04_데이터 정제하기

• 누락값 처리

> 누락값 확인하기

- 누락값 : NaN으로 표기되어 있는 값

```
import numpy as np
print(np.NaN)
print(np.nan)
print(np.NAN)
```

Out : nan
nan
nan

• 누락값 처리

> 누락값 확인하기

- 누락값 : NaN으로 표기되어 있는 값

```
print(np.NaN == np.NaN)
print(np.NaN == None)
print(np.NaN == False)
print(np.NaN == 0)
```

Out : False

False

False

False

값 자체가 없기에 어떠한 값과 비교해도 똑같지 않다.

비교나 검색 등으로 NaN을 정의할 수 없음

데이터 정제하기

Type text here

• 누락값 처리

04

> isnull, notnull 메서드로 누락값 확인하기

- isnull : 누락값이 있으면 True
- notnull : 누락값이 없으면 True

```
import pandas as pd
print(pd.isnull(np.NaN))
print(pd.isnull(None))
print(pd.notnull(np.NaN))
print(pd.notnull(False))
print(pd.notnull(0))
```

Out : True

True

False

True

True

데이터 정제하기

- 누락값 처리

> 누락값 개수 확인

- ebola 데이터셋 가져오기

```
ebola = pd.read_csv('data/country_timeseries.csv')
ebola.head()
```

Out :

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone	Cases_Nigeria	Cases_Senegal	Cases_UnitedStates	Cases_Spain	Cases_Mali
0	1/5/2015	289	2776.0	NaN	10030.0	NaN	NaN	NaN	NaN	NaN
1	1/4/2015	288	2775.0	NaN	9780.0	NaN	NaN	NaN	NaN	NaN
2	1/3/2015	287	2769.0	8166.0	9722.0	NaN	NaN	NaN	NaN	NaN
3	1/2/2015	286	NaN	8157.0	NaN	NaN	NaN	NaN	NaN	NaN
4	12/31/2014	284	2730.0	8115.0	9633.0	NaN	NaN	NaN	NaN	NaN

Deaths_Guinea	Deaths_Liberia	Deaths_SierraLeone	Deaths_Nigeria	Deaths_Senegal	Deaths_UnitedStates	Deaths_Spain	Deaths_Mali
1786.0	NaN	2977.0	NaN	NaN	NaN	NaN	NaN
1781.0	NaN	2943.0	NaN	NaN	NaN	NaN	NaN
1767.0	3496.0	2915.0	NaN	NaN	NaN	NaN	NaN
NaN	3496.0	NaN	NaN	NaN	NaN	NaN	NaN
1739.0	3471.0	2827.0	NaN	NaN	NaN	NaN	NaN

데이터 정제하기

• 누락값 처리

> 누락값 개수 확인

- count() 메서드 사용 0이 아닌 NaN값이 아닌 갯수 알기

방법2방법2 ebola.count()

Out :

```
Date                    122
Day                     122
Cases_Guinea             93
Cases_Liberia            83
Cases_SierraLeone        87
Cases_Nigeria            38
Cases_Senegal            25
Cases_UnitedStates       18
Cases_Spain              16
Cases_Mali               12
Deaths_Guinea            92
Deaths_Liberia           81
Deaths_SierraLeone       87
Deaths_Nigeria           38
Deaths_Senegal           22
Deaths_UnitedStates      18
Deaths_Spain             16
Deaths_Mali              12
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122 entries, 0 to 121
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Date                 122 non-null    object
 1   Day                  122 non-null    int64
 2   Cases_Guinea         93 non-null     float64
 3   Cases_Liberia        83 non-null     float64
 4   Cases_SierraLeone    87 non-null     float64
 5   Cases_Nigeria        38 non-null     float64
 6   Cases_Senegal        25 non-null     float64
 7   Cases_UnitedStates   18 non-null     float64
 8   Cases_Spain          16 non-null     float64
 9   Cases_Mali           12 non-null     float64
 10  Deaths_Guinea        92 non-null     float64
 11  Deaths_Liberia       81 non-null     float64
 12  Deaths_SierraLeone   87 non-null     float64
 13  Deaths_Nigeria       38 non-null     float64
 14  Deaths_Senegal       22 non-null     float64
 15  Deaths_UnitedStates  18 non-null     float64
 16  Deaths_Spain         16 non-null     float64
 17  Deaths_Mali          12 non-null     float64
dtypes: float64(16), int64(1), object(1)
memory usage: 17.3+ KB
```

• 누락값 처리

> 누락값 개수 확인

- count() 메서드 사용

방법1)
```
ebola.shape[0] - ebola.count()
```

Out :

```
Date                    0
Day                     0
Cases_Guinea           29
Cases_Liberia          39
Cases_SierraLeone      35
Cases_Nigeria          84
Cases_Senegal          97
Cases_UnitedStates    104
Cases_Spain           106
Cases_Mali            110
Deaths_Guinea          30
Deaths_Liberia         41
Deaths_SierraLeone     35
Deaths_Nigeria         84
Deaths_Senegal        100
Deaths_UnitedStates   104
Deaths_Spain          106
Deaths_Mali           110
dtype: int64
```

ebola.shape[0]의 값은 전체 행의 길이

데이터 정제하기

- 누락값 처리

> 누락값 개수 확인
  - isnull() 메서드 사용

방법2) `ebola.isnull()` True와 False로 답나옴

Out :

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone	Cases_Nigeria	Cases_Senegal	Cases_UnitedStates	Cases_Spain	Cases_Mali	Deaths_G
0	False	False	False	True	False	True	True	True	True	True	
1	False	False	False	True	False	True	True	True	True	True	
2	False	False	False	False	False	True	True	True	True	True	
3	False	False	True	False	True	True	True	True	True	True	
4	False	False	False	False	False	True	True	True	True	True	
...	...	...	...	...	...	...	...	...	...	...	
117	False	False	False	False	False	True	True	True	True	True	
118	False	False	False	True	True	True	True	True	True	True	
119	False	False	False	True	True	True	True	True	True	True	
120	False	False	False	True	True	True	True	True	True	True	
121	False	False	False	True	True	True	True	True	True	True	

• 누락값 처리

> 누락값 개수 확인

- isnull() 과 sum() 메서드 함께 사용

```
ebola.isnull().sum()
```

Out :

```
Date                    0
Day                     0
Cases_Guinea           29
Cases_Liberia          39
Cases_SierraLeone      35
Cases_Nigeria          84
Cases_Senegal          97
Cases_UnitedStates    104
Cases_Spain           106
Cases_Mali            110
Deaths_Guinea          30
Deaths_Liberia         41
Deaths_SierraLeone     35
Deaths_Nigeria         84
Deaths_Senegal        100
Deaths_UnitedStates   104
Deaths_Spain          106
Deaths_Mali           110
dtype: int64
```

전체 누락값 갯수
-
ebola.isull().sum(),sum()

• 누락값 처리

> 누락값 개수 확인

- value_counts() 메서드 사용 – 시리즈에 사용

방법3

```
ebola['Cases_Guinea'].value_counts(dropna = False)
```

Out :

```
NaN       29
86.0       3
495.0      2
112.0      2
390.0      2
          ..
1199.0     1
1298.0     1
1350.0     1
1472.0     1
49.0       1
```

dropna값이 false일때와 true일대

```
#언제 누락값을 채울지 아님 버릴지 결정할때-비율 따져서 볼것
ebola.notnull().sum() / ebola.shape[0] *100
해보고 null 값 어느정도인지 볼 것!
ebola.isnull().sum() / ebola.shape[0] *100
```

누락값 않으면 지우기로, 적으면 채우기

데이터 정제하기

• 누락값 처리

> **누락값 변경하기**

[누락값 처리방법]
-채우기
-지우기

<채우기>

- fillna 메서드

방법1

```
ebola.fillna(0)
```

0으로 채우기

Out :

Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone
1/5/2015	289	2776.0	0.0	10030.0
1/4/2015	288	2775.0	0.0	9780.0
1/3/2015	287	2769.0	8166.0	9722.0
1/2/2015	286	0.0	8157.0	0.0
12/31/2014	284	2730.0	8115.0	9633.0

누락값을 fillna로 ffil/bfill 등으로 채우는 경우는 매우 적음
보통 통계적인 nan값을 넣음

채우기 방법은 데이터간의 연관이 있어야만 사용함
-앞뒤값이 시간적 인과적 연관이 증가도는 감소하는 경향이 있어야

• 누락값 처리

> 누락값 변경하기

- fillna 메서드 - ffill

방법2

```
ebola.fillna(method = 'ffill')
```

Out :

Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone
1/5/2015	289	2776.0	NaN	10030.0
1/4/2015	288	2775.0	NaN	9780.0
1/3/2015	287	2769.0	8166.0	9722.0
1/2/2015	286	2769.0	8157.0	9722.0
12/31/2014	284	2730.0	8115.0	9633.0

앞 데이터를 똑같이 채워주기에 앞에 데이터가 없으면 값을 채우지 못한다.

• 누락값 처리

> 누락값 변경하기

- fillna 메서드 - bfill

방법3

```
ebola.fillna(method = 'bfill')
```

Out :

Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone
1/5/2015	289	2776.0	8166.0	10030.0
1/4/2015	288	2775.0	8166.0	9780.0
1/3/2015	287	2769.0	8166.0	9722.0
1/2/2015	286	2730.0	8157.0	9633.0
12/31/2014	284	2730.0	8115.0	9633.0
...	...	...	...	...
3/27/2014	5	103.0	8.0	6.0
3/26/2014	4	86.0	NaN	NaN
3/25/2014	3	86.0	NaN	NaN
3/24/2014	2	86.0	NaN	NaN
3/22/2014	0	49.0	NaN	NaN

남은 NaN은 0으로 채우기

```
ebola.fillna(method='bfill').fillna(0)
```

데이터 정제하기

• 누락값 처리

> 누락값 변경하기

- interpolate 메서드 보간법

방법4

```
ebola.interpolate()
```

Out :

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone
0	1/5/2015	289	2776.0	NaN	10030.0
1	1/4/2015	288	2775.0	NaN	9780.0
2	1/3/2015	287	2769.0	8166.0	9722.0
3	1/2/2015	286	2749.5	8157.0	9677.5
4	12/31/2014	284	2730.0	8115.0	9633.0
...	...	...	...	...	...
117	3/27/2014	5	103.0	8.0	6.0
118	3/26/2014	4	86.0	8.0	6.0
119	3/25/2014	3	86.0	8.0	6.0
120	3/24/2014	2	86.0	8.0	6.0
121	3/22/2014	0	49.0	8.0	6.0

** 문제 있음
---0명이라고 해야함

데이터 정제하기

- 누락값 처리

> **누락값 삭제하기**
  - dropna 메서드 NAN값을 모두 확인하여 row값을 지움

지우기 방법

```
ebola.dropna()
```

Out :

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone	Cases_Nigeria
19	11/18/2014	241	2047.0	7082.0	6190.0	20.0

```
ebola.dropna(axis = 1)
```

Out :

	Date	Day
0	1/5/2015	289
1	1/4/2015	288
2	1/3/2015	287
3	1/2/2015	286
4	12/31/2014	284

• 누락값 처리

> 누락값 일부분만 삭제하기
  - dropna(subset = ['열 이름'])

```
ebola.dropna(subset=['Cases_Guinea'])
```

Out :

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone
0	1/5/2015	289	2776.0	NaN	10030.0
1	1/4/2015	288	2775.0	NaN	9780.0
2	1/3/2015	287	2769.0	8166.0	9722.0
4	12/31/2014	284	2730.0	8115.0	9633.0
5	12/28/2014	281	2706.0	8018.0	9446.0

• 간단한 함수 만들기

> **제곱 함수와 n 제곱 함수 만들기**

- 제곱 사용자 함수

```
def my_sq(x):
    return x ** 2
```

- n 제곱 사용자 함수

```
def my_exp(x, n):
    return x ** n
```

- 사용자 함수 사용

```
print(my_sq(4))
print(my_exp(3, 4))
```

Out : 16
81

• 간단한 함수 만들기

> apply 메소드 사용하기 – 기초

- 데이터프레임 준비

```
import pandas as pd


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

Out :

	a	b
0	10	20
1	20	30
2	30	40

• apply 메소드 사용하기 – 기초

> **시리즈에 apply 메소드 사용**

- 제곱 함수 (my_sq) 적용 : 인자 1개

함수는 실행이 아니라 정보임

```
sq = df['a'].apply(my_sq)
print(sq)
```

Out :

```
0     100
1     400
2     900
Name: a, dtype: int64
```

함수 my_sq에 시리즈 a의 값을 넣는다
모든 시리즈값이 함수로 적용됨

```
def my_sq(x):
    return x ** 2
```

	a	b
0	10	20
1	20	30
2	30	40

• apply 메소드 사용하기 – 기초

> 시리즈에 apply 메소드 사용

- n 제곱 함수 (my_exp) 적용 : 인자 2개

```
ex = df['a'].apply(my_exp, n=2)
print(ex)
```

Out :

```
0     100
1     400
2     900
Name: a, dtype: int64
```

함수 인자 x에는 시리즈의 값 적용
함수인자 n은 n=2 값을 씀

```
def my_exp(x, n):
    return x ** n
```

	a	b
0	10	20
1	20	30
2	30	40

• apply 메소드 사용하기 – 기초

> **데이터프레임에 apply 메소드 사용**

보통 apply는 시리즈에서 사용함

- 사용자 함수 준비

```
def print_me(x):
    print(x)
```

- 열방향 적용 axis = 0

```
df.apply(print_me, axis = 0)
```

Out :

```
0    10
1    20
2    30
Name: a, dtype: int64
0    20
1    30
2    40
Name: b, dtype: int64

a    None
b    None
dtype: object
```

리턴값이 없으므로 값이 None값

• apply 메소드 사용하기 – 기초

> 데이터프레임에 apply 메소드 사용

- 행방향 적용 axis = 1

```
df.apply(print_me, axis = 0)
```

Out :

```
a    10
b    20
Name: 0, dtype: int64
a    20
b    30
Name: 1, dtype: int64
a    30
b    40
Name: 2, dtype: int64

0    None
1    None
2    None
dtype: object
```

• apply 메소드 사용하기 – 기초

> apply 메소드 활용

- 시험 점수 데이터

```
exam = pd.read_csv('data/exam.csv')
exam.head()
```

Out :

	id	nclass	math	english	science
0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78
3	4	1	30	98	58
4	5	2	25	80	65

• apply 메소드 사용하기 – 기초

> apply 메소드 활용

- 100점 만점 점수를 10점 만점으로 바꾸기

방법4

```
exam['math'].apply(lambda x : x // 10)
```

Out :

```
0     5
1     6
2     4
3     3
4     2
5     5
6     8
7     9
8     2
9     5
10    6
11    4
12    4
13    4
14    7
15    5
16    6
17    8
18    8
19    7
Name: math, dtype: int64
```

lambda 표현식이 apply에 잘 사용되므로 잘 익혀야 함Type text here

예:df['a'].apply(lambda x,n: x ** n , n=2)

람바 사용 안하고도 기본 기능으로도 가능함
꼭 해야 할 경우에 사용

• apply 메소드 사용하기 – 기초

> **apply 메소드 활용** 대이타프레임 게산 잘 안함

- 100점 만점 점수를 10점 만점으로 바꾸기

```
exam.apply(lambda x : x.apply(lambda y: y // 10), axis = 1)
```

Out :

	id	nclass	math	english	science
0	0	0	5	9	5
1	0	0	6	9	6
2	0	0	4	8	7
3	0	0	3	9	5
4	0	0	2	8	6
5	0	0	5	8	9
6	0	0	8	9	4
7	0	0	9	7	2
8	0	0	2	9	1

• apply 메소드 사용하기 – 기초

> 데이터프레임에 apply 메소드 활용

- 총 점수, 평균 구하기

```
exam.apply(lambda x : x['math'] + x['english'] + x['science'],
           axis = 1)
```

Out :

0	198
1	217
2	209
3	186
4	170
5	237
6	215
7	193
8	133

• apply 메소드 사용하기 – 기초

> 데이터프레임에 apply 메소드 활용

- 총 점수, 평균 구하기

```
exam.apply(lambda x : (x['math'] + x['english'] + x['science'])/3,
                axis = 1)
```

Out :

0	66.000000
1	72.333333
2	69.666667
3	62.000000
4	56.666667
5	79.000000
6	71.666667
7	64.333333
8	44.333333

• apply 메소드 사용하기 – 기초

> 데이터프레임에 apply 메소드 활용

- 총 점수, 평균 구하기 – 가장 흔한 방법

```
exam['math'] + exam['english'] + exam['science']
```

Out :

0	198
1	217
2	209
3	186
4	170
5	237
6	215
7	193
8	133

lambda 표현식이 apply에 잘 사용되므로 잘 익혀야 함

예:df['a'].apply(lambda x,n: x ** n , n=2)

• apply 메소드 사용하기 – 고급

> apply 메소드 활용법

- 타이타닉 데이터

```
titanic = pd.read_csv('data/titanic.csv')
titanic.head()
```

Out :

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

parch: 부모,자녀의 수

sibsp: 형제자매 수

cabin: 객차
embarked: 선착장

• apply 메소드 사용하기 – 고급

> apply 메소드 활용법

- 타이타닉 데이터

```
titanic.info()
```

Out :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

• apply 메소드 사용하기 – 고급

> apply 메소드 활용법

- apply 함수를 사용하여 누락값 개수 확인하기

```
def count_missing(vec):
    null_vec = pd.isnull(vec)
    null_count = np.sum(null_vec)
    return null_count

cmis_col = titanic.apply(count_missing)
cmis_col
```

Out :

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

• apply 메소드 사용하기 – 고급

> apply 메소드 활용법

- apply 함수를 사용하여 누락값 비율 확인하기

```
def prop_missing(vec):
    num = count_missing(vec)
    dem = vec.size
    return num / dem

pmis_col = titanic.apply(prop_missing)
print(pmis_col)
```

Type text here

Out :

```
PassengerId    0.000000
Survived       0.000000
Pclass         0.000000
Name           0.000000
Sex            0.000000
Age            0.198653
SibSp          0.000000
Parch          0.000000
Ticket         0.000000
Fare           0.000000
Cabin          0.771044
Embarked       0.002245
dtype: float64
```

• 데이터 집계

> groupby 메서드로 평균값 구하기

- 갭마인더 데이터

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

- year 열을 기준으로 그룹화하고 lifeExp 평균 구하기

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

Out :

```
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 메서드의 집계 메서드** 집계함수

시리즈 등 열이나 행을 하나의 값으로 반환

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
describe	통계 요약 정보 반환
first	첫번째 행 반환
last	마지막 행 반환
nth	n번째 행 반환

• 데이터 집계

> groupby에 사용자 함수 적용

- agg() : apply와 유사 apply()를 대신 써도 됨

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

• 데이터 집계

> groupby에 사용자 함수 적용

- agg() : apply와 유사

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

Out :

```
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

> groupby에 사용자 함수 적용

- 2개의 인자 사용법

```
#agg없이 계산하기
df.groupby('year')['lifeExp'].mean() - global_mean
```

```
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
```

```
global_mean = df['lifeExp'].mean()
print(global_mean)
```

Out : 59.47443936619714

• 데이터 집계

> groupby에 사용자 함수 적용

- 2개의 인자 사용법

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

Out :

```
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 메서드 한번에 적용

- mean, std 적용

```
df.groupby('year')['lifeExp'].agg(['mean', 'std'])
```

Out :

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

pd에서는 없으므로 문자로 주어야 함
sum,len 등은 그냥 써도 됨

• 데이터 집계

> 여러 개의 집계 메서드 한번에 적용

- 넘파이 함수 np.mean, np.std, 사용자 함수 적용

방법2방

```
import numpy as np
df.groupby('year')['lifeExp'].agg([np.mean, np.std, my_mean])
```

Out :

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

그냥 'mean' 등으로 사용 가능

```
#agg 사용법
df.groupby('year').agg({'lifeExp':['mean','sum'], 'pop':'sum'})
```

• 데이터 집계

> 그룹 오브젝트

- 그룹 오브젝트의 속성 확인하기

```
grouped = df.groupby('year')
grouped
```

Out : `<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001F947FA54C0>`

- 그룹 인덱스 확인

```
grouped.groups
```

Out :
```
{1957: [937], 1972: [1456], 1977: [1493], 1992: [776, 1544],
1997: [1545], 2002: [1342, 106], 2007: [251, 215]}
```

• 데이터 집계

> 그룹 오브젝트

- 그룹명을 지정하여 데이터 추출

```
grouped.get_group(2002)
```

Out :

	country	continent	year	lifeExp	pop	gdpPercap	fill_lifeExp
1342	Serbia	Europe	2002	73.213	10111559	7236.075251	73.213
106	Bangladesh	Asia	2002	NaN	135656790	1136.390430	73.213

- 각 그룹 정보 추출

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

Out :

```
(1957,       country continent  year  lifeExp      pop    gdpPercap  fill_lifeExp
937  Malaysia      Asia  1957   52.102  7739235  1810.066992        52.102)
(1972,         country continent  year  lifeExp      pop    gdpPercap  fill_lifeExp
1456  Swaziland    Africa  1972      NaN  480105  3364.836625          NaN)
```