In [195]:
```
# Ignore the warnings
import warnings
warnings.filterwarnings('ignore')

# Data manipulation and visualization
import seaborn as sns
    # set grid
sns.set_style("whitegrid")

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
%matplotlib inline
from matplotlib import font_manager, rc
from matplotlib.ticker import FuncFormatter

plt.rcParams['font.family'] = 'Malgun Gothic'
plt.rc('axes', unicode_minus=False)
from matplotlib.ticker import ScalarFormatter

from haversine import haversine

import folium
import json

from scipy.spatial import Voronoi, voronoi_plot_2d

import geopandas
#!pip install descartes
import requests as rq
from bs4 import BeautifulSoup
```

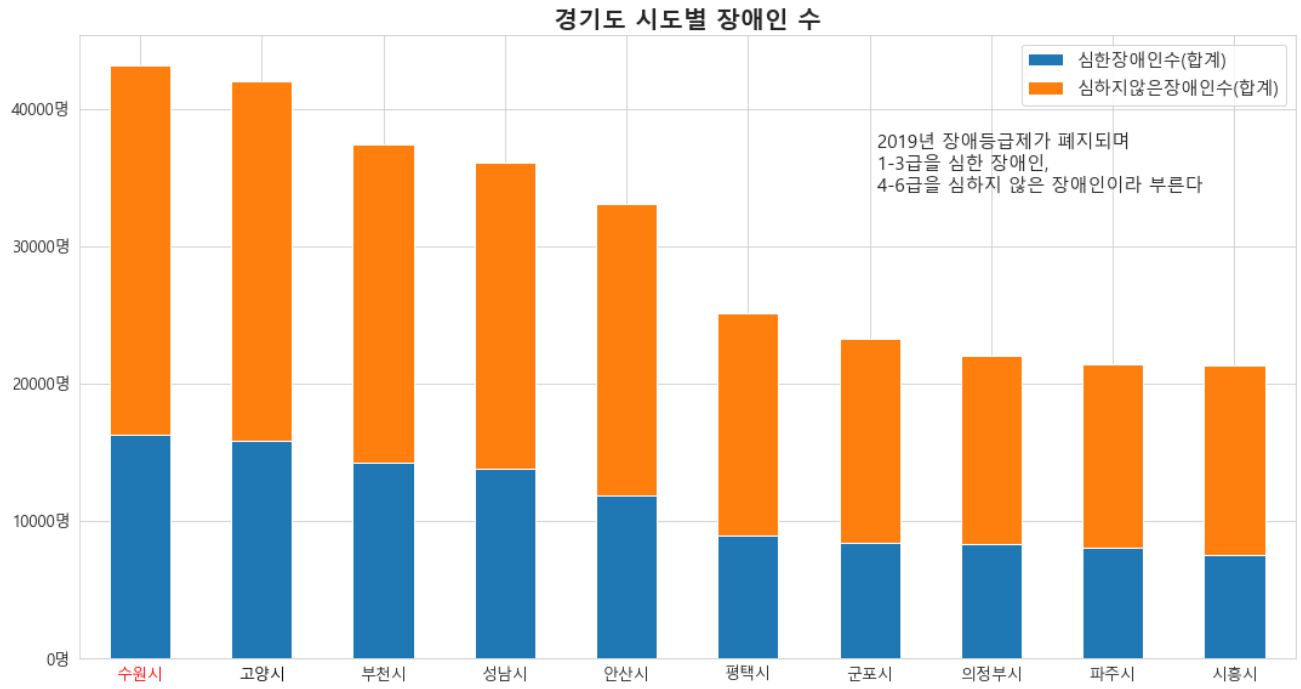
In [196]:
```
df1 = pd.read_csv('./data/1. 경기도내등록장애인집계현황(읍면동별,유형별,정도별).csv')
df2 = pd.read_csv('./data/2. 경기도_수원시_전동휠체어급속충전기.csv', encoding = 'cp949')
df_in = pd.read_csv('./data/3. 장애인(베리어프리)실내문화공간목록.csv')
df_out = pd.read_csv('./data/4. 장애인(베리어프리)실외문화공간목록.csv')
pop = pd.read_csv('./data/5. 주민등록인구집계현황.csv', encoding = 'cp949')
경기도집계 = pd.read_csv('./data/6. 등록장애인집계현황(시군별,유형별,성별).csv', encoding='ut
유형연령별 = pd.read_csv('./data/7. 등록장애인집계현황(장애유형별,연령별).csv', encoding = 'u
연령별 = pd.read_csv('./data/8. 전국_연령별_장애유형별_성별_등록장애인수.csv', encoding='utf-8
df9 = pd.read_csv('./data/9. 장애인복지시설목록.csv')
geo_df = geopandas.read_file('./data/10. 수원시_행정경계(읍면동).geojson')
code = pd.read_csv('./data/11. pos00006m.csv')
sb = pd.read_csv('./data/12. 경기도_시군별보조기기서비스센터현황.csv', encoding='utf-8')
# 태블로
# 13. 수원시공중화장실.csv
```

장애인 현황

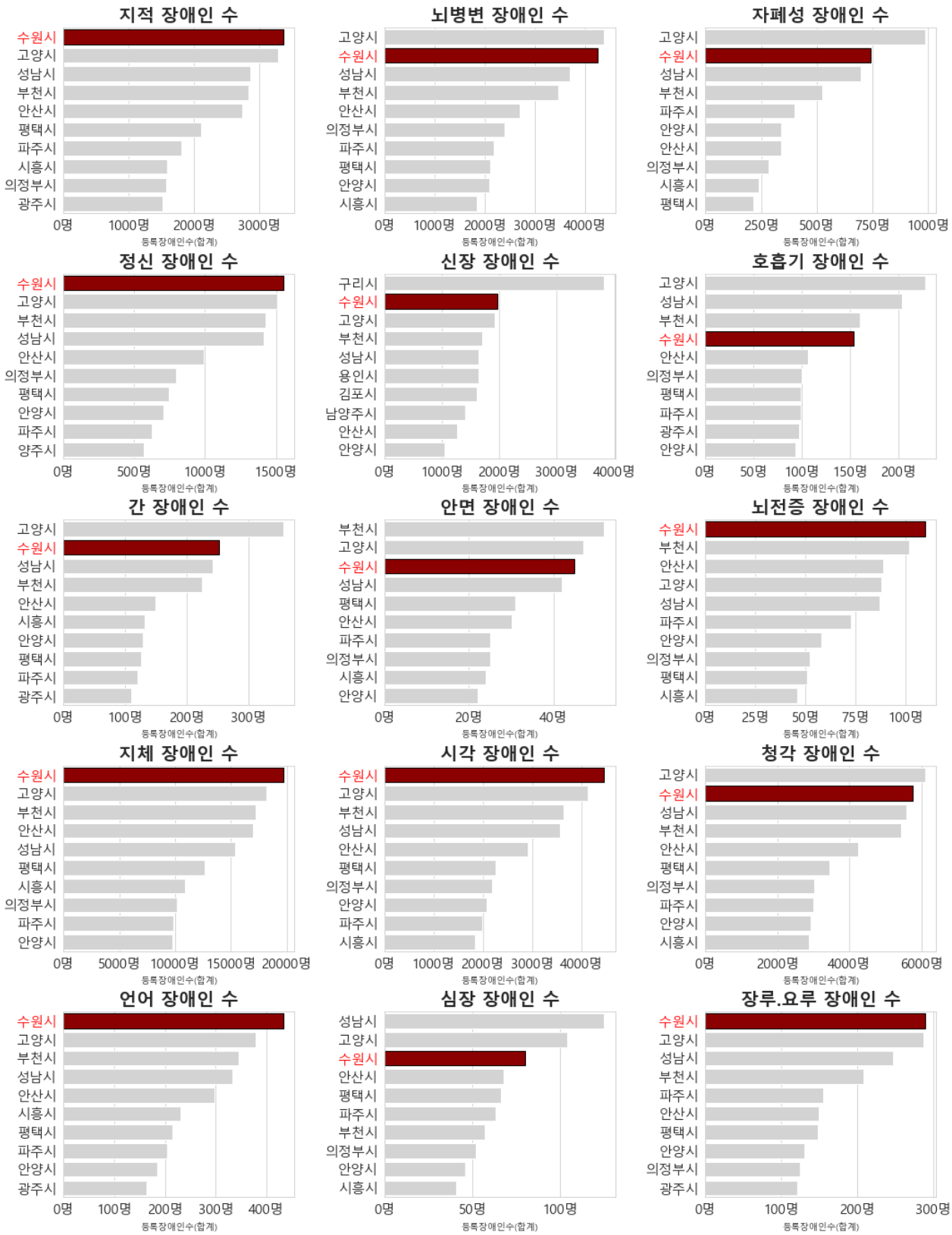
In [197]:
```
def person(x, pos):
    return f'{np.int64(x)}명'
```

In [198]:

```
fig, ax = plt.subplots(1,1, figsize = (15, 8))
data = df1.loc[(df1['장애유형'] != '소계')].groupby('시군명')[['등록장애인수(합계)','심한장애
data = data.set_index('시군명')[['심한장애인수(합계)','심하지않은장애인수(합계)']]
data.plot.bar(stacked = True, ax = ax)
ax.set_title('경기도 시도별 장애인 수', fontsize = 20, fontweight = 'bold')
ax.tick_params(labelsize = 13)
for ytick, color in zip(ax.get_xticklabels(), ['red','black']):
    ytick.set_color(color)
formatter = FuncFormatter(person)
ax.yaxis.set_major_formatter(formatter)
ax.set(ylabel = None)
ax.set(xlabel = None)
ax.annotate('2019년 장애등급제가 폐지되며₩n1-3급을 심한 장애인,₩n4-6급을 심하지 않은 장애인이라
                xy = (6,380), fontsize = 15)
plt.xticks(rotation=0)
plt.legend(fontsize = 'x-large')
fig.tight_layout()
fig.show()
```



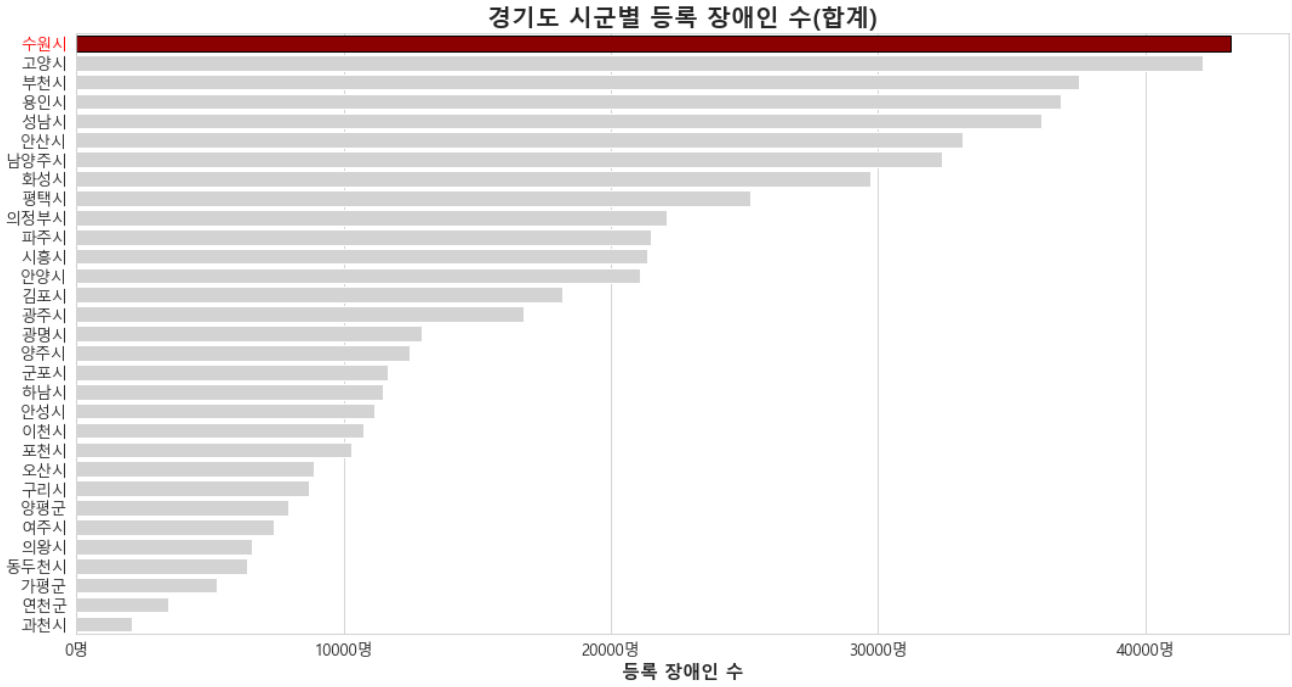
In [199]:
```
fig, ax = plt.subplots(5,3, figsize = (15, 20))
k = 0
num_li = [0,1,1,0,1,3,1,2,0,0,0,1,0,2,0]
for n, typ in enumerate(df1.loc[(df1['장애유형'] != '소계')&(df1['장애유형'] != '장루,요루')&
    cnt = n % 3
    data = df1.loc[df1['장애유형'] == typ].groupby('시군명')['등록장애인수(합계)'].sum().to_f
    sns.barplot(x = '등록장애인수(합계)', y = '시군명', data = data, color = 'lightgray', ax
    ax[k, cnt].set_title(f'{typ} 장애인 수', fontsize = 20, fontweight = 'bold')
    ax[k, cnt].tick_params(labelsize = 15)
    ax[k, cnt].set_ylabel('시도', fontsize = 15, fontweight = 'bold')
    ax[k, cnt].patches[num_li[n]].set_facecolor('darkred')
    ax[k, cnt].patches[num_li[n]].set_edgecolor('black')
    ax[k, cnt].set(ylabel = None)
    formatter = FuncFormatter(person)
    ax[k, cnt].xaxis.set_major_formatter(formatter)
    for ytick in ax[k, cnt].get_yticklabels():
        if ytick.get_text() == '수원시':
            ytick.set_color('red')
    if cnt == 2:
        k += 1
#fig.suptitle('시도별 유형별 장애인 수' ,fontsize = 25, fontweight = 'bold')
fig.tight_layout()
fig.subplots_adjust(top=0.94)
fig.savefig('../시도별 유형별 장애인 수.png', dpi = 300)
fig.show()
```



- 수원시가 더 골고루 퍼져있음

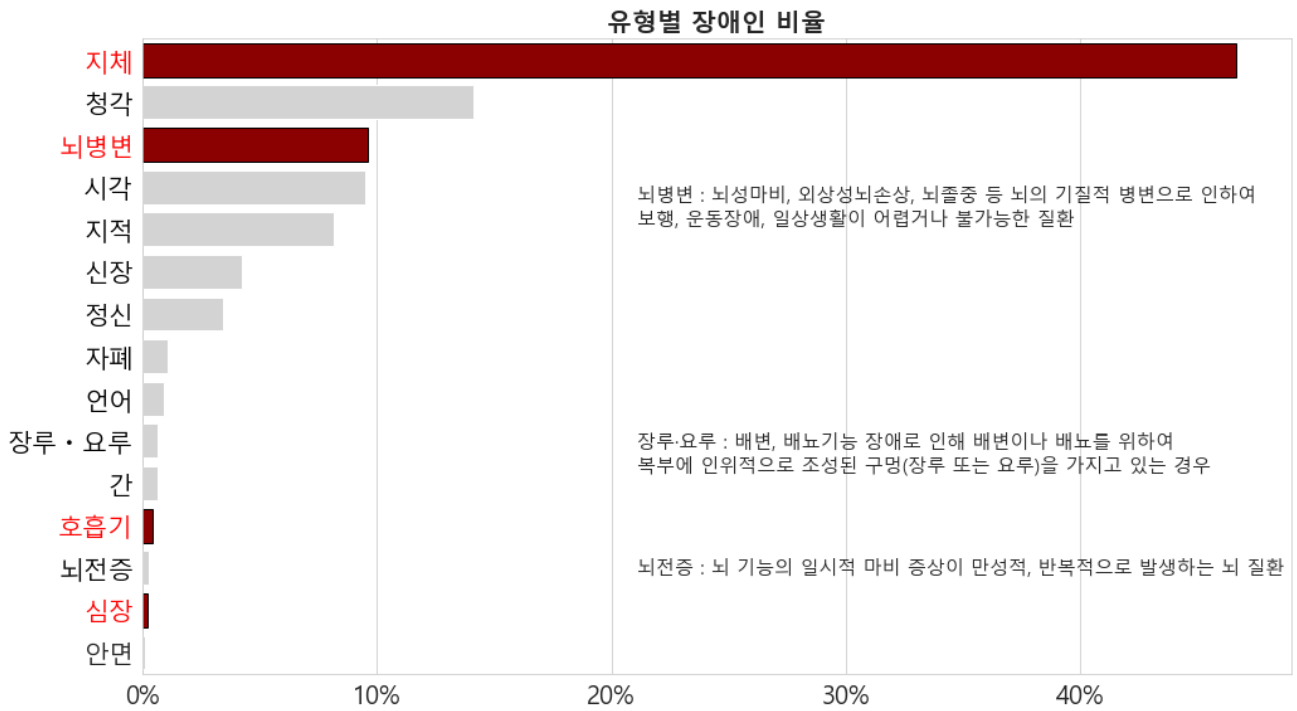
In [200]:
```
경기도집계 = 경기도집계[['기준년도','시군명','합계(명)']].copy()
경기도집계 = 경기도집계[경기도집계['기준년도'] == 2021]
```

In [201]:
```
fig, ax = plt.subplots(1,1, figsize = (15, 8))

data = 경기도집계.sort_values(by = '합계(명)', ascending=False)
sns.barplot(x = '합계(명)', y = '시군명', data = data, color = 'lightgray', ax = ax)
ax.set_title('경기도 시군별 등록 장애인 수(합계)', fontsize=20, fontweight='bold')
ax.tick_params(labelsize = 13)
ax.set_xlabel('등록 장애인 수', fontsize=15, fontweight='bold')
ax.set_ylabel('시군명', fontsize=15, fontweight='bold')
for ytick, color in zip(ax.get_yticklabels(), ['red']):
    ytick.set_color(color)
ax.set(ylabel = None)
formatter = FuncFormatter(person)
ax.xaxis.set_major_formatter(formatter)
ax.patches[0].set_facecolor('darkred')
ax.patches[0].set_edgecolor('black')
fig.tight_layout()
fig.show()
```

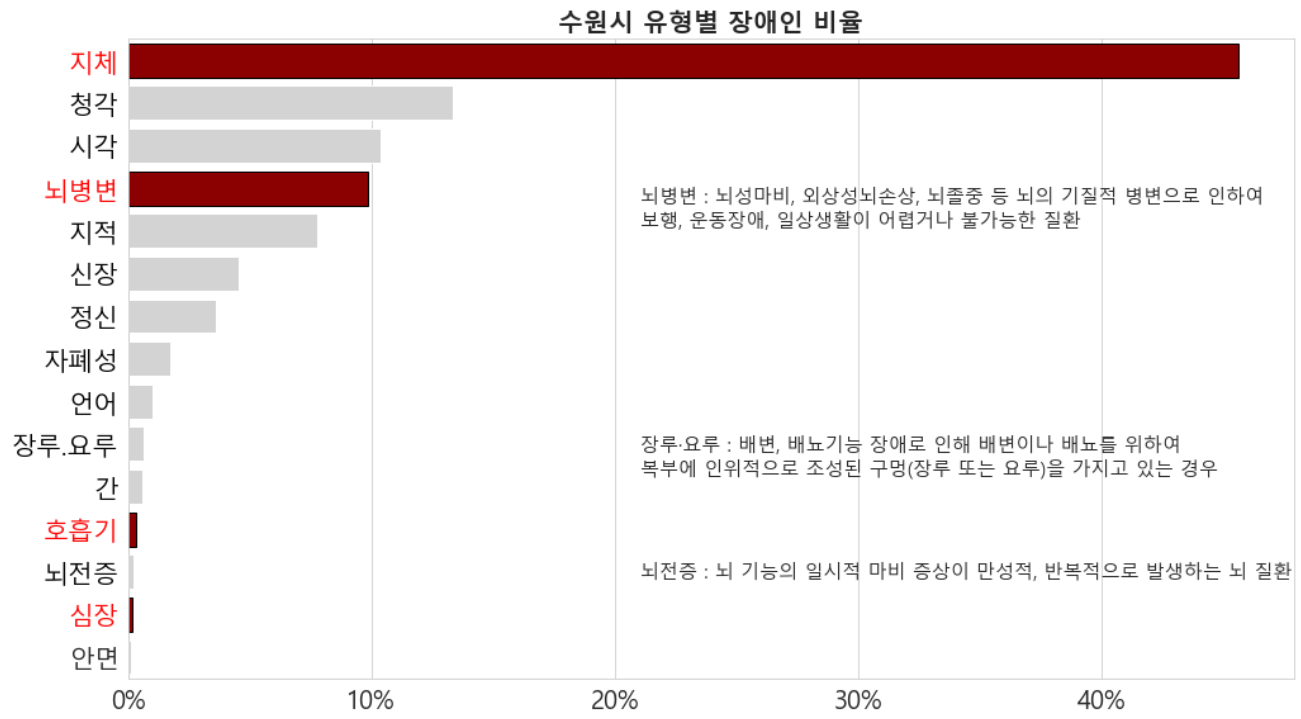


In [202]:
```
유형연령별 = 유형연령별[유형연령별['기준년도'] == 2021].copy()
유형연령별.columns = ['기준년도','장애유형','총계(명)','10세미만(명)','10대(명)', ₩
                  '20대(명)','30대(명)','40대(명)','50대(명)', ₩
                  '60대(명)','70대(명)','80대(명)','90세이상(명)']
for column_name in 유형연령별:
    if column_name != '기준년도' and column_name != '장애유형':
        유형연령별[column_name] = 유형연령별[column_name].str.replace(',','').copy()
        유형연령별[column_name] = 유형연령별[column_name].astype('int32').copy()

유형별총인구 = 0
for i in range(len(유형연령별)):
    유형별총인구 += 유형연령별['총계(명)'][i]
유형연령별['총계(비율)'] = 유형연령별['총계(명)'] / 유형별총인구 * 100
```

In [268]:
```
def percent(x, pos):
    return f'{np.int64(x)}%'
fig, ax = plt.subplots(1,1, figsize = (15, 8))

data = 유형연령별.sort_values(by = '총계(비율)', ascending=False)
sns.barplot(x = '총계(비율)', y = '장애유형', data = data, color = 'lightgray', ax = ax)
ax.set_title('유형별 장애인 비율', fontsize=20, fontweight='bold')
ax.tick_params(labelsize = 20)
ax.set_xlabel('총계(비율)', fontsize=15, fontweight='bold')
ax.set_ylabel('장애유형', fontsize=15, fontweight='bold')
for ytick, color in zip(ax.get_yticklabels(), ['red','black','red','black','black','black','b
    ytick.set_color(color)
for n in [0,2,11,13]:
    ax.patches[n].set_facecolor('darkred')
    ax.patches[n].set_edgecolor('black')
formatter = FuncFormatter(percent)
ax.xaxis.set_major_formatter(formatter)
ax.set(ylabel = None)
ax.set(xlabel = None)
ax.annotate('뇌병변 : 뇌성마비, 외상성뇌손상, 뇌졸중 등 뇌의 기질적 병변으로 인하여Wn보행, 운
            xy=(20,4), fontsize=15)
ax.annotate('장루·요루 : 배변, 배뇨기능 장애로 인해 배변이나 배뇨를 위하여 Wn복부에 인위적으
                xy = (20,10), fontsize = 15)
ax.annotate('뇌전증 : 뇌 기능의 일시적 마비 증상이 만성적, 반복적으로 발생하는 뇌 질환', textc
            xy=(20,12.5), fontsize=15)

plt.xticks(rotation=0)

fig.tight_layout()
fig.show()
```



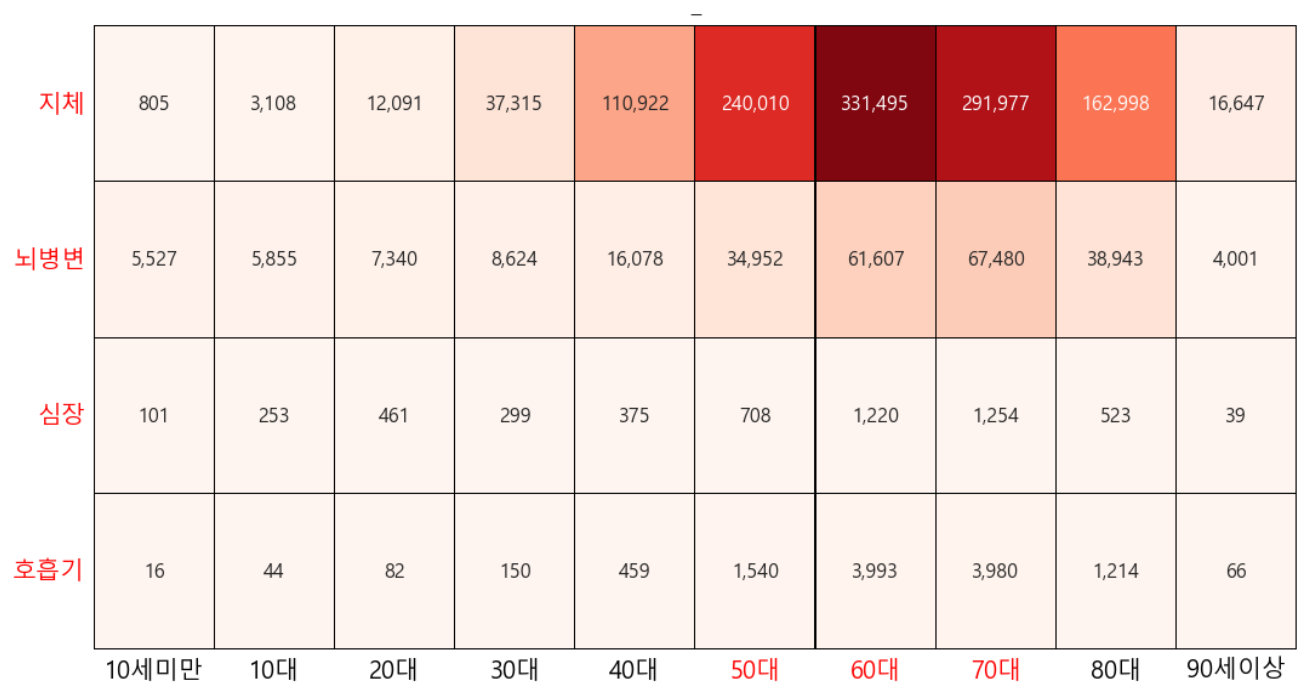
In [267]:
```
fig, ax = plt.subplots(1,1, figsize = (15, 8))

data = (df1.loc[df1['시군명'].str.contains('수원')].groupby('장애유형')['등록장애인수(합계)'].
data['등록장애인수(합계)'] = data['등록장애인수(합계)'] * 100
sns.barplot(y = '장애유형', x = '등록장애인수(합계)', data = data.reset_index(), color = 'lig
ax.set_title('수원시 유형별 장애인 비율', fontsize=20, fontweight='bold')
ax.tick_params(labelsize = 20)
ax.set_xlabel('총계(비율)', fontsize=15, fontweight='bold')
ax.set_ylabel('장애유형', fontsize=15, fontweight='bold')
for ytick, color in zip(ax.get_yticklabels(), ['red','black','black','red','black','black','b
    ytick.set_color(color)
for n in [0,3,11,13]:
    ax.patches[n].set_facecolor('darkred')
    ax.patches[n].set_edgecolor('black')
formatter = FuncFormatter(percent)
ax.xaxis.set_major_formatter(formatter)
ax.set(ylabel = None)
ax.set(xlabel = None)
ax.annotate('뇌병변 : 뇌성마비, 외상성뇌손상, 뇌졸중 등 뇌의 기질적 병변으로 인하여Wn보행, 운
           xy=(20,4), fontsize=15)
ax.annotate('장루·요루 : 배변, 배뇨기능 장애로 인해 배변이나 배뇨를 위하여 Wn복부에 인위적으
               xy = (20,10), fontsize = 15)
ax.annotate('뇌전증 : 뇌 기능의 일시적 마비 증상이 만성적, 반복적으로 발생하는 뇌 질환', textc
           xy=(20,12.5), fontsize=15)

fig.tight_layout()
fig.show()
```



45.6%

In [205]:
```
연령별 = 연령별.rename(columns=연령별.iloc[0])
연령별 = 연령별.drop(연령별.index[0])
소계 = 연령별['연령별(2)'] == '소계'
연령별.drop(['연령별(2)'], axis=1, inplace=True)
연령별T = 연령별.T
연령별T = 연령별T.rename(columns=연령별T.iloc[0])
연령별T = 연령별T.drop(연령별T.index[0])
계 = 연령별T['연령별(1)'] == '계'
연령별 = 연령별T[계]
연령별.drop(['연령별(1)'],axis=1,inplace=True)
연령별 = 연령별.T.reset_index()
널 = ~연령별['index'].isnull()
연령별 = 연령별[널]
for column_name in 연령별:
    if column_name != 'index':
        연령별[column_name] = 연령별[column_name].str.replace(',','').copy()
        연령별[column_name] = 연령별[column_name].astype('float32').copy()
연령별 = 연령별.set_index('index')
연령별 = 연령별.T
연령별 = 연령별.drop(연령별.index[0])
연령별['10세미만'] = 연령별['0-4세'] + 연령별['5-9세']
연령별['10대'] = 연령별['10-14세'] + 연령별['15-19세']
연령별['20대'] = 연령별['20-24세'] + 연령별['25-29세']
연령별['30대'] = 연령별['30-34세'] + 연령별['35-39세']
연령별['40대'] = 연령별['40-44세'] + 연령별['45-49세']
연령별['50대'] = 연령별['50-54세'] + 연령별['55-59세']
연령별['60대'] = 연령별['60-64세'] + 연령별['65-69세']
연령별['70대'] = 연령별['70-74세'] + 연령별['75-79세']
연령별['80대'] = 연령별['80-84세'] + 연령별['85-89세']
연령별['90세이상'] = 연령별['90-94세'] + 연령별['95-99세'] + 연령별['100-104세'] + 연령별['10
                    연령별['110-114세'] + 연령별['115-119세'] + 연령별['120-124세']
연령별 = 연령별[['합계','10세미만','10대','20대','30대','40대','50대','60대','70대','80대','90
연령 = 연령별.copy()
for i in range(len(연령별)):
    연령별['10세미만'][i] = 연령별['10세미만'][i] / 연령별['합계'][i]
    연령별['10대'][i] = 연령별['10대'][i] / 연령별['합계'][i]
    연령별['20대'][i] = 연령별['20대'][i] / 연령별['합계'][i]
    연령별['30대'][i] = 연령별['30대'][i] / 연령별['합계'][i]
    연령별['40대'][i] = 연령별['40대'][i] / 연령별['합계'][i]
    연령별['50대'][i] = 연령별['50대'][i] / 연령별['합계'][i]
    연령별['60대'][i] = 연령별['60대'][i] / 연령별['합계'][i]
    연령별['70대'][i] = 연령별['70대'][i] / 연령별['합계'][i]
    연령별['80대'][i] = 연령별['80대'][i] / 연령별['합계'][i]
    연령별['90세이상'][i] = 연령별['90세이상'][i] / 연령별['합계'][i]
연령별.drop(['합계'],axis=1,inplace=True)
연령.drop(['합계'],axis=1,inplace=True)
```

In [206]:
```
연령 = 연령.loc[['지체','뇌병변','심장','호흡기'],:]
fig, ax = plt.subplots(1,1, figsize = (15, 8))
ax = sns.heatmap(연령, annot=True, ₩
                 fmt = '.0f', vmin = 0.0,vmax=350000, cmap = 'Reds',linewidths = 0.1,
                 linecolor = 'black', annot_kws={"size": 16}, cbar=False)

ax.set_title('一', fontsize=10, fontweight='bold')
ax.set_xlabel('연령대', fontsize=20, fontweight='bold')
ax.set_ylabel('장애유형', fontsize=20, fontweight='bold')

for ytick, color in zip(ax.get_yticklabels(), ['red','red','red','red']):
    ytick.set_color(color)
    ytick.set_fontsize(20)
for xtick, color in zip(ax.get_xticklabels(), ['black','black','black','black','black','red',
    xtick.set_color(color)
    xtick.set_fontsize(20)
plt.yticks(rotation=0)
ax.set(ylabel = None, xlabel = None)
for t in ax.texts:
    t.set_text('{:,d}'.format(int(t.get_text())))
fig.tight_layout()
fig.show()
```



장애인 휠체어(행정동 기준)

In [207]:
```
df2['동'] = df2['소재지지번주소'].map(lambda x: x.split(' ')[-2])
code['ctgg_nm'] = code['ctgg_nm'].map(lambda x: str(x).split(' ')[0])
```

In [208]:
```
cd_dic = {}
for i in code.loc[code['ctgg_nm'] == '수원시']['adstrd_nm'].unique():
    cd = code.loc[code['adstrd_nm'] == i,'lgdng_nm'].values[0]
    cd_dic[cd] = i
```

In [209]:
```
cd_dic['오목천동'] = '평동'
cd_dic['탑동'] = '서둔동'
cd_dic['남창동'] = '행궁동'
cd_dic['매산로2가'] = '매산동'
cd_dic['매향동'] = '행궁동'
cd_dic['신풍동'] = '행궁동'
cd_dic['북수동'] = '행궁동'
```

In [210]:
```
hang_dic = {'SK청솔노인복지관':'정자1동', '경기도의료원 수원병원' : '정자2동','권선1동주민센터
'우만종합사회복지관' : '우만1동','우만1동주민센터':'우만1동', '화서1동주민센터' : '화서1동',
'벽적골공원' : '영통2동', '영통구청' : '매탄3동','매탄1동주민센터': '매탄1동'}
```

In [211]:
```
df2 = df2.replace({'동' : cd_dic})
```

In [212]:
```
hang_dic = {'SK청솔노인복지관':'정자1동', '경기도의료원 수원병원' : '정자2동','권선1동주민센터
'우만종합사회복지관' : '우만1동','우만1동주민센터':'우만1동', '화서1동주민센터' : '화서1동',
'벽적골공원' : '영통2동', '영통구청' : '매탄3동','매탄1동주민센터': '매탄1동'}
```

In [213]:
```
df2 = df2.replace({'동' : cd_dic})
df2.loc[(df2['시설명'] == 'SK청솔노인복지관')|(df2['시설명'] == '경기도의료원 수원병원')|(df2
```

In [214]:
```
wheel = df2[['시설명','시도명','시군구명','위도','경도','동시사용가능대수','동']]
```

In [215]:
```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12)

folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)

m
```

Out[215]: Make this Notebook Trusted to load map: File -> Trust Notebook

장애인이 많은 지역은 좌측아래쪽에 많은 모습입니다.

In [216]:
```
data = pd.DataFrame(columns = ['읍면동명','등록장애인수(합계)'])

with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12, color = 'grey')

folium.GeoJson(geo, name='json_data').add_to(m)

for i in range(wheel.shape[0]):
    folium.CircleMarker(
        [wheel['위도'].iloc[i], wheel['경도'].iloc[i]],
        radius = 5,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

m
```

Out[216]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [217]:
```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12)

#folium.GeoJson(geo, name='json_data').add_to(m)
folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)
for i in range(wheel.shape[0]):
    folium.CircleMarker(
        [wheel['위도'].iloc[i], wheel['경도'].iloc[i]],
        radius = 5,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

m
```

Out[217]: Make this Notebook Trusted to load map: File -> Trust Notebook

- 빨간색 : 충전소 위치

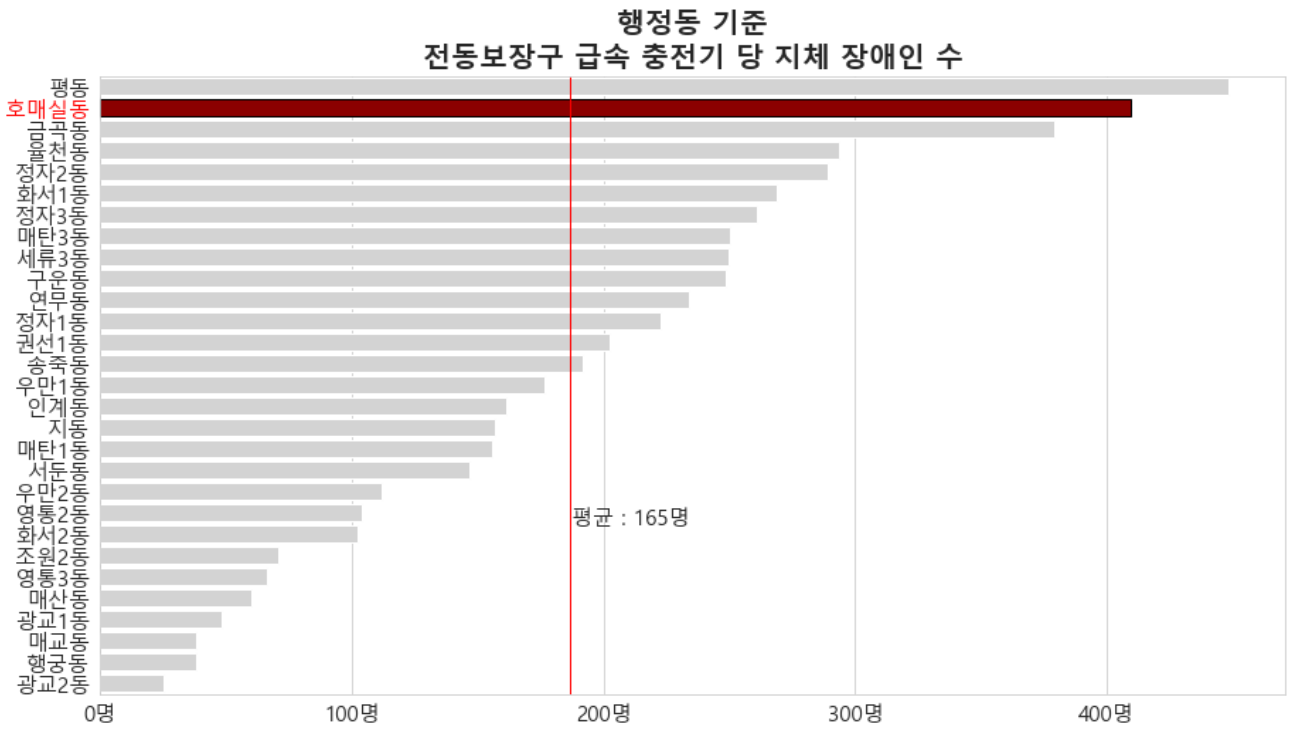
충전소가 중앙쪽에 몰려있는 모습입니다.

또한, 장애인이 많았던 지역에는 비교적 충전소가 많아 보이지는 않습니다.

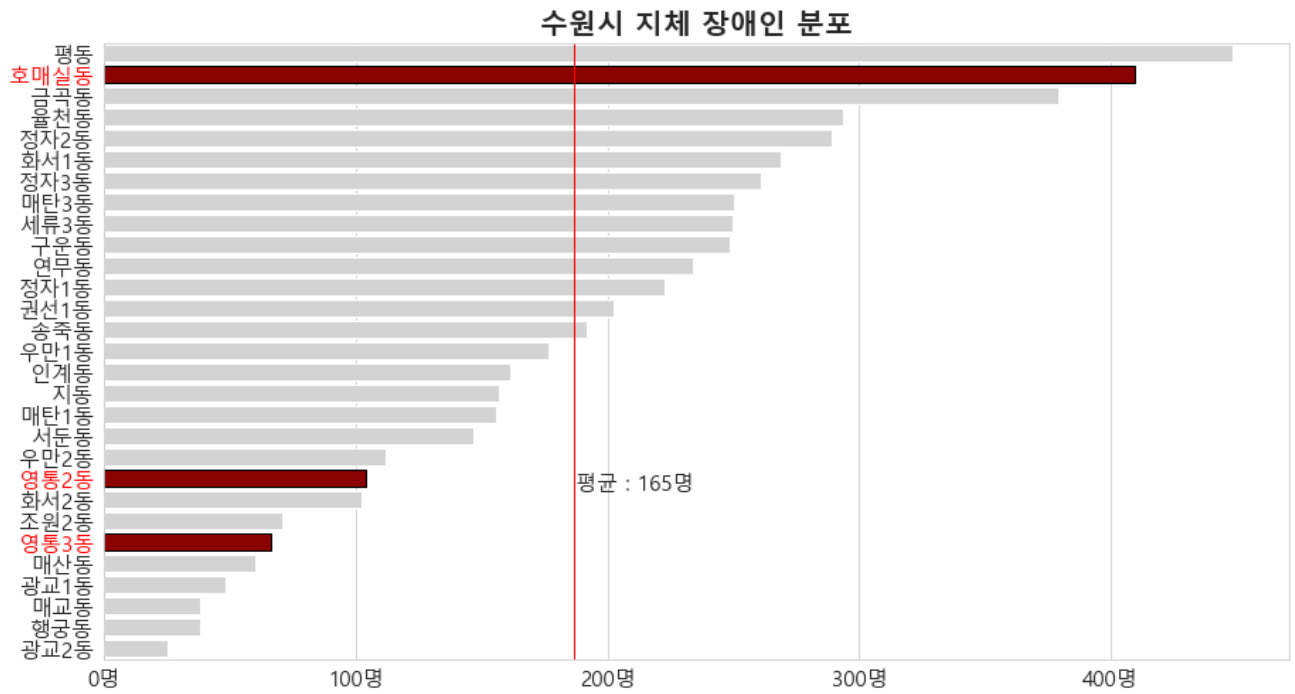
In [218]:
```
per_fac = (df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명
per_fac.columns = ['법정동','기기당장애인수']
```

In [219]:
```
per_fac_drop = per_fac.dropna()
```

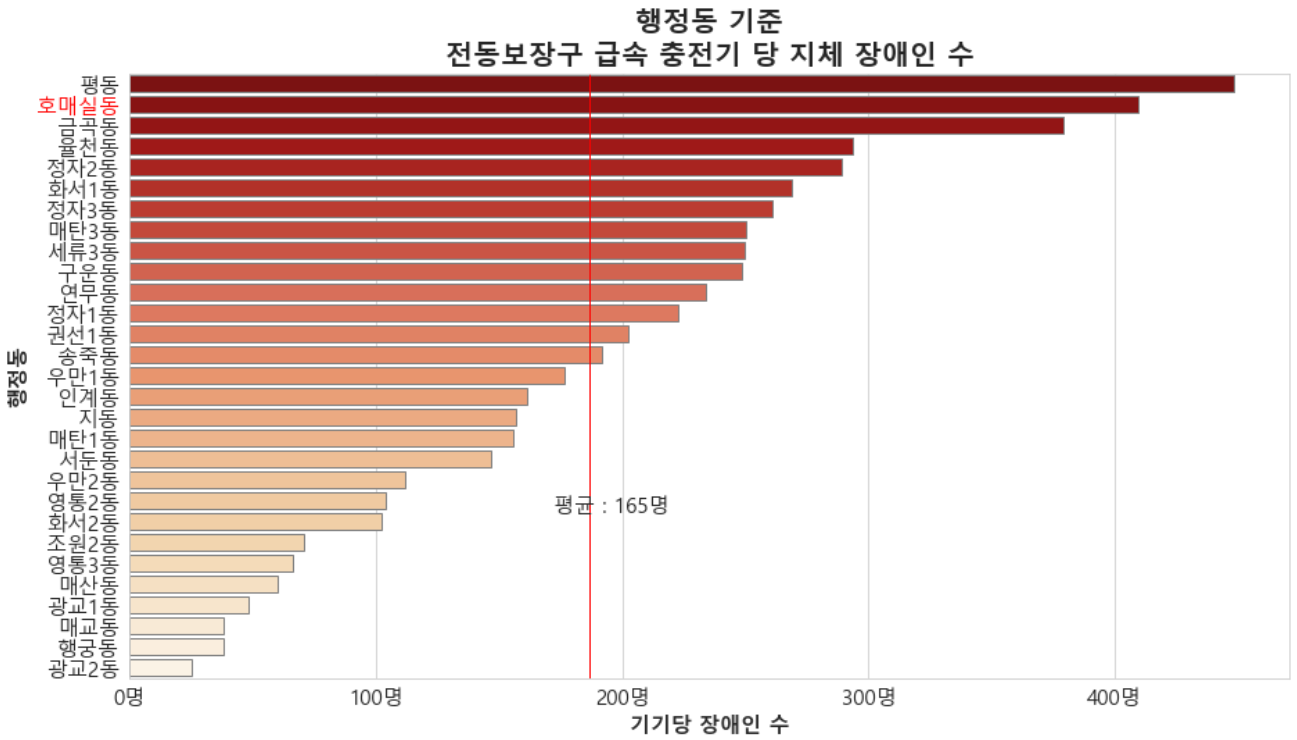
In [248]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 8))
g = sns.barplot(x = '기기당장애인수', y = '법정동', data = per_fac_drop.sort_values(by = '기
g.set_title('행정동 기준Wn전동보장구 급속 충전기 당 지체 장애인 수', fontsize = 20, fontweight
formatter = FuncFormatter(person)
g.xaxis.set_major_formatter(formatter)
g.tick_params(labelsize = 15)
g.set_xlabel('기기당 장애인 수', fontsize = 15, fontweight = 'bold')
g.set_ylabel('행정동', fontsize = 15, fontweight = 'bold')
ax.axvline(x=per_fac_drop['기기당장애인수'].mean(), color='r', linewidth=1)
ax.annotate('평균 : 165명', textcoords = 'offset points',
                xy = (120,22), fontsize = 15)
ax.set(ylabel = None, xlabel = None)
formatter = FuncFormatter(person)
ax.xaxis.set_major_formatter(formatter)
ax.patches[1].set_facecolor('darkred')
ax.patches[1].set_edgecolor('black')
for ytick in ax.get_yticklabels():
        if (ytick.get_text() == '호매실동'):
            ytick.set_color('red')
fig.show()
```



In [249]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 8))
g = sns.barplot(x = '기기당장애인수', y = '법정동', data = per_fac_drop.sort_values(by = '기
g.set_title('수원시 지체 장애인 분포', fontsize = 20, fontweight = 'bold')
formatter = FuncFormatter(person)
g.xaxis.set_major_formatter(formatter)
g.tick_params(labelsize = 15)
g.set_xlabel('기기당 장애인 수', fontsize = 15, fontweight = 'bold')
g.set_ylabel('행정동', fontsize = 15, fontweight = 'bold')
ax.axvline(x=per_fac_drop['기기당장애인수'].mean(), color='r', linewidth=1)
ax.annotate('평균 : 165명', textcoords = 'offset points',
                xy = (120,22), fontsize = 15)
ax.set(ylabel = None, xlabel = None)
formatter = FuncFormatter(person)
ax.xaxis.set_major_formatter(formatter)
ax.patches[1].set_facecolor('darkred')
ax.patches[1].set_edgecolor('black')
ax.patches[20].set_facecolor('darkred')
ax.patches[20].set_edgecolor('black')
ax.patches[23].set_facecolor('darkred')
ax.patches[23].set_edgecolor('black')
for ytick in ax.get_yticklabels():
        if (ytick.get_text() == '호매실동') or '영통' in ytick.get_text():
            ytick.set_color('red')
fig.show()
```

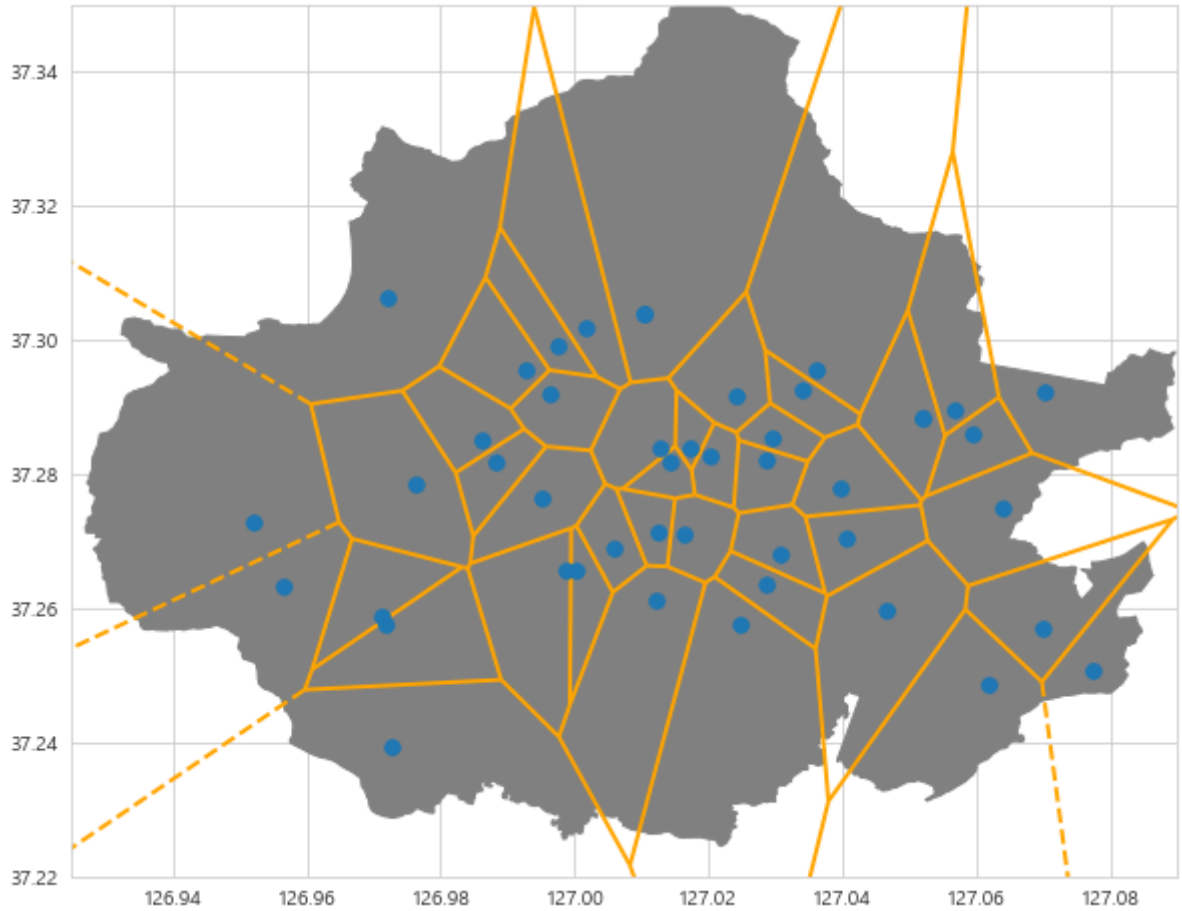


In [222]:
```
fig, ax = plt.subplots(1, 1, figsize = (15, 8))
g = sns.barplot(x = '기기당장애인수', y = '법정동', data = per_fac_drop.sort_values(by = '기
g.set_title('행정동 기준Wn전동보장구 급속 충전기 당 지체 장애인 수', fontsize = 20, fontweight
formatter = FuncFormatter(person)
g.xaxis.set_major_formatter(formatter)
g.tick_params(labelsize = 15)
g.set_xlabel('기기당 장애인 수', fontsize = 15, fontweight = 'bold')
g.set_ylabel('행정동', fontsize = 15, fontweight = 'bold')
ax.axvline(x=per_fac_drop['기기당장애인수'].mean(), color='r', linewidth=1)
ax.annotate('평균 : 165명', textcoords = 'offset points',
                xy = (110,22), fontsize = 15)
for ytick in ax.get_yticklabels():
        if ytick.get_text() == '호매실동':
            ytick.set_color('red')
fig.show()
```



지체 장애인 모두가 전동 휠체어를 타는 것이 아니기 때문에 정확한 해석이 아니라는 한계가 있다.
평동이 가장 많은 지체 장애인을 소화해야하는 모습입니다.

In [223]:
```
geo_data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명
geo_mg = pd.merge(geo_df,geo_data, how = 'left', left_on = 'ADM_DR_NM', right_on = '읍면동명'

fig, ax = plt.subplots(1,1, figsize = (15 , 8))

vor = Voronoi(wheel[['경도','위도']])
voronoi_plot_2d(vor, ax = ax, line_colors = 'orange', show_vertices=False, line_width=2, poi
ax.scatter(x = wheel['경도'], y = wheel['위도'], c = 'r')
geo_mg.plot(ax = ax, edgecolor='grey', color = 'grey')
plt.xlim(126.925, 127.09)
plt.ylim(37.22, 37.35)
fig.show()
```



파란색 : 휠체어 전동 충전소

위 그림은 보로노이 다이어그램 입니다.
파란색 점(휠체어 충전소) 기준으로 최대한 가까운 두 점을 수직이등분선을 이용해 점이 꼭 하나씩 포함되도록 평면을 분할을 합니다.
그렇기 때문에 각 공간 당 파란색 점(휠체어 충전소)은 하나가 존재하게 됩니다.

충전소(파란색 원)당 얼마나 많은 공간을 소화해야하는지 보여줍니다.

이동할 때 행정경계에 얽매이지는 않기 때문에 경계면을 제외하고 충전소 하나가 얼마나 많은 공간을 소화해야하는 확인해보겠습니다.

중앙쪽은 비교적 적은 지역을 소화를 해야하며 외각쪽은 비교적 넓은 지역을 소화해야하는 모습입니다.

In [224]:
```
from folium.features import DivIcon
```

In [225]:

```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

yt_wheel = wheel.loc[wheel['동'].str.contains('영통')]
center = yt_wheel[['위도','경도']].mean().to_list()

m = folium.Map(location = center, titles = 'Maps', zoom_start = 14)

#folium.GeoJson(geo, name='json_data').add_to(m)

folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)

for i in range(yt_wheel.shape[0]):
    folium.CircleMarker(
        [yt_wheel['위도'].iloc[i], yt_wheel['경도'].iloc[i]],
        radius = 5,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

val = (yt_wheel[['위도','경도']].iloc[0].values + yt_wheel[['위도','경도']].iloc[1].values) /

folium.map.Marker(
    (val[0],val[1]), icon=DivIcon(
        icon_size=(150,36),
        icon_anchor=(0,0),
        html='<div style="font-weight: bold; font-size: 12pt">1.15 km</div>'
        )
    ).add_to(m)

val = (yt_wheel[['위도','경도']].iloc[0].values + yt_wheel[['위도','경도']].iloc[2].values) /
folium.map.Marker(
    (val[0], val[1]), icon=DivIcon(
        icon_size=(150,36),
        icon_anchor=(0,0),
        html='<div style="font-weight: bold; font-size: 12pt">0.97 km</div>',
        )
    ).add_to(m)

val = (yt_wheel[['위도','경도']].iloc[1].values + yt_wheel[['위도','경도']].iloc[2].values) /
folium.map.Marker(
    (val[0], val[1]),icon=DivIcon(
        icon_size=(150,36),
        icon_anchor=(0,0),
        html='<div style="font-weight: bold; font-size: 12pt">1.39 km</div>',
        )
    ).add_to(m)

folium.PolyLine([yt_wheel[['위도','경도']].iloc[0].values, yt_wheel[['위도','경도']].iloc[1].v
folium.PolyLine([yt_wheel[['위도','경도']].iloc[0].values, yt_wheel[['위도','경도']].iloc[2].v
folium.PolyLine([yt_wheel[['위도','경도']].iloc[1].values, yt_wheel[['위도','경도']].iloc[2].v

m
```

Out[225]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [226]:
```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
ind = data['읍면동명'].str.contains('영통|광교1동|호매실|세류|인계')
data = data.loc[ind]
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 11.5)

folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)


for i in range(wheel.loc[wheel['동'].str.contains('영통|광교1동|호매실|세류|인계')].shape[0]):
    folium.CircleMarker(
        [wheel.loc[wheel['동'].str.contains('영통|광교1동|호매실|세류|인계')]['위도'].iloc[i],
        radius = 4,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

m
```

Out[226]: Make this Notebook Trusted to load map: File -> Trust Notebook

수원시 인구 분포

In [227]:
```
pop_sw = pop.loc[(pop['연도'] == 2021)&(pop['월'] == 7)&(pop['행정구역명'].str.contains('경기
pop_sw['행정동'] = pop_sw['행정구역명'].map(lambda x: x.strip().split(' ')[-1])
```

In [228]:
```
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12)

folium.Choropleth(
    geo_data = geo,
    data = pop_sw, columns = ['행정동','총 인구수'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)

m
```

Out[228]: Make this Notebook Trusted to load map: File -> Trust Notebook

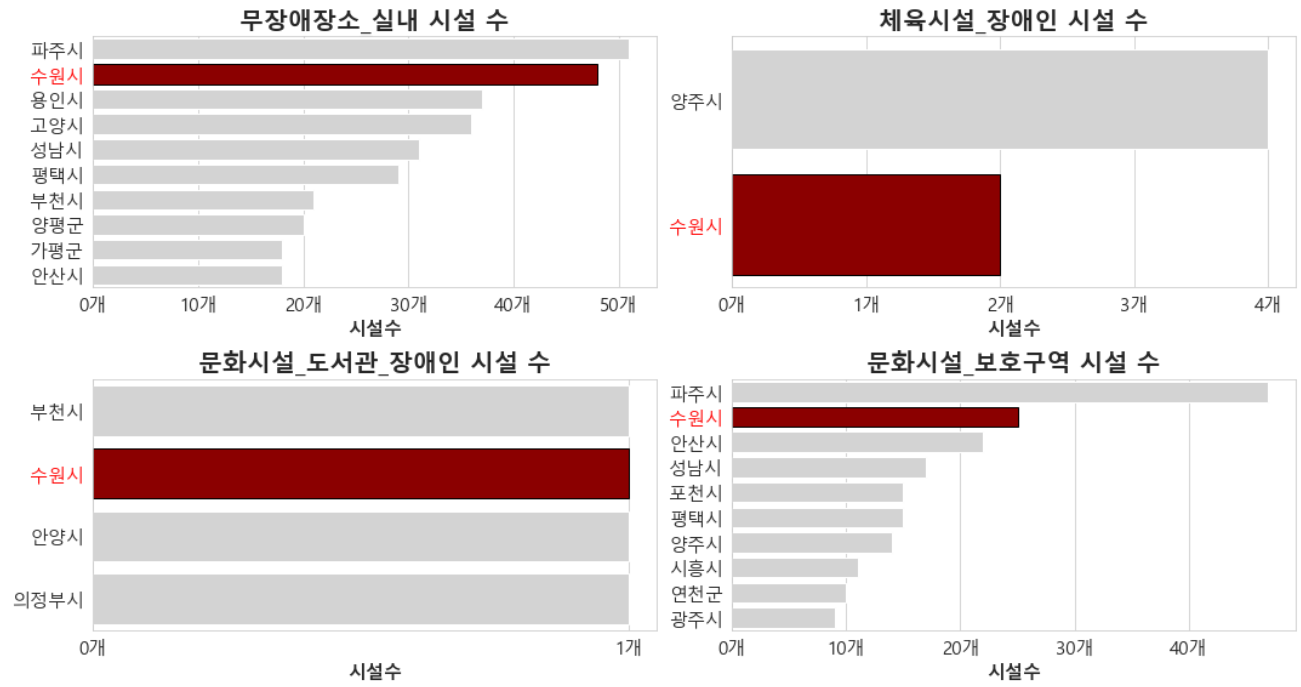
In [229]:
```
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12)

folium.Choropleth(
    geo_data = geo,
    data = pop_sw, columns = ['행정동','총 인구수'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)

for i in range(wheel.shape[0]):
    folium.CircleMarker(
        [wheel['위도'].iloc[i], wheel['경도'].iloc[i]],
        radius = 5,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

m
```

Out[229]: Make this Notebook Trusted to load map: File -> Trust Notebook

실내 문화 공간

In [230]:
```
df_in['SIGNGU_NM'] = df_in['SIGNGU_NM'].map(lambda x: str(x).split(' ')[0])
```

In [231]:
```
def count(x, pos):
    return f'{np.int64(x)}개'

fig, ax = plt.subplots(2,2, figsize = (15, 8))
k = 0
for n, typ in enumerate(df_in['MLSFC_NM'].unique()):
    cnt = n % 2
    data = df_in.loc[((df_in['CTPRVN_NM'] == '경기')|(df_in['CTPRVN_NM'] == '경기도'))&(df_in
    sns.barplot(x = 'ESNTL_ID', y = 'SIGNGU_NM', data = data, ax = ax[k, cnt], color = 'light
    ax[k, cnt].set_title(f'{typ} 시설 수', fontsize = 20, fontweight = 'bold')
    ax[k, cnt].tick_params(labelsize = 15)
    ax[k, cnt].set_xlabel('시설수', fontsize = 15, fontweight = 'bold')
    ax[k, cnt].set_ylabel('시도', fontsize = 15, fontweight = 'bold')
    ax[k, cnt].set(ylabel = None)
    formatter = FuncFormatter(count)
    ax[k, cnt].xaxis.set_major_formatter(formatter)
    for ytick in ax[k, cnt].get_yticklabels():
        if ytick.get_text() == '수원시':
            ytick.set_color('red')
    if cnt == 1:
        k += 1
ax[0, 0].patches[1].set_facecolor('darkred')
ax[0, 0].patches[1].set_edgecolor('black')
ax[0, 1].patches[1].set_facecolor('darkred')
ax[0, 1].patches[1].set_edgecolor('black')
ax[1, 0].patches[1].set_facecolor('darkred')
ax[1, 0].patches[1].set_edgecolor('black')
ax[1, 1].patches[1].set_facecolor('darkred')
ax[1, 1].patches[1].set_edgecolor('black')
ax[0,1].set_xticks([0,1,2,3,4])
ax[1,0].set_xticks([0,1])
fig.tight_layout()
fig.show()
```

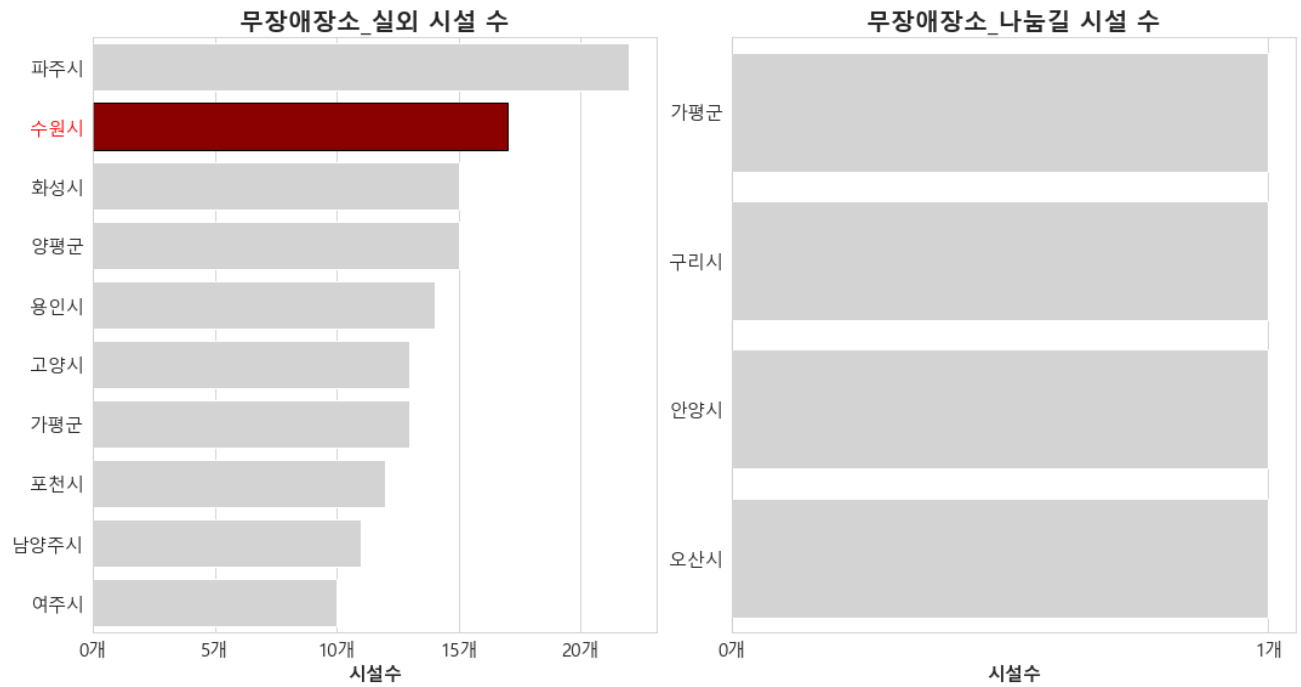


수원시는 장애인을 위한 실내 문화 공간이 비교적 많은 편으로 보입니다.
(체육시설 이용 자체를 별로 하지 않는다해서 따로 지도 x)

실외 문화 공간

In [232]:
```
df_out['SIGNGU_NM'] = df_out['SIGNGU_NM'].map(lambda x: str(x).split(' ')[0])
```

In [233]:
```
def count(x, pos):
    return f'{np.int64(x)}개'
fig, ax = plt.subplots(1,2, figsize = (15, 8))

for n, typ in enumerate(df_out['MLSFC_NM'].unique()):
    data = df_out.loc[((df_out['CTPRVN_NM'] == '경기')|(df_out['CTPRVN_NM'] == '경기도'))&(df
    sns.barplot(x = 'ESNTL_ID', y = 'SIGNGU_NM', data = data, color = 'lightgray', ax = ax[n]
    ax[n].set_title(f'{typ} 시설 수', fontsize = 20, fontweight = 'bold')
    ax[n].tick_params(labelsize = 15)
    ax[n].set_xlabel('시설수', fontsize = 15, fontweight = 'bold')
    ax[n].set_ylabel('시도', fontsize = 15, fontweight = 'bold')
    ax[n].set(ylabel = None)
    formatter = FuncFormatter(count)
    ax[n].xaxis.set_major_formatter(formatter)
    for ytick in ax[n].get_yticklabels():
        if ytick.get_text() == '수원시':
            ytick.set_color('red')
ax[0].patches[1].set_facecolor('darkred')
ax[0].patches[1].set_edgecolor('black')
ax[1].set_xticks([0,1])
fig.tight_layout()
fig.show()
```



전동보장구

In [234]:
```
sb_df = sb.loc[sb['시군명'].str.contains('수원시')][['기관명','WGS84위도','WGS84경도']]
sb_df['유형'] = '보조기기 서비스센터'
```

In [235]:
```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
ind = data['읍면동명'].str.contains('파장동|세류1동|세류2동|곡선동|입북동|권선2동|고등동|매탄2
data = data.loc[-ind]
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

center = [37.2805727, 127.0286009]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 12)

folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)


for i in range(sb_df.shape[0]):
    folium.CircleMarker(
        [sb_df['WGS84위도'].iloc[i],
         sb_df['WGS84경도'].iloc[i]],
        radius = 4,
        color = 'blue',
        fill_color = 'blue'
            ).add_to(m)

for i in range(wheel.shape[0]):
    folium.CircleMarker(
        [wheel['위도'].iloc[i], wheel['경도'].iloc[i]],
        radius = 4,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)


m
```

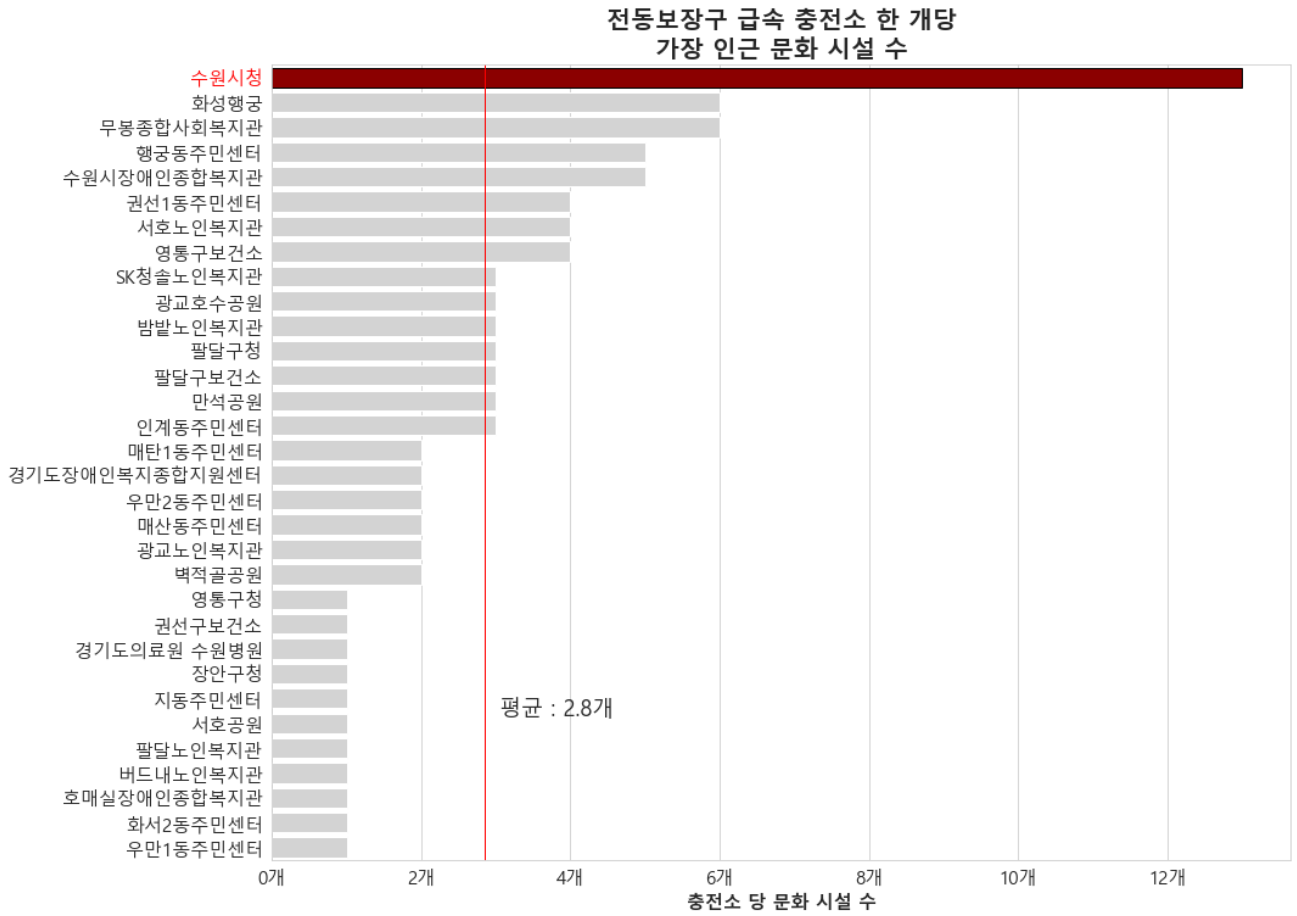
Out[235]: Make this Notebook Trusted to load map: File -> Trust Notebook

수원시청

In [236]:
```
sw_in = df_in.loc[df_in['SIGNGU_NM'] == '수원시']
sw_out = df_out.loc[df_out['SIGNGU_NM'] == '수원시']
sw_cul = pd.concat([sw_in, sw_out])
sw_cul = sw_cul.loc[sw_cul['FCLTY_NM'] != '[백년가게](신)대원옥']
min_name_li = []
min_km_li = []
max_name_li = []
max_km_li = []
for k in sw_cul['FCLTY_NM'].unique():
    #print(f'===={k}====')
    a = sw_cul.loc[sw_cul['FCLTY_NM'] == k][['FCLTY_LA','FCLTY_LO']].iloc[0]
    dis_min = 10000
    dis_max = 0
    fac_min = ''
    fac_max = ''
    for n, name in enumerate(wheel['시설명'].unique()):
        #print(f'{name}')
        dis = haversine(a, wheel.loc[wheel['시설명'] == name][['위도','경도']].iloc[0], unit =
        #print(f'{dis}KM')
        if dis_min >= dis:
            dis_min = dis
            fac_min = name
        if dis_max <= dis:
            dis_max = dis
            fac_max = name
    min_name_li.append(fac_min)
    min_km_li.append(dis_min)
    max_name_li.append(fac_max)
    max_km_li.append(dis_max)
```

In [237]:
```
cul_df = pd.DataFrame({'시설명' : sw_cul['FCLTY_NM'].unique(), '최단충전소' : min_name_li, '최
                       '최장충전소' : max_name_li, '최장거리(km)' : max_km_li})
```

In [238]:
```
def count(x, pos):
    return f'{np.int64(x)}개'
data = cul_df.groupby(['최단충전소'])[['시설명']].count().reset_index().sort_values(by = '시설
fig, ax = plt.subplots(1, 1, figsize = (15, 12))
g = sns.barplot(x = '시설명', y = '최단충전소', data = data, color = 'lightgray')
g.set_title('전동보장구 급속 충전소 한 개당Wn가장 인근 문화 시설 수', fontsize = 20, fontweigh
formatter = FuncFormatter(count)
g.xaxis.set_major_formatter(formatter)
g.tick_params(labelsize = 15)
g.set_xlabel('충전소 당 문화 시설 수', fontsize = 15, fontweight = 'bold')
g.set_ylabel('행정동', fontsize = 15, fontweight = 'bold')
ax.axvline(x=data['시설명'].mean(), color='r', linewidth=1)
ax.annotate('평균 : 2.8개', textcoords = 'offset points',
                xy = (3,27), fontsize = 18)
ax.patches[0].set_facecolor('darkred')
ax.patches[0].set_edgecolor('black')
ax.set(ylabel = None)
for ytick in g.get_yticklabels():
        if ytick.get_text() == '수원시청':
            ytick.set_color('red')
fig.show()
```



In [239]:
```
xy_df = pd.merge(cul_df, sw_cul[['FCLTY_NM','FCLTY_LA','FCLTY_LO']].drop_duplicates(), how =
xy = pd.merge(xy_df, wheel[['시설명','위도','경도']], how = 'left', left_on = '최단충전소', r
```

In [240]:
```
data = df1.loc[(df1['시군명'] == '수원시')&(df1['장애유형'] == '지체')].groupby('읍면동명')[[
with open('./data/10. 수원시_행정경계(읍면동).geojson', encoding = 'utf-8') as file:
    geo = json.loads(file.read())
    file.close()

#center = [37.2805727, 127.0286009]
center = [37.263453000000005, 127.02866399999999]

m = folium.Map(location = center, titles = 'Maps', zoom_start = 15)

folium.Choropleth(
    geo_data = geo,
    data = data, columns = ['읍면동명','등록장애인수(합계)'],
    key_on = 'feature.properties.ADM_DR_NM',
    fill_color = 'BuPu',
    legend_name = '수원시 지체 장애인 수').add_to(m)


for i in range(wheel.loc[wheel['시설명'] != '수원시청'].shape[0]):
    folium.CircleMarker(
        [wheel.loc[wheel['시설명'] != '수원시청']['위도'].iloc[i], wheel.loc[wheel['시설명']
        radius = 5,
        color = 'red',
        fill_color = 'red'
            ).add_to(m)

folium.CircleMarker(
        [wheel.loc[wheel['시설명'] == '수원시청']['위도'].iloc[0], wheel.loc[wheel['시설명'] =
        radius = 15,
        color = 'orange',
        fill_color = 'orange'
            ).add_to(m)

for i in range(sw_cul.shape[0]):
    folium.CircleMarker(
        [sw_cul['FCLTY_LA'].iloc[i], sw_cul['FCLTY_LO'].iloc[i]],
        radius = 5,
        color = 'blue',
        fill_color = 'blue'
            ).add_to(m)

for i in range(xy.shape[0]):
    folium.PolyLine([xy[['FCLTY_LA','FCLTY_LO']].iloc[i].values, xy[['위도','경도']].iloc[i].v
m
```

Out[240]: Make this Notebook Trusted to load map: File -> Trust Notebook

복지시설

In [241]:
```
col = df9.iloc[0,:7].values
df9 = df9.iloc[2:,:7]
df9.columns = col
```

In [242]:
```
latitude = []
longitude = []
error = []
for i,ad in enumerate(df9['주 소']):
    url = f'http://api.vworld.kr/req/address?service=address&request=getcoord&version=2.0&crs=
    req = rq.get(url)
    html = req.text
    soup = BeautifulSoup(html, 'html.parser')
    try:
        latitude.append(soup.result.point.y.text)
        longitude.append(soup.result.point.x.text)
    except:
        error.append(i)
```

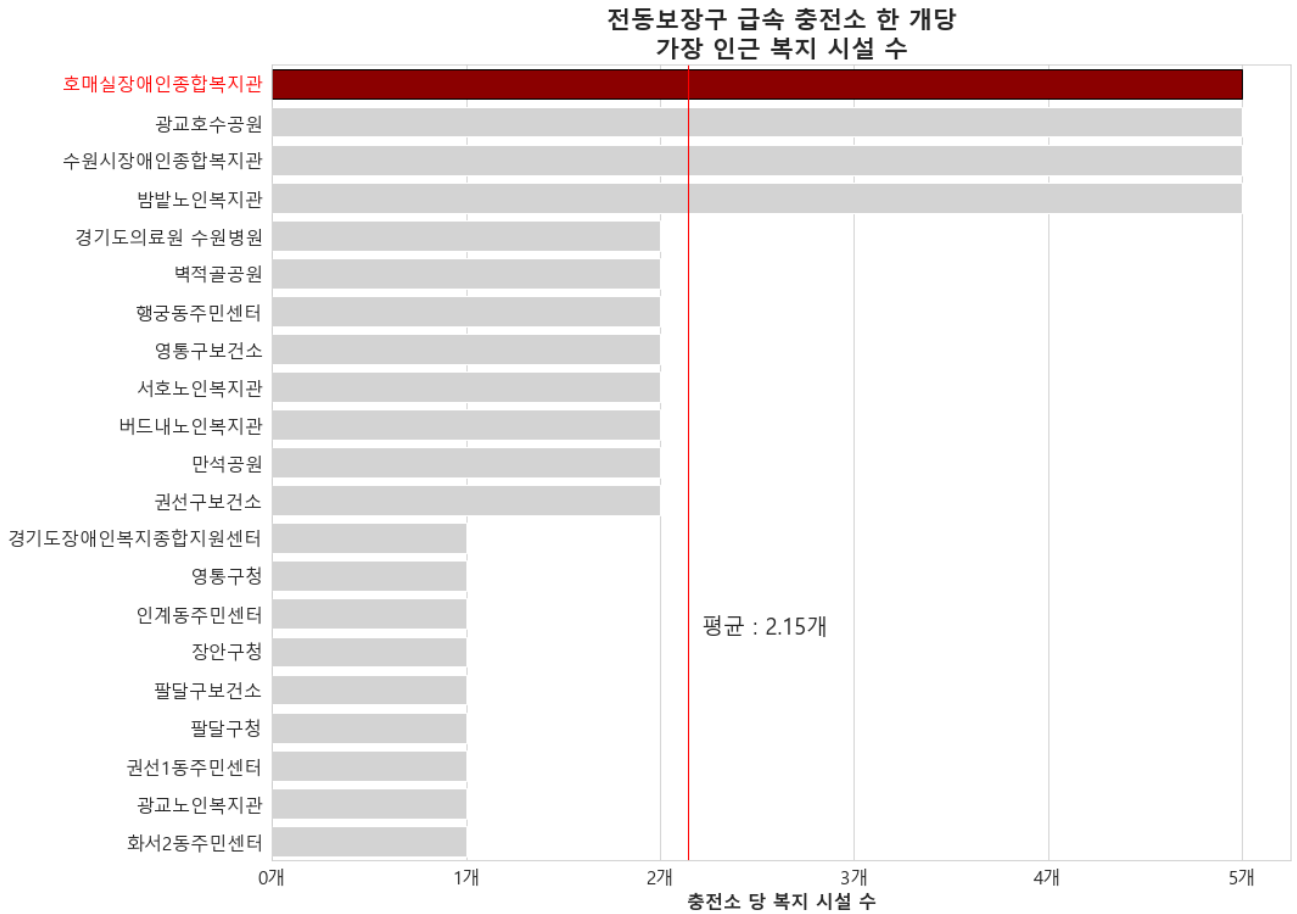
In [243]:
```
latitude.insert(6, '37.30133446087698')
longitude.insert(6, '126.9798916556318')
latitude.insert(28, '37.28760089827169')
longitude.insert(28, '126.98932032679632')
latitude.insert(42, '37.2657993477275')
longitude.insert(42, '127.0643350844663')
```

In [244]:
```
df9['latitude'] = latitude
df9['longitude'] = longitude
```

In [245]:
```
min_name_li = []
min_km_li = []
max_name_li = []
max_km_li = []
for k in df9['시설명'].unique():
    #print(f'===={k}====')
    a = df9.loc[df9['시설명'] == k][['latitude','longitude']].iloc[0]
    a = np.float64(a)
    dis_min = 10000
    dis_max = 0
    fac_min = ''
    fac_max = ''
    for n, name in enumerate(wheel['시설명'].unique()):
        #print(f'{name}')
        dis = haversine(a, wheel.loc[wheel['시설명'] == name][['위도','경도']].iloc[0], unit =
        #print(f'{dis}KM')
        if dis_min >= dis:
            dis_min = dis
            fac_min = name
        if dis_max <= dis:
            dis_max = dis
            fac_max = name
    min_name_li.append(fac_min)
    min_km_li.append(dis_min)
    max_name_li.append(fac_max)
    max_km_li.append(dis_max)
```

In [246]:
```
wel_df = pd.DataFrame({'시설명' : df9['시설명'].unique(), '최단충전소' : min_name_li, '최단거리
                       '최장충전소' : max_name_li, '최장거리(km)' : max_km_li})
```

In [247]:
```
data = wel_df.groupby(['최단충전소'])[['시설명']].count().reset_index().sort_values(by = '시설
fig, ax = plt.subplots(1, 1, figsize = (15, 12))
g = sns.barplot(x = '시설명', y = '최단충전소', data = data, color = 'lightgray')
g.set_title('전동보장구 급속 충전소 한 개당Wn가장 인근 복지 시설 수', fontsize = 20, fontweigh
formatter = FuncFormatter(count)
g.xaxis.set_major_formatter(formatter)
g.tick_params(labelsize = 15)
g.set_xlabel('충전소 당 복지 시설 수', fontsize = 15, fontweight = 'bold')
g.set_ylabel('충전소', fontsize = 15, fontweight = 'bold')
ax.axvline(x=data['시설명'].mean(), color='r', linewidth=1)
ax.annotate('평균 : 2.15개', textcoords = 'offset points',
                xy = (2.2,15), fontsize = 18)
ax.patches[0].set_facecolor('darkred')
ax.patches[0].set_edgecolor('black')
ax.set(ylabel = None)
for ytick in g.get_yticklabels():
        if '호매실' in ytick.get_text() :
            ytick.set_color('red')
fig.show()
```



In [ ]: