

# 그룹 연산

- 데이터 집계
- 데이터 변환
- 데이터 필터링
- 그룹 오브젝트

## ■ 데이터 집계

- groupby 메소드로 평균값 구하기
  - 갭마인더 데이터 집합 가져오기

```
import pandas as pd
df = pd.read_csv('data/gapminder.tsv', sep='\t')
```

- year 열을 기준으로 그룹화한 다음 lifeExp 열의 평균 구하기

```
avg_life_exp_by_year = df.groupby('year')['lifeExp'].mean()
print(avg_life_exp_by_year)
```

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

## ■ 데이터 집계

### ● groupby 메소드의 분할/반영/결합 과정 확인하기

#### - 분할 작업

```
years = df['year'].unique()  
print(years)
```

```
[1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007]
```

#### - 반영 작업

```
year_means = []  
for y in years:  
    m = df.loc[df.year == y, :].lifeExp.mean()  
    year_means.append((y, m))  
print(year_means)
```

```
[(1952, 49.057619718309866), (1957, 51.50740112676056), (1962, 53.609249014084504), (1967,  
55.678289577464795), (1972, 57.64738647887324), (1977, 59.57015746478874), (1982, 61.533197  
18309859), (1987, 63.21261267605633), (1992, 64.16033802816901), (1997, 65.01467605633802),  
(2002, 65.69492253521126), (2007, 67.00742253521126)]
```

## ■ 데이터 집계

- groupby 메소드의 분할/반영/결합 과정 확인하기
  - 결합 작업

```
df2 = pd.DataFrame({  
    'year' : [ y for y, m in year_means ],  
    '' : [ m for y, m in year_means ]  
})  
df2
```

	year	
0	1952	49.057620
1	1957	51.507401
2	1962	53.609249
3	1967	55.678290
4	1972	57.647386
5	1977	59.570157
6	1982	61.533197
7	1987	63.212613
8	1992	64.160338
9	1997	65.014676
10	2002	65.694923
11	2007	67.007423

## ■ 데이터 집계

### ● groupby 메소드와 함께 사용하는 집계 메소드

메소드	설명
count	누락값을 제외한 데이터 수를 반환
size	누락값을 포함한 데이터 수를 반환
mean	평균값 반환
std	표준편차 반환
min	최소값 반환
quantile(q=0.25 / 0.50 / 0.75)	백분위수 25% / 50% / 75%
max	최대값 반환
sum	전체 합 반환
var	분산 반환
sem	평균의 표준편차 반환
describe	데이터 수, 평균, 표준편차, 최소값, 백분위수, 최대값 반환
first	첫번째 행 반환
last	마지막 행 반환
nth	n번째 행 반환

## ■ 데이터 집계

### ● 사용자 함수 적용 agg() : apply()와 유사

- 입력받은 열의 평균값을 구하는 사용자 함수

```
def my_mean(values):  
    n = len(values)  
    sum = 0  
    for value in values:  
        sum += value  
    return sum / n
```

- agg() 적용

```
agg_my_mean = df.groupby('year').lifeExp.agg(my_mean)  
print(agg_my_mean)
```

```
year  
1952    49.057620  
1957    51.507401  
1962    53.609249  
1967    55.678290  
1972    57.647386  
1977    59.570157  
1982    61.533197  
1987    63.212613  
1992    64.160338  
1997    65.014676  
2002    65.694923  
2007    67.007423  
Name: lifeExp, dtype: float64
```

mean 메소드와 동일한 결과 값

## ■ 데이터 집계

### ● 2개의 인자 사용

- 연도별 평균 수명에서 전체 평균 수명 차이를 구하는 사용자 함수

```
def my_mean_diff(values, diff_value):  
    n = len(values)  
    sum = 0  
    for value in values:  
        sum += value  
    mean = sum / n  
    return mean - diff_value
```

- 전체 평균 수명

```
global_mean = df.lifeExp.mean()  
print(global_mean)
```

59.474439366197174

## ■ 데이터 집계

### ● 2개의 인자 사용

- agg() 적용

```
agg_mean_diff = df.groupby('year').lifeExp.agg(  
    my_mean_diff, diff_value=global_mean)  
print(agg_mean_diff)
```

```
year  
1952    -10.416820  
1957     -7.967038  
1962     -5.865190  
1967     -3.796150  
1972     -1.827053  
1977      0.095718  
1982      2.058758  
1987      3.738173  
1992      4.685899  
1997      5.540237  
2002      6.220483  
2007      7.532983  
Name: lifeExp, dtype: float64
```



## ■ 데이터 집계

### ● 여러 개의 집계 메소드 한번에 적용하기 - 리스트

- np.count\_nonzero / np.mean / np.std 적용

```
import numpy as np
gdf = df.groupby('year').lifeExp.agg([np.count_nonzero, np.mean, np.std])
print(gdf)
```

	count_nonzero	mean	std
year			
1952	142.0	49.057620	12.225956
1957	142.0	51.507401	12.231286
1962	142.0	53.609249	12.097245
1967	142.0	55.678290	11.718858
1972	142.0	57.647386	11.381953
1977	142.0	59.570157	11.227229
1982	142.0	61.533197	10.770618
1987	142.0	63.212613	10.556285
1992	142.0	64.160338	11.227380
1997	142.0	65.014676	11.559439
2002	142.0	65.694923	12.279823
2007	142.0	67.007423	12.073021

## ■ 데이터 집계

- 여러 개의 집계 메소드를 여러 개의 열에 적용하기 - 딕셔너리
  - np.count\_nonzero / np.mean / np.std 적용

```
gdf_dict = df.groupby('year').agg(  
    {'lifeExp': 'mean', 'pop': 'median', 'gdpPercap': 'median'}  
)  
print(gdf_dict)
```

	lifeExp	pop	gdpPercap
year			
1952	49.057620	3943953.0	1968.528344
1957	51.507401	4282942.0	2173.220291
1962	53.609249	4686039.5	2335.439533
1967	55.678290	5170175.5	2678.334741
1972	57.647386	5877996.5	3339.129407
1977	59.570157	6404036.5	3798.609244
1982	61.533197	7007320.0	4216.228428
1987	63.212613	7774861.5	4280.300366
1992	64.160338	8688686.5	4386.085502
1997	65.014676	9735063.5	4781.825478
2002	65.694923	10372918.5	5319.804524
2007	67.007423	10517531.0	6124.371109

## ■ 데이터 변환

### ● 표준점수 계산하기

- 데이터의 평균과 표준편차의 차이를 표준점수라고 부름
- 표준점수를 구하면 데이터의 평균값은 0이 되고 표준편차는 1이 됨
- 통계에서 자주 사용되는 지표이며, Z점수 / z-점수 등의 표현 사용
- 표준점수를 계산하는 사용자 함수

```
def my_zscore(x):  
    return (x - x.mean()) / x.std()
```

- 각 연도별 lifeExp 열의 표준점수 계산

```
trans_z = df.groupby('year').lifeExp.transform(my_zscore)  
print(trans_z.head())
```

```
0    -1.656854  
1    -1.731249  
2    -1.786543  
3    -1.848157  
4    -1.894173  
Name: lifeExp, dtype: float64
```

## ■ 데이터 변환

### ● 누락값을 평균값으로 처리하기

- tips 데이터 집합에서 10개의 데이터 가져오기

```
import seaborn as sns
import numpy as np

np.random.seed(1234)
tips_10 = sns.load_dataset('tips').sample(10)
tips_10
```

	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
31	18.35	2.50	Male	No	Sat	Dinner	4
52	34.81	5.20	Female	No	Sun	Dinner	4
71	17.07	3.00	Female	No	Sat	Dinner	3
6	8.77	2.00	Male	No	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4
131	20.27	2.83	Female	No	Thur	Lunch	2
157	25.00	3.75	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4

## ■ 데이터 변환

### ● 누락값을 평균값으로 처리하기

- total\_bill 열의 값 4개를 임의의 선택하여 누락값으로 바꾸기

```
tips_10.loc[np.random.permutation(tips_10.index)[:4], 'total_bill'] = np.NaN  
tips_10
```

	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
31	NaN	2.50	Male	No	Sat	Dinner	4
52	NaN	5.20	Female	No	Sun	Dinner	4
71	NaN	3.00	Female	No	Sat	Dinner	3
6	8.77	2.00	Male	No	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4
131	20.27	2.83	Female	No	Thur	Lunch	2
157	25.00	3.75	Female	No	Sun	Dinner	4
5	NaN	4.71	Male	No	Sun	Dinner	4

## ■ 데이터 변환

### ● 누락값을 평균값으로 처리하기

- 흡연자와 비흡연자 평균을 따로 계산

```
def fill_na_mean(x):  
    avg = x.mean()  
    return x.fillna(avg)
```

```
tb_grp_mean = tips_10.groupby('smoker').total_bill.transform(fill_na_mean)  
tips_10['fill_total_bill'] = tb_grp_mean  
tips_10
```

	total_bill	tip	sex	smoker	day	time	size	fill_total_bill
187	30.46	2.00	Male	Yes	Sun	Dinner	5	30.460000
179	34.63	3.55	Male	Yes	Sun	Dinner	2	34.630000
31	NaN	2.50	Male	No	Sat	Dinner	4	18.013333
52	NaN	5.20	Female	No	Sun	Dinner	4	18.013333
71	NaN	3.00	Female	No	Sat	Dinner	3	18.013333
6	8.77	2.00	Male	No	Sun	Dinner	2	8.770000
95	40.17	4.73	Male	Yes	Fri	Dinner	4	40.170000
131	20.27	2.83	Female	No	Thur	Lunch	2	20.270000
157	25.00	3.75	Female	No	Sun	Dinner	4	25.000000
5	NaN	4.71	Male	No	Sun	Dinner	4	18.013333

## ■ 데이터 변환

### ● apply / agg / transform

#### 1. 숫자로만 구성된 데이터프레임

```
def plus(value):  
    return value + 2
```

```
import pandas as pd  
df = pd.DataFrame({'a': [10, 20, 30], 'b': [20, 30, 40]})
```

	a	b
0	10	20
1	20	30
2	30	40

```
df.apply(plus)  
df.agg(plus)  
df.transform(plus)
```

	a	b
0	12	22
1	22	32
2	32	42

```
df.a.apply(plus)  
df.a.agg(plus)  
df.a.transform(plus)
```

0	12
1	22
2	32

Name: a, dtype: int64

apply / agg / transform 모두 가능하고 차이 없음

## ■ 데이터 변환

### ● apply / agg / transform

#### 2. 적용시킬 사용자 함수 내에서 집계 함수 사용

```
def mean(value):  
    return value.min()
```

```
df.apply(mean)  
df.agg(mean)
```

```
a    10  
b    20  
dtype: int64
```

```
df.a.agg(mean)
```

```
10
```

```
df.a.apply(mean)
```

**AttributeError:** 'int' object has no attribute 'min'

```
df.transform(mean)  
df.a.transform(mean)
```

**ValueError:** transforms cannot produce aggregated results

apply : 데이터프레임 가능  
agg : 데이터프레임 / 시리즈 가능  
transform : 모두 불가



## ■ 데이터 변환

### ● apply / agg / transform

#### 3. 숫자 데이터 열과 문자 데이터 열이 함께 있는 데이터프레임

```
df2 = pd.DataFrame({  
    'a': [10, 20, 30, 40],  
    'b': [20, 30, 40, 50],  
    'c': ['서울', '대전', '부산', '제주'],  
    'd': ['Female', 'Male', 'Male', 'Female']  
})
```

	a	b	c	d
0	10	20	서울	Female
1	20	30	대전	Male
2	30	40	부산	Male
3	40	50	제주	Female

```
df2.apply(mean)  
df2.agg(mean)  
df2.transform(mean)
```

```
df2.apply(plus)  
df2.agg(plus)  
df2.transform(plus)
```

apply / agg / transform 모두 가능하지만  
TypeError 와 같이 숫자 + 문자 연산을 수행하는 경우에는 불가

**TypeError:** can only concatenate str (not "int") to str

## ■ 데이터 변환

### ● apply / agg / transform

#### 4. 그룹화한 데이터

```
df2_grp = df2.groupby('c')
```

```
df2_grp.transform(plus)
```

	a	b
0	12	22
1	22	32
2	32	42
3	42	52

```
df2_grp.apply(plus)  
df2_grp.agg(plus)
```

**TypeError:** can only concatenate str (not "int") to str

transform은 연산이 가능한 열 데이터만 필터링하여 연산  
기존 인덱스를 유지하면서 연산

```
df2_grp.apply(mean)  
df2_grp.agg(mean)
```

	a	b	c
d			
Female	10	20	서울
Male	20	30	대전

```
df2_grp.transform(mean)
```

	a	b	c
0	10	20	서울
1	20	30	대전
2	20	30	대전
3	10	20	서울

## ■ 데이터 필터링

### ● 그룹화한 데이터에서 원하는 데이터를 걸러내기

- tips 데이터 집합 불러오기

```
import seaborn as sns
tips = sns.load_dataset('tips')
print(tips.shape)
```

(244, 7)

- size 열의 데이터 빈도 확인

```
print(tips['size'].value_counts())
```

2	156
3	38
4	37
5	5
6	4
1	4

Name: size, dtype: int64

## ■ 데이터 필터링

- 그룹화한 데이터에서 원하는 데이터를 걸러내기
  - 주문횟수가 30번 이상인 데이터만 그룹화

```
tips_filtered = tips.groupby('size').filter(  
    lambda x: x['size'].count() >= 30)  
print(tips_filtered.shape)
```

(231, 7)

- 필터링 후 size 열의 데이터 빈도 확인

```
print(tips_filtered['size'].value_counts())
```

2      156

3      38

4      37

Name: size, dtype: int64

## ■ 그룹 오브젝트

- groupby 메소드가 반환하는 값인 그룹 오브젝트 확인
  - tips 데이터 집합에서 임의의 10개 데이터 추출

```
# import numpy as np
# np.random.seed(1234)
# tips_10 = sns.load_dataset('tips').sample(10)
# print(tips_10)

tips_10 = sns.load_dataset('tips').sample(
    10, random_state=1234)
print(tips_10)
```

	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
31	18.35	2.50	Male	No	Sat	Dinner	4
52	34.81	5.20	Female	No	Sun	Dinner	4
71	17.07	3.00	Female	No	Sat	Dinner	3
6	8.77	2.00	Male	No	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4
131	20.27	2.83	Female	No	Thur	Lunch	2
157	25.00	3.75	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4

## ■ 그룹 오브젝트

- groupby 메소드가 반환하는 값인 그룹 오브젝트 확인
  - 흡연자 그룹화

```
grouped = tips_10.groupby('smoker')  
print(grouped)
```

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000020514B68BE0>

- 그룹 오브젝트에 포함된 그룹 확인 (그룹화 된 인덱스)

```
print(grouped.groups)
```

{'Yes': [187, 179, 95], 'No': [31, 52, 71, 6, 131, 157, 5]}

- 그룹 오브젝트의 평균 구하기

```
avgs = grouped.mean()  
print(avgs)
```

	total_bill	tip	size
smoker			
Yes	35.086667	3.426667	3.666667
No	21.365714	3.427143	3.285714

그룹 오브젝트에는 성별, 요일, 시간 열과 같이 평균값을 구할 수 없는 열도 포함되지만  
계산할 수 있는 열만 골라주는 기능이 제공되므로 바로 평균값 추출 가능

## ■ 그룹 오브젝트

### ● 그룹 오브젝트에서 데이터 추출하고 반복하기

- 그룹명을 지정하여 흡연자 데이터만 추출

```
yes = grouped.get_group('Yes')
print(yes)
```

	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4

- 반복문으로 그룹 오브젝트의 각 그룹 정보 추출

```
for smoke_group in grouped:
    print(smoke_group)
```

('Yes',	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4)
('No',	total_bill	tip	sex	smoker	day	time	size
31	18.35	2.50	Male	No	Sat	Dinner	4
52	34.81	5.20	Female	No	Sun	Dinner	4
71	17.07	3.00	Female	No	Sat	Dinner	3
6	8.77	2.00	Male	No	Sun	Dinner	2
131	20.27	2.83	Female	No	Thur	Lunch	2
157	25.00	3.75	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4)

튜플로 구성  
( 그룹명, 데이터프레임 )

## ■ 그룹 오브젝트

### ● 그룹 오브젝트에서 데이터 추출하고 반복하기

- 각 그룹을 데이터프레임 형태로 확인하기

```
for smoke_group in grouped:  
    print('그룹명:', smoke_group[0])  
    print(smoke_group[1])
```

그룹명: Yes

	total_bill	tip	sex	smoker	day	time	size
187	30.46	2.00	Male	Yes	Sun	Dinner	5
179	34.63	3.55	Male	Yes	Sun	Dinner	2
95	40.17	4.73	Male	Yes	Fri	Dinner	4

그룹명: No

	total_bill	tip	sex	smoker	day	time	size
31	18.35	2.50	Male	No	Sat	Dinner	4
52	34.81	5.20	Female	No	Sun	Dinner	4
71	17.07	3.00	Female	No	Sat	Dinner	3
6	8.77	2.00	Male	No	Sun	Dinner	2
131	20.27	2.83	Female	No	Thur	Lunch	2
157	25.00	3.75	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4



## ■ 그룹 오브젝트

### ● 여러 열을 사용해 그룹 오브젝트 만들고 계산하기

- smoker, day 열을 기준으로 그룹화

```
smoker_day = tips_10.groupby(['smoker', 'day'])
group_avg = smoker_day.mean()
group_avg
```

		total_bill	tip	size
smoker	day			
Yes	Thur	NaN	NaN	NaN
	Fri	40.1700	4.730	4.0
	Sat	NaN	NaN	NaN
	Sun	32.5450	2.775	3.5
No	Thur	20.2700	2.830	2.0
	Fri	NaN	NaN	NaN
	Sat	17.7100	2.750	3.5
	Sun	23.4675	3.915	3.5

## ■ 그룹 오브젝트

### ● 여러 열을 사용해 그룹 오브젝트 만들고 계산하기

- 그룹 오브젝트 자료형 및 열 이름 확인

```
print(type(group_avg))  
print(group_avg.columns)
```

```
<class 'pandas.core.frame.DataFrame'>  
Index(['total_bill', 'tip', 'size'], dtype='object')
```

- 인덱스 확인

```
print(group_avg.index)
```

```
MultiIndex([( 'Yes', 'Thur'),  
            ( 'Yes', 'Fri'),  
            ( 'Yes', 'Sat'),  
            ( 'Yes', 'Sun'),  
            ( 'No', 'Thur'),  
            ( 'No', 'Fri'),  
            ( 'No', 'Sat'),  
            ( 'No', 'Sun')],  
           names=['smoker', 'day'])
```

**MultiIndex : 여러 개 인덱스로 구성되어 있는 것**

## ■ 그룹 오브젝트

- 여러 열을 사용해 그룹 오브젝트 만들고 계산하기
  - 멀티 인덱스에서 값 조회

```
group_avg.loc['Yes']
```

	total_bill	tip	size
day			
Thur	NaN	NaN	NaN
Fri	40.170	4.730	4.0
Sat	NaN	NaN	NaN
Sun	32.545	2.775	3.5

```
group_avg.loc['Yes', 'Fri']
```

```
total_bill    40.17
tip           4.73
size          4.00
Name: (Yes, Fri), dtype: float64
```

```
group_avg.loc[('Yes', 'Fri'), :]
```

```
total_bill    40.17
tip           4.73
size          4.00
Name: (Yes, Fri), dtype: float64
```

```
group_avg.loc['Yes', 'Fri', :]
```

		total_bill	tip	size
smoker	day			
Yes	Fri	40.17	4.73	4.0

## ■ 그룹 오브젝트

### ● 여러 열을 사용해 그룹 오브젝트 만들고 계산하기

- 인덱스 초기화 : `reset_index` 메소드

```
group_method = tips_10.groupby(['smoker', 'day']).mean().reset_index()  
group_method
```

- 인덱스 초기화 : `as_index` 인자

```
group_param = tips_10.groupby(['smoker', 'day'], as_index=False).mean()  
group_param
```

		total_bill	tip	size
smoker	day			
Yes	Thur	NaN	NaN	NaN
	Fri	40.1700	4.730	4.0
	Sat	NaN	NaN	NaN
	Sun	32.5450	2.775	3.5
No	Thur	20.2700	2.830	2.0
	Fri	NaN	NaN	NaN
	Sat	17.7100	2.750	3.5
	Sun	23.4675	3.915	3.5



	smoker	day	total_bill	tip	size
0	Yes	Thur	NaN	NaN	NaN
1	Yes	Fri	40.1700	4.730	4.0
2	Yes	Sat	NaN	NaN	NaN
3	Yes	Sun	32.5450	2.775	3.5
4	No	Thur	20.2700	2.830	2.0
5	No	Fri	NaN	NaN	NaN
6	No	Sat	17.7100	2.750	3.5