D&A

Basic Session 6차시

판다스 기초2

2022 / 04 / 12

D&A 운영진 이예진

CONTENTS.

01. map, apply, applymap

임의의 함수 조작하기

method	인수	리턴값
map	Series	Series
apply	Series, Dataframe	Series, Dataframe
applymap	Dataframe	Dataframe

01. map

df.map(func)

- DataFrame의 특정 열에만 적용 가능 (Series)
- 1차원 배열(입력)의 값에 대해 임의의 함수를 적용하고, 1차원 배열(출력)로 리턴

```
# 사용할 함수 정의
def tenFunc(x):
    return x*10
```

데이터 프레임 적용시

```
abalone.map(tenFunc)
```

```
AttributeError: 'DataFrame' object has no attribute 'map'
```

시리즈 적용시

```
abalone['length'].map(tenFunc)
```

```
0       4.55
1       3.50
2       5.30
3       4.40
4       3.30
        ...
4172    5.65
4173    5.90
4174    6.00
4175    6.25
4176    7.10
Name: length, Length: 4177, dtype: float64
```

시리즈(row) 적용시

```
abalone.iloc[1].map(tenFunc)
```

```
sex               MMMMMMMMMM
length                 3.5
diameter              2.65
height                 0.9
whole_weight         2.255
shucked_weight       0.995
viscera_weight       0.485
shell_weight           0.7
rings                   70
Name: 1, dtype: object
```

01. apply

df.apply(func)

- 데이터프레임, 특정 열에 대해 적용

데이터 프레임 적용

```
abalone.apply(tenFunc)
```

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	MMMMMMMMM	4.55	3.65	0.95	5.140	2.245	1.010	1.500	150
1	MMMMMMMMM	3.50	2.65	0.90	2.255	0.995	0.485	0.700	70
2	FFFFFFFFF	5.30	4.20	1.35	6.770	2.565	1.415	2.100	90
3	MMMMMMMMM	4.40	3.65	1.25	5.160	2.155	1.140	1.550	100
4	IIIIIIIII	3.30	2.55	0.80	2.050	0.895	0.395	0.550	70
...	...	...	...	...	...	...	...	...	...
4172	FFFFFFFFF	5.65	4.50	1.65	8.870	3.700	2.390	2.490	110
4173	MMMMMMMMM	5.90	4.40	1.35	9.660	4.390	2.145	2.605	100
4174	MMMMMMMMM	6.00	4.75	2.05	11.760	5.255	2.875	3.080	90
4175	FFFFFFFFF	6.25	4.85	1.50	10.945	5.310	2.610	2.960	100
4176	MMMMMMMMM	7.10	5.55	1.95	19.485	9.455	3.765	4.950	120

4177 rows × 9 columns

시리즈(특정 열)적용시

```
abalone['length'].apply(tenFunc)
```

```
0       4.55
1       3.50
2       5.30
3       4.40
4       3.30
       ...
4172    5.65
4173    5.90
4174    6.00
4175    6.25
4176    7.10
Name: length, Le
```

특정 데이터 프레임 적용

```
abalone[['length']].apply(tenFunc)
```

	length
0	4.55
1	3.50
2	5.30
3	4.40
4	3.30
...	...
4172	5.65
4173	5.90
4174	6.00
4175	6.25
4176	7.10

4177 rows × 1 col

시리즈(row) 적용시

```
abalone.iloc[1].apply(tenFunc)
```

```
sex               MMMMMMMMMM
length                   3.5
diameter                2.65
height                   0.9
whole_weight           2.255
shucked_weight         0.995
viscera_weight         0.485
shell_weight             0.7
rings                     70
Name: 1, dtype: object
```

01. applymap

df.applymap(func)

- 데이터프레임에 대해 적용

데이터 프레임 적용

```
abalone.applymap(tenFunc)
```

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	MMMMMMMMMM	4.55	3.65	0.95	5.140	2.245	1.010	1.500	150
1	MMMMMMMMMM	3.50	2.65	0.90	2.255	0.995	0.485	0.700	70
2	FFFFFFFFFF	5.30	4.20	1.35	6.770	2.565	1.415	2.100	90
3	MMMMMMMMMM	4.40	3.65	1.25	5.160	2.155	1.140	1.550	100
4	IIIIIIIIII	3.30	2.55	0.80	2.050	0.895	0.395	0.550	70
...	...	...	...	...	...	...	...	...	...
4172	FFFFFFFFFF	5.65	4.50	1.65	8.870	3.700	2.390	2.490	110
4173	MMMMMMMMMM	5.90	4.40	1.35	9.660	4.390	2.145	2.605	100
4174	MMMMMMMMMM	6.00	4.75	2.05	11.760	5.255	2.875	3.080	90
4175	FFFFFFFFFF	6.25	4.85	1.50	10.945	5.310	2.610	2.960	100
4176	MMMMMMMMMM	7.10	5.55	1.95	19.485	9.455	3.765	4.950	120

4177 rows × 9 columns

특정 데이터 프레임 적용

```
abalone[['length']].applymap(tenFunc)
```

	length
0	4.55
1	3.50
2	5.30
3	4.40
4	3.30
...	...
4172	5.65
4173	5.90
4174	6.00
4175	6.25
4176	7.10

4177 rows × 1 col

시리즈(특정 열)적용시

```
abalone['length'].applymap(tenFunc)
```

```
AttributeError: 'Series' object has no attribute 'applymap'
```

시리즈(row) 적용시

```
abalone.iloc[1].applymap(tenFunc)
```

```
AttributeError: 'Series' object has no attribute 'applymap'
```

02. Groupby

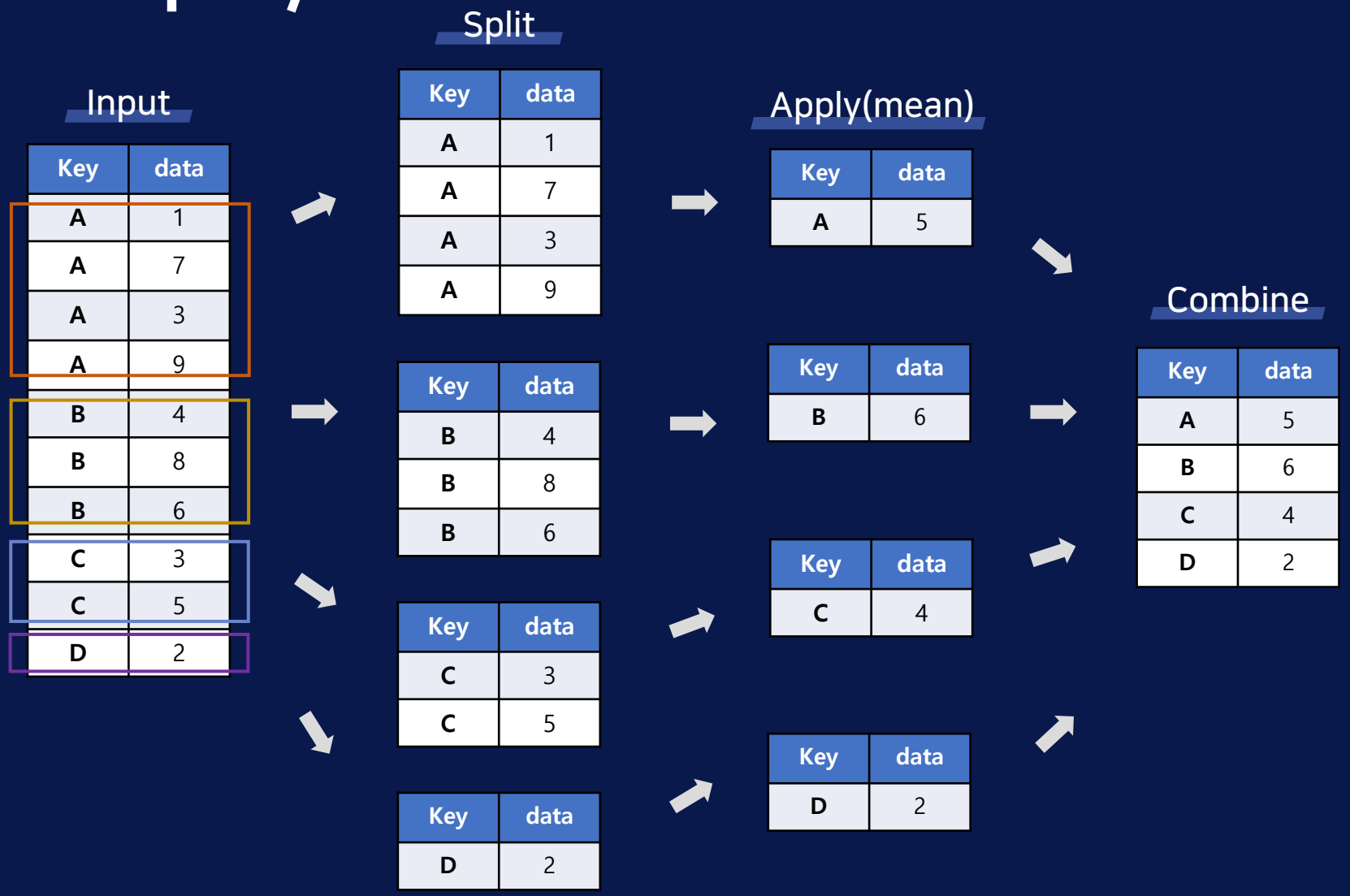
Groupby란?

집단, 그룹별로 데이터를 집계, 요약하는 방법

Key	data
A	1
A	7
A	3
A	9
B	4
B	8
B	6
C	3
C	5
D	2

Key	data
A	5
B	6
C	4
D	2

02. Groupby


Input
Split
Apply(mean)
Combine
Key
data
A 1
A 7
A 3
A 9
B 4
B 8
B 6
C 3
C 5
D 2
A 5
B 6
C 4
D 2

02. Groupby

groupby() - split

```
abalone['sex'].value_counts()
```

```
M    1528
I    1342
F    1307
Name: sex, dtype: int64
```

```
for key,group in abalone.iloc[:,:5].groupby(abalone['sex']):
    print('key: ', key)
    print('number: ', len(group))
    print(group)
```

```
key:  F
number:  1307
      sex  length  diameter  height  whole_weight
2       F   0.530     0.420   0.135        0.6770
6       F   0.530     0.415   0.150        0.7775
7       F   0.545     0.425   0.125        0.7680
9       F   0.550     0.440   0.150        0.8945
10      F   0.525     0.380   0.140        0.6065
...   ..     ...       ...     ...           ...
4160    F   0.585     0.475   0.165        1.0530
4161    F   0.585     0.455   0.170        0.9945
4168    F   0.515     0.400   0.125        0.6150
4172    F   0.565     0.450   0.165        0.8870
4175    F   0.625     0.485   0.150        1.0945

[1307 rows x 5 columns]
```

```
key:  I
number:  1342
      sex  length  diameter  height  whole_weight
4       I   0.330     0.255   0.080        0.2050
5       I   0.425     0.300   0.095        0.3515
16      I   0.355     0.280   0.085        0.2905
21      I   0.380     0.275   0.100        0.2255
42      I   0.240     0.175   0.045        0.0700
...   ..     ...       ...     ...           ...
4158    I   0.480     0.355   0.110        0.4495
4163    I   0.390     0.310   0.085        0.3440
4164    I   0.390     0.290   0.100        0.2845
4165    I   0.405     0.300   0.085        0.3035
4166    I   0.475     0.365   0.115        0.4990

[1342 rows x 5 columns]
```

```
key:  M
number:  1528
      sex  length  diameter  height  whole_weight
0       M   0.455     0.365   0.095        0.5140
1       M   0.350     0.265   0.090        0.2255
3       M   0.440     0.365   0.125        0.5160
8       M   0.475     0.370   0.125        0.5095
11      M   0.430     0.350   0.110        0.4060
...   ..     ...       ...     ...           ...
4170    M   0.550     0.430   0.130        0.8395
4171    M   0.560     0.430   0.155        0.8675
4173    M   0.590     0.440   0.135        0.9660
4174    M   0.600     0.475   0.205        1.1760
4176    M   0.710     0.555   0.195        1.9485

[1528 rows x 5 columns]
```

02. Groupby

groupby() - split

get_group()

그룹 안에 데이터를 확인하고 싶은 경우에 사용

```
abalone.groupby('sex').get_group('F')
```

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
6	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.3300	20
7	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.2600	16
9	F	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.3200	19
10	F	0.525	0.380	0.140	0.6065	0.1940	0.1475	0.2100	14
...	...	...	...	...	...	...	...	...	...
4160	F	0.585	0.475	0.165	1.0530	0.4580	0.2170	0.3000	11
4161	F	0.585	0.455	0.170	0.9945	0.4255	0.2630	0.2845	11
4168	F	0.515	0.400	0.125	0.6150	0.2865	0.1230	0.1765	8
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10

1307 rows × 9 columns

02. Groupby

groupby () - apply, combine

- df.groupby('key')['column'].집계함수

- df.groupby('key'). filter(조건식 함수)

Aggregation
  - 그룹별로 결과를 얻는 조작
  - Aggregation을 사용하고 싶은 경우 agg()를 사용해 처리 가능

- df.groupby('key')['column'].agg(매핑함수)
    - 집계함수
    - 사용자 지정함수, lambda
    - 한 column에 여러함수를 동시에 사용
    - 여러 column에 여러함수를 동시에 사용
    - 여러 column에 원하는 함수를 사용

- df.groupby('key')['column'].apply(매핑함수)

02. Groupby

df.groupby('key')['column'].집계함수

집계함수 : sum. Mean, count, max, min, size 등

	key	data01	data02
0	A	1	4
1	B	2	4
2	C	3	5
3	A	2	0
4	A	3	6
5	B	4	1

df.groupby('key').집계함수

```
# key값으로 묶어서 합sum
df.groupby('key').sum()
```

key	data01	data02
A	6	10
B	6	5
C	3	5

```
# key, data01 값으로 묶어서 합 sum
df.groupby(['key','data01']).sum()
```

		data02
key	data01	
A	1	4
	2	0
	3	6
B	2	4
	4	1
C	3	5

df.groupby('key')['column'].집계함수

```
df.groupby('key')['data01'].sum()
```

```
key
A    6
B    6
C    3
Name: data01, dtype: int64
```

```
df.groupby('key')[['data01']].sum()
```

	data01
key	
A	6
B	6
C	3

02. Groupby

df.groupby(‘key’).filter(조건식함수)

그룹의 속성을 기준으로 데이터를 필터링할 때 사용

	key	data01	data02
0	A	1	4
1	B	2	4
2	C	3	5
3	A	2	0
4	A	3	6
5	B	4	1

```
df.groupby('key').filter(lambda x : x['data01'].mean()>2)
```

	key	data01	data02
1	B	2	4
2	C	3	5
5	B	4	1

```
# key값으로 묶어서 평균 mean
df.groupby('key').mean()
```

key	data01	data02
A	2.0	3.333333
B	3.0	2.500000
C	3.0	5.000000

02. Groupby

df.groupby(‘key’)[‘column’].agg(매핑함수)

여러 개의 통계함수를 적용시키고 싶을 때 사용

df.groupby(‘key’)[‘column’].aggregate(매핑함수)

	key	data01	data02
0	A	1	4
1	B	2	4
2	C	3	5
3	A	2	0
4	A	3	6
5	B	4	1

```
df.groupby('key')['data01'].agg([min, np.median, max])
```

key	min	median	max
A	1	2	3
B	2	3	4
C	3	3	3

```
df.groupby('key')[['data01','data02']].agg([min, np.median, max])
```

	data01			data02		
	min	median	max	min	median	max
key						
A	1	2	3	0	4.0	6
B	2	3	4	1	2.5	4
C	3	3	3	5	5.0	5

```
df.groupby('key').agg(['min', np.median, 'max'])
df.groupby('key').agg([min, np.median, max])
```

	data01			data02		
	min	median	max	min	median	max
key						
A	1	2	3	0	4.0	6
B	2	3	4	1	2.5	4
C	3	3	3	5	5.0	5

02. Groupby

df.groupby('key')['column'].agg(매핑함수)

여러 개의 통계함수를 적용시키고 싶을 때 사용

df.groupby('key')['column'].aggregate(매핑함수)

```
df.groupby('key').agg({'data01':'min',
                       'data02':np.sum})
```

key	data01	data02
A	1	10
B	2	5
C	3	5

```
df.groupby('key')[['data01','data02']].agg(['min','mean'])
```

	data01		data02	
	min	mean	min	mean
key				
A	1	2	0	3.333333
B	2	3	1	2.500000
C	3	3	5	5.000000

```
df.groupby('key')[['data01','data02']].agg([('최소','min'), ('평균','mean')])
```

	data01		data02	
	최소	평균	최소	평균
key				
A	1	2	0	3.333333
B	2	3	1	2.500000
C	3	3	5	5.000000

```
df.groupby('key').agg({'data01':[('최소','min')],
                       'data02':[('평균','mean')]})
```

	data01	data02
	최소	평균
key		
A	1	3.333333
B	2	2.500000
C	3	5.000000

02. Groupby

df.groupby('key')['column'].apply(매핑함수)

apply

```
df.groupby('key')['data01'].apply(min)
```

```
key
A    1
B    2
C    3
Name: data01, dtype: int64
```

```
df.groupby('key').apply(min)
```

	key	data01	data02
key			
A	A	1	0
B	B	2	1
C	C	3	5

```
df.groupby('key').apply(lambda x: x.data01.min())
```

```
key
A    1
B    2
C    3
dtype: int64
```

agg

```
df.groupby('key')['data01'].agg(min)
```

```
key
A    1
B    2
C    3
Name: data01, dtype: int64
```

```
df.groupby('key').agg(min)
```

	data01	data02
key		
A	1	0
B	2	1
C	3	5

```
df.groupby('key').agg(lambda x: x.data01.min())
```

	data01	data02
key		
A	1	1
B	2	2
C	3	3

03. Pivot table

pivot, 피벗

- 데이터 테이블 재배치, 구조 변경
- 여러 column을 index, values, columns 값으로 사용 가능
- Group연산, 테이블 요약, 그래프 등을 위해 사용

df.pivot(index=None, columns=None, values=None)

index : index 로 사용될 컬럼
columns : column 으로 사용될 컬럼
values : value에 채우고자 하는 컬럼

	A	B	C	D	E
0	2000	3	20	3	1
1	1988	2	36	4	2
2	1991	5	82	5	1
3	2002	10	1	0	2
4	1998	12	68	6	2
5	1996	7	26	1	1

```
df.pivot('A','B')
```

	C						D						E					
B	2	3	5	7	10	12	2	3	5	7	10	12	2	3	5	7	10	12
A																		
1988	36.0	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN
1991	NaN	NaN	82.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN
1996	NaN	NaN	NaN	26.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN
1998	NaN	NaN	NaN	NaN	NaN	68.0	NaN	NaN	NaN	NaN	NaN	6.0	NaN	NaN	NaN	NaN	NaN	2.0
2000	NaN	20.0	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
2002	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	2.0	NaN

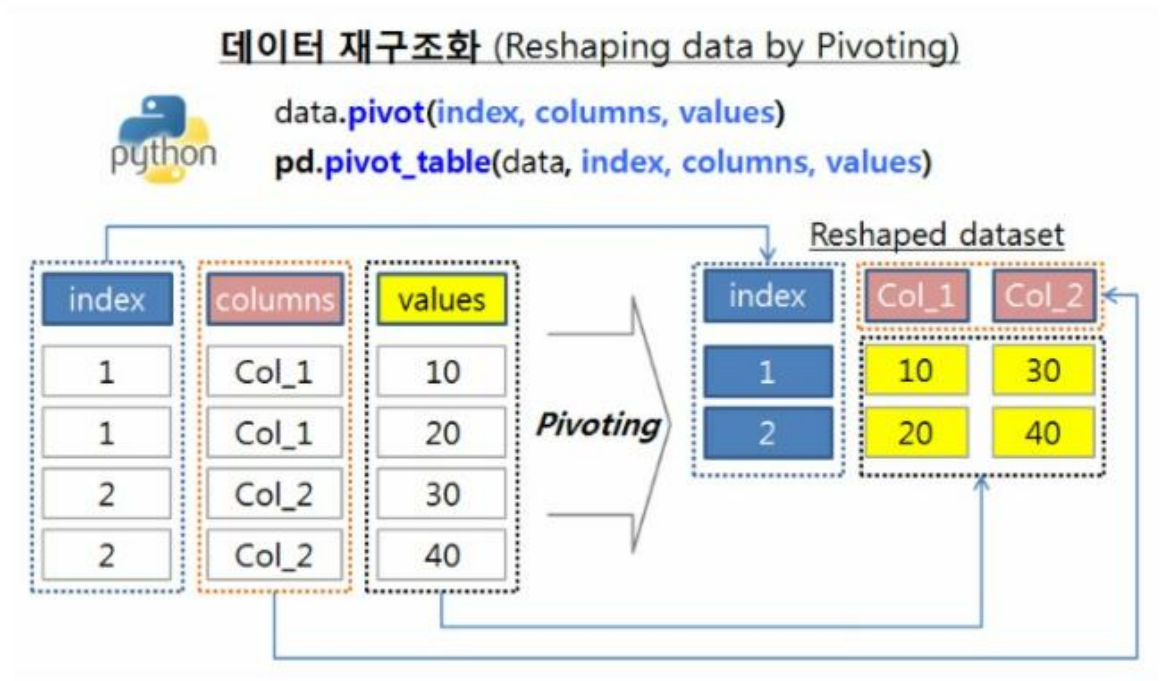
```
df.pivot('A','B','C')
```

B	2	3	5	7	10	12
A						
1988	36.0	NaN	NaN	NaN	NaN	NaN
1991	NaN	NaN	82.0	NaN	NaN	NaN
1996	NaN	NaN	NaN	26.0	NaN	NaN
1998	NaN	NaN	NaN	NaN	NaN	68.0
2000	NaN	20.0	NaN	NaN	NaN	NaN
2002	NaN	NaN	NaN	NaN	1.0	NaN

03. Pivot table

pivot_table, 피벗 테이블

- 데이터의 요약된 정보를 출력




```
pd.pivot_table(df,
               values=None ,
               index=None,
               columns=None,
               aggfunc='mean',
               fill_value=None,
               margins=False,
               dropna=True,
               margins_name='All')
```

df : 데이터프레임
values : 분석할 열
index : index 로 사용될 컬럼
columns : column 으로 사용될 컬럼
aggfunc : 집계함수(데이터 축약 시 사용할 함수)
fill_value : NaN 대체 값
margins : 분석 결과를 오른쪽과 아래에 붙일지 여부(행/열 별 총합)
margins_name : 마진 열(행)의 이름

03. Pivot table

pivot_table, 피벗 테이블

- 데이터의 요약된 정보를 출력

	A	B	C	D	E
0	2000	3	20	3	1
1	1988	12	1	3	2
2	2000	12	20	5	1
3	2000	3	1	0	2
4	1998	12	20	5	2
5	1996	3	1	0	1
6	1996	12	1	0	1

```
pd.pivot_table(df,
               values='C',
               index='A',
               columns='B',
               aggfunc='mean',
               fill_value=None,
               margins=False,
               dropna=True,
               margins_name='All')
```

B	3	12
A		
1988	NaN	1.0
1996	1.0	1.0
1998	NaN	20.0
2000	10.5	20.0

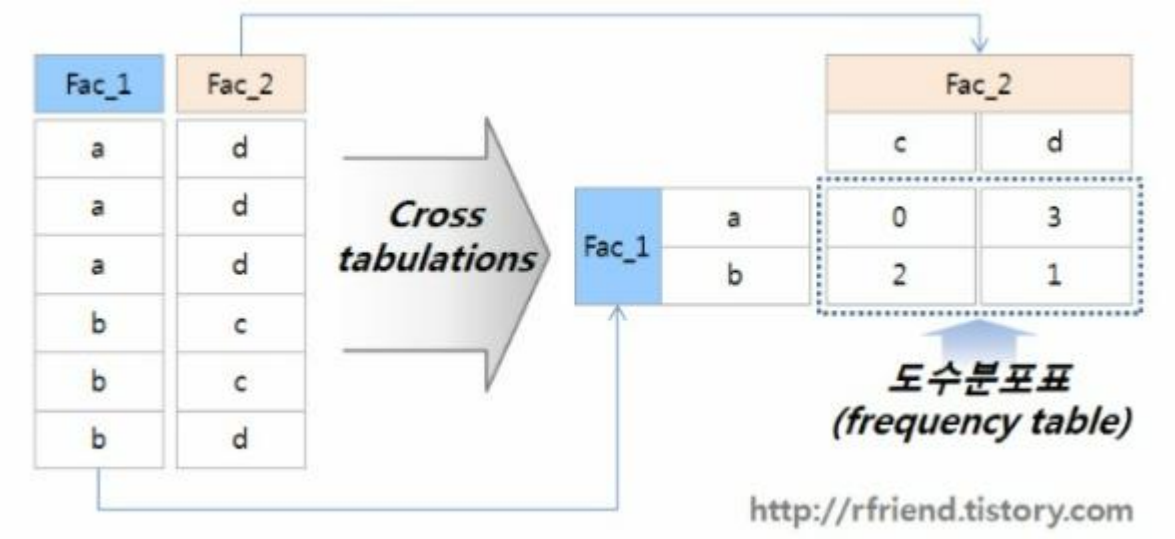
```
pd.pivot_table(df,
               values='C',
               index='A',
               columns='B',
               aggfunc='mean',
               fill_value=0,
               margins=True,
               dropna=True,
               margins_name='All_mean')
```

B	3	12	All_mean
A			
1988	0.000000	1.0	1.000000
1996	1.000000	1.0	1.000000
1998	0.000000	20.0	20.000000
2000	10.500000	20.0	13.666667
All_mean	7.333333	10.5	9.142857

04. Crosstab

Crosstab

범주형 변수를 기준으로 개수 파악이나 수치형 데이터를 넣어 계산할 때 사용
pivot_table의 일종으로, pivot_table의 특수한 형태
두 column의 교차 빈도, 비율, 덧셈 등을 구할 때 주로 사용




```
pd.crosstab(index,
            columns,
            values,
            aggfunc,
            rownames,
            colnames,
            margins,
            normalize)
```

04. Crosstab

Crosstab

	ID	hour	A	B	C	score
0	ID01	2	139	country	no	3
1	ID02	3	148	country	no	5
2	ID03	3	149	country	no	7
3	ID04	5	151	country	no	10
4	ID05	7	154	city	no	12
5	ID06	2	149	country	no	7
6	ID07	8	155	city	yes	13
7	ID08	9	155	city	yes	13
8	ID09	6	154	city	no	12
9	ID10	9	156	city	yes	13
10	ID11	6	153	city	no	12
11	ID12	2	151	country	no	6

```
pd.crosstab(index=df['B'],
            columns=df['C'])
```

C	no	yes
B		
city	3	3
country	6	0

```
pd.crosstab(values=df['score'],
            index=df['B'],
            columns=df['C'],
            aggfunc='mean')
```

C	no	yes
B		
city	12.000000	13.0
country	6.333333	NaN

groupby + pivot = pivot_table

crosstab ⊂ pivot_table

	사용이유
groupby	특정 컬럼을 기준으로 그룹화하여 테이블에 존재하는 행들을 그룹별로 구분하기 위해
pivot_table	다양한 요소들을 활용해 데이터를 빠르게 분석하기 위해
crosstab	두 칼럼에 교차 빈도, 비율, 덧셈 등을 구할 때 사용

첨부자료 출처

03. pivot_table

피벗테이블 이미지 출처
: https://rfriend.tistory.com/275

04. crosstab

크로스탭 이미지 출처
: https://rfriend.tistory.com/280

D&A

Basic Session 6차시 판다스 기초2

Thank You.

2022 / 04 / 12

D&A 운영진 이예진