In [1]:

```
#-*coding:utf-8 -*-

import pathlib
import random
from functools import reduce
from collections import defaultdict

import pandas as pd
import geopandas as gpd
import folium
import shapely
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import xgboost
import sklearn.cluster
import tensorflow as tf
import matplotlib as mpl
import seaborn as sns

from pandas import DataFrame
from geoband import API
from folium import Map, CircleMarker, Vega, Popup
from vincent import Bar
import math
```

In [2]:

```
input_path = pathlib.Path('./input')
if not input_path.is_dir():
    input_path.mkdir()

from geoband.API import *
GetCompasData('SBJ_2102_001', '1', '1.   _   .csv')
GetCompasData('SBJ_2102_001', '2', '2.   _   .csv')
GetCompasData('SBJ_2102_001', '3', '3.   _   (  )_   .csv')
GetCompasData('SBJ_2102_001', '4', '4.   _     (  )_   .csv')
GetCompasData('SBJ_2102_001', '5', '5.   _     (  )_   .csv')
GetCompasData('SBJ_2102_001', '6', '6.   _    (  )_   .csv')
GetCompasData('SBJ_2102_001', '7', '7.   _   (   )_   .csv')
GetCompasData('SBJ_2102_001', '8', '8.   _     (   )_   .csv')
GetCompasData('SBJ_2102_001', '9', '9.   _     (   )_   .csv')
GetCompasData('SBJ_2102_001', '10', '10.   _    (   )_   .csv')
GetCompasData('SBJ_2102_001', '11', '11.   _     (  )_   .csv')
GetCompasData('SBJ_2102_001', '12', '12.   _  (  )_   .csv')
GetCompasData('SBJ_2102_001', '13', '13.   _   _   .csv')
GetCompasData('SBJ_2102_001', '14', '14.   _   .csv')
GetCompasData('SBJ_2102_001', '15', '15.   _    _   .csv')
GetCompasData('SBJ_2102_001', '16', '16.   _   _   .geojson')
GetCompasData('SBJ_2102_001', '17', '17.   _   _   .csv')
GetCompasData('SBJ_2102_001', '18', '18.   _     (2017~2020).csv')
GetCompasData('SBJ_2102_001', '19', '19.   _   _     _  .geojson')
GetCompasData('SBJ_2102_001', '20', '20.   _   .csv')
GetCompasData('SBJ_2102_001', '21', '21.   _   .csv')
GetCompasData('SBJ_2102_001', '22', '22.   _   _   .csv')
GetCompasData('SBJ_2102_001', '23', '23.   _     _ .geojson')
GetCompasData('SBJ_2102_001', '24', '24.   _      _  .geojson')
GetCompasData('SBJ_2102_001', '25', '25.   _  _   _     .csv')
GetCompasData('SBJ_2102_001', '26', '26.   _   .csv')
GetCompasData('SBJ_2102_001', '27', '27.   _   _   .csv')
GetCompasData('SBJ_2102_001', '28', '28.   _   _     _   .csv')
GetCompasData('SBJ_2102_001', '29', '29.   _   (  ).csv')
GetCompasData('SBJ_2102_001', '30', '30.   _   (   ).geojson')
GetCompasData('SBJ_2102_001', '31', '31.   _   (   ).geojson')
GetCompasData('SBJ_2102_001', '32', '32.   _   (   ).geojson')
GetCompasData('SBJ_2102_001', '33', '33.   _  .geojson')
GetCompasData('SBJ_2102_001', '34', '34.   _    _   .zip')

for path in list(input_path.glob('*.csv')) + list(input_path.glob('*.geojson')):
    print(path)
```

```
input/1.  _   .csv
input/10.  _    (   )_    .csv
input/11.  _     (  )_    .csv
input/12.  _  (  )_    .csv
input/14.  _    .csv
input/13.  _  _    .csv
input/15.  _   _    .csv
input/18.  _      (2017~2020).csv
input/17.  _  _    .csv
input/2.  _  .csv
input/20.  _    .csv
input/21.  _    .csv
input/22.  _  _    .csv
input/25.  _  _  _     .csv
input/26.  _    .csv
input/27.  _  _    .csv
input/28.  _  _     _   .csv
input/29.  _    (  ).csv
input/3.  _  (  )_    .csv
input/4.  _    (  )_    .csv
input/5.  _    (  )_    .csv
input/6.  _   (  )_    .csv
input/7.  _  (   )_    .csv
input/8.  _    (   )_    .csv
input/9.  _    (   )_    .csv
input/16.  _  _    .geojson
input/19.  _  _      _  .geojson
input/23.  _     _  .geojson
input/24.  _      _  .geojson
input/30.  _    (   ).geojson
input/31.  _    (   ).geojson
input/32.  _    (   ).geojson
input/33.  _   .geojson
```

In [ ]:

In [ ]:

In [3]:

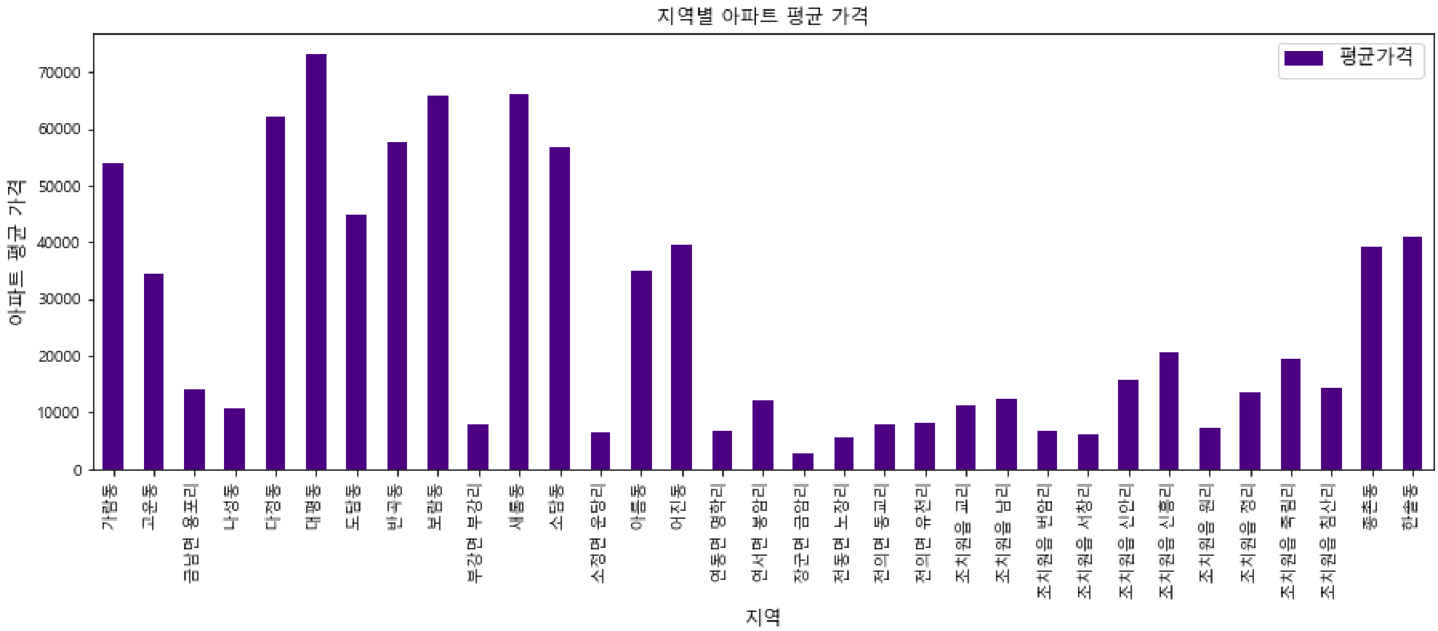
```
#
  _  = pd.read_csv(input_path.joinpath('3.   _   (  )_    .csv'), encoding = 'utf-8' )
  _

#
for i in range(len(   _  ['   '])):
      _  ['   '][i] =     _  ['   '][i][7:]

# col
  _  .columns

#
  _  ['    (  )'] =    _  ['    (  )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_    =    _  .groupby('   ')['    (  )'].mean()
data_

#
import matplotlib
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_   .plot(kind ='bar', title ='             ', figsize =(15, 5), legend=True, fontsize = 10, color='indigo')
ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('         ', fontsize = 12)
ax.legend(['    '], fontsize=12)
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  import sys
```

Out[3]:

```
<matplotlib.legend.Legend at 0x7f21b1e8ae80>
```



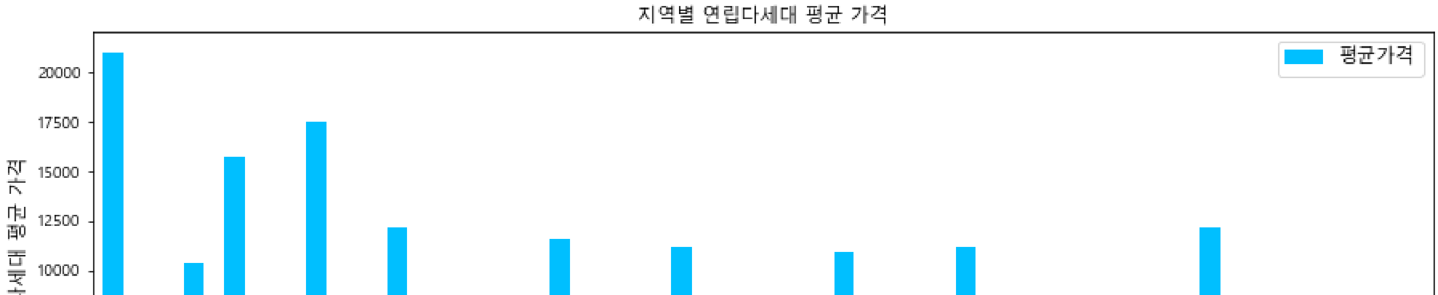
In [4]:

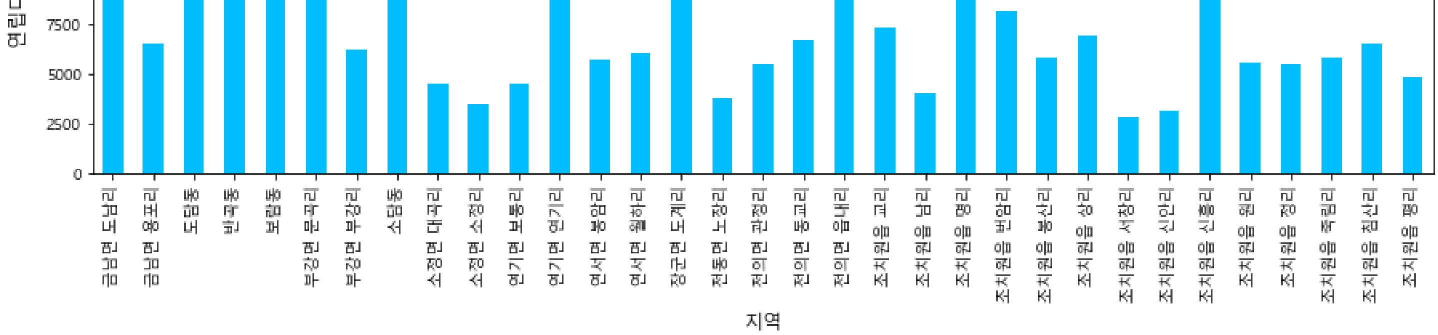
```
#
   _  = pd.read_csv(input_path.joinpath('4.   _     (  )_     .csv'), thousands = ',', encoding = 'utf-8' )
   _

#
for i in range(len(      _  ['    '])):
       _  ['    '][i] =       _  ['    '][i][7:]

# col
   _  .columns

#
data_      =      _  .groupby('    ')['    (  )'].mean()
data_

#
import matplotlib
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_     .plot(kind ='bar', title ='                ', figsize =(15, 5), legend=True, fontsize = 10, color='deepskyblue')
ax.set_xlabel('    ', fontsize = 12)
ax.set_ylabel('              ', fontsize = 12)
ax.legend(['      '], fontsize=12)
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

Out[4]:

```
<matplotlib.legend.Legend at 0x7f212068e128>
```





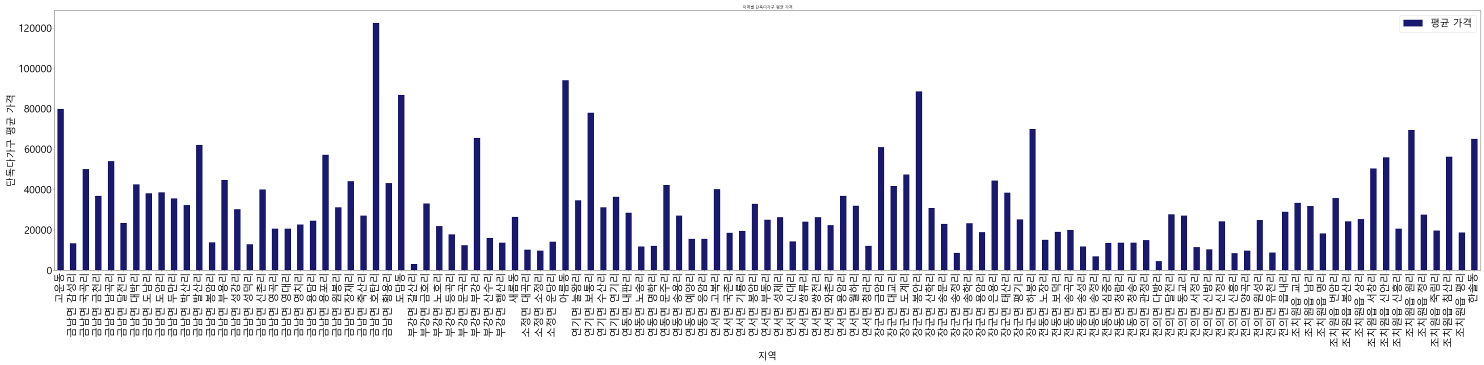
In [5]:

```
#
   _ = pd.read_csv(input_path.joinpath('5.  _    ( )_    .csv'), encoding = 'utf-8' )
   _

#
for i in range(len(    _ ['   '])):
      _ ['   '][i] =      _ ['   '][i][7:]

# col
   _  .columns

#
   _ ['    ( )'] =      _ ['    ( )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_     =      _  .groupby('   ')['    ( )'].mean()
data_

#
import matplotlib
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_    .plot(kind='bar', title='               ', figsize=(80,15), legend=True, fontsize=32, color='midnightblue')
ax.set_xlabel('  ',fontsize=30)
ax.set_ylabel('           ', fontsize=30)
ax.legend(['     '],fontsize=30)
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

Out[5]:

```
<matplotlib.legend.Legend at 0x7f21203c0ac8>
```



In [6]:

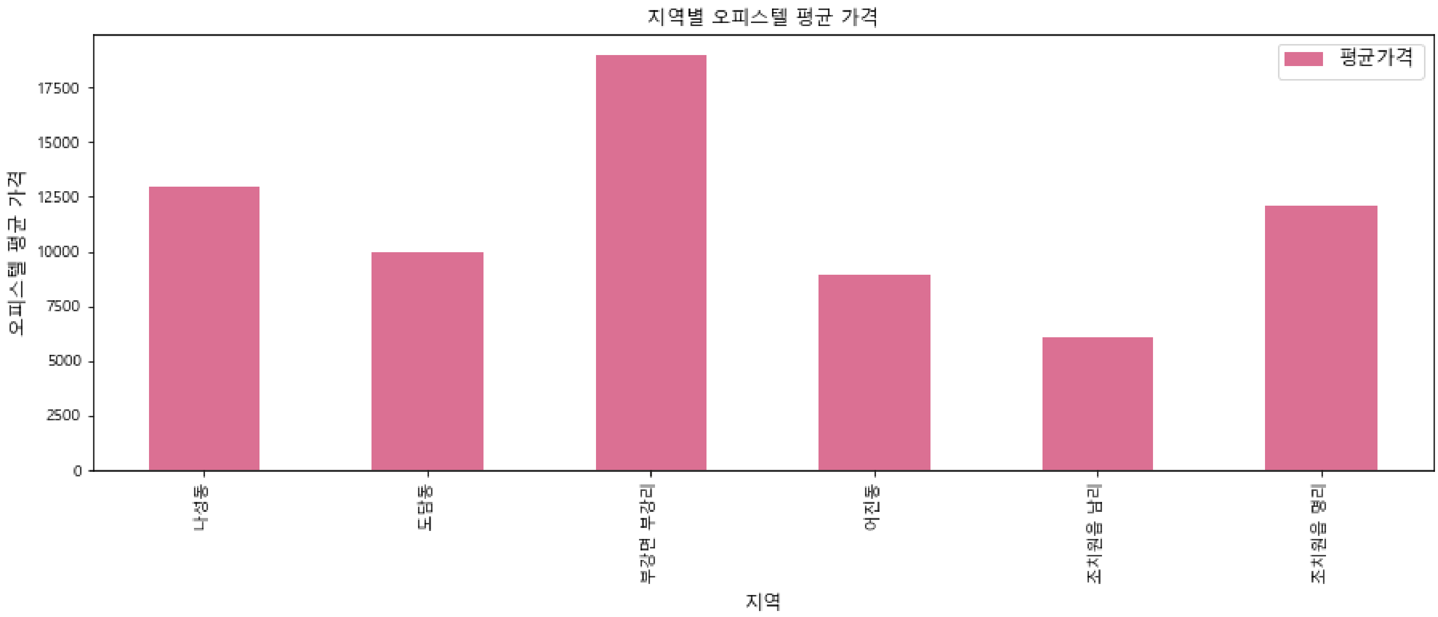
```
#
  _ = pd.read_csv(input_path.joinpath('6.  _    ( )_    .csv'), encoding = 'utf-8' )
  _

#
for i in range(len(    _ ['   '])):
      _ ['   '][i] =      _ ['   '][i][7:]

# col
  _  .columns
```

```
#
  _ ['   ( )'] =    _ ['   ( )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_    =    _ .groupby('   ')['   ( )'].mean()
data_

#
import matplotlib
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_    .plot(kind ='bar', title ='             ', figsize =(15, 5), legend=True, fontsize = 10, color='palevioletred')
ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('         ', fontsize = 12)
ax.legend(['    '], fontsize=12)
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

Out[6]:

```
<matplotlib.legend.Legend at 0x7f21203d3208>
```



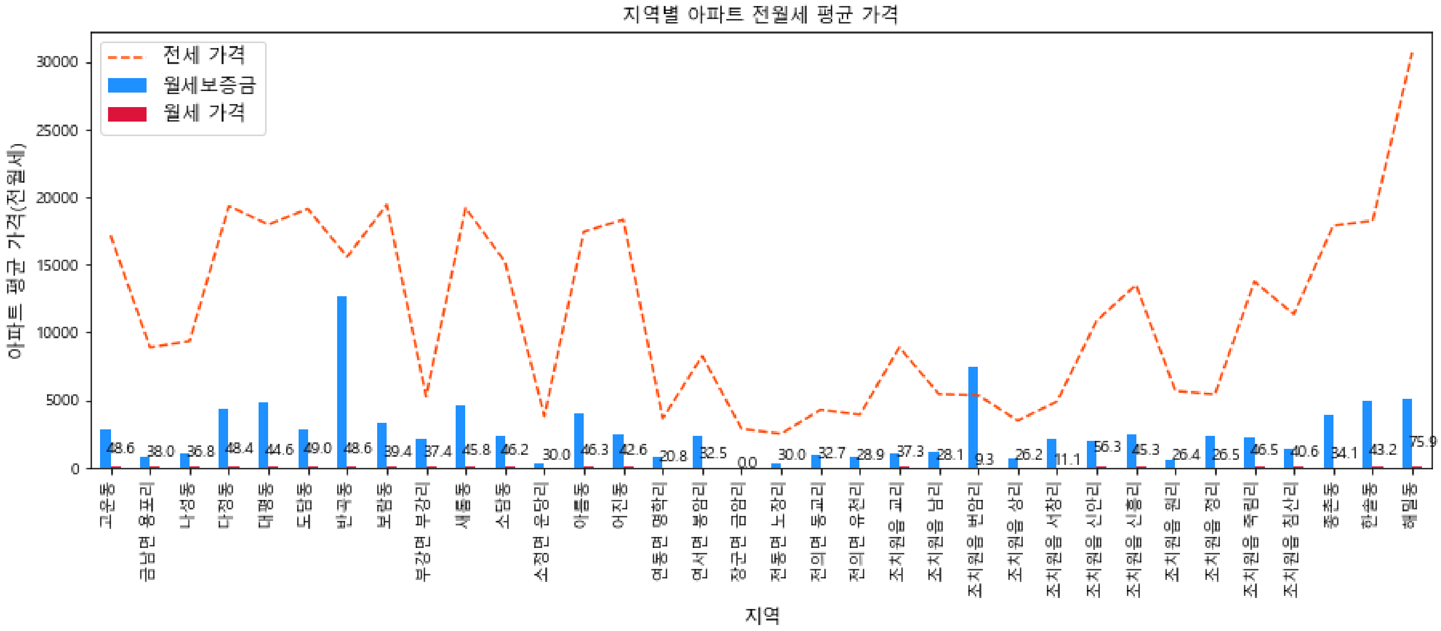
In [ ]:

,

In [7]:

```
#
  _  = pd.read_csv(input_path.joinpath('7.  _  (  )_   .csv'),thousands = ',',encoding='utf-8' )
  _

#
  _  .columns

#
for i in range(len(  _  ['  '])):
    _  ['  '][i] =   _  ['  '][i][7:]

#     ,
data_  _   =   _  .groupby(['  ', '   '])[['  (  )', '  (  )']].mean()
data_  _   = data_  _  .unstack().fillna(0)
data_  _

data_  _  _    = data_  _  ['  (  )','  ']
data_  _  _   = data_  _  ['  (  )','  ']
data_        = data_        ['  (  )','  ']
data_  _  = data_  _  ['  (  )','  ']

#data_  _  ['  (  )',' '] = data_  _
del(data_  _  ['  (  )','  '])
del(data_  _  ['   (  )','  '])

round(data_  _  ,2)

#
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_  _  .plot(kind ='bar', title ='                ', figsize =(15, 5), legend=True, fontsize = 10, color=[ 'dodgerblue', 'crimson'])
plt.plot(data_  _  , linestyle='--', color = 'orangered')
ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('        (   )', fontsize = 12)
ax.legend(['     ','     ', '     '], fontsize=12)

for i, v in enumerate(data_  _  ['  (  )','  ']):
    ax.text(i-0.1, v+v*20, str(round(v, 1)))
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

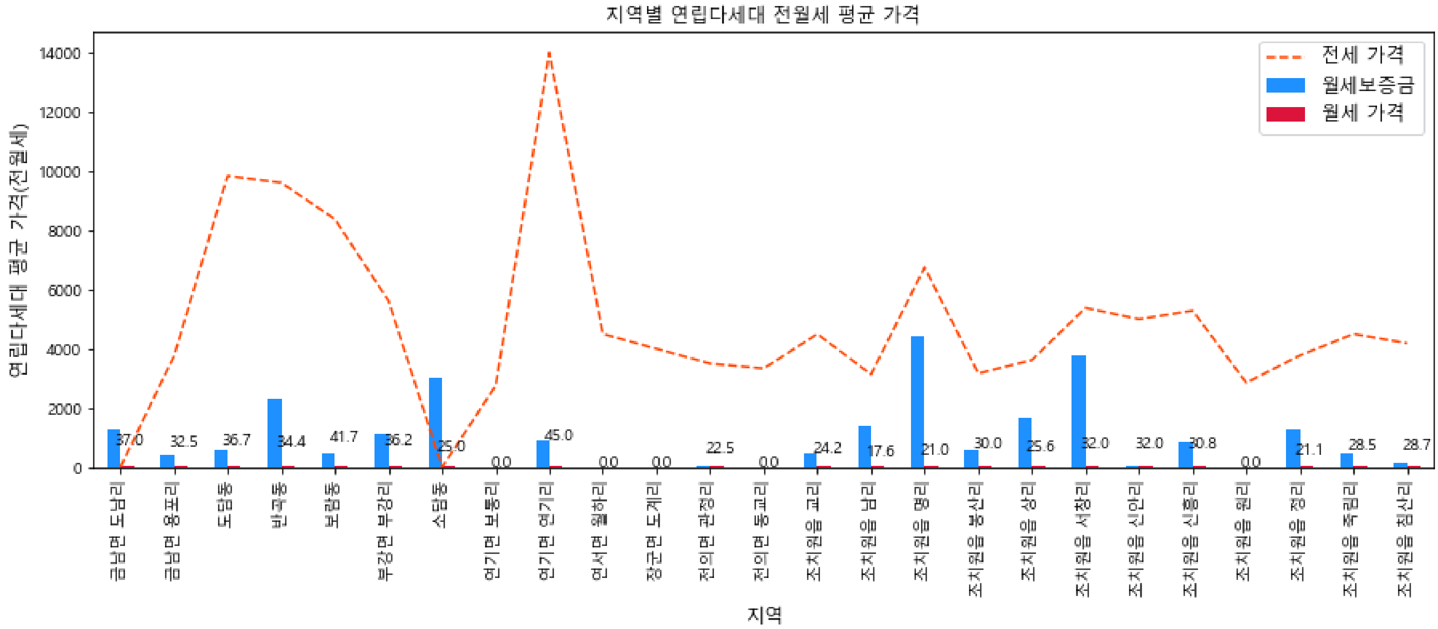


In [8]:

```
#
  _   = pd.read_csv(input_path.joinpath('8.  _     (  )_    .csv'),thousands = ',',encoding='utf-8' )
  _

#
  _   .columns

#
for i in range(len(     _   ['   '])):
      _   ['   '][i] =       _   ['   '][i][7:]

#       ,
data_     _     =      _   .groupby(['   ', '   '])[['  (  )', '  (  )']].mean()
data_     _     = data_     _     .unstack().fillna(0)
data_     _

data_     _   _       = data_     _   ['   (  )','   ']
data_     _   _     = data_     _   ['  (  )','   ']
data_     _     = data_     _   ['   (  )','   ']

#data_     _   ['  (  )',' '] = data_     _
del(data_     _   ['  (  )','  '])
del(data_     _   ['   (  )','  '])

round(data_     _     ,2)

#
```

```
#
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_   _   .plot(kind ='bar', title ='                  ', figsize =(15, 5), legend=True, fontsize = 10, color=[ 'dodgerblue', 'crimson'])
plt.plot(data_    _  , linestyle='--', color = 'orangered')
ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('          (   )', fontsize = 12)
ax.legend(['     ','     ', '     '], fontsize=12)

for i, v in enumerate(data_     _   ['  (  )',' ']):
    ax.text(i-0.1, v+v*20, str(round(v, 1)))
```

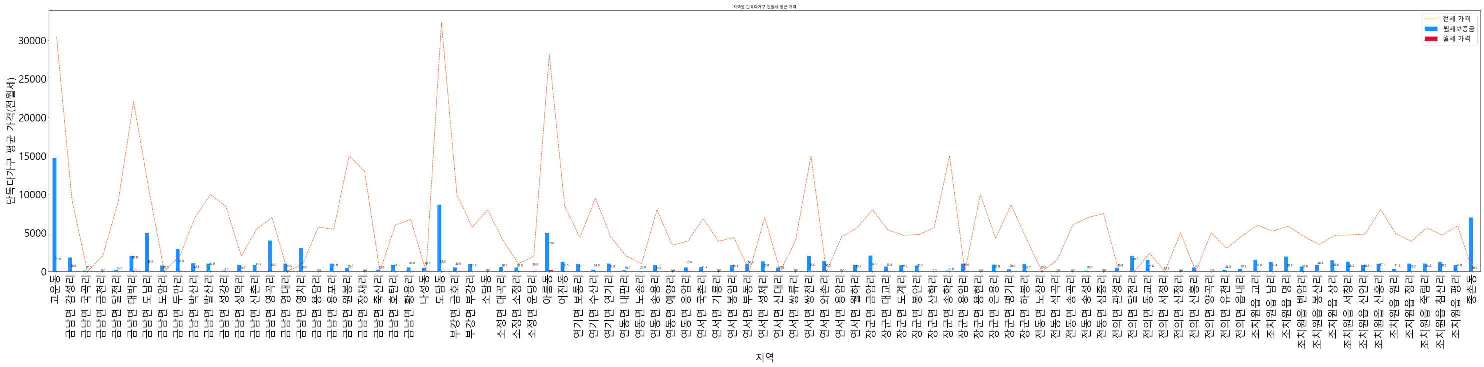
```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```



In [9]:

```
#
    _   = pd.read_csv(input_path.joinpath('9.   _     (   )_     .csv'),thousands = ',',encoding='utf-8' )
    _

#
    _   .columns

#
for i in range(len(     _   ['   '])):
        _   ['   '][i] =      _   ['   '][i][7:]

#       ,
data_    _    =     _   .groupby(['    ', '     '])[['   (  )', '  (  )']].mean()
data_    _    = data_    _    .unstack().fillna(0)
data_    _

data_    _  _      = data_    _   ['   (  )',' ']
data_    _  _   = data_    _   ['  (  )',' ']
data_    _   = data_    _   ['   (  )',' ']

#data_    _   ['  (  )',' '] = data_    _
del(data_    _   ['  (  )',' '])
del(data_    _   ['   (  )',' '])

round(data_    _   ,2)

#
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_   _   .plot(kind ='bar', title ='                  ', figsize =(80, 15), legend=True, fontsize = 32, color=[ 'dodgerblue', 'crimson'])
plt.plot(data_    _  , linestyle='--', color = 'orangered')
ax.set_xlabel('  ', fontsize = 30)
ax.set_ylabel('          (   )', fontsize = 30)
ax.legend(['     ','     ', '     '], fontsize=20)

for i, v in enumerate(data_     _   ['  (  )','  ']):
    ax.text(i-0.1, v+v*20, str(round(v, 1)))
```

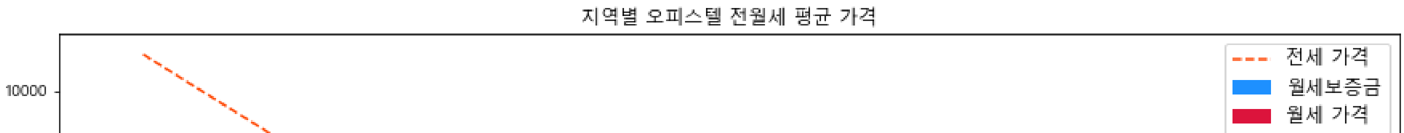
```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

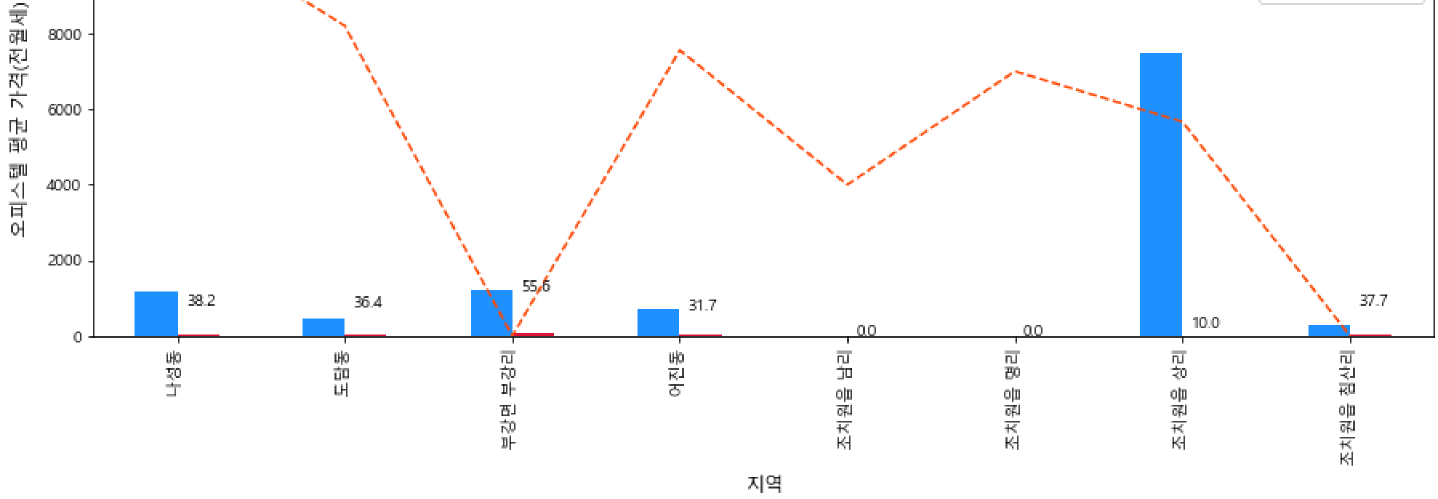


In [10]:

```
#
   _    = pd.read_csv(input_path.joinpath('10.   _   (  )_    .csv'),thousands = ',',encoding='utf-8' )
   _

#
   _   .columns

#
for i in range(len(    _   ['   '])):
      _   ['   '][i] =     _   ['   '][i][7:]

#      ,
data_   _     =    _   .groupby(['    ', '     '])[['   (  )', '  (  )']].mean()
data_   _     = data_   _   .unstack().fillna(0)
data_   _

data_   _   _      = data_    _   ['   (  )','  ']
data_   _   _    = data_   _   ['  (  )','  ']
data_   _    = data_    _   ['   (  )','  ']

#data_   _   ['  (  )','  '] = data_    _
del(data_    _   ['  (  )','  '])
del(data_    _   ['   (  )','  '])

round(data_   _   ,2)

#
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

#
ax = data_   _   .plot(kind ='bar', title ='                  ', figsize =(15, 5), legend=True, fontsize = 10, color=[ 'dodgerblue', 'crimson'])
plt.plot(data_    _   , linestyle='--', color = 'orangered')
ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('         (   )', fontsize = 12)
ax.legend(['     ','     ', '     '], fontsize=12)

for i, v in enumerate(data_    _   ['  (  )','  ']):
    ax.text(i+0.05, v+v*20, str(round(v, 1)))
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```





In [ ]:

```

```

geojson

In [11]:

```
_     = gpd.read_file(input_path.joinpath('19.  _  _    _ .geojson'))
_
```

Out[11]:

	gid	201710_20 _	201810_20 _	201910_20 _	202010_20 _	201710_30 _	201810_30 _	201910_30 _	202010_30 _	201710_40 _	...	202010_80 _	201710_90 _
0	667567	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
1	667568	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2	668566	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
3	668567	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
4	668568	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
47391	919332	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
47392	919333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
47393	919334	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
47394	920332	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
47395	920333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN

47396 rows × 38 columns

47396 rows × 38 columns

In [12]:

```
  = gpd.read_file(input_path.joinpath('23.  _    _  .geojson'))
```

Out[12]:

	BDTYP_CD	BULD_NM	BULD_NM_DC	BULD_SE_CD	BUL_MAN_NO	EMD_CD	GRO_FLO_CO	LNBR_MNNM	LNBR_SLNO	UND_FLO_CO	
0	10299	None	None	0	55874	350	1	177	4	0	MUL
1	10299	None	None	0	55795	340	1	42	23	0	MUL
2	04401	None	None	0	55266	107	7	723	0	2	MUL
3	04001	None	None	0	60761	107	9	722	0	3	MUL
4	03999	None	None	0	55331	107	8	721	0	3	MUL
...	...	...	...	...	...	...	...	...	...	...	
54952	01001	None	None	0	36537	340	1	135	0	0	MUL
54953	14001	None	None	0	40633	340	1	135	0	0	MUL
54954	01001	None	None	0	19892	340	1	135	0	0	MUL
54955	14001	None	None	0	19893	340	1	135	0	0	MUL
54956	01001	119-1	None	0	54174	340	1	119	1	0	MUL

54957 rows × 11 columns

In [13]:

```
  = gpd.read_file(input_path.joinpath('24.  _    _  .geojson'))
```

Out[13]:

	gid	val	geometry
0	667567	NaN	MULTIPOLYGON (((127.12716 36.70773, 127.12716 ...
1	667568	NaN	MULTIPOLYGON (((127.12716 36.70863, 127.12716 ...
2	668566	NaN	MULTIPOLYGON (((127.12829 36.70683, 127.12828 ...
3	668567	NaN	MULTIPOLYGON (((127.12828 36.70773, 127.12828 ...
4	668568	NaN	MULTIPOLYGON (((127.12828 36.70863, 127.12828 ...
...	...	...	...
47391	919332	NaN	MULTIPOLYGON (((127.40956 36.49642, 127.40956 ...

	gid	val	geometry
47392	919333	NaN	MULTIPOLYGON (((127.40956 36.49732, 127.40956 ...
47393	919334	NaN	MULTIPOLYGON (((127.40956 36.49823, 127.40955 ...
47394	920332	NaN	MULTIPOLYGON (((127.41067 36.49642, 127.41067 ...
47395	920333	NaN	MULTIPOLYGON (((127.41067 36.49732, 127.41067 ...

47396 rows × 3 columns

In [14]:

```
_    = gpd.read_file(input_path.joinpath('30.  _   (   ).geojson'))
_
```

Out[14]:

	SIG_CD	SIG_KOR_NM	geometry
0	36110		MULTIPOLYGON (((127.17841 36.59687, 127.17839 ...

In [15]:

```
_    = gpd.read_file(input_path.joinpath('31.  _   (   ).geojson'))
_
```

Out[15]:

	EMD_CD	EMD_KOR_NM	geometry
0	36110101		MULTIPOLYGON (((127.30833 36.47977, 127.30818 ...
1	36110102		MULTIPOLYGON (((127.30480 36.48845, 127.30480 ...
2	36110103		MULTIPOLYGON (((127.29383 36.48508, 127.29381 ...
3	36110104		MULTIPOLYGON (((127.27215 36.46388, 127.27197 ...
4	36110105		MULTIPOLYGON (((127.24631 36.46300, 127.24626 ...
5	36110106		MULTIPOLYGON (((127.25661 36.47033, 127.25653 ...
6	36110107		MULTIPOLYGON (((127.26603 36.49182, 127.26744 ...
7	36110108		MULTIPOLYGON (((127.23978 36.48556, 127.23966 ...
8	36110109		MULTIPOLYGON (((127.24003 36.48809, 127.24004 ...
9	36110110		MULTIPOLYGON (((127.25544 36.49534, 127.25543 ...
10	36110111		MULTIPOLYGON (((127.25357 36.50688, 127.25357 ...
11	36110112		MULTIPOLYGON (((127.23895 36.53520, 127.23960 ...
12	36110113		MULTIPOLYGON (((127.25238 36.51029, 127.25218 ...
13	36110114		MULTIPOLYGON (((127.25357 36.50688, 127.25348 ...
14	36110115		MULTIPOLYGON (((127.26859 36.53949, 127.26858 ...
15	36110116		MULTIPOLYGON (((127.27076 36.52337, 127.27005 ...
16	36110117		MULTIPOLYGON (((127.34569 36.51897, 127.34628 ...
17	36110118		MULTIPOLYGON (((127.31732 36.48175, 127.31730 ...
18	36110250		MULTIPOLYGON (((127.30663 36.60062, 127.30662 ...
19	36110310		MULTIPOLYGON (((127.28524 36.55518, 127.28605

	EMD_CD	EMD_KOR_NM	geometry
20	36110320		MULTIPOLYGON (((127.32202 36.58302, 127.32209 ...
21	36110330		MULTIPOLYGON (((127.36891 36.48959, 127.36891 ...
22	36110340		MULTIPOLYGON (((127.36098 36.49088, 127.36059 ...
23	36110350		MULTIPOLYGON (((127.23952 36.48582, 127.23966 ...
24	36110360		MULTIPOLYGON (((127.19288 36.60110, 127.19298 ...
25	36110370		MULTIPOLYGON (((127.19609 36.60448, 127.19604 ...
26	36110380		MULTIPOLYGON (((127.27948 36.63350, 127.27948 ...
27	36110390		MULTIPOLYGON (((127.20791 36.71901, 127.20781 ...

In [16]:

```
_    = gpd.read_file(input_path.joinpath('32.  _   (   ).geojson'))
_
```

Out[16]:

	ADM_DR_CD	ADM_DR_NM	geometry
0	2901011		MULTIPOLYGON (((127.29172 36.63614, 127.29203 ...
1	2901031		MULTIPOLYGON (((127.24092 36.46741, 127.24087 ...
2	2901032		MULTIPOLYGON (((127.35572 36.55450, 127.35568 ...
3	2901033		MULTIPOLYGON (((127.37706 36.56788, 127.37711 ...
4	2901034		MULTIPOLYGON (((127.36424 36.51353, 127.36375 ...
5	2901035		MULTIPOLYGON (((127.21187 36.57860, 127.21233 ...
6	2901036		MULTIPOLYGON (((127.21134 36.62035, 127.21169 ...
7	2901037		MULTIPOLYGON (((127.20998 36.71848, 127.21068 ...
8	2901038		MULTIPOLYGON (((127.25304 36.69594, 127.25410 ...
9	2901039		MULTIPOLYGON (((127.14980 36.72605, 127.14981 ...
10	2901053		MULTIPOLYGON (((127.26461 36.52069, 127.26459 ...
11	2901060		MULTIPOLYGON (((127.23853 36.53540, 127.23895 ...
12	2901056		MULTIPOLYGON (((127.24233 36.50869, 127.24234 ...
13	2901065		MULTIPOLYGON (((127.31207 36.50321, 127.31209 ...
14	2901062		MULTIPOLYGON (((127.26244 36.47868, 127.26244 ...
15	2901059		MULTIPOLYGON (((127.25182 36.52672, 127.25193 ...
16	2901061		MULTIPOLYGON (((127.26244 36.47868, 127.26181 ...
17	2901064		MULTIPOLYGON (((127.27743 36.47628, 127.27743 ...
18	2901066		MULTIPOLYGON (((127.29381 36.48496, 127.29381 ...

In [17]:

```
 = gpd.read_file(input_path.joinpath('33.  _   .geojson'))
```

Out[17]:

```
                        PNU  JIBUN                                        geometry
     0  3611037031101950001  195-1  MULTIPOLYGON (((127.17929 36.65234, 127.17930
                                                                              ...
     1  3611034029102230001  223-1  MULTIPOLYGON (((127.30726 36.46622, 127.30716
                                                                              ...
     2  3611034028101540000    154  MULTIPOLYGON (((127.30510 36.45196, 127.30502
                                                                              ...
     3  3611034029200630000     63  MULTIPOLYGON (((127.31121 36.47124, 127.31127
                                                                              ...
     4  3611037031104690001  469-1  MULTIPOLYGON (((127.16223 36.64483, 127.16220
                                                                              ...
   ...                  ...    ...                                            ...
198804  3611038024200170001   17-1  MULTIPOLYGON (((127.25132 36.65408, 127.25186
                                                                              ...
198805  3611038024200170008   17-8  MULTIPOLYGON (((127.25186 36.65394, 127.25191
                                                                              ...
198806  3611038024200170009   17-9  MULTIPOLYGON (((127.25189 36.65385, 127.25221
                                                                              ...
198807  3611038024200170002   17-2  MULTIPOLYGON (((127.24888 36.65086, 127.24884
                                                                              ...
198808  3611038024200170010  17-10  MULTIPOLYGON (((127.25175 36.65324, 127.25172
                                                                              ...
```

198809 rows × 3 columns

In [18]:

```
_    = gpd.read_file(input_path.joinpath('16.  _  _    .geojson'))
_
```

Out[18]:

```
          gid  ws_cnt  found_age_1  found_age_2  found_age_3  found_age_4  found_age_5  found_age_6  runoutmon  smbiz_yn  ...  indcd_m_yn  in
    0  667567     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
    1  667568     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
    2  668566     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
    3  668567     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
    4  668568     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
  ...     ...     ...          ...          ...          ...          ...          ...          ...        ...       ...  ...         ...
47391  919332     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
47392  919333     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
47393  919334     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
47394  920332     NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN       NaN  ...        None
```

	gid	ws_cnt	found_age_1	found_age_2	found_age_3	found_age_4	found_age_5	found_age_6	runoutmon	smbiz_yn	...	indcd_m_yn	in
47395	920383	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		None	

47396 rows × 69 columns

In [ ]:

In [19]:

```
#
    = pd.read_csv(input_path.joinpath('14.  _    .csv'))
```

Out[19]:

												lon	lat
0	D		D21	/ /	D21A06	/	G47511		3611025000		90-1, ( )	127.298552	36.599920
1	D		D21	/ /	D21A02	/	G47511		3611037000		40, ( )	127.204298	36.682526
2	F		F15		F15A03		F42201	·	3611036000		322, ( )	127.280990	36.560523
3	Q		Q04		Q04A01		I56194		3611034000		61-3, ( )	127.270561	36.426637
4	D		D11		D11A03		G47599		3611025000		159, (, 2)	127.294177	36.602922
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10768	D		D23	/	D23A04		G45211		3611025000		48-11, ( )	127.299415	36.593481
10769	L		L01		L01A01		L68221		3611034000		529, ( )	127.333858	36.480396
10770	Q		Q06		Q06A02		I56114		3611055000		13, ( )	127.235897	36.500933
10771	N	/ /	N02	/ /	N02A01		R91223		3611056000		51, ( )	127.289353	36.477949
10772	R	/	R05	-	R05A02	/ /	P85620		3611054000		3 160, (, 4)	127.243716	36.506156

10773 rows × 13 columns

In [20]:

```
#
data_     = pd.crosstab(    .    ,     .         , margins=True)
data_
```

Out[20]:

	/ /							/	All
	7	25	74	61	0	3	136	47	353
	13	98	102	184	2	1	284	15	699
	2	2	32	25	0	1	38	11	111
	13	51	83	217	0	3	388	48	803
	5	19	69	61	0	5	230	31	420
	11	14	69	135	8	0	189	10	436
	21	157	159	313	0	4	349	115	1118
	0	11	44	38	0	0	81	38	212
	1	4	22	43	1	0	87	7	165
	3	13	38	45	0	6	115	57	277
	1	28	37	56	3	0	67	2	194
	0	15	14	46	3	0	71	7	156
	5	40	50	162	7	0	164	16	444
	10	78	40	128	3	1	203	2	465
	2	5	22	53	6	0	51	4	143

	/ /							/	All
	7	9	61	139	5	0	182	14	417
	97	105	574	1133	36	3	1402	203	3553
	1	20	48	87	0	3	131	56	346
	1	89	49	77	0	0	174	71	461
All	200	783	1587	3003	74	30	4342	754	10773

In [21]:

```
#
data_    _all = data_     ['All']
data_    _all
```

Out[21]:

```

         353
         699
         111
         803
         420
         436
        1118
         212
         165
         277
         194
         156
         444
         465
         143
         417
        3553
         346
         461
All     10773
Name: All, dtype: int64
```

In [22]:

```
#
    _     = gpd.read_file(input_path.joinpath('32.   _    (   ).geojson'))
    _

    _   ['lon'] = (    _   ['geometry'].bounds['maxx'] +     _   ['geometry'].bounds['minx'])/2
    _   ['lat'] = (    _   ['geometry'].bounds['maxy'] +     _   ['geometry'].bounds['miny'])/2
data_    _lon_lat =     _   [['ADM_DR_NM', 'lon', 'lat']]
data_    _lon_lat=data_    _lon_lat.rename(columns={'ADM_DR_NM':'    '})
data_    _lon_lat
```

Out[22]:

	lon	lat
0	127.282863	36.607199
1	127.279812	36.507771
2	127.330061	36.550175
3	127.381030	36.528753
4	127.290863	36.462920
5	127.207249	36.507231
6	127.247954	36.583444
7	127.199439	36.655099
8	127.253810	36.656332
9	127.167851	36.711410
10	127.265374	36.508349
11	127.236059	36.517374
12	127.247568	36.503792
13	127.307823	36.490650
14	127.252152	36.488832
15	127.251252	36.517544

	lon	lat
15	127.251252	36.517544
~~16~~	~~127.249586~~	~~36.473352~~
17	127.274502	36.470178
18	127.288813	36.478255

In [23]:

```
#          ,
data_    _info = pd.merge( data_    _all,data_    _lon_lat, on ="    ")
data_    _info
```

Out[23]:

	All	lon	lat
0	353	127.236059	36.517374
1	699	127.290863	36.462920
2	111	127.274502	36.470178
3	803	127.265374	36.508349
4	420	127.288813	36.478255
5	436	127.381030	36.528753
6	1118	127.252152	36.488832
7	212	127.307823	36.490650
8	165	127.167851	36.711410
9	277	127.251252	36.517544
10	194	127.279812	36.507771
11	156	127.330061	36.550175
12	444	127.247954	36.583444
13	465	127.207249	36.507231
14	143	127.253810	36.656332
15	417	127.199439	36.655099
16	3553	127.282863	36.607199
17	346	127.247568	36.503792
18	461	127.249586	36.473352

In [24]:

```
#
data_     =
data_     = data_    .sort_values(by='        ')
data_

data_     = data_    [['        ', '      ', 'lon', 'lat']]

#
data_    .rename(columns={'        ':'    '}, inplace = True)
data_
```

Out[24]:

			lon	lat
6091	/ /	16, ( )	127.236487	36.501126
153	/ /	565, ( )	127.220519	36.499410
9564	/ /	19, ( )	127.233984	36.505851
2597	/ /	233, ( )	127.296533	36.601936
3054	/ /	385-3, ( )	127.203729	36.496271
...	...	...	...	...
8580	/	157, ( )	127.262479	36.479318
1639	/	17, ( )	127.294190	36.595325
1642	/	2 64, ( )	127.289099	36.602117
8606	/	62-15, ( )	127.251201	36.487016

10772	/)	127.243716	36.506156
		lon	**lat**

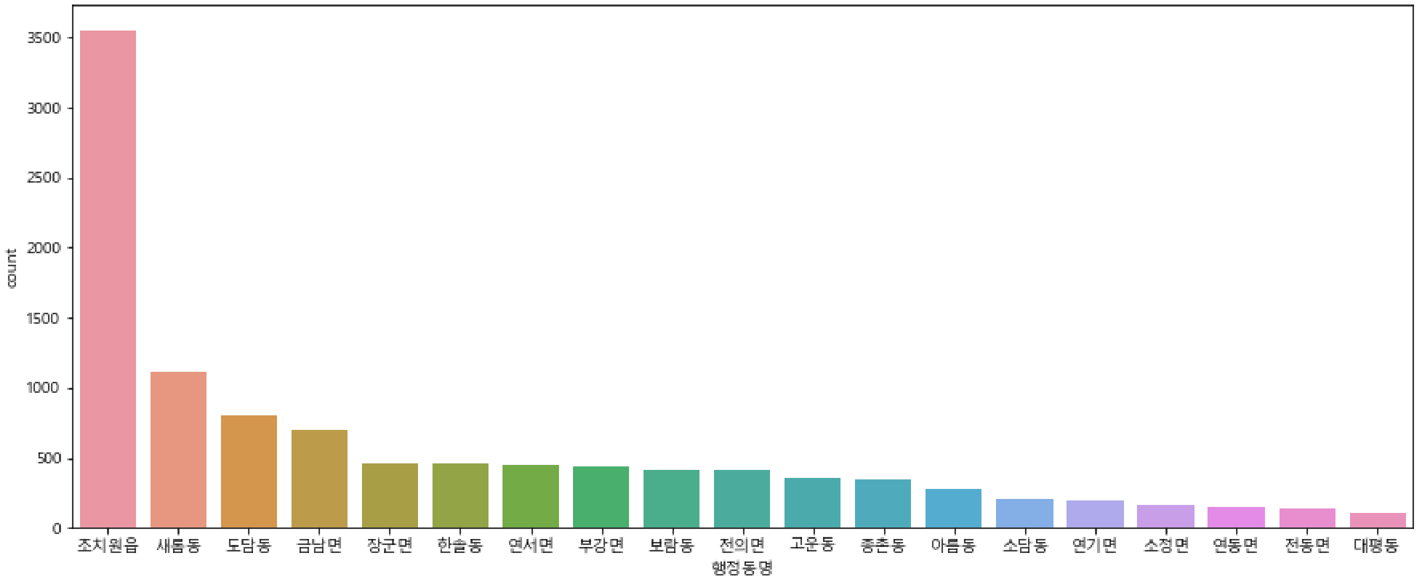
10773 rows × 4 columns

In [25]:

```
#
plt.figure(figsize= (15,6))
sns.countplot(data =      , x="     ", order=    ['     '].value_counts().index)
```

Out[25]:

```
<AxesSubplot:xlabel='     ', ylabel='count'>
```



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [26]:

```
#  ,   ,           float
data_    _info['lon'] = data_    _info.lon.astype(float)
data_    _info['lat'] = data_    _info.lat.astype(float)
data_    _info['All'] = data_    _info.All.astype(float)
data_    _info

#
  _map = folium.Map(location = [data_    _info['lat'].mean(), data_    _info['lon'].mean()],zoom_start = 11)

for item in data_    _info.index:
    latitude = data_    _info.loc[item,'lat']
    longtitude = data_    _info.loc[item,'lon']
    popups = folium.Popup(data_    _info.loc[item, '     '], max_width=100)
    folium.CircleMarker([latitude, longtitude],
                    radius = data_    _info.loc[item,'All']/100,
                    popup = popups,
                    color = 'red',
                    fill = True).add_to(   _map)
  _map
```

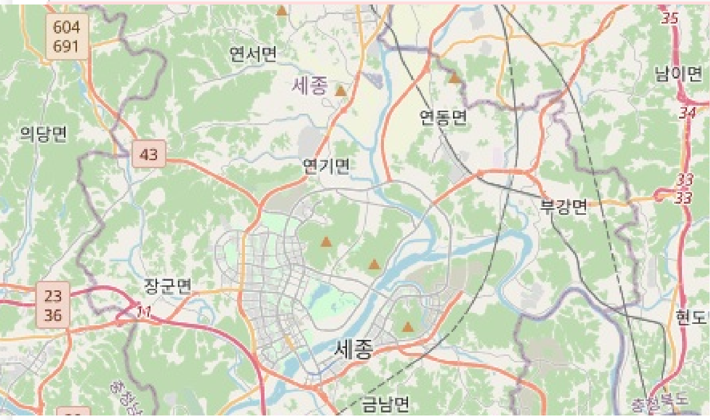
Out[26]:

Make this Notebook Trusted to load map: File -> Trust Notebook

2

In [27]:

```
#
   _    = gpd.read_file(input_path.joinpath('32.   _    (    ).geojson'))
   _

data = data_    _info[['    ', 'All']]
data

#
  _map2 = folium.Map(location = [data_    _info['lat'].mean(), data_    _info['lon'].mean()],zoom_start = 11)

  _map2.choropleth(
   geo_data =      _   ,
   data = data,
   columns=['    ', 'All'],
   key_on = 'feature.properties.ADM_DR_NM',
   fill_color = 'BuPu',
   legend_name ='    ',
   )

  _map2
```


604
691
연서면
세종
남이면
의당면
연동면
43
연기면
부강면
장군면
23
36
세종
금남면
현도
충청북도
35
34
33
33

```
ureWarning: The choropleth  method has been deprecated. Instead use the new C
 notebook 'GeoJSON_and_choropleth' for how to do this.
```

In [ ]:

In [ ]:

In [28]:

```
data_     = 
data_     = data_    [['        ', '      ', 'lon', 'lat']]

#
data_    .rename(columns={'        ':'    '}, inplace = True)
data_


#   ,
data_    ['lon'] = data_    .lon.astype(float)
data_    ['lat'] = data_    .lat.astype(float)


#

    _map = folium.Map(location = [data_    ['lat'].mean(), data_    ['lon'].mean()],zoom_start = 11)

for item in data_    .index:
    latitude = data_    .loc[item,'lat']
    longtitude = data_    .loc[item,'lon']

    if data_    .loc[item, '    '] == '  /  /  ':
        colors = 'dodgerblue'

    elif data_    .loc[item, '    '] == '    ':
        colors = 'burlywood'

    elif data_    .loc[item, '    '] == '      ':
        colors = 'gold'

    elif data_    .loc[item, '    '] == '  ':
        colors = 'darkolivegreen'

    elif data_    .loc[item, '    '] == '  /  ':
        colors = 'slategrey'

    elif data_    .loc[item, '    '] == '  ' :
        colors = 'salmon'

    elif data_    .loc[item, '    '] == '    ' :
        colors = 'blueviolet'

    elif data_    .loc[item, '    '] == '  ':
        colors = 'lihgtpink'
    popups = folium.Popup(data_    .loc[item, '      '], max_width=150)
    folium.CircleMarker([latitude, longtitude],
                        popup = popups,
                 color = colors,
                 fill = True).add_to(    _map)
   _map
```

```
/opt/app-root/lib/python3.6/site-packages/pandas/core/frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
```

Out[28]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [ ]:

In [29]:

```
  _

#

data_  _   =   _ [['  ', '   (  )']]
for i in range(len(data_  _  ['  '])):
   if len(data_  _  .loc[i, '  ']) == 9:
      data_  _  ['  '][i] = data_  _  ['  '][i][:5]
   elif len(data_  _  .loc[i, '  ']) == 8:

      if data_  _  .loc[i, '  '][1] == ' ':
         data_  _  ['  '][i] = data_  _  ['  '][i][:5]
      else:
```

```
    data_  _  ['   '][i] = data_  _  ['   '][i][:5]


data_  _
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/app-root/lib/python3.6/site-packages/IPython/core/interactiveshell.py:3343: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exec(code_obj, self.user_global_ns, self.user_ns)
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[29]:

		( )
0		23000.0
1		32000.0
2		32500.0
3		25100.0
4		32250.0
...	...	...
18404		66800.0
18405		40000.0
18406		134000.0
18407		72800.0
18408		74800.0

18409 rows × 2 columns

In [30]:

```
#        ( )
data_  _    = data_  _  .groupby(data_  _  .   )['   ( )'].mean()
data_  _

data_  _    = data_  _  .reset_index()
data_  _

for i in range(len(data_  _  ['  '])):
    data_  _  ['   '][i] = data_  _  ['   '][i][1:]

data_  _
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

Out[30]:

	( )
0	54000.000000
1	34431.779738
2	13959.120370
3	10585.482759

	( )
4	62177.157095
5	73175.011976
6	44911.933291
7	57494.951456
8	65879.284211
9	7851.290323
10	66054.370662
11	56853.738302
12	6392.857143
13	35044.949764
14	39625.319905
15	6625.719101
16	11966.279070
17	2771.408451
18	5652.380952
19	7903.365385
20	16970.891432
21	39196.429032
22	40881.186555

In [31]:

```
#
  _   = gpd.read_file(input_path.joinpath('31.  _   (  ).geojson'))
  _
```

Out[31]:

	EMD_CD	EMD_KOR_NM	geometry
0	36110101		MULTIPOLYGON (((127.30833 36.47977, 127.30818 ...
1	36110102		MULTIPOLYGON (((127.30480 36.48845, 127.30480 ...
2	36110103		MULTIPOLYGON (((127.29383 36.48508, 127.29381 ...
3	36110104		MULTIPOLYGON (((127.27215 36.46388, 127.27197 ...
4	36110105		MULTIPOLYGON (((127.24631 36.46300, 127.24626 ...
5	36110106		MULTIPOLYGON (((127.25661 36.47033, 127.25653 ...
6	36110107		MULTIPOLYGON (((127.26603 36.49182, 127.26744 ...
7	36110108		MULTIPOLYGON (((127.23978 36.48556, 127.23966 ...
8	36110109		MULTIPOLYGON (((127.24003 36.48809, 127.24004 ...
9	36110110		MULTIPOLYGON (((127.25544 36.49534, 127.25543 ...
10	36110111		MULTIPOLYGON (((127.25357 36.50688, 127.25357 ...
11	36110112		MULTIPOLYGON (((127.23895 36.53520, 127.23960 ...
12	36110113		MULTIPOLYGON (((127.25238 36.51029, 127.25218 ...
13	36110114		MULTIPOLYGON (((127.25357 36.50688, 127.25348 ...
14	36110115		MULTIPOLYGON (((127.26859 36.53949, 127.26858 ...
15	36110116		MULTIPOLYGON (((127.27076 36.52337, 127.27005 ...
16	36110117		MULTIPOLYGON (((127.34569 36.51897, 127.34628 ...

	EMD_CD	EMD_KOR_NM	geometry
17	36110118		MULTIPOLYGON (((127.31732 36.48175, 127.31730 ...
18	36110250		MULTIPOLYGON (((127.30663 36.60062, 127.30662 ...
19	36110310		MULTIPOLYGON (((127.28524 36.55518, 127.28605 ...
20	36110320		MULTIPOLYGON (((127.32202 36.58302, 127.32209 ...
21	36110330		MULTIPOLYGON (((127.36891 36.48959, 127.36891 ...
22	36110340		MULTIPOLYGON (((127.36098 36.49088, 127.36059 ...
23	36110350		MULTIPOLYGON (((127.23952 36.48582, 127.23966 ...
24	36110360		MULTIPOLYGON (((127.19288 36.60110, 127.19298 ...
25	36110370		MULTIPOLYGON (((127.19609 36.60448, 127.19604 ...
26	36110380		MULTIPOLYGON (((127.27948 36.63350, 127.27948 ...
27	36110390		MULTIPOLYGON (((127.20791 36.71901, 127.20781 ...

In [32]:

```
#
  _  ['lon']= (  _  ['geometry'].bounds['maxx'] +   _  ['geometry'].bounds['minx'])/2
  _  ['lat'] = (  _  ['geometry'].bounds['maxy'] +   _  ['geometry'].bounds['miny'])/2

data_  _lon_lat =   _  [['EMD_KOR_NM', 'lon', 'lat']]
data_  _lon_lat = data_  _lon_lat.rename(columns={'EMD_KOR_NM':'  '})
data_  _lon_lat
```

Out[32]:

	lon	lat
0	127.310382	36.491414
1	127.301194	36.485012
2	127.288813	36.478255
3	127.274502	36.470178
4	127.246671	36.471256
5	127.254392	36.477017
6	127.264276	36.486736
7	127.249013	36.484422
8	127.243697	36.493832
9	127.265374	36.500819
10	127.247569	36.503792
11	127.236093	36.517375
12	127.251275	36.517544
13	127.262456	36.515875
14	127.253918	36.534591
15	127.271144	36.527016
16	127.333018	36.525097
17	127.329770	36.496421
18	127.282863	36.607327
19	127.279813	36.507784
20	127.330061	36.550718
21	127.381030	36.528753
22	127.290864	36.462920
23	127.207249	36.507231
24	127.247954	36.583444
25	127.199440	36.655088
26	127.253811	36.656332

	lon	lat
27	127.167851	36.711410

In [33]:

```
#

data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _  .loc[len(data_  _  )] = ['   ', 0]
data_  _
```

Out[33]:

	( )
0	54000.000000
1	34431.779738
2	13959.120370
3	10585.482759
4	62177.157095
5	73175.011976
6	44911.933291
7	57494.951456
8	65879.284211
9	7851.290323
10	66054.370662
11	56853.738302
12	6392.857143
13	35044.949764
14	39625.319905
15	6625.719101
16	11966.279070
17	2771.408451
18	5652.380952
19	7903.365385
20	16970.891432
21	39196.429032
22	40881.186555
23	0.000000
24	0.000000
25	0.000000
26	0.000000
27	0.000000
28	0.000000

In [34]:

```
data_  _  ['   '][1] = '   '
data_  _  ['   '][8] = '   '
data_  _  ['   '][11] ='   '
data_  _  ['   '][14] ='   '
data_  _  ['   '][15] ='   '
data_  _  ['   '][16] ='   '
data_  _  ['   '][17]='   '
data_  _  ['   '][18]='   '
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  """Entry point for launching an IPython kernel.
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [35]:

```
data_  _  _info = pd.merge(data_  _lon_lat, data_  _  , on ='  ', how="outer")
data_  _  _info
```

Out[35]:

	lon	lat	( )
0	127.310382	36.491414	57494.951456
1	127.301194	36.485012	NaN
2	127.288813	36.478255	NaN
3	127.274502	36.470178	73175.011976
4	127.246671	36.471256	54000.000000
5	127.246671	36.471256	0.000000
6	127.254392	36.477017	40881.186555
7	127.264276	36.486736	10585.482759
8	127.249013	36.484422	66054.370662
9	127.243697	36.493832	62177.157095
10	127.265374	36.500819	NaN
11	127.247569	36.503792	39196.429032
12	127.236093	36.517375	NaN
13	127.251275	36.517544	35044.949764
14	127.262456	36.515875	44911.933291
15	127.253918	36.534591	0.000000
16	127.271144	36.527016	0.000000
17	127.333018	36.525097	0.000000
18	127.329770	36.496421	0.000000
19	127.282863	36.607327	16970.891432
20	127.279813	36.507784	0.000000
21	127.330061	36.550718	39625.319905
22	127.381030	36.528753	65879.284211

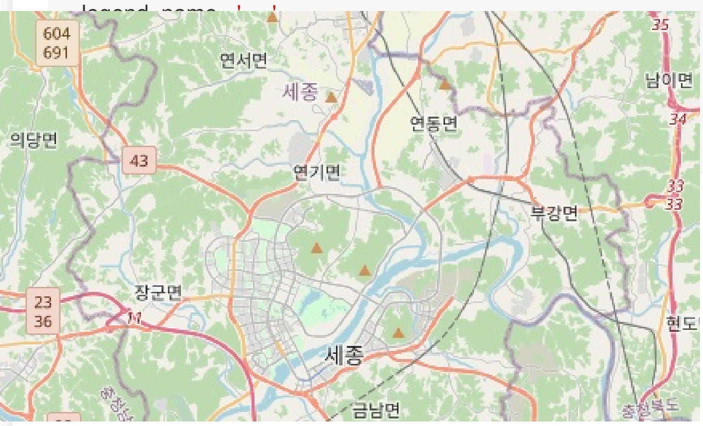
	lon	lat	( )
23	127.290864	36.462900	34431.779738
24	127.207249	36.507231	11966.279070
25	127.247954	36.583444	6625.719101
26	127.199440	36.655088	5652.380952
27	127.253811	36.656332	2771.408451
28	127.167851	36.711410	56853.738302
29	NaN	NaN	13959.120370
30	NaN	NaN	7851.290323
31	NaN	NaN	6392.857143
32	NaN	NaN	7903.365385

In [36]:

```
#  ,     float
data_  _  _info['lon'] = data_  _  _info.lon.astype(float)
data_  _  _info['lat'] = data_  _  _info.lat.astype(float)
```

In [37]:

```
#
   _    = gpd.read_file(input_path.joinpath('31.  _   (   ).geojson'))
   _

   _map = folium.Map(location = [data_  _  _info['lat'].mean(), data_  _  _info['lon'].mean()] ,zoom_start = 11)

data = data_  _  _info[['   ','   (  )']]

folium.Choropleth(
    geo_data =     _   ,
    data = data,
    columns=['  ','   (  )'],
    key_on = 'feature.properties.EMD_KOR_NM',
    fill_color = 'BuPu',
    legend_name = '  '
```



In [ ]:

```

```

，

In [38]:

```
#
 = pd.read_csv(input_path.joinpath('20.  _   .csv') )


#
 = pd.read_csv(input_path.joinpath('21.  _   .csv') )
```

Out[38]:

0	201701	3				
1	201701	9				
2	201701	13				
3	201701	3				
4	201701	1				1
...	...	...	...	...	...	...
80987	202101	22				
80988	202101	2				1
80989	202101	35				
80990	202101	13				1
80991	202101	84				

80992 rows × 6 columns

In [39]:

```
#
data_   =  .groupby('     ')['    '].agg('sum')
data_

#
data_   =  .groupby('     ')['    '].agg('sum')
data_
```

Out[39]:

```
        104194
         77692
         11042
         24195
        150364
         63403
         36692
        131275
         66903
         10559
         77777
         19753
         19127
         54553
         50845
         17857
         35215
        393101
        101396
         83269
Name:     , dtype: int64
```

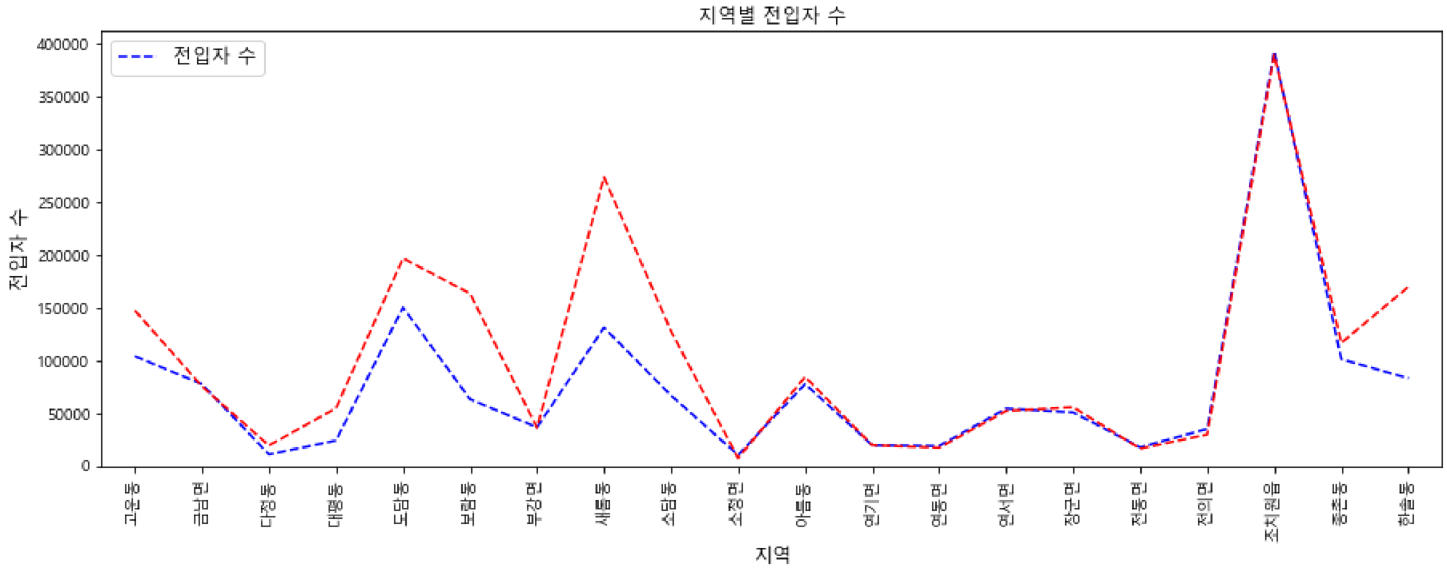
In [40]:

In [40]:

```
#
ax = data_    .plot(kind ='bar',title ='         ', figsize =(15, 5), legend=True, fontsize = 10, color='white')
ax.plot(data_    , linestyle='--', color ='blue')
ax.plot(data_    , linestyle='--', color='red')

ax.set_xlabel('  ', fontsize = 12)
ax.set_ylabel('     ', fontsize = 12)
ax.legend(['     '], fontsize=12)
```

Out[40]:

```
<matplotlib.legend.Legend at 0x7f2112024198>
```



In [ ]:

In [41]:

```
#
  = pd.read_csv(input_path.joinpath('12.   _ ( )_    .csv') , encoding='utf-8')

#
for i in range(len(  ['   '])):
     ['   '][i] =   ['   '][i][8:12]
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[41]:

							( )	( )		
0				-	201801	1	331.0	1,050		NaN
1				-	201801	1	330.0	1,190		NaN
2				-	201801	1	331.0	1,290		NaN
3				-	201801	1	2645.0	10,320		NaN
4				-	201801	1	1652.0	11,450		NaN
...	...	...	...	...	...	...	...	...	...	...
35672				8m	201712	30	155.0	3,525		NaN
35673				8m	201712	30	199.0	4,500		NaN
35674				8m	201712	30	324.0	4,998		NaN

35675	8m	201712	30	30.0 ( )	459 ( )	NaN
35676	8m	201712	31	364.0	22,000	NaN

35677 rows × 10 columns

In [42]:

```
pd.set_option('display.max_rows', 700)

#
  ['   ( )'] =  ['   ( )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_   = round(  .groupby(['   ',' '])[['   ( )','   ( )']].mean(),3)
data_   = data_  .reset_index()
data_
```

Out[42]:

	( )	( )
0	4341.750	32.935
1	48542.837	461.911
2	7507.750	89.330
3	769.556	10.837
4	190000.000	1397.900
5	79145.417	1386.710
6	26593.061	936.914
7	10039.000	265.000
8	25759.342	1184.218
9	14822.839	268.825
10	1041.949	38.188
11	14193.846	407.885
12	27482.000	273.420
13	129.000	2.364
14	28333.333	371.333
15	9444.073	1350.164
16	20762.500	392.375
17	18142.635	805.417
18	30000.000	300.000
19	11386.667	2816.000
20	55305.214	1722.143
21	422455.500	5105.667
22	41564.294	356.882
23	283.000	50.667
24	9333.667	261.667
25	404578.429	4588.286
26	567.000	36.000
27	45116.086	296.407
28	57472.187	563.240
29	2532.000	522.000
30	539130.750	2959.750
31	40453.718	848.910
32	23802.500	2403.833
33	24792.738	1254.615
34	9467.938	297.147
35	2715.809	120.009
36	16939.500	684.999
37	800.000	635.000
38	5906.932	1026.870
39	52329.250	746.700

40	15680.207	746.891
41	8300.000	684.000
42	18665.667	405.667
43	4418.389	130.167
44	1044.000	5.000
45	526990.000	9060.000
46	671.375	220.125
47	1446476.000	4211.000
48	109310.618	3001.560
49	16124.432	1440.765
50	8192.565	306.819
51	1605.250	193.785
52	5879.000	778.500
53	3703.312	994.035
54	21670.500	1171.250
55	14575.807	1238.976
56	43028.000	1665.650
57	4387.250	708.250
58	43027.050	336.278
59	1376042.500	2707.002
60	3561999.000	650000.000
61	30995.500	915.796
62	46301.344	1292.953
63	64567.307	1217.847
64	1144.700	58.105
65	8823.889	125.999
66	8392.352	688.958
67	24657.154	1859.885
68	27061.901	668.118
69	363.000	15.000
70	17910.000	203.555
71	1025.000	28.230
72	5110.455	126.909
73	36190.000	883.667
74	36252.091	1935.273
75	3146.800	334.400
76	33143.525	1962.033
77	21730.041	321.750
78	3263.004	67.649
79	24000.000	1587.000
80	15064.933	1002.253
81	4007.500	157.000
82	23548.984	718.900
83	5039.000	247.000
84	17666.667	649.667
85	1342.500	33.000
86	13762.667	173.511
87	28956.350	1362.554
88	8113.400	321.800
89	28386.029	1383.892
90	12318.337	357.688
91	1386.360	62.246
92	16316.225	571.575
93	5940.000	460.200

	( )	( )
94	3090.000	134.250
95	32773.333	1450.667
96	6365.149	1136.630
97	15627.400	446.467
98	20112.262	921.078
99	22833.333	364.000
100	4530.000	394.111
101	435.407	21.369
102	430506.714	18138.714
103	18288.267	741.633
104	1500.000	509.000
105	28873.798	1203.600
106	16979.480	361.648
107	1356.200	59.667
108	13435.952	548.214
109	21750.000	618.600
110	14052.820	1078.746
111	1855.227	49.544
112	23515.618	782.244
113	5841.333	165.667
114	14012.000	579.000
115	3655.789	157.513
116	4867.000	72.000
117	264583.000	14230.750
118	16829.610	1958.695
119	988.333	281.000
120	16636.008	1647.628
121	7848.952	460.468
122	769.955	115.153
123	4237.733	497.667
124	1578.000	740.500
125	2630.000	352.000
126	3170.344	1286.666
127	29630.400	1987.834
128	11710.181	1160.431
129	4615.167	336.833
130	5197.333	422.333
131	2959.125	754.000
132	254277.108	8508.359
133	1152.779	126.782
134	12151.500	2202.000
135	11631.424	909.082
136	9594.786	349.843
137	1285.705	82.039
138	2867.962	446.853
139	1920.000	443.000
140	3373.968	1057.189
141	8952.872	352.872
142	9105.671	617.128
143	1556.667	168.667
144	5687.000	200.000
145	25400.000	696.000
146	67493.333	1996.400
147	26120.400	568.800

148	4302.944	482.056
149	43922.000	703.000
150	28969.820	702.055
151	29548.042	757.905
152	12050.295	151.459
153	989.070	25.215
154	990.000	198.000
155	6500.000	52.000
156	19583.500	897.500
157	7084.811	439.684
158	23097.429	455.143
159	34234.252	807.118
160	25015.667	266.667
161	5090.000	101.000
162	11500.000	150.500
163	344089.408	9120.551
164	35425.187	329.925
165	302024.463	2169.572

In [43]:

```
#       - ,  -
data_ _  = data_  [['  ',  '   (  )']]
data_ _  = data_  [['  ',  '   ( )']]
```

In [ ]:

```
```

In [ ]:

```
```

In [44]:

```
# 31.  _    (   ).geojson              ,
 _lon_lat = data_  _lon_lat
 _lon_lat
```

Out[44]:

	lon	lat
0	127.310382	36.491414
1	127.301194	36.485012
2	127.288813	36.478255
3	127.274502	36.470178
4	127.246671	36.471256
5	127.254392	36.477017
6	127.264276	36.486736
7	127.249013	36.484422
8	127.243697	36.493832
9	127.265374	36.500819
10	127.247569	36.503792
11	127.236093	36.517375
12	127.251275	36.517544
13	127.262456	36.515875
14	127.253918	36.534591

	lon	lat
14	127.253918	36.534591
~~**15**~~	~~127.271144~~	~~36.527016~~
16	127.333018	36.525097
17	127.329770	36.496421
18	127.282863	36.607327
19	127.279813	36.507784
20	127.330061	36.550718
21	127.381030	36.528753
22	127.290864	36.462920
23	127.207249	36.507231
24	127.247954	36.583444
25	127.199440	36.655088
26	127.253811	36.656332
27	127.167851	36.711410

In [45]:

```
data_   = data_  .reset_index()
data_
```

Out[45]:

	index	( )	( )
0	0	4341.750	32.935
1	1	48542.837	461.911
2	2	7507.750	89.330
3	3	769.556	10.837
4	4	190000.000	1397.900
5	5	79145.417	1386.710
6	6	26593.061	936.914
7	7	10039.000	265.000
8	8	25759.342	1184.218
9	9	14822.839	268.825
10	10	1041.949	38.188
11	11	14193.846	407.885
12	12	27482.000	273.420
13	13	129.000	2.364
14	14	28333.333	371.333
15	15	9444.073	1350.164
16	16	20762.500	392.375
17	17	18142.635	805.417
18	18	30000.000	300.000
19	19	11386.667	2816.000
20	20	55305.214	1722.143
21	21	422455.500	5105.667
22	22	41564.294	356.882
23	23	283.000	50.667
24	24	9333.667	261.667
25	25	404578.429	4588.286
26	26	567.000	36.000
27	27	45116.086	296.407
28	28	57472.187	563.240
29	29	2532.000	522.000
30	30	539130.750	2959.750
31	31	40453.718	848.910
32	32	23802.500	2403.833

	index	(3)	(5)
33	33	24792.738	1254.615
34	34	9467.938	297.147
35	35	2715.809	120.009
36	36	16939.500	684.999
37	37	800.000	635.000
38	38	5906.932	1026.870
39	39	52329.250	746.700
40	40	15680.207	746.891
41	41	8300.000	684.000
42	42	18665.667	405.667
43	43	4418.389	130.167
44	44	1044.000	5.000
45	45	526990.000	9060.000
46	46	671.375	220.125
47	47	1446476.000	4211.000
48	48	109310.618	3001.560
49	49	16124.432	1440.765
50	50	8192.565	306.819
51	51	1605.250	193.785
52	52	5879.000	778.500
53	53	3703.312	994.035
54	54	21670.500	1171.250
55	55	14575.807	1238.976
56	56	43028.000	1665.650
57	57	4387.250	708.250
58	58	43027.050	336.278
59	59	1376042.500	2707.002
60	60	3561999.000	650000.000
61	61	30995.500	915.796
62	62	46301.344	1292.953
63	63	64567.307	1217.847
64	64	1144.700	58.105
65	65	8823.889	125.999
66	66	8392.352	688.958
67	67	24657.154	1859.885
68	68	27061.901	668.118
69	69	363.000	15.000
70	70	17910.000	203.555
71	71	1025.000	28.230
72	72	5110.455	126.909
73	73	36190.000	883.667
74	74	36252.091	1935.273
75	75	3146.800	334.400
76	76	33143.525	1962.033
77	77	21730.041	321.750
78	78	3263.004	67.649
79	79	24000.000	1587.000
80	80	15064.933	1002.253
81	81	4007.500	157.000
82	82	23548.984	718.900
83	83	5039.000	247.000
84	84	17666.667	649.667
85	85	1342.500	33.000
86	86	13762.667	173.511

	index	()	()
87	87	28956.350	1362.554
88	88	8113.400	321.800
89	89	28386.029	1383.892
90	90	12318.337	357.688
91	91	1386.360	62.246
92	92	16316.225	571.575
93	93	5940.000	460.200
94	94	3090.000	134.250
95	95	32773.333	1450.667
96	96	6365.149	1136.630
97	97	15627.400	446.467
98	98	20112.262	921.078
99	99	22833.333	364.000
100	100	4530.000	394.111
101	101	435.407	21.369
102	102	430506.714	18138.714
103	103	18288.267	741.633
104	104	1500.000	509.000
105	105	28873.798	1203.600
106	106	16979.480	361.648
107	107	1356.200	59.667
108	108	13435.952	548.214
109	109	21750.000	618.600
110	110	14052.820	1078.746
111	111	1855.227	49.544
112	112	23515.618	782.244
113	113	5841.333	165.667
114	114	14012.000	579.000
115	115	3655.789	157.513
116	116	4867.000	72.000
117	117	264583.000	14230.750
118	118	16829.610	1958.695
119	119	988.333	281.000
120	120	16636.008	1647.628
121	121	7848.952	460.468
122	122	769.955	115.153
123	123	4237.733	497.667
124	124	1578.000	740.500
125	125	2630.000	352.000
126	126	3170.344	1286.666
127	127	29630.400	1987.834
128	128	11710.181	1160.431
129	129	4615.167	336.833
130	130	5197.333	422.333
131	131	2959.125	754.000
132	132	254277.108	8508.359
133	133	1152.779	126.782
134	134	12151.500	2202.000
135	135	11631.424	909.082
136	136	9594.786	349.843
137	137	1285.705	82.039
138	138	2867.962	446.853
139	139	1920.000	443.000
140	140	3373.968	1057.189

	index	()	()
141	141	8952.872	352.872
142	142	9105.671	617.128
143	143	1556.667	168.667
144	144	5687.000	200.000
145	145	25400.000	696.000
146	146	67493.333	1996.400
147	147	26120.400	568.800
148	148	4302.944	482.056
149	149	43922.000	703.000
150	150	28969.820	702.055
151	151	29548.042	757.905
152	152	12050.295	151.459
153	153	989.070	25.215
154	154	990.000	198.000
155	155	6500.000	52.000
156	156	19583.500	897.500
157	157	7084.811	439.684
158	158	23097.429	455.143
159	159	34234.252	807.118
160	160	25015.667	266.667
161	161	5090.000	101.000
162	162	11500.000	150.500
163	163	344089.408	9120.551
164	164	35425.187	329.925
165	165	302024.463	2169.572

In [46]:

```
#

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '    ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '    ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '    ']
data_    = data_  .set_index('  ')

data_    = data_  [data_  ['  '] == '  ']
data_    = data_  .set_index('  ')
data_    = data_ [data_ ['   '] == '   ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '    ']
data_    = data_   .set_index('  ')

data_     = data_ [data_ ['   '] == '    ']
data_     = data_     .set_index('  ')

data_    = data_ [data_ ['   '] == '   ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '   ']
data_    = data_   .set_index('  ')

data_    = data_ [data_ ['   '] == '   ']
data_    = data_   .set_index('  ')
```

In [47]:

```
#

  _   = Bar(data_   ['    (  )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 10, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 750, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 750, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 400, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

  _   = Bar(data_   ['    (  )'], width = 500, height = 200).axis_titles(x = '  ', y = '    ')
  _   = Bar(data_   ['    ( )'], width = 500, height = 200).axis_titles(x = '  ', y = '    ')
```

```
_   = Bar(data_   ['    (  )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 650, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 700, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 700, height = 200).axis_titles(x = '  ', y = '    ')

 _   = Bar(data_    ['    (  )'], width = 550, height = 200).axis_titles(x = '  ', y = '    ')
 _   = Bar(data_    ['    ( )'], width = 550, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')

_   = Bar(data_   ['    (  )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
_   = Bar(data_   ['    ( )'], width = 100, height = 200).axis_titles(x = '  ', y = '    ')
```

In [48]:

```
#

  _    = gpd.read_file(input_path.joinpath('31.   _    (   ).geojson'))
  _

 _map = folium.Map(location = [  _lon_lat['lat'].mean(),   _lon_lat['lon'].mean()], zoom_start = 11)

#
 _map.choropleth(
    geo_data =      _   ,
    key_on = 'feature.properties.EMD_KOR_NM',
    fill_color = 'White',
    fill_opacity = 0.3,
    legend_name ='     ',
    )

#                 ,
for i in range(len(  _lon_lat['   '])):
    latitude_   =   _lon_lat['lat'][i]
    longtitude_   =   _lon_lat['lon'][i]

    latitude_   = (  _lon_lat['lat'][i]+0.002)
    longtitude_   = (  _lon_lat['lon'][i]+0.002)

    popups_   = folium.Popup(  _lon_lat['   '][i], max_width=100)
    popups_   = folium.Popup(  _lon_lat['   '][i], max_width=100)

    if(  _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(   _  , width = 450, height = 250))
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(   _  , width = 450, height = 250))

    elif(  _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _  , width = 750, height = 250))
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _  , width = 750, height = 250))

    elif(  _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _    , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _    , width = 150, height = 250))

    elif(  _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _    , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _    , width = 150, height = 250))
    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=850).add_child(
            folium.Vega(  _  , width = 800, height = 250))
        popups_   = folium.Popup(max_width=850).add_child(
            folium.Vega(  _  , width = 800, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))
        popups_   = folium.Popup(max_width=500).add_child(
            folium.Vega(  _  , width = 450, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(  _  , width = 150, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=600).add_child(
            folium.Vega(  _  , width = 550, height = 250))
        popups_   = folium.Popup(max_width=600).add_child(
            folium.Vega(  _  , width = 550, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=750).add_child(
            folium.Vega(  _  , width = 700, height = 250))
        popups_   = folium.Popup(max_width=750).add_child(
            folium.Vega(  _   , width = 700, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(  _  , width = 750, height = 250))
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(  _  , width = 750, height = 250))

    elif(  _lon_lat['  '][i] == '  '):
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(  _  , width = 750, height = 250))
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(     , width = 750, height = 250))
    elif( _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _   , width = 750, height = 250))
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _   , width = 750, height = 250))

    elif( _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _   , width = 780, height = 250))
        popups_   = folium.Popup(max_width=800).add_child(
            folium.Vega(   _   , width = 780, height = 250))

    elif( _lon_lat['   '][i] == '    '):
        popups_   = folium.Popup(max_width=650).add_child(
            folium.Vega(    _    , width = 600, height = 250))
        popups_   = folium.Popup(max_width=650).add_child(
            folium.Vega(    _    , width = 600, height = 250))

    elif( _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))

    elif( _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))

    elif( _lon_lat['   '][i] == '   '):
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))
        popups_   = folium.Popup(max_width=200).add_child(
            folium.Vega(   _   , width = 150, height = 250))

    folium.CircleMarker([latitude_  , longtitude_  ],
                        popup = popups_  ,
                        color = 'crimson',
                        fill = True).add_to(  _map)

    folium.CircleMarker([latitude_  , longtitude_  ],
                        popup = popups_  ,
                        color = 'midnightblue',
                        fill = True).add_to(  _map)

  _map
```

```
/opt/app-root/lib/python3.6/site-packages/folium/folium.py:415: FutureWarning: The choropleth  method has been deprecated. Instead use the new C
horopleth class, which has the same arguments. See the example notebook 'GeoJSON_and_choropleth' for how to do this.
  FutureWarning
```

Out[48]:

```
Make this Notebook Trusted to load map: File -> Trust Notebook
```

In [ ]:

```

```

In [ ]:

```

```

(- -/ -   )

In [49]:

```
#
   = pd.read_csv(input_path.joinpath('13.  _  _   .csv'))


#
for i in range(len(   ['   '])):
      ['   '][i] =    ['   '][i][8:]



#
  ['  '] = ' '
  ['    '] = 0
for i in range(len(   ['    ( )'])):
    if    ['    ( )'][i]/3.3 < 10:
          ['  '][i] = '10    '
          ['    '][i] = 1
    elif    ['    ( )'][i]/3.3 < 20:
          ['  '][i] = '10  '
          ['    '][i] = 10
    elif    ['    ( )'][i]/3.3 < 30:
          ['  '][i] = '20  '
          ['    '][i] = 20
    elif    ['    ( )'][i]/3.3 < 40:
          ['  '][i] = '30  '
          ['    '][i] = 30
    elif    ['    ( )'][i]/3.3 < 50:
          ['  '][i] = '40  '
          ['    '][i] = 40
    elif    ['    ( )'][i]/3.3 < 60:
          ['  '][i] = '50  '
          ['    '][i] = 50
    elif    ['    ( )'][i]/3.3 < 70:
          ['  '][i] = '60  '
          ['    '][i] = 60
```

```
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  app.launch_new_instance()
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:32: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:34: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/opt/app-root/lib/python3.6/site-packages/ipykernel_launcher.py:35: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[49]:

			( )	( )				/			
0		19 ( )	105.6138	37,725	5	201710	11		NaN	30	30
1		19 ( )	59.6506	22,169	2	201710	13		NaN	10	10
2		19 ( )	59.6506	22,569	17	201710	13		NaN	10	10
3		19 ( )	84.9544	33,572	16	201710	14		NaN	20	20
4		19 ( )	84.9544	32,072	12	201710	14		NaN	20	20
...	...	...	...	...	...	...	...	...	...	...	...
7413		M5	84.4700	41,220	12	202005	18		NaN	20	20
7414		6-4 L1	84.9500	43,954	4	202003	7		NaN	20	20
7415		6-4 L1	59.0000	49,000	12	202007	6		NaN	10	10
7416		6-4 M1	112.8900	84,000	3	202007	9		NaN	30	30
7417		6-4 M1	84.9500	56,672	21	202009	17		NaN	20	20

7418 rows × 11 columns

In [50]:

```
#   -   - / -
  ['  ( )'] =   ['  ( )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_   = round(   .groupby(['  ','  ','/  ','  '])[['  ( )']].mean(),3)
data_
```

Out[50]:

( )

/

19	**10**	24573.667 ( )
/	**20**	33793.462
	30	38117.167
19 ( )	**10**	25477.764
	20	35283.886
	30	40881.593
1 (1)	**30**	42998.462
2	**60**	63900.000
2 (2)	**30**	49333.000
(2 ,1-1,M6)	**30**	37598.000
18	**30**	41137.222
18 ( )	**30**	39489.599
	40	65683.333
1 ( )	**30**	43532.500
1	**20**	48184.500
	30	54319.000
2	**20**	60034.000
	30	71972.000
	20	49344.000
HO1	**20**	47325.000
	30	57820.000
SR	**10**	11608.148
	10	26500.000
2	**10**	12768.100
	10	11000.000
2	**10**	13009.143
	10	12442.000
	10	11222.429
	10	19505.000
	10	13233.333
11 ( )	**20**	36883.915
12 (2-1,)	**10**	33440.600
	20	43755.500
	30	48539.800
1 (2-1,)	**10**	30387.931
	20	38982.308
	30	45357.896
	40	45143.000
	50	40251.000
2 (2-1 ,)	**20**	38196.027
	30	43488.000
	40	50066.000
3 (2-1)	**10**	27757.952
	20	38429.018
	30	41987.200
	20	31730.000
4 (2-1, e)	**10**	25357.000
	20	47526.000
	20	33526.000
5 (2-1,)	**10**	29381.000

	20	40391.467 ( )
/	30	52541.850
	40	53663.000
6 (2-1)	10	28243.191
	20	41588.951
	30	50459.628
	40	48800.000
8 (2-1,)	10	32879.114
10 (2-1,)	20	44399.931
	30	49265.133
11 ( )	20	42412.000
1	20	42940.000
	30	40201.000
	30	50000.000
2	20	43407.125
3	10	27002.000
	20	40738.500
4	10	31387.000
5 ( )	30	65413.000
6	10	21400.000
	20	46786.667
	30	56080.000
8 ( )	10	39952.000
e (3-1M4)	30	46451.000
	40	67046.000
(3-1 M2)	10	14153.000
	10	27805.000
	20	35628.590
(3-1,L3)	10	26187.080
	20	36635.288
S (3-1 M6)	20	40372.177
	30	47278.500
	50	45000.000
	20	39625.271
	30	76266.667
1	20	49362.000
3	10	27649.333
	20	34439.667
4	20	53200.000
	30	52250.000
11 ( )	20	34628.273
(1-4 H1)	20	34963.962
	10	13000.000
4-1 M2	20	41756.831
	30	50005.179
	40	86646.000
(L4)	20	29928.667
	10	27271.000
(4-1 M3)	10	35142.000
	20	42125.500
	30	50000.000
1	20	56661.500
2	10	30112.556
	20	37509.000

		( )
3	20	46494.000
/	30	59191.333
	10	14973.800
	10	13355.000
4-1 L2	10	30798.008
	20	41439.950
	30	37523.000
4-1 M1	20	46899.081
	30	65450.500
	40	80660.000
3-2 M3 1	10	29086.449
	20	38301.725
3-2 H1	20	40566.910
	30	48323.348
	20	37450.000
2 (3-2 C2-4)	10	8000.000
(3-2 L2)	20	38202.105
	30	42800.938
	20	35091.605
	30	42791.600
	30	42550.000
(3-2,L3)	20	39204.000
	30	45343.606
	20	38985.500
	30	44300.000
I	10	8500.000
4 (3-2 M1)	10	33743.524
	20	42240.000
4	20	32880.556
	30	43790.660
7	20	52425.000
	30	59226.667
10 (S)	20	40500.000
	30	47800.000
1 ( )	10	32786.600
	20	41483.500
	10	31450.000
5 ( )	30	46890.000
6 ( )	20	45390.000
	30	48230.000
8 ( )	20	38358.333
	30	44150.000
12 (2-2,)	10	31533.800
	20	41120.143
	30	65613.000
1 ( )	20	35475.058
	30	42587.000
	30	47890.000
2 ( )	10	27940.450
	20	35364.909
4 ( )	20	35572.143
	30	40188.611

5 ( )	**10**	28148.260 ( )
/	**20**	36582.043
	30	38615.625
6 ( )	**20**	33098.333
	30	44920.286
9 (2-2)	**20**	44571.000
	30	51507.231
10 ( )	**10**	34941.000
	20	48251.000
12 ( )	**10**	31292.889
	20	35730.500
13 (,H1)	**20**	38155.772
14 ( )	**30**	55500.000
1 ( )	**20**	38917.000
	30	46006.154
2 ( )	**10**	30343.125
	20	39110.000
3 (&)	**20**	36492.500
	30	49430.000
3 ( )	**20**	35844.103
	30	41820.667
4 (&)	**20**	35781.375
	30	41601.429
5 ( )	**10**	29747.500
	20	40000.000
	30	41590.000
6 ( )	**30**	48650.000
9 ( )	**30**	46944.500
(M3)	**10**	32918.781
	20	46403.385
	30	52969.000
(M4) 11	**20**	52044.857
	30	55909.667
(M5) 14	**20**	50262.667
	30	56209.833
1	**20**	47133.333
	30	57000.000
5 ( )	**10**	26948.462
	20	34970.000
6 ( )	**10**	29335.455
	20	36767.000
	20	40000.000
S	**20**	40286.161
	30	50711.400
	10	11500.000
C3-3 L2	**10**	11140.000
	10	25336.774
	20	34449.500
3-3 M6	**10**	26067.909
	20	32103.742
H3	**20**	31050.000
S- 1-5 H9	**20**	47224.000
	40	87055.000

	2		**20**	46195.000
LH 1-3	**M1**	/	**20**	27200.000
	9		**30**	49200.000
	L1		**20**	41490.000
	M1		**20**	37215.000
	M5		**20**	41220.000
6-4	**L1**		**10**	35911.333
			20	43954.000
6-4	**M1**		**20**	46892.000
			30	64929.500

In [ ]:

In [51]:

```
#  ['   (  )'] =   ['   (  )'].apply(lambda x: float(x.split()[0].replace(',','')))
data_    = round(   .groupby(['   ','   ','/  ','   '])[['   (  )','   ']].mean(),3)
data_
```

Out[51]:

			( )	
/				
19	**10**		24573.667	10
	20		33793.462	20
	30		38117.167	30
19 ( )	**10**		25477.764	10
	20		35283.886	20
	30		40881.593	30
1 (1)	**30**		42998.462	30
2	**60**		63900.000	60
2 (2)	**30**		49333.000	30
(2 ,1-1,M6)	**30**		37598.000	30
18	**30**		41137.222	30
18 ( )	**30**		39489.599	30
	40		65683.333	40
1 ( )	**30**		43532.500	30
1	**20**		48184.500	20
	30		54319.000	30
2	**20**		60034.000	20
	30		71972.000	30
	20		49344.000	20
HO1	**20**		47325.000	20
	30		57820.000	30
SR	**10**		11608.148	1
	10		26500.000	10
2	**10**		12768.100	1
	10		11000.000	1
2	**10**		13009.143	1
	10		12442.000	1

			( )	
	/	10	11222.429	1
		10	19505.000	10
		10	13233.333	1
11 ( )		20	36883.915	20
12 (2-1,)		10	33440.600	10
		20	43755.500	20
		30	48539.800	30
1 (2-1,)		10	30387.931	10
		20	38982.308	20
		30	45357.896	30
		40	45143.000	40
		50	40251.000	50
2 (2-1 ,)		20	38196.027	20
		30	43488.000	30
		40	50066.000	40
3 (2-1)		10	27757.952	10
		20	38429.018	20
		30	41987.200	30
		20	31730.000	20
4 (2-1, e)		10	25357.000	10
		20	47526.000	20
		20	33526.000	20
5 (2-1,)		10	29381.000	10
		20	40391.467	20
		30	52541.850	30
		40	53663.000	40
6 (2-1)		10	28243.191	10
		20	41588.951	20
		30	50459.628	30
		40	48800.000	40
8 (2-1,)		10	32879.114	10
10 (2-1,)		20	44399.931	20
		30	49265.133	30
11 ( )		20	42412.000	20
1		20	42940.000	20
		30	40201.000	30
		30	50000.000	30
2		20	43407.125	20
3		10	27002.000	10
		20	40738.500	20
4		10	31387.000	10
5 ( )		30	65413.000	30
6		10	21400.000	10
		20	46786.667	20
		30	56080.000	30
8 ( )		10	39952.000	10
e (3-1M4)		30	46451.000	30
		40	67046.000	40
(3-1 M2)		10	14153.000	1
		10	27805.000	10
		20	35628.590	20
(3-1,L3)		10	26187.080	10

		( )	
	20	36635.288	20
S (3-1 M6) /	20	40372.177	20
	30	47278.500	30
	50	45000.000	50
	20	39625.271	20
	30	76266.667	30
1	20	49362.000	20
3	10	27649.333	10
	20	34439.667	20
4	20	53200.000	20
	30	52250.000	30
11 ( )	20	34628.273	20
(1-4 H1)	20	34963.962	20
	10	13000.000	10
4-1 M2	20	41756.831	20
	30	50005.179	30
	40	86646.000	40
(L4)	20	29928.667	20
	10	27271.000	10
(4-1 M3)	10	35142.000	10
	20	42125.500	20
	30	50000.000	30
1	20	56661.500	20
2	10	30112.556	10
	20	37509.000	20
3	20	46494.000	20
	30	59191.333	30
	10	14973.800	1
	10	13355.000	10
4-1 L2	10	30798.008	10
	20	41439.950	20
	30	37523.000	30
4-1 M1	20	46899.081	20
	30	65450.500	30
	40	80660.000	40
3-2 M3 1	10	29086.449	10
	20	38301.725	20
3-2 H1	20	40566.910	20
	30	48323.348	30
	20	37450.000	20
2 (3-2 C2-4)	10	8000.000	1
(3-2 L2)	20	38202.105	20
	30	42800.938	30
	20	35091.605	20
	30	42791.600	30
	30	42550.000	30
(3-2,L3)	20	39204.000	20
	30	45343.606	30
	20	38985.500	20
	30	44300.000	30
I	10	8500.000	1
4 (3-2 M1)	10	33743.524	10

	20	42240.000 ( )	20
4 /	**20**	32880.556	20
	30	43790.660	30
7	**20**	52425.000	20
	30	59226.667	30
10 (S)	**20**	40500.000	20
	30	47800.000	30
1 ( )	**10**	32786.600	10
	20	41483.500	20
	10	31450.000	10
5 ( )	**30**	46890.000	30
6 ( )	**20**	45390.000	20
	30	48230.000	30
8 ( )	**20**	38358.333	20
	30	44150.000	30
12 (2-2,)	**10**	31533.800	10
	20	41120.143	20
	30	65613.000	30
1 ( )	**20**	35475.058	20
	30	42587.000	30
	30	47890.000	30
2 ( )	**10**	27940.450	10
	20	35364.909	20
4 ( )	**20**	35572.143	20
	30	40188.611	30
5 ( )	**10**	28148.260	10
	20	36582.043	20
	30	38615.625	30
6 ( )	**20**	33098.333	20
	30	44920.286	30
9 (2-2)	**20**	44571.000	20
	30	51507.231	30
10 ( )	**10**	34941.000	10
	20	48251.000	20
12 ( )	**10**	31292.889	10
	20	35730.500	20
13 (,H1)	**20**	38155.772	20
14 ( )	**30**	55500.000	30
1 ( )	**20**	38917.000	20
	30	46006.154	30
2 ( )	**10**	30343.125	10
	20	39110.000	20
3 (&)	**20**	36492.500	20
	30	49430.000	30
3 ( )	**20**	35844.103	20
	30	41820.667	30
4 (&)	**20**	35781.375	20
	30	41601.429	30
5 ( )	**10**	29747.500	10
	20	40000.000	20
	30	41590.000	30
6 ( )	**30**	48650.000	30
9 ( )	**30**	46944.500	30
(M3)	**10**	32918.781	10

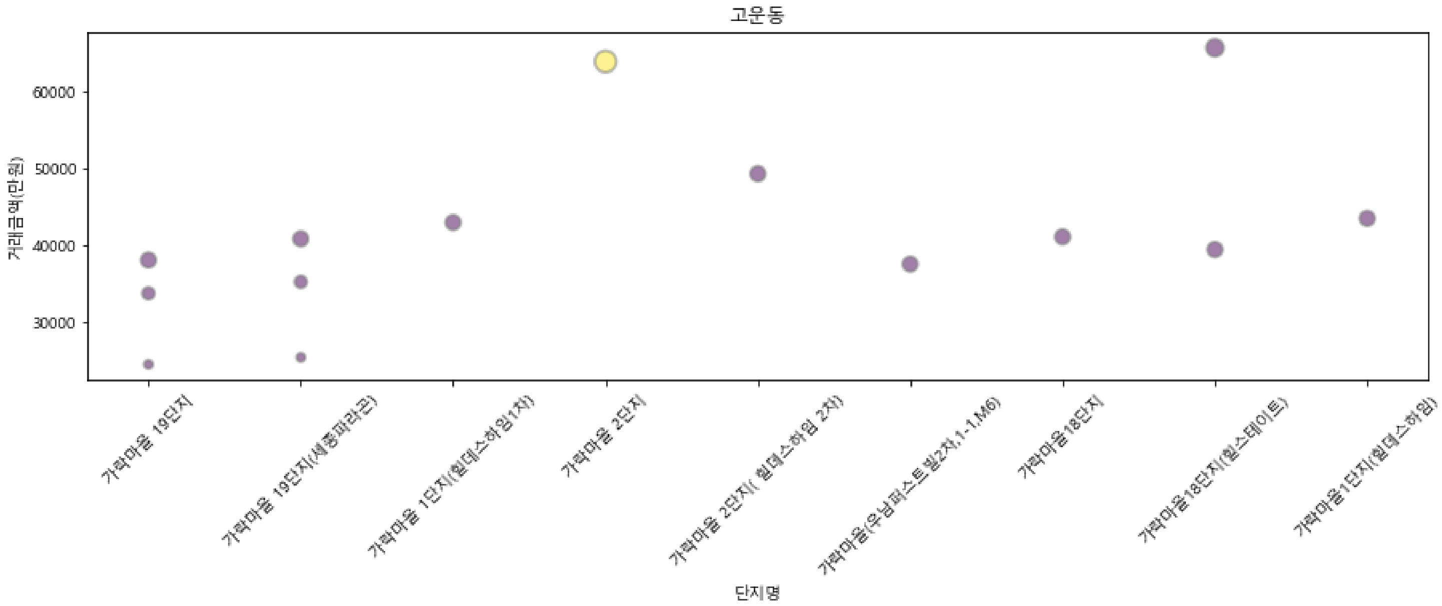
	20	46403.385	20
/	**30**	52969.000	30
(M4) 11	**20**	52044.857	20
	30	55909.667	30
(M5) 14	**20**	50262.667	20
	30	56209.833	30
1	**20**	47133.333	20
	30	57000.000	30
5 ( )	**10**	26948.462	10
	20	34970.000	20
6 ( )	**10**	29335.455	10
	20	36767.000	20
	20	40000.000	20
S	**20**	40286.161	20
	30	50711.400	30
	10	11500.000	1
C3-3 L2	**10**	11140.000	1
	10	25336.774	10
	20	34449.500	20
3-3 M6	**10**	26067.909	10
	20	32103.742	20
H3	**20**	31050.000	20
S- 1-5 H9	**20**	47224.000	20
	40	87055.000	40
2	**20**	46195.000	20
LH 1-3 M1	**20**	27200.000	20
9	**30**	49200.000	30
L1	**20**	41490.000	20
M1	**20**	37215.000	20
M5	**20**	41220.000	20
6-4 L1	**10**	35911.333	10
	20	43954.000	20
6-4 M1	**20**	46892.000	20
	30	64929.500	30

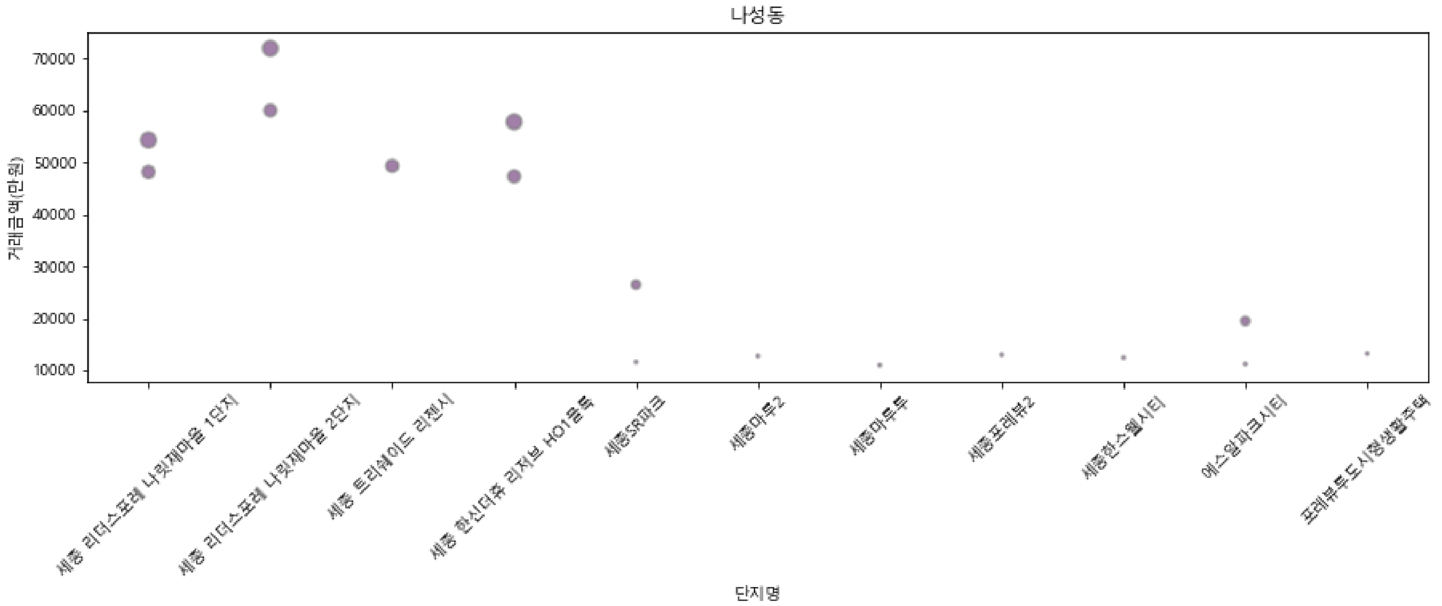
In [53]:

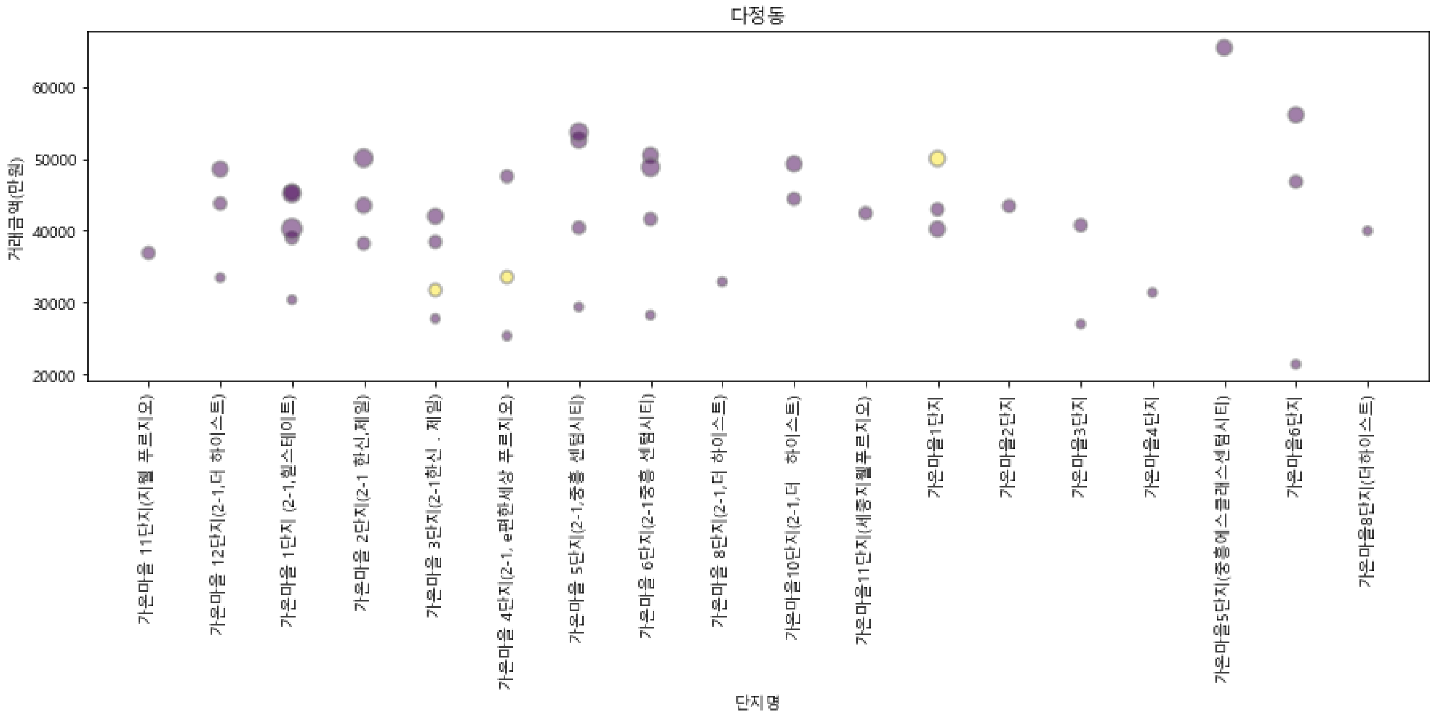
```
#
data_    = data_   .reset_index()
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
data_  _    = data_   [data_   ['   '] == '   ']
```

In [54]:

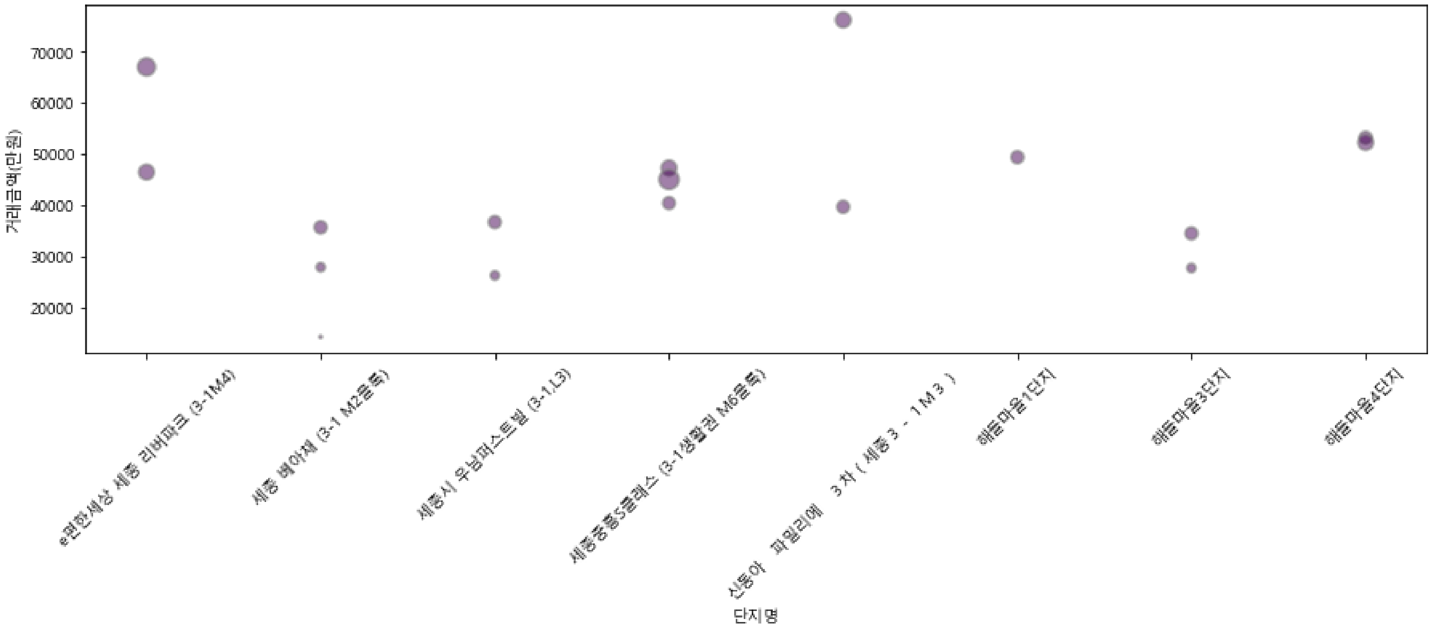
```
#
import matplotlib
matplotlib.font_manager._rebuild()
plt.rc("font", family="Malgun Gothic")

# ' '   , ' '   ,
_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 45)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')


_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 45)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 90)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 45)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 90)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 90)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 90)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.xticks(rotation = 45)
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')

_scatter = plt.figure(figsize=(15,4))
_scatter = plt.scatter(data_  _  ['   '],data_  _  ['    (  )'], c = data_  _  ['/   ']==' ', edgecolors='grey', linewidth=2, s=data_  _  ['    ']*3, alpha = 0.5)
_scatter = plt.xlabel('   ')
_scatter = plt.ylabel('    (  )')
_scatter = plt.title('   ')
```



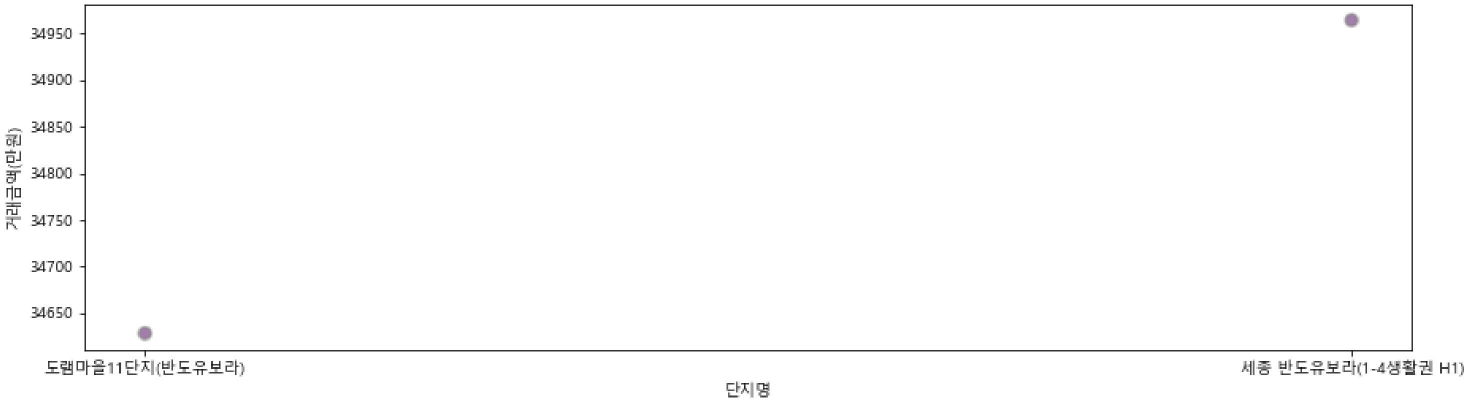




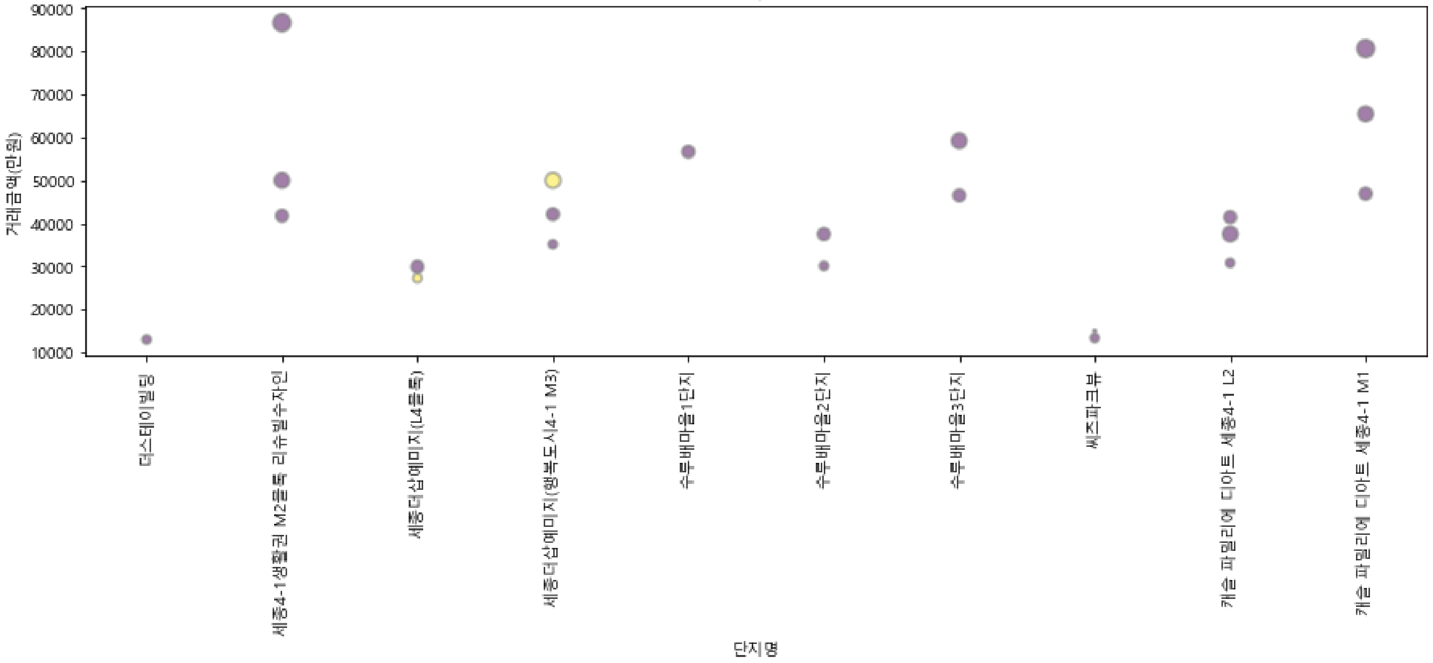
대평동


거래금액(만원)
70000
60000
50000
40000
30000
20000
e편한세상 세종 리버파크 (3-1M4)
세종 벽아체 (3-1 M2블록)
세종시 우남퍼스트빌 (3-1,L3)
세종중흥S클래스 (3-1생활권 M6블록)
신동아 파밀리에 3 차 (세종 3 - 1 M 3)
해들마을1단지
해들마을3단지
해들마을4단지
단지명

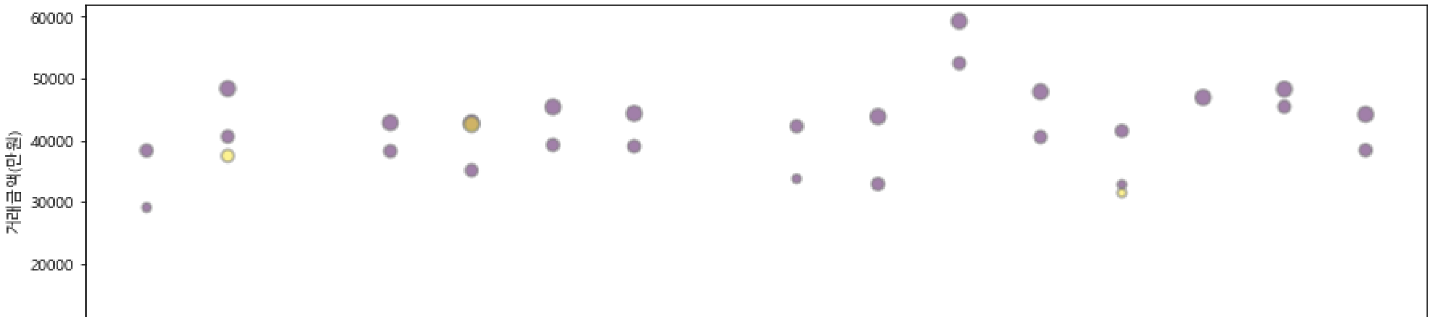
도담동

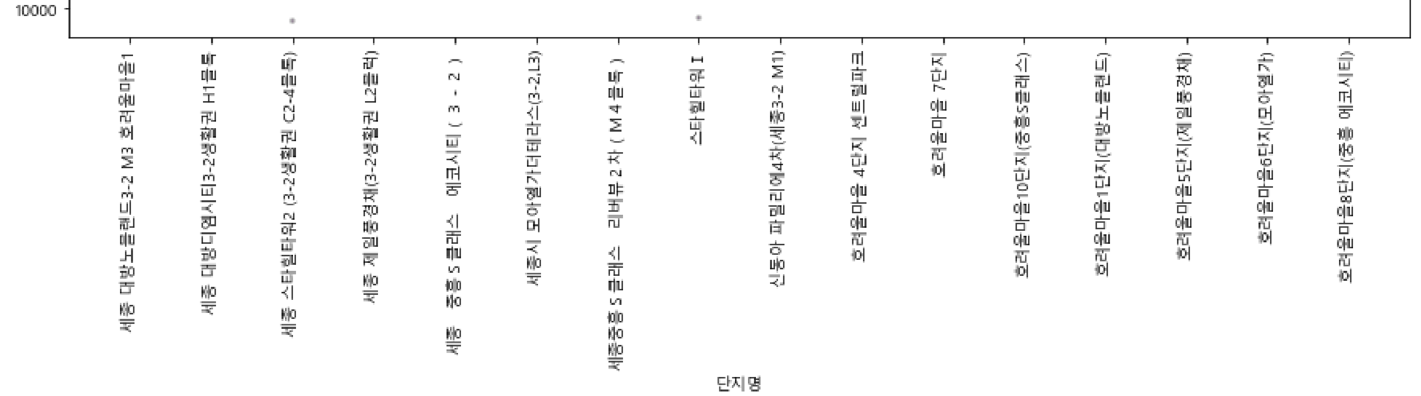

거래금액(만원)
34950
34900
34850
34800
34750
34700
34650
도램마을11단지(반도유보라)
세종 반도유보라(1-4생활권 H1)
단지명

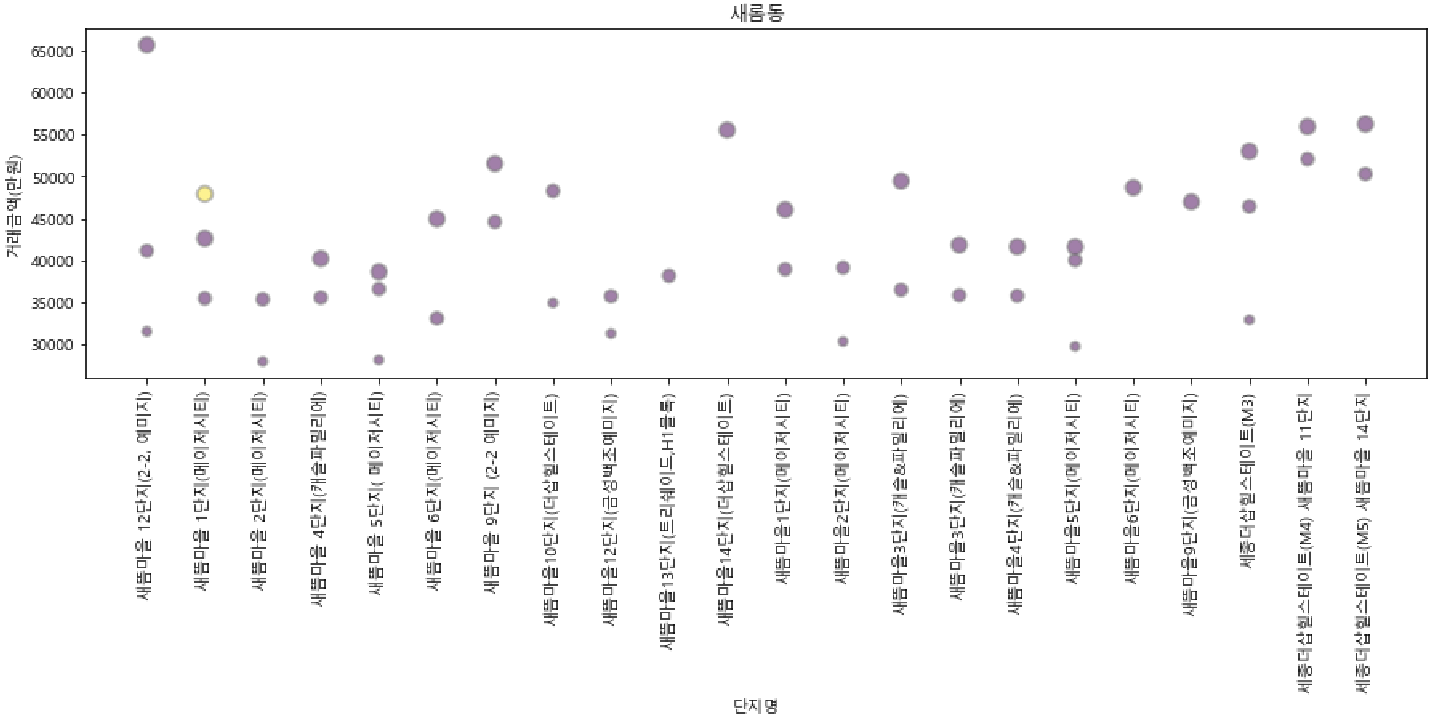
반곡동

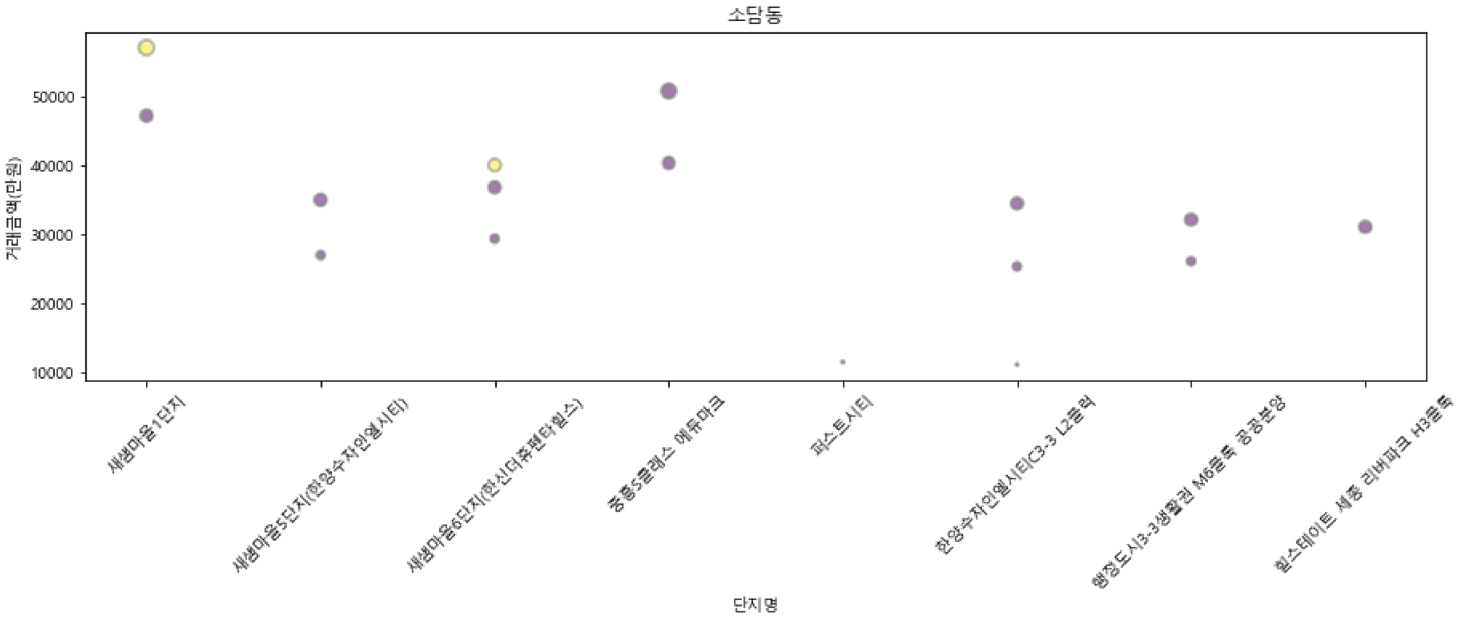

거래금액(만원)
90000
80000
70000
60000
50000
40000
30000
20000
10000
더스테이빌딩
세종4-1생활권 M2블록 리슈빌수자인
세종더샵예미지(L4블록)
세종더샵예미지(행복도시4-1 M3)
수루배마을1단지
수루배마을2단지
수루배마을3단지
세즈파크뷰
캐슬 파밀리에 디아트 세종4-1 L2
캐슬 파밀리에 디아트 세종4-1 M1
단지명

보람동

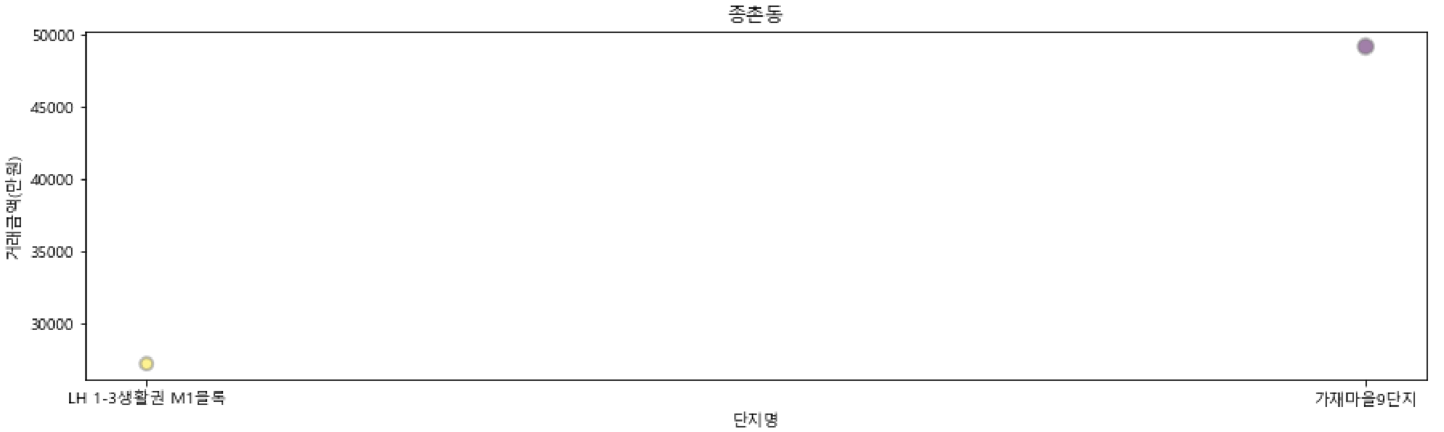

거래금액(만원)
60000
50000
40000
30000
20000

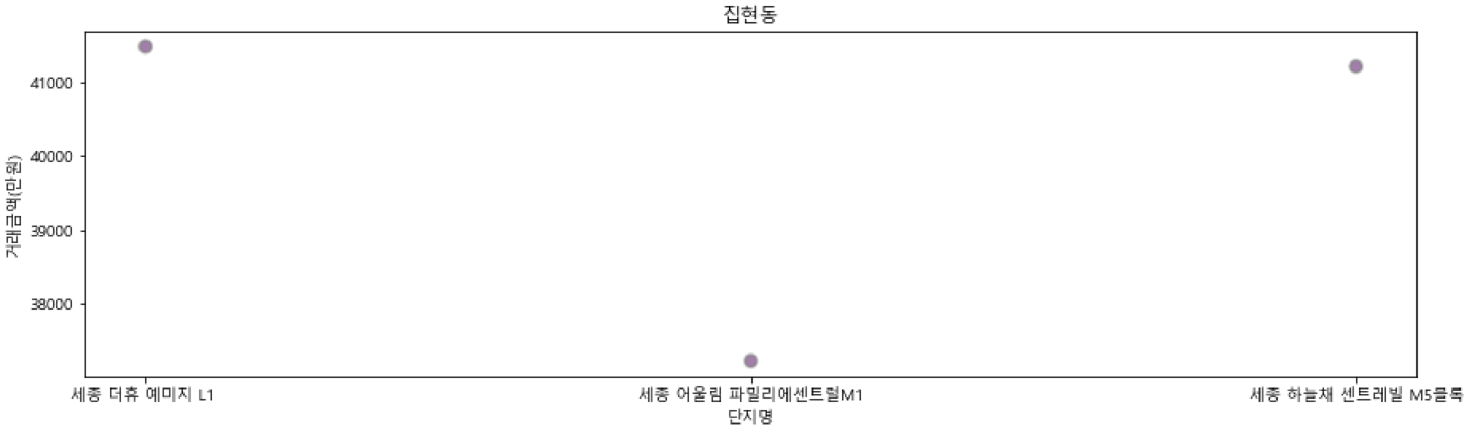

10000
세종 대방노블랜드3-2 M3 호려울마을1
세종 대방디엠시티3-2생활권 H1블록
세종 스타힐타워2 (3-2생활권 C2-4블록)
세종 제일풍경채(3-2생활권 L2블럭)
세종 중흥 S 클래스 에코시티 (3 - 2)
세종시 모아엘가더테라스(3-2,L3)
세종중흥 S 클래스 리버뷰 2 차 (M 4 블록)
스타힐타워 I
신동아 파밀리에4차(세종3-2 M1)
호려울마을 4단지 센트럴파크
호려울마을 7단지
호려울마을10단지(중흥S클래스)
호려울마을1단지(대방노블랜드)
호려울마을5단지(제일풍경채)
호려울마을6단지(모아엘가)
호려울마을8단지(중흥 에코시티)
단지명

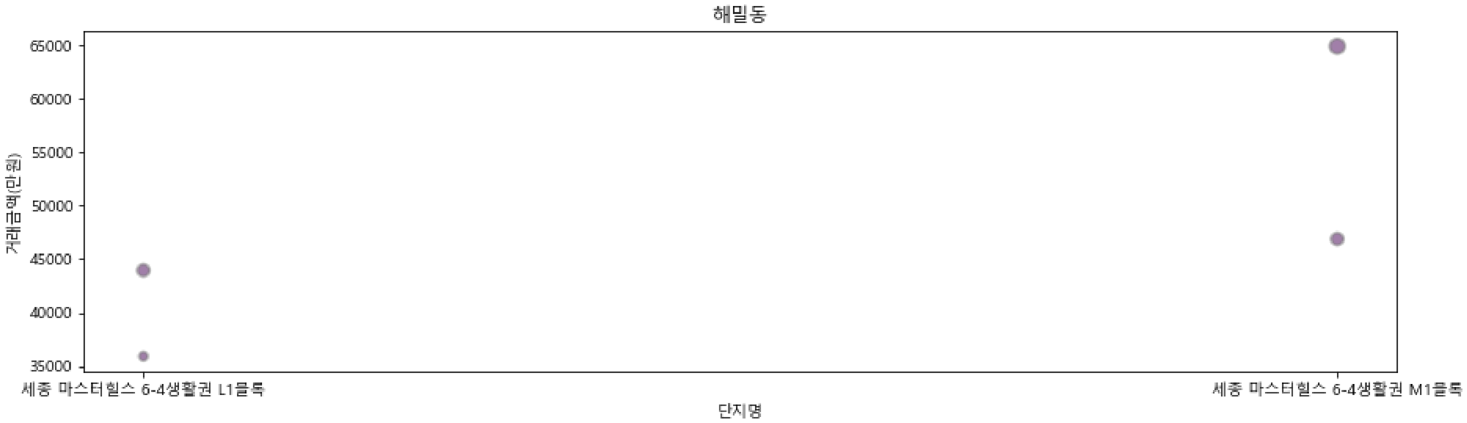

새롬동
65000
60000
55000
50000
45000
40000
35000
30000
거래금액(만원)
새뜸마을 12단지(2-2, 예미지)
새뜸마을 1단지(메이저시티)
새뜸마을 2단지(메이저시티)
새뜸마을 4단지(캐슬파밀리에)
새뜸마을 5단지(메이저시티)
새뜸마을 6단지(메이저시티)
새뜸마을 9단지 (2-2 예미지)
새뜸마을10단지(더샵힐스테이트)
새뜸마을12단지(금성백조예미지)
새뜸마을13단지(트리쉐이드,H1블록)
새뜸마을14단지(더샵힐스테이트)
새뜸마을1단지(메이저시티)
새뜸마을2단지(메이저시티)
새뜸마을3단지(캐슬&파밀리에)
새뜸마을3단지(캐슬파밀리에)
새뜸마을4단지(캐슬&파밀리에)
새뜸마을5단지(메이저시티)
새뜸마을6단지(메이저시티)
새뜸마을9단지(금성백조예미지)
세종더샵힐스테이트(M3)
세종더샵힐스테이트(M4) 새뜸마을 11단지
세종더샵힐스테이트(M5) 새뜸마을 14단지
단지명


소담동
50000
40000
30000
20000
10000
거래금액(만원)
새샘마을1단지
새샘마을5단지(한양수자인엘시티)
새샘마을6단지(한신더휴펜타힐스)
중흥S클래스 에듀마크
퍼스트시티
한양수자인엘시티C3-3 L2블럭
행정도시3-3생활권 M6블록 공공분양
힐스테이트 세종 리버파크 H3블록
단지명

어진동
85000
80000
75000
70000
65000
60000
55000
50000
거래금액(만원)







In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [55]:

In [55]:

```
#
  _  = pd.read_csv(input_path.joinpath('22.  _  _   .csv'))

#
for i in range(  _ .shape[1]):
    if(   _ .columns[i] != '  '):
        _data =   _ .columns[i][7:]
         _  =   _ .rename({  _ .columns[i]: _data }, axis = 'columns')
  _
```

Out[55]:

	(_ _)	( )	(0~9)	(10~19)	(20~29)	(30~39)	(40~49)	(50~59)	(60~69)	...	(10~19)	(20~29)	(30~39)	(40~49)	(50~59)	(60~69)	(70~79)	(80~89)	(9
0	121937	121937	17864	13917	13463	22372	23665	15288	8922	...	21252	18385	31411	33519	21592	15113	7736	4672	
1	23529	23529	2525	2473	3461	3783	4126	3393	2131	...	2050	2811	2436	3186	3110	2676	1725	930	
2	1463	1463	56	93	220	176	263	296	203	...	54	156	104	109	210	221	138	102	
3	1828	1828	87	123	210	228	247	368	318	...	63	112	116	123	245	307	224	212	
4	3554	3554	235	287	472	505	588	650	396	...	170	255	274	314	457	511	353	251	
5	5033	5033	263	287	616	620	767	1013	814	...	206	369	354	387	715	836	606	437	
6	3427	3427	132	199	540	510	574	648	451	...	144	371	272	294	556	514	302	244	
7	4182	4182	222	328	532	526	616	770	637	...	200	312	268	355	524	657	517	338	
8	3312	3312	194	285	403	392	520	642	431	...	172	230	197	273	454	453	377	300	
9	2124	2124	67	97	245	225	289	488	385	...	56	119	92	125	287	374	259	208	
10	1464	1464	89	126	190	170	273	264	184	...	96	110	71	133	190	190	150	130	
11	17376	17376	3185	2466	1760	3364	3790	1833	662	...	1662	838	1412	2327	1221	607	271	136	
12	0	0	0	0	0	0	0	0	0	...	1997	2002	3141	3036	1582	839	281	139	
13	15108	15108	2633	1930	1569	3111	3218	1573	666	...	2630	2438	3977	4249	2183	1190	550	322	
14	24906	24906	5273	3370	1963	5463	5369	2016	1042	...	2075	874	1907	3091	1225	613	237	107	
15	14631	14631	2903	1853	1282	3299	3025	1334	602	...	2119	1411	2886	3138	1597	1049	432	225	
16	0	0	0	0	0	0	0	0	0	...	2337	1352	3562	3714	1796	1240	429	145	
17	0	0	0	0	0	0	0	0	0	...	1755	1611	3967	3050	1845	901	308	143	
18	0	0	0	0	0	0	0	0	0	...	1238	981	2001	1930	1128	672	220	109	

```
19    ( _  0      0    (0~0    0       0       0       0       0       0    ...   527      648      1150     948      840      418      114      80     (9
           )   (    )     9  (10~19  (20~29  (30~39  (40~49  (50~59  (60~69  ...  (10~19   (20~29   (30~39   (40~49   (50~59   (60~69   (70~79   (80~89
                          )      )       )       )       )       )       )            )        )        )        )        )        )        )        )
20         0      0      0       0       0       0       0       0       0    ...   1701     1385     3224     2737     1427      845      243      114

21 rows × 1249 columns
```

In [56]:

```
data_  _  =   _ .transpose().reset_index()[1:]
data_  _  = data_  _ .rename(columns={'index':' _  ',0:'      ', 1:'   ', 2:'  ',3:'  ', 4:'  ',5:'  ',6:'  ',7:'  ',8:'  ',9:'  ', 10:'  ', 11:'  ',12:'  ', 13:'  ',14:'
 ',15:'  ',16:'  ',17:'  ',18:'  ',19:'  ',20:'  '})
data_  _
```

Out[56]:

```
              _                                                                     ...

   1    ( _  _ )  121937  23529  1463  1828  3554  5033  3427  4182  3312  ...  17376    0  15108  24906  14631     0     0     0     0     0
   2    (      )  121937  23529  1463  1828  3554  5033  3427  4182  3312  ...  17376    0  15108  24906  14631     0     0     0     0     0
   3     (0~9 )    17864   2525    56    87   235   263   132   222   194  ...   3185    0   2633   5273   2903     0     0     0     0     0
   4   (10~19 )    13917   2473    93   123   287   287   199   328   285  ...   2466    0   1930   3370   1853     0     0     0     0     0
   5   (20~29 )    13463   3461   220   210   472   616   540   532   403  ...   1760    0   1569   1963   1282     0     0     0     0     0
 ...       ...      ...    ...   ...   ...   ...   ...   ...   ...   ...  ...    ...  ...    ...    ...    ...   ...   ...   ...   ...   ...
1244   (60~69 )    15113   2676   221   307   511   836   514   657   453  ...    607  839   1190    613   1049  1240   901   672   418   845
1245   (70~79 )     7736   1725   138   224   353   606   302   517   377  ...    271  281    550    237    432   429   308   220   114   243
1246   (80~89 )     4672    930   102   212   251   437   244   338   300  ...    136  139    322    107    225   145   143   109    80   114
1247   (90~99 )      915    162    23    45    32    91    38    54    60  ...     26   40     69     27     30    37    42    28    24    28
1248   (100 _ )       22      7     1     0     1     2     1     2     1  ...      2    0      0      0      0     0     3     0     0     0

1248 rows × 22 columns
```

In [57]:

```
data_  _ ['  '] = '0'
data_  _ ['  '] = '  '
data_  _ [' '] = '2017'
for i in range(1,len(data_  _ ['_  '])+1):
    data_  _ ['  '][i] = data_  _ ['_  '][i][-7:-2]
    data_  _ [' '][i] = data_  _ ['_  '][i][:4]
    data_  _ ['  '][i] = data_  _ ['_  '][i][7:9]

data_  _
```

Out[57]:

```
              _                                                                     ...

   1    ( _  _ )  121937  23529  1463  1828  3554  5033  3427  4182  3312  ...  24906  14631     0     0     0     0     0   _  _   _    (
   2    (      )  121937  23529  1463  1828  3554  5033  3427  4182  3312  ...  24906  14631     0     0     0     0     0              (
   3     (0~9 )    17864   2525    56    87   235   263   132   222   194  ...   5273   2903     0     0     0     0     0  (0~9    )   (0
   4   (10~19 )    13917   2473    93   123   287   287   199   328   285  ...   3370   1853     0     0     0     0     0  10~19   9   (1
   5   (20~29 )    13463   3461   220   210   472   616   540   532   403  ...   1963   1282     0     0     0     0     0  20~29   9   (2
 ...       ...      ...    ...   ...   ...   ...   ...   ...   ...   ...  ...    ...    ...   ...   ...   ...   ...   ...    ...  ...  ...
1244   (60~69 )    15113   2676   221   307   511   836   514   657   453  ...    613   1049  1240   901   672   418   845  60~69   9   (6
1245   (70~79 )     7736   1725   138   224   353   606   302   517   377  ...    237    432   429   308   220   114   243  70~79   9   (7
1246   (80~89 )     4672    930   102   212   251   437   244   338   300  ...    107    225   145   143   109    80   114  80~89   9   (8
1247   (90~99 )      915    162    23    45    32    91    38    54    60  ...     27     30    37    42    28    24    28  90~99   9   (9
1248   (100 _ )       22      7     1     0     1     2     1     2     1  ...      0      0     0     3     0     0     0  00 _   _   (1

1248 rows × 25 columns
```

In [58]:

```
index_  1 = data_  _ [data_  _ ['  '] == '_   _'].index
index_  2 = data_  _ [data_  _ ['  '] == '_   '].index
index_ 2 = data_ _ [data_ _ [' '] ==       '].index
data_ _  = data_ _ .drop(index_ 1)
data_ _  = data_ _ .drop(index_ 2)
data_ _
```

Out[58]:

	_										...										
3	(0~9)	17864	2525	56	87	235	263	132	222	194	...	5273	2903	0	0	0	0	0	(0~9	)	(0
4	(10~19)	13917	2473	93	123	287	287	199	328	285	...	3370	1853	0	0	0	0	0	10~19	9	(1
5	(20~29)	13463	3461	220	210	472	616	540	532	403	...	1963	1282	0	0	0	0	0	20~29	9	(2
6	(30~39)	22372	3783	176	228	505	620	510	526	392	...	5463	3299	0	0	0	0	0	30~39	9	(3
7	(40~49)	23665	4126	263	247	588	767	574	616	520	...	5369	3025	0	0	0	0	0	40~49	9	(4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1244	(60~69)	15113	2676	221	307	511	836	514	657	453	...	613	1049	1240	901	672	418	845	60~69	9	(6
1245	(70~79)	7736	1725	138	224	353	606	302	517	377	...	237	432	429	308	220	114	243	70~79	9	(7
1246	(80~89)	4672	930	102	212	251	437	244	338	300	...	107	225	145	143	109	80	114	80~89	9	(8
1247	(90~99)	915	162	23	45	32	91	38	54	60	...	27	30	37	42	28	24	28	90~99	9	(9
1248	(100 _)	22	7	1	0	1	2	1	2	1	...	0	0	0	3	0	0	0	00 _	_	(1

1056 rows × 25 columns

In [59]:

```
data_ _  = data_ _ .reset_index()
del(data_ _ ['index'])
i = data_ _ .loc[data_ _ [' '] == ' (0~9'][' '].index
j = data_ _ .loc[data_ _ [' '] == '00 _ '][' '].index
data_ _ [' '][i] = '0~9'
data_ _ [' '][j] = '00 _ '
del(data_ _ [' _ '])
data_ _
```

Out[59]:

											...										
0	17864	2525	56	87	235	263	132	222	194	67	...	5273	2903	0	0	0	0	0	0~9	)	(0
1	13917	2473	93	123	287	287	199	328	285	97	...	3370	1853	0	0	0	0	0	10~19	9	(1
2	13463	3461	220	210	472	616	540	532	403	245	...	1963	1282	0	0	0	0	0	20~29	9	(2
3	22372	3783	176	228	505	620	510	526	392	225	...	5463	3299	0	0	0	0	0	30~39	9	(3
4	23665	4126	263	247	588	767	574	616	520	289	...	5369	3025	0	0	0	0	0	40~49	9	(4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1051	15113	2676	221	307	511	836	514	657	453	374	...	613	1049	1240	901	672	418	845	60~69	9	(6
1052	7736	1725	138	224	353	606	302	517	377	259	...	237	432	429	308	220	114	243	70~79	9	(7
1053	4672	930	102	212	251	437	244	338	300	208	...	107	225	145	143	109	80	114	80~89	9	(8
1054	915	162	23	45	32	91	38	54	60	41	...	27	30	37	42	28	24	28	90~99	9	(9
1055	22	7	1	0	1	2	1	2	1	2	...	0	0	0	3	0	0	0	00 _	_	(1

1056 rows × 24 columns

In [60]:

```
data_ _ .groupby([' ',' ',' ']).sum()
```

Out[60]:

													...							
(0	**)**	**0~9**	1068348	98161	2432	2895	9920	9413	5547	9776	6931	2526	...	84654	140764	119449	107492	129029	140166	7105
(1	**9**	**10~19**	873399	111294	3500	4244	10979	11887	8454	13032	12078	3884	...	87026	94979	104514	102776	95367	90385	3634
	_	**00 _**	65	31	0	0	0	2	0	0	15	0	...	1	0	0	0	0	0	6

```
                                                                                                   ...
(2  9  20~29   803633  170854   9651   9238  22897  25818  25763  24541  17666  10019  ...   45950   87742   80562   43552   59842   52828  35863
(3  9  30~39  1320393  154401   9868   8810  21926  26071  25103  23501  16257   9363  ...   76014  181140  143090   91676  141095  146844  91318
(4  9  40~49  1447894  185701  11605  10713  26492  32478  27450  29107  23090  12224  ...  120268  158174  162372  142768  149064  146257  69352
(5  9  50~59   941561  165861  14794  16179  30934  46129  31899  36738  29295  22082  ...   60284   82853   86751   58104   68877   59675  40088
(6  9  60~69   571311  113362  11271  16376  21500  42037  24203  32489  22971  20599  ...   23909   39480   38244   24116   35468   38027  20088
(7  9  70~79   268077   61655   5573   8352  13664  23460  12377  18811  13816  11400  ...    9842   11340   16650    8460   15748   13741   5944
(8  9  80~89    97687   22901   2214   4285   6220   9659   6166   9239   7596   5070  ...    2604    2214    5128    2370    3582    2793    842
(9  9  90~99     9272    2220    151    371    579    948    500    629    524    255  ...     410     367     733     378     371     306    164
(0  )    0~9  1021119   92339   2198   3676   8999   9047   5742  10056   6682   2283  ...   81472  135568  113496   99664  125795  132264  68255
(1  9  10~19   863678  106080   3091   3266   9896  10854   7941  10708  10117   3478  ...   86555  100561  103621   97259   97265   91377  38437
    _   00 _     1287     305     27     37     33    168     75     99     44    125  ...      56      11      22      19      74      53     27
(2  9  20~29   783021  137593   6416   6377  13801  19884  17750  16165  13393   6614  ...   46402  105031   99105   44047   68819   59221  39401
(3  9  30~39  1395810  135713   5318   6472  13392  18702  12660  15800  11304   5098  ...   93442  197704  166400  117728  166160  167244  96306
(4  9  40~49  1335969  160539   5389   7009  17194  20527  14452  18354  16043   7491  ...  120602  158077  162154  143942  142859  141143  67090
(5  9  50~59   876874  153402  10568  12349  23748  36730  25761  29030  23339  15566  ...   55132   86362   82211   51278   71620   67768  41089
(6  9  60~69   585313  116640   9207  14090  22451  37505  22001  29053  21048  17408  ...   26658   42358   43537   25924   42415   44795  20907
(7  9  70~79   333202   79031   6910  11176  17996  29263  14586  24448  19401  12028  ...   12577   13316   22148   10876   18311   14793   6405
(8  9  80~89   200244   40276   5325  10181  10743  19997  10653  14873  13141  10518  ...    7077    7622   13353    5764    9414    5277   3242
(9  9  90~99    33938    6540    809   1781   1228   3482   1353   2496   2571   1560  ...    1089    1623    2189    1428    1464    1248    885

22 rows × 21 columns
```

In [ ]:

In [ ]:

In [ ]: