```
import pandas as pd
import numpy as np
```

```
a=pd.Series(np.random.randn(5),index=['a','b','c','d','e'])
b=pd.Series(np.random.randn(5),index=['a','b','c','d','e'])
```

```
a
```

```
a    0.060983
b    0.411111
c   -1.092578
d    0.501245
e    1.490457
dtype: float64
```

```
list(a.values)
```

```
[0.06098314870593321,
 0.4111106258442768,
 -1.0925777334962299,
 0.5012454563318473,
 1.4904574218068243]
```

```
a[a>0] *= -1
```

```
a
```

```
a   -0.060983
b   -0.411111
c   -1.092578
d   -0.501245
e   -1.490457
dtype: float64
```

```
a.abs()
```

```
a    0.060983
b    0.411111
c    1.092578
d    0.501245
e    1.490457
dtype: float64
```

```
d={'one':a,'two':b}
df=pd.DataFrame(d)
```

```
df
```

	one	two
a	-0.060983	-0.803175
b	-0.411111	-0.869954
c	-1.092578	0.698315
d	-0.501245	-1.066385

```
df.columns
```

```
Index(['one', 'two'], dtype='object')
```

```
df
```

	one	two
a	-0.060983	-0.803175
b	-0.411111	-0.869954
c	-1.092578	0.698315
d	-0.501245	-1.066385
e	-1.490457	0.633481

```
df.head(2)
```

	one	two
a	-0.060983	-0.803175
b	-0.411111	-0.869954

```
df.sample(2)
```

	one	two
c	-1.092578	0.698315
a	-0.060983	-0.803175

```
df.head().T
```

	a	b	c	d	e
one	-0.060983	-0.411111	-1.092578	-0.501245	-1.490457
two	-0.803175	-0.869954	0.698315	-1.066385	0.633481

```
import seaborn as sns
```

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

```
tips.head()
```

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

```
tips.groupby(['day']).agg({'tip':'mean','size':'max'})
```

	tip	size
day		
Thur	2.771452	6
Fri	2.734737	4
Sat	2.993103	5
Sun	3.255132	6

```
tips_grp=tips.groupby(['day'])
```

```
tips_grp.agg({'tip':['mean','max']})
```

	tip	
	mean	max
day		
Thur	2.771452	6.70
Fri	2.734737	4.73
Sat	2.993103	10.00
Sun	3.255132	6.50

```
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
```

```
student_data = pd.DataFrame({
    'school_code': ['s001','s002','s003','s001','s002','s004'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill'
    'date_Of_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','1
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},
    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

grp=student_data.groupby('school_code')



grp.agg({'weight':['mean','max']})
```

	weight	
	mean	max
school_code		
s001	32.5	35
s002	31.5	32
s003	33.0	33
s004	32.0	32

```
grp2=student_data.groupby('class')


grp2.agg({'weight':'mean'})
```

	weight
class	
V	32.666667
VI	31.666667

```
grp3=student_data.groupby(['school_code','class'])


grp3.agg({'weight':'mean'})
```

		weight
school_code	class	
s001	V	35.0

```
grp4=student_data.groupby('school_code')
```

s002	V	31.5

```
grp4.get_group('s001')
```

	school_code	class	name	date_Of_Birth	age	height	weight	address
S1	s001	V	Alberto Franco	15/05/2002	12	173	35	street1
S4	s001	VI	Eesha Hinton	25/09/1998	13	167	30	street1

```
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
orders_data = pd.DataFrame({
'ord_no':[70001,70009,70002,70004,70007,70005,70008,70010,70003,70012,70011,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45, 75.29,3045.
'ord_date': ['2012-10-05','2012-09-10','2012-10-05','2012-08-17','2012-09-10','2012-07-27'
'customer_id':[3005,3001,3002,3009,3005,3007,3002,3004,3009,3008,3003,3002],
'salesman_id': [5002,5005,5001,5003,5002,5001,5001,5006,5003,5002,5007,5001]})
print("Original Orders DataFrame:")
```

```
Original Orders DataFrame:
```

```
orders_data.shape
```

```
(12, 5)
```

```
orders_data.head(2)
```

	ord_no	purch_amt	ord_date	customer_id	salesman_id
0	70001	150.50	2012-10-05	3005	5002
1	70009	270.65	2012-09-10	3001	5005

```
grp=orders_data.groupby('ord_no')
```

```
grp.agg({'purch_amt':['mean','max','min']})
```

	purch_amt		
	mean	max	min
ord_no			
70001	150.50	150.50	150.50
70002	65.26	65.26	65.26
70003	2480.40	2480.40	2480.40
70004	110.50	110.50	110.50
70005	2400.60	2400.60	2400.60
70007	948.50	948.50	948.50
70008	5760.00	5760.00	5760.00
70009	270.65	270.65	270.65
70010	1983.43	1983.43	1983.43

```
grp=orders_data.groupby(['salesman_id','customer_id'])
```

```
grp.size().reset_index().groupby(['salesman_id','customer_id'])[[0]].max()
```

		0
salesman_id	customer_id	
5001	3002	3
	3007	1
5002	3005	2
	3008	1
5003	3009	2
5005	3001	1
5006	3004	1
5007	3003	1

```
grp=orders_data.groupby(['salesman_id','customer_id'])

res=grp.agg({'purch_amt':'sum'})

res1=res['purch_amt'].groupby(level=0,group_keys=False)

res1.nlargest()
```

```
     salesman_id  customer_id
     5001         3002           8870.86
```

```
                3007          2400.60
5002            3005          1099.00
                3008           250.45
5003            3009          2590.90
5005            3001           270.65
5006            3004          1983.43
5007            3003            75.29
Name: purch_amt, dtype: float64
```

```
df=orders_data.copy()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ord_no       12 non-null     int64
 1   purch_amt    12 non-null     float64
 2   ord_date     12 non-null     object
 3   customer_id  12 non-null     int64
 4   salesman_id  12 non-null     int64
dtypes: float64(1), int64(3), object(1)
memory usage: 608.0+ bytes
```

```
df['ord_date']=pd.to_datetime(df['ord_date'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ord_no       12 non-null     int64
 1   purch_amt    12 non-null     float64
 2   ord_date     12 non-null     datetime64[ns]
 3   customer_id  12 non-null     int64
 4   salesman_id  12 non-null     int64
dtypes: datetime64[ns](1), float64(1), int64(3)
memory usage: 608.0 bytes
```

```
df1=df.set_index('ord_date').groupby(pd.Grouper(freq='1M')).agg({'purch_amt':'sum'})
```

```
df1
```

	purch_amt
ord_date	
2012-04-30	3045.60
2012-05-31	0.00
2012-06-30	250.45

```
df.head()
```

	ord_no	purch_amt	ord_date	customer_id	salesman_id
0	70001	150.50	2012-10-05	3005	5002
1	70009	270.65	2012-09-10	3001	5005
2	70002	65.26	2012-10-05	3002	5001
3	70004	110.50	2012-08-17	3009	5003
4	70007	948.50	2012-09-10	3005	5002

```
df.groupby([df['ord_date'].dt.year,df['ord_date'].dt.month]).agg({'purch_amt':'sum'})
```

		purch_amt
ord_date	ord_date	
2012	4	3045.60
	6	250.45
	7	2400.60
	8	185.79
	9	6979.15
	10	4679.59

```
df = pd.DataFrame( {'id' : [1, 2, 1, 1, 2, 1, 2],
                    'type' : [10, 15, 11, 20, 21, 12, 14],
                    'book' : ['Math','English','Physics','Math','English','Physics','Engli
```

```
df.head()
```

	id	type	book
0	1	10	Math

```
df = pd.DataFrame({
    'id': [1, 1, 2, 3, 3, 4, 4, 4],
    'value': ['a', 'a', 'b', None, 'a', 'a', None, 'b']
})
```

4	2	21	English

```
df.head()
```

	id	value
0	1	a
1	1	a
2	2	b
3	3	None
4	3	a

```
df.groupby('value')['id'].nunique()
```

```
value
a    3
b    2
Name: id, dtype: int64
```

```
df = pd.DataFrame({
    'school_code': ['s001','s002','s003','s001','s002','s004'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill'
    'date_Of_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','1
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},
    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])
```

```
gp = df.groupby('school_code')
```

```
gp.groups.keys()
```

```
dict_keys(['s001', 's002', 's003', 's004'])
```

```
import pandas as pd
pd.set_option('display.max_rows', None)
df = pd.DataFrame({
'book_name':['Book1','Book2','Book3','Book4','Book1','Book2','Book3','Book5'],
'book_type':['Math','Physics','Computer','Science','Math','Physics','Computer','English'],
```

```
'book_id':[1,2,3,4,1,2,3,5]})
print("Original Orders DataFrame:")
print(df)
```

```
Original Orders DataFrame:
  book_name book_type  book_id
0     Book1      Math        1
1     Book2   Physics        2
2     Book3  Computer        3
3     Book4   Science        4
4     Book1      Math        1
5     Book2   Physics        2
6     Book3  Computer        3
7     Book5   English        5
```

```
df.groupby(['book_name','book_type'])['book_type'].count().reset_index(name="count")
```

	book_name	book_type	count
0	Book1	Math	2
1	Book2	Physics	2
2	Book3	Computer	2
3	Book4	Science	1
4	Book5	English	1

```
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
    'school_code': ['s001','s002','s003','s001','s002','s001'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill'
    'date_Of_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','1
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},
    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])
print("Original DataFrame:")
print(df)
```

```
Original DataFrame:
   school_code class            name date_Of_Birth   age  height  weight  \
S1        s001     V  Alberto Franco     15/05/2002   12     173      35
S2        s002     V    Gino Mcneill     17/05/2002   12     192      32
S3        s003    VI     Ryan Parkes     16/02/1999   13     186      33
S4        s001    VI    Eesha Hinton     25/09/1998   13     167      30
S5        s002     V    Gino Mcneill     11/05/2002   14     151      31
S6        s001    VI    David Parkes     15/09/1997   12     159      32

    address
S1  street1
```

```
S2  street2
S3  street3
S4  street1
S5  street2
S6  street4
```

```
df.groupby(['school_code','class']).agg({'height':['mean','min','max'],'age':['mean','min'
```

		height			age		
		mean	min	max	mean	min	max
school_code	class						
s001	V	173.0	173	173	12.0	12	12
	VI	163.0	159	167	12.5	12	13
s002	V	171.5	151	192	13.0	12	14
s003	VI	186.0	186	186	13.0	13	13

```
df = pd.DataFrame( {'id' : ['A','A','A','A','A','A','B','B','B','B','B'],
                    'type' : [1,1,1,1,2,2,1,1,1,2,2],
                    'book' : ['Math','Math','English','Physics','Math','English','Physics'
```

```
df.head()
```

	id	type	book
0	A	1	Math
1	A	1	Math
2	A	1	English
3	A	1	Physics
4	A	2	Math

```
df1=df.groupby(['id','type'])['book'].unique().reset_index(name='book')
```

```
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
'ord_no':[70001,70009,70002,70004,70007,70005,70008,70010,70003,70012,70011,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45, 75.29,3045.
'ord_date': ['05-10-2012','09-10-2012','05-10-2012','08-17-2012','10-09-2012','07-27-2012'
'customer_id':[3001,3001,3005,3001,3005,3001,3005,3001,3005,3001,3005,3005],
'salesman_id': [5002,5005,5001,5003,5002,5001,5001,5006,5003,5002,5007,5001]})
print("Original Orders DataFrame:")
print(df)
```

```
df['ord_date']= pd.to_datetime(df['ord_date'])
```

```
Original Orders DataFrame:
    ord_no  purch_amt    ord_date  customer_id  salesman_id
0    70001     150.50  05-10-2012         3001         5002
1    70009     270.65  09-10-2012         3001         5005
2    70002      65.26  05-10-2012         3005         5001
3    70004     110.50  08-17-2012         3001         5003
4    70007     948.50  10-09-2012         3005         5002
5    70005    2400.60  07-27-2012         3001         5001
6    70008    5760.00  10-09-2012         3005         5001
7    70010    1983.43  10-10-2012         3001         5006
8    70003    2480.40  10-10-2012         3005         5003
9    70012     250.45  06-17-2012         3001         5002
10   70011      75.29  07-08-2012         3005         5007
```

```
grp=df.groupby(['customer_id','salesman_id']).agg({'purch_amt':'sum'})
```

```
grp['perc']=grp.apply(lambda x: 100*x / x.sum())
```

```
grp
```

		purch_amt	perc
customer_id	**salesman_id**		
3001	**5001**	2400.60	13.685510
	5002	400.95	2.285764
	5003	110.50	0.629946
	5005	270.65	1.542941
	5006	1983.43	11.307278
3005	**5001**	8870.86	50.571626
	5002	948.50	5.407276
	5003	2480.40	14.140440
	5007	75.29	0.429219

```
import pandas as pd
student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame
        'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William',
        'marks': [201, 200, 198, 219, 201]})
```

```
merged_data=pd.merge(student_data1,student_data2,on='student_id')
```

```
merged_data.head()
```

	student_id	name_x	marks_x	name_y	marks_y
0	S4	Ed Bernal	222	Scarlette Fisher	201
1	S5	Kwame Morin	199	Carla Williamson	200

```
data1 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                      'key2': ['K0', 'K1', 'K0', 'K1'],
                      'P': ['P0', 'P1', 'P2', 'P3'],
                      'Q': ['Q0', 'Q1', 'Q2', 'Q3']})
data2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'R': ['R0', 'R1', 'R2', 'R3'],
                      'S': ['S0', 'S1', 'S2', 'S3']})
print("Original DataFrames:")
```

```
Original DataFrames:
```

```
merge=pd.merge(data1,data2,on=['key1','key2'],how='right')
```

```
merge
```

	key1	key2	P	Q	R	S
0	K0	K0	P0	Q0	R0	S0
1	K1	K0	P2	Q2	R1	S1
2	K1	K0	P2	Q2	R2	S2
3	K2	K0	NaN	NaN	R3	S3

```
data1
```

	key1	key2	P	Q
0	K0	K0	P0	Q0
1	K0	K1	P1	Q1
2	K1	K0	P2	Q2
3	K2	K1	P3	Q3

```
data2
```

	key1	key2	R	S
0	K0	K0	R0	S0
1	K1	K0	R1	S1
2	K1	K0	R2	S2
3	K2	K0	R3	S3

```
import pandas as pd
df=pd.read_csv('Automobile_data.csv')
```

```
df.head()
```

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average mileag
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	2
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	2
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	1

```
45400.0
```

```
df[df['price']==df.price.max()][['company','price']]
```

	company	price
35	mercedes-benz	45400.0

```
df['company'].value_counts()
```

```
toyota          7
bmw             6
mazda           5
nissan          5
audi            4
mercedes-benz   4
mitsubishi      4
volkswagen      4
alfa-romero     3
chevrolet       3
honda           3
isuzu           3
jaguar          3
porsche         3
```

```
dodge            2
volvo            2
Name: company, dtype: int64
```

```
df_grp=df.groupby('company')
```

```
df.sort_values(by=['price'])
```

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	avera mile
13	16	chevrolet	hatchback	88.4	141.1	l	three	48	
27	36	mazda	hatchback	93.1	159.1	ohc	four	68	
48	66	toyota	hatchback	95.7	158.7	ohc	four	62	
36	49	mitsubishi	hatchback	93.7	157.3	ohc	four	68	
28	37	mazda	hatchback	93.1	159.1	ohc	four	68	
...	...	...	...	...	...	...	...	...	
11	14	bmw	sedan	103.5	193.8	ohc	six	182	
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	
22	31	isuzu	sedan	94.5	155.9	ohc	four	70	
23	32	isuzu	sedan	94.5	155.9	ohc	four	70	
47	63	porsche	hatchback	98.4	175.7	dohcv	eight	288	

```
GermanCars = {'Company': ['Ford', 'Mercedes', 'BMV', 'Audi'], 'Price': [23845, 171995, 135
japaneseCars = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80, 182
```

```
df1=pd.DataFrame(GermanCars)
df2=pd.DataFrame(japaneseCars)
```

```
pd.concat([df1,df2], keys=["Germany", "Japan"])
```

		Company	Price	horsepower
Germany	**0**	Ford	23845.0	NaN
	1	Mercedes	171995.0	NaN
	2	BMV	135925.0	NaN

```
df1 = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'Price': [23845, 17995, 135925 , 714
df2 = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80, 182 , 160]}
df1=pd.DataFrame(df1)
df2=pd.DataFrame(df2)
```

```
pd.merge(df1,df2,on='Company',how='inner')
```

	Company	Price	horsepower
0	Toyota	23845	141
1	Honda	17995	80
2	BMV	135925	182
3	Audi	71400	160

```
df2
```

	Company	Price
0	Toyota	29995
1	Honda	23600
2	Nissan	61500
3	Mitsubishi	58900

```
url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv'

chipo = pd.read_csv(url, sep = '\t')

chipo.head()
```

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39

```
df=chipo.copy()
```

```
price=[float(x[1:]) for x in df.item_price]
```

```
df.item_price=price
```

```
len(df[df['item_price']>10])
```

```
1130
```

```
df.shape
```

```
(4622, 5)
```

```
df['item_name'].nunique()
```

```
50
```

```
df1=df.drop_duplicates(['item_name','quantity'])
```

```
df1.shape
```

```
(103, 5)
```

```
df[(df['item_name']=='Chicken Bowl') & (df['quantity']==1)]
```

	order_id	quantity	item_name	choice_description	item_price
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	10.98
13	7	1	Chicken Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
19	10	1	Chicken Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	8.75
26	13	1	Chicken Bowl	[Roasted Chili Corn Salsa (Medium), [Pinto Bea...	8.49
42	20	1	Chicken Bowl	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	11.25
...	...	...	...	...	...
4590	1825	1	Chicken Bowl	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	11.25
4591	1825	1	Chicken Bowl	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	8.75

```
df.sort_values(by='item_name')
```

	order_id	quantity	item_name	choice_description	item_price
3389	1360	2	6 Pack Soft Drink	[Diet Coke]	12.98
341	148	1	6 Pack Soft Drink	[Diet Coke]	6.49
1849	749	1	6 Pack Soft Drink	[Coke]	6.49
1860	754	1	6 Pack Soft Drink	[Diet Coke]	6.49
2713	1076	1	6 Pack Soft Drink	[Coke]	6.49
...	...	...	...	...	...
2384	948	1	Veggie Soft Tacos	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	8.75
781	322	1	Veggie Soft Tacos	[Fresh Tomato Salsa, [Black Beans, Cheese, Sou...	8.75

```
df.sort_values(by='item_price',ascending=False).head(1)['quantity']
```

```
3598    15
Name: quantity, dtype: int64
```

```
df[df['item_price']==df['item_price'].max()]['quantity']
```

```
3598    15
Name: quantity, dtype: int64
```

```
len(df[(df['item_name']=='Canned Soda') & (df['quantity']>1)])
```

```
20
```

```
euro12 = pd.read_csv('https://raw.githubusercontent.com/guipsamora/pandas_exercises/master
euro12.head()
```

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals	Pe
0	Croatia	4	13	12	51.9%	16.0%	32	0	0	

```
df=euro12.copy()
```

```
df['Team'].nunique()
```

```
16
```

```
rb=df[['Team','Yellow Cards', 'Red Cards']]
```

```
rb.sort_values(by=['Red Cards','Yellow Cards'],ascending=False)
```

	Team	Yellow Cards	Red Cards
6	Greece	9	1
9	Poland	7	1
11	Republic of Ireland	6	1
7	Italy	16	0
10	Portugal	12	0
13	Spain	11	0
0	Croatia	9	0
1	Czech Republic	7	0
14	Sweden	7	0
4	France	6	0
12	Russia	6	0
3	England	5	0
8	Netherlands	5	0
15	Ukraine	5	0
2	Denmark	4	0
5	Germany	4	0

```
df[df['Goals']>6]
```

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals	P
5	Germany	10	32	32	47.8%	15.6%	80	2	1	
13	Spain	12	42	33	55.9%	16.0%	100	0	1	

2 rows × 35 columns

```
df[df['Team'].str.startswith('G')]
```

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals	Pe
5	Germany	10	32	32	47.8%	15.6%	80	2	1	
6	Greece	5	8	18	30.7%	19.2%	32	1	1	

2 rows × 35 columns

```
df.iloc[:,:-3]
```

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals
0	Croatia	4	13	12	51.9%	16.0%	32	0	0
1	Czech Republic	4	13	18	41.9%	12.9%	39	0	0
2	Denmark	4	10	10	50.0%	20.0%	27	1	0
3	England	5	11	18	50.0%	17.2%	40	0	0

```
df[df['Team'].isin(['England', 'Italy', 'Russia'])][['Team','Shooting Accuracy']]
```

	Team	Shooting Accuracy
3	England	50.0%
7	Italy	43.0%
12	Russia	22.5%

```
import pandas as pd
drinks = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/drink
drinks.head()
```

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_a
0	Afghanistan	0	0	0	
1	Albania	89	132	54	
2	Algeria	25	0	14	
3	Andorra	245	138	312	
4	Angola	217	57	45	

```
df=drinks.copy()
```

```
df.groupby('continent').agg({'beer_servings':'mean'}).max()
```

```
beer_servings    193.777778
dtype: float64
```

```
df.groupby('continent').wine_servings.describe()
```

continent	count	mean	std	min	25%	50%	75%	max
AF	53.0	16.264151	38.846419	0.0	1.0	2.0	13.00	233.0
AS	44.0	9.068182	21.667034	0.0	0.0	1.0	8.00	123.0

```
df.groupby('continent').mean()
```

continent	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcoh
AF	61.471698	16.339623	16.264151	3.0075
AS	37.045455	60.840909	9.068182	2.1704
EU	193.777778	132.555556	142.222222	8.6177
OC	89.687500	58.437500	35.625000	3.3812
SA	175.083333	114.750000	62.416667	6.3083

```
df.groupby('continent').agg({'spirit_servings':['mean', 'min', 'max']})
```

continent	spirit_servings		
	mean	min	max
AF	16.339623	0	152
AS	60.840909	0	326
EU	132.555556	0	373
OC	58.437500	0	254
SA	114.750000	25	302

```
df.head()
```

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_a
0	Afghanistan	0	0	0	
1	Albania	89	132	54	
2	Algeria	25	0	14	
3	Andorra	245	138	312	
4	Angola	217	57	45	

```
users = pd.read_table('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.us
                      sep='|', index_col='user_id')
```

```
users.head()
```

user_id	age	gender	occupation	zip_code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

```
df=users.copy()
```

```
df['sex_n']=df['gender'].apply(lambda x:1 if x=='M' else 0 )
```

```
df['sex_n'].head()
```

```
user_id
1    1
2    0
3    1
4    1
5    0
Name: sex_n, dtype: int64
```

```
a=df.groupby('occupation').sex_n.sum()/df.occupation.value_counts()*100
```

```
a.sort_values(ascending=False)
```

```
doctor           100.000000
engineer          97.014925
technician        96.296296
retired           92.857143
programmer        90.909091
executive         90.625000
scientist         90.322581
entertainment     88.888889
lawyer            83.333333
salesman          75.000000
educator          72.631579
student           69.387755
other             65.714286
marketing         61.538462
writer            57.777778
none              55.555556
administrator     54.430380
artist            53.571429
librarian         43.137255
healthcare        31.250000
```

```
homemaker          14.285714
```

```
df.groupby('occupation').age.agg(['min','max'])
```

occupation	min	max
administrator	21	70
artist	19	48
doctor	28	64
educator	23	63
engineer	22	70
entertainment	15	50
executive	22	69
healthcare	22	62
homemaker	20	50
lawyer	21	53
librarian	23	69
marketing	24	55
none	11	55
other	13	64
programmer	20	63
retired	51	73
salesman	18	66
scientist	23	55
student	7	42
technician	21	55
writer	18	60

```
df.head(2)
```

user_id	age	gender	occupation	zip_code	sex_n
1	24	M	technician	85711	1
2	53	F	other	94043	0

```
oc_cnt=df.groupby(['occupation','gender']).agg({'gender':'count'})
```

```
oc_cnt1=df.groupby(['occupation']).agg('count')
```

```
oc_cnt.div(oc_cnt1,level="occupation").loc[:,'gender']*100
```

```
occupation     gender
administrator  F          45.569620
               M          54.430380
artist         F          46.428571
               M          53.571429
doctor         M         100.000000
educator       F          27.368421
               M          72.631579
engineer       F           2.985075
               M          97.014925
entertainment  F          11.111111
               M          88.888889
executive      F           9.375000
               M          90.625000
healthcare     F          68.750000
               M          31.250000
homemaker      F          85.714286
               M          14.285714
lawyer         F          16.666667
               M          83.333333
librarian      F          56.862745
               M          43.137255
marketing      F          38.461538
               M          61.538462
none           F          44.444444
               M          55.555556
other          F          34.285714
               M          65.714286
programmer     F           9.090909
               M          90.909091
retired        F           7.142857
               M          92.857143
salesman       F          25.000000
               M          75.000000
scientist      F           9.677419
               M          90.322581
student        F          30.612245
               M          69.387755
technician     F           3.703704
               M          96.296296
writer         F          42.222222
               M          57.777778
Name: gender, dtype: float64
```

```
raw_data = {'regiment': ['Nighthawks', 'Nighthawks', 'Nighthawks', 'Nighthawks', 'Dragoons
        'company': ['1st', '1st', '2nd', '2nd', '1st', '1st', '2nd', '2nd','1st', '1st', '
        'name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze', 'Jacon', 'Ryaner', 'Sone'
        'preTestScore': [4, 24, 31, 2, 3, 4, 24, 31, 2, 3, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70, 25, 94, 57, 62, 70, 62, 70]}
```

```
df = pd.DataFrame(raw_data, columns = raw_data.keys())
```

```
df.head(2)
```

	regiment	company	name	preTestScore	postTestScore
0	Nighthawks	1st	Miller	4	25
1	Nighthawks	1st	Jacobson	24	94

```
df.groupby('regiment').agg({'preTestScore':'mean'})
```

regiment	preTestScore
Dragoons	15.50
Nighthawks	15.25
Scouts	2.50

```
df.groupby('company').describe()
```

	preTestScore								postTestScore		
	count	mean	std	min	25%	50%	75%	max	count	mean	st
company											
1st	6.0	6.666667	8.524475	2.0	3.00	3.5	4.00	24.0	6.0	57.666667	27
2nd	6.0	15.500000	14.652645	2.0	2.25	13.5	29.25	31.0	6.0	67.000000	14

```
df.groupby('company').preTestScore.mean()
```

```
company
1st     6.666667
2nd    15.500000
Name: preTestScore, dtype: float64
```

```
df.groupby(['company','regiment']).preTestScore.mean().unstack()
```

regiment	Dragoons	Nighthawks	Scouts
company			
1st	3.5	14.0	2.5
2nd	27.5	16.5	2.5

```
df.groupby(['company','regiment']).size()
```

```
company  regiment
1st      Dragoons      2
         Nighthawks    2
```

```
          Scouts        2
2nd       Dragoons      2
          Nighthawks    2
          Scouts        2
dtype: int64
```

```
df.head()
```

	regiment	company	name	preTestScore	postTestScore
0	Nighthawks	1st	Miller	4	25
1	Nighthawks	1st	Jacobson	24	94
2	Nighthawks	2nd	Ali	31	57
3	Nighthawks	2nd	Milner	2	62
4	Dragoons	1st	Cooze	3	70

```
s3=df.sort_values(by='postTestScore',ascending=False)['postTestScore'].iloc[3]
```

```
res=df.set_index('postTestScore').loc[s3]
```

```
res.iloc[:,]
```

postTestScore	regiment	company	name	preTestScore
70	Dragoons	1st	Cooze	3
70	Scouts	1st	Piger	3
70	Scouts	2nd	Ali	3

```
import pandas as pd
df=pd.read_csv('states.csv')
```

```
df.head()
```

	Date	State	Confirmed	Recovered	Deceased	Other	Tested
0	2020-01-30	Kerala	1	0	0	0	NaN
1	2020-01-30	India	1	0	0	0	NaN
2	2020-02-02	Kerala	2	0	0	0	NaN
3	2020-02-02	India	2	0	0	0	NaN
4	2020-02-03	Kerala	3	0	0	0	NaN

```
df.Confirmed.value_counts()
```

```
1          252
2          113
0          100
7           85
3           69
          ...
466991       1
4682         1
865705       1
16204        1
1592908      1
Name: Confirmed, Length: 18355, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21675 entries, 0 to 21674
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       21675 non-null  object 
 1   State      21675 non-null  object 
 2   Confirmed  21675 non-null  int64  
 3   Recovered  21675 non-null  int64  
 4   Deceased   21675 non-null  int64  
 5   Other      21675 non-null  int64  
 6   Tested     20912 non-null  float64
dtypes: float64(1), int64(4), object(2)
memory usage: 1.2+ MB
```

```
df['Date']=pd.to_datetime(df['Date'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21675 entries, 0 to 21674
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       21675 non-null  datetime64[ns]
 1   State      21675 non-null  object        
 2   Confirmed  21675 non-null  int64         
 3   Recovered  21675 non-null  int64         
 4   Deceased   21675 non-null  int64         
 5   Other      21675 non-null  int64         
 6   Tested     20912 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(4), object(1)
memory usage: 1.2+ MB
```

```
df.head(2)
```

	Date	State	Confirmed	Recovered	Deceased	Other	Tested

```
df['year_Month']=df.Date.dt.strftime('%Y-%m')
```

1	2020-01-30	India	1	0	0	0	NaN

```
df.head(2)
```

	Date	State	Confirmed	Recovered	Deceased	Other	Tested	year_Month
0	2020-01-30	Kerala	1	0	0	0	NaN	2020-01
1	2020-01-30	India	1	0	0	0	NaN	2020-01

```
df.groupby(['State','year_Month'],as_index=False).agg({'Confirmed':'mean'})
```

	State	year_Month	Confirmed
0	Andaman and Nicobar Islands	2020-03	7.500000e+00
1	Andaman and Nicobar Islands	2020-04	1.700000e+01
2	Andaman and Nicobar Islands	2020-05	3.300000e+01
3	Andaman and Nicobar Islands	2020-06	4.763333e+01
4	Andaman and Nicobar Islands	2020-07	2.193226e+02
...	...	...	...
725	West Bengal	2021-06	1.460300e+06
726	West Bengal	2021-07	1.515897e+06
727	West Bengal	2021-08	1.538948e+06
728	West Bengal	2021-09	1.559141e+06
729	West Bengal	2021-10	1.580588e+06

730 rows × 3 columns

```
df.State.drop_duplicates()
```

```
0                                  Kerala
1                                   India
8                                   Delhi
10                              Telangana
14                              Rajasthan
18                                Haryana
23                          Uttar Pradesh
41                                 Ladakh
44                              Tamil Nadu
58                      Jammu and Kashmir
59                              Karnataka
62                            Maharashtra
```

```
63                                  Punjab
95                           Andhra Pradesh
125                        Himachal Pradesh
153                              Uttarakhand
163                                   Odisha
182                               Puducherry
189                              West Bengal
210                               Chandigarh
211                             Chhattisgarh
213                                  Gujarat
243                           Madhya Pradesh
278                                    Bihar
338                                  Manipur
355                                      Goa
366                                  Mizoram
377          Andaman and Nicobar Islands
519                                    Assam
528                                Jharkhand
579                          Arunachal Pradesh
722                                 Nagaland
729                                  Tripura
808      Dadra and Nagar Haveli and Daman and Diu
955                                Meghalaya
1699                                  Sikkim
2244                       State Unassigned
11074                           Lakshadweep
Name: State, dtype: object
```

```
df[(df['State']=='Goa') | (df['State']=='Maharashtra')]
```

	Date	State	Confirmed	Recovered	Deceased	Other	Tested	year_Mon
62	2020-03-09	Maharashtra	2	0	0	0	NaN	2020-
75	2020-03-10	Maharashtra	5	0	0	0	NaN	2020-
88	2020-03-11	Maharashtra	11	0	0	0	NaN	2020-
102	2020-03-12	Maharashtra	14	0	0	0	NaN	2020-
116	2020-03-13	Maharashtra	17	0	0	0	NaN	2020-
...	...	...	...	...	...	...	...	
21583	2021-10-29	Maharashtra	6609292	6447038	140170	3619	62439900.0	2021-
21610	2021-10-30	Goa	178085	174339	3364	0	1466038.0	2021-

```
sc=df.groupby('State',as_index=False).agg({'Confirmed':'mean','Recovered':'mean'})
```

```
sc.head()
```

	State	Confirmed	Recovered
0	Andaman and Nicobar Islands	4379.210256	4205.738462
1	Andhra Pradesh	933748.871452	893478.210351
2	Arunachal Pradesh	20053.852941	18902.238754
3	Assam	256571.387931	242595.243103
4	Bihar	328151.838710	314659.419355

```
sc['rec_rate']=round(sc['Recovered']/sc['Confirmed']*100,2)
```

```
sc
```

	State	Confirmed	Recovered	rec_rate
0	Andaman and Nicobar Islands	4.379210e+03	4.205738e+03	96.04
1	Andhra Pradesh	9.337489e+05	8.934782e+05	95.69
2	Arunachal Pradesh	2.005385e+04	1.890224e+04	94.26
3	Assam	2.565714e+05	2.425952e+05	94.55
4	Bihar	3.281518e+05	3.146594e+05	95.89
5	Chandigarh	2.731641e+04	2.594569e+04	94.98
6	Chhattisgarh	4.158606e+05	3.933605e+05	94.59
7	Dadra and Nagar Haveli and Daman and Diu	4.968734e+03	4.767277e+03	95.95

```
import pandas as pd
```

9	Goa	7.274961e+04	6.840625e+04	94.03

```
df=pd.DataFrame(data=[('121', 'US', 'approved', '1000', '2018-12-18'),
  ('122', 'US', 'declined', '2000', '2018-12-19'),
  ('123', 'US', 'approved', '2000', '2019-01-01'),
  ('124', 'DE', 'approved', '2000', '2019-01-07')],
                columns=('id', 'country', 'state', 'amount', 'trans_date'))
```

13		India	1.335854e+07	1.259547e+07	94.29

```
df.head()
```

	id	country	state	amount	trans_date
0	121	US	approved	1000	2018-12-18
1	122	US	declined	2000	2018-12-19
2	123	US	approved	2000	2019-01-01
3	124	DE	approved	2000	2019-01-07

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          4 non-null      object
 1   country     4 non-null      object
 2   state       4 non-null      object
 3   amount      4 non-null      object
 4   trans_date  4 non-null      datetime64[ns]
 5   month       4 non-null      object
dtypes: datetime64[ns](1), object(5)
memory usage: 320.0+ bytes
```

30		Sikkim	1.044720e+04	9.333645e+03	89.34

```
df['trans_date']=pd.to_datetime(df['trans_date'])
```

```
df['month']=df['trans_date'].dt.strftime('%Y-%m')
```

```
df.groupby(['month','country']).agg({'amount':'sum','id':'count'}).reset_index()
```

	month	country	amount	id
0	2018-12	US	10002000	2
1	2019-01	DE	2000	1
2	2019-01	US	2000	1

```
x=10
y=100%90
if(x!=y):
  print(x,y)
else:
  print(x,y)
```

```
10 10
```

```
words = ['eat', 'sleep', 'drink', 'sleep', 'drink', 'sleep', 'go', 'come']
element = 'sleep'

# reversing the list
words.reverse()

# finding the index of element
index = words.index(element)

# printing the final index
print(len(words) - index - 1)
```

```
5
```

```
a=list(map(int,input().split(' ')))
b=list(map(int,input().split(' ')))
```

```
8
1 2 3 4 6 6 7 6
```

```
b
```

```
[1, 2, 3, 4, 6, 6, 7, 6]
```

```
b
```

```
[1, 2, 3, 4, 6, 6, 7, 6]
```

```
b.reverse()
```

```
index=b.index(6)
print(len(b)-index-1)
```

```
7
```

```
dic
```

```
{}
```

```
b
```

```
[6, 7, 6, 6, 4, 3, 2, 1]
```

```
b=[1,2,3,4,5]
dic={}
for i in b:
  if i not in dic:
    dic[i]=0
  else:
    dic[i]+=1

if max(list(dic.values()))==0:
  print(-1)
else:
  index=b.index(6)
  print(len(b)-index-1)
```

```
-1
```

```
k=[]



# finding the index of element
index = b.index(a[1])
# printing the final index

print(index,result)
k=[]
for i in b:
  k.append(b.index(a[1]))

b.reverse()
result=len(b) - index

if len(k)==1:
  print(-1)
else:
  print(result)
```

```
5 8
3
```

```
# Here we are using generator as yield()
#Using generator memory will be saved & spped is fast
def square(lst):
  for i in lst:
    yield(i*i)

square_list=square([1,2,3,4,5])
for i in square_list:
  print(i,end=",")
```

```
    1,4,9,16,25,
```

```
# Here without Generator we are soing squaring
def square(lst):
  result=[]
  for i in lst:
    result.append(i*i)
  return result

square_list=square([1,2,3,4,5])
for i in square_list:
  print(i,end=",")
```

```
    1,4,9,16,25,
```

```
# Here we are using generator in list comprension
# ()--> is generator
square_list1=(i*i for i in [1,2,3,4,5])
for i in square_list1:
  print(i,end=",")
```

```
    1,4,9,16,25,
```

```
# Program to print the tuple using Iterator protocols
tup = (87, 90, 100, 500)

# get an iterator using iter()
tup_iter = iter(tup)

# Infinite loop
while True:
   try:
       # To fetch the next element
       print(next(tup_iter),end=" ")
       # if exception is raised, break from the loop
   except StopIteration:
       break
```

```
    87 90 100 500
```

```
def outer_func(msg):
  def inner_func():
    print(msg)
  return inner_func

result=outer_func("Hello")
result()
```

```
Hello
```

```
# len() function is used for a string
print (len("Javatpoint"))

# len() function is used for a list
print (len([110, 210, 130, 321]))
```

```
10
4
```

```
print("Hello")
```

```
Hello
```

```
# Hello Baliram Pinate How are you doing?
# I am doing fine how you doing.
```

```
a=[1,2,3,4,2,3,7,8]
```

```
dic={}
for i in a:
  if i not in dic:
    dic[i]=0
  else:
    if i in dic:
      dic.get(i)==0
      print(i)
      break
```

```
2
```

Colab paid products - Cancel contracts here