```
means
```

Out [5]:
```
key1  key2
a     1       0.625748
      2       0.722116
b     1      -0.403888
      2       1.294092
Name: data1, dtype: float64
```

In [6]:
```
means.unstack()
```

Out [6]:

key2	1	2
key1		
a	0.625748	0.722116
b	-0.403888	1.294092

In [7]:
```
#길이만 같으면 어떤 배열이라도 상관없음.

states=np.array(["OH","CA","CA","OH","OH","CA","OH"])
years=[2005,2005,2006,2005,2006,2005,2006]
df["data1"].groupby([states,years]).mean()
```

Out [7]:
```
CA  2005   -0.648278
    2006    0.326834
OH  2005    0.959920
    2006   -0.190290
Name: data1, dtype: float64
```

In [8]:
```
df.groupby("key1").mean()
```

Out [8]:

	key2	data1	data2
key1			
a	1.5	-0.223603	-0.326566
b	1.5	0.445102	1.104837

In [9]:
```
df.groupby("key2").mean(numeric_only=True) #numeric_only:오직 숫자, 소수, bool값만 있는 열에 대해서 카운트, data1은 숫자 데이터 X
```

Out [9]:

	data1	data2
key2		
1	0.143001	0.563415

In [36]:
```
grouped=df.groupby("key1")
grouped["data1"].nsmallest(2) #nsmallest : 오름차순으로 정렬한 후, n 개 가져오기
```

Out [36]:
```
key1
a     5   -2.018672
      0    0.625748
b     4   -0.403888
      3    1.294092
Name: data1, dtype: float64
```

In [37]:
```
def peak_to_peak(arr):
    return arr.max()-arr.min()

grouped.agg(peak_to_peak)
```

Out [37]:

	key2	data1	data2
key1			
a	1	2.740788	0.747409
b	1	1.697981	0.666660

In [38]:
```
grouped.describe()
```

Out [38]:

	key2								data1									
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%
key1																			
a	2.0	1.5	0.707107	1.0	1.25	1.5	1.75	2.0	3.0	-0.223603	...	0.673932	0.722116	3.0	-0.326566	0.39993	-0.782508	-0.472300	-0.16209
b	2.0	1.5	0.707107	1.0	1.25	1.5	1.75	2.0	2.0	0.445102	...	0.869597	1.294092	2.0	1.104837	0.47140	0.771507	0.938172	1.10483

2 rows × 24 columns

10.2.1 열에 여러 가지 함수 적용하기

In [39]:
```
tips=pd.read_csv("example/tips.csv")
tips.head()
```

Out [39]:

	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

```
dtype: int64
```

10.3.5 그룹 가중평균과 상관관계

In [79]:
```
df=pd.DataFrame({"category":["a","a","a","a",
                             "b","b","b","b"],
                 "data":np.random.standard_normal(8),
                 "weights":np.random.uniform(size=8)})

df
```

Out[79]:

	category	data	weights
0	a	0.301460	0.053101
1	a	0.370559	0.955775
2	a	1.027696	0.750453
3	a	-0.420586	0.670332
4	b	0.042830	0.208056
5	b	0.513502	0.945251
6	b	-2.056398	0.748753
7	b	-2.447351	0.706652

In [80]:
```
grouped=df.groupby("category")

def get_wavg(group):
    return np.average(group["data"],weights=group["weights"])

grouped.apply(get_wavg)
```

Out[80]:
```
category
a    0.353746
b   -1.063689
dtype: float64
```

In [81]:
```
close_px=pd.read_csv("example/stock_px.csv", parse_dates=True, index_col=0)

close_px.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2214 entries, 2003-01-02 to 2011-10-14
Data columns (total 4 columns):
#   Column  Non-Null Count  Dtype
```