BITAmin
2주차 Session

7기 교육부 이효석

BITAMIN

분석프로그래밍2

D Deleting

C Creating

CV Changing Values

S Sorting

FM Function Mapping

GA Group Aggragation

M Merging

A Appending

BITAMIN

Deleting

BITAmin

B Deleting (Columns)

사전작업-데이터불러오기 + 살펴보기

```
In [210]:
import pandas as pd
import numpy as np

df = pd.read_csv('서울시 코로나19 확진자 현황encoding.csv',encoding='UTF8')
df
```

Out[210]:

	연번	확진일	환자번호	국적	환자정보	지역	여행력	접촉력	조치사항	상태	이동경로	등록일	수정일	노출여부
0	27735	2021-02-23	NaN	NaN	NaN	서대문구	NaN	기타 확진자 접촉	NaN	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
1	27734	2021-02-23	NaN	NaN	NaN	타시도	NaN	기타 확진자 접촉	NaN	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
2	27733	2021-02-23	NaN	NaN	NaN	타시도	NaN	기타 확진자 접촉	NaN	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
3	27732	2021-02-23	NaN	NaN	NaN	양천구	NaN	양천구 소재 유치원/어린이집 관련	NaN	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
4	27731	2021-02-23	NaN	NaN	NaN	양천구	NaN	기타 확진자 접촉	NaN	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27730	5	2020-01-31	NaN	NaN	NaN	성북구	NaN	기타 확진자 접촉	NaN	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y

```
In [3]:
df.isnull().sum()
```

Out[3]:

```
연번          0
확진일         0
환자번호    27735
국적      27735
환자정보    27735
지역          0
여행력     26828
접촉력         0
조치사항    27735
상태         3552
이동경로      1852
등록일         0
수정일         0
노출여부        0
```

1.앞으로의 진행을 위한 데이터를 불러오자.

2.딱 보니 데이터에 결측치가 많아 보인다.

3.환자번호, 국적, 환자정보, 조치사항 열은 모두 결측치로 이루어져있다. 해당 열들을 지워보자.

B Deleting (Columns)

열 버리기 – drop

```
In [3]: df.drop(['환자번호','국적','환자정보','조치사항'],axis=1,inplace=True)

In [ ]: df = df.drop(['환자번호','국적','환자정보','조치사항'],axis=1)

In [4]: df
```

Out[4]:

	연번	확진일	지역	여행력	접촉력	상태	이동경로	등록일	수정일	노출여부
0	27735	2021-02-23	서대문구	NaN	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
1	27734	2021-02-23	타시도	NaN	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
2	27733	2021-02-23	타시도	NaN	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
3	27732	2021-02-23	양천구	NaN	양천구 소재 유치원/어린이집 관련	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
4	27731	2021-02-23	양천구	NaN	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
...	...	...	...	...	...	...	...	...	...	...
27730	5	2020-01-31	성북구	NaN	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27731	4	2020-01-30	마포구	중국	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27732	3	2020-01-30	종로구	NaN	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27733	2	2020-01-30	중랑구	중국	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27734	1	2020-01-24	강서구	중국	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y

27735 rows × 10 columns

1.열을 버리는 방법은 drop메소드를 이용하는 것이다.

2.drop메소드에서 열을 버리고 싶다면 axis를 통해서 축을 수직으로 설정해서 버려야한다.

3. drop메소드는 기존 객체를 변경하지 않기 때문에 원본객체인 df를 변경하기 위해서는 inplace= True를 추가해주거나 원본객체에 새로 assign해줘야한다.

B Deleting (Columns)

열 버리기 - del

```
In [7]: del df['여행력']
```

```
In [12]: df
```

Out[12]:

	연번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부
0	27735	2021-02-23	서대문구	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
1	27734	2021-02-23	타시도	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
2	27733	2021-02-23	타시도	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
3	27732	2021-02-23	양천구	양천구 소재 유치원/어린이집 관련	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
4	27731	2021-02-23	양천구	기타 확진자 접촉	NaN	NaN	2021-02-24 10:18	2021-02-24 10:18	Y
...	...	...	...	...	...	...	...	...	...
27730	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27731	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27732	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27733	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
27734	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y

27735 rows × 9 columns

1.del 함수를 통해서 열 버리기

2.del 함수는 기존객체를 변경하므로 다시 할당해주는 과정을 거칠 필요가 없다.

B Deleting (Rows)

결측치가 있는 행 제거

```
In [8]: a = list(df.isnull().query('상태==True').index)

In [9]: df = df.drop(a,axis=0).reset_index(drop=True)

In [10]: df
```

Out[10]:

	연번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부
0	27319	2021-02-20	광진구	감염경로 조사중	사망	NaN	2021-02-24 10:18	2021-02-24 10:19	Y
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y
...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y

24183 rows × 9 columns

1.상태 열에 결측치가 있는 행을 제거하고싶다.

2.수평축을 기준으로 drop을 하고 싶으므로 axis =0으로 설정

3.reset_index()를 하는 이유를 다시 상기하자.

간단한 메소드인 dropna대신 drop의 이해를 위해 조금 꼬아서 작성된 코드!

BITAMIN

Creating

D
Deleting

C
Creating

CV
Changing Values

S
Sorting

FM
Function Mapping

GA
Group Aggragation

M
Merging

A
Appending

B Creating (Columns)

열 생성하기

```
df['0']= 0
```

```
df
```

	연번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부	0
0	27319	2021-02-20	광진구	감염경로 조사중	사망	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	0
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	0
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	0
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	0
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	0
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	0
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	0
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	0
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	0
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	0

```
df = df.drop(['0'],axis=1)
```

1. DataFrame객체['추가하려는 열의 이름']= 추가하려는 열의 value

2. value를 하나 할당한다면 모든 행에 동일한 값이 입력된다.

3. 각 행마다 다른 값을 넣고 싶다면 리스트를 활용해 입력하면 된다.

B Creating (Columns)

열 생성하기 – using other columns

```
df['영등포구']= df.지역.isin(['영등포구'])
df
```

	연번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부	영등포구
0	27319	2021-02-20	광진구	감염경로 조사중	사망	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False

24183 rows × 10 columns

1.지역이 영등포구인지 아닌지 확인하기위한 열을 생성하자.

2.지역 column에 영등포구의 유무에 따라 bool자료형으로 출력되도록 영등포구 column을 생성하자.

BITAMIN

Changing Values

D Deleting

C Creating

CV Changing Values

S Sorting

FM Function Mapping

GA Group Aggragation

M Merging

A Appending

B Changing Values

값 변경하기(리스트)

```
In [19]:  a = [1,2,3,4,5]
          a[2] = '중간'
          a

Out[19]: [1, 2, '중간', 4, 5]
```

값 변경하기(데이터프레임)

```
In [21]:  df.loc[[0,1],'이동경로']='미확인'
          #df.iloc[[0,1],5]='미확인'
          #df['이동경로'][[0,1]]='미확인'
          df
```

Out[21]:

	연번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부	영등포구
0	27319	2021-02-20	광진구	감염경로 조사중	사망	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False

24183 rows × 10 columns

1. 리스트의 요소값 변경하기.
   핵심은 indexing ***

2. Indexing을 활용하여 value값을
   변경한다는 메커니즘을 인지하자.

3. 데이터프레임에는 다양한 indexing
   방법이 있다.
   (주석처리된 코드도 실행해보자)

B Changing Values(Rename)

Column명 변경하기(리스트 활용)

```
In [22]: col = list(df.columns)
         col[0] = '순번'
         col # 영어로 바꾸는 경우는 많음

Out[22]: ['순번', '확진일', '지역', '접촉력', '상태', '이동경로', '등록일', '수정일', '노출여부', '영등포구']

In [23]: df.columns = col# 하나만 바꾸고 싶은데 다 써야해?

In [24]: df
```

Out[24]:

	순번	확진일	지역	접촉력	상태	이동경로	등록일	수정일	노출여부	영등포구
0	27319	2021-02-20	광진구	감염경로 조사중	사망	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False

1.Column명 을 모두 변경하고 싶은 경우 – 주로 한글을 영어로 변경

2.Column명 일부만 변경하고 싶다면 해당 방식은 불편할 수 있다.

B Changing Values(Rename)

Column명 변경하기(rename 메소드 활용)

```
In [26]: df = df.rename(columns = {'지역': '거주구','접촉력': '감염경로'})

In [27]: df
```

Out[27]:

	순번	확진일	거주구	감염경로	상태	이동경로	등록일	수정일	노출여부	영등포구
0	27319	2021-02-20	광진구	감염경로 조사중	사망	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
1	27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
2	27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
3	27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
4	27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24179	4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24180	3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24181	2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
24182	1	2020-01-24	강서구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False

24183 rows × 10 columns

1.Column명 일부를 변경하고 싶다면 더 유용한 방법

2.Dictionary를 활용하기 때문

BITAMIN

Sorting

BITAmin

B Sorting(Columns)

Columns 순서 바꾸기

```
In [30]: list(df.columns)
Out[30]: ['순번', '확진일', '거주구', '감염경로', '상태', '이동경로', '등록일', '수정일', '노출여부', '영등포구']

In [31]: # indexing 이용
         #날짜들끼리 묶어보고싶음
         df = df[['순번', '확진일', '등록일', '수정일', '거주구', '감염경로', '상태', '이동경로', '노출여부', '영등포구']]

In [32]: df
```

Out[32]:

	순번	확진일	등록일	수정일	거주구	감염경로	상태	이동경로	노출여부	영등포구
0	27319	2021-02-20	2021-02-24 10:18	2021-02-24 10:19	광진구	감염경로 조사중	사망	미확인	Y	False
1	27269	2021-02-20	2021-02-24 10:18	2021-02-24 10:19	기타	용산구 소재 병원 관련	퇴원	미확인	Y	False
2	27226	2021-02-19	2021-02-24 10:18	2021-02-24 10:18	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	Y	False
3	27225	2021-02-19	2021-02-24 10:18	2021-02-24 10:18	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	Y	False
4	27200	2021-02-19	2021-02-24 10:18	2021-02-24 10:19	광진구	기타 확진자 접촉	퇴원	NaN	Y	False
...	...	...	...	...	...	...	...	...	...	...
24178	5	2020-01-31	2021-02-24 10:15	2021-02-24 10:15	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
24179	4	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	마포구	해외유입	퇴원	이동경로 공개기간 경과	Y	False
24180	3	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	Y	False
24181	2	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	중랑구	해외유입	퇴원	이동경로 공개기간 경과	Y	False
24182	1	2020-01-24	2021-02-24 10:15	2021-02-24 10:15	강서구	해외유입	퇴원	이동경로 공개기간 경과	Y	False

24183 rows × 10 columns

1. Column명들을 하나하나 칠건가?

2. Indexing을 통해서 새로운 DataFrame을 구성함으로써 column들의 순서를 바꿔주자

B Sorting(Values)

특정 열의 값들을 기준으로 DataFrame 정렬하기

```
In [70]: df.sort_values(by='순번')
```

Out[70]:

	순번	확진일	등록일	수정일	거주구	감염경로	상태	이동경로	노출여부	영등포구
24182	1	2020-01-24	2021-02-24 10:15	2021-02-24 10:15	강서구	해외유입	퇴원	이동경로 공개기간 경과	Y	False
24181	2	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	중랑구	해외유입	퇴원	이동경로 공개기간 경과	Y	False
24180	3	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	Y	False
24179	4	2020-01-30	2021-02-24 10:15	2021-02-24 10:15	마포구	해외유입	퇴원	이동경로 공개기간 경과	Y	False
24178	5	2020-01-31	2021-02-24 10:15	2021-02-24 10:15	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
...	...	...	...	...	...	...	...	...	...	...
4	27200	2021-02-19	2021-02-24 10:18	2021-02-24 10:19	광진구	기타 확진자 접촉	퇴원	NaN	Y	False
3	27225	2021-02-19	2021-02-24 10:18	2021-02-24 10:18	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	Y	False
2	27226	2021-02-19	2021-02-24 10:18	2021-02-24 10:18	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	Y	False
1	27269	2021-02-20	2021-02-24 10:18	2021-02-24 10:19	기타	용산구 소재 병원 관련	퇴원	미확인?	Y	False
0	27319	2021-02-20	2021-02-24 10:18	2021-02-24 10:19	광진구	감염경로 조사중	사망	미확인?	Y	False

```
In [71]: df.sort_values(by=['거주구','확진일'])#여러개의 칼럼을 기준으로 정렬
         # default는 오름차순 내림차순 정렬을 원한다면 ascending=False를 해주자.
```

Out[71]:

	순번	확진일	등록일	수정일	거주구	감염경로	상태	이동경로	노출여부	영등포구
24128	55	2020-02-26	2021-02-24 10:15	2021-02-24 10:15	강남구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
24129	54	2020-02-26	2021-02-24 10:15	2021-02-24 10:15	강남구	타시도 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
24118	65	2020-02-27	2021-02-24 10:15	2021-02-24 10:15	강남구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
24106	77	2020-02-28	2021-02-24 10:15	2021-02-24 10:15	강남구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
24107	76	2020-02-28	2021-02-24 10:15	2021-02-24 10:15	강남구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
...	...	...	...	...	...	...	...	...	...	...
757	24852	2021-02-04	2021-02-24 10:18	2021-02-24 10:18	타시도	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
650	25005	2021-02-05	2021-02-24 10:18	2021-02-24 10:18	타시도	감염경로 조사중	퇴원	이동경로 공개기간 경과	Y	False
691	24941	2021-02-05	2021-02-24 10:18	2021-02-24 10:18	타시도	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False
549	25168	2021-02-07	2021-02-24 10:18	2021-02-24 10:18	타시도	성동구 소재 병원 관련(?21.1월)	퇴원	이동경로 공개기간 경과	Y	False
466	25296	2021-02-08	2021-02-24 10:18	2021-02-24 10:18	타시도	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	Y	False

1. DataFrame객체.sort_values(by ='기준 열')

2. 특정 열의 값들의 sorting기준으로 데이터프레임이 정렬된다.

3. 여러 개의 기준열을 정하고 싶다면

4. 여러 개의 기준열을 사용할 경우 리스트에서 앞서 있는 칼럼을 기준으로 먼저 정렬된다.

5. Default는 오름차순 if 내림차순으로 정렬하고싶다면 ascending=False를 추가해주자.

B Reference

특정 열을 행 인덱스로 설정

```
In [28]: df.set_index(['순번']) # 참고 이런게 있다~
```

Out[28]:

순번	확진일	거주구	감염경로	상태	이동경로	등록일	수정일	노출여부	영등포구
27319	2021-02-20	광진구	감염경로 조사중	사망	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
27269	2021-02-20	기타	용산구 소재 병원 관련	퇴원	미확인	2021-02-24 10:18	2021-02-24 10:19	Y	False
27226	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
27225	2021-02-19	양천구	양천구 소재 유치원/어린이집 관련	퇴원	이동경로 공개기간 경과	2021-02-24 10:18	2021-02-24 10:18	Y	False
27200	2021-02-19	광진구	기타 확진자 접촉	퇴원	NaN	2021-02-24 10:18	2021-02-24 10:19	Y	False
...	...	...	...	...	...	...	...	...	...
5	2020-01-31	성북구	기타 확진자 접촉	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
4	2020-01-30	마포구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
3	2020-01-30	종로구	종로구 집단발병	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False
2	2020-01-30	중랑구	해외유입	퇴원	이동경로 공개기간 경과	2021-02-24 10:15	2021-02-24 10:15	Y	False

1.set_index([‘열 이름‘] or ‘열이름’)

2.특정 열을 행 인덱스로 설정

전치 - 행 인덱스와 칼럼 뒤집기

```
In [29]: df.T
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9	...	24173	24174	24175	24176	24177	24178	24179	24180	24181	24
순번	27319	27269	27226	27225	27200	27144	27107	27102	27034	26987	...	10	9	8	7	6	5	4	3	2	
확진일	2021-02-20	2021-02-20	2021-02-19	2021-02-19	2021-02-19	2021-02-19	2020-12-10	2021-02-18	2021-02-18	2021-02-18	...	2020-02-05	2020-02-05	2020-02-02	2020-01-31	2020-01-31	2020-01-31	2020-01-30	2020-01-30	2020-01-30	20
거주구	광진구	기타	양천구	양천구	광진구	양천구	강동구	성북구	송파구	용산구	...	성북구	송파구	타시도	종로구	종로구	성북구	마포구	종로구	중랑구	
감염경로	감염경로조사중	용산구소재병원관련	양천구소재유치원/어린이집관련	양천구소재유치원/어린이집관련	기타확진자접촉	타시도확진자접촉	기타확진자접촉	기타확진자접촉	감염경로조사중	감염경로조사중	...	종로구집단발병	해외유입	해외유입	종로구집단발병	종로구집단발병	기타확진자접촉	해외유입	종로구집단발병	해외유입	

1.T or transpose 메소드를 통해
행 인덱스와 칼럼을 뒤집을 수 있다.

BITAMIN

Function Mapping

D Deleting

C Creating

CV Changing Values

S Sorting

FM Function Mapping

GA Group Aggragation

M Merging

A Appending

B Function Mapping


↳ The process of one-to-one corresponding individual elements in a series or DataFrame to a particular function.

1. 사용자 지정 함수 적용 가능
2. value값들의 일괄적인 연산이 아닌 개별 요소별 연산이 가능

함수 매핑을 위한 메소드

	DataFrame	Series
apply	✓	✓
map		✓
applymap	✓	

B Function Mapping


특정 기능을 실행하고 싶다.

```
In [33]: if df['거주구']=='기타':
             df['거주구'] = '서울 외 지역'
         else:
             df['거주구'] = df['거주구']
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-33-e7a55c414931> in <module>
----> 1 if df['거주구']=='기타':
      2     df['거주구'] = '서울 외 지역'
      3 else:
      4     df['거주구'] = df['거주구']

~₩anaconda3₩lib₩site-packages₩pandas₩core₩generic.py in __nonzero__(self)
   1327
   1328     def __nonzero__(self):
-> 1329         raise ValueError(
   1330             f"The truth value of a {type(self).__name__} is ambiguous. "
   1331             "Use a.empty, a.bool(), a.item(), a.any() or a.all()."

ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().
```

거주구가 기타라면 서울 외 지역으로
아니라면 그대로 요소들을 바꾸고 싶다.

Why error?

B Function Mapping

Apply를 통한 mapping

```
In [110]: df['거주구'].apply(lambda x: '서울 외 지역'if x =='기타'else x)
          # 값들을 분석 목적에 부합하도록 원하는 방식으로 변경
```

```
Out[110]: 0           광진구
          1       서울 외 지역
          2           양천구
          3           양천구
          4           광진구
                     ...
          24178       성북구
          24179       마포구
          24180       종로구
          24181       중랑구
          24182       강서구
          Name: 거주구, Length: 24183, dtype: object
```

1.분석 목적에 맞게 값들을 원하는 방식으로 변경

2.함수를 활용하여 변경가능 function, lambda모두 사용가능

3.함수를 어떻게 구성하냐에 따라 아주 다양한 mapping가능

BITAMIN

Group Aggregation

D Deleting

C Creating

CV Changing Values

S Sorting

FM Function Mapping

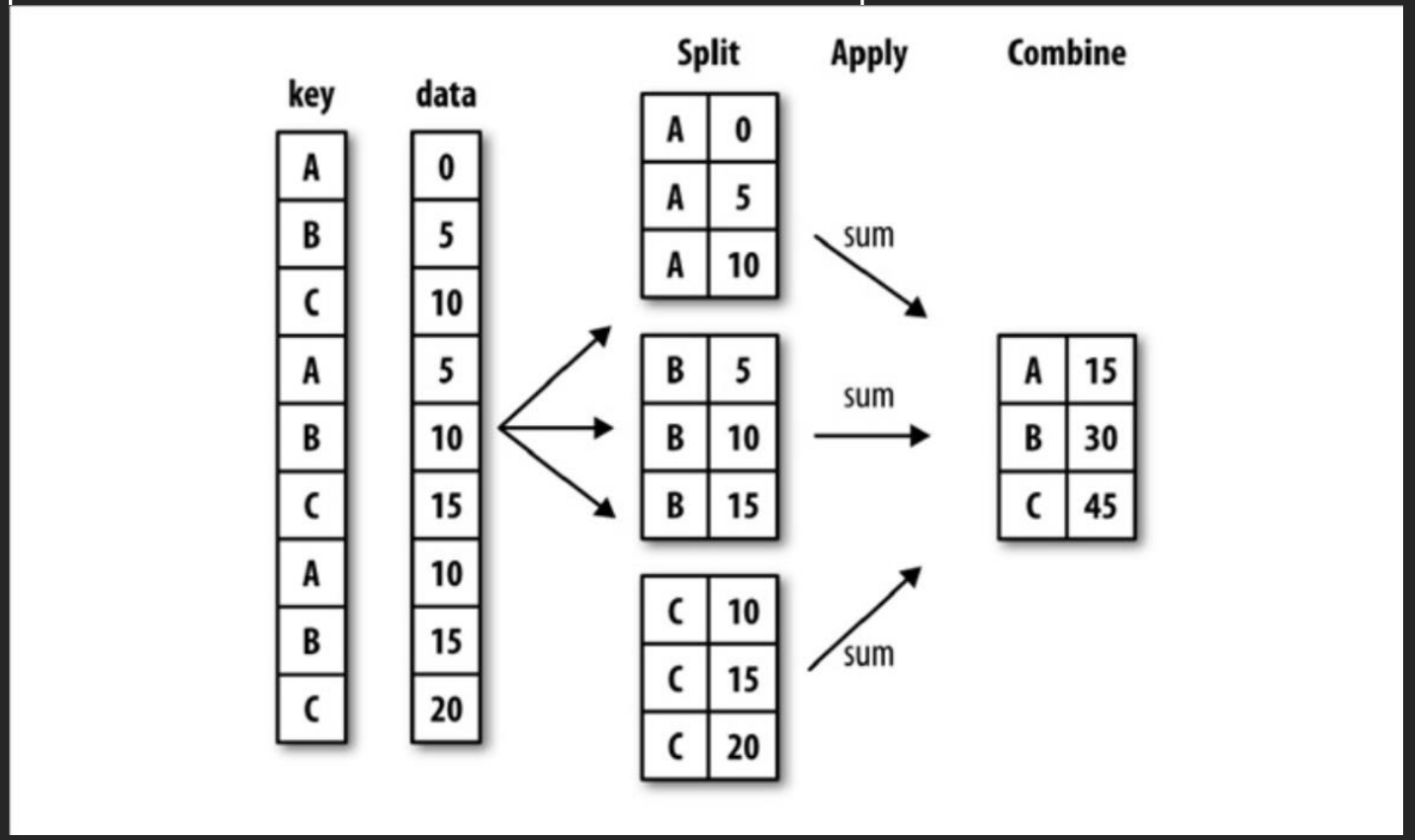
GA Group Aggragation

M Merging

A Appending

B Group Aggregation

groupby를 통한 group aggregation의 구조



기준에 따라
그룹화
⇩
함수를 통해
집계
⇩
그룹별 집계된
결과

B Group Aggregation

데이터프레임 생성

```
g = pd.DataFrame({'key'  : ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
                  'data' : [0, 5, 10, 5, 10, 15, 10, 15, 20]})
g
```

	key	data
0	A	0
1	B	5
2	C	10
3	A	5
4	B	10
5	C	15
6	A	10
7	B	15
8	C	20

기준에 따라 그룹화까지

```
grouped =g.groupby('key')['data']
for name, group in grouped:
    print(name)
    print(group)
```

```
A
0     0
3     5
6    10
Name: data, dtype: int64
B
1     5
4    10
7    15
Name: data, dtype: int64
C
2    10
5    15
8    20
Name: data, dtype: int64
```

```
grouped.get_group('A')
```

```
0     0
3     5
6    10
Name: data, dtype: int64
```

1.그룹화까지의 과정을 시각화

2.get_group()을 이용해서 원하는 그룹의 데이터 추출

B Group Aggregation

전형적인 사용 – key(기준)별 데이터 집계

```
In [44]: g.groupby('key')['data'].sum()

Out[44]: key
         A    15
         B    30
         C    45
         Name: data, dtype: int64

In [45]: g.groupby('key')['data'].size()

Out[45]: key
         A    3
         B    3
         C    3
         Name: data, dtype: int64
```

1.내장 함수 직접적으로 사용

전형적인 사용 – key(기준)별 데이터 집계 + reset_index()의 사용

```
In [47]: g.groupby('key')['data'].sum().reset_index()

Out[47]:
```

	key	data
0	A	15
1	B	30
2	C	45

2.reset_index()의 역할

B Group Aggregation

Using agg method

```
g.groupby('key')['data'].agg('max')
```

```
key
A    10
B    15
C    20
Name: data, dtype: int64
```

```
g.groupby('key')['data'].agg(['max', 'min'])
```

key	max	min
A	10	0
B	15	5
C	20	10

```
g.groupby('key')['data'].agg(lambda x: x.max() - x.min())
```

```
key
A    10
B    10
C    10
Name: data, dtype: int64
```

1.사용자 지정함수를 통해 집계를 하고 싶을 때

2.agg(내가 사용하고싶은 함수)

3.여러 개의 집계함수를 사용하고 싶을 때

B Group Aggregation

df를 활용해 다시 Group Aggregation 진행
특정 기준열에 대해 특정 열의 데이터를 특정 방식으로 집계하고 싶을 때

```
In [53]: df.groupby('확진일')['영등포구'].sum().reset_index()
```

Out[53]:

	확진일	영등포구
0	2020-01-24	0
1	2020-01-30	0
2	2020-01-31	0
3	2020-02-02	0
4	2020-02-05	0
...	...	...
357	2021-02-16	2
358	2021-02-17	0
359	2021-02-18	0
360	2021-02-19	0
361	2021-02-20	0

362 rows × 2 columns

확진일 별로 영등포구에 확진자가 몇 명이 발생했는지 확인하고 싶다!

B Group Aggregation

집계된 데이터의 column명을 바꾸고싶다.

```
df.groupby('확진일')['영등포구'].agg([('영등포구일일확진자수',np.sum)]).reset_index()
```

	확진일	영등포구일일확진자수
0	2020-01-24	0
1	2020-01-30	0
2	2020-01-31	0
3	2020-02-02	0
4	2020-02-05	0
...	...	...
357	2021-02-16	2
358	2021-02-17	0
359	2021-02-18	0
360	2021-02-19	0
361	2021-02-20	0

362 rows × 2 columns

1.집계된 column의 이름을 설정하고싶다.

2. agg 메소드와 함께 리스트에 튜플로 바꿀 column명과 적용할 함수를 넣어주자.

〈사용법〉

DataFrame.groupby('기준열')['집계할 데이터열'].agg([('집계된 데이터의 새로운 column명',적용할 함수)])

B Group Aggregation

특정 기준열에 대해 특정 열의 데이터를 여러 방식으로 집계

```
df.groupby('거주구').agg({'확진일':[('최초 확진일',np.min),('가장최근 확진자 발생일',np.max)]}).reset_index()
```

	거주구	확진일	
		최초 확진일	가장최근 확진자 발생일
0	강남구	2020-02-26	2021-02-11
1	강동구	2020-02-22	2021-02-14
2	강북구	2020-03-04	2021-02-13
3	강서구	2020-01-24	2021-02-17
4	관악구	2020-02-25	2021-02-17
5	광진구	2020-02-27	2021-02-20
6	구로구	2020-02-22	2021-02-17
7	금천구	2020-02-25	2021-02-16
8	기타	2020-03-27	2021-02-20
9	노원구	2020-02-25	2021-02-13
10	도봉구	2020-03-02	2021-02-11
11	동대문구	2020-02-27	2021-02-12

거주구 별로 최초 확진일과
가장 최근 확진자 발생일

* 멀티 인덱스?

〈사용법〉

DataFrame.groupby('기준열').agg({'집계할 특정 열' : [('첫번째 열의 함수1로 집계된 새로운 column명',적용할 함수1),
('첫번째 열의 함수2로 집계된 새로운 column명',적용할 함수2)]})

B Group Aggregation

특정 기준열에 대해 여러 열의 데이터를 여러 방식으로 집계하고 싶을 때

```
df.groupby('거주구').agg({'확진일':[('최초 확진일',np.min)],
                        '감염경로':[('감염경로의 다양성',lambda x: x.nunique())]}).reset_index()
```

	거주구	확진일	감염경로
		최초 확진일	감염경로의 다양성
0	강남구	2020-02-26	105
1	강동구	2020-02-22	68
2	강북구	2020-03-04	83
3	강서구	2020-01-24	108
4	관악구	2020-02-25	127
5	광진구	2020-02-27	69
6	구로구	2020-02-22	83
7	금천구	2020-02-25	60
8	기타	2020-03-27	48
9	노원구	2020-02-25	98

거주구별 최조 확진일과
거주구별 감염경로의 다양성

〈사용법〉

DataFrame.groupby(‘기준열’).agg({‘집계할 첫번째 열’ : [(‘첫번째 열의 함수1로 집계된 새로운 column명‘,첫번째 열에 적용할 함수1)],
‘집계할 두번째 열‘ : [(‘두번째 열의 함수1로 집계된 새로운 column명‘,두번째 열에 적용할 함수1)]})

B Group Aggregation

특정 기준열에 대해 여러 열의 데이터를 여러 방식으로 집계하고 싶을 때

```
In [57]: df.groupby('거주구').agg({'확진일':[('최초 확진일',np.min),('가장최근 확진자 발생일',np.max)],
                                  '감염경로':[('감염경로의 다양성',lambda x: x.nunique())]}).reset_index()
```

Out[57]:

	거주구	확진일		감염경로
		최초 확진일	가장최근 확진자 발생일	감염경로의 다양성
0	강남구	2020-02-26	2021-02-11	105
1	강동구	2020-02-22	2021-02-14	68
2	강북구	2020-03-04	2021-02-13	83
3	강서구	2020-01-24	2021-02-17	108
4	관악구	2020-02-25	2021-02-17	127
5	광진구	2020-02-27	2021-02-20	69
6	구로구	2020-02-22	2021-02-17	83
7	금천구	2020-02-25	2021-02-16	60
8	기타	2020-03-27	2021-02-20	48
9	노원구	2020-02-25	2021-02-13	98
10	도봉구	2020-03-02	2021-02-11	76
11	동대문구	2020-02-27	2021-02-12	84
12	동작구	2020-02-25	2021-02-18	105
13	마포구	2020-01-30	2021-02-11	84
14	서대문구	2020-02-06	2021-02-16	73
15	서초구	2020-02-21	2021-02-14	92
16	성동구	2020-02-19	2021-02-13	69
17	성북구	2020-01-31	2021-02-18	103
18	송파구	2020-02-05	2021-02-18	108

앞선 방식들을 종합적으로 적용

B Group Aggregation

여러 기준열에 대해 특정 열의 데이터를 특정 방식으로 집계

```
df.groupby(['거주구','확진일'])['감염경로'].agg([('감염경로의 다양성',lambda x: x.nunique())]).reset_index()
```

	거주구	확진일	감염경로의 다양성
0	강남구	2020-02-26	2
1	강남구	2020-02-27	1
2	강남구	2020-02-28	1
3	강남구	2020-02-29	1
4	강남구	2020-03-01	1
...	...	...	...
5073	타시도	2021-02-03	2
5074	타시도	2021-02-04	3
5075	타시도	2021-02-05	2
5076	타시도	2021-02-07	1
5077	타시도	2021-02-08	1

거주구별로 확진일별로
감염경로의 다양성

〈사용법〉

DataFrame.groupby([‘기준열1’,‘기준열2’])[‘집계할 데이터열‘].agg([(‘집계된 데이터의 새로운 column명‘,적용할 함수)])

B PivotTable

피벗테이블 작성

```
#피벗테이블
pd.pivot_table(df, index = '확진일',
               columns='거주구',
               values='감염경로',
               aggfunc = Series.nunique)
```

거주구	강남구	강동구	강북구	강서구	관악구	광진구	구로구	금천구	기타	노원구	...	성북구	송파구	양천구	영등포구	용산구	은평구	종로구	중구	중랑구	타시도
확진일																					
2020-01-24	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1.0	NaN
2020-01-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN
2020-02-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0
2020-02-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-02-16	NaN	NaN	NaN	NaN	NaN	NaN	2.0	1.0	NaN	NaN	...	NaN	1.0	NaN	1.0	2.0	1.0	NaN	NaN	NaN	NaN
2021-02-17	NaN	NaN	NaN	1.0	1.0	NaN	1.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-02-18	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	...	1.0	1.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN
2021-02-19	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	...	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-02-20	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

362 rows × 27 columns

pd.pivot_table(데이터프레임,
index = 행으로 사용할 열,
columns = 열로 사용할 열,
values = 데이터로 사용할 열,
aggfunc = 데이터 집계 함수)

기준열을 2개를 사용해 집계되는 groupby와 같은 의미

다만, pivot_table은 기준을 행과 열로 사용한다.

BITAMIN

Merging DataFrames

B Merging DataFrames

데이터프레임 생성

```
from pandas import DataFrame, Series

df1 = DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
                 'data1': range(7)})
df2 = DataFrame({'key': ['a', 'b', 'd'], 'data2': range(3)})

display(df1)
display(df2)
```

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

	key	data2
0	a	0
1	b	1
2	d	2

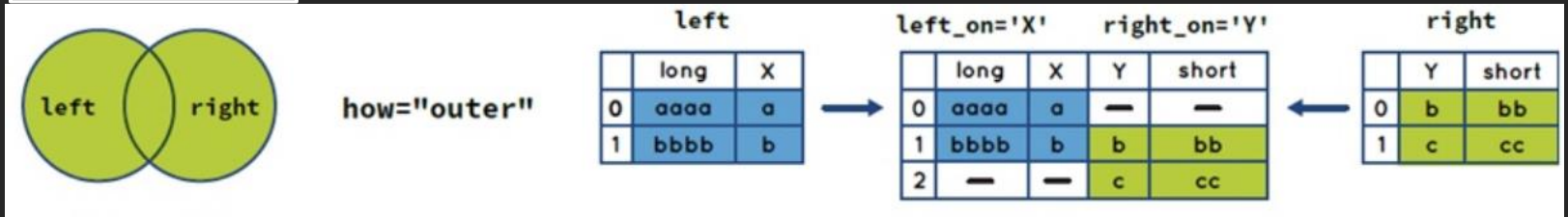
병합 merge
pd.merge(데이터프레임1, 데이터프레임2)

기본 설정 (default)
on = None (공통인 모든 열을 키로 인식)
how = 'inner' (기준 열의 데이터가 양쪽에 공통으로 존재할 때만 추출)

추가 설정
on = '특정 열'
(특정 열을 기준으로 병합되며 공통되는 이름의 열 존재시
B_x B_y 이름으로 자동 변경)
how = 'outer','inner', 'left', 'right'

B Merging DataFrames

how ='outer'



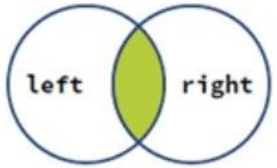

```
In [150]: pd.merge(df1, df2,how='outer')
Out[150]:
```

	key	data1	data2
0	b	0.0	1.0
1	b	1.0	1.0
2	b	6.0	1.0
3	a	2.0	0.0
4	a	4.0	0.0
5	a	5.0	0.0
6	c	3.0	NaN
7	d	NaN	2.0

기준이 되는 열의 데이터가 어느 한쪽에만 속하더라도 포함

B Merging DataFrames

how ='inner'



how="inner"

	long	X
0	aaaa	a
1	bbbb	b

	long	X	Y	short
0	bbbb	b	b	bb

	Y	short
0	b	bb
1	c	cc

```
In [149]: pd.merge(df1, df2,how='inner')
```

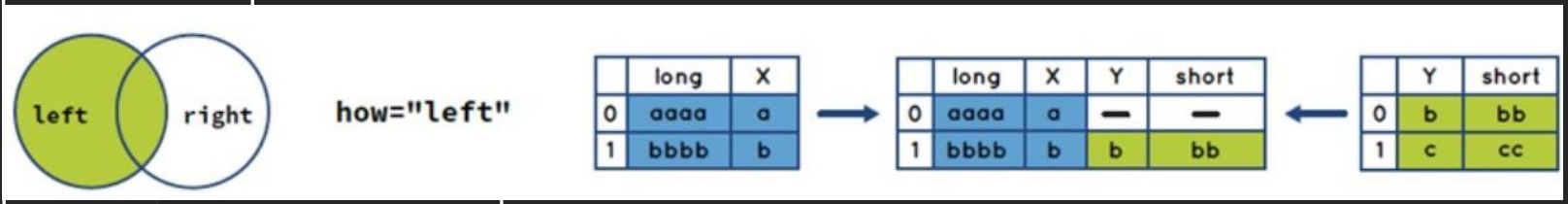
Out[149]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

기준열의 데이터가 양쪽에 공통으로 존재하는 경우만 포함

B Merging DataFrames

how ='left'




```
In [151]: pd.merge(df1, df2,how='left')
```

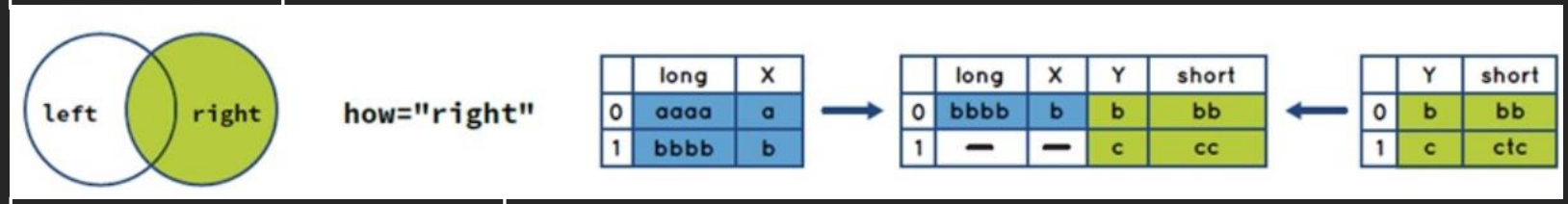
Out[151]:

	key	data1	data2
0	b	0	1.0
1	b	1	1.0
2	a	2	0.0
3	c	3	NaN
4	a	4	0.0
5	a	5	0.0
6	b	6	1.0

기준열의 데이터를 왼쪽에 위치한
데이터프레임을 기준으로 병합

B Merging DataFrames

how ='right'




```
In [152]: pd.merge(df1, df2,how='right')
```

Out[152]:

	key	data1	data2
0	a	2.0	0
1	a	4.0	0
2	a	5.0	0
3	b	0.0	1
4	b	1.0	1
5	b	6.0	1
6	d	NaN	2

기준열의 데이터를 오른쪽에 위치
한 데이터프레임을 기준으로 병합

BITAMIN

Appending DataFrames

D Deleting

C Creating

CV Changing Values

S Sorting

FM Function Mapping

GA Group Aggragation

M Merging

A Appending

B Appending DataFrames

데이터 준비

```
In [76]: df_1 = pd.read_csv('train.csv',encoding='cp949')
         df_1
```

Out[76]:

	DateTime	사용자	세션	신규방문자	페이지뷰
0	2018-09-09 00:00:00	19	19	8	206
1	2018-09-09 01:00:00	20	19	9	259
2	2018-09-09 02:00:00	12	9	1	48
3	2018-09-09 03:00:00	10	10	2	102
4	2018-09-09 04:00:00	6	5	3	18
...	...	...	...	...	...
19003	2020-11-08 19:00:00	124	123	19	3128
19004	2020-11-08 20:00:00	166	159	29	4864
19005	2020-11-08 21:00:00	184	173	32	3426
19006	2020-11-08 22:00:00	163	155	34	2845
19007	2020-11-08 23:00:00	160	152	33	3293

19008 rows × 5 columns

```
In [77]: df_2 = pd.read_csv('2차_train.csv',encoding='cp949')
         df_2
```

Out[77]:

	DateTime	사용자	세션	신규방문자	페이지뷰
0	2020-11-09 00:00:00	134	147	33	4082
1	2020-11-09 01:00:00	88	77	22	2444
2	2020-11-09 02:00:00	54	48	12	1148
3	2020-11-09 03:00:00	35	31	3	557
4	2020-11-09 04:00:00	31	34	7	711
...	...	...	...	...	...
715	2020-12-08 19:00:00	118	121	27	3337
716	2020-12-08 20:00:00	182	184	58	4600
717	2020-12-08 21:00:00	169	159	49	3983
718	2020-12-08 22:00:00	158	167	44	4578
719	2020-12-08 23:00:00	151	158	38	3097

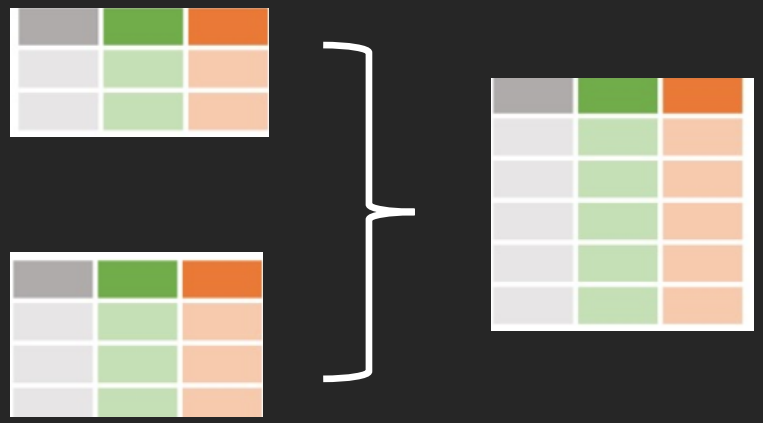
720 rows × 5 columns

pd.concat([데이터프레임1, 데이터프레임2])

데이터를 그냥 붙인다고 생각하면 편하다!

B Appending DataFrames

axis =0(default)



pd.concat([데이터프레임1, 데이터프레임2])

위아래로 concat

B Appending DataFrames

axis =0(default) 실습

```
In [179]: pd.concat([df_1,df_2],axis=0)
```

Out[179]:

	DateTime	사용자	세션	신규방문자	페이지뷰
0	2018-09-09 00:00:00	19	19	8	206
1	2018-09-09 01:00:00	20	19	9	259
2	2018-09-09 02:00:00	12	9	1	48
3	2018-09-09 03:00:00	10	10	2	102
4	2018-09-09 04:00:00	6	5	3	18
...	...	...	...	...	...
715	2020-12-08 19:00:00	118	121	27	3337
716	2020-12-08 20:00:00	182	184	58	4600
717	2020-12-08 21:00:00	169	159	49	3983
718	2020-12-08 22:00:00	158	167	44	4578
719	2020-12-08 23:00:00	151	158	38	3097

19728 rows × 5 columns

B Appending DataFrames

axis =1

pd.concat([데이터프레임1, 데이터프레임2],axis =1)

양옆으로 concat

B Appending DataFrames

axis =1 실습

```
In [78]: df_c = pd.concat([df_1,df_2],axis=0)
         df_1 = df_c[['DateTime','사용자','세션']]
         df_2 = df_c[['DateTime','신규방문자','페이지뷰']]

In [73]: pd.concat([df_1,df_2],axis=1)
```

Out[73]:

	DateTime	사용자	세션	DateTime	신규방문자	페이지뷰
0	2018-09-09 00:00:00	19	19	2018-09-09 00:00:00	8	206
1	2018-09-09 01:00:00	20	19	2018-09-09 01:00:00	9	259
2	2018-09-09 02:00:00	12	9	2018-09-09 02:00:00	1	48
3	2018-09-09 03:00:00	10	10	2018-09-09 03:00:00	2	102
4	2018-09-09 04:00:00	6	5	2018-09-09 04:00:00	3	18
...	...	...	...	...	...	...
715	2020-12-08 19:00:00	118	121	2020-12-08 19:00:00	27	3337
716	2020-12-08 20:00:00	182	184	2020-12-08 20:00:00	58	4600
717	2020-12-08 21:00:00	169	159	2020-12-08 21:00:00	49	3983
718	2020-12-08 22:00:00	158	167	2020-12-08 22:00:00	44	4578
719	2020-12-08 23:00:00	151	158	2020-12-08 23:00:00	38	3097

19728 rows × 6 columns

Q & A

수고하셨습니다