05 판다스를 활용한 데이터 이해(실전 데이터 탐색)

02 캘리포니아 데이터 확인해 보기

In [1]:

```
import pandas as pd
```

In [2]:

```
print("pandas 버전 ", pd.__version__)
```

```
pandas 버전  0.23.0
```

In [3]:

```
test = pd.read_csv("./california_housing/california_housing_test.csv")
train = pd.read_csv("./california_housing/california_housing_train.csv")
```

In [4]:

```
### 데이터 확인
print("test 데이터 셋 행열 크기 :", test.shape)
print("train 데이터 셋 행열 크기 : ", train.shape)
```

```
test 데이터 셋 행열 크기 : (3000, 9)
train 데이터 셋 행열 크기 :  (17000, 9)
```

In [5]:

```
### 데이터 5행 확인
test.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.
1	-118.30	34.26	43.0	1510.0	310.0	809.0	277.
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	495.
3	-118.36	33.82	28.0	67.0	15.0	49.0	11.
4	-119.67	36.33	19.0	1241.0	244.0	850.0	237.

In [6]:

```
### 데이터 5행 확인
train.head()
```

Out[6]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.

In [7]:

```
### 어떤 컬럼명을 가지고 있을까?
print(test.columns)
print(train.columns)
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [8]:

```
### 데이터는 어떤 자료형을 갖는가?
print(test.dtypes)
print()
print(train.dtypes)
```

```
longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
dtype: object

longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
dtype: object
```

In [9]:

```
### 데이터는 어떤 자료형을 갖는가?
print(test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
longitude             3000 non-null float64
latitude              3000 non-null float64
housing_median_age    3000 non-null float64
total_rooms           3000 non-null float64
total_bedrooms        3000 non-null float64
population            3000 non-null float64
households            3000 non-null float64
median_income         3000 non-null float64
median_house_value    3000 non-null float64
dtypes: float64(9)
memory usage: 211.0 KB
None
```

In [10]:

```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
longitude             17000 non-null float64
latitude              17000 non-null float64
housing_median_age    17000 non-null float64
total_rooms           17000 non-null float64
total_bedrooms        17000 non-null float64
population            17000 non-null float64
households            17000 non-null float64
median_income         17000 non-null float64
median_house_value    17000 non-null float64
dtypes: float64(9)
memory usage: 1.2 MB
None
```

In [11]:

```
### 데이터는 어떤 값들을 갖는가?
train.describe()
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	popu
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.0
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824	1429.5
std	2.005166	2.137340	12.586937	2179.947071	421.499452	1147.8
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.0
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000	790.0
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000	1167.0
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000	1721.0
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000	35682.0

- 1. longitude: A measure of how far west a house is; a higher value is farther west
- 2. latitude: A measure of how far north a house is; a higher value is farther north
- 3. housingMedianAge: Median age of a house within a block; a lower number is a newer building
- 4. totalRooms: Total number of rooms within a block
- 5. totalBedrooms: Total number of bedrooms within a block
- 6. population: Total number of people residing within a block
- 7. households: Total number of households, a group of people residing within a home unit, for a block
- 8. medianIncome: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
- 9. medianHouseValue: Median house value for households within a block (measured in US Dollars)

In [12]:

```
import matplotlib.pyplot as plt
train.hist(bins=50, figsize=(20,15))
plt.show()
```

```
<Figure size 2000x1500 with 9 Axes>
```

In [13]:

```
train.median_house_value.value_counts
```

Out[13]:

```
<bound method IndexOpsMixin.value_counts of 0         66900.0
1         80100.0
2         85700.0
3         73400.0
4         65500.0
5         74000.0
6         82400.0
7         48500.0
8         58400.0
9         48100.0
10        86500.0
11        62000.0
12        48600.0
13        70400.0
14        45000.0
15        69100.0
16        94900.0
17        25000.0
18        44000.0
19        27500.0
20        44400.0
21        59200.0
22        50000.0
23        71300.0
24        53500.0
25       100000.0
26        71100.0
27        80900.0
28        68600.0
29        74300.0
           ...
16970     75500.0
16971     62500.0
16972     70500.0
16973     68300.0
16974     81300.0
16975     82800.0
16976    116100.0
16977     86400.0
16978     70500.0
16979     70200.0
16980     67000.0
16981     72200.0
16982    107000.0
16983     74600.0
16984     70000.0
16985     69000.0
16986     90100.0
16987     68400.0
16988     66900.0
16989     58100.0
16990     78300.0
16991     73200.0
16992     50800.0
16993    106700.0
```

```
16994     76100.0
16995    111400.0
16996     79000.0
16997    103600.0
16998     85800.0
16999     94600.0
Name: median_house_value, Length: 17000, dtype: float64>
```

plot

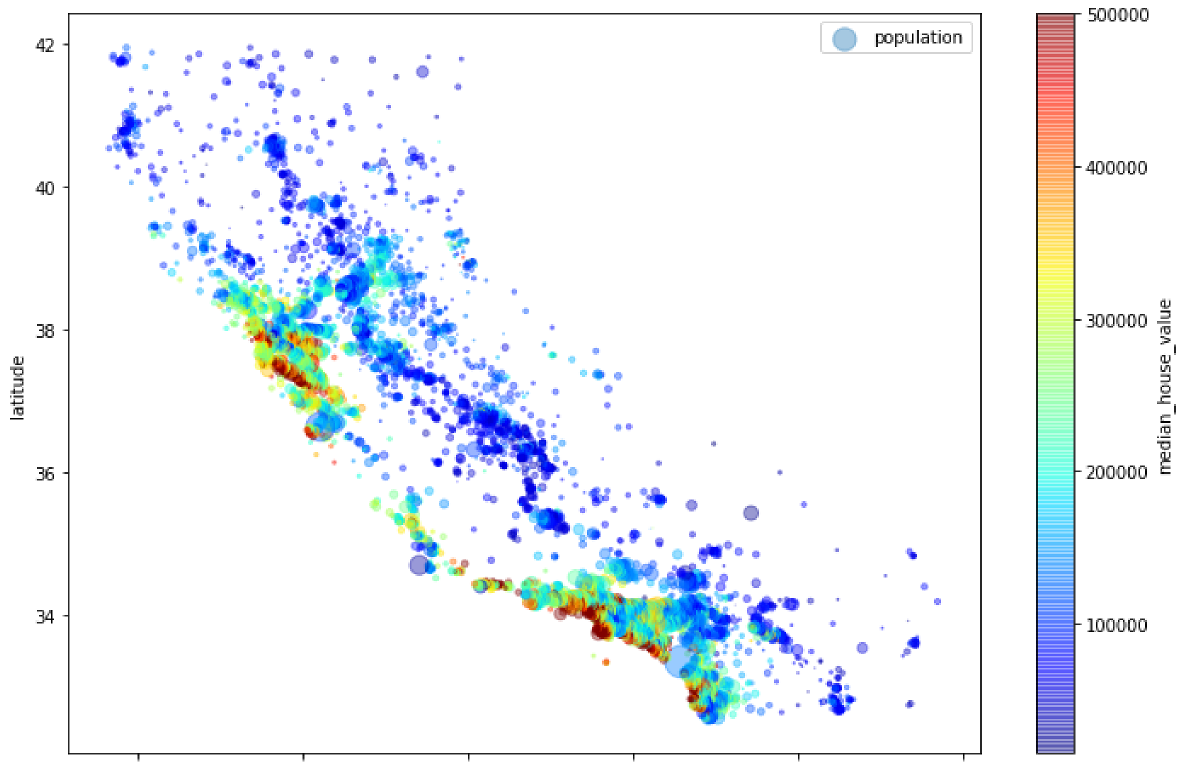
- plot은 matplotlib를 내부에서 임포트하여 사용.
- kind를 통해 여러가지 플롯 작성 가능(bar, pie, hist, kde, box, scatter, area)
  - plot.bar, plot.hist등으로 접근 가능
- cmap : 색지정
- c : 각각의 포인트의 색
- s : 각각의 포인트의 사이즈
- alpha : 투명도

In [16]:

```python
### 위도 경도에 따른 산점도 분포
train.plot(kind="scatter", x="longitude", y="latitude",
              alpha=0.4, s=train["population"]/100,
              label="population", c="median_house_value",
              figsize=(12,8),
              cmap=plt.get_cmap("jet"), colorbar=True)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2502ca5e5f8>
```



In [17]:

```
train.columns
```

Out[17]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [18]:

```
sel = ['total_rooms', 'total_bedrooms', 'population']

temp_train = train[ sel  ]

print("데이터 가공 셋의 크기 : ", temp_train.shape)
print("데이터 가공 셋의 일부 : ", temp_train.head())
```

```
데이터 가공 셋의 크기 :  (17000, 3)
데이터 가공 셋의 일부 :     total_rooms  total_bedrooms  population
0       5612.0          1283.0      1015.0
1       7650.0          1901.0      1129.0
2        720.0           174.0       333.0
3       1501.0           337.0       515.0
4       1454.0           326.0       624.0
```

In [19]:

```
temp_train.describe()
```

Out[19]:

	total_rooms	total_bedrooms	population
count	17000.000000	17000.000000	17000.000000
mean	2643.664412	539.410824	1429.573941
std	2179.947071	421.499452	1147.852959
min	2.000000	1.000000	3.000000
25%	1462.000000	297.000000	790.000000
50%	2127.000000	434.000000	1167.000000
75%	3151.250000	648.250000	1721.000000
max	37937.000000	6445.000000	35682.000000

In [20]:

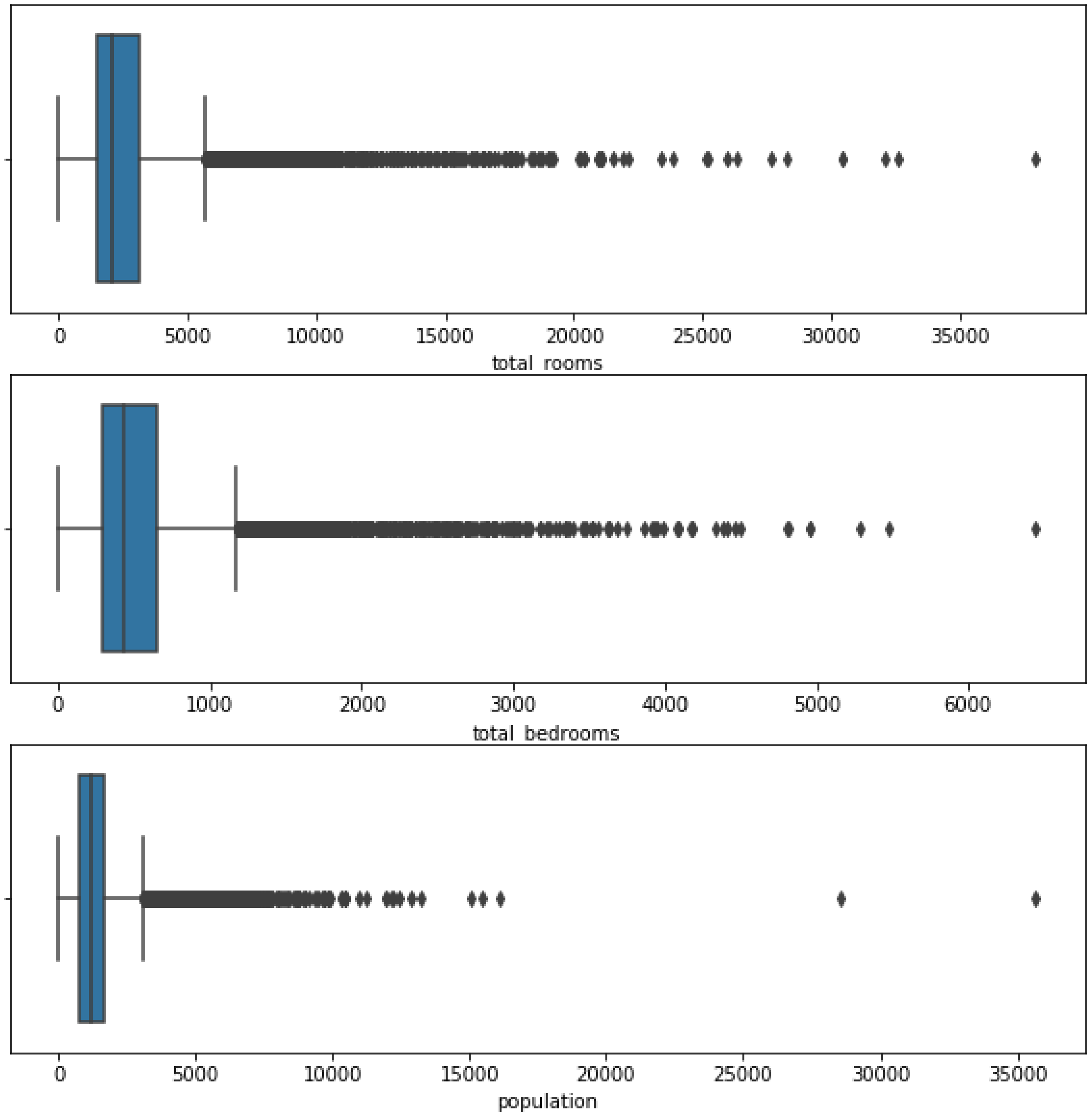
```
import seaborn as sns
```

In [21]:

```
plt.figure(figsize=(10,10))

plt.subplot(3,1,1)
sns.boxplot(x="total_rooms", data=temp_train)
plt.subplot(3,1,2)
sns.boxplot(x="total_bedrooms", data=temp_train)
plt.subplot(3,1,3)
sns.boxplot(x="population", data=temp_train)
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2502efc5be0>
```

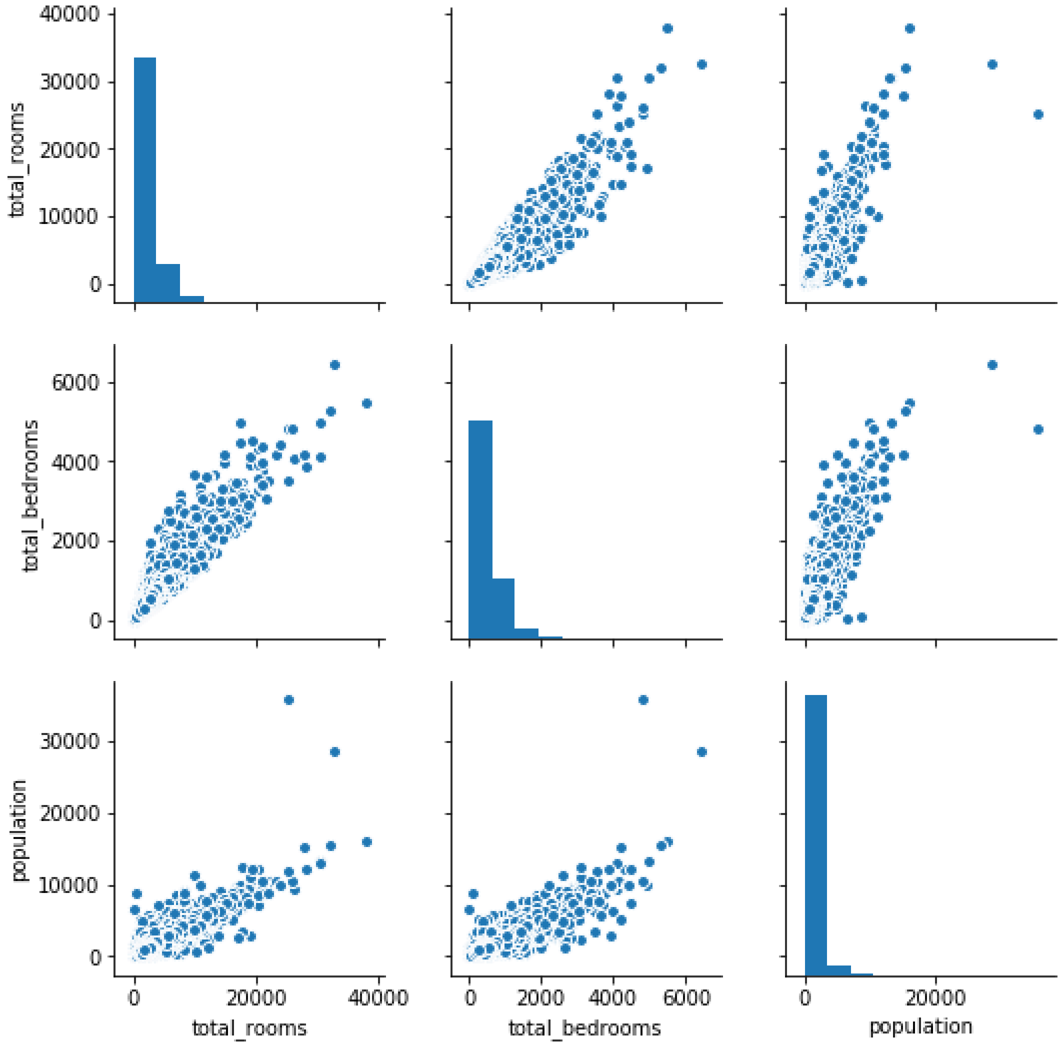


In [22]:

```
sns.pairplot(temp_train)
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x2502ef22978>
```



iloc, Loc 이해하기

In [23]:

```
train.columns
```

Out[23]:

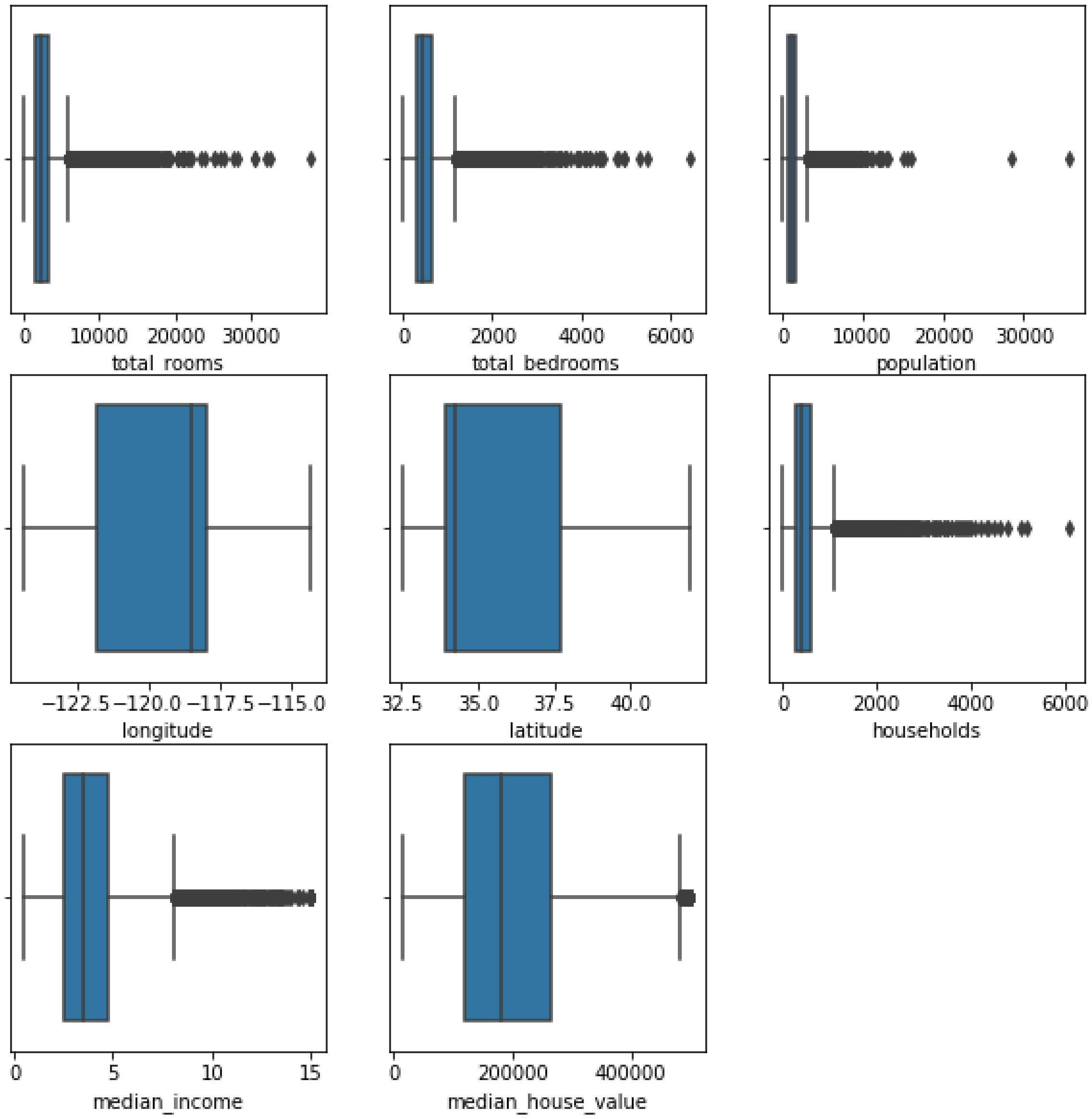
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [24]:

```
plt.figure(figsize=(10,10))

plt.subplot(3,3,1)
sns.boxplot(x="total_rooms", data=train)
plt.subplot(3,3,2)
sns.boxplot(x="total_bedrooms", data=train)
plt.subplot(3,3,3)
sns.boxplot(x="population", data=train)

plt.subplot(3,3,4)
sns.boxplot(x="longitude", data=train)
plt.subplot(3,3,5)
sns.boxplot(x="latitude", data=train)
plt.subplot(3,3,6)
sns.boxplot(x="households", data=train)

plt.subplot(3,3,7)
sns.boxplot(x="median_income", data=train)
plt.subplot(3,3,8)
sns.boxplot(x="median_house_value", data=train)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2502f9606d8>
```

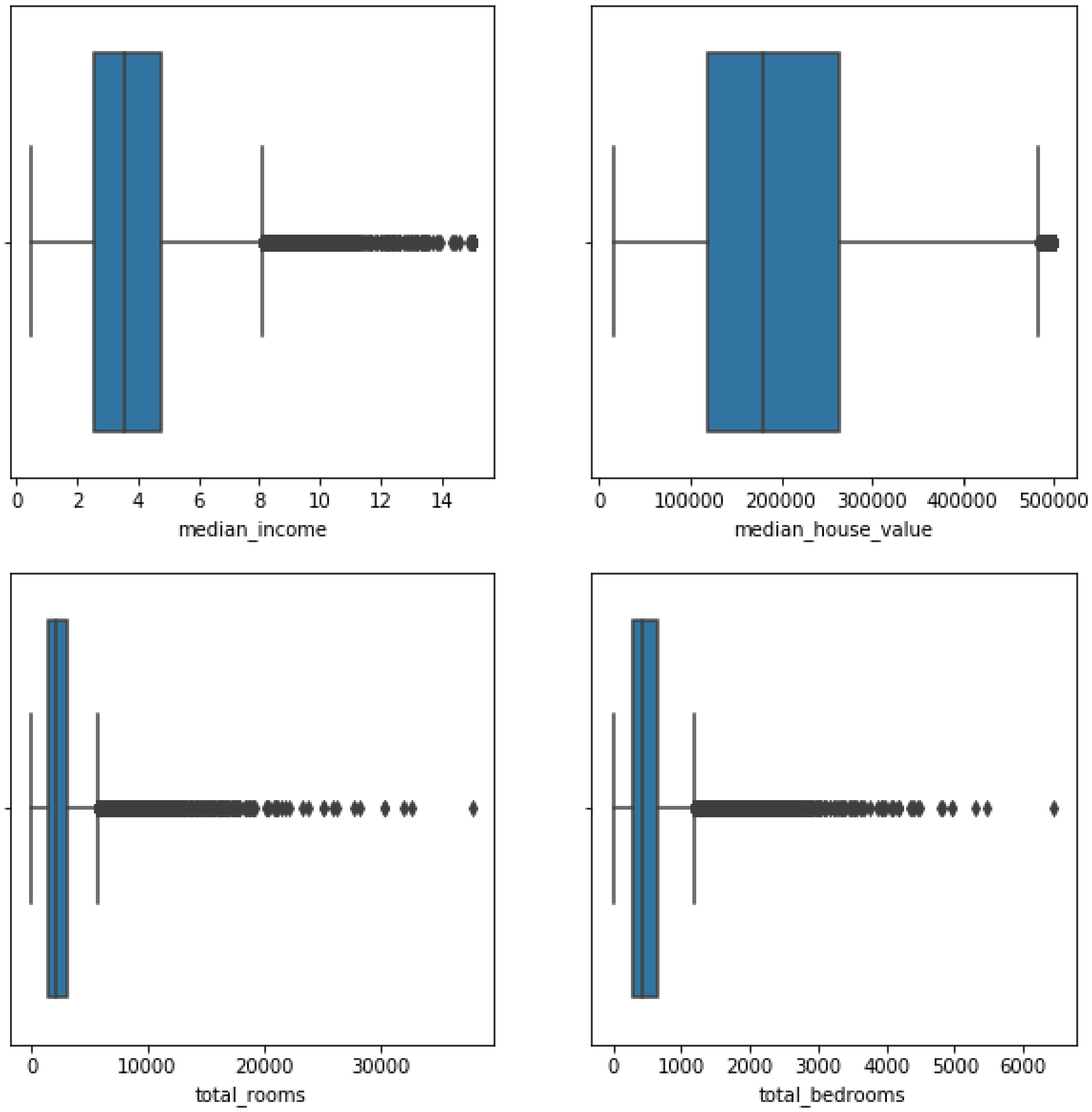

0
10000
20000
30000
total_rooms
0
2000
4000
6000
total_bedrooms
0
10000
20000
30000
population
−122.5
−120.0
−117.5
−115.0
longitude
32.5
35.0
37.5
40.0
latitude
0
2000
4000
6000
households
0
5
10
15
median_income
0
200000
400000
median_house_value

In [25]:

```
plt.figure(figsize=(10,10))

plt.subplot(2,2,1)
sns.boxplot(x="median_income", data=train)
plt.subplot(2,2,2)
sns.boxplot(x="median_house_value", data=train)

plt.subplot(2,2,3)
sns.boxplot(x="total_rooms", data=train)
plt.subplot(2,2,4)
sns.boxplot(x="total_bedrooms", data=train)
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2502fa19dd8>
```



In [26]:

```
## 두 컬럼 선택
temp02 = train.loc[:,  [ "median_income", "median_house_value" ] ]
temp02.head()
```

Out[26]:

	median_income	median_house_value
0	1.4936	66900.0
1	1.8200	80100.0
2	1.6509	85700.0
3	3.1917	73400.0
4	1.9250	65500.0

In [27]:

```
train.columns
```

Out[27]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [28]:

```
## 두 컬럼 선택 8열, 9열
temp03 = train.iloc[:,  [7, 8] ]
print( temp03.head() )

temp03 = train.iloc[:,  [-2, -1] ]
print( temp03.head() )
```

```
   median_income  median_house_value
0         1.4936             66900.0
1         1.8200             80100.0
2         1.6509             85700.0
3         3.1917             73400.0
4         1.9250             65500.0
   median_income  median_house_value
0         1.4936             66900.0
1         1.8200             80100.0
2         1.6509             85700.0
3         3.1917             73400.0
4         1.9250             65500.0
```

In [29]:

```
temp04 = train.iloc[:,  [6, 7, 8] ]
print(temp04.head() )
```

```
   households  median_income  median_house_value
0       472.0         1.4936             66900.0
1       463.0         1.8200             80100.0
2       117.0         1.6509             85700.0
3       226.0         3.1917             73400.0
4       262.0         1.9250             65500.0
```

In [30]:

```
## 그렇다면 일부 열의 부분을 가져올 수 없을까?
## range 와
scope = list(range(6,9,1))   # 6번째부터 8번째까지 범위 지정.
temp = train.iloc[:, scope ] # 6,7,8 열을 가져온다.
print(temp.head() )

temp = train.iloc[:, 6:9:1 ] # 6,7,8 열을 가져온다.
print(temp.head() )
```

```
   households  median_income  median_house_value
0       472.0         1.4936             66900.0
1       463.0         1.8200             80100.0
2       117.0         1.6509             85700.0
3       226.0         3.1917             73400.0
4       262.0         1.9250             65500.0
   households  median_income  median_house_value
0       472.0         1.4936             66900.0
1       463.0         1.8200             80100.0
2       117.0         1.6509             85700.0
3       226.0         3.1917             73400.0
4       262.0         1.9250             65500.0
```

In [31]:

```
train.head()
```

Out[31]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.

In [32]:

```
train.total_rooms.describe()
```

Out[32]:

```
count    17000.000000
mean      2643.664412
std       2179.947071
min          2.000000
25%       1462.000000
50%       2127.000000
75%       3151.250000
max      37937.000000
Name: total_rooms, dtype: float64
```

조건을 이용하여 데이터 그룹을 시켜보자.

In [33]:

```
# 전체 방의 수를 위의 값을 기준으로 네 그룹으로 나눈다.
# A1 : 75~100  3151 ~
# A2 : 50~75   2127 ~ 3151
# A3 : 25~50   1462 ~ 2127
# A4 : 0~25    ~1462

tmp_A1 = train[ train['total_rooms']> 3151]
print(tmp_A1.head())
print(tmp_A1.shape)
```

```
    longitude  latitude  housing_median_age  total_rooms  total_bedrooms  ₩
0     -114.31     34.19                15.0       5612.0          1283.0
1     -114.47     34.40                19.0       7650.0          1901.0
8     -114.59     33.61                34.0       4789.0          1175.0
10    -114.60     33.62                16.0       3741.0           801.0
38    -115.48     32.68                15.0       3414.0           666.0

    population  households  median_income  median_house_value
0       1015.0       472.0         1.4936             66900.0
1       1129.0       463.0         1.8200             80100.0
8       3134.0      1056.0         2.1782             58400.0
10      2434.0       824.0         2.6797             86500.0
38      2097.0       622.0         2.3319             91200.0
(4250, 9)
```

In [34]:

```
import numpy as np
```

In [35]:

```
## 두개의 조건문 np.where를 이용하여 확인
bool_val = np.where( (train['total_rooms']> 1462) & (train['total_rooms'] <= 2127), True, False)
type(bool_val)
print(bool_val)
```

```
[False False False ... False False  True]
```

In [36]:

```
train[ bool_val ]
```

Out[36]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	house
3	-114.57	33.64	14.0	1501.0	337.0	515.0	
9	-114.60	34.83	46.0	1497.0	309.0	787.0	
11	-114.60	33.60	21.0	1988.0	483.0	1182.0	
16	-114.65	33.60	28.0	1678.0	322.0	666.0	
20	-114.68	33.49	20.0	1491.0	360.0	1135.0	
24	-115.22	33.54	18.0	1706.0	397.0	3424.0	
26	-115.37	32.82	30.0	1602.0	322.0	1130.0	
30	-115.38	32.82	38.0	1892.0	394.0	1175.0	
40	-115.49	32.69	17.0	1960.0	389.0	1691.0	
41	-115.49	32.67	29.0	1523.0	440.0	1302.0	
46	-115.51	33.24	32.0	1995.0	523.0	1069.0	
53	-115.52	32.98	32.0	1615.0	382.0	1307.0	
54	-115.52	32.97	24.0	1617.0	366.0	1416.0	
55	-115.52	32.97	10.0	1879.0	387.0	1376.0	
56	-115.52	32.77	18.0	1715.0	337.0	1166.0	
61	-115.53	32.97	35.0	1583.0	340.0	933.0	
63	-115.53	32.73	14.0	1527.0	325.0	1453.0	
65	-115.54	32.99	17.0	1697.0	268.0	911.0	
66	-115.54	32.98	27.0	1513.0	395.0	1121.0	
68	-115.54	32.79	23.0	1712.0	403.0	1370.0	
71	-115.55	32.82	34.0	1540.0	316.0	1013.0	
77	-115.56	32.80	28.0	1672.0	416.0	1335.0	
84	-115.56	32.78	29.0	1568.0	283.0	848.0	
89	-115.57	32.83	31.0	1494.0	289.0	959.0	
92	-115.57	32.78	20.0	1534.0	235.0	871.0	
97	-115.58	32.79	14.0	1687.0	507.0	762.0	
99	-115.59	32.85	20.0	1608.0	274.0	862.0	
103	-115.60	32.87	3.0	1629.0	317.0	1005.0	
107	-115.69	32.79	18.0	1564.0	340.0	1161.0	
110	-115.73	33.35	23.0	1586.0	448.0	338.0	
...	...	...	...	...	...	...	
16894	-124.00	40.22	16.0	2088.0	535.0	816.0	
16895	-124.01	40.89	28.0	1470.0	336.0	811.0	
16901	-124.05	40.59	32.0	1878.0	340.0	937.0	

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	house
16903	-124.06	40.88	12.0	2087.0	424.0	1603.0	
16905	-124.07	40.87	47.0	1765.0	326.0	796.0	
16907	-124.07	40.81	23.0	2103.0	411.0	1019.0	
16910	-124.08	40.94	18.0	1550.0	345.0	941.0	
16916	-124.09	40.88	31.0	1982.0	495.0	1052.0	
16921	-124.10	41.04	26.0	1633.0	380.0	890.0	
16926	-124.10	40.50	30.0	1927.0	393.0	996.0	
16928	-124.11	40.95	19.0	1734.0	365.0	866.0	
16930	-124.11	40.93	17.0	1661.0	329.0	948.0	
16934	-124.13	40.79	32.0	2017.0	359.0	855.0	
16941	-124.14	40.79	38.0	1552.0	290.0	873.0	
16945	-124.14	40.59	22.0	1665.0	405.0	826.0	
16947	-124.14	40.58	25.0	1899.0	357.0	891.0	
16952	-124.15	40.80	47.0	1486.0	335.0	765.0	
16954	-124.15	40.78	36.0	2112.0	374.0	829.0	
16958	-124.16	41.02	23.0	1672.0	385.0	1060.0	
16961	-124.16	40.80	52.0	1703.0	500.0	952.0	
16965	-124.16	40.78	46.0	1975.0	346.0	791.0	
16970	-124.17	40.80	52.0	1606.0	419.0	891.0	
16971	-124.17	40.80	52.0	1557.0	344.0	758.0	
16973	-124.17	40.78	39.0	1606.0	330.0	731.0	
16974	-124.17	40.77	30.0	1895.0	366.0	990.0	
16975	-124.17	40.76	26.0	1776.0	361.0	992.0	
16977	-124.17	40.62	32.0	1595.0	309.0	706.0	
16978	-124.18	40.79	39.0	1836.0	352.0	883.0	
16980	-124.18	40.78	34.0	1592.0	364.0	950.0	
16999	-124.35	40.54	52.0	1820.0	300.0	806.0	

4249 rows × 9 columns

In [37]:

```
bool_val = np.where( (train['total_rooms']> 2127) & (train['total_rooms'] <= 3151), True, False)
print(bool_val.shape)
tmp_A2 = train[ bool_val ]
print(tmp_A2.shape)
```

```
(17000,)
(4247, 9)
```

In [38]:

```
bool_val = np.where( (train['total_rooms']> 1462) & (train['total_rooms'] <= 2127), True, False)
print(bool_val.shape)
tmp_A3 = train[ bool_val ]
print(tmp_A3.shape)
```

```
(17000,)
(4249, 9)
```

In [39]:

```
bool_val = np.where( (train['total_rooms']> 1462) , True, False)
print(bool_val.shape)
tmp_A4 = train[ bool_val ]
print(tmp_A4.shape)
```

```
(17000,)
(12746, 9)
```

In [40]:

```
print(tmp_A1.shape, tmp_A2.shape, tmp_A3.shape, tmp_A4.shape )
```

```
(4250, 9) (4247, 9) (4249, 9) (12746, 9)
```

In [41]:

```
### 새로운 컬럼 room_level 만들기
# 전체 방의 수를 위의 값을 기준으로 네 그룹으로 나눈다.
# A1 : 75~100  3151 ~
# A2 : 50~75   2127 ~ 3151
# A3 : 25~50   1462 ~ 2127
# A4 : 0~25    ~1462

bool_val = np.where( (train['total_rooms']> 1462) & (train['total_rooms'] <= 2127), True, False)
train.loc[bool_val, "room_level"] = 3
train['room_level'].head()
```

Out[41]:

```
0    NaN
1    NaN
2    NaN
3    3.0
4    NaN
Name: room_level, dtype: float64
```

In [42]:

```
bool_val = np.where( (train['total_rooms']> 2127) & (train['total_rooms'] <= 3151), True, False)
train.loc[bool_val, "room_level"] = 2
train['room_level'].head()
```

Out[42]:

```
0    NaN
1    NaN
2    NaN
3    3.0
4    NaN
Name: room_level, dtype: float64
```

In [43]:

```
### 새로운 컬럼 room_level 만들기
bool_val = np.where( (train['total_rooms']> 3151) , True, False)
train.loc[bool_val, "room_level"] = 1

bool_val = np.where( (train['total_rooms'] <= 1462) , True, False)
train.loc[bool_val, "room_level"] = 4
train['room_level'].head()
```

Out[43]:

```
0    1.0
1    1.0
2    4.0
3    3.0
4    4.0
Name: room_level, dtype: float64
```

In [44]:

```
train.columns
```

Out[44]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'room_level'],
      dtype='object')
```

groupby를 활용한 그룹별 평균

In [45]:

```
### room_level의 그룹별 나이대 알아보기
print(train.groupby('room_level')['housing_median_age'].mean())
```

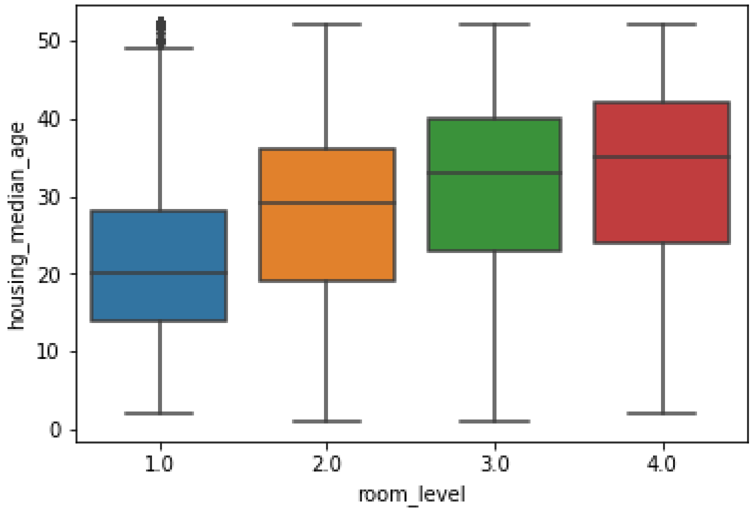
```
room_level
1.0    21.170353
2.0    28.872145
3.0    31.580137
4.0    32.731782
Name: housing_median_age, dtype: float64
```

In [46]:

```
### room_level별 boxplot
### 방이 적으면 적을 수록 나이대가 높다.
### 젊은 층이 많을 수록 지역별 총 방의 수는 많음을 알 수 있다.
sns.boxplot(x="room_level", y="housing_median_age", data=train)
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2502fae7e48>
```



REF

- plot in pandas : https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html (https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html)

In [ ]:

```
```