학습 내용

- 캘리포니아 데이터 살펴보기
- 원하는 데이터를 선택하는 것을 실습을 통해 알아본다.

01 캘리포니아 데이터 가져오기

In [1]:
```
import pandas as pd
print("pandas 버전 ", pd.__version__)
```

```
pandas 버전  2.0.3
```

In [2]:
```
train = pd.read_csv("https://storage.googleapis.com/mledu-datasets/california_housin
test = pd.read_csv("https://storage.googleapis.com/mledu-datasets/california_housing
train.shape, test.shape
```

Out[2]:
```
((17000, 9), (3000, 9))
```

In [3]:
```
### 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 [4]:
```
### 데이터 5행 확인
test.head()
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0	277.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	495.0
3	-118.36	33.82	28.0	67.0	15.0	49.0	11.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0	237.0

In [5]:
```
### 데이터 5행 확인
train.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0

In [6]:
```
### 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 [7]:
```
### 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 [8]:
```
### 데이터는 어떤 자료형을 갖는가?
print(test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           3000 non-null   float64
 1   latitude            3000 non-null   float64
 2   housing_median_age  3000 non-null   float64
 3   total_rooms         3000 non-null   float64
 4   total_bedrooms      3000 non-null   float64
 5   population          3000 non-null   float64
 6   households          3000 non-null   float64
 7   median_income       3000 non-null   float64
 8   median_house_value  3000 non-null   float64
dtypes: float64(9)
memory usage: 211.1 KB
None
```

In [9]:
```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           17000 non-null  float64
 1   latitude            17000 non-null  float64
 2   housing_median_age  17000 non-null  float64
 3   total_rooms         17000 non-null  float64
 4   total_bedrooms      17000 non-null  float64
 5   population          17000 non-null  float64
 6   households          17000 non-null  float64
 7   median_income       17000 non-null  float64
 8   median_house_value  17000 non-null  float64
dtypes: float64(9)
memory usage: 1.2 MB
None
```

In [10]:
```
### 데이터는 어떤 값들을 갖는가?
train.describe()
```

Out[10]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	17000.0000
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824	1429.5739
std	2.005166	2.137340	12.586937	2179.947071	421.499452	1147.8529
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.0000
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000	790.0000
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000	1167.0000
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000	1721.0000
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000	35682.0000

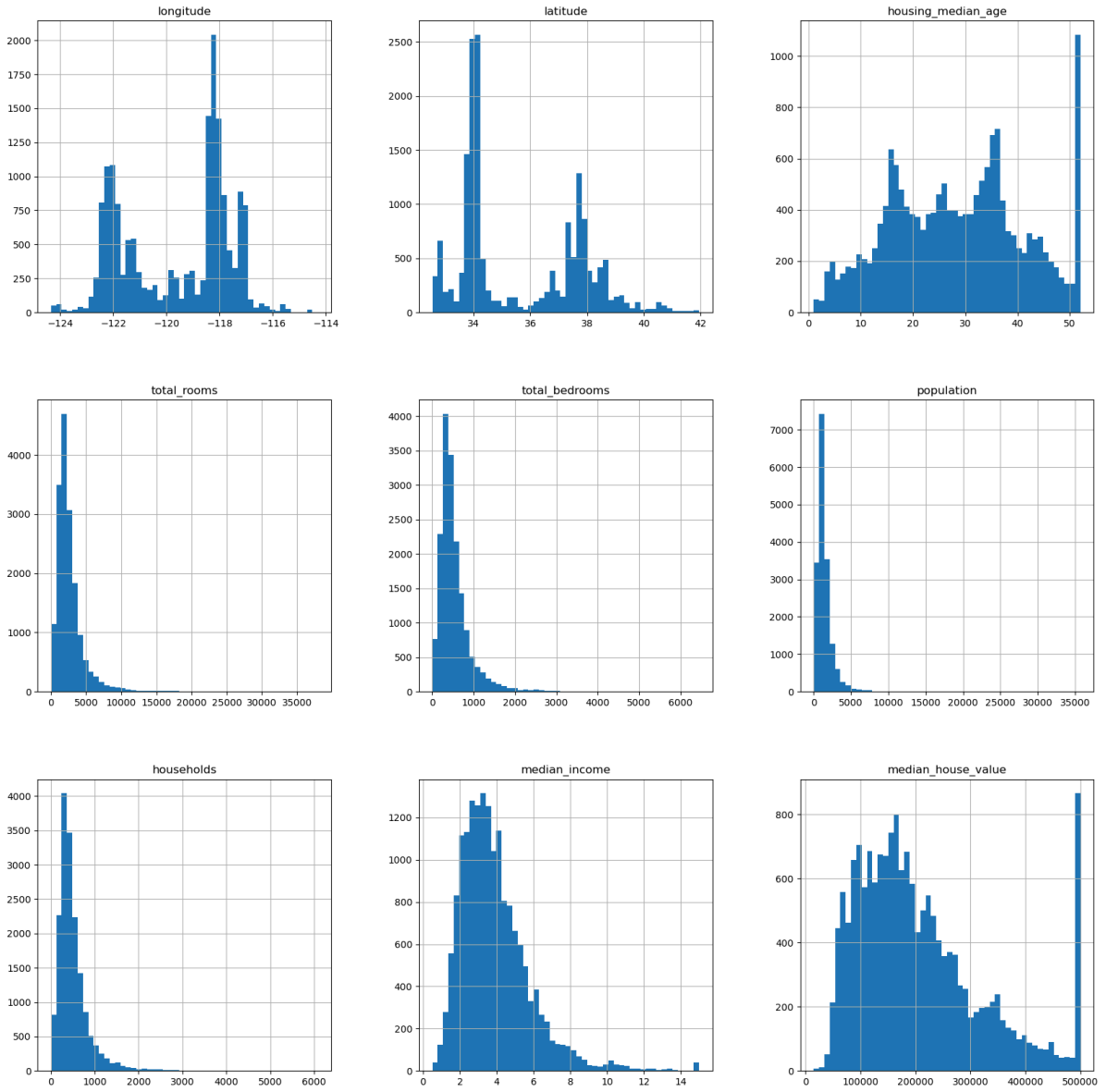
데이터 셋 설명

컬럼명	설명	예제 값
longitude	경도 - 지리적 위치 좌표	-118.49
latitude	위도 - 지리적 위치 좌표	34.25
housing_median_age	해당 지역 주택의 중간 연령	29.0
total_rooms	해당 지역 전체 방의 수	2127.0
total_bedrooms	해당 지역 전체 침실의 수	434.0
population	해당 지역 인구 수	1167.0
households	해당 지역 가구 수	409.0
median_income	해당 지역 가구의 소득 중앙값입니다. 단위는 만 달러 (tens of thousands of USD)로 표시됩니다 (예: 3.5446은 35446 USD를 의미)	3.54
median_house_value	해당 지역 주택의 가격 중앙값입니다. 단위는 USD입니다.	180400.0

02 기본 시각화

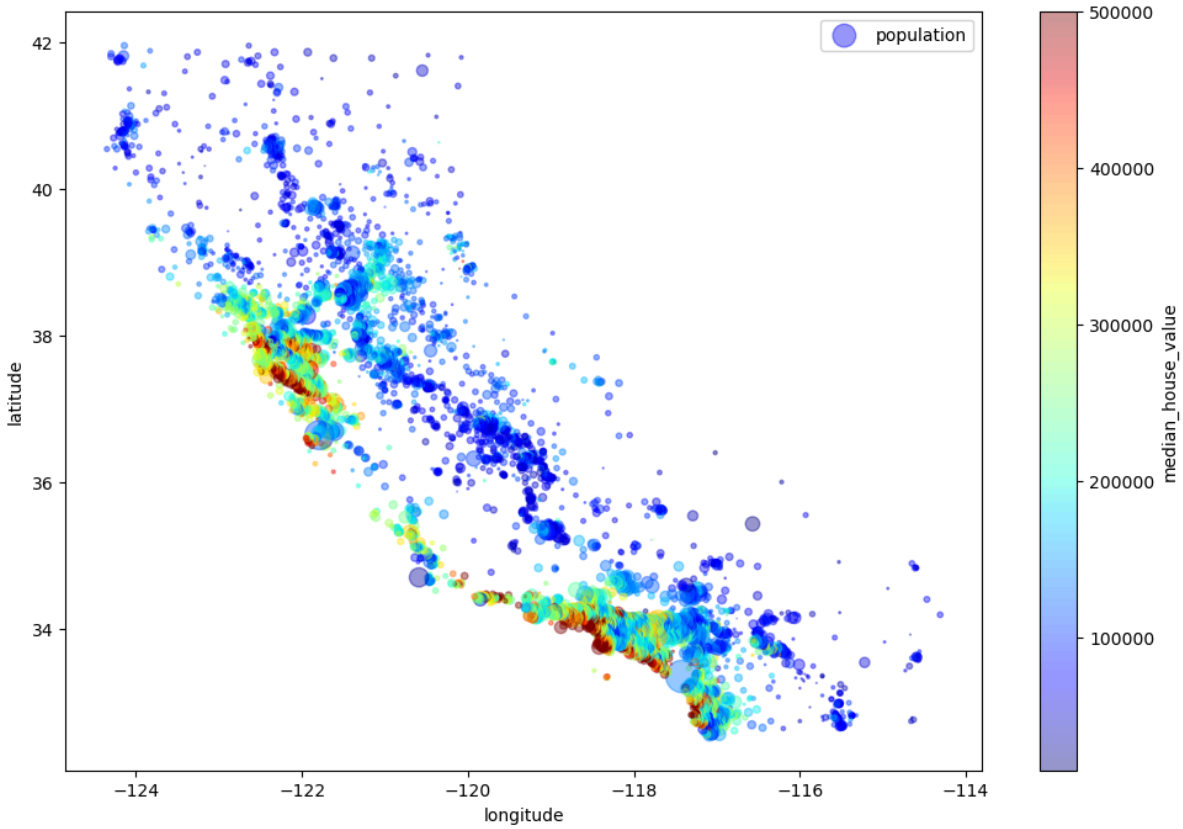
- train.hist()는 pandas DataFrame의 메서드로, 각 숫자 열에 대해 히스토그램을 생성
- bins=50: 히스토그램을 50개의 구간으로 나누어, 데이터 분포를 더 세밀하게 표시
- figsize=(20,20): matplotlib의 figure 크기를 20x20인치로 설정
- 히스토그램 분석을 통해 데이터의 편향성, 이상치, 분포 형태(정규 분포, 왼쪽/오른쪽 치우침 등)를 파악할 수 있으며, 이는 머신러닝 모델링 전 데이터 전처리 단계에서 매우 중요한 통찰을 제공

In [13]:
```
import matplotlib.pyplot as plt
train.hist(bins=50, figsize=(20,20))
plt.show()
```



In [14]:
```
### 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[14]:
```
<Axes: xlabel='longitude', ylabel='latitude'>
```



- 고가 주택 지역: 높은 주택 가격(빨간색)은 특정 클러스터(경도 -122° ~ -120°, 위도 37° ~ 38°)에 집중되어 있으며, 이는 해안가나 도시 지역의 특성을 반영할 가능성이 높습니다.
- 인구와 주택 가격의 관계 : 상관관계: 인구가 많은 지역(큰 포인트)과 높은 주택 가격(빨간색/주황색)이 북서부에서 겹치는 경향이 있습니다. 이는 도시 중심부에서 인구 밀도와 부동산 가치가 높은 것을 보여줍니다.
  - 예외: 그러나 중부 캘리포니아의 일부 인구 밀집 지역에서는 중간 가격대(녹색/노란색)가 나타나며, 인구와 가격이 항상 비례하지는 않음을 시사합니다.

In [15]:
```
train.columns
```

Out[15]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [16]:
```
sel = ['total_rooms', 'total_bedrooms', 'population']

temp_train = train[ sel  ]

print("데이터 가공 셋의 크기 : ", temp_train.shape)
print("데이터 가공 셋의 일부 : ")
print(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 [17]:
```
temp_train.describe()
```

Out[17]:

	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 [18]:
```
import seaborn as sns
```

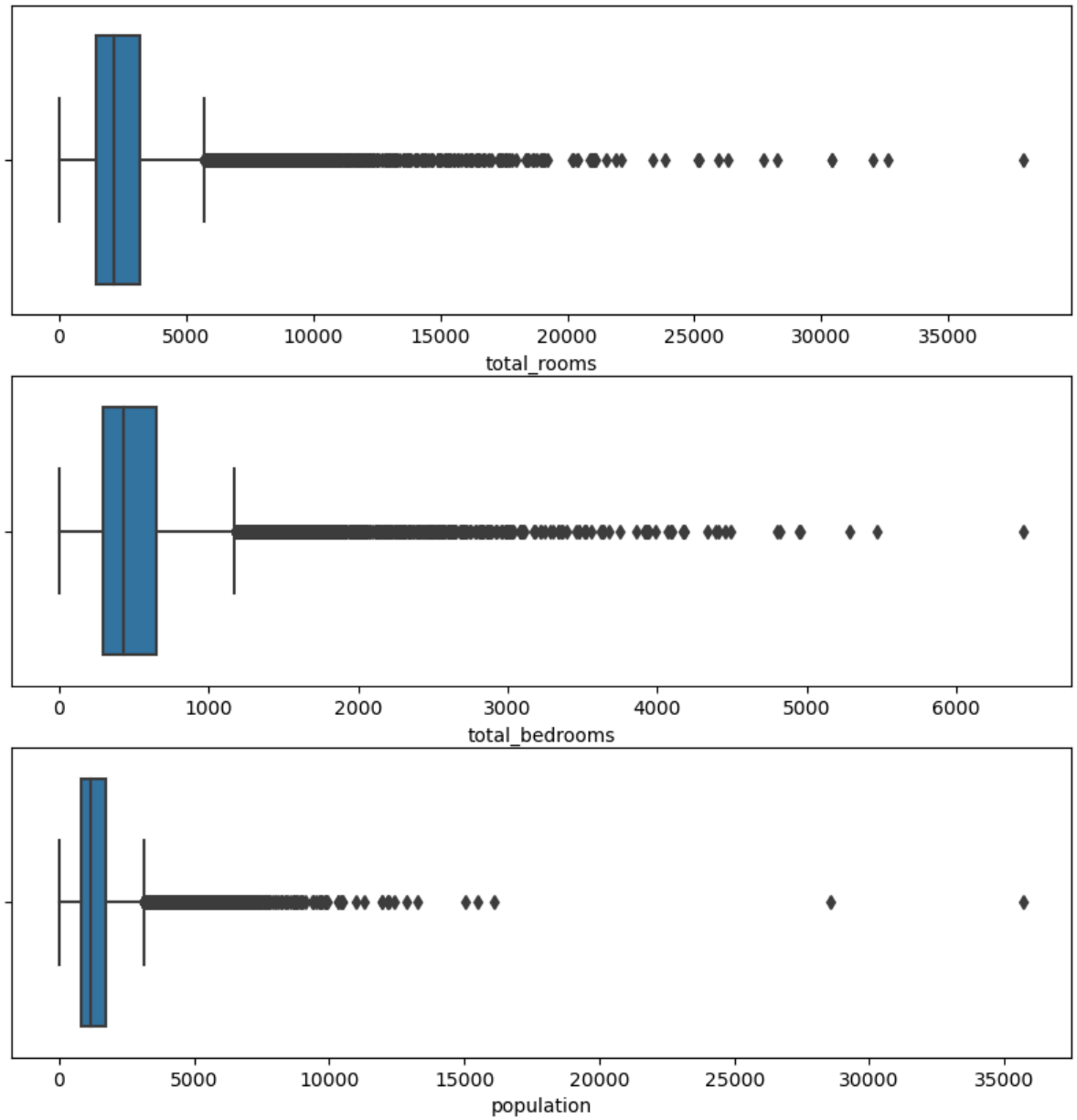
In [19]:
```
plt.figure(figsize=(10,10))

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[19]:
```
<Axes: xlabel='population'>
```



- total_rooms 박스플롯
  - 방의 총 개수 분포를 보여줍니다
  - 중앙값(박스 중간 선)은 약 2,127개 근처
  - 박스 상단과 하단은 25%와 75% 분위수(약 1,462개와 3,151개)
  - 수염(whisker) 밖의 점들은 이상치로, 상당히 많은 방을 가진 지역이 있음을 보여줍니다
  - 최대값은 37,937개로 매우 큰 이상치가 존재합니다
- total_bedrooms 박스플롯
  - 침실의 총 개수 분포를 보여줍니다
  - 중앙값은 약 434개
  - 마찬가지로 오른쪽으로 긴 꼬리를 가진 분포로, 많은 이상치가 있습니다
  - 최대값은 6,445개의 침실로 일반적인 지역보다 매우 큰 값입니다
- population 박스플롯
  - 인구수 분포를 보여줍니다
  - 중앙값은 약 1,167명

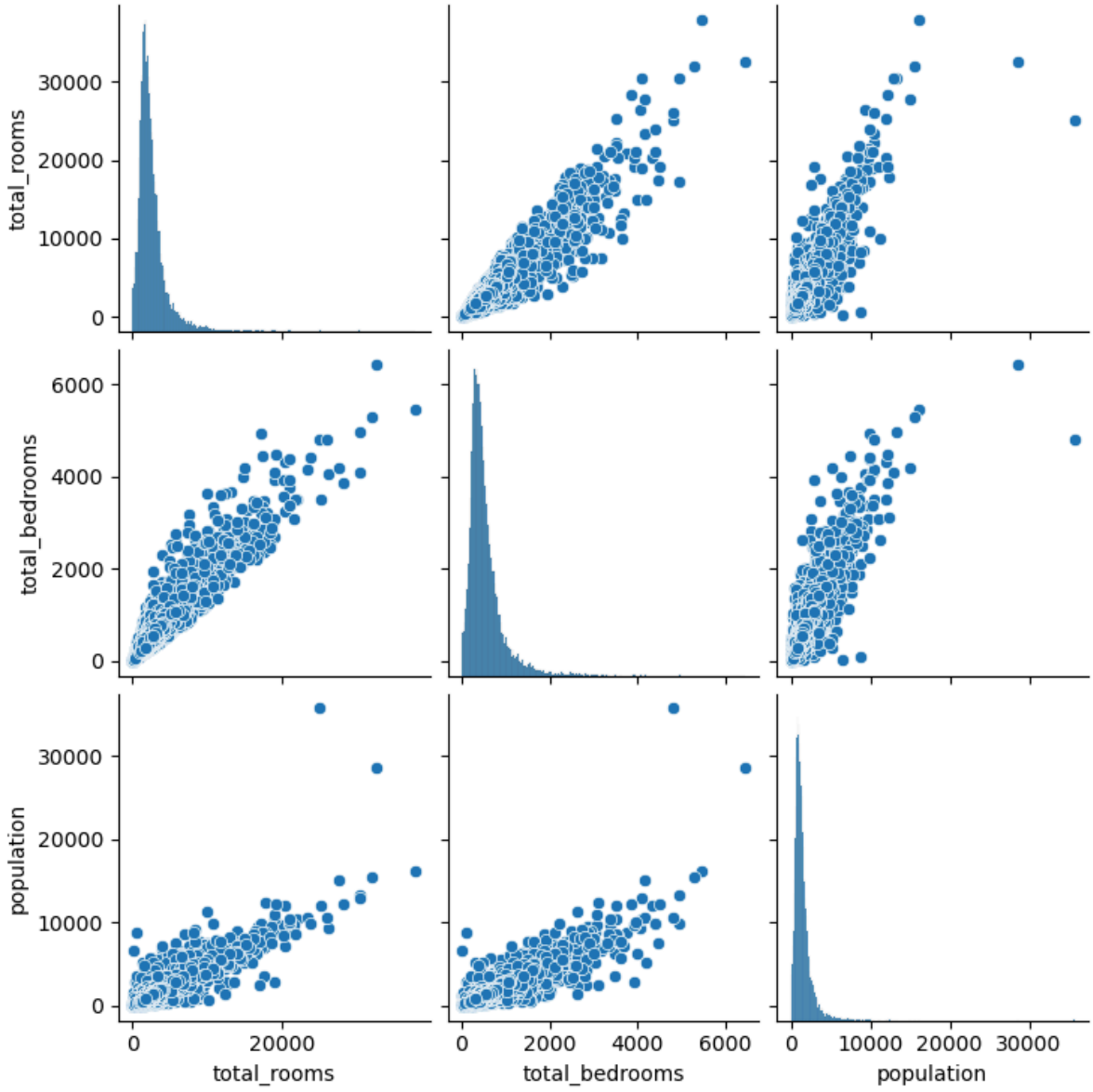
- 역시 오른쪽으로 편향된 분포를 보이며, 인구가 매우 많은 지역(최대 35,682명)이 이상치로 존재합니다

- 이 세 변수 모두 오른쪽으로 치우친(right-skewed) 분포를 보이며, 많은 이상치가 존재합니다. 이는 머신러닝 모델링을 위해 로그 변환 등의 전처리가 필요할 수 있음을 시사
- 또한 이상치가 많다는 것은 이 특성들이 정규화나 표준화 없이 사용될 경우 모델의 성능에 부정적인 영향을 미칠 수 있다는 것을 의미

In [20]:
```
sns.pairplot(temp_train)
```

```
c:\Users\colab\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x1d5a80b4450>
```



iloc, Loc 이해하기

In [21]:
```
train.columns
```

Out[21]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

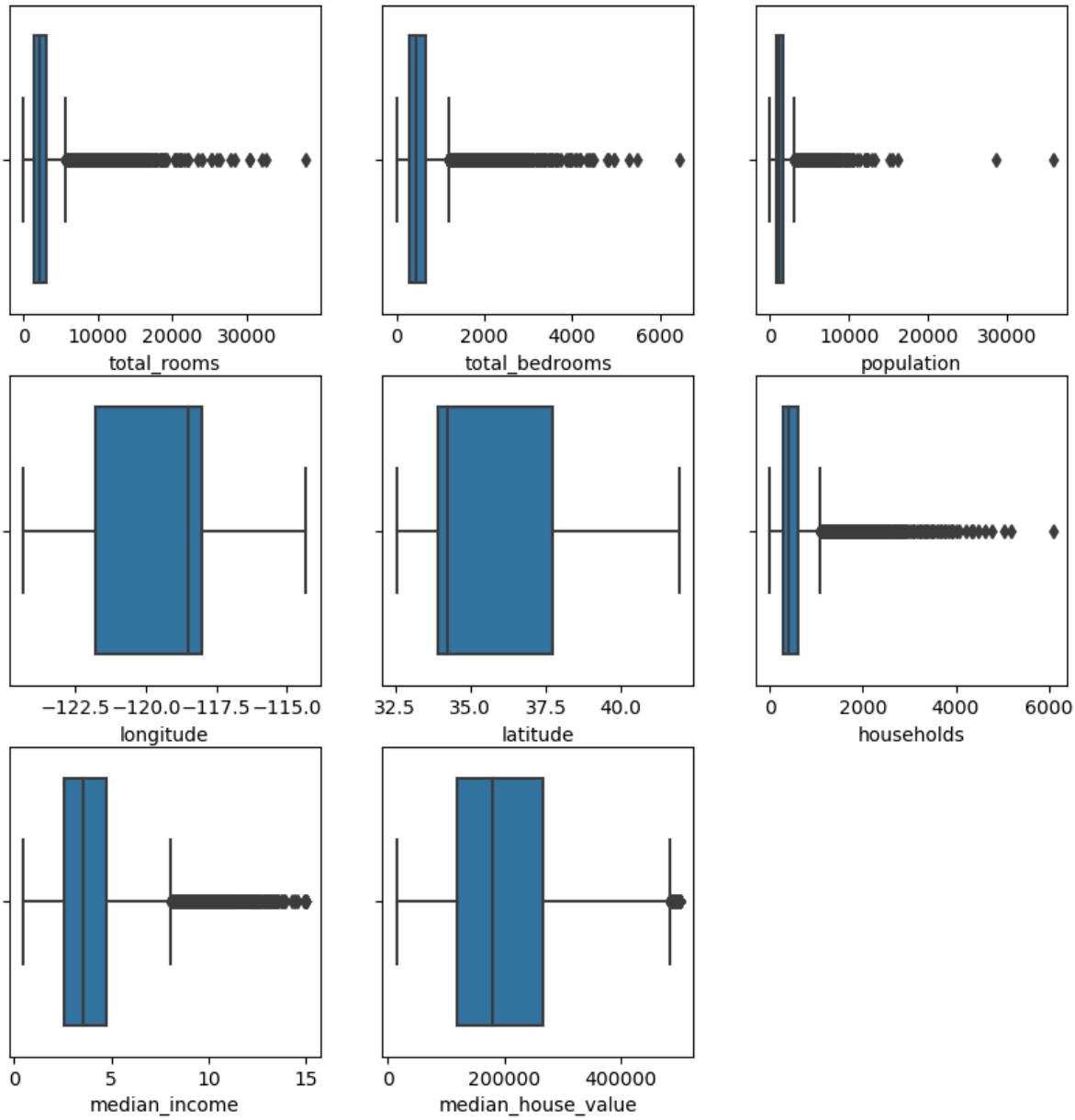
In [22]:
```
plt.figure(figsize=(10,10))

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[22]:
```
<Axes: xlabel='median_house_value'>
```



- total_rooms :
  - 대부분의 지역에서는 방 수가 10,000개 미만이지만, 극소수의 지역에서는 30,000개 이상으로 매우 높은 이상치가 존재합니다.
  - 분포가 오른쪽으로 심하게 치우쳐 있어 로그 변환 같은 전처리가 필요할 수 있습니다.

데이터 전처리 관점에서의 분석:

- total_rooms, total_bedrooms, population, households 변수들은 모두 심한 비대칭 분포와 많은 이상치를 가지고 있어, 로그 변환이나 스케일링이 필요합니다.
- 지리적 위치 데이터(longitude, latitude)는 비교적 균등한 분포를 보이므로 추가 전처리가 덜 필요할 수 있습니다.
- median_income과 median_house_value도 변환이 필요할 수 있지만, 다른 변수들에 비해 덜 치우쳐 있습니다.

모델링 관점에서의 분석:

- 이러한 비대칭성과 이상치는 선형 회귀와 같은 모델의 성능에 부정적인 영향을 미칠 수 있습니다.

- 이상치 처리 방법(제거, 윈저화 등)을 고려해야 합니다.
- 변수들 간의 관계(예: total_rooms와 total_bedrooms의 상관관계)를 추가로 분석해볼 필요가 있습니다.

In [23]:
```
## 두 컬럼 선택
temp02 = train.loc[:,  [ "median_income", "median_house_value" ] ]
temp02.head()
```

Out[23]:

	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 [24]:
```
train.columns
```

Out[24]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [25]:
```
## 두 컬럼 선택 8열, 9열
temp03 = train.iloc[:,  [7, 8] ]
print( temp03.head() )

print()

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 [26]:
```
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 [27]:
```
## 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() )

print()

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 [28]:
```
train.head()
```

Out[28]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0

In [29]:
```
train.total_rooms.describe()
```

Out[29]:
```
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
```

03 조건을 이용하여 데이터를 그룹화 시켜보자.

In [32]:
```
# 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

# 기술 통계를 미리 계산
stats = train['total_rooms'].describe()
stats
```

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
```

In [36]:
```
# 25%와 75% 사이의 데이터 추출 (IQR)
q1_value = stats['25%']  # 1462.0
q2_value = stats['50%']  # 2127.0
q3_value = stats['75%']  # 3151.25
q1_value, q2_value, q3_value
```

Out[36]:
```
(1462.0, 2127.0, 3151.25)
```

In [37]:
```
tmp_A1 = train[ train['total_rooms']> q3_value]
print(tmp_A1.shape)

tmp_A1.head()
```

```
(4250, 9)
```

Out[37]:

	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.(
8	-114.59	33.61	34.0	4789.0	1175.0	3134.0	1056.(
10	-114.60	33.62	16.0	3741.0	801.0	2434.0	824.(
38	-115.48	32.68	15.0	3414.0	666.0	2097.0	622.(

In [38]:
```
tmp_A2 = train[ (train['total_rooms']> q2_value) & (train['total_rooms'] <= q3_value
print(tmp_A2.shape)

tmp_A2.head()
```

```
(4247, 9)
```

Out[38]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
6	-114.58	33.61	25.0	2907.0	680.0	1841.0	633.(
13	-114.61	34.83	31.0	2478.0	464.0	1346.0	479.(
15	-114.65	34.89	17.0	2556.0	587.0	1005.0	401.(
42	-115.49	32.67	25.0	2322.0	573.0	2185.0	602.(
45	-115.50	32.67	35.0	2159.0	492.0	1694.0	475.(

In [39]:
```
tmp_A3 = train[ (train['total_rooms']> q1_value) & (train['total_rooms'] <= q2_value
print(tmp_A3.shape)

tmp_A3.head()
```

```
(4249, 9)
```

Out[39]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.(
9	-114.60	34.83	46.0	1497.0	309.0	787.0	271.(
11	-114.60	33.60	21.0	1988.0	483.0	1182.0	437.(
16	-114.65	33.60	28.0	1678.0	322.0	666.0	256.(
20	-114.68	33.49	20.0	1491.0	360.0	1135.0	303.(

In [40]:
```
tmp_A4 = train [ train['total_rooms']> q1_value ]
print(tmp_A4.shape)

tmp_A4.head()
```

```
(12746, 9)
```

Out[40]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0
6	-114.58	33.61	25.0	2907.0	680.0	1841.0	633.0
8	-114.59	33.61	34.0	4789.0	1175.0	3134.0	1056.0

In [42]:
```
print(tmp_A1.shape, tmp_A2.shape, tmp_A3.shape, tmp_A4.shape)
```

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

실습해보기 - room_level 컬럼 추가하기

In [44]:
```
# 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
```

In [45]:
```
import numpy as np
```

In [46]:
```
### 새로운 컬럼 room_level 만들기
bool_val = np.where( (train['total_rooms']> q3_value) , True, False)
train.loc[bool_val, "room_level"] = 1

bool_val = np.where( (train['total_rooms']> q2_value) & (train['total_rooms'] <= q3_
train.loc[bool_val, "room_level"] = 2

bool_val = np.where( (train['total_rooms']> q1_value) & (train['total_rooms'] <= q2_
train.loc[bool_val, "room_level"] = 3

bool_val = np.where( (train['total_rooms'] <= 1462) , True, False)
train.loc[bool_val, "room_level"] = 4
train['room_level'].head(15)
```

Out[46]:
```
0     1.0
1     1.0
2     4.0
3     3.0
4     4.0
5     4.0
6     2.0
7     4.0
8     1.0
9     3.0
10    1.0
11    3.0
12    4.0
13    2.0
14    4.0
Name: room_level, dtype: float64
```

In [47]:
```
# 한 줄로 room_level 컬럼 생성
train['room_level2'] = pd.qcut(train['total_rooms'], q=4, labels=[4, 3, 2, 1])
```

In [48]:
```
train
```

Out[48]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	4
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	4
2	-114.56	33.69	17.0	720.0	174.0	333.0	1
3	-114.57	33.64	14.0	1501.0	337.0	515.0	2
4	-114.57	33.57	20.0	1454.0	326.0	624.0	2
...	...	...	...	...	...	...	
16995	-124.26	40.58	52.0	2217.0	394.0	907.0	3
16996	-124.27	40.69	36.0	2349.0	528.0	1194.0	4
16997	-124.30	41.84	17.0	2677.0	531.0	1244.0	4
16998	-124.30	41.80	19.0	2672.0	552.0	1298.0	4
16999	-124.35	40.54	52.0	1820.0	300.0	806.0	2

17000 rows × 11 columns

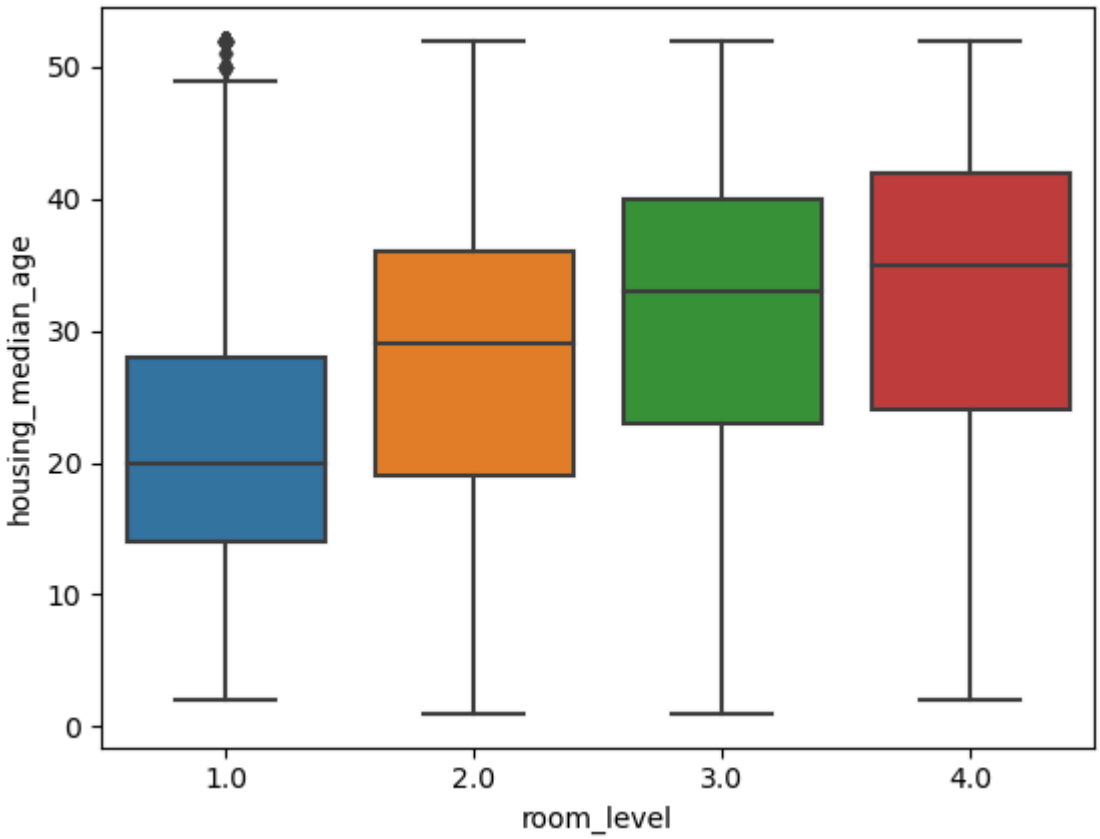
- pd.qcut(train['total_rooms'], q=4): total_rooms 컬럼을 4개의 구간으로 나눕니다. 각 구간은 데이터의 25%씩을 포함하도록 사분위수 기준으로 자동 계산됩니다.
- labels=[4, 3, 2, 1]: 각 구간에 레이블을 지정합니다.

In [49]:
```
### 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 [50]:
```
### 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[50]:
```
<Axes: xlabel='room_level', ylabel='housing_median_age'>
```



종합적인 해석:

- 뚜렷한 경향성: 방의 수(room_level)와 주택 연령(housing_median_age) 사이에 명확한 경향성이 있습니다. 방이 많을수록 주택 연령이 낮고, 방이 적을수록 주택 연령이 높습니다.
- 도시화 패턴: 이는 도시화 패턴을 반영할 수 있습니다. 최근에 지어진 건물들(연령이 낮은)이 더 많은 방을 가지는 경향이 있고, 오래된 주택들은 상대적으로 방 수가 적을 수 있습니다.
- 부동산 개발 트렌드: 시간이 지남에 따라 주택 설계가 변화했을 가능성도 있습니다. 최근의 개발 트렌드가 더 많은 방을 갖춘 주택을 선호하는 방향으로 바뀌었을 수 있습니다.
- 이상치: 각 그룹에 몇몇 이상치가 존재하지만, 특히 room_level 1.0 그룹에서 주택 연령이 50년 이상인 몇몇 이상치가 보입니다.

지도 시각화

In [51]:
```
# 위도(latitude), 경도(longitude) 를 이용한 위치표시
import folium

print(folium.__version__)
```

```
0.19.5
```

In [52]:
```
df = train.copy()
```

In [53]:
```
df_name = df.index
df_lati = df['latitude']
```

```
df_long = df['longitude']
```

In [54]:
```
import numpy as np

df_lati = list(df_lati)
df_long = list(df_long)
df_loc = np.array([df_lati, df_long]).T
print( df_loc.shape )
print( np.mean( df_lati) , np.mean(df_long)  )
df_loc
```

Out[54]:
```
(17000, 2)
35.62522470588235 -119.5621082352941
array([[  34.19, -114.31],
       [  34.4 , -114.47],
       [  33.69, -114.56],
       ...,
       [  41.84, -124.3 ],
       [  41.8 , -124.3 ],
       [  40.54, -124.35]])
```

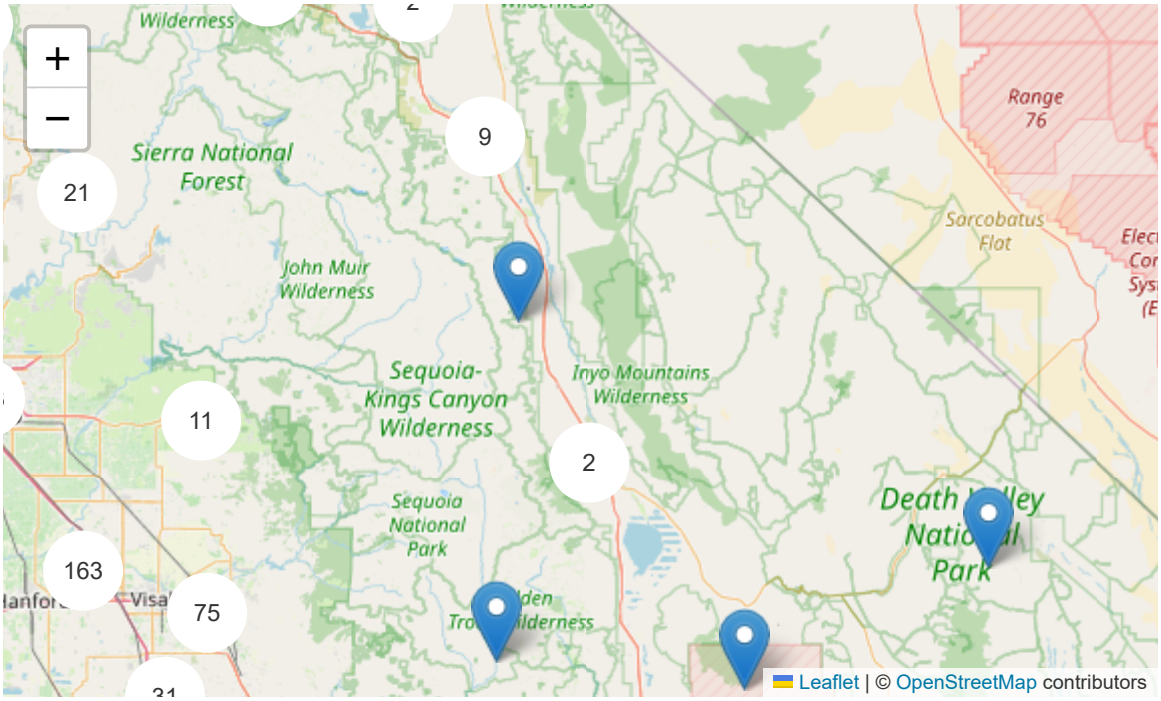
In [55]:
```
from folium import plugins
import os

house_map = folium.Map(location=[ np.mean( df_lati), np.mean(df_long) ],
                       zoom_start=6)

df_name = list(df_name)
plugins.MarkerCluster(df_loc, popups=df_name).add_to(house_map)

house_map.save(os.path.join('.', 'california_location.html'))
house_map
```

Out[55]:



Reference

- https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html