="zip",y = "factory", data= fasla)
```

Out[96]:
```
<AxesSubplot:xlabel='zip', ylabel='factory'>
```



In [ ]: