DATE: __/__/____
PAGE: _____

→ Split, apply and combine.

→ titanic.groupby("pclass")["age"].mean()

finds the mean of age on basis of their class.

→ gbo["age"].mean()
- mean of male & female ages respectively.

gbo["age"].mean().plot(kind="bar")
- plotting bar graph on mean value.

→ gbo = titanic.groupby("sex")["pclass"].mean()

gbo["age"].mean().plot(kind="bar")

→ aggregate

apply many functions on the same group of data with the same line.

titanic.groupby("sex")["age"].agg[

"mean", "median", "sum", "min"...

all these functions will be applied.

Good Write

→ mean of every attribute of either sex of either class.

```
df.groupby(["pclass", "sex"]).mean()
```

→ mean of age

```
df.groupby(["pclass", "sex"])["age"].mean()
```

→ mean of particular age of each sex.

```
df.groupby(["Sex", "age"]).mean()
```

→ Create own multi-index.

for fetching population of a particular state in some random year, we will set index as:

```
df.set_index(["state", 'year'], inplace='True')
```

→ Sorting a Multi-index.

```
df.sort_index()
```
↳ ascending = "True" or ascending = "False"

- for an particular index.

```
df.sort_index(level = 1)
```
↳ state = 0, year = 1

Good Write

- df.sort_index (level = [1, 0], ascending = [False, True])

↳ sort in descending acc to year (1st)

- df.loc [[7]]
  ↳ 7th index
- df.loc [["AK", "CK"]]
  ↳ data of AK and CK
- df.loc ["AK" : "CK"]
  ↳ data from AK to CK (slicing).

→ find mean of the population of MT in 1992.

- df.loc [("MT", 1992)]

⇒ for CA 2013
df.loc [("CA", 2013)]

→ Slice an index from AK 1992 to AK 1995.

df.loc [("AK", 1992) : ("AK", 1995)]

→ for state AK 2011 and for state AL 1993

df.loc [("AK", 2011) : ("AL", 1993)]
all values of AK from 2011
all values of AL from 1993

Good Write

→ df.loc[(:,1990)] → This will give error

→ df.loc[(:, 1990, :)]
- all values of year 1990.

→ data = pd.series(np.random.randn(9), index=
level 0 → [['a','a','a','b','b','c','c','d','d
level 1 → [1,2,3,1,3,1,2,2,3]])

a three times	a	1	3 —
		2	5 —
		3	8 —
b.	b	1	0 —
c,		3	4 —
d	c	1	2 —
2		2	1 —
times	d	2	3 —
		3	6 —

random values from 0-9

→ find values of following:

1) data['b'] →

1	0
3	4

2) data['b',:'c'] →

1	0
3	4
1	2
2	1

3) data.loc[['b','d']] →

1	0
3	4
2	3
3	6

4) data[:,2] →

a	2	5
c	2	1
d	2	3

Good Write

→ ~~frame: pd.Dataframe(12).reshape(4,3)~~

→ frame: pd.Dataframe(np.arange(12).reshape((4,3)), index=[['a','a','b','b'], [1,2,1,2]], columns=[['ohio','ohio','colarado'], ['green','Red','Green']]

table:

		ohio		colarado
		green	red	Green
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

frame.index.names = ['key1', 'key2'] (level 0, level 1)

frame.columns.names = ['state', 'color'] (level 0, level 1)

	state	ohio		colarado
	color	green	red	green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

→ frame['ohio']

key1	key2	State color	ohio green	ohio red
a	1		0	1
	2		3	4
b	1		6	7
	2		9	10

→ frame.swaplevel('key1', 'key2')

key2	key1	State color	ohio green	ohio red	colorado green
1	a		0	1	2
2	a		3	4	5
1	b		6	7	8
2	b		9	10	11

→ frame.sort_index(level=1)

key1	key2	State color	ohio green	ohio red	colorado green
a	1		0	1	2
b	1		6	7	8
a	2		3	4	5
b	2		9	10	11

→ frame.swaplevel(0,1).sort_index(level=0)

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

→ frame = pd.DataFrame({'a': range(7),
take values from 7 to 0 with interval -1 → 'b': range(7, 0, -1),
'c': ['one', 'one', 'one', 'two',
'two', 'two', 'two'],
'd': [0,1,2,0,1,2,3]})

table-

	a	b	c	d
0		7	one	0
1		6	one	1
2		5	one	2
3		4	two	0
4		3	two	1
5		2	two	2
6.		1	two	3

Ques 1 frame2 = frame.set_index(['c','d'])

c	d	a	b
one	0	0	7
	1	1	6
	2	2	5
two	0	3	4
	1	4	3
	2	5	2
	3	6	1

Q: 2 frame2 = frame.set_index(['c', 'd'], drop = false)

c	d	a	b	c	d
one	0	0	7	one	0
	1	1	6	one	1
	2	2	5	one	2
two	0	3	4	two	0
	1	4	3	two	1
	2	5	2	two	2
	3	6	1	two	3