

## 4장. 지리정보 분석

1. Folium
2. 지리정보 분석 후 맵 생성
3. 행정구역별 의료기간 현황 분석하기

# 1. Folium

## □ 포리움(Folium)

- ▣ 참조 사이트 : <http://python-visualization.github.io/folium/>
- ▣ 파이썬 생태계의 강점인 데이터와 Leaflet.js 라이브러리의 매핑 강점을 토대로 제작
- ▣ Python으로 데이터를 조작한 다음, Folium을 통해 리플릿 맵에서 시각화
- ▣ 위도(latitude) : 적도를 기준으로 남쪽으로 남극점까지 90°, 북쪽으로 북극점까지 90°로 나누어 표시(우리나라 적도의 북쪽인 북위 34° ~ 38° 사이에 위치)
- ▣ 경도(longitude) : 런던 그리니치천문대를 지나는 본초 자오선을 중심으로 동서로 나누어 동경 180°, 서경 180°로 분리(서울의 경우 동경 127°에 위치).

# 1. Folium

## □ Folium 설치 및 버전확인

```
#pip install folium
import numpy as np
import folium
from folium import plugins
print(folium.__version__)
```

## □ Folium 객체 생성

```
m = folium.Map(location=[37.566345, 126.977893])
m.save('data/map1.html') #파일이 저장될 위치
m
```

## □ zoo\_start 속성 및 ScrollZoomToggler

```
m = folium.Map([37.566345, 126.977893], zoom_start=4)
m.save(os.path.join('results', 'map2.html'))
m
```

# 1. Folium

## □ 맵의 유형

- 포리움은 기본적으로 'Open Street Map'을 기반으로 동작
- 'Stamen Terrain', 'Stamen Toner', 'Mapbox Bright', 와 'Mapbox Control room tiles' 형식을 내장

```
map_osm = folium.Map(location=[37.566345, 126.977893], zoom_start=17, tiles='Stamen Terrain')
map_osm.save('data_al/map3.html')
map_osm = folium.Map(location=[37.566345, 126.977893], zoom_start=17, tiles='Stamen Toner')
map_osm.save('data_al/map4.html')
```

- Cloudmade'나 'Mapbox'를 사용하는 경우에는 사이트에 등록 시 발급받은 API 키 정보를 아래와 같이 'API\_key' 속성으로 지정

```
map_osm = folium.Map(location=[37.5660, 126.9757], tiles='Mapbox', API_key='API키값')
```

# 1. Folium

## □ 마커(Marker)와 팝업(Popup)의 설정

- 포리움은 다양한 형식의 마커(특정 위치를 표시하는 표식)과 마커를 클릭하였을 때 나타나는 정보(Popup)을 지정할 수 있다.
- Marker() 메소드를 이용하여 생성
- 마커의 인자값으로 위경도 값 리스트와 마커를 클릭할 시 보여줄 문자열을 전달하고, 생성한 포리움 객체에 추가(.add\_to())하면 간단하게 마커를 생성

```
map_osm = folium.Map(location=[37.566345, 126.977893], zoom_start=17)
folium.Marker([37.566345, 126.977893], popup='서울특별시청').add_to(map_osm)
folium.Marker([37.5658859, 126.9754788], popup='덕수궁').add_to(map_osm)
map_osm.save('data_al/map5.html')
```

# 1. Folium

## □ 마커(Marker)와 팝업(Popup)의 설정

- 포리움 마커는 부트스트랩(bootstrap)을 이용, 아이콘 타입을 설정할 수 있으며, 범위를 설정하기 위하여 circle 속성을 줄 수 있다.
- 덕수궁의 위치를 좀더 크게 마커로 표시하고, 서울특별시청은 적색의 'info-sign' 마커로 표시한 예

```
m=folium.Map(location=[37.566345, 126.977893], zoom_start=17)
folium.Marker([37.566345, 126.977893], popup='서울특별시청',
icon=folium.Icon(color='red',icon='info-sign')).add_to(m)
folium.CircleMarker([37.5658859, 126.9754788], radius=100,
color='#3186cc', fill_color='#3186cc', popup='덕수궁').add_to(m)
map_osm.save('data_al/map6.html')
```

# 1. Folium

## □ 마커(Marker)와 팝업(Popup)의 설정

```
N = 100
data = np.array(
    [
        np.random.uniform(low=35, high=38, size=N), # Random latitudes.
        np.random.uniform(low=125, high=129, size=N), # Random longitudes.
    ] ).T
popups = [str(i) for i in range(N)] # Popups texts are simple numbers.
m = folium.Map([37.566345, 126.977893], zoom_start=8)
plugins.MarkerCluster(data, popups=popups).add_to(m)
m.save(os.path.join('results', 'Plugins_1.html'))
m
```

## 2. 지리정보 분석 후 맵 생성하기

### 지리 정보 분석 후 맵 생성하기

목표	특정 주소에 대한 지리 정보를 분석한 뒤 위치를 시각화한 맵을 생성한다.
핵심 개념	위도와 경도의 GPS 좌표, 지리 정보 분석, 지리 정보 시각화 라이브러리
데이터 수집	1. 커피 매장의 주소 데이터: 6장에서 크롤링으로 수집한 CoffeeBean.csv 파일(예제소스로 제공) 2. 행정구역 주소체계 데이터: 국가통계포털에서 다운로드한 '행정구역_시군구_별_성별_인구수.xlsx' 파일
데이터 준비 및 탐색	1. 데이터 정제: 주소의 행정구역 이름을 정확한 이름으로 수정 2. 데이터 조합: 필요한 컬럼을 추출하고 병합

### 분석 모델 구축 및 결과 시각화

#### 1. 포리움 라이브러리로 생성한 맵



#### 2. 특정 주소의 위치를 시각화한 맵





## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ▪ 주소 데이터 수집하기

- 사용자 폴더에 data 폴더를 만들고 CoffeeBean.csv 파일을 복사하여 붙여넣음

#### ▪ 행정구역 주소 체계 데이터 수집하기

1. 국가통계포털 사이트(<http://kosis.kr>)에서 '행정구역'으로 데이터를 검색



그림 9-1 국가통계포털 사이트에서 데이터 검색

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ■ 행정구역 주소 체계 데이터 수집하기

2. 검색 결과 중에서 '주민등록인구현황:행정구역(시군구)별, 성별 인구수'를 선택



그림 9-2 검색 결과 중에서 '주민등록인구현황:행정구역(시군구)별, 성별 인구수' 선택

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ▣ 행정구역 주소 체계 데이터 수집하기

2. 행정구역이 상위 레벨(시도)만 있고 하위 레벨(군구)은 보이지 않음



KOSIS

1) 행정구역(시군구)별, 성별 인구수

자료검색일: 2020-10-06 / 수록기간: 월, 년 1992 ~ 2020.09 / 자료문의처 : 044-205-3158

일괄설정 + 항목 [3/3] 행정구역(시군구)별 [18...] 시점 [3/145]

주석정보 주소정보 행렬전환 분석 차트 부가

행정구역(시군구)별	2020. 09			2020. 08	
	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	총인구수 (명)	남자인
전국	51,841,786	25,851,003	25,990,783	51,839,953	
서울특별시	9,699,232	4,719,170	4,980,062	9,708,247	
2) 부산광역시	3,399,749	1,665,554	1,734,195	3,401,072	
대구광역시	2,426,849	1,197,914	1,228,935	2,428,228	
인천광역시	2,942,553	1,474,105	1,468,448	2,943,491	
광주광역시	1,453,952	719,445	734,507	1,454,154	
대전광역시	1,469,099	733,356	735,743	1,469,431	
울산광역시	1,139,368	585,387	553,981	1,140,310	

그림 9-3 검색 데이터 확인 - 행정구역(시군구)별 확인

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ■ 행정구역 주소 체계 데이터 수집하기

3. 행정구역(시군구)별] 탭을 클릭하고, [2 레벨 전체선택]을 체크해서 선택



그림 9-4 검색 데이터 확인 - 행정구역 2 레벨 선택

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ▪ 행정구역 주소 체계 데이터 수집하기

4. [시점] 탭을 클릭하고, 시점 목록에서 2020.01을 선택한 뒤 버튼을 클릭



그림 9-5 검색 데이터 확인 - 시점 설정

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 수집

#### ▣ 행정구역 주소 체계 데이터 수집하기

##### 5. 파일형태를 'EXCEL'로 선택하고 버튼을 클릭

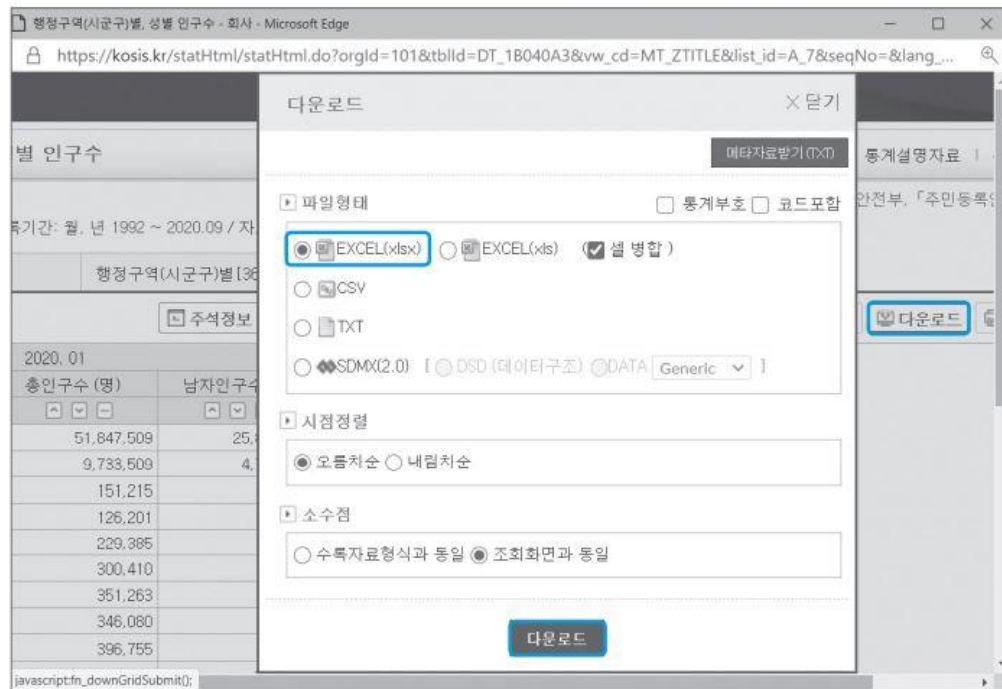


그림 9-6 데이터 파일 다운로드

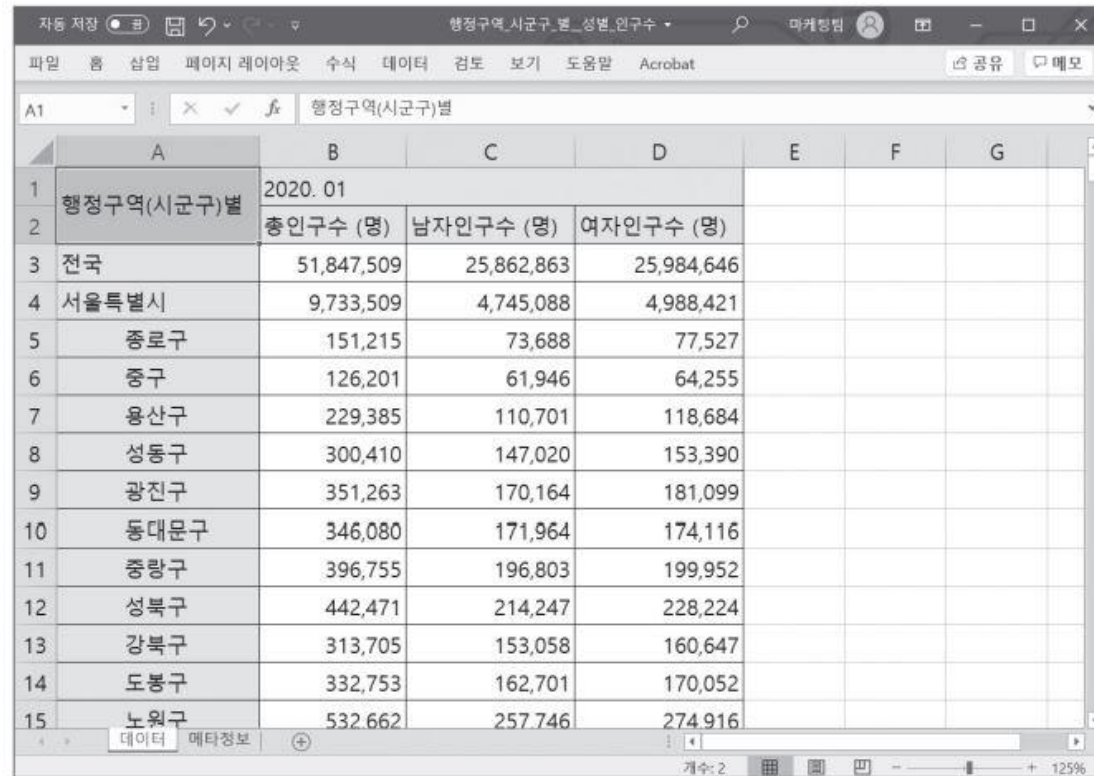
##### 6. 다운로드된 파일은 이름을 '행정구역\_시군구\_별\_성별\_인구수.xlsx'로 수정한 뒤에 9장\_data 폴더로 이동

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

1. 국가통계포털에서 다운로드한 '행정구역\_시군구\_별\_성별\_인구수.xlsx' 파일을 열어서 확인



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G
1	행정구역(시군구)별	2020. 01					
2		총인구수 (명)	남자인구수 (명)	여자인구수 (명)			
3	전국	51,847,509	25,862,863	25,984,646			
4	서울특별시	9,733,509	4,745,088	4,988,421			
5	종로구	151,215	73,688	77,527			
6	중구	126,201	61,946	64,255			
7	용산구	229,385	110,701	118,684			
8	성동구	300,410	147,020	153,390			
9	광진구	351,263	170,164	181,099			
10	동대문구	346,080	171,964	174,116			
11	중랑구	396,755	196,803	199,952			
12	성북구	442,471	214,247	228,224			
13	강북구	313,705	153,058	160,647			
14	도봉구	332,753	162,701	170,052			
15	노원구	532,662	257,746	274,916			

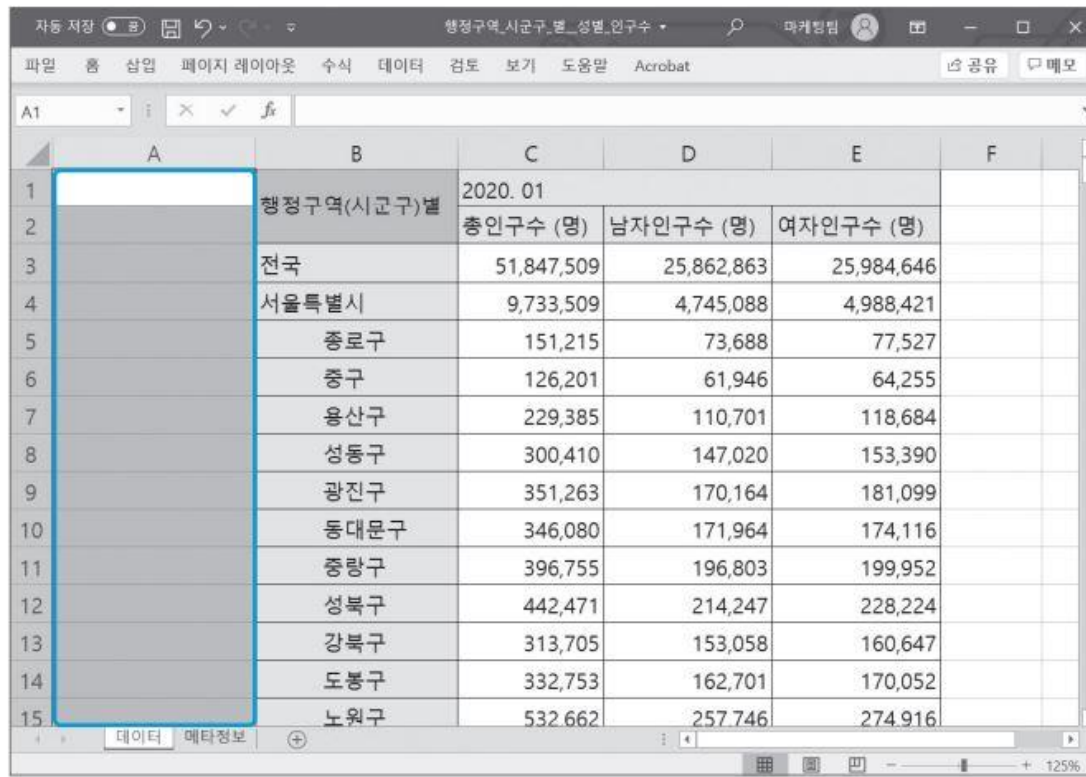
그림 9-7 데이터 파일 확인하기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

2. 엑셀에서 데이터 정리하기 먼저, 왼쪽에 빈 열을 삽입



	A	B	C	D	E	F
1		행정구역(시군구)별	2020. 01			
2			총인구수 (명)	남자인구수 (명)	여자인구수 (명)	
3		전국	51,847,509	25,862,863	25,984,646	
4		서울특별시	9,733,509	4,745,088	4,988,421	
5		종로구	151,215	73,688	77,527	
6		중구	126,201	61,946	64,255	
7		용산구	229,385	110,701	118,684	
8		성동구	300,410	147,020	153,390	
9		광진구	351,263	170,164	181,099	
10		동대문구	346,080	171,964	174,116	
11		중랑구	396,755	196,803	199,952	
12		성북구	442,471	214,247	228,224	
13		강북구	313,705	153,058	160,647	
14		도봉구	332,753	162,701	170,052	
15		노원구	532,662	257,746	274,916	

그림 9-8 빈 열 삽입하기



## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

3. 시 이름을 1 레벨로, 군/구 이름을 2 레벨로 옮기는 작업을 실시  
1 레벨로 분리할 시 이름을 잘라내어 A열에 붙여넣음

	A	B	C	D	E	F
1			2020. 01			
2	행정구역(시군구)별	행정구역(시군구)별	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	
3	전국		51,847,509	25,862,863	25,984,646	
4	서울특별시		9,733,509	4,745,088	4,988,421	
5		종로구	151,215	73,688	77,527	
6		중구	126,201	61,946	64,255	
7		용산구	229,385	110,701	118,684	
8		성동구	300,410	147,020	153,390	
9		광진구	351,263	170,164	181,099	
10		동대문구	346,080	171,964	174,116	
11		종랑구	396,755	196,803	199,952	
12		성북구	442,471	214,247	228,224	
13		강북구	313,705	153,058	160,647	
14		도봉구	332,753	162,701	170,052	
15		노원구	532,662	257,746	274,916	

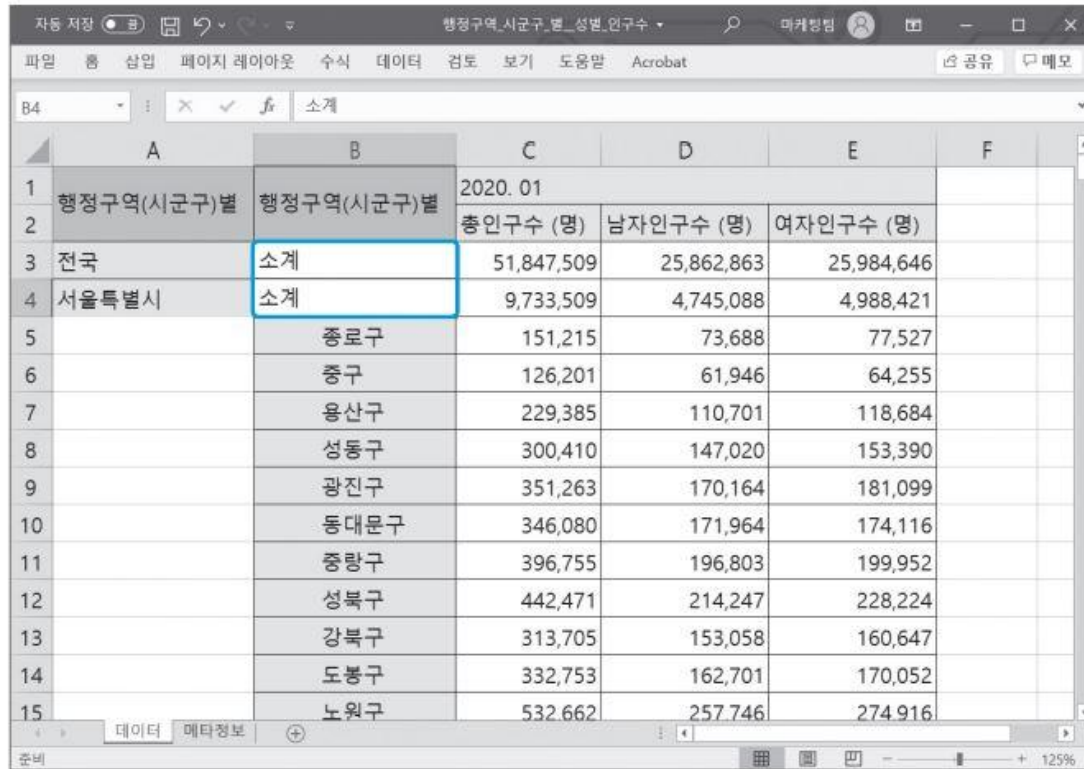
그림 9-9 1 레벨로 분리할 시 이름을 잘라내어 A열에 붙여넣기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

##### 4. 잘라낸 시 이름 자리에 합계를 나타내는 '소계'를 입력



	A	B	C	D	E	F
1	행정구역(시군구)별	행정구역(시군구)별	2020. 01			
2			총인구수 (명)	남자인구수 (명)	여자인구수 (명)	
3	전국	소계	51,847,509	25,862,863	25,984,646	
4	서울특별시	소계	9,733,509	4,745,088	4,988,421	
5		종로구	151,215	73,688	77,527	
6		중구	126,201	61,946	64,255	
7		용산구	229,385	110,701	118,684	
8		성동구	300,410	147,020	153,390	
9		광진구	351,263	170,164	181,099	
10		동대문구	346,080	171,964	174,116	
11		중랑구	396,755	196,803	199,952	
12		성북구	442,471	214,247	228,224	
13		강북구	313,705	153,058	160,647	
14		도봉구	332,753	162,701	170,052	
15		노원구	532,662	257,746	274,916	

그림 9-10 잘라낸 시 이름 자리에 '소계' 입력하기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

##### 5. 엑셀의 채우기 기능을 이용해 빈 자리에 시 이름을 복사

The first screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F
16	서울특별시	은평구	479,457	231,171	248,286	
17	서울특별시	서대문구	310,353	149,114	161,239	
18	서울특별시	마포구				
19	서울특별시	양천구				
20	서울특별시	강서구				
21	서울특별시	구로구				
22	서울특별시	금천구				
23	서울특별시	영등포구				
24	서울특별시	동작구				
25	서울특별시	관악구				
26	서울특별시	서초구				
27	서울특별시	강남구				
28	서울특별시	송파구				
29	서울특별시	강동구				

The second screenshot shows a larger Excel spreadsheet with the following data:

	A	B	C	D	E	F
268	경상남도	양산시	351,168	176,291	174,877	
269	경상남도	의령군	27,131	13,223	13,908	
270	경상남도	합안군	65,491	33,120	32,371	
271	경상남도	창녕군	62,182	30,945	31,237	
272	경상남도	고성군	52,204	26,174	26,030	
273	경상남도	남해군	43,539	20,939	22,600	
274	경상남도	하동군	46,331	23,037	23,294	
275	경상남도	산청군	35,336	17,300	18,036	
276	경상남도	함양군	39,555	19,058	20,497	
277	경상남도	거창군	62,049	30,275	31,774	
278	경상남도	합천군	45,054	21,590	23,464	
279	제주특별자치도	소계	670,749	337,231	333,518	
280	제주특별자치도	제주시	489,202	245,389	243,813	
281	제주특별자치도	서귀포시	181,547	91,842	89,705	

그림 9-11 빈 자리에 시 이름 복사하여 채워넣기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ■ 행정구역 주소 체계 데이터 준비하기

6. 첫 번째 행을 삭제하고 열 이름을 '행정구역(시군구)별(1)'과 '행정구역(시군구)별(2)'로 변경



	A	B	C	D	E	F
1	행정구역(시군구)별(1)	행정구역(시군구)별(2)	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	
2	전국	소계	51,847,509	25,862,863	25,984,646	
3	서울특별시	소계	9,733,509	4,745,088	4,988,421	
4	서울특별시	종로구	151,215	73,688	77,527	
5	서울특별시	중구	126,201	61,946	64,255	
6	서울특별시	용산구	229,385	110,701	118,684	
7	서울특별시	성동구	300,410	147,020	153,390	

그림 9-12 첫 번째 행의 A, B열 이름 변경하기

6. 작업이 끝나면 파일을 '행정구역\_시군구\_별\_성별\_인구수\_2.xlsx'로 저장

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ▪ 분석할 커피 매장의 주소 데이터 준비하기

##### 1. '9장\_주소데이터분석'으로 파일 이름을 변경 후 입력

In [1]:

```
import pandas as pd
CB = pd.read_csv('data/CoffeeBean.csv', encoding = 'CP949', index_col = 0, header = 0,
                 engine = 'python')
CB.head() #작업 내용 확인용 출력
```

Out[1]:

	store	address	phone
0	학동역 DT점	서울시 강남구 학동로 211 1층	02-3444-8873
1	수서점	서울시 강남구 광평로 280 수서동 724호	02-3412-2328
2	차병원점	서울시 강남구 논현로 566 강남차병원1층	02-538-7818
3	강남대로점	서울시 서초구 강남대로 369 1층	02-588-8778
4	메가박스점	서울 강남구 삼성동 159 코엑스몰 지하2층	02-6002-8328

In [1]: CoffeeBean.csv 파일을 CB 객체로 로드하고, 상위 5개 행의 데이터를 출력하여 head( ) 확인

파이썬에서 CSV 파일을 파싱하는 과정에서 문제가 발생할 수 있는데, 이를 해결하기 위해 engine = 'python' 속성 추가

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

- 분석할 커피 매장의 주소 데이터 준비하기

- 2. 주소 데이터를 행정구역 주소 체계에 맞게 정리하기

In [2]:	<pre>addr = [] for address in CB.address:     addr.append(str(address).split()) addr #작업 내용 확인용 출력</pre>
Out[2]:	<pre>[['서울시', '강남구', '학동로', '211', '1층'],  ['서울시', '강남구', '광평로', '280', '수서동', '724호'],  ...,  ['경기도', '안양시', '동안구', '시민대로', '260', '1층', '104,105호'],  ['경기도', '하남시', '미사대로', '750', '신세계백화점', '지하1층', '식품관']]</pre>

In [2]: for 반복문을 이용하여 각 address 컬럼의 값을 분리하고 `split()` addr 리스트로 만듦.

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

- 분석할 커피 매장의 주소 데이터 준비하기
  - 2. 주소 데이터를 행정구역 주소 체계에 맞게 정리하기

In [3]:	<pre>addr2 = [] for i in range(len(addr)):     if addr[i][0] == "서울": addr[i][0] = "서울특별시"     elif addr[i][0] == "서울시": addr[i][0] = "서울특별시"     elif addr[i][0] == "부산시": addr[i][0] = "부산광역시"     elif addr[i][0] == "인천": addr[i][0] = "인천광역시"     elif addr[i][0] == "광주": addr[i][0] = "광주광역시"     elif addr[i][0] == "대전시": addr[i][0] = "대전광역시"     elif addr[i][0] == "울산시": addr[i][0] = "울산광역시"     elif addr[i][0] == "세종시": addr[i][0] = "세종특별자치시"     elif addr[i][0] == "경기": addr[i][0] = "경기도"     elif addr[i][0] == "충북": addr[i][0] = "충청북도"     elif addr[i][0] == "충남": addr[i][0] = "충청남도"     elif addr[i][0] == "전북": addr[i][0] = "전라북도"     elif addr[i][0] == "전남": addr[i][0] = "전라남도"     elif addr[i][0] == "경북": addr[i][0] = "경상북도"     elif addr[i][0] == "경남": addr[i][0] = "경상남도"     elif addr[i][0] == "제주": addr[i][0] = "제주특별자치도"     elif addr[i][0] == "제주도": addr[i][0] = "제주특별자치도"     elif addr[i][0] == "제주시": addr[i][0] = "제주특별자치도" addr2.append(' '.join(addr[i])) addr2 #작업 내용 확인용 출력</pre>
Out[3]:	<pre>['서울특별시 강남구 학동로 211 1층',  '서울특별시 강남구 광평로 280 수서동 724호',  ...,  '경기도 안양시 동안구 시민대로 260, 1층 104,105호',  '경기도 하남시 미사대로 750, 신세계백화점 지하1층 식품관']</pre>
In [ ]:	<pre>n() addr2 리스트를 만들</pre>

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 데이터 준비 및 탐색

#### ▪ 분석할 커피 매장의 주소 데이터 준비하기

##### 2. 주소 데이터를 행정구역 주소 체계에 맞게 정리하기

In [4]:	addr2 = pd.DataFrame(addr2, columns = ['address2'])																														
In [5]:	CB2 = pd.concat([CB, addr2], axis = 1 ) CB2.head() #작업 내용 확인용 출력																														
Out[5]:	<table><tr><th></th><th>store</th><th>address</th><th>phone</th><th>address2</th></tr><tr><td>0</td><td>학동역 DT점</td><td>서울시 강남구 학동로 211 1층</td><td>02-3444-1111</td><td>서울특별시 강남구 학동로 211 1층</td></tr><tr><td>1</td><td>수서점</td><td>서울시 강남구 광평로 280 수서동 724호</td><td>02-3412-1111</td><td>서울특별시 강남구 광평로 280 수서동 724호</td></tr><tr><td>2</td><td>차병원점</td><td>서울시 강남구 논현로 566 강남차병원1층</td><td>02-538-1111</td><td>서울특별시 강남구 논현로 566 강남차병원1층</td></tr><tr><td>3</td><td>강남대로점</td><td>서울시 서초구 강남대로 369 1층</td><td>02-588-1111</td><td>서울특별시 서초구 강남대로 369 1층</td></tr><tr><td>4</td><td>메가박스점</td><td>서울 강남구 삼성동 159 코엑스몰 지하2층</td><td>02-6002-1111</td><td>서울특별시 강남구 삼성동 159 코엑스몰 지하2층</td></tr></table>		store	address	phone	address2	0	학동역 DT점	서울시 강남구 학동로 211 1층	02-3444-1111	서울특별시 강남구 학동로 211 1층	1	수서점	서울시 강남구 광평로 280 수서동 724호	02-3412-1111	서울특별시 강남구 광평로 280 수서동 724호	2	차병원점	서울시 강남구 논현로 566 강남차병원1층	02-538-1111	서울특별시 강남구 논현로 566 강남차병원1층	3	강남대로점	서울시 서초구 강남대로 369 1층	02-588-1111	서울특별시 서초구 강남대로 369 1층	4	메가박스점	서울 강남구 삼성동 159 코엑스몰 지하2층	02-6002-1111	서울특별시 강남구 삼성동 159 코엑스몰 지하2층
	store	address	phone	address2																											
0	학동역 DT점	서울시 강남구 학동로 211 1층	02-3444-1111	서울특별시 강남구 학동로 211 1층																											
1	수서점	서울시 강남구 광평로 280 수서동 724호	02-3412-1111	서울특별시 강남구 광평로 280 수서동 724호																											
2	차병원점	서울시 강남구 논현로 566 강남차병원1층	02-538-1111	서울특별시 강남구 논현로 566 강남차병원1층																											
3	강남대로점	서울시 서초구 강남대로 369 1층	02-588-1111	서울특별시 서초구 강남대로 369 1층																											
4	메가박스점	서울 강남구 삼성동 159 코엑스몰 지하2층	02-6002-1111	서울특별시 강남구 삼성동 159 코엑스몰 지하2층																											
In [6]:	CB2.to_csv('data/CoffeeBean_2.csv', encoding = 'CP949', index = False)																														

In [4]: addr2를 DataFrame 타입으로 변경하고, 컬럼 이름을 address2로 지정

In [5]: CB와 addr2를 옆으로axis=1 결합하여concat() CB2를 만듦

In [6]: 시도 이름이 수정된 데이터를 CB2로 저장. 작업 완료



## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체 생성하기

1. 구글맵([maps.google.co.kr](https://maps.google.co.kr))에서 우리나라 국보1호인 '송례문'을 검색 → 마커 위에서 마우스 오른쪽 버튼을 클릭하여 [이곳이 궁금한가요?]를 선택

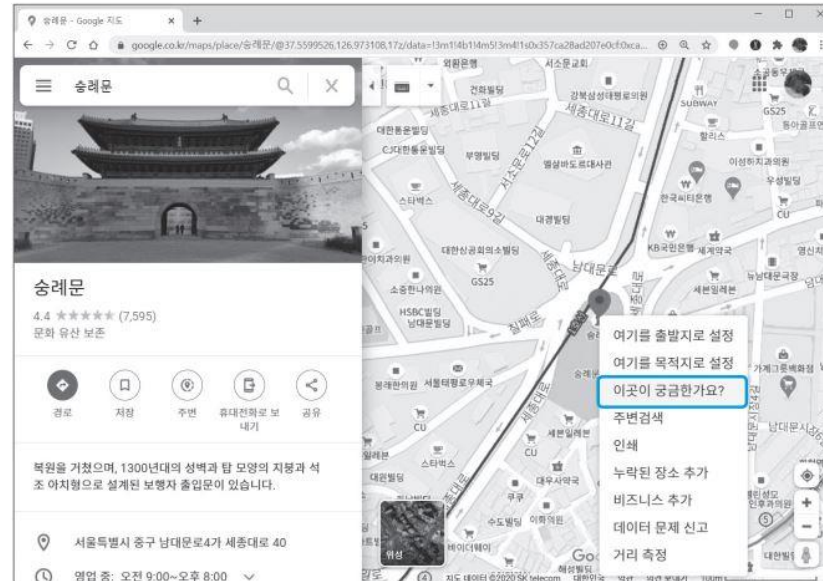


그림 9-13 구글맵에서 좌표 구하기 1 - 송례문을 검색하여 찾기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체 생성하기

##### 2. 송례문의 위도와 경도 좌표를 복사

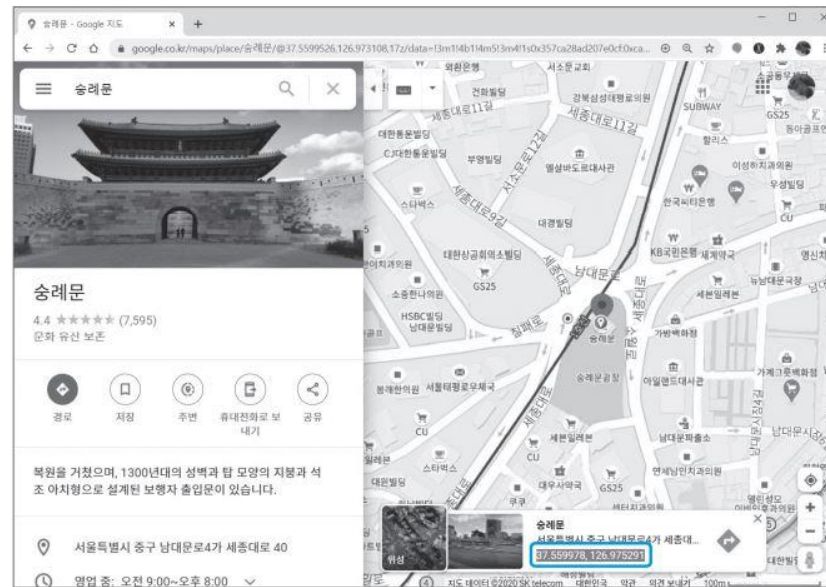


그림 9-14 구글맵에서 좌표 구하기 2 - 송례문의 좌표 복사하기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체 생성하기

##### 3. 복사한 좌표를 사용한 지도 객체를 생성

In [9]:	map_osm = folium.Map(location = [37.559978, 126.975291], zoom_start = 16)
In [10]:	map_osm.save('data/map.html')

In [9]: 복사한 승례문의 좌표를 `folium.Map()` 함수의 `location` 속성 값으로 설정  
지도의 크기를 확대하여 `zoom_start = 16` 지도 객체인 `map_osm`을 생성

In[10]: 생성한 지도 객체 `map_osm`를 파일로 저장

##### 4. data 폴더에 저장된 map.html 파일을 더블클릭해서 열어 지도 파일을 확인

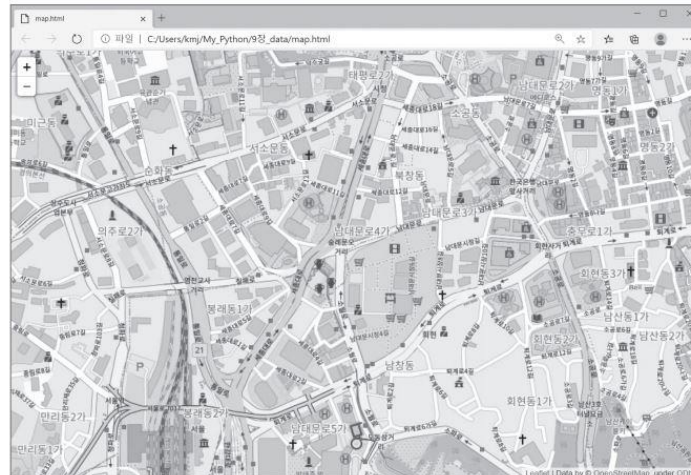


그림 9-15 지도 객체 파일 열기

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

- 지도 객체에 커피 매장 위치 표시하기
  1. Geocoder-Xr를 사용하기 위해 설치하기

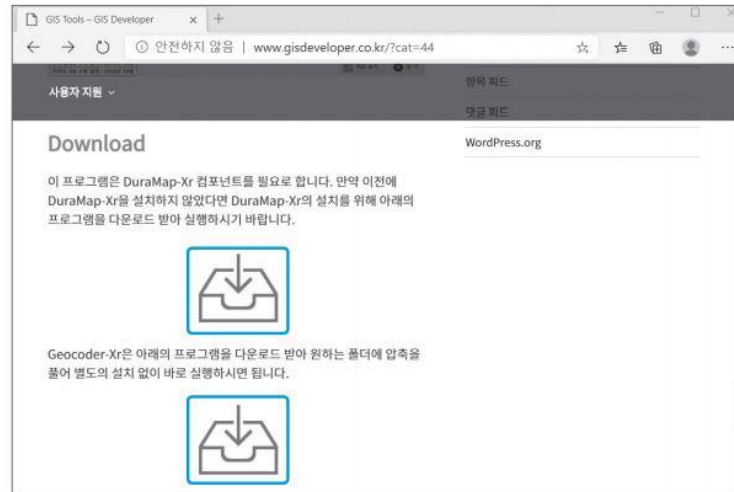


그림 9-16 DuraMap-Xr 컴포넌트와 Geocoder-Xr 다운로드 버튼

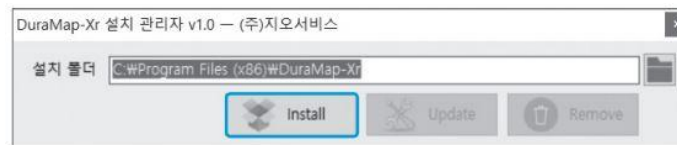


그림 9-17 DuraMap-Xr 컴포넌트 설치

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

- 지도 객체에 커피 매장 위치 표시하기

- 2. 압축을 풀고 Geocoder-Xr\_v4.1 폴더 안에 있는 XrGeocoder.exe를 실행

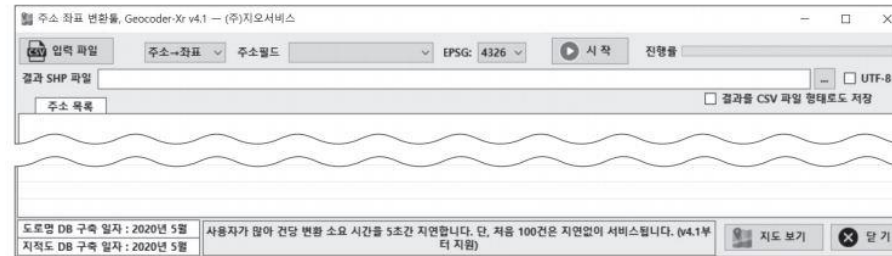


그림 9-18 Geocoder-Xr 실행 화면

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체에 커피 매장 위치 표시하기

##### 3. 주소의 좌표 구하기

- 1 Geocoder-Xr을 실행한 후 [입력 파일]을 클릭해 CoffeeBean\_2.csv로 선택
- 2 좌표를 구할 주소가 있는 [주소필드]를 address2로 설정,
- 3 결과를 저장할 파일 경로를 나타내는 [결과 SHP 파일]에 '9장\_data/CB\_geo.shp'로 입력.
- 4 결과를 CSV 파일 형태로도 저장'을 체크해서 선택
- 5 <시작> 버튼을 클릭

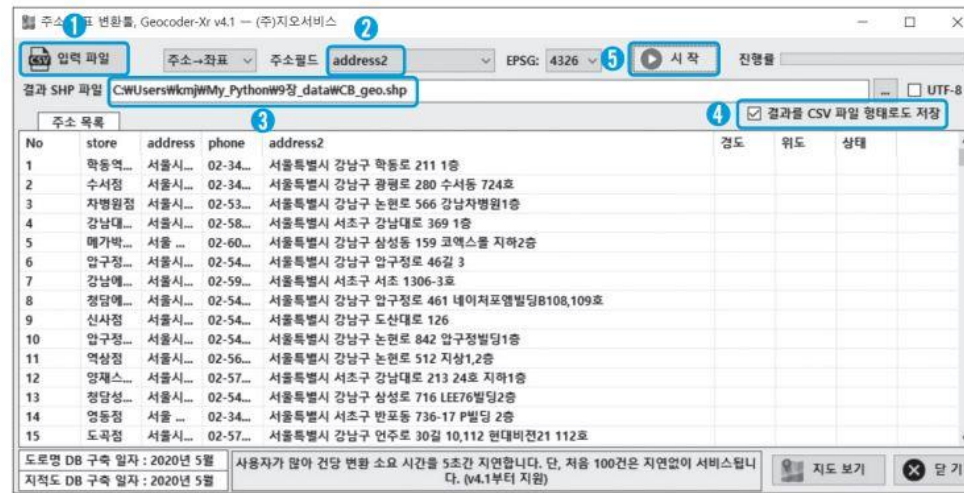


그림 9-19 좌표 구하기 1 - 필요 항목 설정

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체에 커피 매장 위치 표시하기

#### 3. 주소의 좌표 구하기

- CB\_geo.shp.csv 파일: 변환된 주소 좌표
- CB\_geo.shp.err.csv 파일: 주소가 정확하지 않아서 좌표 변환을 하지 못한 항목

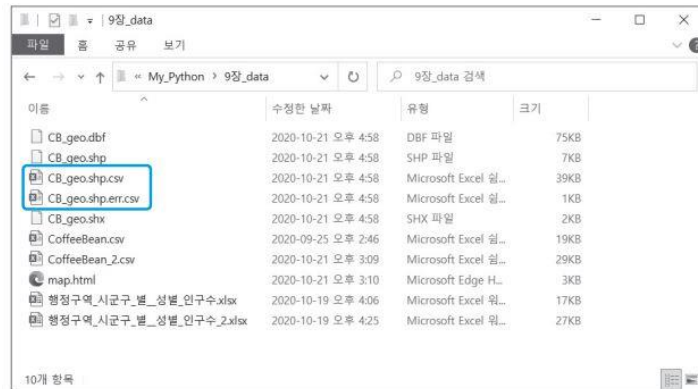


그림 9-20 좌표 구하기 2 - 주소의 좌표 변환 완료 후 생성된 파일

No	store	address	phone	address2	경도	위도	상태
1	학동역 DT	서울시 강'02-3444-9	서울특별시	127.032	37.5147	정좌표	
2	수서점	서울시 강'02-3412-2	서울특별시	127.103	37.4873	정좌표	
3	차병원점	서울시 강'02-538-7	서울특별시				
4	강남대로점	서울시 서'02-588-5	서울특별시				
5	메가박스점	서울시 강남'02-6002-3	서울특별시				
6	압구정점	서울시 강'02-541-5	서울특별시				
8	청담에스점	서울시 강'02-548-6	서울특별시				
9	신사점	서울시 강'02-548-2	서울특별시				
10	압구정역점	서울시 강'02-544-6	서울특별시				
11	역삼점	서울시 강'02-569-8	서울특별시				

No	store	address	phone	address2	경도	위도	상태
7	강남에스점	서울시 서'02-593-5	서울특별시	?	?	?	실패
23	논현팩스점	서울시 강'02-513-3	서울특별시	?	?	?	실패
50	순화점	서울시 중'02-2220-8	서울특별시	?	?	?	실패
60	신촌점	서울시 서'02-363-5	서울특별시	?	?	?	실패
87	무교점	서울시 중'02-753-2	서울특별시	?	?	?	실패
164	고대참살이	서울시 성'02-923-7	서울특별시	?	?	?	실패
174	강서그랜드	서울시 강'02-2699-7	서울특별시	?	?	?	실패

그림 9-21 변환된 주소 좌표를 나타내는 CB\_geo.shp.csv와 변환을 하지 못한 항목을 모아둔 CB\_geo.shp.err.csv

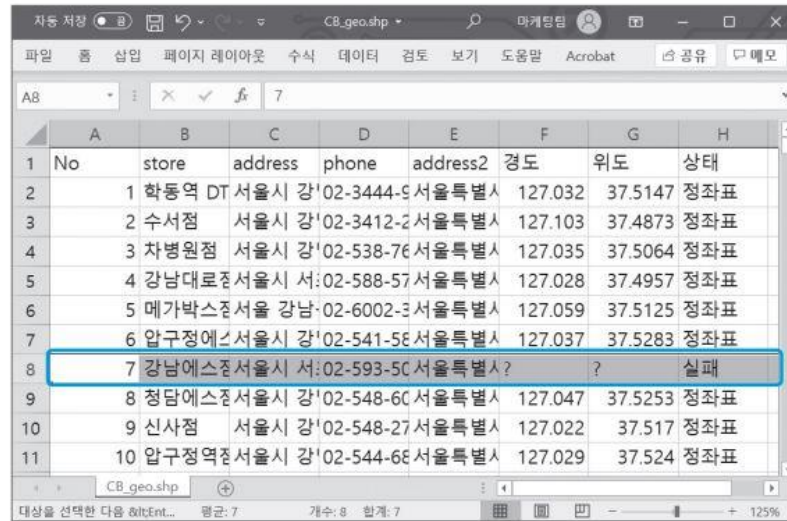


## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

- 지도 객체에 커피 매장 위치 표시하기

4. 좌표 변환을 하지 못한 항목을 복사해서 CB\_geo.shp.csv 파일에 붙여넣음



	A	B	C	D	E	F	G	H
1	No	store	address	phone	address2	경도	위도	상태
2	1	학동역 DT	서울시 강'02-3444-5	서울특별시	127.032	37.5147	정좌표	
3	2	수서점	서울시 강'02-3412-2	서울특별시	127.103	37.4873	정좌표	
4	3	차병원점	서울시 강'02-538-7	서울특별시	127.035	37.5064	정좌표	
5	4	강남대로점	서울시 서:02-588-57	서울특별시	127.028	37.4957	정좌표	
6	5	메가박스점	서울 강남:02-6002-3	서울특별시	127.059	37.5125	정좌표	
7	6	압구정역점	서울시 강'02-541-5	서울특별시	127.037	37.5283	정좌표	
8	7	강남에스점	서울시 서:02-593-5C	서울특별시?	?	?	실패	
9	8	청담에스점	서울시 강'02-548-6C	서울특별시	127.047	37.5253	정좌표	
10	9	신사점	서울시 강'02-548-27	서울특별시	127.022	37.517	정좌표	
11	10	압구정역점	서울시 강'02-544-6	서울특별시	127.029	37.524	정좌표	

그림 9-22 좌표 변환을 하지 못한 항목을 복사하여 CB\_geo.shp.csv에 붙여넣기



## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체에 커피 매장 위치 표시하기

5. CB\_geo.shp.csv 파일에서 address2에 적힌 주소를 구글맵에서 검색하여 경도와 위도 좌표를 찾음

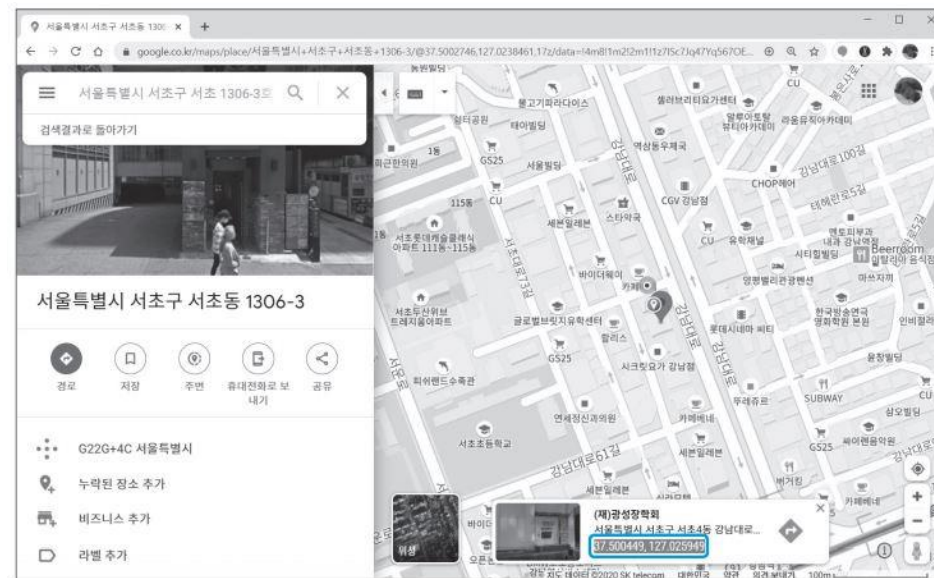


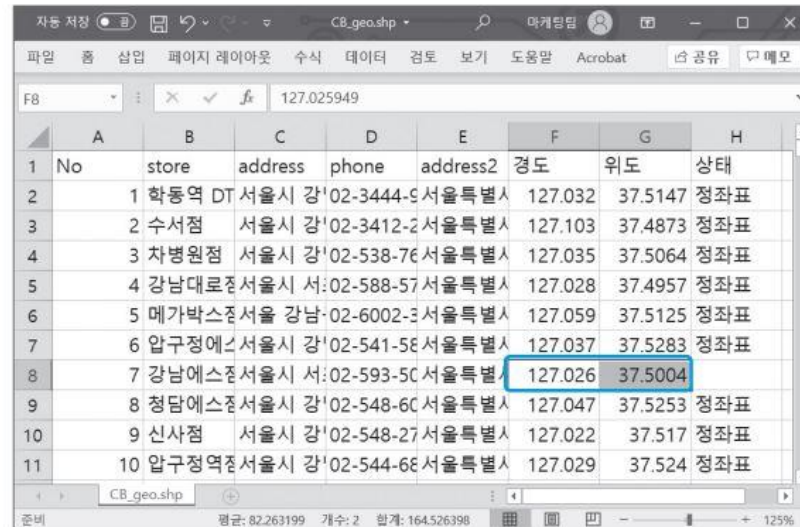
그림 9-23 구글맵에서 주소를 검색하여 찾은 위도와 경도 좌표 복사

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ■ 지도 객체에 커피 매장 위치 표시하기

6. 복사한 위도와 경도를 CB\_geo.shp.csv 파일의 경도와 위도에 붙여넣음



	A	B	C	D	E	F	G	H
1	No	store	address	phone	address2	경도	위도	상태
2	1	학동역 DT	서울시 강'02-3444-9	서울특별시	127.032	37.5147	정좌표	
3	2	수서점	서울시 강'02-3412-2	서울특별시	127.103	37.4873	정좌표	
4	3	차병원점	서울시 강'02-538-76	서울특별시	127.035	37.5064	정좌표	
5	4	강남대로점	서울시 서:02-588-57	서울특별시	127.028	37.4957	정좌표	
6	5	메가박스점	서울 강남:02-6002-3	서울특별시	127.059	37.5125	정좌표	
7	6	압구정에스점	서울시 강'02-541-58	서울특별시	127.037	37.5283	정좌표	
8	7	강남에스점	서울시 서:02-593-5C	서울특별시	127.026	37.5004		
9	8	청담에스점	서울시 강'02-548-6C	서울특별시	127.047	37.5253	정좌표	
10	9	신사점	서울시 강'02-548-27	서울특별시	127.022	37.517	정좌표	
11	10	압구정역점	서울시 강'02-544-68	서울특별시	127.029	37.524	정좌표	

그림 9-24 누락된 주소의 좌표를 검색하여 추가

7. 좌표 검색 과정을 반복하여 모든 좌표가 추가되었다면 CB\_geo.shp\_2.csv 파일로 저장

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

#### ▪ 지도 객체에 커피 매장 위치 표시하기

##### 8. 지도에 매장 위치 표시하기

In [11]:	CB_geoData = pd.read_csv('9장_data/CB_geo.shp_2.csv', encoding = 'cp949', engine = 'python')
In [12]:	map_CB = folium.Map(location = [37.560284, 126.975334], zoom_start = 15)
In [13]:	for i, store in CB_geoData.iterrows(): folium.Marker(location = [store['위도'], store['경도']], popup = store['store'], icon = folium.Icon(color = 'red', icon = 'star')).add_to(map_CB)
In [14]:	map_CB.save('data/map_CB.html')
In [15]:	<b>import webbrowser</b> webbrowser.open('data/map_CB.html')

In [11]: CB\_geo.shp\_2.csv 파일을 CB\_geoData 객체로 로드

In [12]: 지도 객체인 map\_CB를 생성

In [13]: for 반복문을 사용하여 CB\_geoData 객체에 있는 매장 정보를 하나씩 읽음

매장에 대한 마커의 팝업 글자는 매장 이름으로 설정하고 `popup = store['store']`, 마커 모양은 빨간색 별 모양으로 설정하여 `icon = 'star'` 마커를 만든 뒤 `folium.Marker()` 지도 객체 `map_CB`에 추가 `add_to(map_CB)`.

In [14]: 완성된 지오맵을 저장

## 2. 지리 정보 분석 후 맵 생성하기

### ■ 분석 모델 구축 및 시각화

- 지도 객체에 커피 매장 위치 표시하기

- 8. 지도에 매장 위치 표시하기

```
In [15]: import webbrowser  
webbrowser.open('C:/Users/kmj/My_Python/9장_data/map_CB.html')
```

In [15]: 저장한 지오맵 파일을 웹 브라우저에서 열어 확인



그림 9-25 커피 매장의 지오맵

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 미리보기

행정구역별 의료기관 현황 분석하기	
목표	행정구역별로 공공보건의료기관 수를 파악하고 인구수 대비 공공보건의료기관 비율을 비교 분석한다. 분석 결과는 블록맵으로 시각화한다.
핵심 개념	블록맵
데이터 수집	1. 공공보건의료기관현황.csv: 공공데이터포털에서 다운로드 2. 행정구역_시군구_별__성별_인구수_2.xlsx: 9장 01 절의 프로젝트에서 정리한 파일
데이터 준비 및 탐색	1. 행정구역 이름으로 주소 수정 2. 행정구역별 공공보건의료기관 수 집계 3. 행정구역별 인구수 데이터 정리 4. 테이블에 필요한 컬럼 추출 후 테이블 병합
분석 모델 구축 및 결과 시각화	

1. 행정구역별 공공보건의료기관 수



2. 행정구역별 인구수 대비 공공보건의료기관 비율



### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 목표설정

1. 행정구역별로 공공보건의료기관 수를 파악
2. 행정구역 별로 인구수 대비 공공보건의료기관 비율을 비교 분석

#### ■ 핵심 개념 이해

##### ■ 블록맵

- 구역의 경계선을 단순화한 뒤 블록 형태로 그려서 지도를 나타내는 시각화 기법
- 행정구역별 데이터 크기를 시각화할 때 많이 사용

#### ■ 데이터 수집

##### ■ 전국 공공보건의료기관 현황 데이터

- 행정구역별 공공보건의료기관 수를 파악하고 인구수 대비 공공보건의료기관 비율을 비교 분석할 때 사용
- 공공데이터포털 사이트에서 다운로드

##### ■ 행정구역별 인구수 데이터

- 행정구역별 인구수 대비 공공보건의료기관 비율을 비교 분석할 때 사용하는 데이터
- 앞의 9장 01절의 프로젝트에서 정리한 '행정구역\_시군구\_별\_성별\_인 구수\_2.xlsx' 파일을 사용

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 수집

##### ▪ 전국 공공보건의료기관 현황 데이터 수집하기

1. 공공데이터포털 사이트(www.data.go.kr)에서 '공공보건 의료기관 현황'으로 검색



그림 9-26 '공공보건 의료기관 현황'으로 검색한 데이터 목록

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 수집

- 전국 공공보건의료기관 현황 데이터 수집하기
  2. 파일데이터 상세 페이지가 나타나면 버튼을 클릭

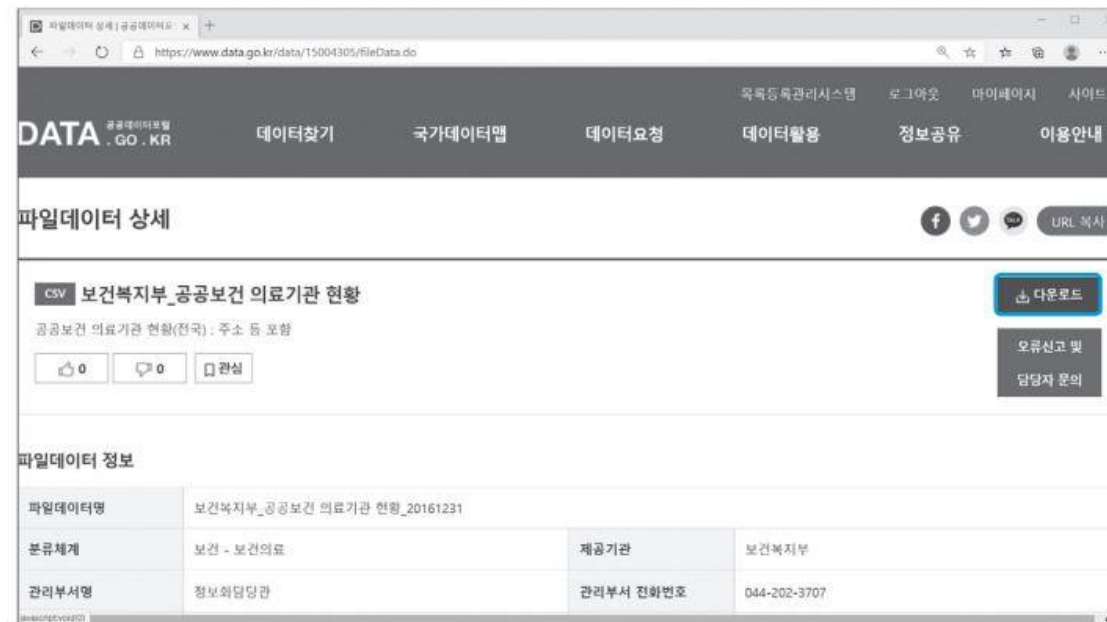


그림 9-27 공공보건의료기관 현황 데이터 다운로드

2. 다운로드한 파일은 파일명을 '공공보건의료기관현황.csv'로 바꾼 뒤 9장\_data 폴더에 저장



### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

- 1. 데이터 파일 확인하기

In [1]:

import pandas as pd  
pd.set\_option('mode.chained\_assignment', None)  
import numpy as np  
  
data = pd.read\_csv('data/공공보건의료기관현황.csv', index\_col = 0,  
                  encoding = ' CP949', engine = 'python')  
data.head() #작업 확인용 출력

Out[1]:

	병원명	설립 형태	근거 법령	관계 행정 기관	관계 공 공단체	심평원 요양 기관번호	종별 구분	병 상 수	소재지 우편번호	주소	홈페이지	대표 전화	FAX	비고
연														
1	강원도 재 활병원	시도립	강원도재활병원설치 및 운영에 관한 조례	강원도	해당없 음	32200641	병원	165	24227	강원도 춘천시 충열로 142번길 24-16	www.grh.or.kr	033- 248- 7700	033- 248- 7723	NaN
2	강원도 삼 척의료원	특수법 인	지방의료원의 설립 및 운영에 관한 법률	보건복지 부(강원 도)	지방의 료원	32100060	종합 병원	152	25920	강원도 삼척시 오성천 로 418	http://ksmc.or.kr	033- 572- 1141	033- 573- 8424	NaN
3	강원도 영 월의료원	특수법 인	지방의료원의 설립 및 운영에 관한 법률	보건복지 부(강원 도)	지방의 료원	32100078	종합 병원	214	26234	강원도 영월군 영월읍 중앙1로 59	http://www.youngwol.org	033- 370- 9117	033- 370- 9137	NaN
4	강원도 원 주의료원	특수법 인	지방의료원의 설립 및 운영에 관한 법률	보건복지 부(강원 도)	지방의 료원	32100086	종합 병원	237	26448	강원도 원주시 서원대 로 387(개운동)	www.kwmc.or.kr	033- 760- 4500	033- 761- 5121	NaN
5	강원도 강 릉의료원	특수법 인	지방의료원의 설립 및 운영에 관한 법률	보건복지 부(강원 도)	지방의 료원	32100159	종합 병원	137	25535	강원도 강릉시 경강로 2007(남문동 164-1)	http://www.gnmc.or.kr	033- 646- 6910	033- 610- 1415	NaN

In [1]: '공공보건의료기관현황.csv' 파일을 data 객체로 로드하고, 상위 다섯 개 행의 데이터를 출력하여 data.head() 확인

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ■ 공공보건의료기관 현황 데이터 준비하기

##### 2. 주소 정리하기

In [2]:	<pre># 주소에서 시도, 군구 정보 분리 addr = pd.DataFrame(data['주소'].apply(lambda v: v.split()[2:]).tolist(), columns = ('시도', '군구')) addr.head() #작업 확인용 출력</pre>																		
Out[2]:	<table><tr><th></th><th>시도</th><th>군구</th></tr><tr><td>0</td><td>강원도</td><td>춘천시</td></tr><tr><td>1</td><td>강원도</td><td>삼척시</td></tr><tr><td>2</td><td>강원도</td><td>영월군</td></tr><tr><td>3</td><td>강원도</td><td>원주시</td></tr><tr><td>4</td><td>강원도</td><td>강릉시</td></tr></table>		시도	군구	0	강원도	춘천시	1	강원도	삼척시	2	강원도	영월군	3	강원도	원주시	4	강원도	강릉시
	시도	군구																	
0	강원도	춘천시																	
1	강원도	삼척시																	
2	강원도	영월군																	
3	강원도	원주시																	
4	강원도	강릉시																	

In [2]: data 객체에서 ['주소'] 컬럼의 값을 띄어쓰기를 기준으로 분리하여split(), 시군과 군구 정보에 해당하는 0~1번 컬럼 [:2]을 추출해서 컬럼 이름을 '시도', '군구'로 나타내고columns=(' 시도', '군구'), 데이터프레임 객체인 addr을 생성  
생성된 addr 객체의 내용을 출력하 여addr.head( ) 확인

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 3. 시도 이름에서 잘못된 내용이 있는지 확인

In [3]:	addr['시도'].unique()
Out[3]:	array(['강원도', '경기도', '경기', '경남', '창원시', '경상남도', '경상북도', '경산시', '경북', '인천광역시', '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도', '부산광역시', '전라북도', '충북', '서울특별시', '서울시', '부산특별시', '대전시', '충남', '전남', '충청남도', '울산광역시', '전북', '천안시', '충청북도'], dtype = object)

In [3]: addr 객체의 ['시도'] 컬럼 값에서 고유값을 확인 `addr['시도'].unique()`.

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▫ 공공보건의료기관 현황 데이터 준비하기

##### 4. 잘못된 위치를 찾아서 값을 수정(창원시)

In [4]:	addr[addr['시도'] == '창원시']										
Out[4]:	<table><thead><tr><th></th><th>시도</th><th>군구</th></tr></thead><tbody><tr><td>27</td><td>창원시</td><td>의창구</td></tr><tr><td>31</td><td>창원시</td><td>마산합포구3.15대로</td></tr></tbody></table>			시도	군구	27	창원시	의창구	31	창원시	마산합포구3.15대로
	시도	군구									
27	창원시	의창구									
31	창원시	마산합포구3.15대로									
In [5]:	addr.iloc[27] = ['경상남도', '창원시'] addr.iloc[31] = ['경상남도', '창원시']										
In [6]:	addr.iloc[27]										
Out[6]:	시도    경상남도 군구    창원시 Name: 27, dtype: object										
In [7]:	addr.iloc[31]										
Out[7]:	시도    경상남도 군구    창원시 Name: 31, dtype: object										

In [4]: ['시도'] 컬럼 값이 '창원시'로 되어 있는 행 번호를 찾아보니 27번과 31번

In [5]: 27번과 31번의 값을 ['경상남도', '창원시']로 수정

In [6],[7]: 수정한 내용을 확인

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 5. 잘못된 위치를 찾아서 값을 수정(경산시,천안시)

In [8]:	addr[addr['시도'] == '경산시']									
Out[8]:	<table><tr><th></th><th>시도</th><th>군구</th></tr><tr><td>47</td><td>경산시</td><td>경안로</td></tr></table>		시도	군구	47	경산시	경안로			
	시도	군구								
47	경산시	경안로								
In [9]:	addr.iloc[27] = ['경상남도', '경산시']									
In [10]:	addr[addr['시도'] == '천안시']									
Out[10]:	<table><tr><th></th><th>시도</th><th>군구</th></tr><tr><td>209</td><td>천안시</td><td>동남구</td></tr><tr><td>210</td><td>천안시</td><td>동남구</td></tr></table>		시도	군구	209	천안시	동남구	210	천안시	동남구
	시도	군구								
209	천안시	동남구								
210	천안시	동남구								
In [11]:	addr.iloc[209] = ['충청남도', '천안시'] addr.iloc[210] = ['충청남도', '천안시']									

##### 6. 다시 addr 객체의 ['시도'] 컬럼 값에서 수정할 내용이 있는지 확인

In [12]:	addr['시도'].unique()
Out[12]:	array(['강원도', '경기도', '경기', '경남', '경상남도', '경상북도', ' 경북', '인천광역시', '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도', '부산광역시', '전라북도', '충북', '서울특별시', '서울시', '부산특별시', '대전시', '충남', '전남', '충청남도', '울산광역시', '전북', '천안시', '충청북도'], dtype = object)

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 7. '경기', '경남'과 같이 축약된 이름을 정확한 표준 이름으로 수정

In [13]:	<pre>addr_aliases = {'경기':'경기도', '경남':'경상남도', '경북':'경상북도',                '충북':'충청북도', '서울시':'서울특별시', '부산특별시':                '부산광역시', '대전시':'대전광역시', '충남':'충청남도',                '전남':'전라남도', '전북':'전라북도'}</pre>
In [14]:	<pre>addr['시도'] = addr['시도'].apply(lambda v: addr_aliases.get(v, v))</pre>
In [15]:	<pre>addr['시도'].unique()</pre>
Out[15]:	<pre>array(['강원도', '경기도', '경상남도', '경상북도', '인천광역시',        '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도',        '부산광역시', '전라북도', '충청북도', '서울특별시', '충청남도',        '울산광역시'], dtype = object)</pre>

In [13]: 변경할 이름에 대한 '축약이름:표준이름'의 addr\_aliases 딕셔너리를 정의

In [14]: addr\_aliases 딕셔너리를 적용하여 ['시도'] 컬럼의 값을 변경

In [15]: addr 객체의 ['시도'] 컬럼 고유값을 출력하여 빠짐없이 변경되었는지 확인

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 8. ['군구'] 컬럼에서 정리할 사항이 있는지 탐색

In [16]:	addr['군구'].unique()
Out[16]:	array(['춘천시', '삼척시', '영월군', '원주시', '강릉시', '속초시', '정선군', '수원시', '이천시', '안성시', '의정부시', '포천시', '파주시', '용인시', '평택시', '시흥시', '여주시', '남양주시', '동두천시', '안산시', '부천시', '통영시', '사천시', '창원시', '김해시', '양산시', '거창군', '남해군', '의령군', '포항시', '김천시', '안동시', '울진군', '경주시', '구미시', '영주시', '상주시', '문경시', '경산시', '의성군', '청도군', '고령군', '칠곡군', '봉화군', '울릉군', '부평구', '북구', '순천시', '대덕구', '태백시', '동해시', '화성시', '광산구', '남구', '중구', '아란13길', '서구', '전주시', '진주시', '청주시', '종로구', '성남시', '동구', '화순군', '강동구', '사상구', '달서구', '해운대구', '유성구', '가평군', '양주시', '고양시', '홍천군', '양구군', '청원군', '계룡시', '논산시', '함평군', '양평군', '수성구', '달성군', '연수구', '노원구', '기장군', '공주시', '강북구', '광진구', '나주시', '창녕군', '목포시', '고흥군', '연제구', '동매로', '서초구', '은평구', '중랑구', '강남구', '동작구', '동대문구', '양천구', '성동구', '송파구', '울주군', '계양구', '옹진군', '보성군', '광양시', '영광군', '무안군', '진도군', '강진군', '곡성군', '여수시', '신안군', '장성군', '완주군', '부안군', '정읍시', '남원시', '군산시', '고창군', '진안군', '제주시', '서귀포시', '천안시', '보령시', '서산시', '서천군', '홍성군', '제천시', '충주시', '영동군', '단양군'], dtype=object)

In [16]: addr 객체의 ['군구'] 컬럼의 고유값을 확인

'아란13길'을 인터넷에서 검색해보면 제주시에 있는 도로명이므로 '제주시'로 수정

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▫ 공공보건의료기관 현황 데이터 준비하기

##### 8. ['군구'] 컬럼에서 정리할 사항이 있는지 탐색

In [17]:	addr['군구'].unique()						
Out[17]:	<table><tr><th></th><th>시도</th><th>군구</th></tr><tr><td>75</td><td>제주특별자치도</td><td>아란13길</td></tr></table>		시도	군구	75	제주특별자치도	아란13길
	시도	군구					
75	제주특별자치도	아란13길					
In [18]:	addr.iloc[75] = ['제주특별자치도', '제주시']						

In [17]: ['군구'] 컬럼 값이 '아란13길'로 되어 있는 행 번호를 찾아보니 75번

In [18]: 75번 행의 값을 ['제주특별자치도', '제주시']로 수정



### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 9. 행정구역별 공공보건의료기관의 수 구하기

In [19]:	<pre>addr['시도군구'] = addr.apply(lambda r: r['시도'] + ' ' + r['군구'], axis = 1) addr.head()  #작업 확인용 출력</pre>		
Out[19]:	시도	군구	시도군구
0	강원도	춘천시	강원도 춘천시
1	강원도	삼척시	강원도 삼척시
2	강원도	영월군	강원도 영월군
3	강원도	원주시	강원도 원주시
4	강원도	강릉시	강원도 강릉시

In [19]: ['시도']와 ['군구'] 컬럼 값을 연결하여 만든 값으로 addr 객체에 새로운 ['시도군구'] 컬럼을 추가

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

- 9. 행정구역별 공공보건의료기관의 수 구하기

In [20]:	addr['count'] = 0 addr.head() #작업 확인용 출력				
Out[20]:		시도	군구	시도군구	count
	0	강원도	춘천시	강원도 춘천시	0
	1	강원도	삼척시	강원도 삼척시	0
	2	강원도	영월군	강원도 영월군	0
	3	강원도	원주시	강원도 원주시	0
	4	강원도	강릉시	강원도 강릉시	0

In [20]: addr 객체에 ['count'] 컬럼을 추가

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 9. 행정구역별 공공보건의료기관의 수 구하기

In [21]:	addr['count'] = 0 addr.head() #작업 확인용 출력				
Out[21]:		시도	군구	시도군구	count
	0	강원도	강릉시	강원도 강릉시	4
	1	강원도	동해시	강원도 동해시	1
	2	강원도	삼척시	강원도 삼척시	1
	3	강원도	속초시	강원도 속초시	1
	4	강원도	양구군	강원도 양구군	1

In [21]: ['시도'], ['군구'], ['시도군구'] 컬럼을 기준으로 그룹을 만들 `addr.groupby(['시도', '군구', '시도군구'], as_index = False).`

그룹별 원소의 개수를 구하여 `count()` ['count'] 컬럼에 저장

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 공공보건의료기관 현황 데이터 준비하기

##### 10. 데이터 병합에 사용할 인덱스를 설정

In [22]:

addr\_group = addr\_group.set\_index("시도군구")  
addr\_group.head() #작업 확인용 출력

Out[22]:

	시도	군구	count
시도군구			
	강원도 강릉시	강원도 강릉시	4
	강원도 동해시	강원도 동해시	1
	강원도 삼척시	강원도 삼척시	1
	강원도 속초시	강원도 속초시	1
	강원도 양구군	강원도 양구군	1

In [22]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▫ 행정구역별 인구수 데이터 준비하기

###### 1. 데이터 정리하기

In [23]:

population = pd.read\_excel('data/행정구역\_시군구\_별\_성별\_인구수\_2.xlsx', encoding = 'CP949')  
population.head() #작업 확인용 출력

Out[23]:

	행정구역(시군구)별(1)	행정구역(시군구)별(2)	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
0	전국	소계	51847509	25862863	25984646
1	서울특별시	소계	9733509	4745088	4988421
2	서울특별시	종로구	151215	73688	77527
3	서울특별시	중구	126201	61946	64255
4	서울특별시	용산구	229385	110701	118684

In [24]:

population = population.rename(columns = {'행정구역(시군구)별(1)': '시도', '행정구역(시군구)별(2)': '군구'})  
population.head() #작업 확인용 출력

Out[24]:

	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
0	전국	소계	51847509	25862863	25984646
1	서울특별시	소계	9733509	4745088	4988421
2	서울특별시	종로구	151215	73688	77527
3	서울특별시	중구	126201	61946	64255
4	서울특별시	용산구	229385	110701	118684

In [23]: '행정구역\_시군구\_별\_성별\_인구수\_2.xlsx' 파일을 population 객체로 로드하고, 출력하여 확인

In [24]: rename() 함수를 사용하여 컬럼 이름을 변경

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▫ 행정구역별 인구수 데이터 준비하기

2. ['군구'] 컬럼에 포함되어 있는 왼쪽 띄어쓰기 공백을 제거

['시도군구'] 컬럼을 만들고 addr\_group과 병합하기 위해 인덱스로 설정

In [25]:	for element in range(0,len(population)): population['군구'][element] = population['군구'][element].strip()																																										
In [26]:	population['시도군구'] = population.apply(lambda r: r['시도'] + ' ' + r['군구'], axis = 1) population.head() <i>#작업 확인용 출력</i>																																										
Out[26]:	<table><tr><th></th><th>시도</th><th>군구</th><th>총인구수 (명)</th><th>남자인구수 (명)</th><th>여자인구수 (명)</th><th>시도군구</th></tr><tr><td>0</td><td>전국</td><td>소계</td><td>51847509</td><td>25862863</td><td>25984646</td><td>전국 소계</td></tr><tr><td>1</td><td>서울특별시</td><td>소계</td><td>9733509</td><td>4745088</td><td>4988421</td><td>서울특별시 소계</td></tr><tr><td>2</td><td>서울특별시</td><td>종로구</td><td>151215</td><td>73688</td><td>77527</td><td>서울특별시 종로구</td></tr><tr><td>3</td><td>서울특별시</td><td>중구</td><td>126201</td><td>61946</td><td>64255</td><td>서울특별시 중구</td></tr><tr><td>4</td><td>서울특별시</td><td>용산구</td><td>229385</td><td>110701</td><td>118684</td><td>서울특별시 용산구</td></tr></table>		시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	시도군구	0	전국	소계	51847509	25862863	25984646	전국 소계	1	서울특별시	소계	9733509	4745088	4988421	서울특별시 소계	2	서울특별시	종로구	151215	73688	77527	서울특별시 종로구	3	서울특별시	중구	126201	61946	64255	서울특별시 중구	4	서울특별시	용산구	229385	110701	118684	서울특별시 용산구
	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	시도군구																																					
0	전국	소계	51847509	25862863	25984646	전국 소계																																					
1	서울특별시	소계	9733509	4745088	4988421	서울특별시 소계																																					
2	서울특별시	종로구	151215	73688	77527	서울특별시 종로구																																					
3	서울특별시	중구	126201	61946	64255	서울특별시 중구																																					
4	서울특별시	용산구	229385	110701	118684	서울특별시 용산구																																					

In [25]: ['군구'] 컬럼의 문자열 앞뒤에 포함된 띄어쓰기 공백을 모두 제거 `strip()`

In [26]: ['시도']와 ['군구'] 컬럼 값을 연결하여 새로운 ['시도군구'] 컬럼에 추가

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▫ 행정구역별 인구수 데이터 준비하기

2. ['군구'] 컬럼에 포함되어 있는 왼쪽 띄어쓰기 공백을 제거

['시도군구'] 컬럼을 만들고 addr\_group과 병합하기 위해 인덱스로 설정

In [27]:	population = population[population.군구 != '소계']																																										
In [28]:	population = population.set_index("시도군구")  population.head() #작업 확인용 출력																																										
Out[28]:	<table><tr><th></th><th>시도</th><th>군구</th><th>총인구수 (명)</th><th>남자인구수 (명)</th><th>여자인구수 (명)</th></tr><tr><td></td><td colspan="5">시도군구</td></tr><tr><td>서울특별시 종로구</td><td>서울특별시</td><td>종로구</td><td>151215</td><td>73688</td><td>77527</td></tr><tr><td>서울특별시 중구</td><td>서울특별시</td><td>중구</td><td>126201</td><td>61946</td><td>64255</td></tr><tr><td>서울특별시 용산구</td><td>서울특별시</td><td>용산구</td><td>229385</td><td>110701</td><td>118684</td></tr><tr><td>서울특별시 성동구</td><td>서울특별시</td><td>성동구</td><td>300410</td><td>147020</td><td>153390</td></tr><tr><td>서울특별시 광진구</td><td>서울특별시</td><td>광진구</td><td>351263</td><td>170164</td><td>181099</td></tr></table>		시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)		시도군구					서울특별시 종로구	서울특별시	종로구	151215	73688	77527	서울특별시 중구	서울특별시	중구	126201	61946	64255	서울특별시 용산구	서울특별시	용산구	229385	110701	118684	서울특별시 성동구	서울특별시	성동구	300410	147020	153390	서울특별시 광진구	서울특별시	광진구	351263	170164	181099
	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)																																						
	시도군구																																										
서울특별시 종로구	서울특별시	종로구	151215	73688	77527																																						
서울특별시 중구	서울특별시	중구	126201	61946	64255																																						
서울특별시 용산구	서울특별시	용산구	229385	110701	118684																																						
서울특별시 성동구	서울특별시	성동구	300410	147020	153390																																						
서울특별시 광진구	서울특별시	광진구	351263	170164	181099																																						

In [27]: ['군구'] 컬럼 값이 '소계'인 행은 필요 없으므로 제외

In [28]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ■ 행정구역별 인구수 데이터 준비하기

##### 3. addr\_group과 population을 인덱스 기준으로 병합

- 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [29]:

addr\_population\_merge = pd.merge(addr\_group,population, how = 'inner',  
left\_index = True, right\_index = True)

addr\_population\_merge.head() #작업 확인용 출력

Out[29]:

시도군구				count	시도_y	군구_y	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
강원도	강릉시	강원도	강릉시	4	강원도	강릉시	213328	105775	107553
강원도	동해시	강원도	동해시	1	강원도	동해시	90417	45782	44635
강원도	삼척시	강원도	삼척시	1	강원도	삼척시	66806	33811	32995
강원도	속초시	강원도	속초시	1	강원도	속초시	81840	40470	41370
강원도	양구군	강원도	양구군	1	강원도	양구군	22526	11937	10589

In [29]: addr\_group과 population을 내부병합으로 병합



### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ■ 행정구역별 인구수 데이터 준비하기

##### 3. addr\_group과 population을 인덱스 기준으로 병합

- 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [30]:	<pre>local_MC_Population = addr_population_merge[['시도_x', '군구_x',   'count', '총인구수 (명)']]  local_MC_Population.head() #작업 확인용 출력</pre>																												
Out[30]:	<table><tr><th>시도_x</th><th>군구_x</th><th>count</th><th>총인구수 (명)</th></tr><tr><td colspan="4">시도군구</td></tr><tr><td>강원도 강릉시</td><td>강원도 강릉시</td><td>4</td><td>213328</td></tr><tr><td>강원도 동해시</td><td>강원도 동해시</td><td>1</td><td>90417</td></tr><tr><td>강원도 삼척시</td><td>강원도 삼척시</td><td>1</td><td>66806</td></tr><tr><td>강원도 속초시</td><td>강원도 속초시</td><td>1</td><td>81840</td></tr><tr><td>강원도 양구군</td><td>강원도 양구군</td><td>1</td><td>22526</td></tr></table>	시도_x	군구_x	count	총인구수 (명)	시도군구				강원도 강릉시	강원도 강릉시	4	213328	강원도 동해시	강원도 동해시	1	90417	강원도 삼척시	강원도 삼척시	1	66806	강원도 속초시	강원도 속초시	1	81840	강원도 양구군	강원도 양구군	1	22526
시도_x	군구_x	count	총인구수 (명)																										
시도군구																													
강원도 강릉시	강원도 강릉시	4	213328																										
강원도 동해시	강원도 동해시	1	90417																										
강원도 삼척시	강원도 삼척시	1	66806																										
강원도 속초시	강원도 속초시	1	81840																										
강원도 양구군	강원도 양구군	1	22526																										

In [30]: 필요한 컬럼 4개만 추출하여 local\_MC\_Population 객체를 생성

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 데이터 준비 및 탐색

##### ▪ 행정구역별 인구수 데이터 준비하기

##### 3. addr\_group과 population을 인덱스 기준으로 병합

- 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [31]:	<pre>#컬럼 이름 변경 local_MC_Population = local_MC_Population.rename(columns = {'시도_x':  '시도', '군구_x': '군구','총인구수 (명)': '인구수'}) MC_count = local_MC_Population['count'] local_MC_Population['MC_ratio'] = MC_count.div(local_MC_Population['  인구수'], axis = 0)*100000 local_MC_Population.head() #작업 확인용 출력</pre>																																										
Out[31]:	<table><tr><th></th><th>시도</th><th>군구</th><th>count</th><th>인구수</th><th>MC_ratio</th></tr><tr><td></td><td colspan="5">시도군구</td></tr><tr><td></td><td>강원도</td><td>강릉시</td><td>강원도</td><td>강릉시</td><td>4 213328 1.875047</td></tr><tr><td></td><td>강원도</td><td>동해시</td><td>강원도</td><td>동해시</td><td>1 90417 1.105987</td></tr><tr><td></td><td>강원도</td><td>삼척시</td><td>강원도</td><td>삼척시</td><td>1 66806 1.496872</td></tr><tr><td></td><td>강원도</td><td>속초시</td><td>강원도</td><td>속초시</td><td>1 81840 1.221896</td></tr><tr><td></td><td>강원도</td><td>양구군</td><td>강원도</td><td>양구군</td><td>1 22526 4.439315</td></tr></table>		시도	군구	count	인구수	MC_ratio		시도군구						강원도	강릉시	강원도	강릉시	4 213328 1.875047		강원도	동해시	강원도	동해시	1 90417 1.105987		강원도	삼척시	강원도	삼척시	1 66806 1.496872		강원도	속초시	강원도	속초시	1 81840 1.221896		강원도	양구군	강원도	양구군	1 22526 4.439315
	시도	군구	count	인구수	MC_ratio																																						
	시도군구																																										
	강원도	강릉시	강원도	강릉시	4 213328 1.875047																																						
	강원도	동해시	강원도	동해시	1 90417 1.105987																																						
	강원도	삼척시	강원도	삼척시	1 66806 1.496872																																						
	강원도	속초시	강원도	속초시	1 81840 1.221896																																						
	강원도	양구군	강원도	양구군	1 22526 4.439315																																						

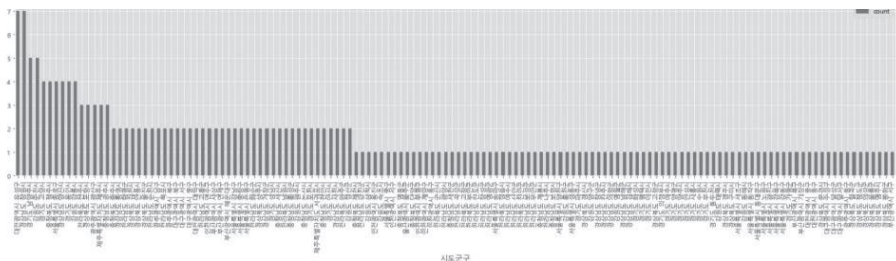
In [31]: 인구수 대비 공공보건의료기관 비율을 구하여 local\_MC\_Population의 ['MC\_ratio'] 컬럼에 추가

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 바 차트 그리기

##### 1. 행정구역별 공공보건의료기관 수에 대한 바 차트를 그리기

In [32]:	<pre>from matplotlib import pyplot as plt from matplotlib import rcParams, style style.use('ggplot')  from matplotlib import font_manager, rc font_name = font_manager.FontProperties(fname = "c:/Windows/Fonts/  malgun.ttf").get_name() rc('font', family = font_name)</pre>
In [33]:	<pre>MC_ratio = local_MC_Population[['count']] MC_ratio = MC_ratio.sort_values('count', ascending = False) plt.rcParams["figure.figsize"] = (25, 5) MC_ratio.plot(kind = 'bar', rot = 90) plt.show()</pre>
Out[33]:	

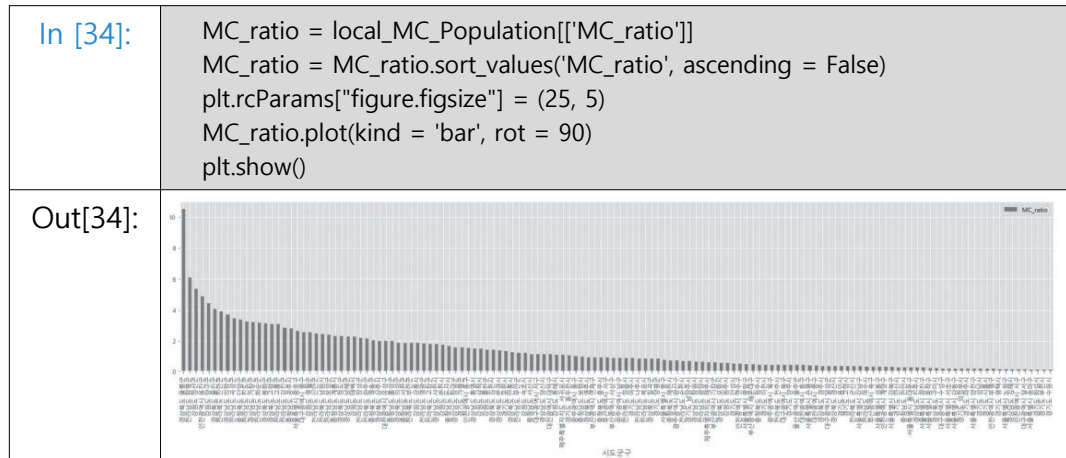
In [33]: local\_MC\_Population 객체의 ['count'] 컬럼 값을 오름차순으로 정렬하여, 행정 구역별 공공보건의료기관 수에 대한 바 차트를 그림

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 바 차트 그리기

##### 2. 행정구역별로 인구수 대비 공공보건의료기관 비율에 대한 바 차트를 그리기



In [34]: local\_MC\_Population 객체의 ['MC\_ratio'] 컬럼 값을 오름차순으로 정렬하여, 행정구역별로 인구수 대비 공공보건의료기관 비율에 대한 바 차트를 그

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ▪ 블록맵으로 시각화하기

##### 1. 데이터 준비하기

In [35]:	<pre>import os path = os.getcwd()</pre>																																																
In [36]:	<pre>data_draw_korea = pd.read_csv(path+'WW9장_dataWWdata_draw_korea.csv',                                 index_col = 0, encoding = 'UTF-8', engine = 'python')  data_draw_korea.head() #작업 확인용 출력</pre>																																																
Out[36]:	<table><tr><th></th><th>인구수</th><th>shortName</th><th>x</th><th>y</th><th>면적</th><th>광역시도</th><th>행정구역</th></tr><tr><td>0</td><td>202520</td><td>강릉</td><td>11</td><td>4</td><td>1040.07</td><td>강원도</td><td>강릉시</td></tr><tr><td>1</td><td>25589</td><td>고성(강원)</td><td>9</td><td>0</td><td>664.19</td><td>강원도</td><td>고성군</td></tr><tr><td>2</td><td>86747</td><td>동해</td><td>11</td><td>5</td><td>180.01</td><td>강원도</td><td>동해시</td></tr><tr><td>3</td><td>63986</td><td>삼척</td><td>11</td><td>8</td><td>1185.80</td><td>강원도</td><td>삼척시</td></tr><tr><td>4</td><td>76733</td><td>속초</td><td>9</td><td>1</td><td>105.25</td><td>강원도</td><td>속초시</td></tr></table>		인구수	shortName	x	y	면적	광역시도	행정구역	0	202520	강릉	11	4	1040.07	강원도	강릉시	1	25589	고성(강원)	9	0	664.19	강원도	고성군	2	86747	동해	11	5	180.01	강원도	동해시	3	63986	삼척	11	8	1185.80	강원도	삼척시	4	76733	속초	9	1	105.25	강원도	속초시
	인구수	shortName	x	y	면적	광역시도	행정구역																																										
0	202520	강릉	11	4	1040.07	강원도	강릉시																																										
1	25589	고성(강원)	9	0	664.19	강원도	고성군																																										
2	86747	동해	11	5	180.01	강원도	동해시																																										
3	63986	삼척	11	8	1185.80	강원도	삼척시																																										
4	76733	속초	9	1	105.25	강원도	속초시																																										

In [35]: 현재 사용 중인 폴더(디렉토리)의 경로를 구하여 path에 저장

In [36]: data\_draw\_korea.csv 파일을 로드하여 data\_draw\_korea 객체에 저장하고, 출력하여 내용을 확인

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 블록맵으로 시각화하기

##### 2. 행정구역 이름 매핑하기

In [37]:	<pre>data_draw_korea['시도군구'] = data_draw_korea.apply(lambda r: r['광역 시도'] + ' ' + r['행정구역'], axis = 1)</pre>
In [38]:	<pre>data_draw_korea = data_draw_korea.set_index("시도군구")  data_draw_korea.head() #작업 확인용 출력</pre>
Out[38]:	<pre>      인구수  shortName  x  y   면적  광역시도  행정구역 시도군구 강원도 강릉시  202520   강릉  11  4  1040.07  강원도  강릉시 강원도 고성군  25589   고성(강원)  9  0   664.19  강원도  고성군 강원도 동해시  86747   동해  11  5   180.01  강원도  동해시 강원도 삼척시  63986   삼척  11  8  1185.80  강원도  삼척시 강원도 속초시  76733   속초  9  1   105.25  강원도  속초시</pre>

In [37]: ['광역시도']와 ['행정구역'] 컬럼 값을 연결하여 새로운 ['시도군구'] 컬럼으로 추가

In [38]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

- 2. 행정구역 이름 매핑하기

In [39]:

```
data_draw_korea_MC_Population_all = pd.merge(data_draw_korea,local_MC_Population, how = 'outer', left_index = True, right_index = True)

data_draw_korea_MC_Population_all.head()
```

Out[39]:

	인구수_x	shortName	x	y	면적	광역시도	행정구역	시도	군구	count	인구수_y	MC_ratio
시도군구												
강원도 강릉시	202520	강릉	11	4	1040.07	강원도	강릉시	강원도	강릉시	4.0	213328.0	1.875047
강원도 고성군	25589	고성(강원)	9	0	664.19	강원도	고성군	NaN	NaN	NaN	NaN	NaN
강원도 동해시	86747	동해	11	5	180.01	강원도	동해시	강원도	동해시	1.0	90417.0	1.105987
강원도 삼척시	63986	삼척	11	8	1185.80	강원도	삼척시	강원도	삼척시	1.0	66806.0	1.496872
강원도 속초시	76733	속초	9	1	105.25	강원도	속초시	강원도	속초시	1.0	81840.0	1.221896

In [39]: data\_draw\_korea와 local\_MC\_Population을 외부병합으로how='outer' 병합

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 블록맵으로 시각화하기

##### 3. 블록맵으로 시각화하기

```
In [40]: BORDER_LINES = [  
    [3, 2), (5, 2), (5, 3), (9, 3), (9, 1)], # 인천  
    [(2, 5), (3, 5), (3, 4), (8, 4), (8, 7), (7, 7), (7, 9), (4, 9), (4, 7), (1, 7)], # 서울  
    [(1, 6), (1, 9), (3, 9), (3, 10), (8, 10), (8, 9), (9, 9), (9, 8), (10, 8), (10, 5), (9, 5), (9, 3)], # 경기도  
    [(9, 12), (9, 10), (8, 10)], # 강원도  
    [(10, 5), (11, 5), (11, 4), (12, 4), (12, 5), (13, 5), (11, 8), (11, 9), (10, 9), (10, 8)], # 충청북도  
    [(14, 4), (15, 4), (15, 6)], # 대전시  
    [(14, 7), (14, 9), (13, 9), (13, 11), (13, 13)], # 경상북도  
    [(14, 8), (16, 8), (16, 10), (15, 10), (15, 11), (14, 11), (14, 12), (13, 12)], # 대구시  
    [(15, 11), (16, 11), (16, 13)], # 울산시  
    [(17, 1), (17, 3), (18, 3), (18, 6), (15, 6)], # 전라북도  
    [(19, 2), (19, 4), (21, 4), (21, 3), (22, 3), (22, 2), (19, 2)], # 광주시  
    [(18, 5), (20, 5), (20, 6)], # 전라남도  
    [(16, 9), (18, 9), (18, 8), (19, 8), (19, 9), (20, 9), (20, 10)], # 부산시  
]
```

In [40]: 블록맵의 행정구역 경계선을 그리기 위해 행정구역의 블록 위치인 x, y 데이터를 정의



### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 블록맵으로 시각화하기

##### 3. 블록맵으로 시각화하기

In [41]:

```
def draw_blockMap(blockedMap, targetData, title, color):
    whitelabelmin = (max(blockedMap[targetData]) -
                     min(blockedMap[targetData])) * 0.25 +
                     min(blockedMap[targetData])
    datalabel = targetData
    vmin = min(blockedMap[targetData])
    vmax = max(blockedMap[targetData])
    mapdata = blockedMap.pivot(index = 'y', columns = 'x', values = targetData)
    masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

    plt.figure(figsize = (8, 13))
    plt.title(title)
    plt.pcolor(masked_mapdata, vmin = vmin, vmax = vmax, cmap =
               color, edgecolor = '#aaaaaa', linewidth = 0.5)
    #지역 이름 표시
    for idx, row in blockedMap.iterrows():
        annocolor = 'white' if row[targetData] > whitelabelmin else
        'black'
        #광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시
        if row['광역시도'].endswith('시') and not row['광역시도'].
        startswith('세종'):
            dispname = '{}\n{}'.format(row['광역시도'][:2], row['행정구
            역'][:1])
            if len(row['행정구역']) <= 2:
                dispname += row['행정구역'][-1]
            else:
                dispname = row['행정구역'][:1]
            #서대문구, 서귀포시 같이 이름이 3자 이상이면 작은 글자로 표시
```

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ■ 블록맵으로 시각화하기

##### 3. 블록맵으로 시각화하기

```
In [41]: if len(dispname.splitlines()[-1]) >= 3:
          fontsize, linespacing = 9.5, 1.5
          else:
          fontsize, linespacing = 11, 1.2
          plt.annotate(dispname, (row['x']+0.5, row['y']+0.5), weight = 'bold',
          fontsize = fontsize, ha = 'center', va = 'center', color = annocolor,
          linespacing = linespacing)

          # 시도 경계를 그린다.
          for path in BORDER_LINES:
          ys, xs = zip(*path)
          plt.plot(xs, ys, c = 'black', lw = 4)
          plt.gca().invert_yaxis()
          # plt.gca().set_aspect(1)
          plt.axis('off')

          cb = plt.colorbar(shrink = 1, aspect = 10)
          cb.set_label(datalabel)
          plt.tight_layout()
          plt.savefig('..장_data' + 'blockMap_' + targetData + '.png')
          plt.show()
```

In [41]: 블록맵의 블록에 데이터를 매핑하고 색을 표시하여 블록맵을 그린 뒤 저장하는 함수를 정의

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

##### ▪ 블록맵으로 시각화하기

##### 4. 행정구역별 공공보건의료기관 수를 블록맵으로 시각화

In [42]:	<code>draw_blockMap(data_draw_korea_MC_Population_all, 'count', '행정구역 별 공공보건의료기관 수', 'Blues')</code>
----------	--

In [42]: data\_draw\_korea\_MC\_Population\_all 객체의 ['count'] 컬럼 값에 대해 Blues 색상 스펙트럼을 사용하여 블록맵 작성

##### 5. 인구수 대비 공공보건의료기관 비율을 블록맵으로 시각화

In [43]:	<code>draw_blockMap(data_draw_korea_MC_Population_all, 'MC_ratio', '행정구 역별 인구수 대비 공공보건의료기관 비율', 'Reds')</code>
----------	--

In [43]: data\_draw\_korea\_MC\_Population\_all 객체의 ['MC\_ratio'] 컬럼 값에 대해 Reds 색상 스펙트럼을 사용하여 블록맵 작성

### 3. 행정구역별 의료기관 현황 분석하기

#### ■ 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기 - 결과



그림 9-28 행정구역별 공공보건의료기관 수 블록맵(좌)과 인구수 대비 공공보건의료기관 비율 블록맵(우)