Group 2 Project

Contents

Analysing Research Questions

Running Regression Models

Importing Libraries and Dataset

Libraries

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import scipy.stats as stats
```

Dataset

In [2]:

```
cali = pd.read_csv('california-housing-data.csv')

cali
```

Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20640 rows × 10 columns

In [3]:
```
cali.describe()
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000

Cleaning Data

- back to top

Before beginning to analyse the data, cleaning it by removing NaN values as well as potential outliers

Checking and removing NaN values

In [4]:
```
cali.isna().sum()
```

Out[4]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

In [5]:
```
cali_cleaned = cali.dropna()
cali_cleaned
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 10 columns

In [6]:
```
cali_cleaned.isna().sum()
```

Out[6]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

In [7]:
```
cali_cleaned.describe()
```

Out[7]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
count	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	20433.0000
mean	-119.570689	35.633221	28.633094	2636.504233	537.870553	1424.9469
std	2.003578	2.136348	12.591805	2185.269567	421.385070	1133.2084
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.0000
25%	-121.800000	33.930000	18.000000	1450.000000	296.000000	787.0000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.0000
75%	-118.010000	37.720000	37.000000	3143.000000	647.000000	1722.0000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.0000

Processing Data

- back to top

Manipulating/deriving new data from set to aid in analysis.

New column household avg/density

In [8]:

```
cali_cleaned = pd.DataFrame(cali_cleaned)
cali_cleaned['household_density'] = cali_cleaned['population'] / cali_cleaned['hou
cali_cleaned = cali_cleaned.round({'household_density': 2})
cali_cleaned
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 11 columns

Getting dummy variables for ocean proximity

In [9]:

```
dummies = pd.get_dummies(cali_cleaned['ocean_proximity'])
cali_cleaned = pd.merge(cali_cleaned, dummies, left_index=True,
         right_index=True)
cali_cleaned
```

Out[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 16 columns

Standardising scores for house value, house age, and income

In [10]:

```
cali_cleaned['house_value_standard'] = stats.zscore(cali_cleaned['median_house_valu
cali_cleaned['housing_age_standard'] = stats.zscore(cali_cleaned['housing_median_ag
cali_cleaned['income_standard'] = stats.zscore(cali_cleaned['median_income'])
cali_cleaned
```

Out[10]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 19 columns

Dropping apartments ie. household density>=10

In [11]:
```
apartments = cali_cleaned[ (cali_cleaned['household_density'] >= 10)].index
cali_cleaned.drop(apartments , inplace=True)
cali_cleaned
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

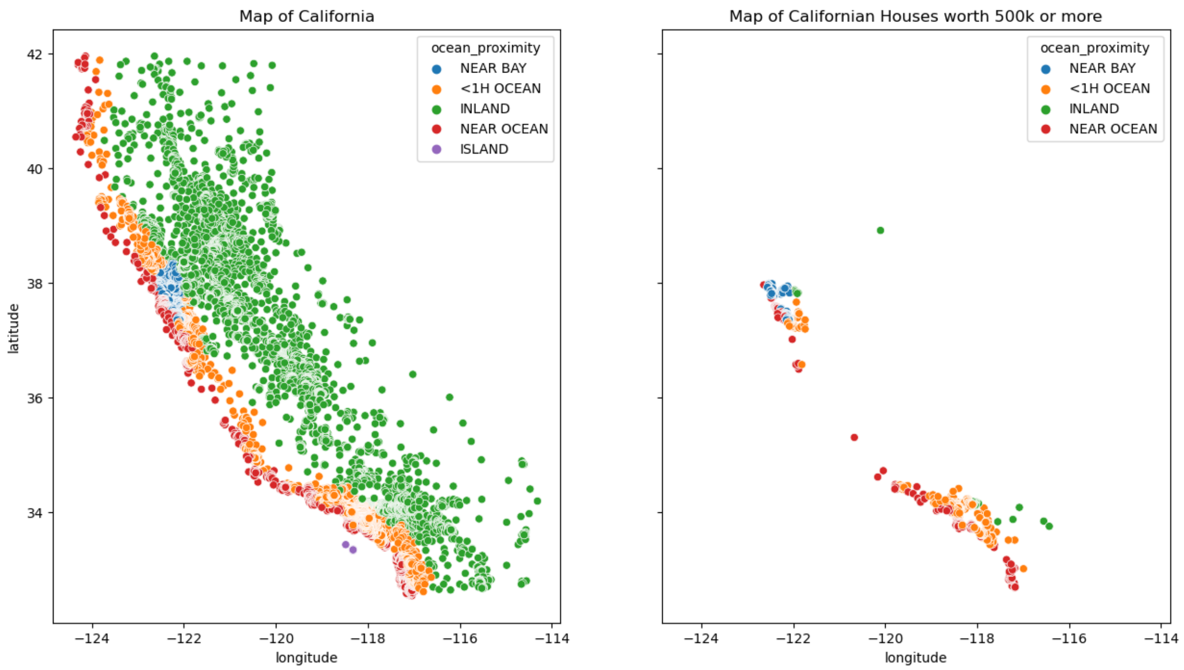
20396 rows × 19 columns

Visualising Data

- back to top

Map of california in correspondence with house proximity

In [17]:
```
fig, axes = plt.subplots(1, 2, figsize=(15,8),sharex=True,sharey=True)
sns.scatterplot(ax=axes[0],data=cali_cleaned, x='longitude',
                y='latitude', hue='ocean_proximity').set_title('Map of California'

cali_house_cap = cali_cleaned.loc[(cali_cleaned['median_house_value']>500000)]
sns.scatterplot(ax=axes[1],data=cali_house_cap, x='longitude',
                y='latitude',hue='ocean_proximity').set_title('Map of Californian H
```



Analyse Assigned Research Questions

- back to top

What is the median household income for each no. of people in household?

- back to top

In [28]:
```
#correlation between median income and household average
cali_cleaned["median_income"].corr(cali_cleaned["household_average"])
```

Out[28]:
```
-0.05714589068503206
```

In [29]:
```
#finding how many people have each number of people per household
cali_cleaned['household_average'].value_counts()
```

Out[29]:
```
3.0    10765
2.0     5846
4.0     2902
5.0      581
1.0      150
6.0      105
7.0       15
8.0       15
Name: household_average, dtype: int64
```

In [41]:
```
#Average median income for households of 3
df3 = cali_cleaned.loc[cali_cleaned['household_average']==3]
df3["median_income"].mean()*10000
```

Out[41]:
```
42449.11407338598
```

In [42]:
```
#Average median income For households of 2
df2 = cali_cleaned.loc[cali_cleaned['household_average']==2]
df2["median_income"].mean()*10000
```

Out[42]:
```
36542.92011631885
```

In [43]:
```
#Average median income For households of 1
df1 = cali_cleaned.loc[cali_cleaned['household_average']==1]
df1["median_income"].mean()*10000
```

Out[43]:
```
27058.393333333333
```

In [44]:
```
#Average median income For households of 4
df4 = cali_cleaned.loc[cali_cleaned['household_average']==4]
df4["median_income"].mean()*10000
```

Out[44]:
```
32275.57753273605
```

In [45]:
```
#Average median income For households of 5
df5 = cali_cleaned.loc[cali_cleaned['household_average']==5]
df5["median_income"].mean()*10000
```

Out[45]:
```
27988.593803786578
```

In [46]:
```
#Average median income For households of 6
df6 = cali_cleaned.loc[cali_cleaned['household_average']==6]
df6["median_income"].mean()*10000
```

Out[46]:
```
31731.00952380952
```

In [47]:
```
#Average median income For households of 7
df7 = cali_cleaned.loc[cali_cleaned['household_average']==7]
seven["median_income"].mean()*10000
```

Out[47]:
```
35601.53333333333
```

In [48]:
```
#Average median income For households of 8
df8 = cali_cleaned.loc[cali_cleaned['household_average']==8]
df8["median_income"].mean()*10000
```

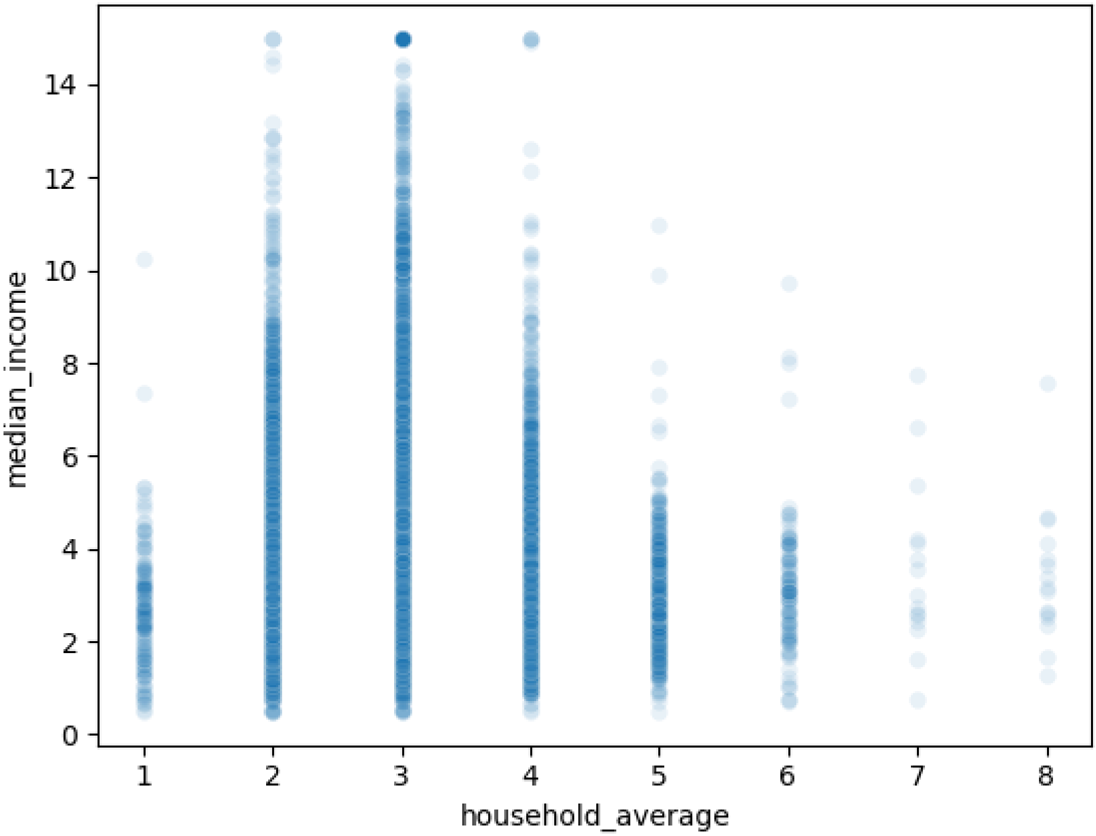
Out[48]:
```
34028.46666666667
```

In [38]:
```
#graphical representation
sns.scatterplot (x="household_average",
```

```
                    y="median_income",
                    data=cali_cleaned,
                    alpha = 0.1)
```

Out[38]:

```
<Axes: xlabel='household_average', ylabel='median_income'>
```



In [4]:

```
#binning the houshold averages
bins = [0, 1, 2, 3, 4, 5, 6, 7, 8]
cali1['Bins'] = pd.cut(cali1['household_average'], bins)
cali1['Bins'].value_counts()
```

Out[4]:

```
(2, 3]    10880
(3, 4]     6191
(1, 2]     1638
(4, 5]     1372
(5, 6]      231
(6, 7]       43
(7, 8]       17
(0, 1]        3
Name: Bins, dtype: int64
```

In [11]:

```
#(2,3] people
two_to_three = cali1.loc[(cali1['household_average']>2) & (cali1['household_averag
two_to_three["median_income"].mean()*10000
```

Out[11]:

```
40605.89577205883
```

In [10]:

```
#(3,4] people
three_to_four = cali1.loc[(cali1['household_average']>3) & (cali1['household_avera
three_to_four["median_income"].mean()*10000
```

Out[10]:

```
39530.190276207395
```

In [12]:

```
#(1,2] people
one_to_two = cali1.loc[(cali1['household_average']>1) & (cali1['household_average'
one_to_two["median_income"].mean()*10000
```

Out[12]: 33088.88888888889

In [13]:
```
#(4,5] people
four_to_five = cali1.loc[(cali1['household_average']>1) & (cali1['household_averag
four_to_five["median_income"].mean()*10000
```

Out[13]: 28643.026967930033

In [14]:
```
#(5,6] people
five_to_six = cali1.loc[(cali1['household_average']>5) & (cali1['household_average
five_to_six["median_income"].mean()*10000
```

Out[14]: 29289.70995670995

In [15]:
```
#(6,7] people
six_to_seven = cali1.loc[(cali1['household_average']>6) & (cali1['household_averag
six_to_seven["median_income"].mean()*10000
```

Out[15]: 31025.813953488367

In [16]:
```
#(7,8] people
seven_to_eight = cali1.loc[(cali1['household_average']>7) & (cali1['household_avera
seven_to_eight["median_income"].mean()*10000
```

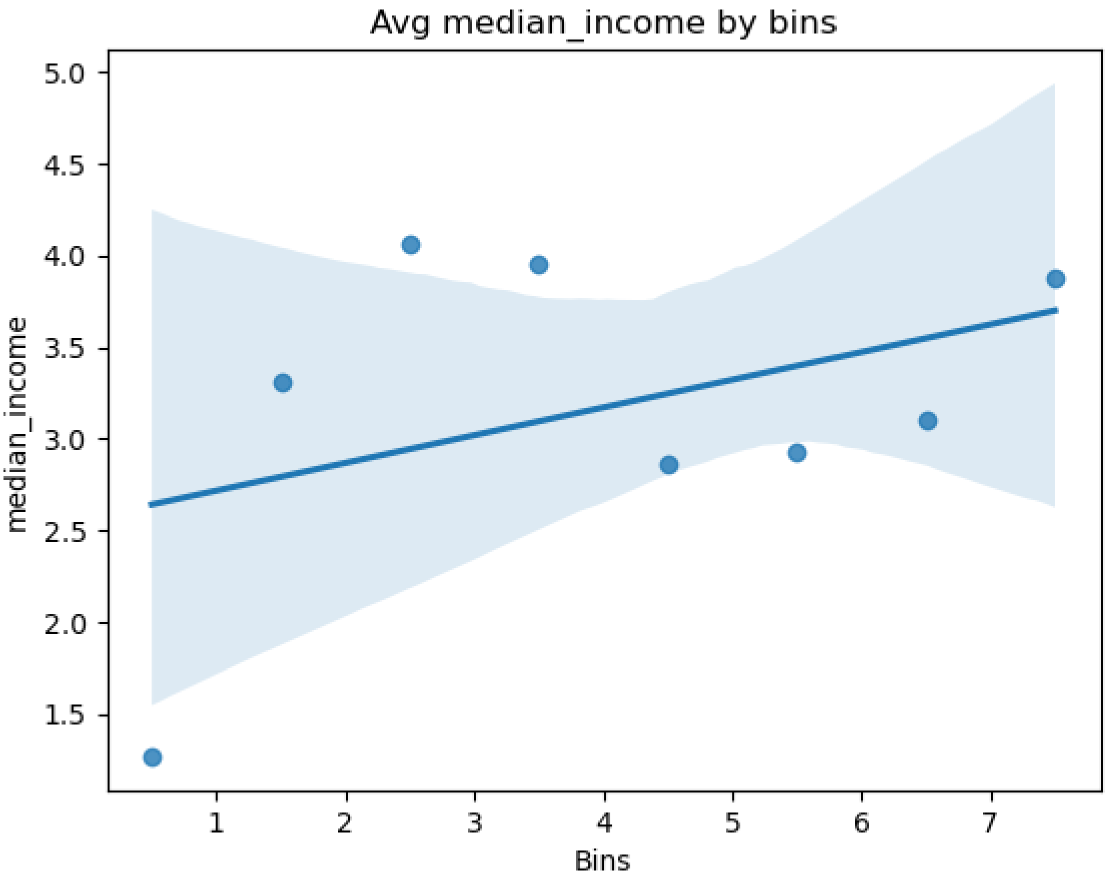
Out[16]: 38716.23529411765

In [17]:
```
#(0,1] people
zero_to_one = cali1.loc[(cali1['household_average']>0) & (cali1['household_average
zero_to_one["median_income"].mean()*10000
```

Out[17]: 12692.666666666668

- 2-4 people households have highest median income (around average income in California): most likely reflecting stable families
- 4-7 people households have below average income in california: most likely reflecting working class families who lack education -> low education leads to less education around reproduction, leading to large families and not high paying jobs
- 1 people households have lowest median income: likely reflecting students.

In [25]:
```
#Graph and regression plot of binned household averages and median income
bins = [0, 1, 2, 3, 4, 5, 6, 7, 8]
cali1['Bins'] = pd.cut(cali1['household_average'], bins)
grouped = cali1.groupby('Bins')['median_income'].mean()
bin_centers = [(bin.left + bin.right) / 2 for bin in grouped.index]

sns.regplot(x=bin_centers, y=grouped)
plt.xlabel('Bins')
plt.ylabel('median_income')
plt.title('Avg median_income by bins')
plt.show()
```



Follows same pattern as the rounded household average and average median income.

Relationship between no. of rooms and house value

- back to top

In [41]:
```
#finding correlation between total rooms and median house value
cali_cleaned["total_rooms"].corr(cali_cleaned["median_house_value"])
```

Out[41]:
```
0.13339889410877884
```

In [44]:
```
#since correlation is very weak, will find average rooms and use them
cali_cleaned['avg_room'] = round(cali_cleaned['total_rooms']/cali_cleaned['househo
cali_cleaned['avg_room'].value_counts()
```

Out[44]:
```
5.0      6643
6.0      5116
4.0      4281
7.0      2023
3.0       995
8.0       722
9.0       190
2.0       151
10.0       61
11.0       40
12.0       18
17.0       16
13.0       12
15.0       11
14.0       10
1.0         9
20.0        8
19.0        8
21.0        7
16.0        5
22.0        5
29.0        5
24.0        5
18.0        4
26.0        4
25.0        4
28.0        3
23.0        3
37.0        3
36.0        3
27.0        2
62.0        2
35.0        2
53.0        2
39.0        1
41.0        1
40.0        1
30.0        1
31.0        1
32.0        1
34.0        1
51.0        1
133.0       1
142.0       1
56.0        1
48.0        1
60.0        1
Name: avg_room, dtype: int64
```

In [45]:
```
#Finding more accurate correlation
cali_cleaned["avg_room"].corr(cali_cleaned["median_house_value"])
```

Out[45]:
```
0.15157485197638526
```

In [36]:
```
#correlation of average room and median house value using data given by houses witl
#to find more accurate correlation
excali = cali_cleaned.loc[cali_cleaned['avg_room']<8]
excali["avg_room"].corr(excali["median_house_value"])
```

Out[36]:
```
0.228717799494547
```

In [38]:
```
#finding training and testing scores of regression model of average rooms less thai
import pandas as pd
```

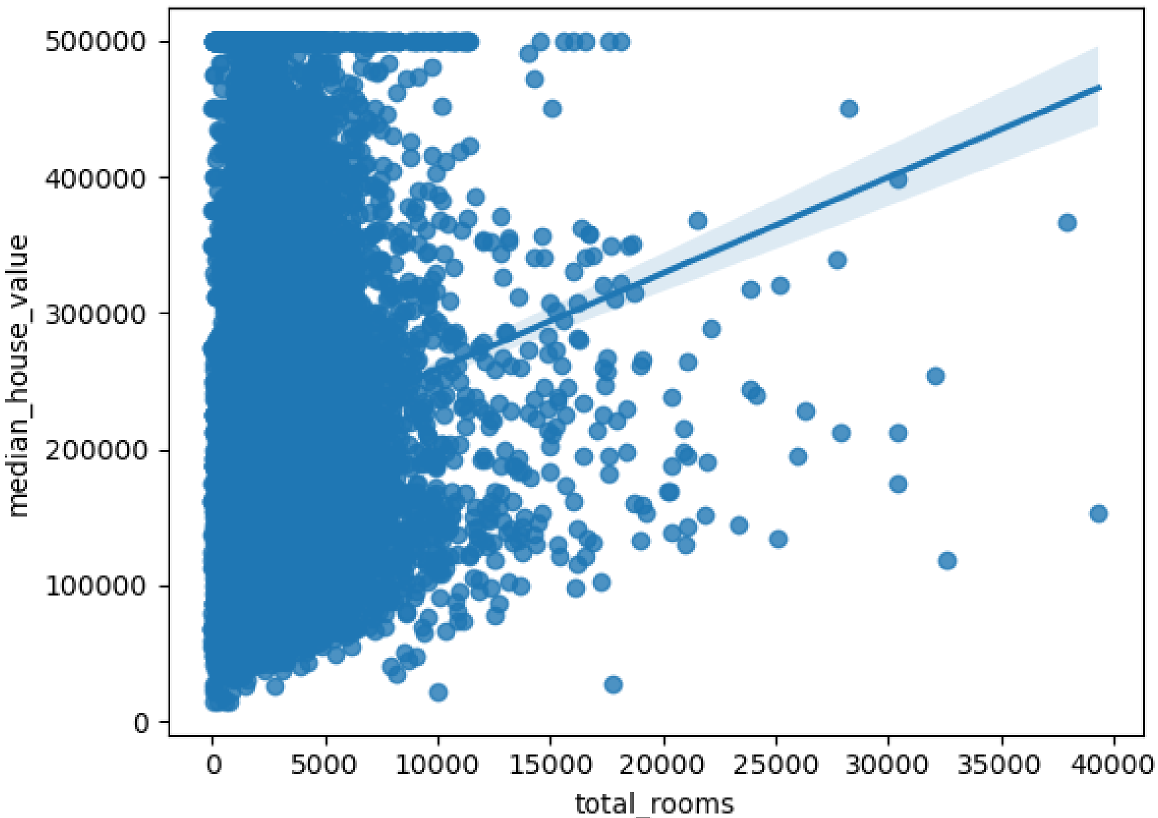
```python
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

X = excali[['avg_room']] #setting explanatory variables
Y = excali['median_house_value']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_st
linear = LinearRegression(fit_intercept=True)
linear.fit(X_train,Y_train)
training_score = linear.score(X, Y) # calculate rsq for the training set
preds_linear = linear.predict(X_test)
rsquared_linear = r2_score(Y_test,preds_linear)

print("Correlation score is",np.round(np.sqrt(training_score), 3))
print("Coefficients are",np.round(linear.coef_, 3))
print("Intercept is",np.round(linear.intercept_,3))
print("Training score is",np.round(training_score, 3))
print("Testing score is",np.round(rsquared_linear, 3))
```

```
Correlation score is 0.229
Coefficients are [22619.115]
Intercept is 84079.691
Training score is 0.052
Testing score is 0.059
```

Below are graphical representations of data used.

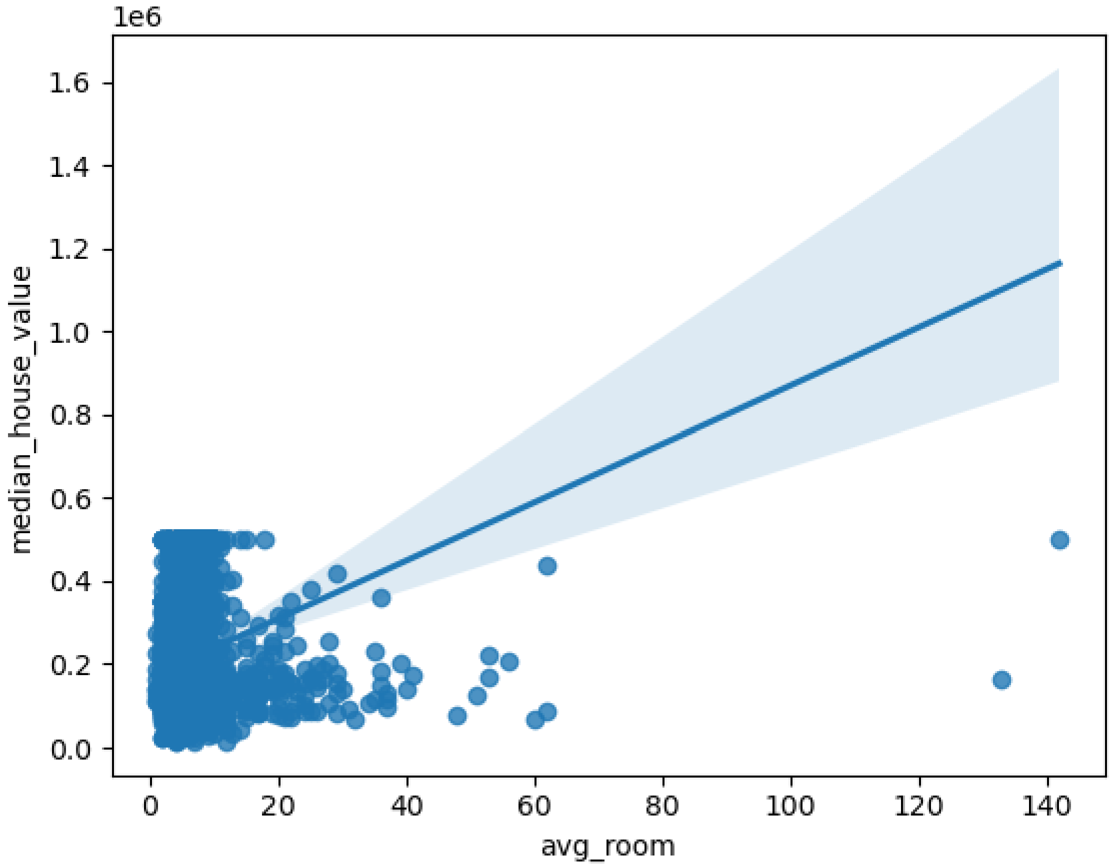
In [29]:
```python
sns.regplot(x="total_rooms",
                    y="median_house_value",
                    data=cali_cleaned)
```

Out[29]: <Axes: xlabel='total_rooms', ylabel='median_house_value'>



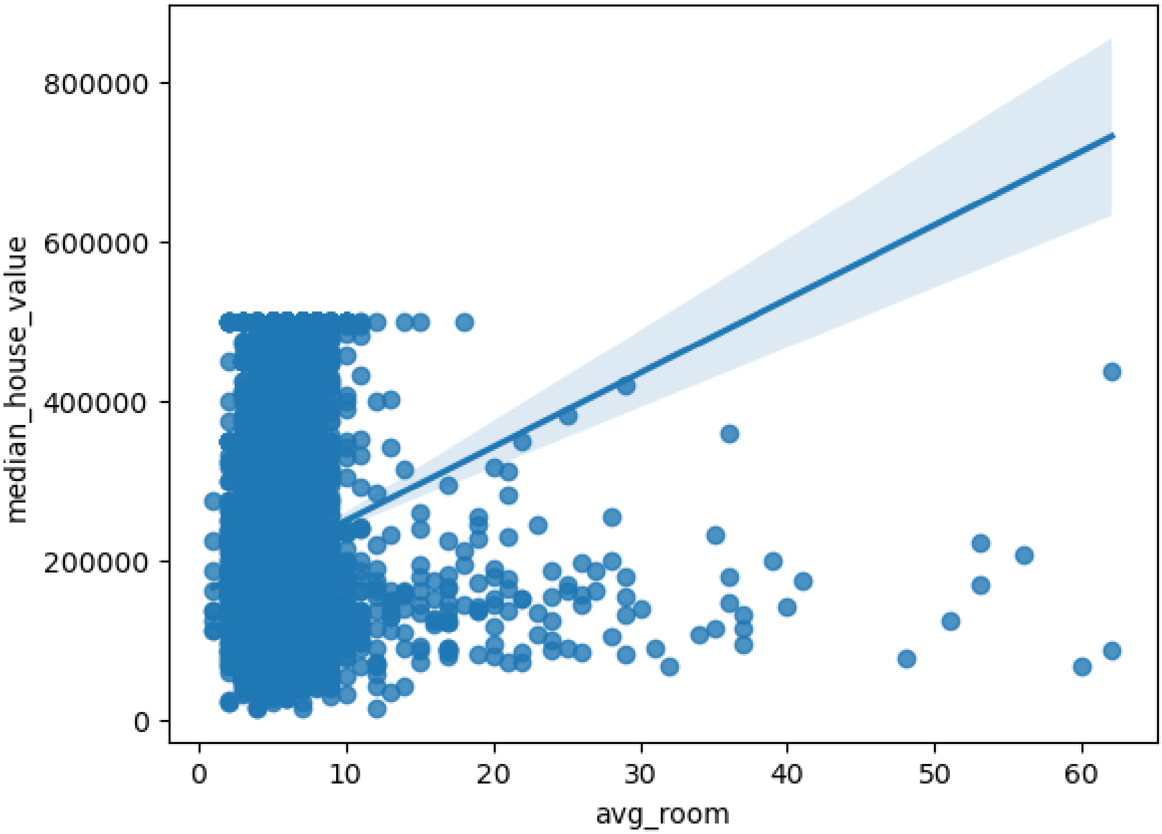
In [34]:
```
sns.regplot(x="avg_room",
            y="median_house_value",
            data=cali_cleaned)
```

Out[34]: <Axes: xlabel='avg_room', ylabel='median_house_value'>



In [35]:
```
sns.regplot(x="avg_room",
            y="median_house_value",
            data=excali)
```

Out[35]: <Axes: xlabel='avg_room', ylabel='median_house_value'>



Analysis of Houses worth 500k or more

- back to top

Subset of dataframe with houses worth 500k or more

In [12]:
```
cali_anal1 = cali_cleaned.loc[cali_cleaned['median_house_value']>=500000]
cali_anal1
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
89	-122.27	37.80	52	249	78.0	396	
459	-122.25	37.87	52	609	236.0	1349	
493	-122.24	37.86	52	1668	225.0	517	
494	-122.24	37.85	52	3726	474.0	1366	
509	-122.23	37.83	52	2990	379.0	947	
...	...	...	...	...	...	...	
20422	-118.90	34.14	35	1503	263.0	576	
20426	-118.69	34.18	11	1177	138.0	415	
20427	-118.80	34.19	4	15572	2222.0	5495	
20436	-118.69	34.21	10	3663	409.0	1179	
20443	-118.85	34.27	50	187	33.0	130	

985 rows × 18 columns

In [13]:
```
#mean, max and min of median house value and median income
average = cali_cleaned.pivot_table (index = 'ocean_proximity', values = ['median_i
                    aggfunc = ['mean', 'max', 'min']).round(2)
average
```

Out[13]:

	mean		max		
	median_house_value	median_income	median_house_value	median_income	medi
ocean_proximity					
<1H OCEAN	240234.94	4.23	500001.0	15.0	
INLAND	124863.96	3.21	500001.0	15.0	
NEAR BAY	259097.08	4.17	500001.0	15.0	
NEAR OCEAN	249288.90	4.01	500001.0	15.0	

In [14]:
```
#creat 3 bins of median income and median house value
median_income_bin = pd.cut (cali_cleaned['median_income'], 3, precision = 2)
median_house_value_bin = pd.cut (cali_cleaned['median_house_value'], 3, precision :
```

In [15]:
```
#number of households and population in each region
cali_cleaned.pivot_table (index = 'ocean_proximity', values = ['households', 'popul
```

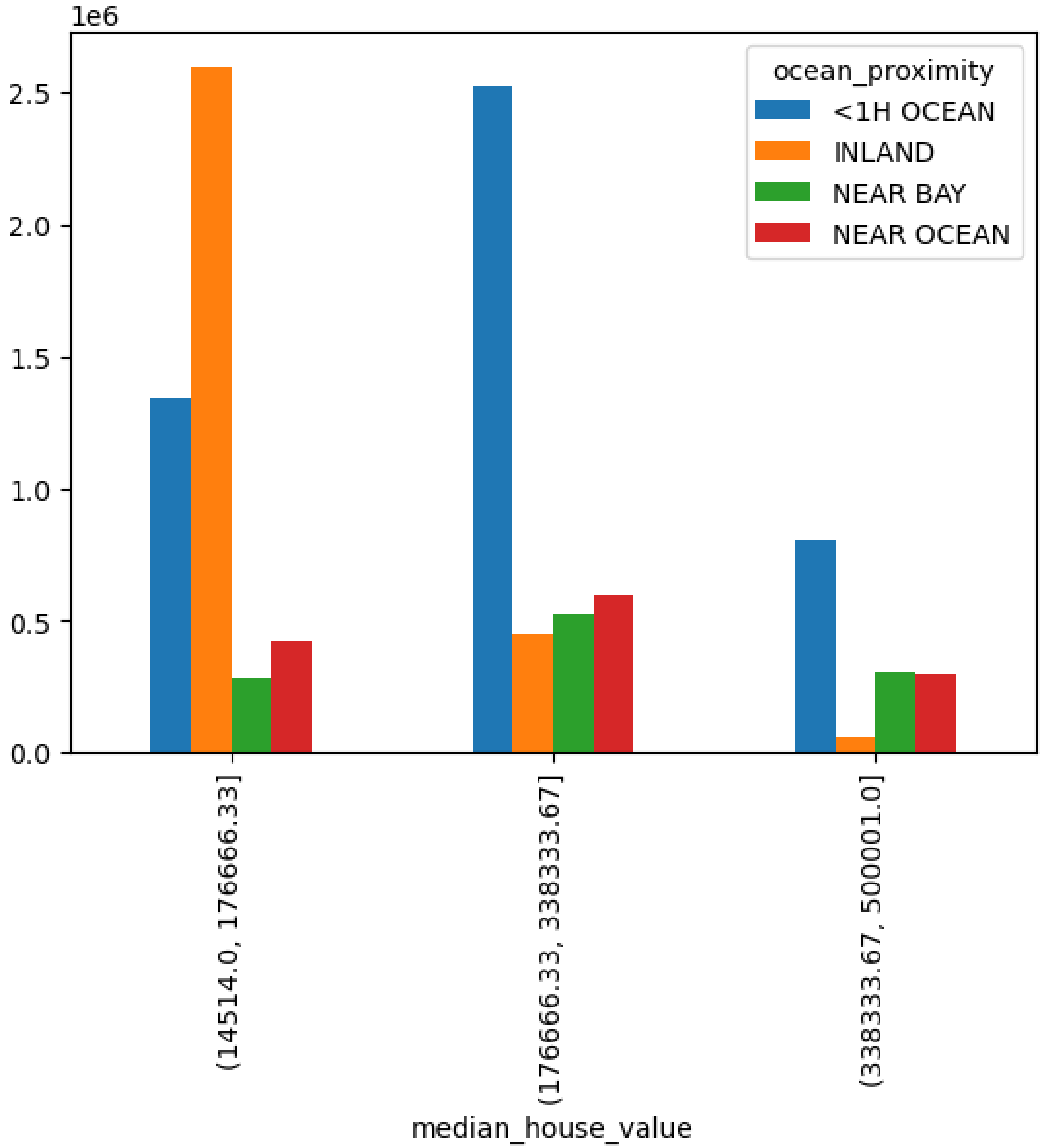
Out[15]:

	households	population
ocean_proximity		
<1H OCEAN	4673888.0	13709960.0
INLAND	3102795.0	8983973.0
NEAR BAY	1105769.0	2783919.0
NEAR OCEAN	1316823.0	3539006.0

In [16]:
```
#number of households living in different house values in each region
group1 = cali_cleaned.groupby([median_house_value_bin,  'ocean_proximity'])['housel
group1.plot.bar()
group1
```

Out[16]:

ocean_proximity	<1H OCEAN	INLAND	NEAR BAY	NEAR OCEAN
median_house_value				
(14514.0, 176666.33]	1344731.0	2597615.0	280408.0	421600.0
(176666.33, 338333.67]	2526024.0	448770.0	523145.0	597487.0
(338333.67, 500001.0]	803133.0	56410.0	302216.0	297736.0

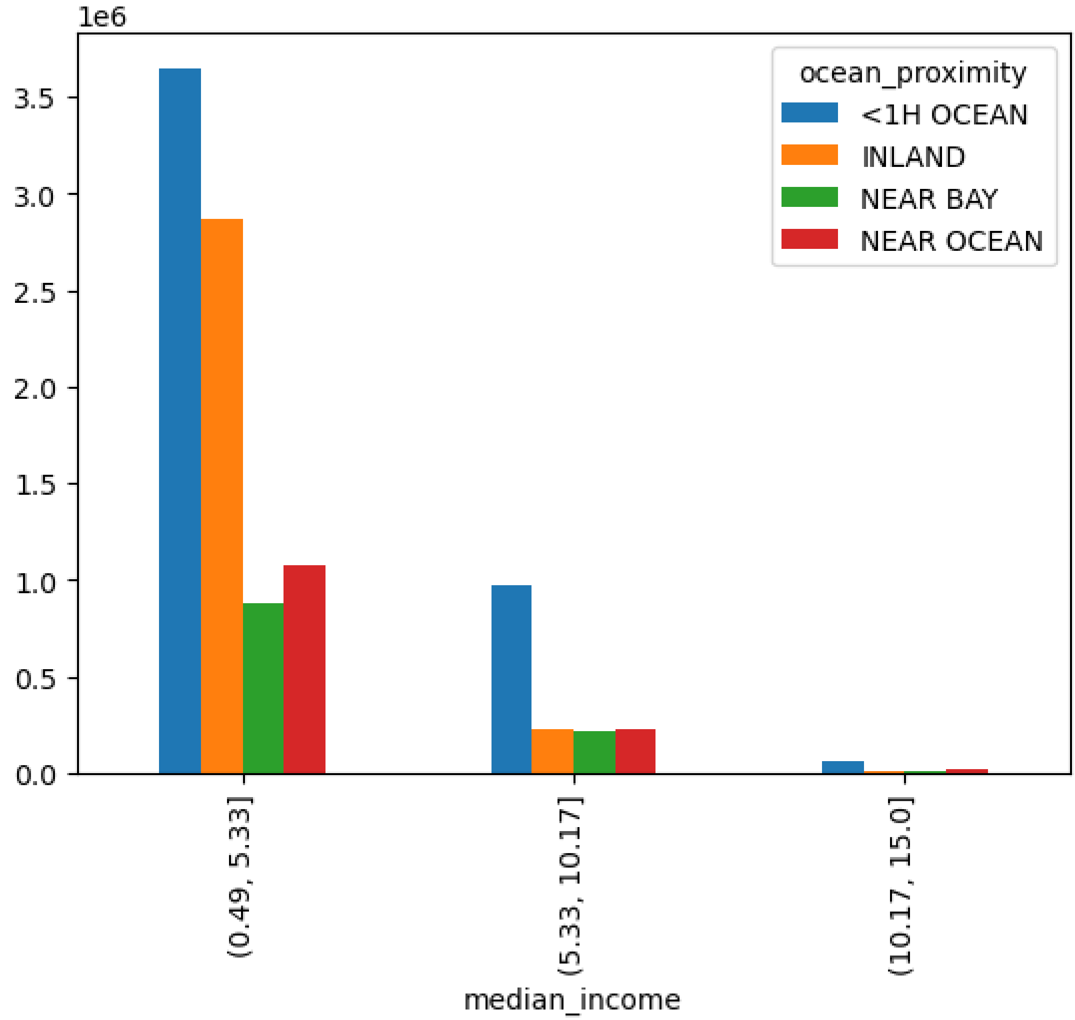


In [17]:

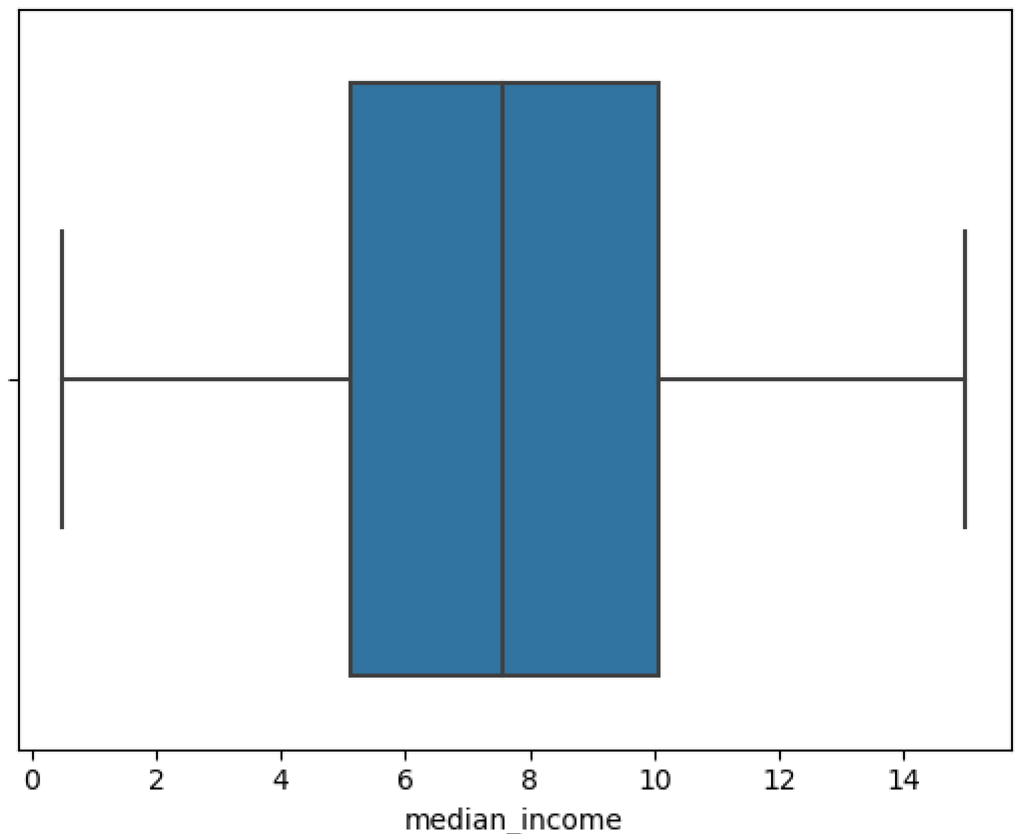
```
#number of households living in each region with different median income
group2 = cali_cleaned.groupby([median_income_bin,  'ocean_proximity'])['households
group2.plot.bar()
group2
```

Out[17]:

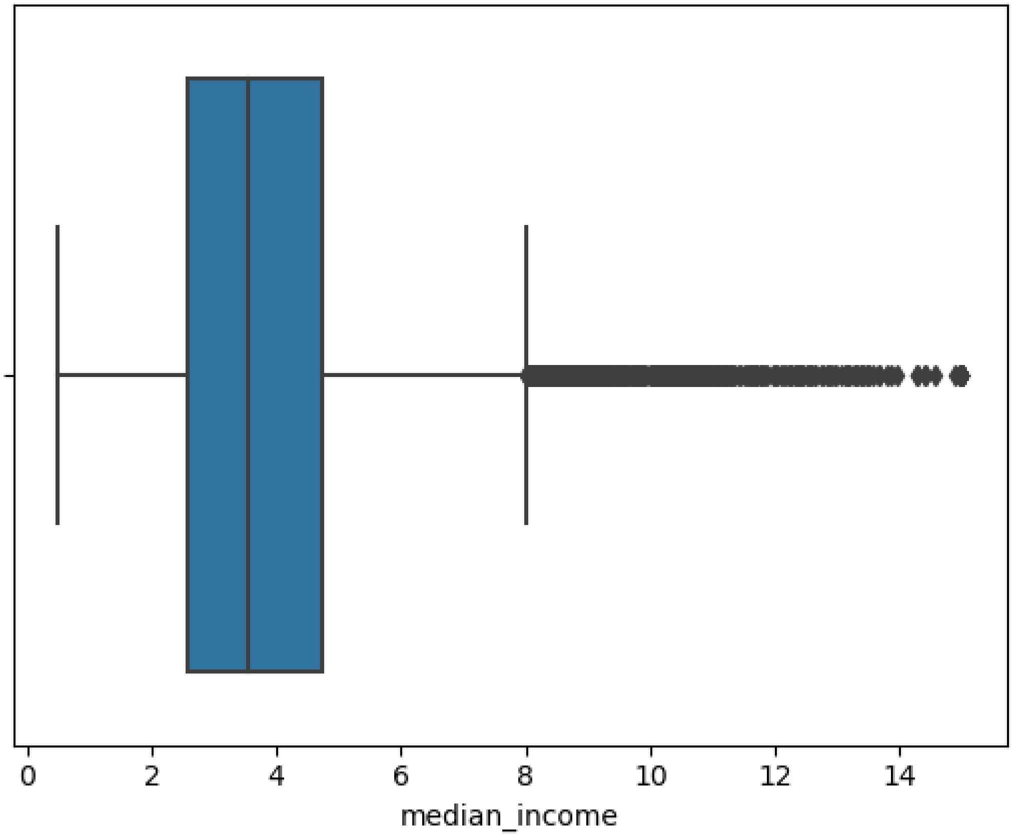
ocean_proximity	<1H OCEAN	INLAND	NEAR BAY	NEAR OCEAN
median_income				
(0.49, 5.33]	3644971.0	2865448.0	875107.0	1075700.0
(5.33, 10.17]	968887.0	232438.0	217529.0	224354.0
(10.17, 15.0]	60030.0	4909.0	13133.0	16769.0



In [18]:
```
sns.boxplot(x=cap.median_income);
```



In [19]:
```
sns.boxplot(x=cali_cleaned.median_income);
```



In [15]:
```
# finding correlation between household_average and median_income
cali_cleaned['household_average'].corr(cali_cleaned['median_income'])
```

Out[15]:
```
-0.05587034863438407
```

Very weak negative relationship between household average and median income

In [16]:
```
#  finding correlation between household_average and median_house_value
cali_cleaned['household_average'].corr(cali_cleaned['median_house_value'])
```

Out[16]:
```
-0.24295460983910497
```

Weak negative relationship between household average and median house value.

There is barely any correlation between household average and the above

In [17]:
```
#  finding correlation between median_income and median_house_value
cali_cleaned['median_income'].corr(cali_cleaned['median_house_value'])
```

Out[17]:
```
0.6895984666143862
```

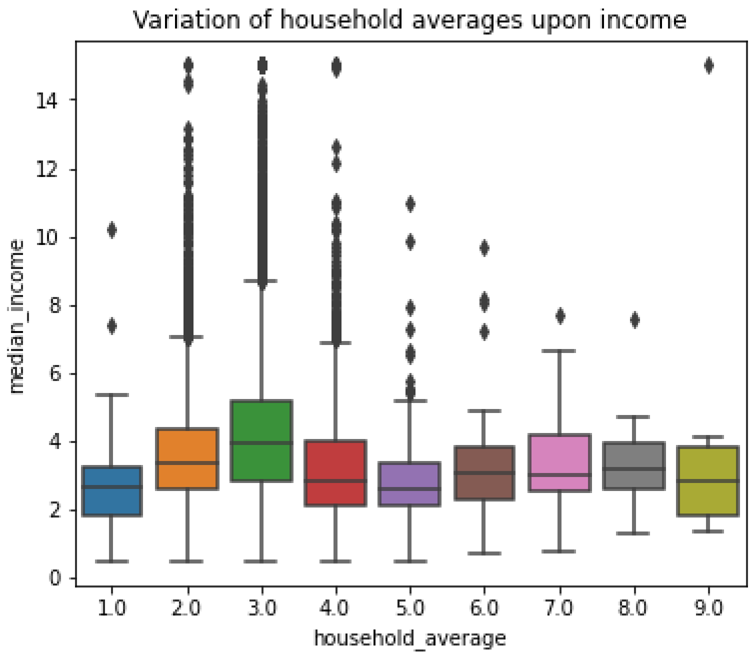
Since median_income and median_house_value have a high correlation, let's analyse the two variables by grouping them into separate household averages and compare.

Moderate positive relationship between median income and median house value

In [18]:
```
plt.rcParams["figure.figsize"] = (6,5)
```

In [45]:
```
# creating a boxplot graph showing the median_income spread and centre across house
sns.boxplot(data=cali_cleaned, x='household_average', y='median_income')
plt.title('Variation of household averages upon income')
```

Out[45]: Text(0.5, 1.0, 'Variation of household averages upon income')



In [46]:
```
# grouping median_income by household averages and finding its median
grouped1 = cali_cleaned.groupby(['household_average'])
grouped1['median_income'].median()
```

Out[46]:
```
household_average
1.0    2.63715
2.0    3.37500
3.0    3.91500
4.0    2.84500
5.0    2.60000
6.0    3.08040
7.0    3.01320
8.0    3.16670
9.0    2.83420
Name: median_income, dtype: float64
```

We can see here that for median house income, the median is highest for households with 3 people per household by 39150 USD, followed by median from 2 people per household of 33750 USD and then household average of 8 with a median of 31667 USD. The lowest median is a household average of 5 with median income 26000 USD followed by household aervage of 1 with income 26371.50 USD and then household average of 9 with 28342 USD.

In [47]:
```
# finding median_income median for each household average
grouped1['median_income'].mean()
```

Out[47]:

```
household_average
1.0    2.705839
2.0    3.654292
3.0    4.244911
4.0    3.227558
5.0    2.798859
6.0    3.173101
7.0    3.560153
8.0    3.402847
9.0    4.361043
Name: median_income, dtype: float64
```

If we were to compare median to average, we can see that household average 9 has the highest average income of 43610 USD followed by household average 3 with 42449 USD then household average 2 of 36542 USD. This is because the average median income of houhsehold averages 9,3 and 2 are stretched out by the median income cap of 150000 USD.

In [48]:

```
# summary of median_income descriptions for each household average
grouped1['median_income'].describe()
```

Out[48]:

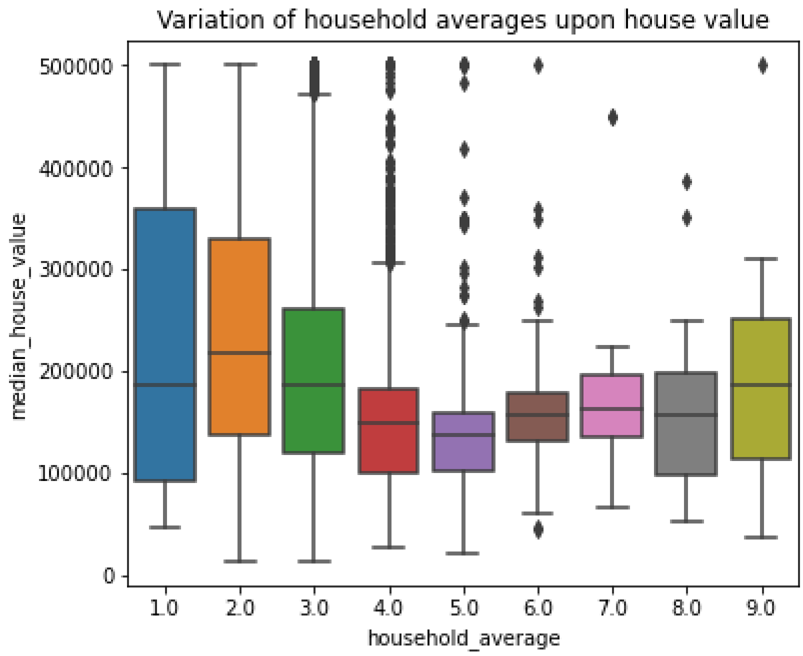
household_average	count	mean	std	min	25%	50%	75%	max
1.0	150.0	2.705839	1.302792	0.4999	1.811725	2.63715	3.254575	10.2264
2.0	5846.0	3.654292	1.635944	0.4999	2.561050	3.37500	4.349550	15.0001
3.0	10765.0	4.244911	2.041487	0.4999	2.844500	3.91500	5.184200	15.0001
4.0	2902.0	3.227558	1.610982	0.4999	2.098525	2.84500	4.014725	15.0001
5.0	581.0	2.798859	1.104827	0.4999	2.107800	2.60000	3.375000	10.9704
6.0	105.0	3.173101	1.422777	0.7160	2.289100	3.08040	3.791700	9.7066
7.0	15.0	3.560153	1.855751	0.7526	2.509650	3.01320	4.163800	7.7197
8.0	15.0	3.402847	1.511880	1.2863	2.565600	3.16670	3.923600	7.5752
9.0	7.0	4.361043	4.801695	1.3750	1.815250	2.83420	3.843750	15.0001

In [51]:

```
# creating boxplot graph of median_house_value across each household average catego
sns.boxplot(data=cali_cleaned, x='household_average', y='median_house_value')
plt.title('Variation of household averages upon house value')
```

Out[51]:

```
Text(0.5, 1.0, 'Variation of household averages upon house value')
```



In [53]:
```
# finding the average of median_house_value for all the dataset
cali_cleaned['median_house_value'].mean()
```

Out[53]:
```
206886.3656430884
```

In [52]:
```
# finding the median of median_house_value for each household_average category
grouped1['median_house_value'].median()
```

Out[52]:
```
household_average
1.0    187500.0
2.0    218450.0
3.0    186100.0
4.0    149200.0
5.0    137500.0
6.0    157500.0
7.0    162500.0
8.0    156800.0
9.0    187200.0
Name: median_house_value, dtype: float64
```

The house average with the highest median house value is household of 2 people with a value of 218450 USD, followed by household average of 1 with 187500 USD then household of 9 of 187200 USD. The household average of 5 people have the lowest median house value of 137500 USD, followed by household average of 4 with 149200 USD then household average of 8 with 156800.

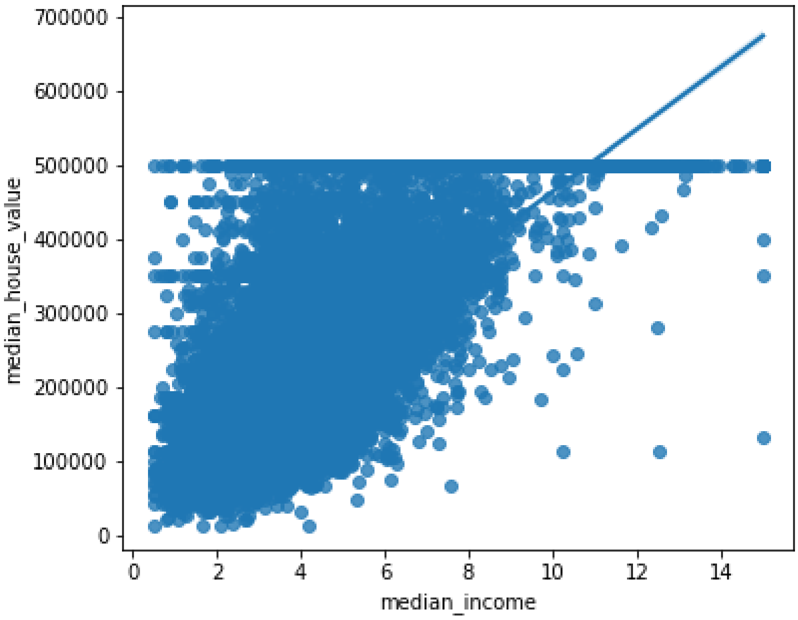
In [25]:
```
# finding the average  for each household_average category
grouped1['median_house_value'].mean()
```

Out[25]:
```
household_average
1.0    240754.153333
2.0    241236.971605
3.0    206382.820437
4.0    153093.080634
5.0    140058.531842
6.0    161584.771429
7.0    189580.000000
8.0    171600.000000
9.0    207957.285714
Name: median_house_value, dtype: float64
```

We can see here that the household average of 2 people have the highest average house value of 241237 USD, followed by household average of 1 with 240754 USD and then household average of 9 wit 207957 USD. The household average of 5 people have the lowest mean house value of 140059 USD, followed nu household average of 4 with 153093 USD then household of 6 with 161585 USD. Household average of 6 is the only household average that has outliers that are outside the lower bound of the data, pulling its mean down and giving us a disproportionate view of the data.

In [54]:
```
# creating a regplot to visualise the linearity between median_income and median_ho
sns.regplot(data=cali_cleaned, x='median_income', y='median_house_value')
```

Out[54]: <AxesSubplot:xlabel='median_income', ylabel='median_house_value'>



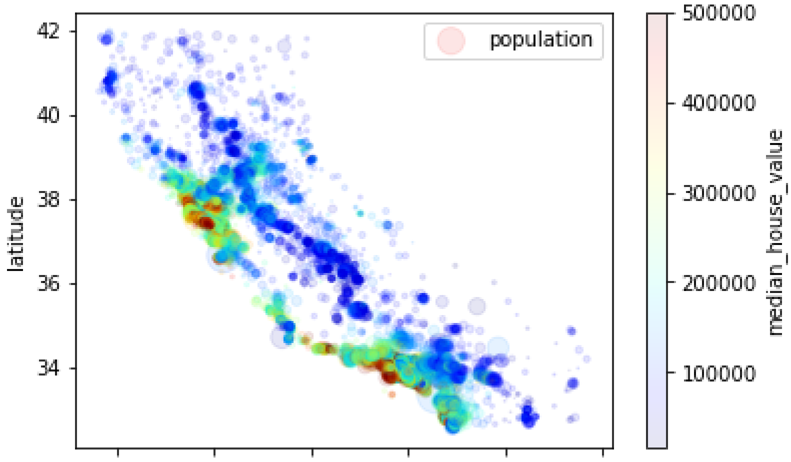
Analysis of location, population and house price

- back to top

correlation between rooms and beds with location, hence house prices

In [161…
```
cali_cleaned.plot(kind = "scatter", x='longitude', y = 'latitude', alpha = 0.1, s
                  ,label = "population", c = 'median_house_value', cmap=plt.get_cma
# map of houses based on their location, colour based on median house value, size o
```

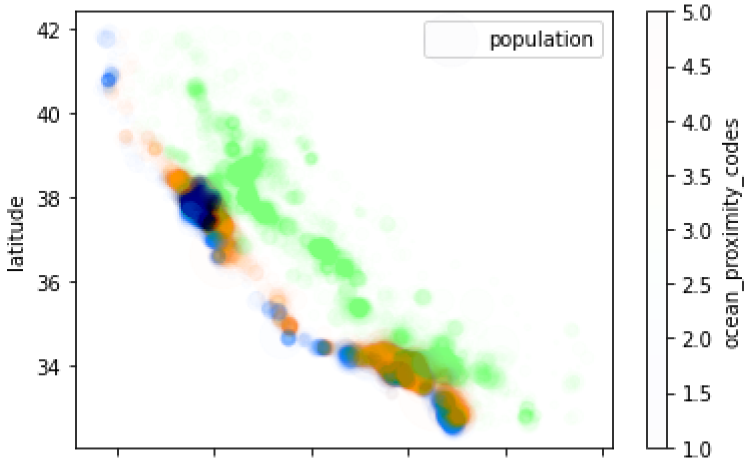
Out[161]: <AxesSubplot:xlabel='longitude', ylabel='latitude'>



The red, yellow, green areas represent higher median house value. As they are mostly around the bay area, and ocean side, it shows that the location is closely related to the house value. Use correlation to support this.

In [180…
```
ocean_proximity_codes = {'NEAR BAY':1, 'NEAR OCEAN':2, 'INLAND':3, '<1H OCEAN':4,
#assinging numerical codes to each ocean proximity category

cali_cleaned['ocean_proximity_codes']=cali_cleaned['ocean_proximity'].map(ocean_pro
# adding the codes to the data set

cali_cleaned.plot(kind = "scatter", x='longitude', y = 'latitude', alpha = 0.01, s
                  ,label = "population", c = 'ocean_proximity_codes' , cmap = 'jet'
# plotting location of blocks with population as the scale and colour coded by ocea
```

Out[180]: <AxesSubplot:xlabel='longitude', ylabel='latitude'>



In [155…
```
rooms = cali_cleaned[['total_rooms', 'total_bedrooms', 'median_house_value','<1H OC
             'ISLAND', 'NEAR BAY', 'NEAR OCEAN']].corr()
rooms[['total_rooms', 'total_bedrooms']] # correlations between rooms and median ho
```

Out[155]:

	total_rooms	total_bedrooms
total_rooms	1.000000	0.930245
total_bedrooms	0.930245	1.000000
median_house_value	0.133277	0.049476
<1H OCEAN	-0.004824	0.017281
INLAND	0.027384	-0.005572
ISLAND	-0.007644	-0.004400
NEAR BAY	-0.023529	-0.019744
NEAR OCEAN	-0.008482	0.000846

Total bedrooms pos correlated to near ocean, meaning in near ocean areas, there are more total bedrooms Total rooms neg correlated to almost all, however least negative is

correlation between population and median house value

In [51]:
```
population_value_correlation = cali_cleaned['population'].corr(cali_cleaned['mediar
# calculating correlation between population and median house value
print('correlation between population and median house value:', population_value_c
```

```
correlation between population and median house value: -0.0246
```

In [56]:
```
ocean_proximity = cali_cleaned[['population', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEA
correlations = ocean_proximity.corr()
# correlation between median house value and the different ocean proximity categor

correlations['median_house_value'].sort_values(ascending = False)
```

Out[56]:
```
median_house_value    1.000000
<1H OCEAN             0.257809
NEAR BAY              0.160076
NEAR OCEAN            0.141283
ISLAND                0.023552
population           -0.024620
INLAND               -0.485493
Name: median_house_value, dtype: float64
```

In [75]:
```
cali_cleaned[['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN']].sum()
# number of blocks within each ocean proximity
```

Out[75]:
```
<1H OCEAN     9024
INLAND        6472
ISLAND           5
NEAR BAY      2264
NEAR OCEAN    2619
dtype: int64
```

calculate: ratio of population to average house price (below calc / population of each ocean prox category) calculate: average house price by ocean proximity (add up house prices for each category/sumoceanproximity^) calculate: population of each ocean prox. add all populations up for each category.

Houses which are less than 1 hour away from the ocean have the strongest relationship with median house price.

Population is negatively related to house price, meaning that on average, an increase in population will lead to a decrease in median house value.

This may be the cause of houses less than an hour from ocean having the strongest relationship with value, because they would be less populated than areas that are 'near bay' or 'near ocean'. Also, '<1H ocean' is relatively close to the ocean showing that location also relates.

SO CONCLUSION,

- population neg correlated to house value
- houses close to bay are positively correlated to house value
- houses near the ocean are more densely populated
- This results in '<H OCEAN' category being worth the most.
    - it is the 3rd closest to the ocean
    - it is the most populated

In [87]:
```
cali_cleaned.pivot_table(index = 'ocean_proximity', values =['median_house_value',
                         aggfunc = ['sum','count','mean', 'max', 'min'])
# pivot table showing sum, count, mean, max, min of ocean proximity, median house v
```

Out[87]:

	sum		count		
	median_house_value	population	median_house_value	population	median_house
ocean_proximity					
<1H OCEAN	2.168543e+09	13698753.0	9024	9024	240308.4
INLAND	8.076302e+08	8977390.0	6472	6472	124788.3
ISLAND	1.902200e+06	3340.0	5	5	380440.0
NEAR BAY	5.868015e+08	2779678.0	2264	2264	259187.9
NEAR OCEAN	6.531546e+08	3531847.0	2619	2619	249390.8

In [97]:
```
population_houseprice = 240308.447141/9024
print(population_houseprice)
```

```
26.629925436724292
```

In [ ]:

Is there a relationship between median household value and age of house?

- back to top

Comments

- Very weak relationship between house value and age, largest contributor appears to be location

- Even with transforming house value and age of house variables, there is still weak correlation coefficients between them, and thus a regression model based on these 2 variables alone would be very inaccurate

In [13]:
```
cali_cleaned
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20396 rows × 19 columns

In [14]:
```
cali_res = cali_cleaned.drop(columns=['longitude','latitude','total_rooms','total_l
                                      'population','households','median_income',
                                      'household_density','ocean_proximity','incom

cali_res
```

Out[14]:

	housing_median_age	median_house_value	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN	hou
0	41.0	452600.0	0	0	0	1	0	
1	21.0	358500.0	0	0	0	1	0	
2	52.0	352100.0	0	0	0	1	0	
3	52.0	341300.0	0	0	0	1	0	
4	52.0	342200.0	0	0	0	1	0	
...	...	...	...	...	...	...	...	
20635	25.0	78100.0	0	1	0	0	0	
20636	18.0	77100.0	0	1	0	0	0	
20637	17.0	92300.0	0	1	0	0	0	
20638	18.0	84700.0	0	1	0	0	0	
20639	16.0	89400.0	0	1	0	0	0	

20396 rows × 9 columns

After doing squared and z-score, cubing variables seems to make them the most symmetric

In [22]:
```
cali_res['house_age_cube'] = (cali_res['housing_age_standard'])**3
cali_res['house_value_cube'] = (cali_res['house_value_standard'])**3
cali_res
```

Out[22]:

	housing_median_age	median_house_value	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN	hou
0	41.0	452600.0	0	0	0	1	0	
1	21.0	358500.0	0	0	0	1	0	
2	52.0	352100.0	0	0	0	1	0	
3	52.0	341300.0	0	0	0	1	0	
4	52.0	342200.0	0	0	0	1	0	
...	...	...	...	...	...	...	...	
20635	25.0	78100.0	0	1	0	0	0	
20636	18.0	77100.0	0	1	0	0	0	
20637	17.0	92300.0	0	1	0	0	0	
20638	18.0	84700.0	0	1	0	0	0	
20639	16.0	89400.0	0	1	0	0	0	

20396 rows × 11 columns

In [23]:
```
cali_res.house_value_standard.round(0).unique()
```

Out[23]:
```
array([ 2.,  1.,  0., -1.,  3., -2.])
```

In [24]:
```
cali_res.housing_age_standard.round(0).unique()
```

Out[24]:
```
array([ 1., -1.,  2., -2., -0.])
```

In [25]:
```
cali_res.corr()
```

Out[25]:

	housing_median_age	median_house_value	<1H OCEAN	INLAND	ISLAND
housing_median_age	1.000000	0.106004	0.045436	-0.237314	0.017130
median_house_value	0.106004	1.000000	0.257513	-0.484928	0.023546
<1H OCEAN	0.045436	0.257513	1.000000	-0.607898	-0.013956
INLAND	-0.237314	-0.484928	-0.607898	1.000000	-0.010681
ISLAND	0.017130	0.023546	-0.013956	-0.010681	1.000000
NEAR BAY	0.256045	0.159999	-0.315007	-0.241078	-0.005535
NEAR OCEAN	0.021476	0.141014	-0.342236	-0.261916	-0.006013
house_value_standard	0.106004	1.000000	0.257513	-0.484928	0.023546
housing_age_standard	1.000000	0.106004	0.045436	-0.237314	0.017130
house_age_cube	0.880483	0.113993	-0.003711	-0.199621	0.023681
house_value_cube	0.113765	0.828822	0.121106	-0.280546	0.016008

In [26]:
```
cali_res.describe()
```

Out[26]:

	housing_median_age	median_house_value	<1H OCEAN	INLAND	ISLAND	NE
count	20396.000000	20396.000000	20396.000000	20396.000000	20396.000000	20396
mean	28.630663	206910.077956	0.442685	0.317513	0.000245	0
std	12.587017	115405.251320	0.496716	0.465520	0.015656	0
min	1.000000	14999.000000	0.000000	0.000000	0.000000	0
25%	18.000000	119500.000000	0.000000	0.000000	0.000000	0
50%	29.000000	179800.000000	0.000000	0.000000	0.000000	0
75%	37.000000	264900.000000	1.000000	1.000000	0.000000	0
max	52.000000	500001.000000	1.000000	1.000000	1.000000	1

In [27]:
```
cali_res2 = cali_res.drop(columns=['house_value_standard','housing_age_standard'])
cali_res2
```

Out[27]:

	housing_median_age	median_house_value	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN	hou
0	41.0	452600.0	0	0	0	1	0	
1	21.0	358500.0	0	0	0	1	0	
2	52.0	352100.0	0	0	0	1	0	
3	52.0	341300.0	0	0	0	1	0	
4	52.0	342200.0	0	0	0	1	0	
...	...	...	...	...	...	...	...	
20635	25.0	78100.0	0	1	0	0	0	
20636	18.0	77100.0	0	1	0	0	0	
20637	17.0	92300.0	0	1	0	0	0	
20638	18.0	84700.0	0	1	0	0	0	
20639	16.0	89400.0	0	1	0	0	0	

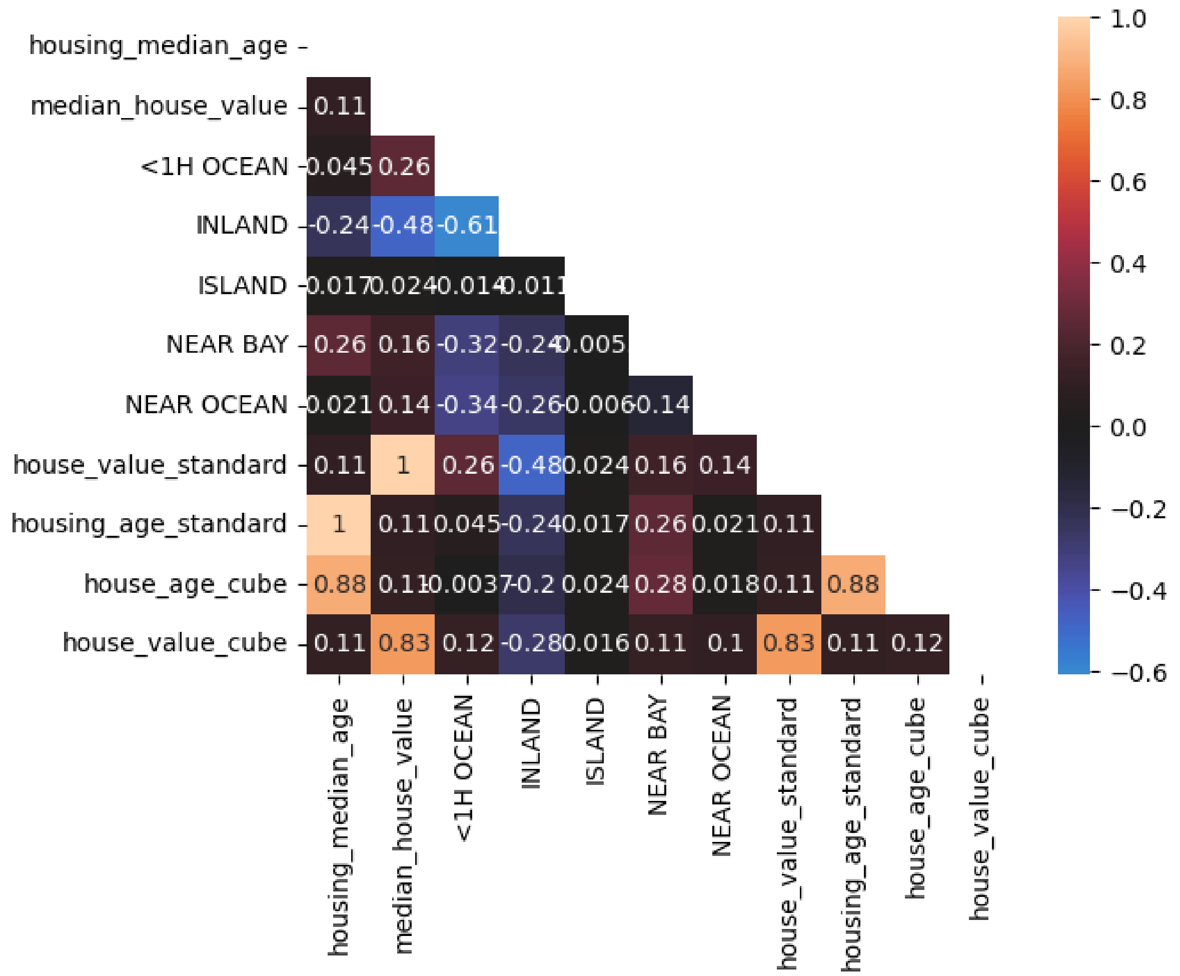
20396 rows × 9 columns

In [31]:

```python
sns.heatmap(cali_res.corr(),annot=True,center=0,
            mask=np.triu(np.ones_like(cali_res.corr(), dtype=bool)),
           )
```

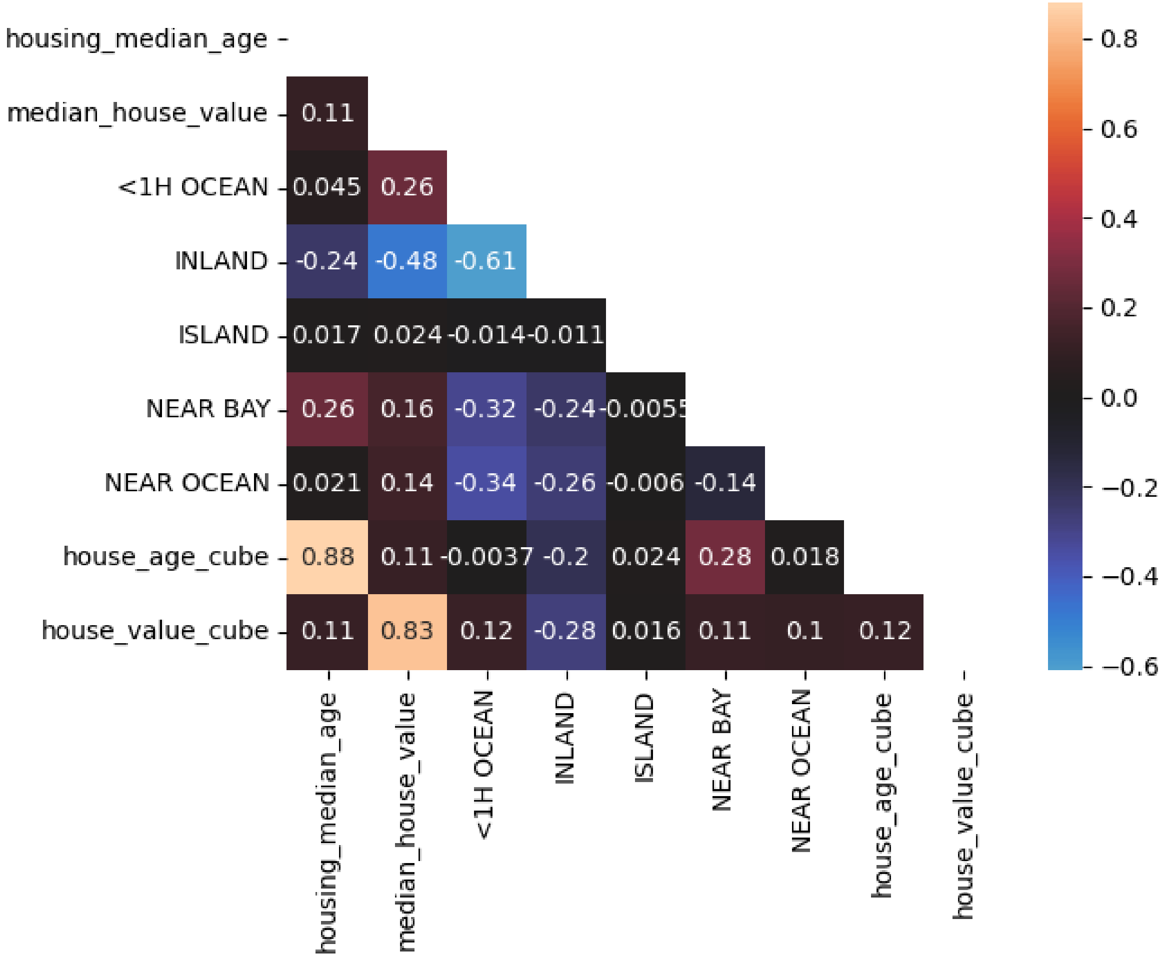
Out[31]:

```
<AxesSubplot:>
```



In [32]:
```
sns.heatmap(cali_res2.corr(),annot=True,center=0,
            mask=np.triu(np.ones_like(cali_res2.corr(), dtype=bool)),
           )
```

Out[32]: <AxesSubplot:>



Check equation of line of best fit for variables to see if regression would be useful

In [33]:
```
from sklearn.linear_model import LinearRegression # linear regression model
from sklearn.model_selection import train_test_split # for splitting the data into
from sklearn.metrics import r2_score # for comparing the predicted and test values
```

In [34]:
```
print(np.poly1d(np.polyfit(cali_res['house_value_standard'], cali_res['housing_age_
```

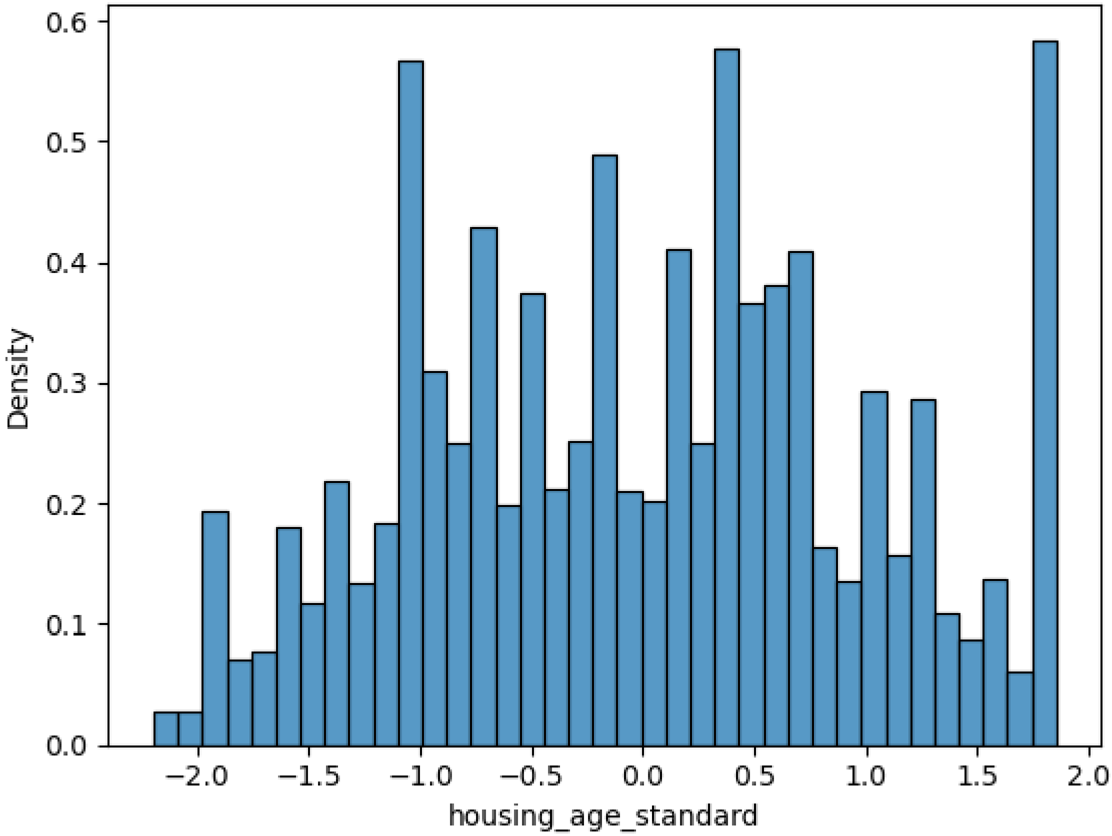
```
0.106 x - 0.000235
```

In [35]:
```
print(np.poly1d(np.polyfit(cali_res2['house_value_cube'], cali_res2['house_age_cub
```

```
0.07207 x - 0.009912
```

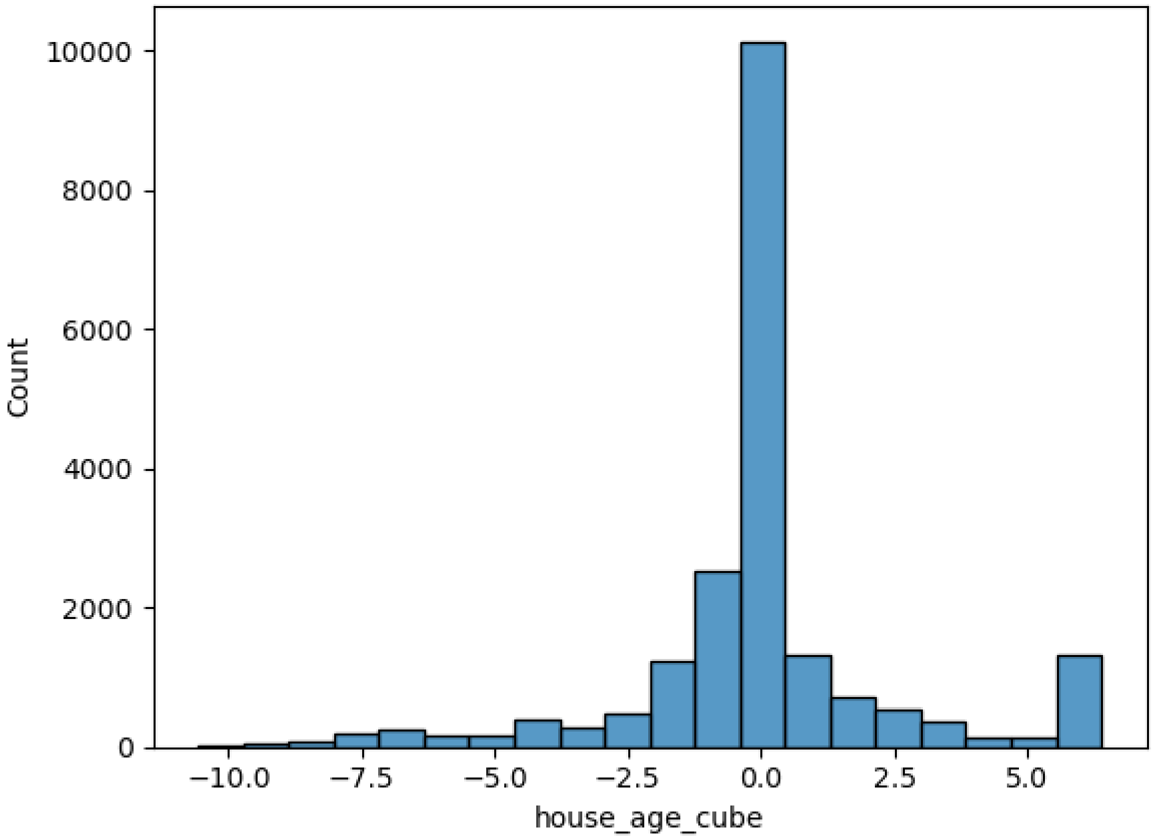
In [36]:
```
sns.histplot(data=cali_res,x='housing_age_standard',stat='density')
```

Out[36]: <AxesSubplot:xlabel='housing_age_standard', ylabel='Density'>



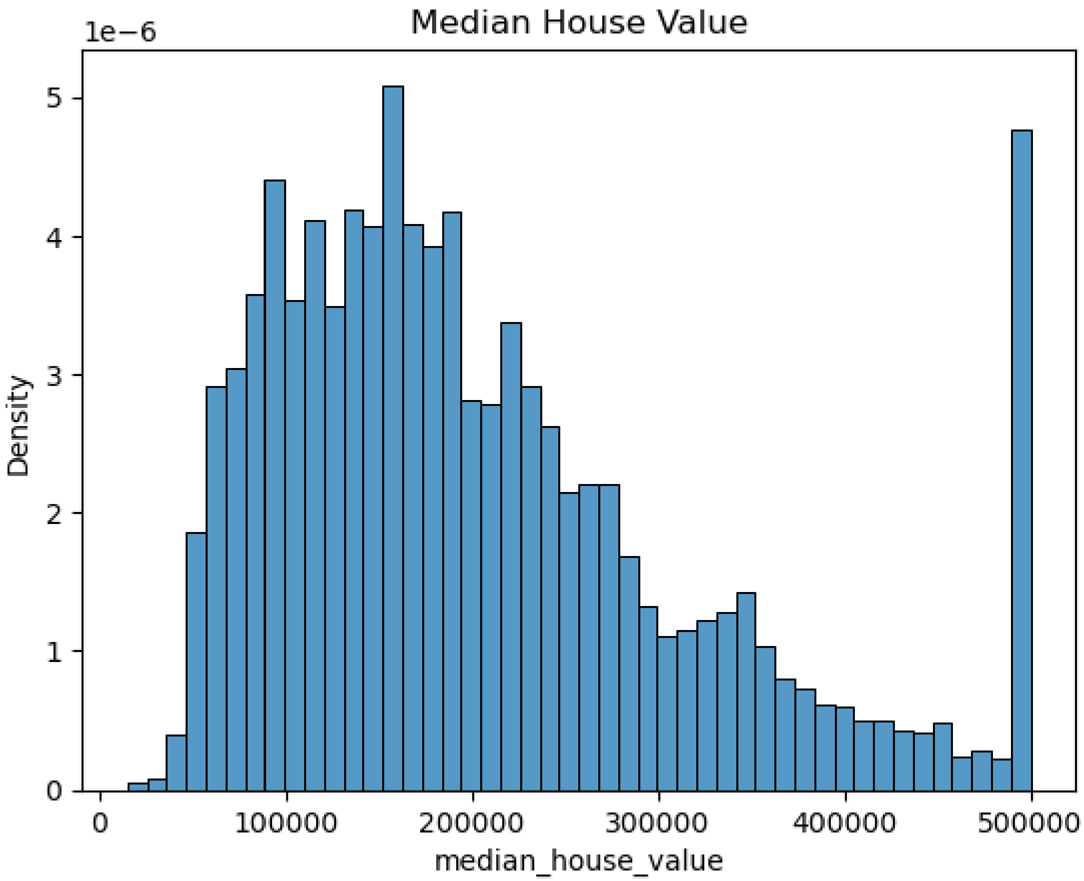
In [37]:
```
sns.histplot(data=cali_res,x='house_age_cube',bins=20)
```

Out[37]: <AxesSubplot:xlabel='house_age_cube', ylabel='Count'>



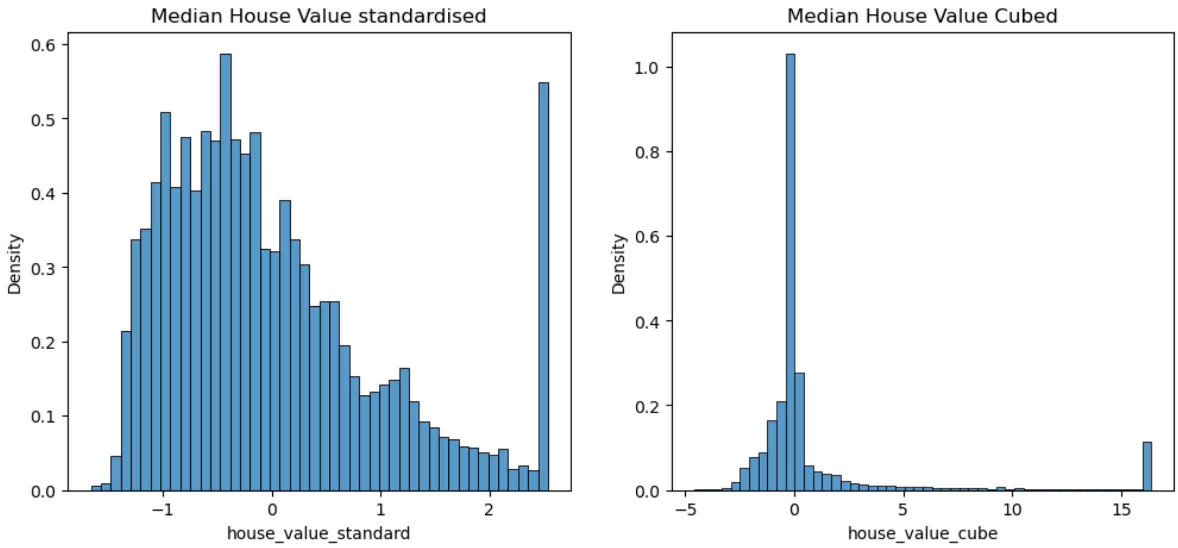
In [28]:
```
sns.histplot(data=cali_res,
             x='median_house_value',stat='density').set_title('Median House Value'
```

Out[28]: Text(0.5, 1.0, 'Median House Value')



In [29]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))

sns.histplot(ax=axes[0],data=cali_res,
             x='house_value_standard',stat='density').set_title('Median House Valu

sns.histplot(ax=axes[1],data=cali_res,
             x='house_value_cube',stat='density',bins=50).set_title('Median House \
```

Out[29]:
```
Text(0.5, 1.0, 'Median House Value Cubed')
```



What relationship is there between house income and age of house?

- back to top

Comments

- Very weak correlation coefficient between house income and house age, but strong correlation between house income and house value as well as location, and house age and house location

In [43]:
```
cali_cleaned.head()
```

Out[43]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [44]:
```
cali_cleaned.corr()
```

Out[44]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	pc
longitude	1.000000	-0.924725	-0.109084	0.045241	0.069382	
latitude	-0.924725	1.000000	0.011443	-0.036322	-0.066648	-
housing_median_age	-0.109084	0.011443	1.000000	-0.360947	-0.320705	-
total_rooms	0.045241	-0.036322	-0.360947	1.000000	0.930262	
total_bedrooms	0.069382	-0.066648	-0.320705	0.930262	1.000000	
population	0.101036	-0.110199	-0.297253	0.863409	0.884084	
households	0.056286	-0.071445	-0.303005	0.918857	0.979693	
median_income	-0.014646	-0.080756	-0.119844	0.198718	-0.007412	
median_house_value	-0.045229	-0.144831	0.106004	0.133298	0.049514	-
household_density	0.169066	-0.163576	-0.003976	-0.106843	-0.144799	
<1H OCEAN	0.321011	-0.447129	0.045436	-0.004787	0.017283	
INLAND	-0.055660	0.351226	-0.237314	0.027300	-0.005662	-
ISLAND	0.009506	-0.016672	0.017130	-0.007636	-0.004393	-
NEAR BAY	-0.474707	0.358862	0.256045	-0.023481	-0.019665	-
NEAR OCEAN	0.046212	-0.161063	0.021476	-0.008466	0.000893	-
house_value_standard	-0.045229	-0.144831	0.106004	0.133298	0.049514	-
housing_age_standard	-0.109084	0.011443	1.000000	-0.360947	-0.320705	-
income_standard	-0.014646	-0.080756	-0.119844	0.198718	-0.007412	

Removing unused variables to focus on median income and house age

In [15]:

```
cali_q2 = cali_cleaned.drop(columns=['longitude','latitude','total_rooms','total_be
                                     'population','households','ocean_proximity']
cali_q2.head()
```

Out[15]:

	housing_median_age	median_income	median_house_value	household_density	<1H OCEAN	INLAND
0	41.0	8.3252	452600.0	2.56	0	(
1	21.0	8.3014	358500.0	2.11	0	(
2	52.0	7.2574	352100.0	2.80	0	(
3	52.0	5.6431	341300.0	2.55	0	(
4	52.0	3.8462	342200.0	2.18	0	(

In [46]:

```
cali_q2.corr()
```

Out[46]:

	housing_median_age	median_income	median_house_value	household_densi
housing_median_age	1.000000	-0.119844	0.106004	-0.0039
median_income	-0.119844	1.000000	0.689084	-0.0654
median_house_value	0.106004	0.689084	1.000000	-0.2649
household_density	-0.003976	-0.065417	-0.264997	1.0000
<1H OCEAN	0.045436	0.169330	0.257513	0.1439
INLAND	-0.237314	-0.238024	-0.484928	0.0116
ISLAND	0.017130	-0.009303	0.023546	-0.0109
NEAR BAY	0.256045	0.055541	0.159999	-0.1474
NEAR OCEAN	0.021476	0.028061	0.141014	-0.0908
house_value_standard	0.106004	0.689084	1.000000	-0.2649
housing_age_standard	1.000000	-0.119844	0.106004	-0.0039
income_standard	-0.119844	1.000000	0.689084	-0.0654

Check to see if variables are skewed/ have non-symmetric distribution

In [47]:

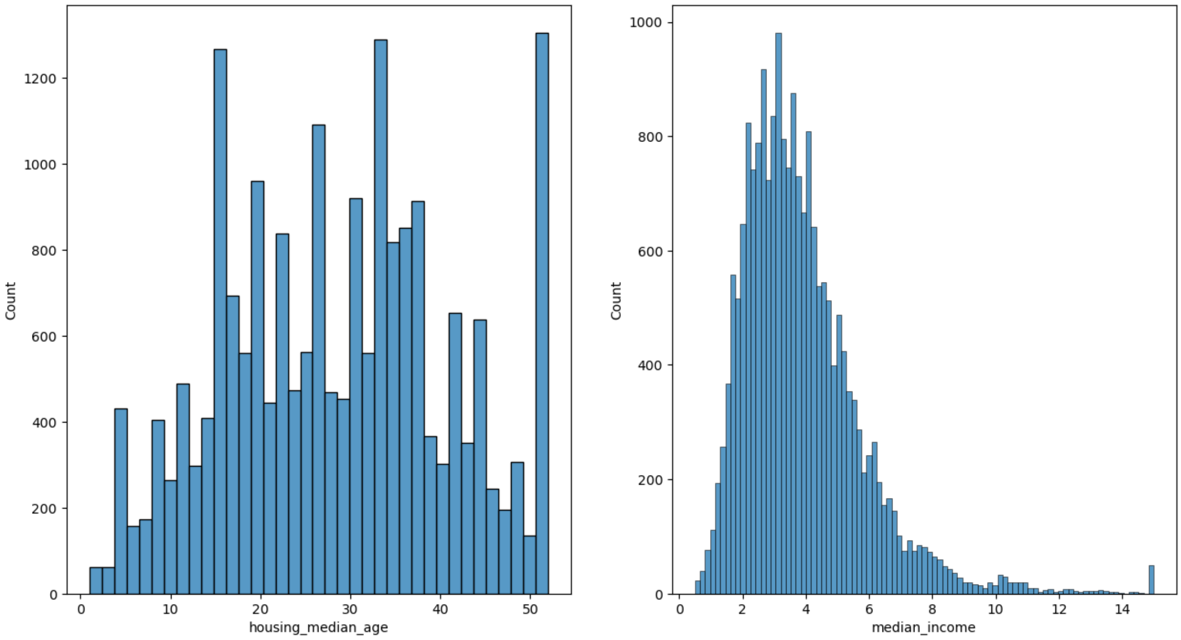
```
plt.figure(figsize=(15,12))

sns.regplot(data=cali_q2, x='median_income', y='housing_median_age')
```

Out[47]:

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

In [48]:
```python
fig, axes = plt.subplots(1, 2, figsize=(15,8),sharex=False,sharey=False)

sns.histplot(ax=axes[0],data=cali_q2, x='housing_median_age')
sns.histplot(ax=axes[1],data=cali_q2, x='median_income')
```

Out[48]: <AxesSubplot:xlabel='median_income', ylabel='Count'>



In [49]:
```python
cali_q2.sort_values(by='housing_median_age',ascending=False)
```

Out[49]:

	housing_median_age	median_income	median_house_value	household_density	<1H OCEAN	INI
17601	52.0	5.6437	312000.0	2.21	1	
975	52.0	5.0677	233300.0	2.56	0	
15807	52.0	3.9079	370000.0	1.86	0	
15806	52.0	3.3702	370000.0	1.82	0	
15800	52.0	4.2222	361600.0	1.72	0	
...	...	...	...	...	...	
12156	2.0	4.1386	177300.0	3.37	1	
12286	1.0	1.6250	55000.0	4.00	0	
18972	1.0	5.2636	191300.0	3.25	0	
19536	1.0	4.2500	189200.0	3.59	0	
3130	1.0	4.8750	141700.0	2.13	0	

20396 rows × 13 columns

In [50]:

```
cali_q2.housing_median_age.describe()
```

Out[50]:

```
count    20396.000000
mean        28.630663
std         12.587017
min          1.000000
25%         18.000000
50%         29.000000
75%         37.000000
max         52.000000
Name: housing_median_age, dtype: float64
```

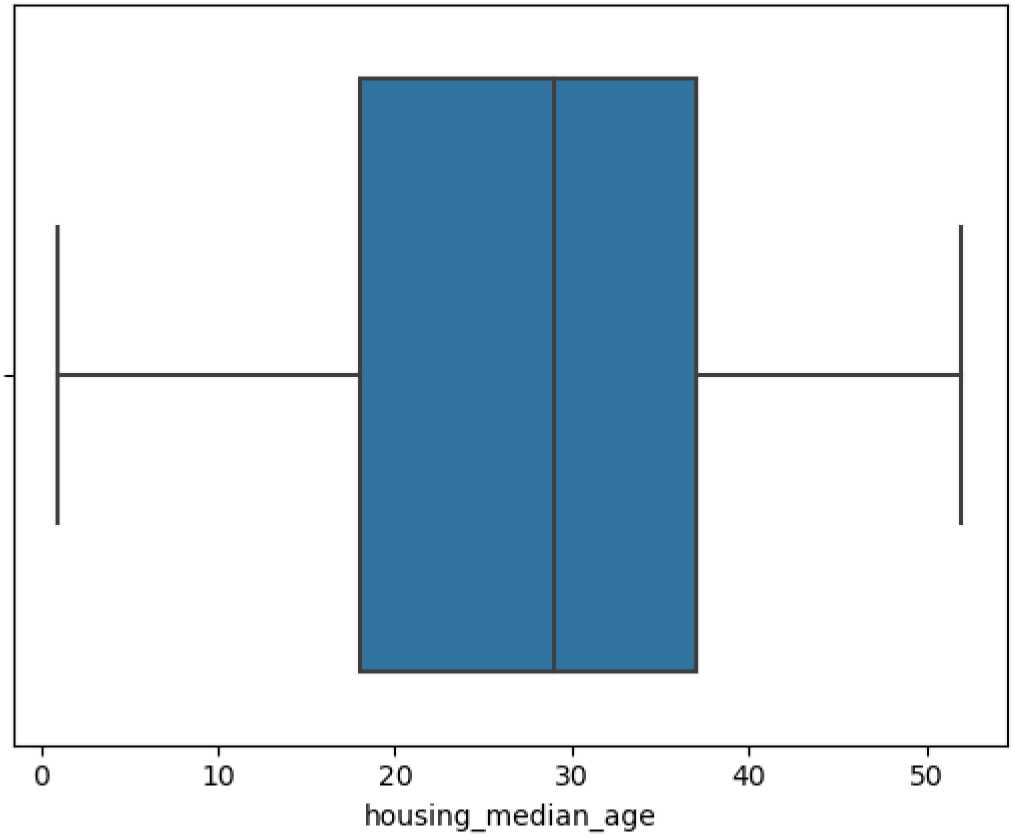
Check for outliers in house age variable

In [51]:

```
sns.boxplot(data=cali_q2,x='housing_median_age')
```

Out[51]:

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



In [32]:
```
print('There are',len(cali_q2.loc[(cali_q2['housing_median_age']>=1) & (cali_q2['h
print('There are',len(cali_q2.loc[(cali_q2['housing_median_age']>=10) & (cali_q2['
print(len(cali_q2.loc[(cali_q2['housing_median_age']>=20) & (cali_q2['housing_medi
print(len(cali_q2.loc[(cali_q2['housing_median_age']>=30) & (cali_q2['housing_medi
print(len(cali_q2.loc[(cali_q2['housing_median_age']>=40) & (cali_q2['housing_medi
print(len(cali_q2.loc[(cali_q2['housing_median_age']>=50) & (cali_q2['housing_medi
print(len(cali_q2.loc[cali_q2['housing_median_age']>=52]))
```

```
There are 1289 houses between 1 and 10 years old.
There are 4473 houses between 10 and 20 years old
4788
5716
2690
182
1258
```

In [48]:
```
cali_q2_52 = cali_q2.loc[cali_q2['housing_median_age']>=52]
cali_q2_52.head()
```

Out[48]:

	housing_median_age	median_income	median_house_value	household_density	<1H OCEAN	INLANI
2	52.0	7.2574	352100.0	2.80	0	(
3	52.0	5.6431	341300.0	2.55	0	(
4	52.0	3.8462	342200.0	2.18	0	(
5	52.0	4.0368	269700.0	2.14	0	(
6	52.0	3.6591	299200.0	2.13	0	(

Investigating houses at cap age of 52 years old

See if the houses at this age have any significant differences with houses under cap age, this is done by splitting dataframe and using various graphs to compare patterns between them

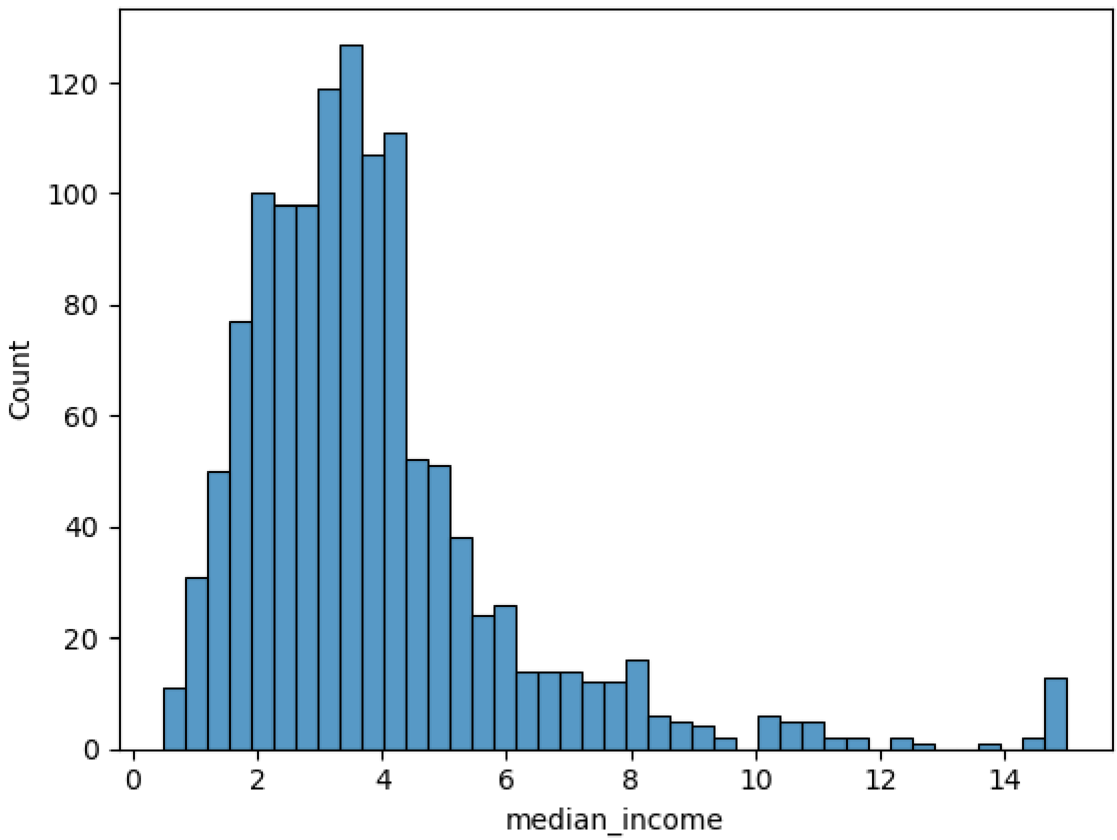
In [50]:
```
cali_q2_1 = cali_q2.loc[cali_q2['housing_median_age']<52]
cali_q2_1.head()
```

Out[50]:

	housing_median_age	median_income	median_house_value	household_density	<1H OCEAN	INLAN
0	41.0	8.3252	452600.0	2.56	0	
1	21.0	8.3014	358500.0	2.11	0	
8	42.0	2.0804	226700.0	2.03	0	
15	50.0	2.1250	140000.0	2.64	0	
18	50.0	1.9911	158700.0	2.36	0	

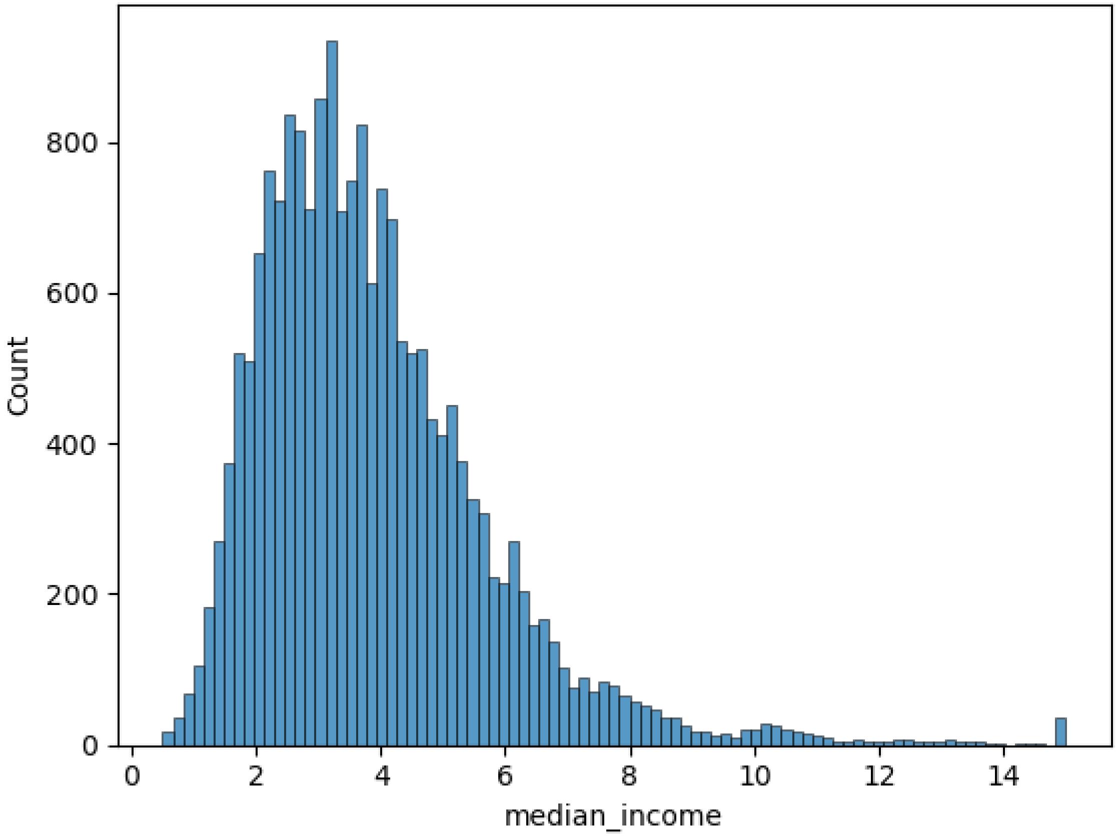
In [56]:
```
sns.histplot(data=cali_q2_52,x='median_income')
```

Out[56]: <AxesSubplot:xlabel='median_income', ylabel='Count'>



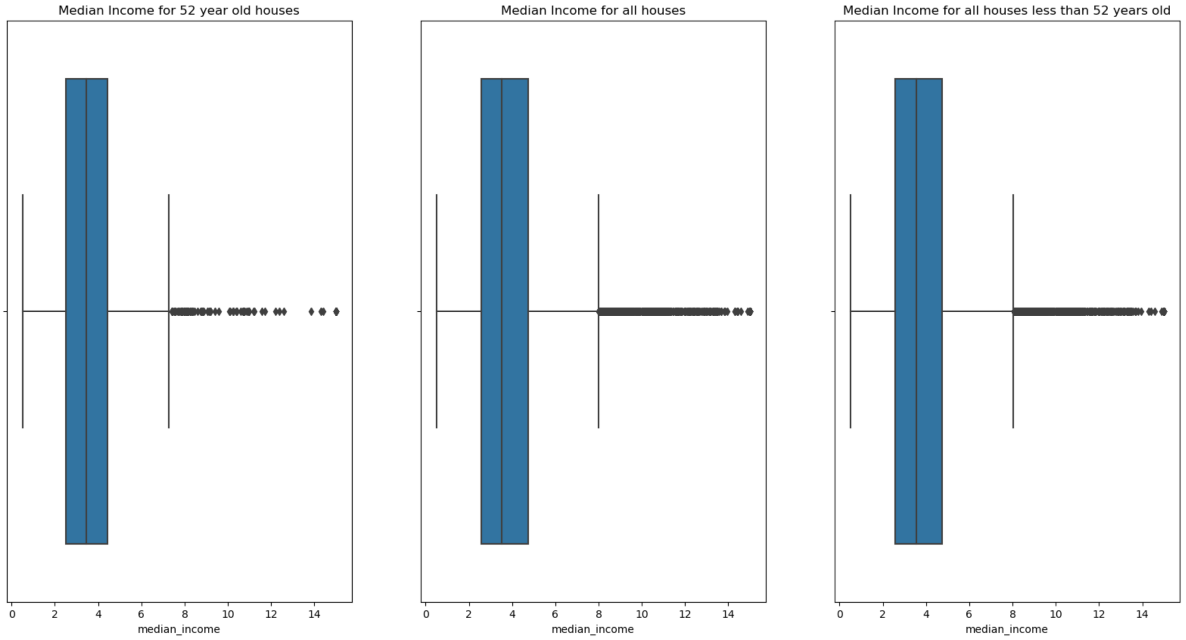
In [57]:
```
sns.histplot(data=cali_q2_1,x='median_income')
```

Out[57]: <AxesSubplot:xlabel='median_income', ylabel='Count'>



In [65]:
```
fig, axes = plt.subplots(1,3,figsize=(20,10),sharex=True)

sns.boxplot(ax=axes[0],data=cali_q2_52,x='median_income').set_title('Median Income
sns.boxplot(ax=axes[1],data=cali_q2,x='median_income').set_title('Median Income fo
sns.boxplot(ax=axes[2],data=cali_q2_1,x='median_income').set_title('Median Income
```

Out[65]:
```
Text(0.5, 1.0, 'Median Income for all houses less than 52 years old')
```



In [66]:
```
cali_q2.median_income.describe()
```

Out[66]:
```
count    20396.000000
mean         3.871039
std          1.896451
min          0.499900
25%          2.564375
50%          3.537500
75%          4.744375
max         15.000100
Name: median_income, dtype: float64
```

In [67]:
```
cali_q2_52.median_income.describe()
```

Out[67]:
```
count    1258.000000
mean        3.876096
std         2.281620
min         0.499900
25%         2.500325
50%         3.443750
75%         4.443125
max        15.000100
Name: median_income, dtype: float64
```

In [68]:
```
cali_q2_1.median_income.describe()
```

Out[68]:
```
count    19138.000000
mean         3.870706
std          1.868424
min          0.499900
25%          2.568125
50%          3.544250
75%          4.760900
max         15.000100
Name: median_income, dtype: float64
```

Calculating IQR for median income for all homes, 52 year old homes, and under 52 year old homes respectively

In [69]:
```
4.744375+(1.5*(4.744375-2.564375))
```

Out[69]:
```
8.014375
```

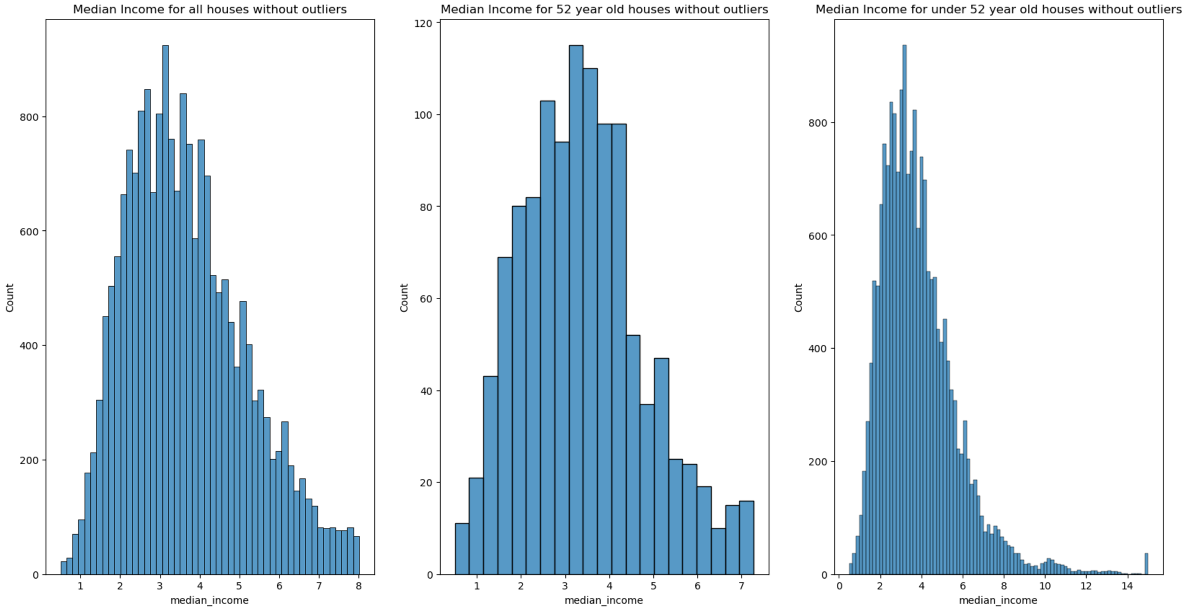
In [70]:
```
4.443125+(1.5*(4.443125-2.500325))
```

Out[70]:
```
7.357325
```

In [72]:
```
4.760900+(1.5*(4.760900-2.568125))
```

Out[72]:
```
8.050062500000001
```

In [51]:
```
fig, axes = plt.subplots(1,3,figsize=(20,10))

cali_q2_2 = cali_q2.loc[cali_q2['median_income']<8.014375]
cali_q2_52_2 = cali_q2_52.loc[cali_q2_52['median_income']<7.357325]

sns.histplot(ax=axes[0],data=cali_q2_2,x='median_income').set_title('Median Income
sns.histplot(ax=axes[1],data=cali_q2_52_2,x='median_income').set_title('Median Inc
sns.histplot(ax=axes[2],data=cali_q2_1,x='median_income').set_title('Median Income
```



Conclusion

- Median income for houses have the same positively-skewed distribution given that they include all age of homes, or if only 52 year old homes or homes below 52 years old. Thus it is reasonable that these 2 variables can be analysed for their impact on house value, but excluding homes that are 52 years old for efficiency.

Regression?

- back to top

Attempt to fit regression model that can calculate indicative price of house based on given variables, using linear regression.

Run regression on all variables

- back to top

Dropping "non-useful" variables, and island houses

In [44]:
```
cali_cleaned_reg = cali_cleaned2.drop(columns=['ocean_proximity','latitude','longi
                                             'house_value_standard','housing_age_s
                                              'ISLAND'])
cali_cleaned_reg = cali_cleaned_reg.drop([8314,8315,8316,8317,8318])
cali_cleaned_reg
```

Out[44]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_incom
0	41.0	880.0	129.0	322.0	126.0	8.325
1	21.0	7099.0	1106.0	2401.0	1138.0	8.301
2	52.0	1467.0	190.0	496.0	177.0	7.257
3	52.0	1274.0	235.0	558.0	219.0	5.643
4	52.0	1627.0	280.0	565.0	259.0	3.846
...	...	...	...	...	...	.
20635	25.0	1665.0	374.0	845.0	330.0	1.560
20636	18.0	697.0	150.0	356.0	114.0	2.556
20637	17.0	2254.0	485.0	1007.0	433.0	1.700
20638	18.0	1860.0	409.0	741.0	349.0	1.867
20639	16.0	2785.0	616.0	1387.0	530.0	2.388

20391 rows × 12 columns

In [45]:
```
cali_cleaned_reg.corr()
```

Out[45]:

	housing_median_age	total_rooms	total_bedrooms	population	households
housing_median_age	1.000000	-0.360953	-0.320740	-0.297181	-0.302950
total_rooms	-0.360953	1.000000	0.930264	0.863401	0.918852
total_bedrooms	-0.320740	0.930264	1.000000	0.884096	0.979705
population	-0.297181	0.863401	0.884096	1.000000	0.913800
households	-0.302950	0.918852	0.979705	0.913800	1.000000
median_income	-0.119757	0.198659	-0.007456	0.004560	0.013740
median_house_value	0.105648	0.133521	0.049621	-0.024615	0.064951
household_density	-0.003801	-0.106946	-0.144869	0.183296	-0.124904
<1H OCEAN	0.045692	-0.004894	0.017223	0.075366	0.040741
INLAND	-0.237205	0.027221	-0.005710	-0.021572	-0.037579
NEAR BAY	0.256209	-0.023525	-0.019690	-0.060827	-0.011115
NEAR OCEAN	0.021585	-0.008513	0.000867	-0.024738	0.002243

Regression model using independent variables on house price

In [53]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

```
nocaps = cali_cleaned_reg[(cali_cleaned_reg.median_house_value <= 500000)]
X = nocaps[['housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'h
           'median_income', '<1H OCEAN', 'INLAND', 'NEAR BAY', 'NEAR OCEAN']]
Y = nocaps[['median_house_value']]


linear1 = LinearRegression(fit_intercept = True)
linear1.fit(X,Y)




X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.8, random_st



coefficients1 = np.round(linear1.coef_, 3)
intercept1 = np.round (linear1.intercept_,3)



training_score = linear1.score(X_train, Y_train)



predictions = linear1.predict(X_test)



test_score = r2_score(Y_test, predictions)



print(training_score)
print(test_score)
```

```
0.6175501027078568
0.5962326887712996
```

Fitting model to houses at cap value 501k

In [54]:

```
caps = cali_cleaned_reg[(cali_cleaned_reg.median_house_value >= 500001)]
predict_X = caps[['housing_median_age', 'total_rooms', 'total_bedrooms', 'populati
           'median_income', '<1H OCEAN', 'INLAND', 'NEAR BAY', 'NEAR OCEAN']]

caps['predicted value'] = linear1.predict(predict_X)
actual_predict = linear1.predict(predict_X)

caps
```

```
C:\Users\joshu\AppData\Local\Temp\ipykernel_20244\2179456940.py:5: SettingWithCopy
Warning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  caps['predicted value'] = linear1.predict(predict_X)
```

Out[54]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_incom
89	52.0	249.0	78.0	396.0	85.0	1.243
459	52.0	609.0	236.0	1349.0	250.0	1.169
493	52.0	1668.0	225.0	517.0	214.0	7.852
494	52.0	3726.0	474.0	1366.0	496.0	9.395
509	52.0	2990.0	379.0	947.0	361.0	7.877
...	...	...	...	...	...	.
20422	35.0	1503.0	263.0	576.0	216.0	5.145
20426	11.0	1177.0	138.0	415.0	119.0	10.047
20427	4.0	15572.0	2222.0	5495.0	2152.0	8.649
20436	10.0	3663.0	409.0	1179.0	371.0	12.542
20443	50.0	187.0	33.0	130.0	35.0	3.343

Showing homes with predicted value more than cap price

In [55]:
```
caps_above = caps.loc[caps['predicted value']>=500001]
caps_above.head()
```

Out[55]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
510	39.0	2492.0	310.0	808.0	315.0	11.8603
511	42.0	2991.0	335.0	1018.0	335.0	13.4990
512	52.0	3242.0	366.0	1001.0	352.0	12.2138
514	52.0	3494.0	396.0	1192.0	383.0	12.3804
3858	42.0	777.0	102.0	284.0	113.0	11.2093

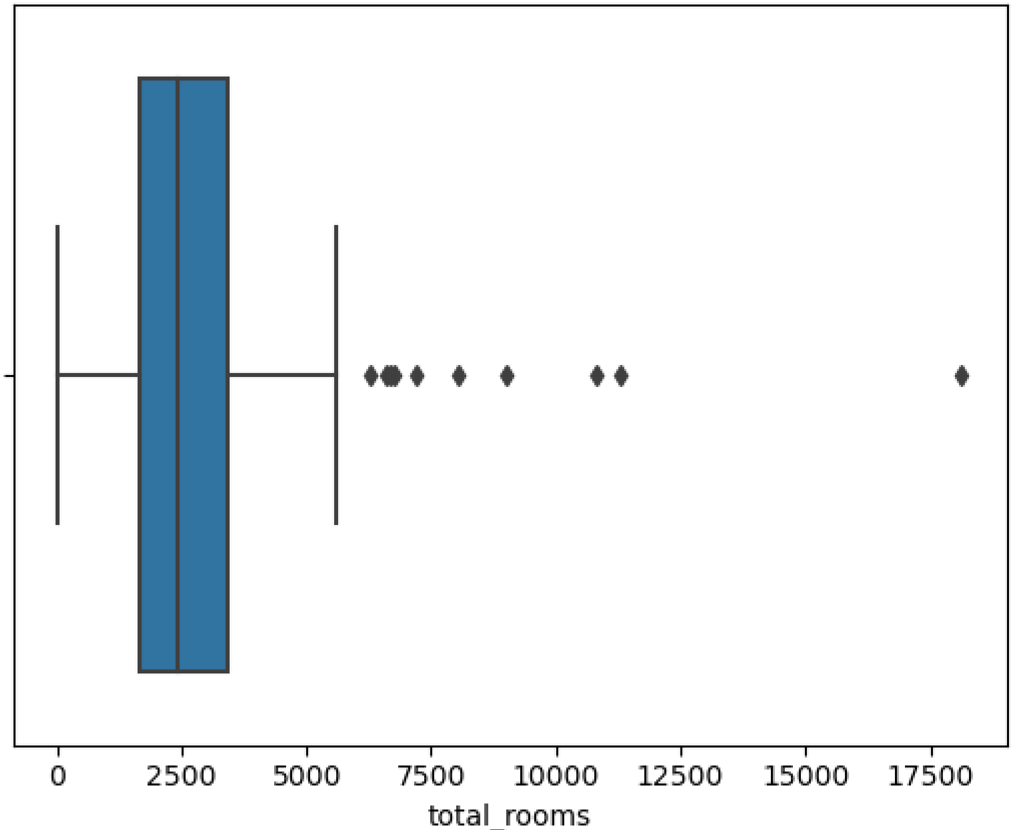
In [56]:
```
caps_above.sort_values(by='total_rooms',ascending=False)
```

Out[56]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_incom
8985	21.0	18132.0	5419.0	7431.0	4930.0	5.335
18361	23.0	11294.0	1377.0	3840.0	1367.0	12.138
5244	27.0	10806.0	1440.0	3511.0	1352.0	12.729
5259	29.0	9013.0	1117.0	2919.0	1061.0	13.947
5252	32.0	8041.0	1141.0	2768.0	1106.0	11.197
...	...	...	...	...	...	.
6399	35.0	249.0	31.0	268.0	29.0	15.000
9418	38.0	240.0	29.0	63.0	34.0	12.254
18363	7.0	189.0	26.0	84.0	29.0	13.809
17858	43.0	91.0	12.0	58.0	16.0	15.000
17118	46.0	30.0	4.0	13.0	5.0	15.000

143 rows × 13 columns

In [57]:
```
sns.boxplot(data=caps_above, x='total_rooms');
```



Comments

- Model seems to fit data of houses below cap value quite well, however for homes at cap price not all entries have predicted values above the capped value of 501k in the dataset.

- It can thus be implied that there might be more factors in play that affect house value that are not featured in the dataset.
- Therefore as an overwhelming majority of predicted values are below their "actual" value of 501k, the model cannot be used reliably to predict house prices.
- Maybe remove row 8985, as most of its values are outliers
- However most of the data predicts house price below cap value, thus model may not be best fit

Multiple linear regression for impact of population and ocean proximity on house price

- back to top

In [24]:
```
from sklearn.linear_model import LinearRegression # importing functions for linear
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

In [39]:
```
X = cali_cleaned[['population', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR
Y = cali_cleaned['median_house_value']

linear = LinearRegression(fit_intercept = True)
linear.fit(X,Y)

coefficients = np.round(linear.coef_, 3)
intercept = np.round(linear.intercept_, 3)

print(coefficients)
print(intercept) # experiment
```

```
[-3.17500000e+00 -9.60537800e+03 -1.25452967e+05  1.27839319e+05
  8.36459200e+03 -1.14556700e+03]
254721.649
```

In [186…
```
nocaps = cali_cleaned[(cali_cleaned.median_house_value <= 500000)]
# creating dataset with only blocks valued below or equal to $500000
X = nocaps[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_b
           'median_income', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCE
# defining X (explanatory variables) as all variables other than median house valu
Y = nocaps[['median_house_value']]
# Y is median house value as this is what we want to predict
# both X and Y are from the nocaps dataset as this is what we want to use to train

linear1 = LinearRegression(fit_intercept = True) #defining the model
linear1.fit(X,Y) # fitting X and Y to the model


X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.8, random_s
# splitting X and Y into a training and testing set

coefficients1 = np.round(linear1.coef_, 3) # obtaining coefficients of model
intercept1 = np.round (linear1.intercept_,3) # obtaining intercepts of model

training_score = linear1.score(X_train, Y_train) # calculating training score

predictions = linear1.predict(X_test) # calculating predicted values using testing

test_score = r2_score(Y_test, predictions) # calculating testing score by comparin
```

```
print('training score:', training_score)
print('testing score:', test_score)
print('coefficients:', coefficients1)
print('intercept:', intercept1)
```

```
training score: 0.6288125852696701
testing score: 0.6126815432933537
coefficients: [[-2.44324730e+04 -2.25714660e+04  9.31378000e+02 -6.65100000e+00
   8.70170000e+01 -3.33540000e+01  5.38430000e+01  3.83430540e+04
  -2.45039880e+04 -6.34841900e+04  1.40597689e+05 -3.15500330e+04
  -2.10594780e+04]]
intercept: [-2062340.455]
```

In [136…

```
caps = cali_cleaned[(cali_cleaned.median_house_value >= 500001)]
# creating a dataset with only the blocks with capped median house values
predict_X = caps[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'to
           'median_income', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCE
# defining the variables the model with predict from
```

In [143…

```
caps['predicted value'] = linear1.predict(predict_X)
# adding a column in the dataset with predicted X values
actual_predict = linear1.predict(predict_X)
```

```
/var/folders/m4/s3xj9xf96hbd8zvh6_dnr3wr0000gq/T/ipykernel_38073/179709594.py:1: S
ettingWithCopyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  caps['predicted value'] = linear1.predict(predict_X)
```

In [146…

```
caps
```

Out[146]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
89	-122.27	37.80	52.0	249.0	78.0	396.0	
459	-122.25	37.87	52.0	609.0	236.0	1349.0	
493	-122.24	37.86	52.0	1668.0	225.0	517.0	
494	-122.24	37.85	52.0	3726.0	474.0	1366.0	
509	-122.23	37.83	52.0	2990.0	379.0	947.0	
...	...	...	...	...	...	...	
20422	-118.90	34.14	35.0	1503.0	263.0	576.0	
20426	-118.69	34.18	11.0	1177.0	138.0	415.0	
20427	-118.80	34.19	4.0	15572.0	2222.0	5495.0	2
20436	-118.69	34.21	10.0	3663.0	409.0	1179.0	
20443	-118.85	34.27	50.0	187.0	33.0	130.0	

953 rows × 20 columns

In [142…

```
caps['predicted value'].describe()
# looking at how the predicted values are distributed
```

Out[142]:
```
count        953.000000
mean      384073.368756
std       113774.890984
min        33989.229071
25%       301049.287940
50%       374833.281018
75%       454766.201815
max       668423.157759
Name: predicted value, dtype: float64
```

Running linear regression model on only houses NEAR BAY

- back to top

In [55]:
```
# finding correlation between housing_median_age and median_house_value
cali_cleaned['housing_median_age'].corr(cali_cleaned['median_house_value'])
```

Out[55]:
```
0.10558929296743953
```

In [29]:
```
# finding correlation between household_average and median_house_value
cali_cleaned['household_average'].corr(cali_cleaned['median_house_value'])
```

Out[29]:
```
-0.24295460983910497
```

In [56]:
```
# finding correlation between median_income and median_house_value
cali_cleaned['median_income'].corr(cali_cleaned['median_house_value'])
```

Out[56]:
```
0.6895984666143862
```

In [57]:
```
# finding correlation between longitude and median_house_value

cali_cleaned['longitude'].corr(cali_cleaned['median_house_value'])
```

Out[57]:
```
-0.045537290840117726
```

In [58]:
```
# finding correlation between latitude and median_house_value

cali_cleaned['latitude'].corr(cali_cleaned['median_house_value'])
```

Out[58]:
```
-0.14449026879295912
```

In [59]:
```
# finding correlation between total_rooms and median_house_value

cali_cleaned['total_rooms'].corr(cali_cleaned['median_house_value'])
```

Out[59]:
```
0.1333988941087788
```

In [60]:
```
# finding correlation between total_bedrooms and median_house_value

cali_cleaned['total_bedrooms'].corr(cali_cleaned['median_house_value'])
```

Out[60]:
```
0.04948646255884484
```

In [35]:
```
cali_cleaned.columns
```

Out[35]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity', 'household_average',
       'zmedianincome', 'zhouseage', 'zhousevalue', 'ONEHROCEAN', 'INLAND',
       'NEAR_BAY', 'NEAR_OCEAN'],
      dtype='object')
```

In [61]:
```
# creating a dataframe of values that only contain NEAR_BAY = 1 and training a regi
# 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population',
cali_cleaned_nearb = cali_cleaned[(cali_cleaned.NEAR_BAY == 1)]
X = cali_cleaned_nearb[['longitude', 'latitude', 'housing_median_age', 'total_room
           'median_income']]
Y = cali_cleaned_nearb[['median_house_value']]



linearbay = LinearRegression(fit_intercept = True)
linearbay.fit(X,Y)




X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.8, random_s



coefficientsbay = np.round(linearbay.coef_, 3)
interceptbay = np.round (linearbay.intercept_,3)



training_score_bay = linearbay.score(X_train, Y_train)



predictions_bay = linearbay.predict(X_test)



test_score_bay = r2_score(Y_test, predictions_bay)



print(linearbay.coef_)
print(linearbay.intercept_)
print(test_score_bay)
```

```
[[-3.03397868e+05 -1.97269781e+05  8.98324465e+02  1.43260249e+01
   5.07827806e+01 -7.96133438e+01  1.08711469e+02  3.73095024e+04]]
[-29584221.2461795]
0.6656384329737136
```

In [62]:
```
# analysing the influence of our regression model on capped house values for near b
# Creating X tests for predictions
capsbay = cali_cleaned_nearb[(cali_cleaned.median_house_value >= 500001)]
predict_X = capsbay[['longitude', 'latitude', 'housing_median_age', 'total_rooms',
           'median_income']]
```

```
C:\Users\zhang\AppData\Local\Temp/ipykernel_12776/481340704.py:2: UserWarning: Boo
lean Series key will be reindexed to match DataFrame index.
  capsbay = cali_cleaned_nearb[(cali_cleaned.median_house_value >= 500001)]
```

In [63]:
```
# predicting new median house values for capped median house values with regressio
capsbay['predicted value'] = linearbay.predict(predict_X)
```

```
actual_predict = linearbay.predict(predict_X)
capsbay
```

```
C:\Users\zhang\AppData\Local\Temp/ipykernel_12776/4021561677.py:2: SettingWithCopy
Warning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  capsbay['predicted value'] = linearbay.predict(predict_X)
```

Out[63]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
89	-122.27	37.80	52	249	78.0	396	
459	-122.25	37.87	52	609	236.0	1349	
493	-122.24	37.86	52	1668	225.0	517	
494	-122.24	37.85	52	3726	474.0	1366	
509	-122.23	37.83	52	2990	379.0	947	
...	...	...	...	...	...	...	
18358	-122.10	37.36	35	2063	266.0	676	
18359	-122.09	37.35	37	1795	285.0	791	
18360	-122.09	37.35	30	1502	186.0	501	
18361	-122.14	37.36	23	11294	1377.0	3840	
18362	-122.15	37.35	23	3814	485.0	1344	

191 rows × 19 columns

In [65]:

```
# Analysing predicted values to see how many median house values are actually over
capsbay['predicted value'].describe()
```

Out[65]:

```
count       191.000000
mean     432379.643994
std      105940.080433
min       66400.273768
25%      368661.082092
50%      417068.558777
75%      499679.327925
max      726746.657841
Name: predicted value, dtype: float64
```

Running linear regression model on homes located NEAR OCEAN

- back to top

In [40]:

```
# creating a dataframe of values that only contain NEAR_OCEAN = 1 and training a re
# 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population',
cali_cleaned_nearo = cali_cleaned[(cali_cleaned.NEAR_OCEAN == 1)]
X = cali_cleaned_nearo[['longitude', 'latitude', 'housing_median_age', 'total_room
         'median_income']]
Y = cali_cleaned_nearo[['median_house_value']]
```

```
linearocean = LinearRegression(fit_intercept = True)
linearocean.fit(X,Y)



X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.8, random_s


coefficientsocean = np.round(linearocean.coef_, 3)
interceptocean = np.round (linearocean.intercept_,3)


training_score_ocean = linearocean.score(X_train, Y_train)


predictions_ocean = linearocean.predict(X_test)


test_score_ocean = r2_score(Y_test, predictions_ocean)


print(linearocean.coef_)
print(linearocean.intercept_)
print(test_score_ocean)
```

```
[[-4.97188658e+04 -4.72926098e+04  1.51895514e+03  7.39973439e-01
   1.66607861e+02 -5.42359943e+01 -1.16381849e+01  4.18298350e+04]]
[-4265862.23227052]
0.5984485957955172
```

In [66]:
```
# analysing the influence of our regression model on capped house values for near c
# Creating X tests for predictions
capsocean = cali_cleaned_nearo[(cali_cleaned.median_house_value >= 500001)]
predict_X = capsocean[['longitude', 'latitude', 'housing_median_age', 'total_rooms
            'median_income']]
capsocean['predicted value'] = linearocean.predict(predict_X)
actual_predict = linearocean.predict(predict_X)
capsocean
```

```
C:\Users\zhang\AppData\Local\Temp/ipykernel_12776/2725001967.py:3: UserWarning: Bo
olean Series key will be reindexed to match DataFrame index.
  capsocean = cali_cleaned_nearo[(cali_cleaned.median_house_value >= 500001)]
C:\Users\zhang\AppData\Local\Temp/ipykernel_12776/2725001967.py:6: SettingWithCopy
Warning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  capsocean['predicted value'] = linearocean.predict(predict_X)
```

Out[66]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
5276	-118.55	33.99	39	2603	456.0	928	
5674	-118.32	33.73	25	1099	168.0	407	
8188	-118.13	33.79	29	2937	524.0	1132	
8189	-118.13	33.78	31	3039	739.0	1199	
8268	-118.17	33.74	36	2006	453.0	807	
...	...	...	...	...	...	...	
20233	-119.29	34.24	27	4742	775.0	1682	
20272	-119.23	34.19	16	5297	810.0	1489	
20273	-119.23	34.17	18	6171	1490.0	2164	
20322	-119.14	34.23	8	243	75.0	102	
20380	-118.83	34.14	16	1316	194.0	450	

In [67]:

```
# Analysing predicted values to see how many median house values are actually over
capsocean['predicted value'].describe()
```

Out[67]:

```
count       209.000000
mean     426297.212385
std      120482.096416
min      137833.167430
25%      339712.430948
50%      416789.478414
75%      496629.083501
max      749180.095275
Name: predicted value, dtype: float64
```

Linear regression of household average and average median income

- back to top

In [69]:

```
#Caluculating testing and training scores for the linear regression of household av
X = grouped[['household_average']]
y = grouped['median_income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

model = LinearRegression()
model.fit(X_train, y_train)

y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

train_score = r2_score(y_train, y_train_pred)
test_score = r2_score(y_test, y_test_pred)

print("Training R^2 score:", train_score)
print("Testing R^2 score:", test_score)
```

```
Training R^2 score: 0.039158003652879136
Testing R^2 score: -1.0884409298044484
```

Taking different approach in attempts to find better correlation: Not rounding the household average

In [6]:
```
#import the house data again, without rounding the household average
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

cali = pd.read_csv('california-housing-data.csv')
cali1 = cali[cali.ocean_proximity != 'ISLAND'].dropna()
cali1['household_average'] = cali1['population']/cali1['households']
apartments = cali1[ (cali1['household_average'] > 8)].index
cali1.drop(apartments , inplace=True)
cali1.head()
```

Out[6]:

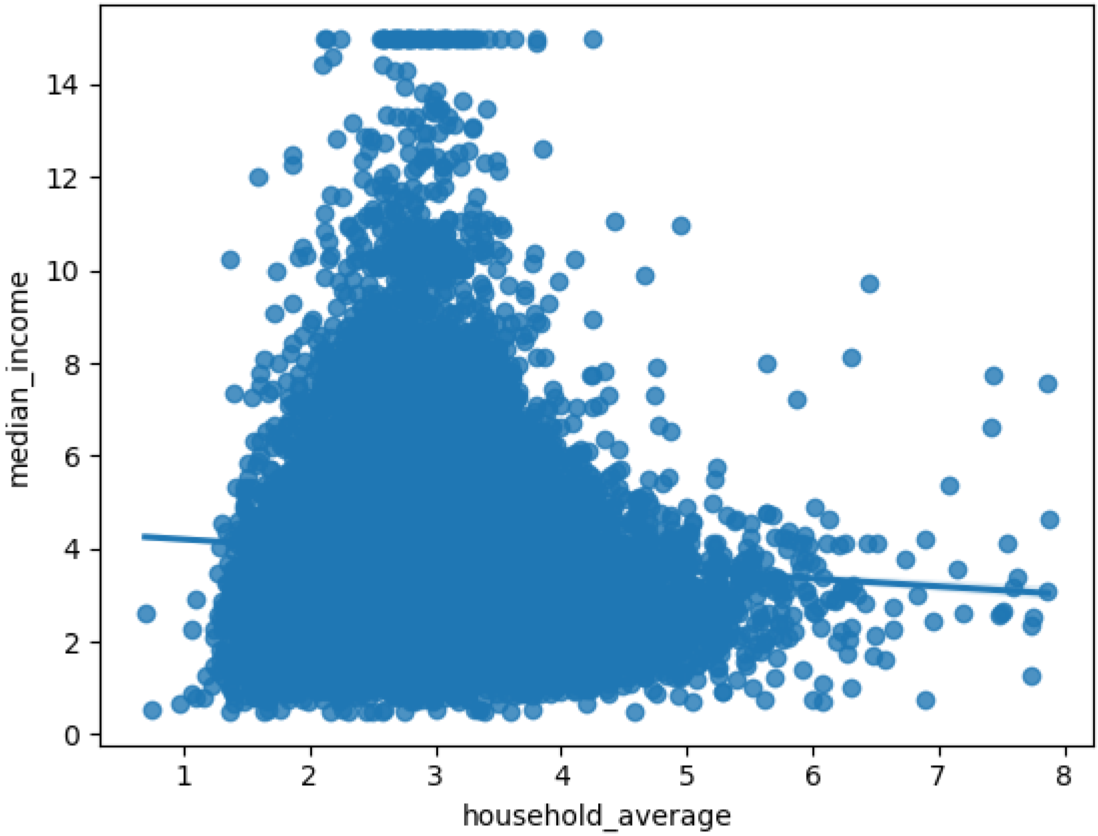
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [2]:
```
#finding correlation of raw household average and median income
cali1["median_income"].corr(cali1["household_average"])
```

Out[2]: -0.06658784472799086

In [3]:
```
#Graph and regression plot of raw household average and median income
sns.regplot(x="household_average",
            y="median_income",
            data=cali1)
```

Out[3]: <Axes: xlabel='household_average', ylabel='median_income'>



In [70]:
```
#Finding training and testing scores of regression plot of raw household average an
X = cali1[['household_average']]
y = cali1['median_income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the training and testing sets
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

# Calculate the R^2 scores
train_score = r2_score(y_train, y_train_pred)
test_score = r2_score(y_test, y_test_pred)

print("Training R^2 score:", train_score)
print("Testing R^2 score:", test_score)
```

```
Training R^2 score: 0.0036357485959800373
Testing R^2 score: 0.00734099939936983
```

back to top