Data Preprocessing

- I. Missing Value
- II. Filtering
- III. 데이터프레임 합치기
- IV. 그룹 연산
- V. pivot_table( )
- VI. Multi-Index
- VII. etc

I. Missing Value

1) 실습용 'titanic' 데이터셋

```
import seaborn as sns
TD = sns.load_dataset('titanic')
```

- 'titanic' Dataset Information

```
TD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

2) 결측치 확인

- 'age' 및 'deck' 열(Column)에서 결측치(NaN) 확인

```
TD.head(10)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	NaN	Southampton	no	False
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	NaN	Southampton	yes	False
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN	Cherbourg	yes	False

- .value_counts(dropna = False)
  - 결측치(NaN)를 포함하여 빈도분석 결과 출력

```
TD['deck'].value_counts(dropna = False)
```

```
NaN    688
C       59
B       47
D       33
E       32
A       15
F       13
G        4
Name: deck, dtype: int64
```

- .isnull( )
  - 결측치(NaN)를 'True'로 출력

```
TD.head(10).isnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False

- 각 열(Column)별로 결측치(NaN) 개수 확인
- .isnull( ).sum(axis = 0)
  - axis = 0: 행(Row)
  - axis = 1: 열(Column)

```
TD.isnull().sum(axis = 0)
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

- 각 행(Row)별로 결측치(NaN) 개수 확인
- .isnull( ).sum(axis = 1)

```
TD.isnull().sum(axis = 1).value_counts()
```

```
1    549
0    182
2    160
dtype: int64
```

- .notnull( )
  - 결측치(NaN)를 'False'로 출력

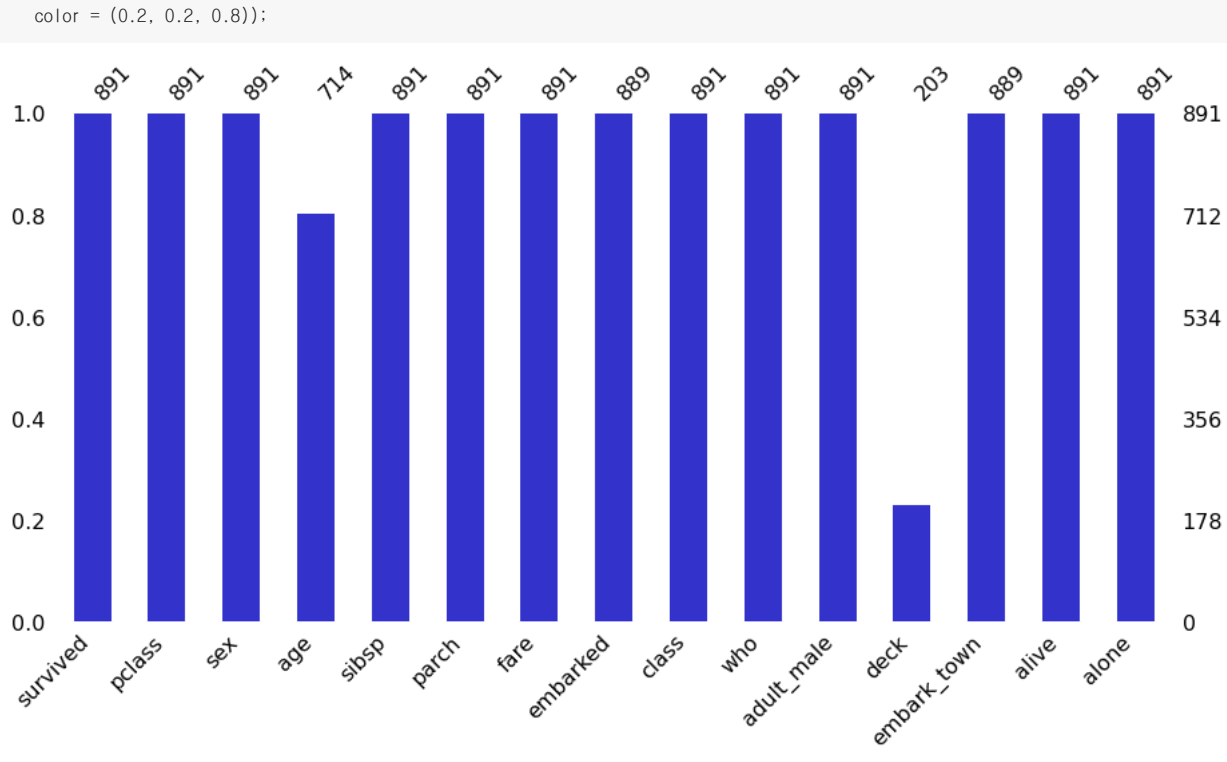
```
TD.head(10).notnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
5	True	True	True	False	True	True	True	True	True	True	True	False	True	True	True
6	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
7	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
8	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
9	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True

(1) 격측치 막대 그래프

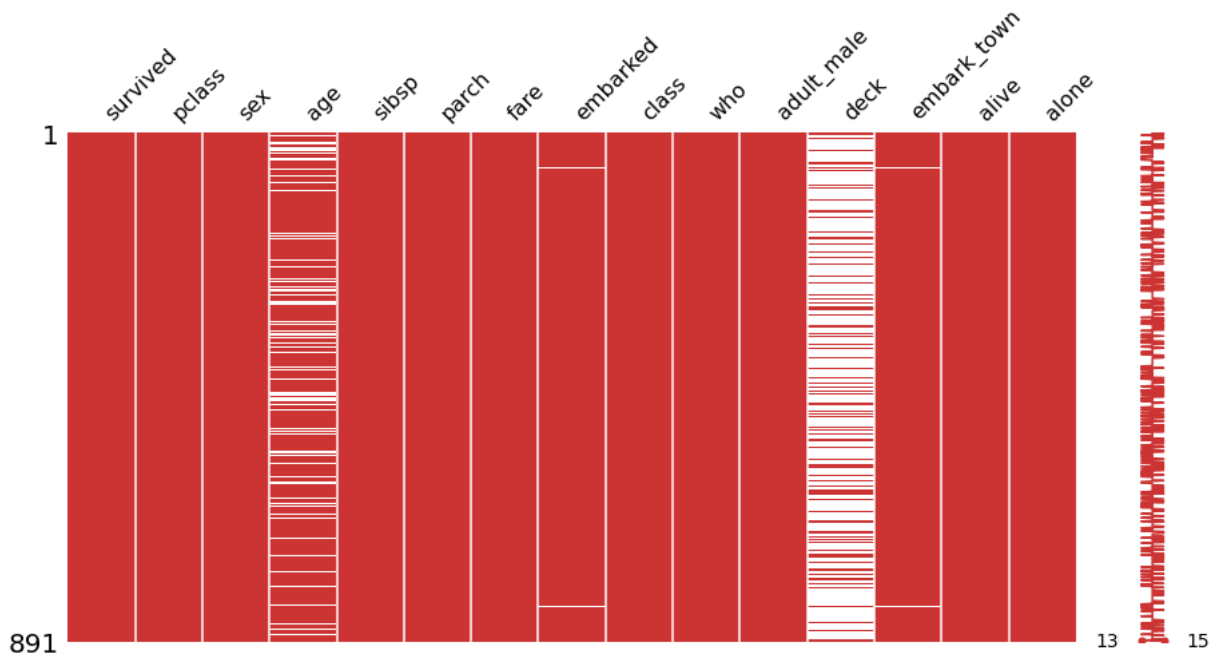
```
import missingno as msno

msno.bar(TD,
         figsize = (15, 7),
         color = (0.2, 0.2, 0.8));
```



(2) 결측치 Matrix

```
msno.matrix(TD,
            figsize = (15, 7),
            color = (0.8, 0.2, 0.2));
```



3) 결측치 삭제

- 각 열(Column)별로 측정값(Non-NaN) 개수 확인

```
TD.notnull().sum(axis = 0)
```

```
survived       891
pclass         891
sex            891
age            714
sibsp          891
parch          891
fare           891
embarked       889
class          891
who            891
adult_male     891
deck           203
embark_town    889
alive          891
alone          891
dtype: int64
```

- 300개 이하 측정값(Non-Null)이 있는 열(Column) 삭제
  - .dropna(thresh = 300, axis = 1)
  - 'deck' 열 삭제

- 15 -> 14

```
TD.dropna(thresh = 300, axis = 1).shape
```

```
(891, 14)
```

- 'age'행(Column) 기준으로 결측치가 있는 행(Row) 삭제
  - .dropna(subset = ['age'], how = 'any', axis = 0)
  - how = 'all' : 모든 값이 결측치인 경우 삭제

```
TD.shape
```

```
(891, 15)
```

- 891 -> 714

```
TD.dropna(subset = ['age'], how = 'any', axis = 0).shape
```

```
(714, 15)
```

- 891 -> 182

```
TD.dropna(subset = ['age', 'embarked', 'deck', 'embark_town'], how = 'any', axis = 0).shape
```

```
(182, 15)
```

4) 격측치 치환

- 연속형 데이터 치환
  - 'age'의 결측치를 평균값으로 치환
  - .fillna(int(DF['age'].mean(axis = 0)), inplace = True)

- 결측치(NaN) 확인

```
TD['age'][4:7]
```

```
4    35.0
5     NaN
6    54.0
Name: age, dtype: float64
```

- 평균값으로 치환
  - 함수 적용 후 결과가 반영된 데이터프레임 반환: inplace = True

```
TD['age'].fillna(int(TD['age'].mean(axis = 0)), inplace = True)
```

- NaN -> 29.0

```
TD['age'][4:7]
```

```
4    35.0
5    29.0
6    54.0
Name: age, dtype: float64
```

- 명목형 데이터 치환
  - 'embark_town'의 결측치를 최빈값으로 치환
  - .fillna(most_freq, inplace = True)

- 결측치(NaN) 확인

```
TD['embark_town'][828:831]
```

```
828    Queenstown
829           NaN
830     Cherbourg
Name: embark_town, dtype: object
```

- 최빈값 확인

```
most_freq = TD['embark_town'].value_counts(dropna = True).idxmax()

most_freq
```

```
'Southampton'
```

- 최빈값으로 치환

```
TD['embark_town'].fillna(most_freq, inplace = True)
```

- NaN -> Southampton

```
TD['embark_town'][828:831]
```

```
828     Queenstown
829    Southampton
830      Cherbourg
Name: embark_town, dtype: object
```

- 결측치 치환 with 'ffill'
  - 이전 데이터포인트로 치환
  - .fillna(method = 'ffill', inplace = True)

```
TD = sns.load_dataset('titanic')

TD['embark_town'][828:831]
```

```
828    Queenstown
829           NaN
830     Cherbourg
Name: embark_town, dtype: object
```

- method = 'ffill'

```
TD['embark_town'].fillna(method = 'ffill', inplace = True)

TD['embark_town'][828:831]
```

```
828    Queenstown
829    Queenstown
830     Cherbourg
Name: embark_town, dtype: object
```

- 결측치 치환 with 'bfill'
  - 다음 데이터포인트로 치환
  - .fillna(method = 'bfill', inplace = True)

```
TD = sns.load_dataset('titanic')

TD['embark_town'][828:831]
```

```
828    Queenstown
829           NaN
830     Cherbourg
Name: embark_town, dtype: object
```

- method = 'bfill'

```
TD['embark_town'].fillna(method = 'bfill', inplace = True)

TD['embark_town'][828:831]
```

```
828    Queenstown
829     Cherbourg
830     Cherbourg
Name: embark_town, dtype: object
```

II. Filtering

1) 실습용 'titanic' 데이터셋

```
import seaborn as sns
TD = sns.load_dataset('titanic')

TD.head(3)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True

2) 'age'가 10살 이상이면서 20살 미만

- (TD.age >= 10) & (TD.age < 20)

```
Filter_1 = (TD.age >= 10) & (TD.age < 20)

TD.loc[Filter_1, :].head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN	Cherbourg	yes	False
14	0	3	female	14.0	0	0	7.8542	S	Third	child	False	NaN	Southampton	no	True
22	1	3	female	15.0	0	0	8.0292	Q	Third	child	False	NaN	Queenstown	yes	True
27	0	1	male	19.0	3	2	263.0000	S	First	man	True	C	Southampton	no	False
38	0	3	female	18.0	2	0	18.0000	S	Third	woman	False	NaN	Southampton	no	False

3) 'age'가 10살 미만이면서 'sex'이 여자

- (TD.age < 10) & (TD.sex == 'female')

```
Filter_2 = (TD.age < 10) & (TD.sex == 'female')

TD.loc[Filter_2, :].head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton	yes	False
24	0	3	female	8.0	3	1	21.0750	S	Third	child	False	NaN	Southampton	no	False
43	1	2	female	3.0	1	2	41.5792	C	Second	child	False	NaN	Cherbourg	yes	False
58	1	2	female	5.0	1	2	27.7500	S	Second	child	False	NaN	Southampton	yes	False
119	0	3	female	2.0	4	2	31.2750	S	Third	child	False	NaN	Southampton	no	False

4) 'age'가 10살 미만 또는 60살 이상

- (TD.age < 10) | (TD.age >= 60)
  - 'age', 'sex', 'alone' 열(Column)만 출력

```
Filter_3 = (TD.age < 10) | (TD.age >= 60)

TD.loc[Filter_3, ['age','sex','alone']].head()
```

5) 'sibsp'에 3 또는 4 또는 5를 포함

- (TD.sibsp == 3) | (TD.sibsp == 4) | (TD.sibsp == 5)

```
Filter_4 = (TD.sibsp == 3) | (TD.sibsp == 4) | (TD.sibsp == 5)

TD.loc[Filter_4, :].head(6)
```

III. 데이터프레임 합치기

1) 데이터프레임 TB1, TB2, TB3, TB4 생성

- TB1

```
import pandas as pd

TB1 = pd.DataFrame({'Name': ['송태섭', '최유정', '이한나', '김소혜'],
                    'Gender': ['남자', '여자', '여자', '여자'],
                    'Age': ['21', '23', '20', '23']})


TB1
```

- TB2

```
TB2 = pd.DataFrame({'Name': ['송태섭', '최유정', '이한나', '김소혜'],
                    'Gender': ['남자', '여자', '여자', '여자'],
                    'Height': [179.1, 177.1, 167.9, 176.1],
                    'Weight': [63.9, 54.9, 50.2, 53.5]})

TB2
```

- TB3

```
TB3 = pd.DataFrame({'Name': ['서태웅', '정대만'],
                    'Gender': ['남자', '남자'],
                    'Age': ['24', '24']})


TB3
```

- TB4

```
TB4 = pd.DataFrame({'Grade': [3, 1, 1, 3],
                    'Picture': ['무', '유', '무', '무']})


TB4
```

2) .concat( )

- 행기준: axis = 0
  - TB1 & TB2

```
pd.concat([TB1, TB2], axis = 0)
```

- 행기준: axis = 0
  - TB1 & TB3
- ignore_index = True
  - Label 새로 구성

```
pd.concat([TB1, TB3], axis = 0, ignore_index = True)
```

- 열기준: axis = 1
  - TB1 & TB2

```
pd.concat([TB1, TB2], axis = 1)
```

- 열기준: axis = 1
  - TB1 & TB4

```
pd.concat([TB1, TB4], axis = 1)
```

3) merge( )

- on = ['Name', 'Gender']
  - TB1 & TB2

```
pd.merge(TB1, TB2, on = ['Name', 'Gender'])
```

IV. 그룹 연산

1) 실습용 'titanic' 데이터셋

```
import seaborn as sns
titanic = sns.load_dataset('titanic')

TD = titanic.loc[:, ['age', 'sex', 'class', 'fare', 'survived']]

TD.head()
```

2) groupby( ) - 'class' 기준

- 'class' 기준의 DataFrameGroupBy 객체 생성

```
grouped = TD.groupby(['class'])
```

```
grouped
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f0872784f10>
```

- 'First' 키 그룹 정보 확인
  - .get_group('First')

```
grouped.get_group('First').head(3)
```

- groupby 결과 확인(3개 그룹)
  - 'First', 'Second', 'Third' 키별 3줄씩 출력
  - .get_group('Key_Name')

```
for key in ['First', 'Second', 'Third']:
  print(grouped.get_group(key).head(3))
  print('₩n')
```

```
    age     sex  class     fare  survived
1  38.0  female  First  71.2833         1
3  35.0  female  First  53.1000         1
6  54.0    male  First  51.8625         0


    age     sex   class     fare  survived
9  14.0  female  Second  30.0708         1
```

```
    15  55.0  female  Second  16.0000         1
    17   NaN    male  Second  13.0000         1


       age     sex  class   fare  survived
0  22.0    male  Third  7.250         0
2  26.0  female  Third  7.925         1
4  35.0    male  Third  8.050         0
```

- 3개 그룹별 평균('age', 'fare', 'survived')

```
grouped.mean()
```

3) groupby( ) - 'class' & 'sex' 기준

- 두 개 키(Key) 사용하여 DataFrameGoupBy 객체 생성
  - 'class', 'sex' 키 적용

```
grouped_TWO = TD.groupby(['class', 'sex'])

grouped_TWO
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f0872758a60>
```

- ('First', 'female') 키 그룹 정보 확인
  - .get_group(('First', 'female'))

```
grouped_TWO.get_group(('First', 'female')).head(3)
```

- groupby 결과 확인(6개 그룹)

```
for key, group in grouped_TWO:
  print('* key :', key)
  print('* number :', len(group))
  print(group.head(3))
  print('₩n')
```

```
    * key : ('First', 'female')
    * number : 94
        age     sex  class     fare  survived
    1   38.0  female  First  71.2833         1
    3   35.0  female  First  53.1000         1
    11  58.0  female  First  26.5500         1


    * key : ('First', 'male')
    * number : 122
        age   sex  class      fare  survived
    6   54.0  male  First   51.8625         0
    23  28.0  male  First   35.5000         1
    27  19.0  male  First  263.0000         0


    * key : ('Second', 'female')
    * number : 76
        age     sex   class     fare  survived
    9   14.0  female  Second  30.0708         1
    15  55.0  female  Second  16.0000         1
    41  27.0  female  Second  21.0000         0


    * key : ('Second', 'male')
    * number : 108
        age   sex   class  fare  survived
    17   NaN  male  Second  13.0         1
    20  35.0  male  Second  26.0         0
    21  34.0  male  Second  13.0         1


    * key : ('Third', 'female')
    * number : 144
        age     sex  class     fare  survived
    2   26.0  female  Third   7.9250         1
    8   27.0  female  Third  11.1333         1
    10   4.0  female  Third  16.7000         1


    * key : ('Third', 'male')
    * number : 347
       age   sex  class    fare  survived
    0  22.0  male  Third  7.2500         0
    4  35.0  male  Third  8.0500         0
    5   NaN  male  Third  8.4583         0
```

- 6개 그룹별 평균('age', 'fare', 'survived')

```
grouped_TWO.mean()
```

4) agg( )

- Aggregation : 여러개의 함수를 groupby 객체에 적용
    - 그룹별로 연산 결과를 집계하여 반환

- grouped

```
grouped.agg(['mean', 'std'])
```

- grouped_TWO

```
grouped_TWO.agg(['mean', 'std'])
```

- fare 열에만 적용

```
grouped.fare.agg(['min', 'max'])
```

- fare 및 age 열에 각각 다른 함수 적용

```
grouped.agg({'fare' : ['min', 'max'], 'age' : ['mean', 'std']})
```

5) filter( )

- 데이터 개수가 200개 이상인 그룹의 결과만 필터링
    - 'First', 'Third'

```
grouped.filter(lambda x : len(x) >= 200).head()
```

- 그룹별 데이터 개수 확인

```
grouped.apply(len)
```

```
class
First     216
Second    184
Third     491
dtype: int64
```

- 'age' 열 평균이 30보다 작은 그룹의 결과만 필터링
    - 'Second', 'Third'

```
grouped.filter(lambda x: x.age.mean() < 30).tail()
```

- 그룹별 'age' 열의 평균

```
grouped.age.mean()
```

```
class
First     38.233441
Second    29.877630
Third     25.140620
Name: age, dtype: float64
```

V. pivot_table( )

1) 실습용 'titanic' 데이터셋

```
import seaborn as sns
titanic = sns.load_dataset('titanic')

TD = titanic.loc[:, ['age', 'sex', 'class', 'fare', 'survived']]

TD.head(3)
```

2) pivot_table( ) 구성요소

- index : 행 인덱스
- column : 열 인덱스
- values : 데이터
- aggfunc : 적용 함수

```
TD_1 = pd.pivot_table(TD,
                      index = 'class',
                      columns = 'sex',
                      values = 'age',
                      aggfunc = 'mean')

TD_1
```

3) 두개의 적용 함수

```
TD_2 = pd.pivot_table(TD,
                      index = 'class',
                      columns = 'sex',
                      values = 'survived',
                      aggfunc = ['mean', 'sum'])

TD_2
```

4) 다중 인덱스, 다중 데이터, 다중 함수

```
TD_3 = pd.pivot_table(TD,
                      index = ['class', 'sex'],
                      columns = 'survived',
                      values = ['age','fare'],
                      aggfunc = {'age' : ['mean', 'std'], 'fare' : ['min', 'max']})

TD_3
```

VI. Multi-Index

- .xs( ): Cross Section

1) 행 멀티인덱스

- names = ['class', 'sex']

```
TD_3.index
```

```
MultiIndex([( 'First', 'female'),
            ( 'First',   'male'),
            ('Second', 'female'),
            ('Second',   'male'),
            ( 'Third', 'female'),
            ( 'Third',   'male')],
           names=['class', 'sex'])
```

- 행 멀티인덱스: 객실 등급이 일등실
  - level = 'class'

```
TD_3.xs('First', level = 'class', axis = 0)
```

- 행 멀티인덱스: 성별이 남자
  - level = 'sex'

```
TD_3.xs('male', level = 'sex', axis = 0)
```

- 행 멀티인덱스: 객실등급이 일등실이면서 성별이 남자
  - level = ['class', 'sex']

```
TD_3.xs(('First', 'male'), level = ['class', 'sex'], axis = 0)
```

2) 열 멀티인덱스

- names = [None, None, 'survived']
- names = [0, 1, 2]

```
TD_3.columns
```

```
MultiIndex([( 'age', 'mean', 0),
            ( 'age', 'mean', 1),
            ( 'age',  'std', 0),
            ( 'age',  'std', 1),
            ('fare',  'max', 0),
            ('fare',  'max', 1),
            ('fare',  'min', 0),
            ('fare',  'min', 1)],
           names=[None, None, 'survived'])
```

- .set_names( )
  - names: ['Header', 'Fuction', 'Survived']

```
```

```
TD_3.columns.set_names(['Header', 'Fuction', 'Survived'], inplace = True)
```

```
TD_3.columns
```

```
MultiIndex([( 'age', 'mean', 0),
            ( 'age', 'mean', 1),
            ( 'age',  'std', 0),
            ( 'age',  'std', 1),
            ('fare',  'max', 0),
            ('fare',  'max', 1),
            ('fare',  'min', 0),
            ('fare',  'min', 1)],
           names=['Header', 'Fuction', 'Survived'])
```

- .set_levels( )
  - Level(Survived): ['Dead', 'Alive']

```
TD_3.columns.set_levels(['Dead', 'Alive'], level = 2, inplace = True)
```

```
<ipython-input-69-ace6f01fc354>:1: FutureWarning: inplace is deprecated and will be removed in a future version.
  TD_3.columns.set_levels(['Dead', 'Alive'], level = 2, inplace = True)
```

```
TD_3.columns
```

```
MultiIndex([( 'age', 'mean',  'Dead'),
            ( 'age', 'mean', 'Alive'),
            ( 'age',  'std',  'Dead'),
            ( 'age',  'std', 'Alive'),
            ('fare',  'max',  'Dead'),
            ('fare',  'max', 'Alive'),
            ('fare',  'min',  'Dead'),
            ('fare',  'min', 'Alive')],
           names=['Header', 'Fuction', 'Survived'])
```

```
TD_3
```

- 열 멀티인덱스: 평균 나이

```
TD_3.xs('mean', level = 'Fuction', axis = 1)
```

- 열 멀티인덱스: 생존자 정보

```
TD_3.xs('Alive', level = 'Survived', axis = 1)
```

- 열 멀티인덱스: 최저 요금

```
TD_3.xs(('fare', 'min'), level = ['Header', 'Fuction'], axis = 1)
```

- 열 멀티인덱스: 사망자의 평균 나이

```
TD_3.xs(('age', 'mean', 'Dead'), level = ['Header', 'Fuction', 'Survived'], axis = 1)
```

VII. etc

1) 실습용 'titanic' 데이터셋

```
import seaborn as sns
titanic = sns.load_dataset('titanic')

TD = titanic.loc[:, ['age', 'sex', 'class', 'fare', 'survived']]

TD.head()
```

2) .value_counts( )

- Series

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

```
male      577
female    314
Name: sex, dtype: int64
```

- DataFrame

```
TD[['sex', 'class']].value_counts()
```

```
sex     class
male    Third     347
female  Third     144
male    First     122
        Second    108
female  First      94
        Second     76
dtype: int64
```

3) .nunique( )

- Series

```
TD['sex'].nunique()
```

```
2
```

- DataFrame

```
TD[['sex', 'class']].nunique()
```

```
sex      2
class    3
dtype: int64
```

4) .replace( )

- Series

```
TD.loc[[0, 4], :]
```

```
TD['sex'] = TD['sex'].replace('male', 'MAN')
```

```
TD.loc[TD['sex'] == 'MAN', :][:2]
```

- DataFrame

```
TD.loc[[1, 9, 0], :]
```

```
TD[['class']] = TD[['class']].replace({'First':'1st', 'Second':'2nd', 'Third':'3rd'})
```

```
TD.loc[[1, 9, 0], :]
```

- Missing Value

```
TD.loc[[5], :]
```

```
import numpy as np

TD[['age']] = TD[['age']].replace(np.nan, int(TD.age.mean()))
```

```
TD.loc[[5], :]
```

#

#

#

The End

#

#

#