```
import pandas as pd

# читаем
df = pd.read_csv('sample_data/california_housing_train.csv')

df.head(n = 10)   # показывает первые строки таблицы, по умолчанию 5
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0
2	-114.56	33.69	17.0	720.0	174.0	333.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0
5	-114.58	33.63	29.0	1387.0	236.0	671.0
6	-114.58	33.61	25.0	2907.0	680.0	1841.0
7	-114.59	34.83	41.0	812.0	168.0	375.0
8	-114.59	33.61	34.0	4789.0	1175.0	3134.0
9	-114.60	34.83	46.0	1497.0	309.0	787.0

```
df.tail(n = 2)  # показывает хвост таблицы
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
16998	-124.30	41.80	19.0	2672.0	552.0	129
16999	-124.35	40.54	52.0	1820.0	300.0	80

```
df.shape        # показывает общий размер таблицы (кол-во строк и столбцов)
```

```
(17000, 9)
```

```
df.isnull()     # показывает есть ли нулевое значение в таблице(если есть, то true)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
0	False	False	False	False	False	Fa
1	False	False	False	False	False	Fa
2	False	False	False	False	False	Fa
3	False	False	False	False	False	Fa
4	False	False	False	False	False	Fa
...	...	...	...	...	...	
16995	False	False	False	False	False	Fa
16996	False	False	False	False	False	Fa
16997	False	False	False	False	False	Fa
16998	False	False	False	False	False	Fa
16999	False	False	False	False	False	Fa

17000 rows × 9 columns

```
df.isnull().sum()     # сумирует данные по столбцам
```

```
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
dtype: int64
```

```
df.dtypes       # показывает типы данных
```

```
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
```

```
df.columns      # покаывзает все колонки
```

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

ВЫБОРКА ДАННЫХ

```
df['latitude']    # выборка данных только по этому столбцу
```

```
0        34.19
1        34.40
2        33.69
3        33.64
4        33.57
         ...
16995    40.58
16996    40.69
16997    41.84
16998    41.80
16999    40.54
Name: latitude, Length: 17000, dtype: float64
```

```
df[['latitude', 'population']]  # выборка по двум столбцам
```

	latitude	population
0	34.19	1015.0
1	34.40	1129.0
2	33.69	333.0
3	33.64	515.0
4	33.57	624.0
...	...	...
16995	40.58	907.0
16996	40.69	1194.0
16997	41.84	1244.0
16998	41.80	1298.0
16999	40.54	806.0

17000 rows × 2 columns

Задание. Необходимо вывести столбец total_rooms, у которого медианный возраст здания (housing_median_age) меньше 20

```
df[df['housing_median_age'] < 20]
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
0	-114.31	34.19	15.0	5612.0	1283.0	101
1	-114.47	34.40	19.0	7650.0	1901.0	112
2	-114.56	33.69	17.0	720.0	174.0	33
3	-114.57	33.64	14.0	1501.0	337.0	51
10	-114.60	33.62	16.0	3741.0	801.0	243
...	...	...	...	...	...	
16983	-124.19	41.78	15.0	3140.0	714.0	164
16987	-124.21	41.77	17.0	3461.0	722.0	194
16991	-124.23	41.75	11.0	3159.0	616.0	134
16997	-124.30	41.84	17.0	2677.0	531.0	124
16998	-124.30	41.80	19.0	2672.0	552.0	129

4826 rows × 9 columns

& - выполнение одновременно **всех** условий

| - выполнение **хотя бы одного** из условий

```
df[df['housing_median_age'] < 20]['total_rooms']
```

```
0        5612.0
1        7650.0
2         720.0
3        1501.0
10       3741.0
          ...
16983    3140.0
16987    3461.0
16991    3159.0
16997    2677.0
16998    2672.0
Name: total_rooms, Length: 4826, dtype: float64
```

```
df[(df['housing_median_age'] < 20) & (df['housing_median_age'] > 10)]['total_rooms']
```

```
0        5612.0
1        7650.0
2         720.0
3        1501.0
10       3741.0
          ...
16983    3140.0
16987    3461.0
16991    3159.0
16997    2677.0
16998    2672.0
Name: total_rooms, Length: 3513, dtype: float64
```

```
df[(df['housing_median_age'] < 20) & (df['housing_median_age'] > 10)][['total_rooms', 'housing_median_age']]
```

	total_rooms	housing_median_age
0	5612.0	15.0
1	7650.0	19.0
2	720.0	17.0
3	1501.0	14.0
10	3741.0	16.0
...	...	...
16983	3140.0	15.0
16987	3461.0	17.0
16991	3159.0	11.0
16997	2677.0	17.0
16998	2672.0	19.0

3513 rows × 2 columns

ПРОСТАЯ СТАТИСТИКА

```
print(df['population'].max())
```

```
35682.0
```

```
print(df['population'].min())
```

```
3.0
```

```
print(df['population'].mean())
```

```
1429.5739411764705
```

```
print(df['population'].sum())
```

```
24302757.0
```

```
df[['population', 'total_rooms']].median()
```

```
population     1167.0
total_rooms    2127.0
dtype: float64
```

```
df.describe()   # вывод информации о всей таблице
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000	1
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824	
std	2.005166	2.137340	12.586937	2179.947071	421.499452	
min	-124.350000	32.540000	1.000000	2.000000	1.000000	
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000	
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000	
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000	
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000	3

count - общее кол-во не пустых строк

mean - среднее значение в столбце

std - стандартное отклонение от среднего значения

min - минимальное значение

max - максимальное значение

Числа **25%, 50%, 75%** - перцентили. **Перцентиль** - это показатель, используемый в статистике, показывающий значение, ниже которого падает определенный процент наблюдений в группе наблюдений

ИЗОБРАЖАЕМ СТАТИСТИЧЕСКИЕ ОТНОШЕНИЯ

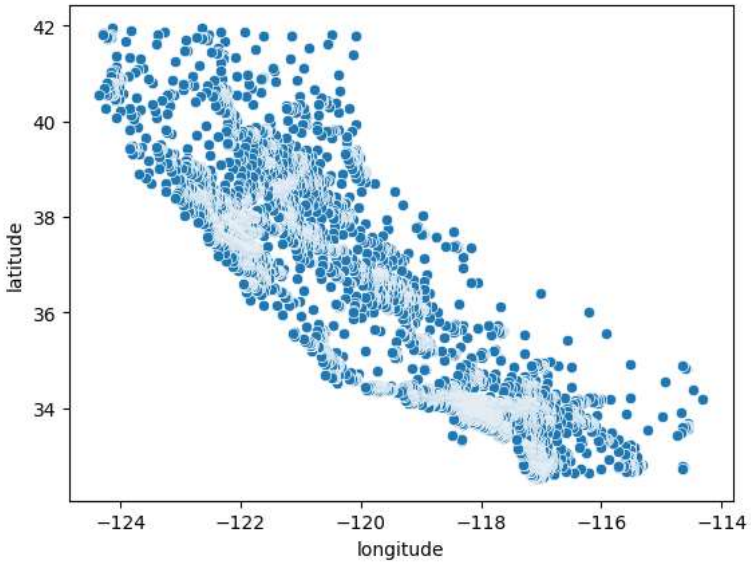
Scatterplot (Точечный график) - Математическая диаграмма, изображающая значение двух переменных в виде точек на декартовой плоскости. Библиотека **seaborn** без труда принимает **pandas DataFrame** (таблицу). Чтобы изобразить отношения между двумя столбцами достаточно указать, какой столбец отобразить по оси x, а какой по оси y.

```
import seaborn as sns
```

Изображение точек долготы по отношению к широте:

```
sns.scatterplot(data=df, x = "longitude", y = "latitude")
```

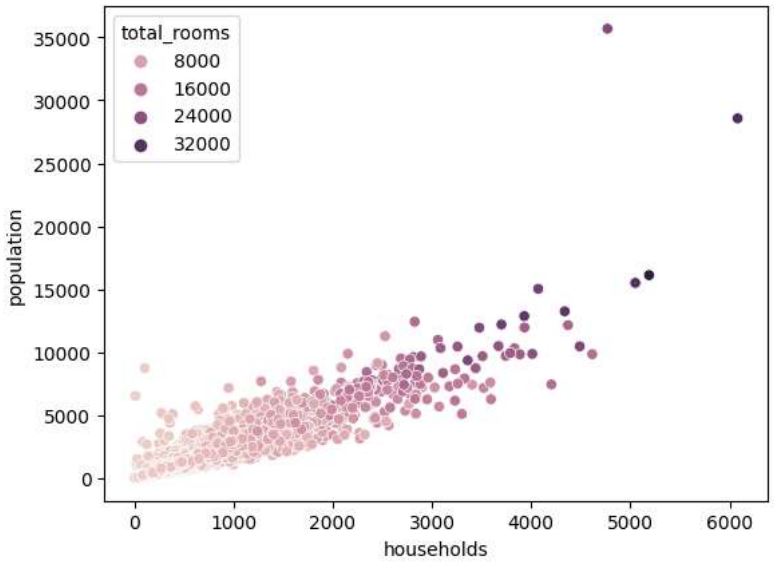
```
<Axes: xlabel='longitude', ylabel='latitude'>
```



Отношение, чем выше кол-во семей, тем выше кол-во людей и соответственно комнат.

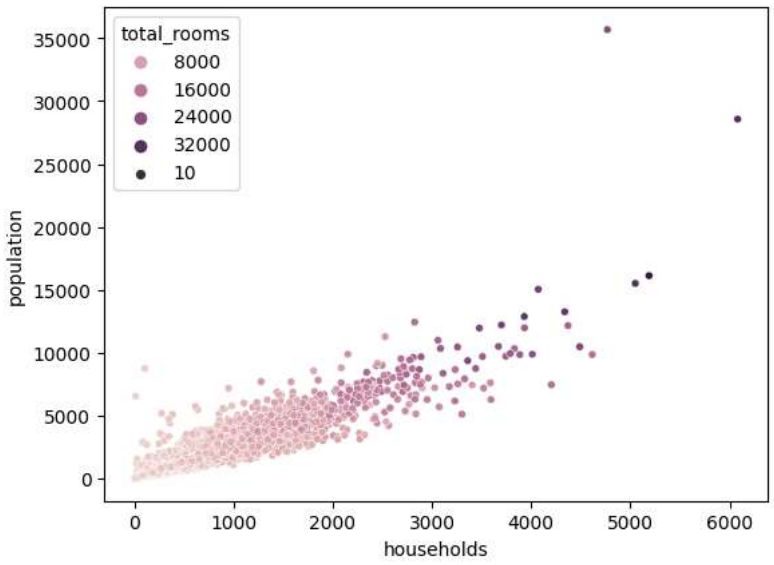
```
sns.scatterplot(data = df, x = "households", y = "population", hue = "total_rooms")   # hue - оттенок, чем насыщеннее - тем больше
```

```
<Axes: xlabel='households', ylabel='population'>
```




```
sns.scatterplot(data = df, x = "households", y = "population", hue = "total_rooms", size = 10)  # size - размер точек
```

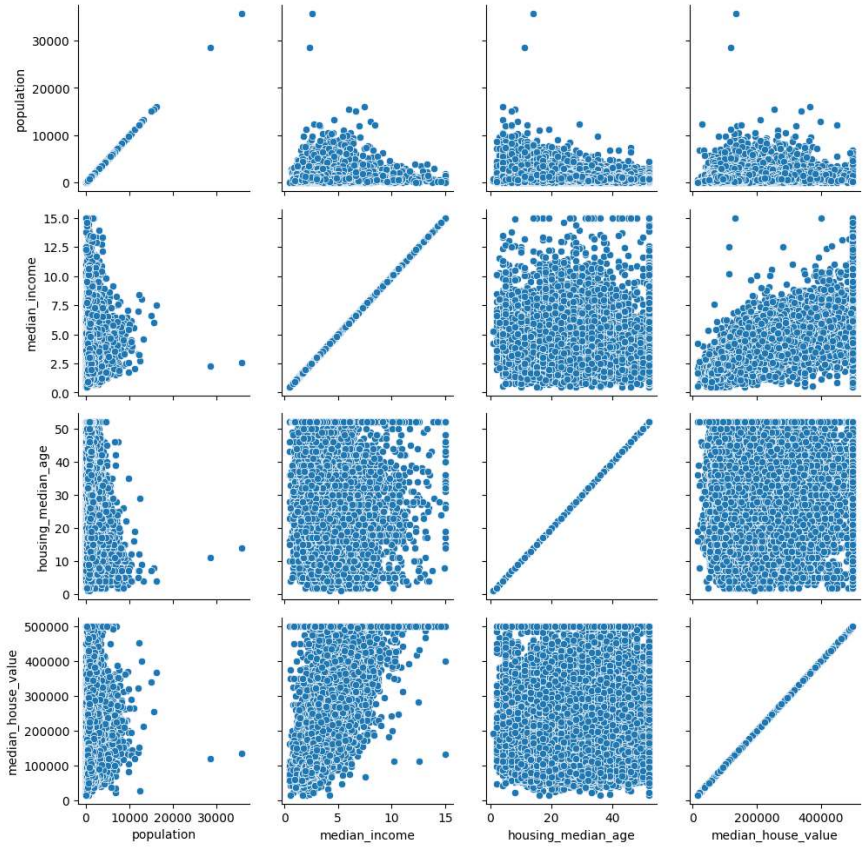
```
<Axes: xlabel='households', ylabel='population'>
```



Мы можем визуализировать сразу несколько отношений, используя класс **PairGrid** внутри **seaborn**. **PairGrid** принимает как аргумент **pandas DataFrame** и визуализирует все возможные отношения между ними, в соответствии с выбранным типом графика.

```
cols = ['population', 'median_income', 'housing_median_age', 'median_house_value']
g = sns.PairGrid(df[cols])
g.map(sns.scatterplot)
```

```
<seaborn.axisgrid.PairGrid at 0x7a19f092e8c0>
```

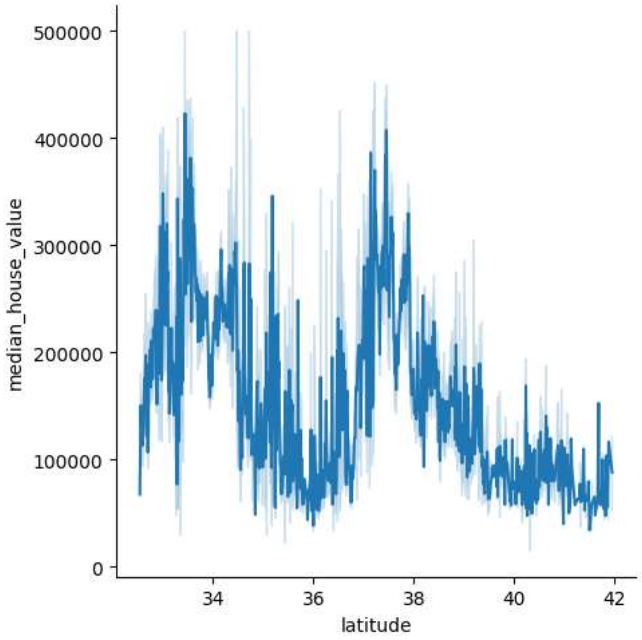


ЛИНЕЙНЫЕ ГРАФИКИ

Хорошо подойдут, если есть временная или какая-либо иная последовательность и значения, которые могут меняться в зависимости от нее. Для генерации линейных графиков в **seaborn** используют **relplot** функцию. Она так же принимает **DataFrame, x, y** - столбцы

```
sns.relplot(x = "latitude", y = "median_house_value", kind = "line", data = df)
```

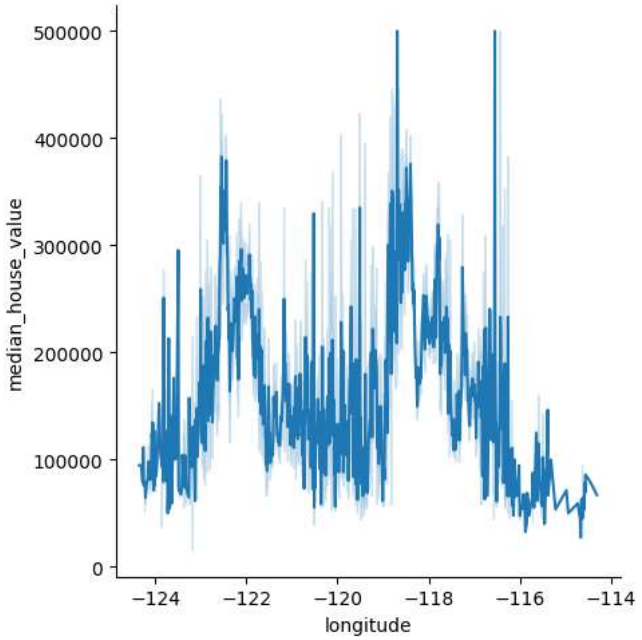
```
<seaborn.axisgrid.FacetGrid at 0x7a19f0209390>
```



Можно видеть, что в определенных меатх долготы цена на дома резко подскакивает. Попробуем визуализировать longitude по отношению к median_house_value и поймем в чем дело, почему цена резко подскакивает

```
sns.relplot(x = "longitude", y = "median_house_value", kind = "line", data = df)
```

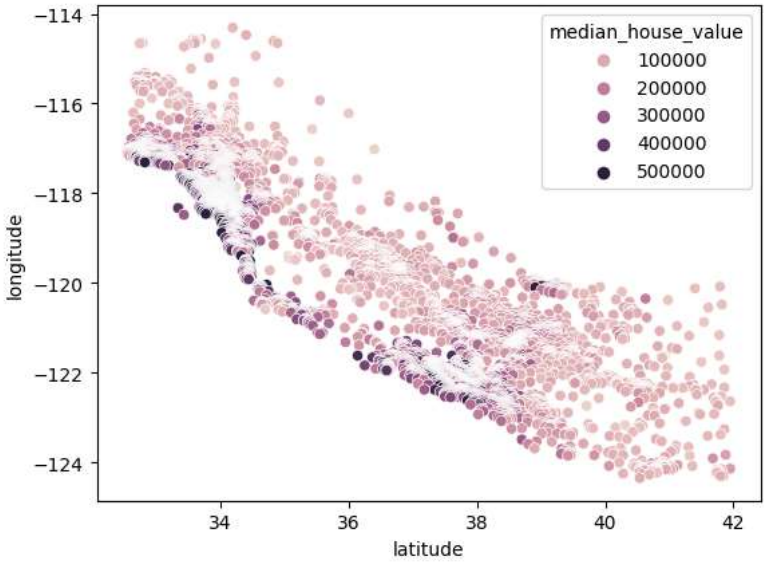
```
<seaborn.axisgrid.FacetGrid at 0x7a19f01f6770>
```



Можно видеть, что в определенных местах широты цена на дома также очень высока. Используя точечный график можно визуализировать эти отношения с большей четкостью. Скорее всего резкий рост цен связан с близостью к ценному объекту, повышающему качество жизни, скорее всего побережью океана или реки.

```
sns.scatterplot(data = df, x = "latitude", y = "longitude", hue = "median_house_value")
```

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

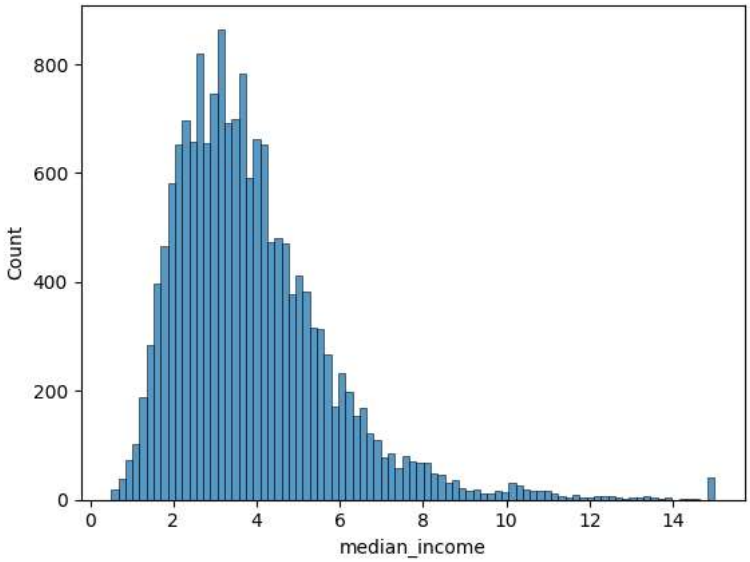


ГИСТОГРАММА

Способ представления табличных данных в грайическом виде - в виде столбчатой диаграммы. По оси x обычно указывают значение, а по оси y - встречаемость (кол-во таких значений в выборке)

```
sns.histplot(data = df, x = "median_income")
```

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

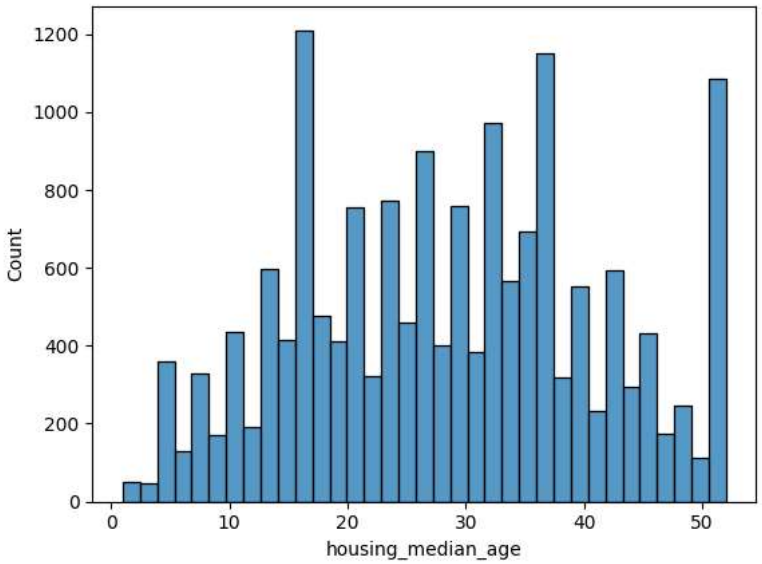


Можно видеть, что у большинства семей доход находится между значениями 2 и 6. И только очень небольшое количество людей обладают доходом > 10.

Изобразим гистограмму по housing_median_age:

```
sns.histplot(data = df, x = "housing_median_age")
```

```
<Axes: xlabel='housing_median_age', ylabel='Count'>
```

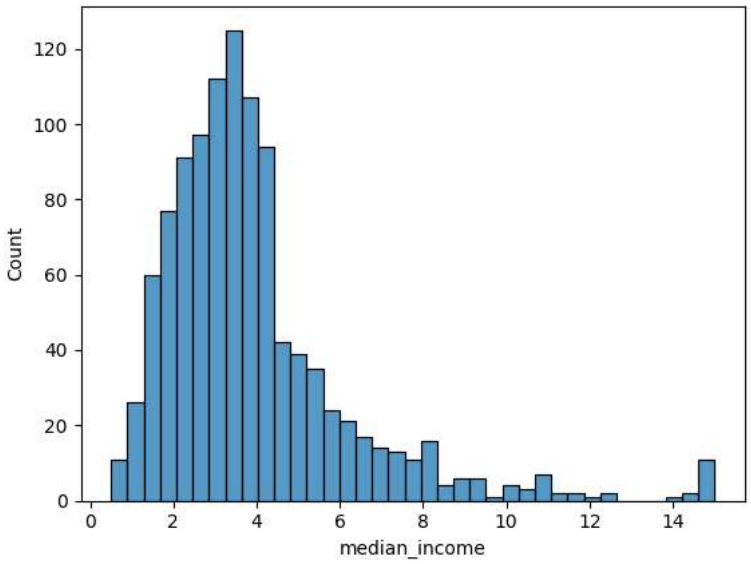


Распределение по фозрасту более равномерное. Большую часть жителей составляют люди в возрасте от 20 до 40 лет. Но и молодежи не мало. Также очень много пожилых людей > 50 лет медианный возраст.

Давайте посмотрим медианный доход у пожилых жителей:

```
sns.histplot(data = df[df['housing_median_age'] > 50], x = "median_income")
```

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



Большого отличия от популяциии в целом не наблюдается. Скорее всего это местные жители.

Давайте разобьем возрастные группы на 3 категории: те кто моложе 20, от 20 до 50 и от 50, чтобы посмотреть влияет ли это на доход:

```
df.loc[df['housing_median_age'] <= 20, 'age_group'] = 'Молодые'
df.loc[(df['housing_median_age'] > 20) & (df['housing_median_age'] <= 50), 'age_group'] = 'Средний возраст'
df.loc[df['housing_median_age'] > 50, 'age_group'] = 'Пожилые'
```

Что в этом случае происходит внутри таблицы? Добавился новый столбец age_group, в котором будет указанасоответствующая категория.

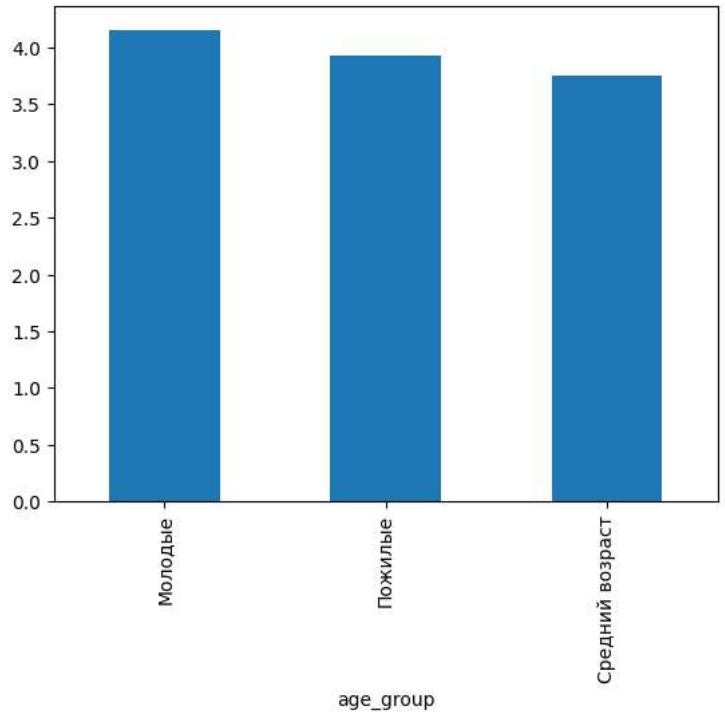
```
df.columns
```

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

Применим **groupby**, чтобы получить среднее значение

```
df.groupby('age_group')['median_income'].mean().plot(kind = 'bar')
```

```
<Axes: xlabel='age_group'>
```



Молодые оказываются самой богатой группой населения. Но отличие в доходе не значительное.

Seaborn так же позволяет нам смотреть распределение по многим параметрам. Давайте поделим группы по доходам на 2. Те, у кого