05 판다스를 활용한 데이터 이해

학습 내용

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

01 캘리포니아 데이터 가져오기

In [1]:

```
import pandas as pd
```

In [2]:

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

```
pandas 버전  1.4.2
```

In [3]:

```
train = pd.read_csv("https://storage.googleapis.com/mledu-datasets/california_housing_train.csv", se
test = pd.read_csv("https://storage.googleapis.com/mledu-datasets/california_housing_test.csv", sep=
train.shape, test.shape
```

Out[3]:

```
((17000, 9), (3000, 9))
```

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	househol
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	househol
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):
 #   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 [10]:

```
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 [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)

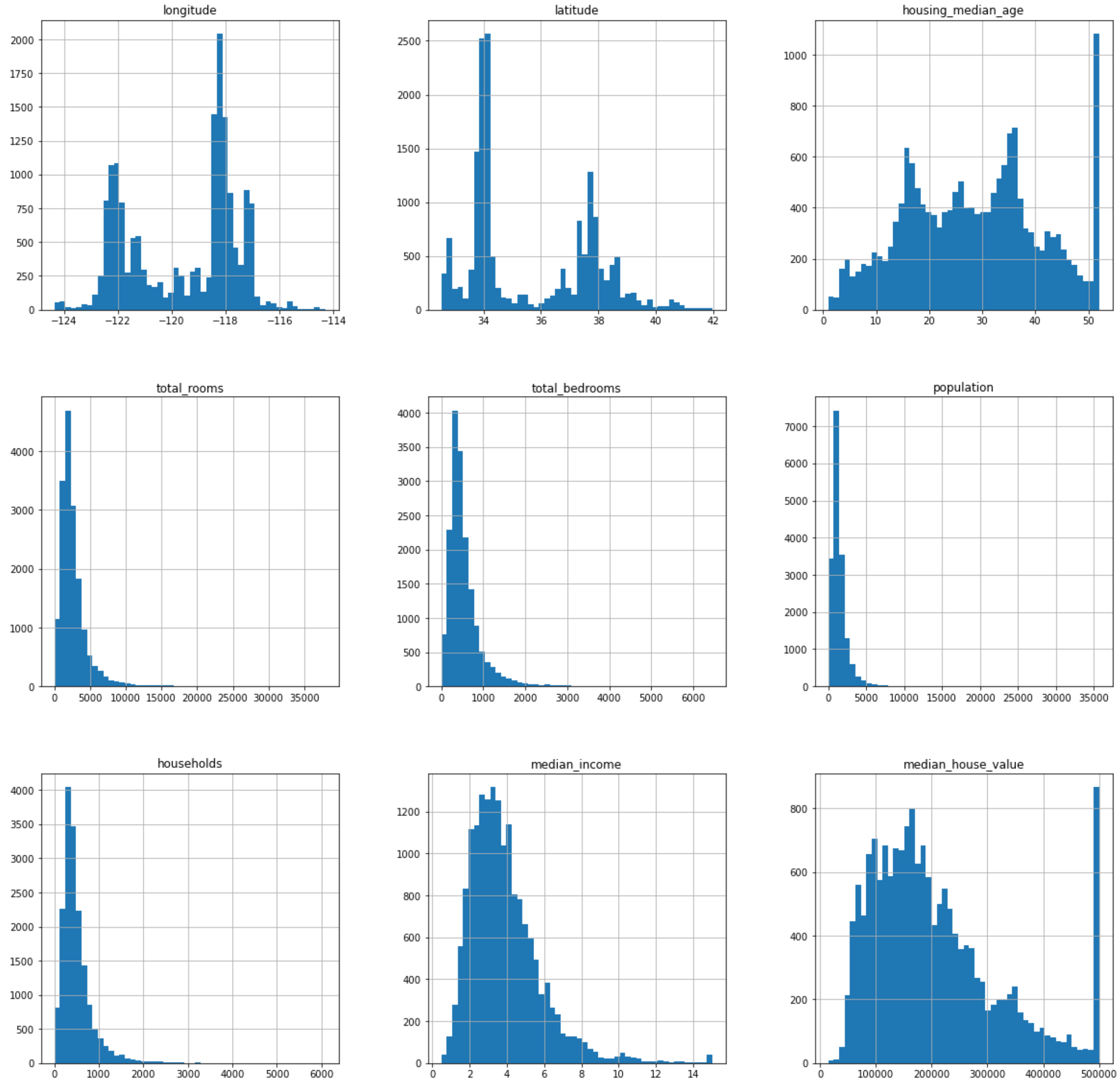
데이터 셋 설명

값	의미	기본값
longitude	집이 서쪽으로 얼마나 떨어져 있는지를 나타내는 척도. 집이 서쪽으로 얼마나 떨어져 있는지를 나타내는 척도. 더 높은 값은 서쪽으로 더 멀리 있다	
latitude	주택이 북쪽으로 얼마나 떨어져 있는지를 나타내는 척도. 더 높은 값은 북쪽으로 더 멀리 있음.	---
housingMedianAge	블록내 주택의 중간값 연식. 낮은 숫자는 최신 건물	---
totalRooms	총 객실 수	---
totalBedrooms	블록 내 총 침실 수	---
population	블록 내에 상주하는 총 인원 수	---
households	주택 단위 내에 거주하는 가구 그룹인 블록의 총 가구 수	---
medianIncome	한 블록 내 가구의 중위 소득(미국 달러 수만달러로 추정)	---
medianHouseValue	블록 내 가구의 중위 House Value(미국 달러로 추정)	---

02 기본 시각화

In [12]:

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

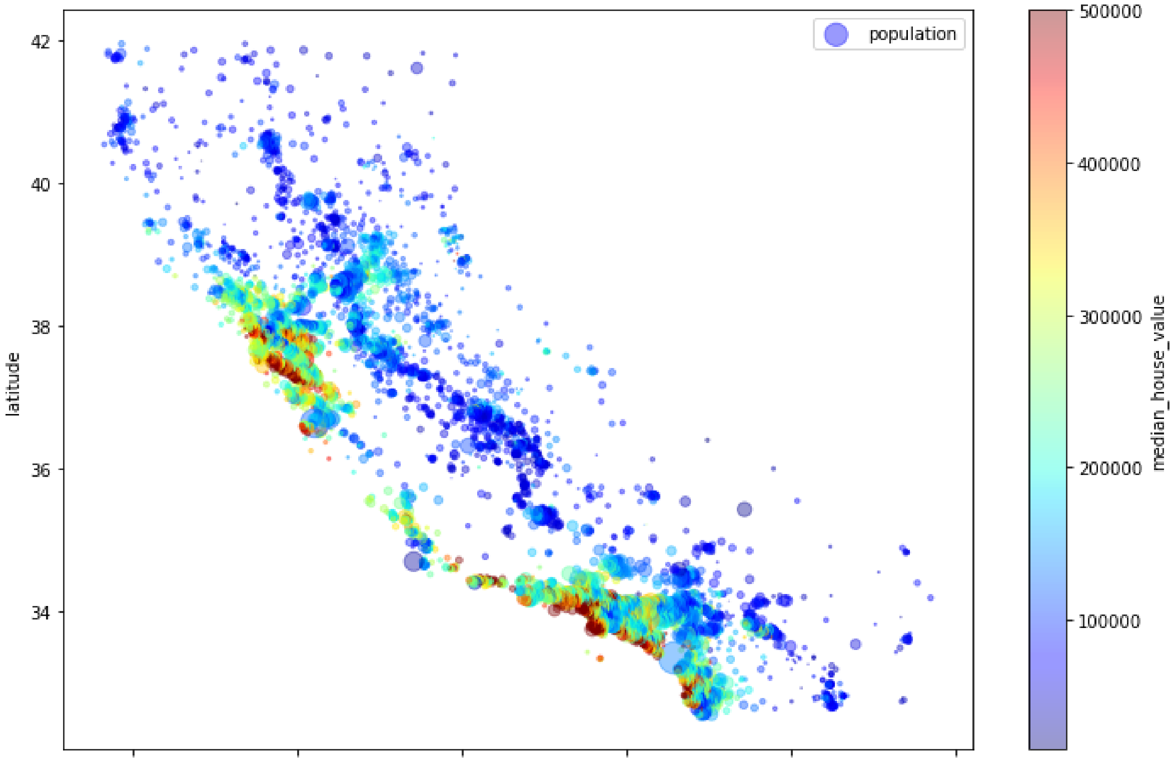


In [13]:

```
### 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[13]:

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



In [14]:

```
train.columns
```

Out[14]:

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

In [15]:

```
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 [16]:

```
temp_train.describe()
```

Out[16]:

	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 [17]:

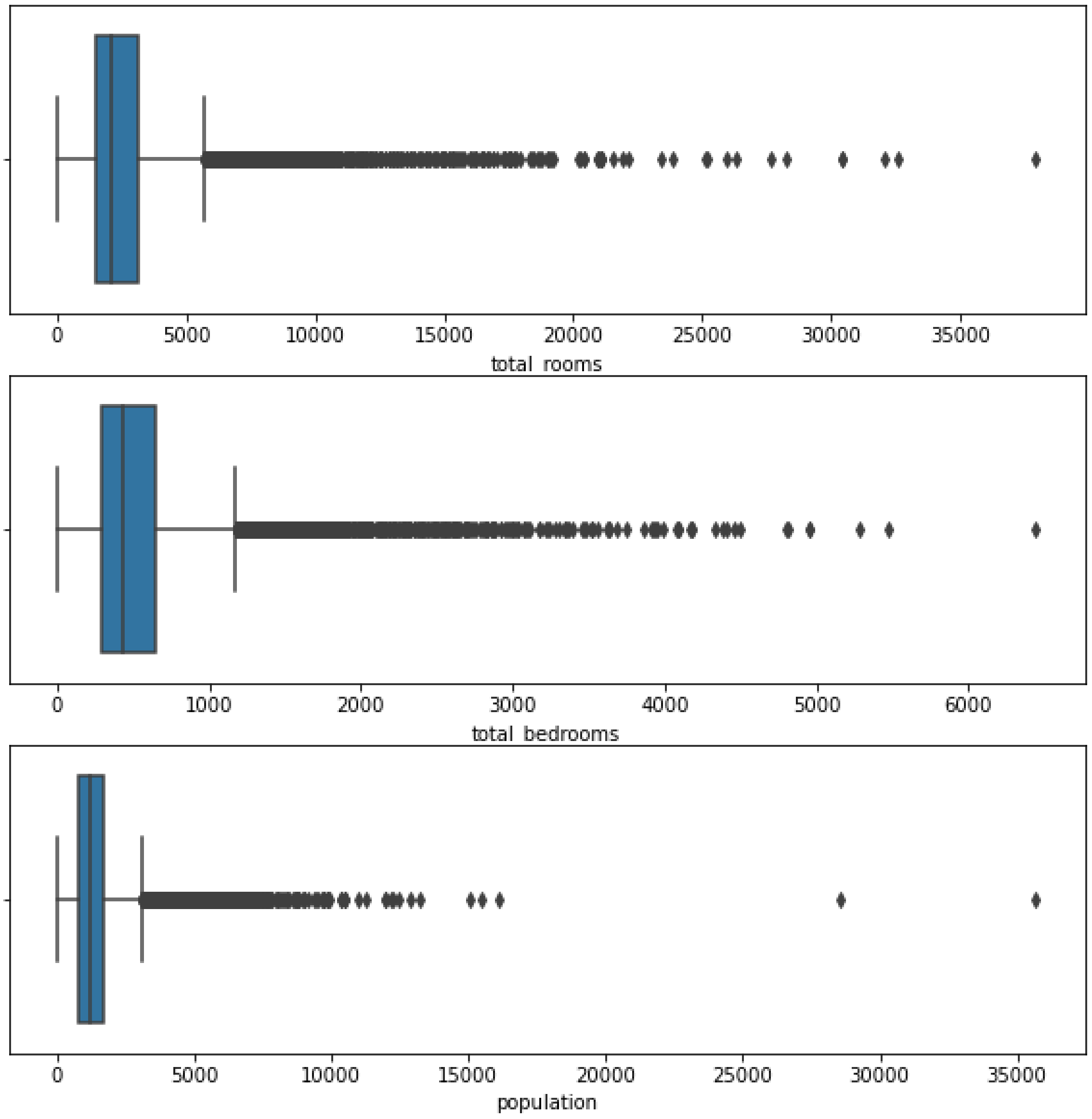
```
import seaborn as sns
```

In [18]:

```
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[18]:

```
<AxesSubplot:xlabel='population'>
```

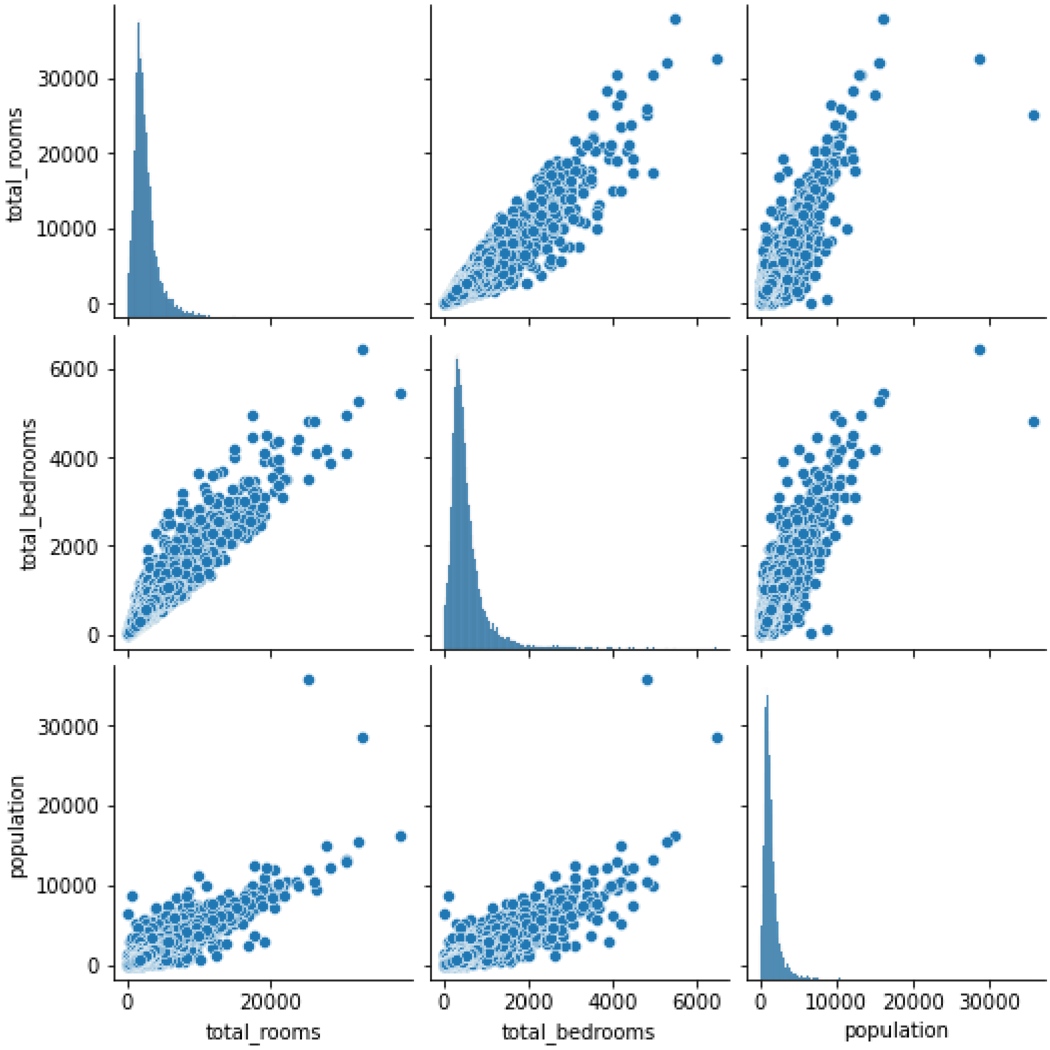


In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x260dbbc4250>
```



iloc, Loc 이해하기

In [20]:

```
train.columns
```

Out[20]:

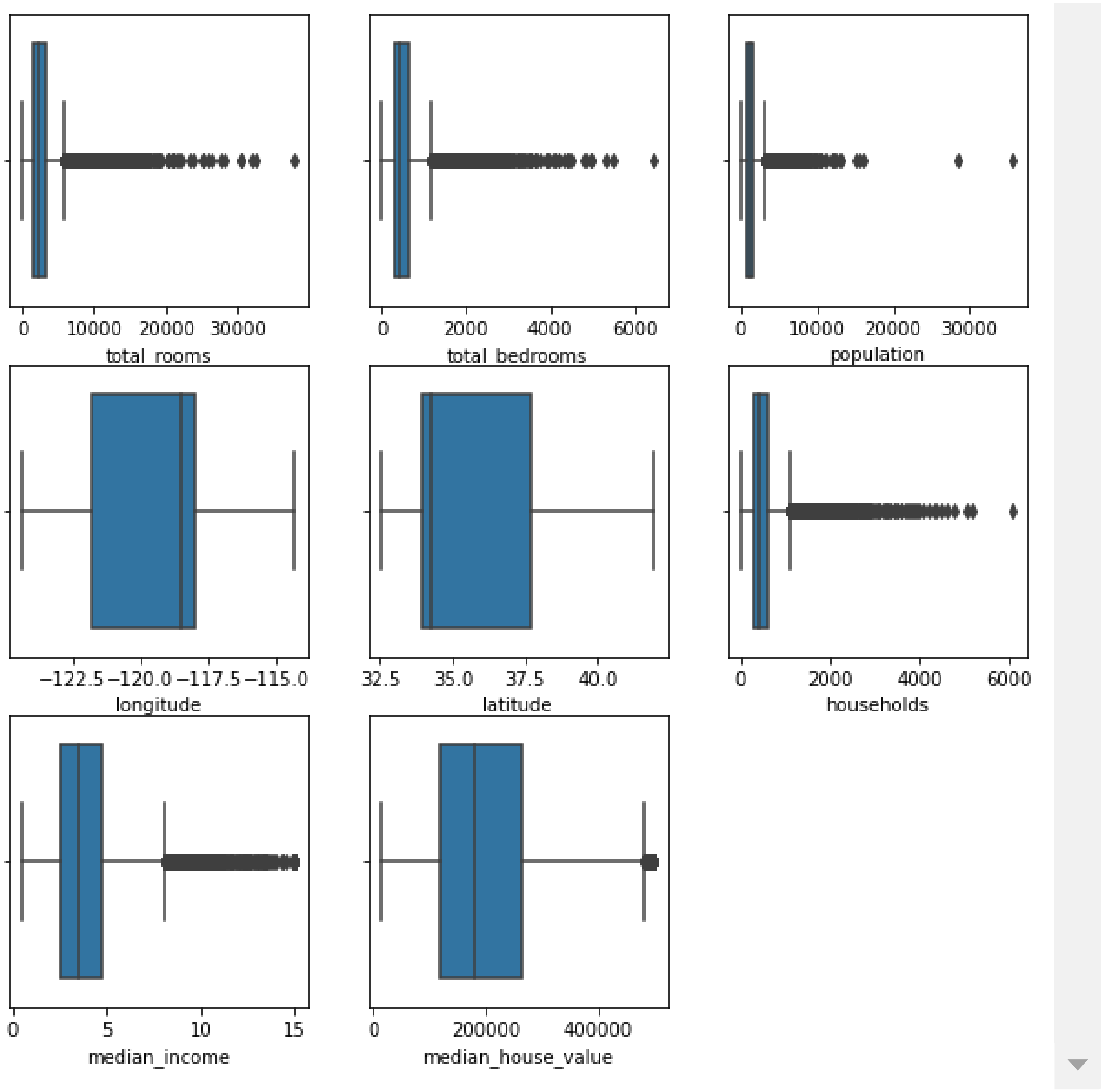
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [21]:

```
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[21]:

```
<AxesSubplot:xlabel='median_house_value'>
```

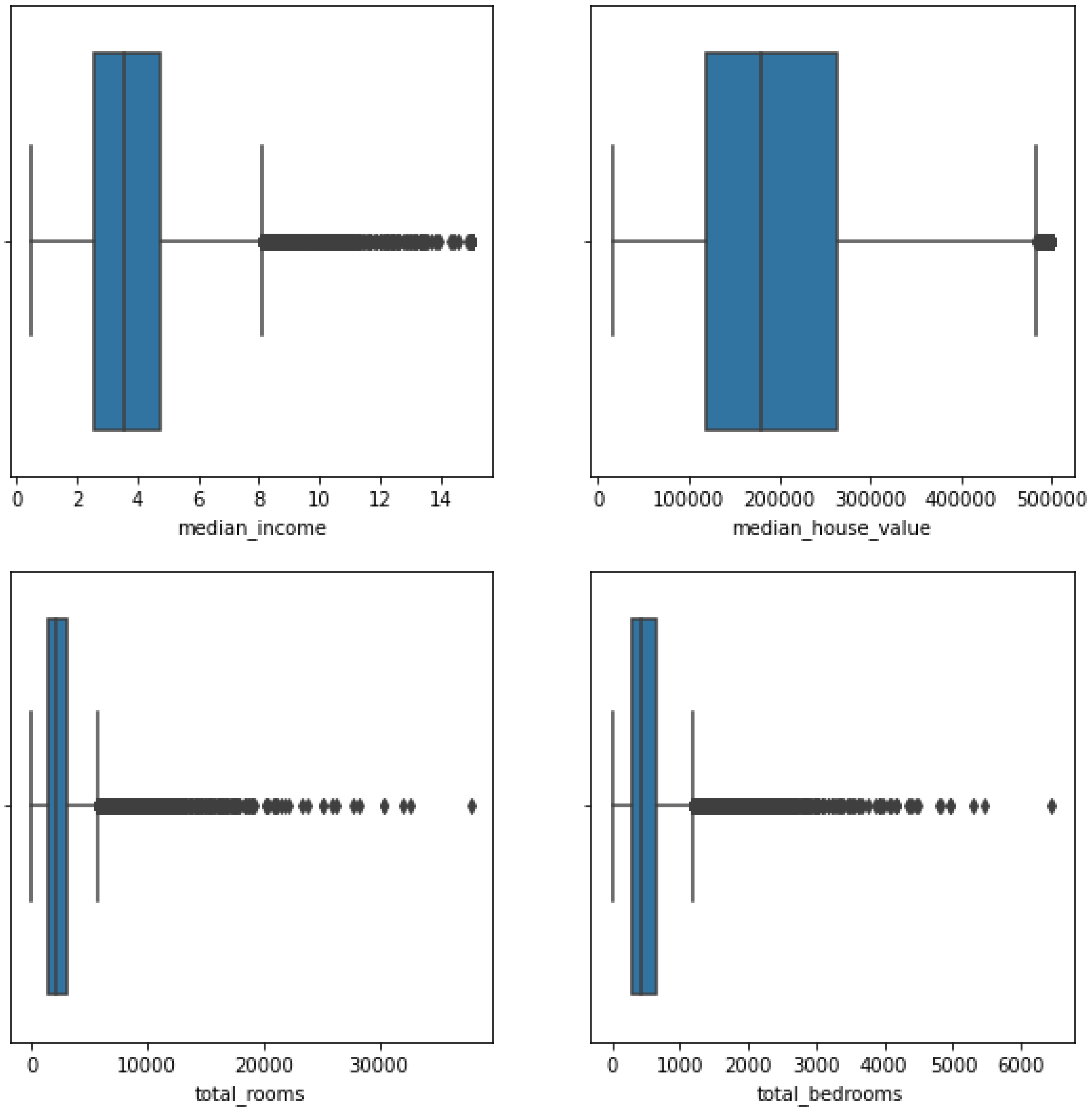

0
10000
20000
30000
total_rooms
0
2000
4000
6000
total_bedrooms
0
10000
20000
30000
population
−122.5
−120.0
−117.5
−115.0
longitude
32.5
35.0
37.5
40.0
latitude
0
2000
4000
6000
households
0
5
10
15
median_income
0
200000
400000
median_house_value

In [22]:

```
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[22]:

```
<AxesSubplot:xlabel='total_bedrooms'>
```



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	householc
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 [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 데이터 그룹을 시켜보자.

In [30]:

```
# 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.shape)

tmp_A1.head()
```

```
(4250, 9)
```

Out[30]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	47
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	46
8	-114.59	33.61	34.0	4789.0	1175.0	3134.0	105
10	-114.60	33.62	16.0	3741.0	801.0	2434.0	82
38	-115.48	32.68	15.0	3414.0	666.0	2097.0	62

In [31]:

```
import numpy as np
```

In [32]:

```
tmp_A2 = train[ (train['total_rooms']> 2127) & (train['total_rooms'] <= 3151) ]
print(tmp_A2.shape)

tmp_A2.head()
```

```
(4247, 9)
```

Out[32]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
6	-114.58	33.61	25.0	2907.0	680.0	1841.0	63
13	-114.61	34.83	31.0	2478.0	464.0	1346.0	47
15	-114.65	34.89	17.0	2556.0	587.0	1005.0	40
42	-115.49	32.67	25.0	2322.0	573.0	2185.0	60
45	-115.50	32.67	35.0	2159.0	492.0	1694.0	47

In [33]:

```
tmp_A3 = train[ (train['total_rooms']> 1462) & (train['total_rooms'] <= 2127) ]
print(tmp_A3.shape)

tmp_A3.head()
```

```
(4249, 9)
```

Out[33]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
3	-114.57	33.64	14.0	1501.0	337.0	515.0	22
9	-114.60	34.83	46.0	1497.0	309.0	787.0	27
11	-114.60	33.60	21.0	1988.0	483.0	1182.0	43
16	-114.65	33.60	28.0	1678.0	322.0	666.0	25
20	-114.68	33.49	20.0	1491.0	360.0	1135.0	30

In [34]:

```
tmp_A4 = train [ train['total_rooms']> 1462 ]
print(tmp_A4.shape)

tmp_A4.head()
```

```
(12746, 9)
```

Out[34]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	householc
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
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226
6	-114.58	33.61	25.0	2907.0	680.0	1841.0	633
8	-114.59	33.61	34.0	4789.0	1175.0	3134.0	1056

In [35]:

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

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

In [36]:

```
### 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

### 새로운 컬럼 room_level 만들기
bool_val = np.where( (train['total_rooms']> 3151) , True, False)
train.loc[bool_val, "room_level"] = 1
train['room_level'].head(15)
```

Out[36]:

```
0     1.0
1     1.0
2     NaN
3     NaN
4     NaN
5     NaN
6     NaN
7     NaN
8     1.0
9     NaN
10    1.0
11    NaN
12    NaN
13    NaN
14    NaN
Name: room_level, dtype: float64
```

In [37]:

```
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(15)
```

Out[37]:

```
0     1.0
1     1.0
2     NaN
3     NaN
4     NaN
5     NaN
6     2.0
7     NaN
8     1.0
9     NaN
10    1.0
11    NaN
12    NaN
13    2.0
14    NaN
Name: room_level, dtype: float64
```

In [38]:

```
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(15)
```

Out[38]:

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

In [39]:

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

Out[39]:

```
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 [40]:

```
train.columns
```

Out[40]:

```
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 [41]:

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

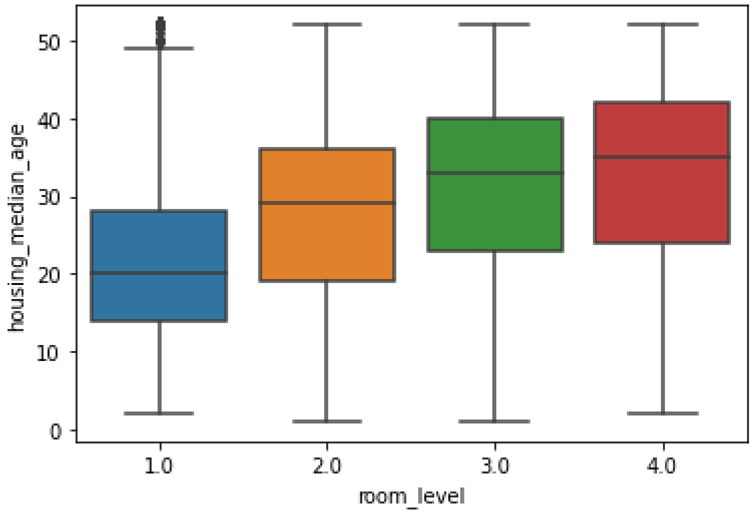
```
room_level
1.0    21.170353
2.0    28.872145
3.0    31.580137
4.0    32.731782
Name: housing_median_age, dtype: float64
```

In [42]:

```
### 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[42]:

```
<AxesSubplot:xlabel='room_level', ylabel='housing_median_age'>
```



[추가 학습] 시각화 - folium

In [44]:

```
# 위도(latitude), 경도(longitude) 를 이용한 위치표시
import folium

print(folium.__version__)
```

```
0.12.1.post1
```

In [46]:

```
df = train.copy()
```

In [47]:

```
df_name = df.index
df_lati = df['latitude']
df_long = df['longitude']
```

In [48]:

```
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
```

```
(17000, 2)
35.62522470588235 -119.5621082352941
```

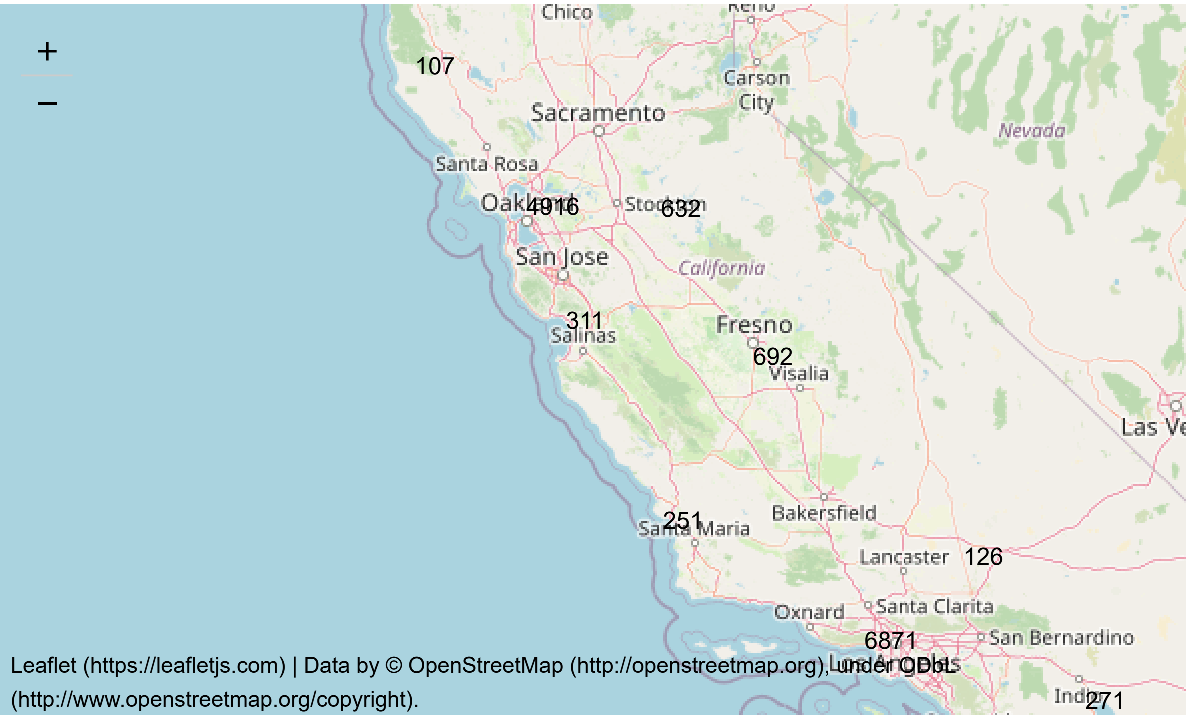
Out[48]:

```
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 [49]:

```
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[49]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Reference

- [https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html (https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html)](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html)