1. 필요한 모듈 불러오기

In [2]:
```
import pandas as pd
import numpy as np
```

2. Case, PatientInfo, PatientRoute csv 파일 불러오기

Case.csv는 case / PatientInfo.csv는 patient_info / PatientRoute.csv는 patient_route로 저장해주세요

In [3]:
```
case = pd.read_csv('COVID19/Case.csv')
patient_info = pd.read_csv('COVID19/PatientInfo.csv')
patient_route = pd.read_csv('COVID19/PatientRoute.csv')
```

3. 각 파일의 행과 열의 개수를 구해주는 함수를 아래와 같이 만들어주세요

In [2]:
```
# 아래 셀에서 작성해주세요
check_shape(case)
check_shape(patient_info)
check_shape(patient_route)
```

```
해당 데이터는 81행 8 열입니다
해당 데이터는 2243행 18 열입니다
해당 데이터는 175행 7 열입니다
```

In [3]:
```
def check_shape(data):
    print('해당 데이터는', len(data), '행 ', len(data.columns),'열 입니다.')

check_shape(case)
check_shape(patient_info)
check_shape(patient_route)
```

```
해당 데이터는 81 행  8 열 입니다.
해당 데이터는 2243 행  18 열 입니다.
해당 데이터는 175 행  7 열 입니다.
```

In [4]:
```
# 모범답안
# %s -> 'abc'

def check_shape(data):
    print('해당 데이터는 %d행 %d열 입니다.' %(data.shape[0], data.shape[1]))

check_shape(case)
check_shape(patient_info)
check_shape(patient_route)
```

```
해당 데이터는 81행 8열 입니다.
해당 데이터는 2243행 18열 입니다.
해당 데이터는 175행 7열 입니다.
```

4. patient_info에서 birth_year를 이용하며 age column에 정확한 나이 숫자로 바꿔주세요

In [4]:
```
patient_info.head()
```

Out[4]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	infection_cas
0	1000000001	2.0	male	1964.0	50s	Korea	Seoul	Gangseo-gu	NaN	oversea inflo
1	1000000002	5.0	male	1987.0	30s	Korea	Seoul	Jungnang-gu	NaN	oversea inflo
2	1000000003	6.0	male	1964.0	50s	Korea	Seoul	Jongno-gu	NaN	contact wi patie
3	1000000004	7.0	male	1991.0	20s	Korea	Seoul	Mapo-gu	NaN	oversea inflo
4	1000000005	9.0	female	1992.0	20s	Korea	Seoul	Seongbuk-gu	NaN	contact wi patie

In [6]:
```
# 모범답안

patient_info["age"] = 2020 - patient_info["birth_year"] + 1
patient_info.head()
```

Out[6]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	infection_cas
0	1000000001	2.0	male	1964.0	57.0	Korea	Seoul	Gangseo-gu	NaN	overse inflc
1	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	overse inflc
2	1000000003	6.0	male	1964.0	57.0	Korea	Seoul	Jongno-gu	NaN	contact w patie
3	1000000004	7.0	male	1991.0	30.0	Korea	Seoul	Mapo-gu	NaN	overse inflc
4	1000000005	9.0	female	1992.0	29.0	Korea	Seoul	Seongbuk-gu	NaN	contact w patie

5. 서울과 제주도의 확진환자 성별 평균 나이를 구해주세요

In [3]:
```
# 아래 셀에서 작성해주세요
seoul_jeju_mean_age
```

Out[3]:

		age
province	sex	
Jeju-do	female	36.000000
	male	35.000000
Seoul	female	45.400000
	male	43.090278

In [9]:
```
seoul_jeju_mean_age = patient_info.pivot_table(index = ['province', 'sex'], agg
seoul_jeju_mean_age
```

Out[9]:

		age
province	sex	
Busan	female	44.944444
	male	39.680851
Chungcheongbuk-do	female	61.000000
	male	40.250000
Chungcheongnam-do	female	42.740741
	male	35.189189
Daegu	female	53.054054
	male	62.346154
Gangwon-do	female	52.571429
	male	58.200000
	female	45.726708

In [10]:
```
seoul_jeju_mean_age.loc[['Jeju-do', 'Seoul']]
```

Out[10]:

		age
province	sex	
Jeju-do	female	36.000000
	male	35.000000
Seoul	female	45.400000
	male	43.090278

In [7]:
```
# 모범답안

temp = pd.DataFrame(patient_info.groupby(['province', 'sex'])['age'].mean())
temp.loc[['Jeju-do', 'Seoul']]
```

Out[7]:

		age
province	sex	
Jeju-do	female	36.000000
	male	35.000000
Seoul	female	45.400000
	male	43.090278

6. patient_info에서 접촉자수(contact_number)와 나이(age)가 NaN인 값은 지우기

In [12]:
```
# 결측값 있는 행, 열 제거하기
# Delete row with NaN : df.dropna(axis = 0)
# Delete column with NaN : df.dropna(axis = 1)

patient_info.dropna(subset = ['contact_number', 'age'])
patient_info
```

Out[12]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease
0	1000000001	2.0	male	1964.0	57.0	Korea	Seoul	Gangseo-gu	NaN
1	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN
2	1000000003	6.0	male	1964.0	57.0	Korea	Seoul	Jongno-gu	NaN
3	1000000004	7.0	male	1991.0	30.0	Korea	Seoul	Mapo-gu	NaN
4	1000000005	9.0	female	1992.0	29.0	Korea	Seoul	Seongbuk-gu	NaN
...	...	...	...	...	...	...	...	...	...
2238	6100000085	NaN	male	1990.0	31.0	Korea	Gyeongsangnam-do	Changwon-si	NaN
2239	7000000001	139.0	male	1998.0	23.0	Korea	Jeju-do	Jeju-do	NaN
2240	7000000002	222.0	female	1998.0	23.0	Korea	Jeju-do	Jeju-do	NaN
2241	7000000003	4345.0	female	1972.0	49.0	Korea	Jeju-do	etc	NaN
2242	7000000004	5534.0	male	1974.0	47.0	Korea	Jeju-do	Jeju-do	NaN

2243 rows × 18 columns

In [10]:
```
# 모범답안

# patient_info.head(10)
# True이면 출력
patient_info_2 = patient_info[patient_info['contact_number'].notnull() & patier
patient_info_2
```

Out[10]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	ir
0	1000000001	2.0	male	1964.0	57.0	Korea	Seoul	Gangseo-gu	NaN	
1	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	
2	1000000003	6.0	male	1964.0	57.0	Korea	Seoul	Jongno-gu	NaN	
3	1000000004	7.0	male	1991.0	30.0	Korea	Seoul	Mapo-gu	NaN	
4	1000000005	9.0	female	1992.0	29.0	Korea	Seoul	Seongbuk-gu	NaN	
...	...	...	...	...	...	...	...	...	...	
1897	6014000002	NaN	male	1998.0	23.0	Korea	Gyeongsangbuk-do	Yeongju-si	NaN	
2239	7000000001	139.0	male	1998.0	23.0	Korea	Jeju-do	Jeju-do	NaN	
2240	7000000002	222.0	female	1998.0	23.0	Korea	Jeju-do	Jeju-do	NaN	
2241	7000000003	4345.0	female	1972.0	49.0	Korea	Jeju-do	etc	NaN	
2242	7000000004	5534.0	male	1974.0	47.0	Korea	Jeju-do	Jeju-do	NaN	

322 rows × 18 columns

In [8]:
```
patient_info['contact_number'].notnull()
```

Out[8]:
```
0         True
1         True
2         True
3         True
4         True
         ...
2238     False
2239      True
2240      True
2241      True
2242      True
Name: contact_number, Length: 2243, dtype: bool
```

7. 위에서 NaN값을 제거한 데이터프레임에서 province와 city를 기준으로 contact_number와 age의 평균을 구해주세요

In [13]:
```
patient_avg = patient_info.pivot_table(index = ['province', 'city'], aggfunc =
patient_avg
```

Out[13]:

province	city	age	contact_number
	Buk-gu	32.000000	12.000000
	Busanjin-gu	56.727273	16.142857
Busan	Dongnae-gu	37.172414	46.586207
	Gangseo-gu	26.666667	14.000000
	Geumjeong-gu	32.500000	15.250000
...	...	...	...
	Buk-gu	36.200000	NaN
	Dong-gu	55.800000	NaN
Ulsan	Jung-gu	35.666667	NaN
	Nam-gu	44.333333	NaN
	Ulju-gun	26.500000	NaN

139 rows × 2 columns

In [12]:
```
# 모범답안

patient_group = patient_info_2.groupby(['province', 'city'])['contact_number',
patient_group
```

Out[12]:

province	city	contact_number	age
	Buk-gu	12.000000	32.000000
	Busanjin-gu	16.142857	51.000000
	Dongnae-gu	46.586207	37.172414
	Gangseo-gu	14.000000	26.666667
	Geumjeong-gu	15.250000	32.500000
	Haeundae-gu	24.437500	43.062500
Busan	Nam-gu	148.500000	50.000000
	Saha-gu	8.000000	38.714286
	Sasang-gu	22.500000	42.000000
	Seo-gu	12.833333	49.333333

8. 7번의 결과 데이터 프레임에서 각 province별 평균 감염자수와 평균 나이를 column에 추가하기

In [4]:
```
# 아래 셀에서 작성해주세요
patient_group
```

Out[4]:

province	city	contact_number	age	province_mean_contact_number	province_mean
Busan	Buk-gu	12.000000	32.000000	26.038454	43.6
	Busanjin-gu	16.142857	51.000000	26.038454	43.6
	Dongnae-gu	46.586207	37.172414	26.038454	43.6
	Gangseo-gu	14.000000	26.666667	26.038454	43.6
	Geumjeong-gu	15.250000	32.500000	26.038454	43.6
	Haeundae-gu	24.437500	43.062500	26.038454	43.6
	Nam-gu	148.500000	50.000000	26.038454	43.6
	Saha-gu	8.000000	38.714286	26.038454	43.6
	Sasang-gu	22.500000	42.000000	26.038454	43.6
	Seo-gu	12.833333	49.333333	26.038454	43.6

In [13]:
```
# 모범답안1 : merge

patient_group.index
```

Out[13]:
```
MultiIndex([(             'Busan',         'Buk-gu'),
            (             'Busan',    'Busanjin-gu'),
            (             'Busan',     'Dongnae-gu'),
            (             'Busan',     'Gangseo-gu'),
            (             'Busan',  'Geumjeong-gu'),
            (             'Busan',    'Haeundae-gu'),
            (             'Busan',         'Nam-gu'),
            (             'Busan',        'Saha-gu'),
            (             'Busan',      'Sasang-gu'),
            (             'Busan',         'Seo-gu'),
            (             'Busan',     'Suyeong-gu'),
            (             'Busan',      'Yeonje-gu'),
            (             'Busan',            'etc'),
            ('Chungcheongnam-do',        'Asan-si'),
            ('Chungcheongnam-do',     'Cheonan-si'),
            ('Chungcheongnam-do',    'Gyeryong-si'),
            ('Chungcheongnam-do',  'Hongseong-gun'),
            ('Chungcheongnam-do',      'Seosan-si'),
            (       'Gangwon-do',        'Wonju-si'),
```

In [18]:
```
[x for x in patient_group.index]
```

Out[18]:
```
[('Busan', 'Buk-gu'),
 ('Busan', 'Busanjin-gu'),
 ('Busan', 'Dongnae-gu'),
 ('Busan', 'Gangseo-gu'),
 ('Busan', 'Geumjeong-gu'),
 ('Busan', 'Haeundae-gu'),
 ('Busan', 'Nam-gu'),
 ('Busan', 'Saha-gu'),
 ('Busan', 'Sasang-gu'),
 ('Busan', 'Seo-gu'),
 ('Busan', 'Suyeong-gu'),
 ('Busan', 'Yeonje-gu'),
 ('Busan', 'etc'),
 ('Chungcheongnam-do', 'Asan-si'),
 ('Chungcheongnam-do', 'Cheonan-si'),
 ('Chungcheongnam-do', 'Gyeryong-si'),
 ('Chungcheongnam-do', 'Hongseong-gun'),
 ('Chungcheongnam-do', 'Seosan-si'),
 ('Gangwon-do', 'Wonju-si'),
```

In [22]:
```
[x[0] for x in patient_group.index]
```

Out[22]:
```
['Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Busan',
 'Chungcheongnam-do',
 'Chungcheongnam-do',
 'Chungcheongnam-do',
 'Chungcheongnam-do',
 'Chungcheongnam-do',
 'Gangwon-do',
```

In [23]:
```
dic = {}

for i in [x[0] for x in patient_group.index]:
    if i in dic.keys():
        dic[i] += 1

    else:
        dic[i] = 1

dic
```

Out[23]:
```
{'Busan': 13,
 'Chungcheongnam-do': 5,
 'Gangwon-do': 1,
 'Gyeonggi-do': 7,
 'Gyeongsangbuk-do': 2,
 'Jeju-do': 2,
 'Jeollabuk-do': 1,
 'Jeollanam-do': 3,
 'Sejong': 1,
 'Seoul': 9}
```

In [26]:
```
temp = patient_info_2.groupby(['province'])['contact_number'].mean()
temp
```

Out[26]:
```
province
Busan                  33.064516
Chungcheongnam-do      12.155963
Daegu                1160.000000
Gangwon-do             10.875000
Gyeonggi-do            39.037037
Gyeongsangbuk-do        3.932203
Jeju-do                66.500000
Jeollabuk-do          113.000000
Jeollanam-do           10.250000
Sejong                 40.000000
Seoul                  28.400000
Name: contact_number, dtype: float64
```

In [28]:
```
patient_info_2[patient_info_2.province == 'Daegu']
```

Out[28]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	infection_case
400	1200000031	31.0	female	1959.0	62.0	Korea	Daegu	NaN	NaN	Shincheonji Church

In [27]:
```
temp.drop(['Daegu'], axis = 0, inplace = True)
temp
```

Out[27]:
```
province
Busan                  33.064516
Chungcheongnam-do      12.155963
Gangwon-do             10.875000
Gyeonggi-do            39.037037
Gyeongsangbuk-do        3.932203
Jeju-do                66.500000
Jeollabuk-do          113.000000
Jeollanam-do           10.250000
Sejong                 40.000000
Seoul                  28.400000
Name: contact_number, dtype: float64
```

In [29]:
```
np.repeat(['a', 'b', 'c'], [1, 2, 3])
```

Out[29]:
```
array(['a', 'b', 'b', 'c', 'c', 'c'], dtype='<U1')
```

In [30]:
```
temp.values
```

Out[30]:
```
array([ 33.06451613,  12.1559633 ,  10.875     ,  39.03703704,
         3.93220339,  66.5       , 113.        ,  10.25      ,
        40.        ,  28.4       ])
```

In [31]:
```
dic.values()
```

Out[31]:
```
dict_values([13, 5, 1, 7, 2, 2, 1, 3, 1, 9])
```

In [32]:
```
list(dic.values())
```

Out[32]:
```
[13, 5, 1, 7, 2, 2, 1, 3, 1, 9]
```

In [33]:
```
np.repeat(temp.values, list(dic.values()))
```

Out[33]:
```
array([ 33.06451613,  33.06451613,  33.06451613,  33.06451613,
        33.06451613,  33.06451613,  33.06451613,  33.06451613,
        33.06451613,  33.06451613,  33.06451613,  33.06451613,
        33.06451613,  12.1559633 ,  12.1559633 ,  12.1559633 ,
        12.1559633 ,  12.1559633 ,  10.875     ,  39.03703704,
        39.03703704,  39.03703704,  39.03703704,  39.03703704,
        39.03703704,  39.03703704,   3.93220339,   3.93220339,
        66.5       ,  66.5       , 113.        ,  10.25      ,
        10.25      ,  10.25      ,  40.        ,  28.4       ,
        28.4       ,  28.4       ,  28.4       ,  28.4       ,
        28.4       ,  28.4       ,  28.4       ,  28.4       ])
```

In [34]:
```
# 원하는 column명을 ''사이에 기재

patient_group['abc'] = np.repeat(temp.values, list(dic.values()))
```

In [36]:
```
# 모범답안2 : join

temp = patient_info_2.groupby(['province'])['contact_number', 'age'].mean()
temp.columns = ['province_mean_contact_number', 'province_mean_age']

temp
```

Out[36]:

province	province_mean_contact_number	province_mean_age
Busan	33.064516	40.784946
Chungcheongnam-do	12.155963	40.733945
Daegu	1160.000000	62.000000
Gangwon-do	10.875000	52.750000
Gyeonggi-do	39.037037	52.555556
Gyeongsangbuk-do	3.932203	44.983051
Jeju-do	66.500000	35.500000
Jeollabuk-do	113.000000	63.000000
Jeollanam-do	10.250000	33.250000
Sejong	40.000000	33.000000
Seoul	28.400000	48.933333

In [38]:
```
patient_group = patient_group.join(temp)
patient_group
```

Out[38]:

province	city	contact_number	age	abc	province_mean_contact_number	p
Busan	Buk-gu	12.000000	32.000000	33.064516	33.064516	
	Busanjin-gu	16.142857	51.000000	33.064516	33.064516	
	Dongnae-gu	46.586207	37.172414	33.064516	33.064516	
	Gangseo-gu	14.000000	26.666667	33.064516	33.064516	
	Geumjeong-gu	15.250000	32.500000	33.064516	33.064516	
	Haeundae-gu	24.437500	43.062500	33.064516	33.064516	
	Nam-gu	148.500000	50.000000	33.064516	33.064516	
	Saha-gu	8.000000	38.714286	33.064516	33.064516	
	Sasang-gu	22.500000	42.000000	33.064516	33.064516	

9. 경기도 시흥시의 contact_number와 age를 추출해주세요

In [39]:
```
# 모범답안

patient_group.loc[('Gyeonggi-do', 'Siheung-si'), ['contact_number', 'age']]
```

Out[39]:
```
contact_number    17.333333
age               55.666667
Name: (Gyeonggi-do, Siheung-si), dtype: float64
```

10. 7번의 결과 데이터프레임을 활용해서 province를 입력하면 contact_number가 가장 높은 5개의 city를 출력해주는 함수를 만들어주세요

In [5]:
```
# 아래 셀에서 작성해주세요
print_city('Seoul')
```

```
1 번째 : Gangseo-gu
2 번째 : Songpa-gu
3 번째 : Jongno-gu
4 번째 : Jungnang-gu
5 번째 : Seodaemun-gu
```

In [6]:
```
# 아래 셀에서 작성해주세요
print_city('Busan')
```

```
1 번째 : Nam-gu
2 번째 : Dongnae-gu
3 번째 : Haeundae-gu
4 번째 : Sasang-gu
5 번째 : Busanjin-gu
```

In [27]:
```
patient_avg.sort_values(by = 'contact_number', ascending = False).loc['Seoul'].
```

Out[27]:
```
'Songpa-gu'
```

In [29]:
```
def print_city(city):
    for x in range(5):
        city_name = patient_avg.sort_values(by = 'contact_number', ascending =
        print("%d 번째 : %s" %(x + 1, city_name))
```

```
1 번째 : Gangseo-gu
2 번째 : Songpa-gu
3 번째 : Jongno-gu
4 번째 : Jungnang-gu
5 번째 : Seodaemun-gu
```

In [33]:
```
print('[Seoul]')
print_city('Seoul')
print('\n')
print('[Busan]')
print_city('Busan')
```

```
[Seoul]
1 번째 : Gangseo-gu
2 번째 : Songpa-gu
3 번째 : Jongno-gu
4 번째 : Jungnang-gu
5 번째 : Seodaemun-gu


[Busan]
1 번째 : Nam-gu
2 번째 : Dongnae-gu
3 번째 : Haeundae-gu
4 번째 : Sasang-gu
5 번째 : Busanjin-gu
```

In [43]:
```
# 모범답안

patient_group.xs('Seoul', axis = 0, level = 0)
```

Out[43]:

city	contact_number	age	abc	province_mean_contact_number	province_mean_age
Gangseo-gu	75.0	57.0	28.4	28.4	48.933333
Jongno-gu	40.8	58.0	28.4	28.4	48.933333
Jungnang-gu	31.0	34.0	28.4	28.4	48.933333
Mapo-gu	9.0	30.0	28.4	28.4	48.933333
Seodaemun-gu	23.0	59.0	28.4	28.4	48.933333
Seongbuk-gu	4.0	45.0	28.4	28.4	48.933333
Seongdong-gu	8.0	78.0	28.4	28.4	48.933333
Songpa-gu	68.0	38.0	28.4	28.4	48.933333
etc	0.0	29.0	28.4	28.4	48.933333

In [44]:
```
patient_group.xs('Seoul', axis = 0, level = 0).sort_values(by = 'contact_numbe
```

Out[44]:

city	contact_number	age	abc	province_mean_contact_number	province_mean_age
Gangseo-gu	75.0	57.0	28.4	28.4	48.933333
Songpa-gu	68.0	38.0	28.4	28.4	48.933333
Jongno-gu	40.8	58.0	28.4	28.4	48.933333
Jungnang-gu	31.0	34.0	28.4	28.4	48.933333
Seodaemun-gu	23.0	59.0	28.4	28.4	48.933333

In [45]:
```
patient_group.xs('Seoul', axis = 0, level = 0).sort_values(by = 'contact_numbe
```

Out[45]:
```
Index(['Gangseo-gu', 'Songpa-gu', 'Jongno-gu', 'Jungnang-gu', 'Seodaemun-g
u'], dtype='object', name='city')
```

In [47]:
```
temp = patient_group.xs('Seoul', axis = 0, level = 0).sort_values(by = 'contact
for n, v in enumerate(temp):
    print(n, v)
```

```
0 Gangseo-gu
1 Songpa-gu
2 Jongno-gu
3 Jungnang-gu
4 Seodaemun-gu
```

In [49]:
```
print_city(province):
temp = patient_group.xs('Seoul', axis = 0, level = 0).sort_values(by = 'contac
for n, v in enumerate(temp):
    print('%d번째 : %s' %(n + 1, v))
```

In [50]:
```
print_city('Busan')
```

```
1번째 : Gangseo-gu
2번째 : Songpa-gu
3번째 : Jongno-gu
4번째 : Jungnang-gu
5번째 : Seodaemun-gu
```

11. patientInfo와 PatientRoute를 patient_id를 기준으로 inner join 해주세요

In [35]:
```
pat_sort_info = patient_info.sort_values(by = 'patient_id')
pat_sort_route = patient_route.sort_values(by = 'patient_id')
patient_inner_id = pd.merge(pat_sort_info, pat_sort_route, how = 'inner')

patient_inner_id
```

Out[35]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	infection_c
0	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	over in
1	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	over in
2	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	over in
3	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	over in
4	1000000003	6.0	male	1964.0	57.0	Korea	Seoul	Jongno-gu	NaN	contact pa
...	...	...	...	...	...	...	...	...	...	
66	2000000011	28.0	female	1989.0	32.0	China	Gyeonggi-do	Goyang-si	NaN	contact pa
67	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	over in
68	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	over in
69	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	over in
70	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	over in

71 rows × 21 columns

In [51]:
```
# 모범답안

patient_merge = pd.merge(patient_info, patient_route, on = 'patient_id', how =
patient_merge
```

Out[51]:

	patient_id	global_num_x	sex	birth_year	age	country	province_x	city_x	disease	infecti
0	1000000001	2.0	male	1964.0	57.0	Korea	Seoul	Gangseo-gu	NaN	
1	1000000001	2.0	male	1964.0	57.0	Korea	Seoul	Gangseo-gu	NaN	
2	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	
3	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	
4	1000000002	5.0	male	1987.0	34.0	Korea	Seoul	Jungnang-gu	NaN	
...	...	...	...	...	...	...	...	...	...	
170	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	
171	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	
172	5000000001	8.0	female	1958.0	63.0	Korea	Jeollabuk-do	Gunsan-si	NaN	
173	5100000001	22.0	male	1974.0	47.0	Korea	Jeollanam-do	NaN	NaN	cor
174	5100000001	22.0	male	1974.0	47.0	Korea	Jeollanam-do	NaN	NaN	cor

175 rows × 24 columns

12. 위에서 merge한 파일에서 각 column마다 NaN의 개수와 비율을 구하기 위해 아래와 같은 data frame을 만들어주세요

In [7]:
```
# 아래 셀에서 작성해주세요
na
```

Out[7]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	infecti
index	0	1	2	3	4	5	6	7	8	
column type	int64	float64	object	float64	float64	object	object	object	object	
null values(num)	0	0	0	0	0	0	0	0	71	
null values(%)	0	0	0	0	0	0	0	0	100	

4 rows × 21 columns

In [54]:
```
# 모범답안

na = {'index': [x for x in range(patient_merge.shape[1])],
      'column type': patient_merge.dtypes,
      'null values(num)': patient_merge.isnull().sum(),
      'null values(%)': patient_merge.isnull().sum() * 100 / patient_merge.shape

pd.DataFrame(na)
```

Out[54]:

	index	column type	null values(num)	null values(%)
patient_id	0	int64	0	0.000000
global_num_x	1	float64	0	0.000000
sex	2	object	0	0.000000
birth_year	3	float64	5	2.857143
age	4	float64	5	2.857143
country	5	object	0	0.000000
province_x	6	object	0	0.000000
city_x	7	object	15	8.571429
disease	8	object	175	100.000000
infection_case	9	object	3	1.714286
infection_order	10	float64	17	9.714286

In [55]:
```
pd.DataFrame(na).T
```

Out[55]:

	patient_id	global_num_x	sex	birth_year	age	country	province_x	city_x	disease
index	0	1	2	3	4	5	6	7	8
column type	int64	float64	object	float64	float64	object	object	object	object
null values(num)	0	0	0	5	5	0	0	15	175
null values(%)	0	0	0	2.85714	2.85714	0	0	8.57143	100

4 rows × 24 columns

13. null values(%)가 100인 column은 제거해주세요

In [67]:
```
na = pd.DataFrame(na).T
```

In [68]:
```
na.loc['null values(%)'] == 100
```

Out[68]:
```
patient_id            False
global_num_x          False
sex                   False
birth_year            False
age                   False
country               False
province_x            False
city_x                False
disease                True
infection_case        False
infection_order       False
infected_by           False
contact_number        False
symptom_onset_date    False
confirmed_date        False
released_date         False
deceased_date          True
state                 False
global_num_y          False
date                  False
province_y            False
city_y                False
latitude              False
longitude             False
Name: null values(%), dtype: bool
```

In [69]:
```
null_100 = na.loc['null values(%)'] == 100
```

In [70]:
```
null_100[null_100 == 100]
```

Out[70]:
```
Series([], Name: null values(%), dtype: bool)
```

In [71]:
```
null_100[null_100 == 100].index
```

Out[71]:
```
Index([], dtype='object')
```

In [72]:
```
drop_col = null_100[null_100 == 100].index
```

In [73]:
```
patient_merge.drop(drop_col, axis = 1, inplace = True)
```

14.

1: patient_info에서 감염 경우가 contact with patient인 경우

2: case에서 group(집단 감염 여부)가 true인 경우

1,2번 두 csv파일을 province, city를 기준으로 inner join해서

sex(성별)을 기준으로 state(완치(released), 자가격리(isolated) 명수 파악

In [8]:
```
# 아래 셀에서 작성해주세요
state_df
```

Out[8]:

	isolated	released
male	55	20
female	82	22

In [74]:
```
# 모범답안

data1 = patient_info[patient_info.infection_case == 'contact with patient']
data2 = case[case.group == True]
data3 = pd.merge(data1, data2, on = ['province', 'city'], how = 'inner')
```

In [75]:
```
data3.groupby(['sex', 'state'])['state'].count()
```

Out[75]:
```
sex     state
female  isolated    82
        released    22
male    isolated    55
        released    20
Name: state, dtype: int64
```

In [76]:
```
data3.groupby('sex')['state'].value_counts()
```

Out[76]:
```
sex     state
female  isolated    82
        released    22
male    isolated    55
        released    20
Name: state, dtype: int64
```

In [77]:
```
pd.DataFrame(data3.groupby('sex')['state'].value_counts()).unstack(level = 1)
```

Out[77]:

	state	
state	isolated	released
sex		
female	82	22
male	55	20

In [78]:
```
# index를 뒤집을 수 있다.

a= pd.DataFrame(data3.groupby('sex')['state'].value_counts()).unstack(level = 1
a.sort_index(ascending = False)
```

Out[78]:

	state	
state	isolated	released
sex		
male	55	20
female	82	22