Исследование объявлений о продаже квартир

В вашем распоряжении данные сервиса Яндекс.Недвижимость — архив объявлений о продаже квартир в Санкт-Петербурге и соседних населённых пунктов за несколько лет. Нужно научиться определять рыночную стоимость объектов недвижимости. Ваша задача — установить параметры. Это позволит построить автоматизированную систему: она отследит аномалии и мошенническую деятельность.

По каждой квартире на продажу доступны два вида данных. Первые вписаны пользователем, вторые — получены автоматически на основе картографических данных. Например, расстояние до центра, аэропорта, ближайшего парка и водоёма.

Откройте файл с данными и изучите общую информацию.

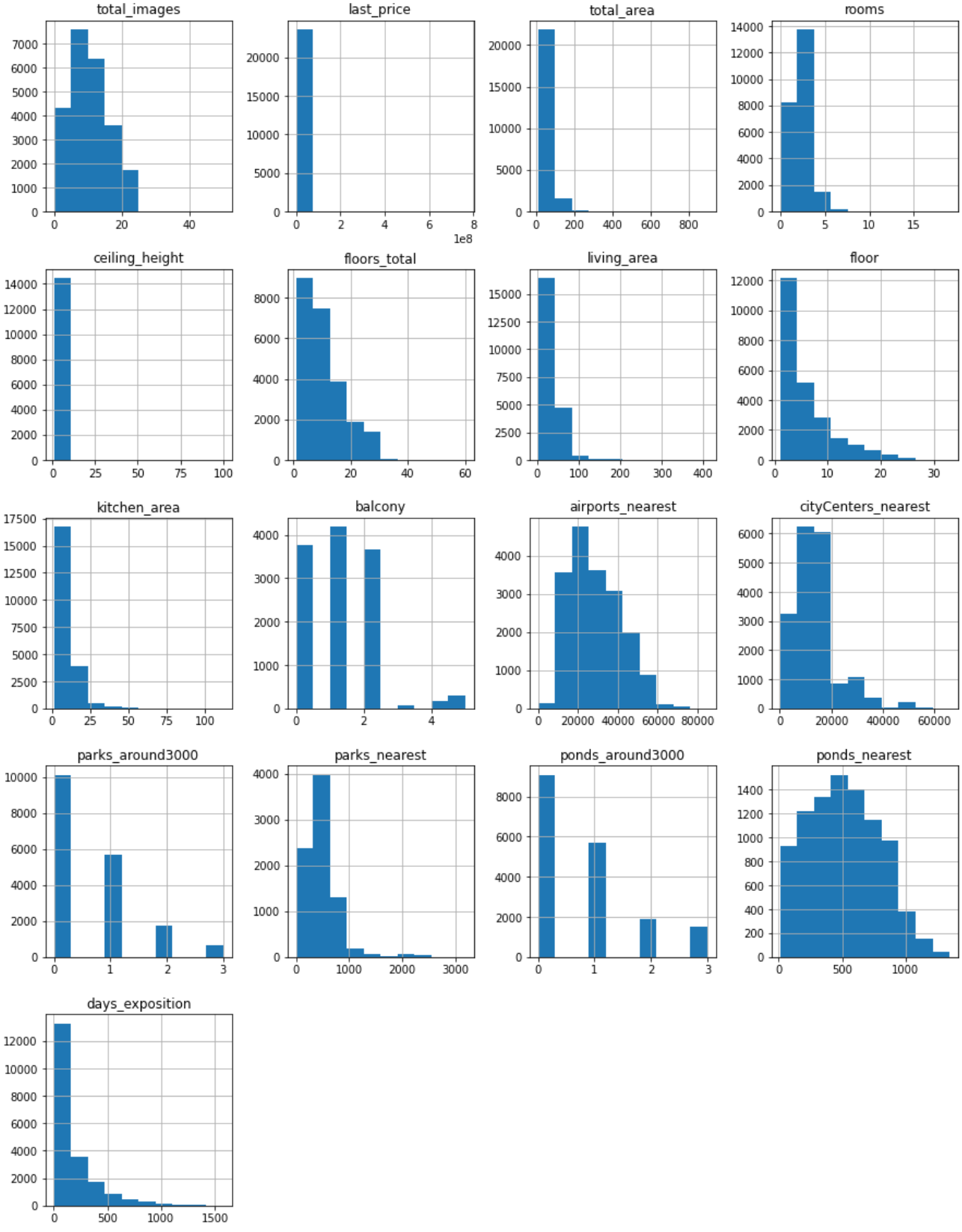
```
import pandas as pd
import matplotlib.pyplot as plt


data = pd.read_csv('/datasets/real_estate_data.csv',  sep='\t')
data.info() #выводим общую информацию
data.head() #выводим первые пять строк таблицы
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   total_images          23699 non-null  int64  
 1   last_price            23699 non-null  float64
 2   total_area            23699 non-null  float64
 3   first_day_exposition  23699 non-null  object 
 4   rooms                 23699 non-null  int64  
 5   ceiling_height        14504 non-null  float64
 6   floors_total          23613 non-null  float64
 7   living_area           21796 non-null  float64
 8   floor                 23699 non-null  int64  
 9   is_apartment          2775 non-null   object 
 10  studio                23699 non-null  bool   
 11  open_plan             23699 non-null  bool   
 12  kitchen_area          21421 non-null  float64
 13  balcony               12180 non-null  float64
 14  locality_name         23650 non-null  object 
 15  airports_nearest      18157 non-null  float64
 16  cityCenters_nearest   18180 non-null  float64
 17  parks_around3000      18181 non-null  float64
 18  parks_nearest         8079 non-null   float64
 19  ponds_around3000      18181 non-null  float64
 20  ponds_nearest         9110 non-null   float64
 21  days_exposition       20518 non-null  float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 3.7+ MB
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70
1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN
2	10	5196000.0	56.0	2015-08-20T00:00:00	2	NaN
3	0	64900000.0	159.0	2015-07-24T00:00:00	3	NaN
4	2	10000000.0	100.0	2018-06-19T00:00:00	2	3.03

5 rows × 22 columns

```
data.hist(figsize=(15, 20))
plt.show()
```


total_images
last_price
1e8
total_area
rooms
ceiling_height
floors_total
living_area
floor
kitchen_area
balcony
airports_nearest
cityCenters_nearest
parks_around3000
parks_nearest
ponds_around3000
ponds_nearest
days_exposition

Предобработка данных

```
data.isna().sum() # определяем количество пропусков в каждом столбце
```

```
total_images                0
last_price                  0
total_area                  0
first_day_exposition        0
rooms                       0
ceiling_height           9195
floors_total               86
living_area              1903
floor                       0
is_apartment            20924
studio                      0
open_plan                   0
kitchen_area             2278
balcony                 11519
locality_name              49
airports_nearest         5542
cityCenters_nearest      5519
parks_around3000         5518
parks_nearest           15620
ponds_around3000         5518
ponds_nearest           14589
days_exposition          3181
dtype: int64
```

Высота потолка ceiling_height

> В 9195 строках не указана высота потолка. Можно предположить, что это квартиры с так называемым "стандартным" потолком, т.к если бы потолки были высокими, то, скорее всего, это было бы указано. Соответственно, заменим отстуствующие значения на среднюю высоту потолка. Воспользуемся медианой, чтобы не включать в среднее значение экстримальные показатели.

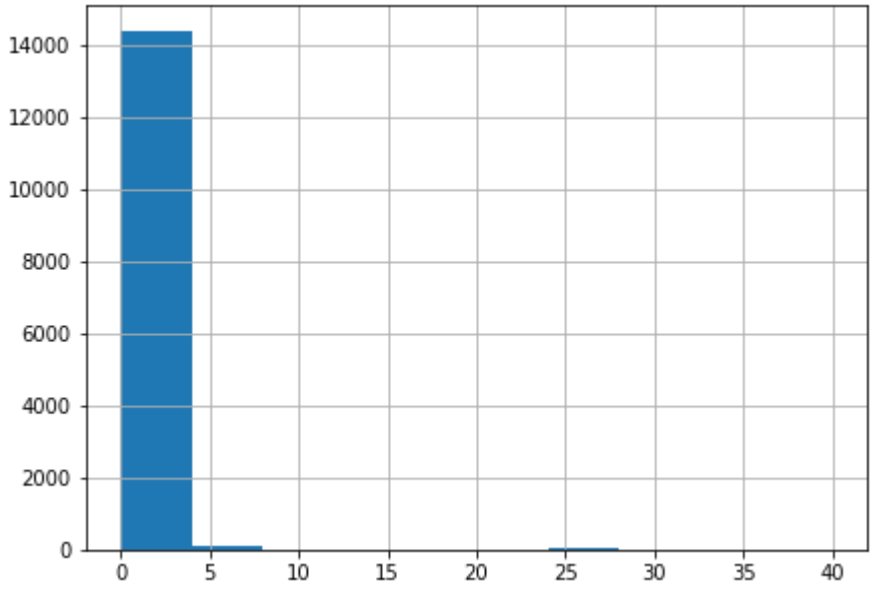
```
data['ceiling_height'].describe()
```

```
count    14504.000000
mean         2.771499
std          1.261056
min          1.000000
25%          2.520000
```

```
50%          2.650000
75%          2.800000
max        100.000000
Name: ceiling_height, dtype: float64
```

Видим очевидные аномальные значения минимума 1 м. и максимума 100 м.

```
data['ceiling_height'].hist(bins = 10, figsize = (7,5), range = (0,40));
```



Видим, что большая часть потолков 2-4,5 м. Есть значения в районе 25ти. Скорее всего это неверно записанные данные. Проверим

```
data.query('ceiling_height>20')
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_
355	17	3600000.0	55.2	2018-07-12T00:00:00	2	
3148	14	2900000.0	75.0	2018-11-12T00:00:00	3	
4643	0	4300000.0	45.0	2018-02-01T00:00:00	2	
4876	7	3000000.0	25.0	2017-09-27T00:00:00	0	
5076	0	3850000.0	30.5	2018-10-03T00:00:00	1	
5246	0	2500000.0	54.0	2017-10-13T00:00:00	2	
5669	4	4400000.0	50.0	2017-08-08T00:00:00	2	
5807	17	8150000.0	80.0	2019-01-09T00:00:00	2	
6246	6	3300000.0	44.4	2019-03-25T00:00:00	2	
9379	5	3950000.0	42.0	2017-03-26T00:00:00	3	
10773	8	3800000.0	58.0	2017-10-13T00:00:00	2	
11285	0	1950000.0	37.0	2019-03-20T00:00:00	1	
14382	9	1700000.0	35.0	2015-12-04T00:00:00	1	
17857	1	3900000.0	56.0	2017-12-22T00:00:00	3	
18545	6	3750000.0	43.0	2019-03-18T00:00:00	2	
20478	11	8000000.0	45.0	2017-07-18T00:00:00	1	
20507	12	5950000.0	60.0	2018-02-19T00:00:00	2	
21377	19	4900000.0	42.0	2017-04-18T00:00:00	1	
21824	20	2450000.0	44.0	2019-02-12T00:00:00	2	
22336	19	9999000.0	92.4	2019-04-05T00:00:00	2	
22869	0	15000000.0	25.0	2018-07-25T00:00:00	1	
22938	14	4000000.0	98.0	2018-03-15T00:00:00	4	

22 rows × 22 columns

Действительно похоже, что кроме 100, в значениях просто потеряна запятая. Исправим

```
data.loc[data['ceiling_height'] > 20, 'ceiling_height'] /= 10
```

```
data.query('ceiling_height>20')#проверим
```

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height

0 rows × 22 columns

```
data.query('ceiling_height >= 6 or ceiling_height <= 2 ').count()
```

```
total_images            23
last_price              23
total_area              23
first_day_exposition    23
rooms                   23
ceiling_height          23
floors_total            23
living_area             21
floor                   23
is_apartment             3
studio                  23
open_plan               23
kitchen_area            19
balcony                  9
locality_name           23
airports_nearest        13
cityCenters_nearest     13
parks_around3000        13
parks_nearest            8
ponds_around3000        13
ponds_nearest            8
days_exposition         19
dtype: int64
```

23 строки со значением потолка выше 6 метров и ниже 2 м. Это меньше одного процента. Удалим их. Пропуски оставим

```
data = data.loc[(data['ceiling_height'] <= 6)&(data['ceiling_height'] >= 2)|(data['ceili
```

Количество этажей в доме floors_total

> Скорее всего значения в этих строках отсутствуют потому, что эти дома одноэтажные. Поэтому заменим значения ячеек на 1. В любом случае, эти строки не должны сильно повлиять на результат исследования, т.к их количество незначительно (86 строк)

```
data['floors_total'] = data['floors_total'].fillna(1)#заменяем значения NAN на 1 в столбі
data['floors_total'] = data['floors_total'].astype(int)#поменяем тип данных на целочислеі
```

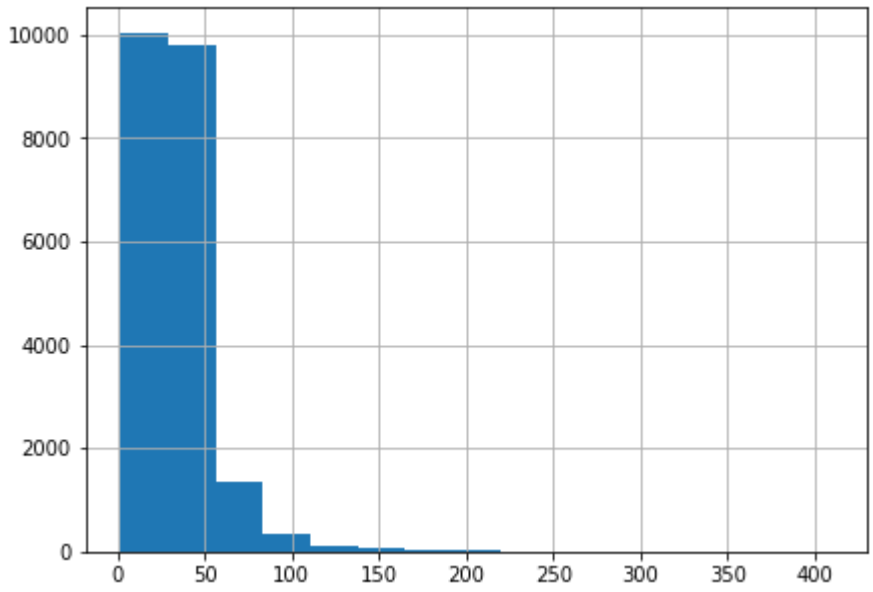
Жилая площадь living_area

> Выбирая квартиру, покупатели чаще ориентируются на общую площадь квартиры, а не на жилую площадь. Поэтому пока оставим ячейки как есть.

```
data[['living_area']].apply (['count', 'min', 'max']).style.format("{:,.2f}")
```

	living_area
count	21,785.00
min	2.00
max	409.70

```
data['living_area'].hist(bins = 15, figsize = (7,5));
```



Количество квартир с жилой площадью больше 130 кв.м. резко снижается. Посчитаем сколько их

```
data.query('living_area > 130').count()
```

```
total_images            176
last_price              176
total_area              176
first_day_exposition    176
rooms                   176
ceiling_height          118
floors_total            176
living_area             176
floor                   176
is_apartment             15
studio                  176
open_plan               176
kitchen_area            164
balcony                  86
locality_name           176
airports_nearest        171
cityCenters_nearest     174
parks_around3000        174
parks_nearest           124
ponds_around3000        174
ponds_nearest           128
days_exposition         137
dtype: int64
```

176 строк это меньше 1%, можно удалить эти строки,как редкие.

```
data = data.loc[(data['living_area'] < 130)|(data['living_area'].isna())]
```

Количество балконов balcony

> Отсутствующие значения количества балконов заменим на 0, т.к. скорее всего они не указаны по причине отсутствия балкона в квартире.

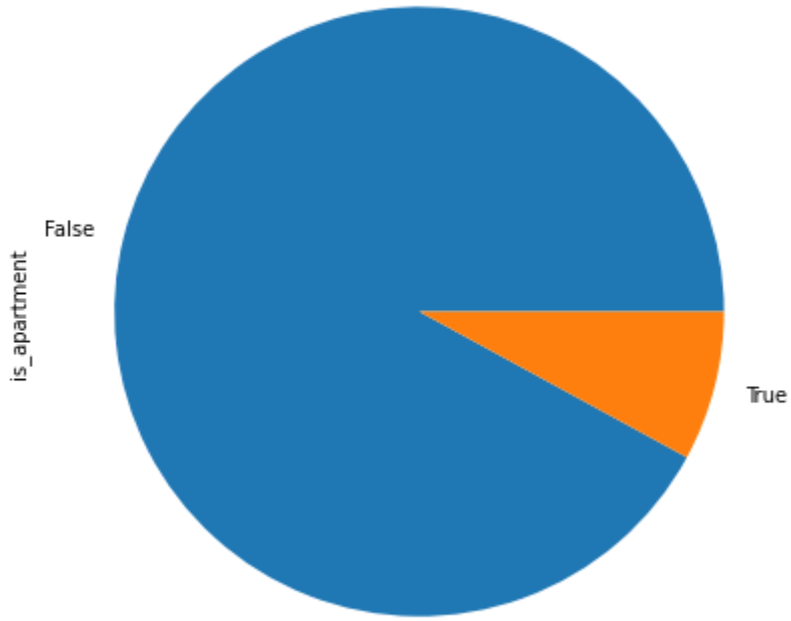
```
data['balcony'] = data['balcony'].fillna(0) #заменяем значения NAN на 0 в столбце количе
data['balcony'] = data['balcony'].astype(int) #заменим тип данных на целочисленный.
```

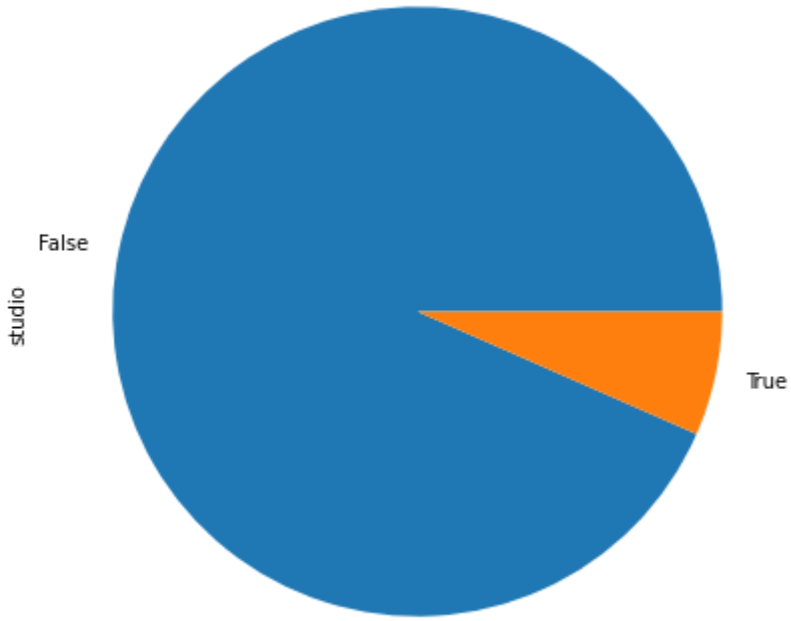
Площадь kitchen_area

> Пока не хватает данных, чтобы сделать замену отсутстсвующих значений по площади кухни. Ни среднее арифметическое, ни медиана не подходят, т.к все квартиры отличаются площадью и брать средеее просто некорректно. Можем предположить, что площадь кухни не указана в апартаментах и студиях, т.к она объединина с жилой и общей площадью.

Проверим это предположение, построив графики зависимости отсутствия значения о площади кухни в апартаментах и студиях

```
no_kitch_sq = data[data['kitchen_area'].isnull()] #вводим переменную со значением NaN в с
#строим график, проверяем в скольких апартаментах не указана площадь кухни
no_kitch_sq['is_apartment'].value_counts().plot(kind='pie', figsize = (7,7))
plt.show() #убираем служебную информацию
#строим график, проверяем в скольких студиях не указана площадь кухни
no_kitch_sq['studio'].value_counts().plot(kind='pie', figsize = (7,7))
plt.show()
```

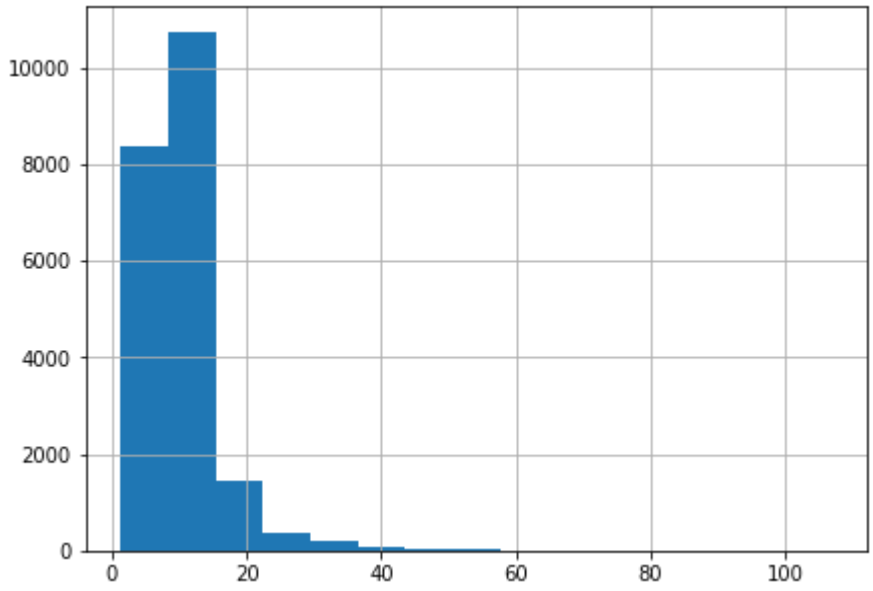





```
data[['kitchen_area']].apply (['count', 'min', 'max']).style.format("{:,.2f}")
```

	kitchen_area
count	21,244.00
min	1.30
max	107.00

```
data['kitchen_area'].hist(bins = 15, figsize = (7,5));
```



Значения больше 50 довольно редки, посчитаем их

```
data.query('kitchen_area > 50').count()
```

```
total_images            35
last_price              35
total_area              35
first_day_exposition    35
rooms                   35
ceiling_height          25
floors_total            35
living_area             33
floor                   35
is_apartment             5
studio                  35
open_plan               35
kitchen_area            35
balcony                 35
locality_name           35
airports_nearest        34
cityCenters_nearest     34
parks_around3000        34
parks_nearest           24
ponds_around3000        34
ponds_nearest           20
days_exposition         28
dtype: int64
```

35 строк. Можем удалить, как редко встречающиеся

```
data = data.loc[(data['kitchen_area'] < 50)|(data['kitchen_area'].isna())]
```

По графикам видно, что отсутствие данных о площади кухни никак не зависит от того квартира это, апартаменты или студия. Оставим значения ячеек как есть.

Город locality_name

> 49 строк с пропущенными значениями местоположения квартир удалим из датафрейма. Т.к расположение является одним из главных критереев выбора жилья, строки где этих значений нет, не имеют смысла.

```
data.loc[data['locality_name'].isna() == True] #посмотрим на данные без указания населени
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_
1097	3	8600000.0	81.70	2016-04-15T00:00:00	3	
2033	6	5398000.0	80.00	2017-05-30T00:00:00	3	
2603	20	3351765.0	42.70	2015-09-20T00:00:00	1	
2632	2	5130593.0	62.40	2015-10-11T00:00:00	2	
3574	10	4200000.0	46.50	2016-05-28T00:00:00	2	
4151	17	17600000.0	89.50	2014-12-09T00:00:00	2	
4189	7	9200000.0	80.00	2015-12-10T00:00:00	3	
4670	1	5500000.0	83.00	2015-08-14T00:00:00	3	
5343	19	13540000.0	85.50	2016-01-20T00:00:00	3	
5707	7	3700000.0	30.00	2016-04-29T00:00:00	1	
6765	20	4895892.0	60.70	2015-03-12T00:00:00	2	
7114	5	4250000.0	56.00	2016-03-16T00:00:00	3	
7330	8	5100000.0	63.00	2015-01-27T00:00:00	3	
7600	8	6800000.0	70.00	2016-01-31T00:00:00	3	
8568	10	16000000.0	155.00	2016-05-09T00:00:00	3	
8986	10	4850000.0	103.10	2018-07-10T00:00:00	3	
9821	13	8000000.0	94.50	2015-01-21T00:00:00	4	
10122	5	8200000.0	83.00	2015-06-24T00:00:00	4	
11248	12	6300000.0	63.10	2015-01-16T00:00:00	4	
12879	12	4400000.0	39.20	2016-04-26T00:00:00	1	
12936	6	6800000.0	73.00	2015-11-01T00:00:00	3	
13223	1	2919911.0	29.40	2015-03-12T00:00:00	1	
13690	7	3500000.0	71.00	2016-06-23T00:00:00	3	
14273	2	4422000.0	60.00	2016-03-23T00:00:00	2	
14342	3	3611000.0	53.50	2017-04-27T00:00:00	1	
15686	13	4700000.0	44.00	2015-12-01T00:00:00	2	
15866	10	3950000.0	44.00	2016-04-16T00:00:00	2	
16499	2	4995573.0	56.90	2016-06-17T00:00:00	2	
16561	3	2450000.0	30.00	2016-06-02T00:00:00	1	
16610	11	11940000.0	112.00	2015-11-19T00:00:00	3	
17535	2	5985000.0	79.80	2018-07-30T00:00:00	3	
17764	9	8400000.0	94.00	2016-01-24T00:00:00	3	

18526	3	10800000.0	86.00	2016-06-24T00:00:00	4
18917	3	2660000.0	37.99	2017-08-17T00:00:00	1
19045	6	4650000.0	48.00	2016-01-25T00:00:00	2
19972	20	4361004.0	62.40	2015-09-20T00:00:00	2
20057	13	11500000.0	102.00	2015-10-14T00:00:00	2
20382	8	1750000.0	72.90	2018-10-27T00:00:00	3
20590	7	3380000.0	56.00	2017-11-06T00:00:00	2
20654	7	6100000.0	43.00	2016-01-13T00:00:00	1
21119	8	3500000.0	43.20	2018-11-11T00:00:00	2
21276	0	17122148.0	178.30	2017-02-10T00:00:00	1
21333	10	5900000.0	58.00	2015-03-12T00:00:00	3
21715	2	6047550.0	80.10	2018-07-30T00:00:00	2
21898	2	5886750.0	83.50	2018-07-30T00:00:00	2
22474	7	24000000.0	128.00	2015-07-24T00:00:00	4
22717	9	3000000.0	35.00	2018-01-02T00:00:00	1
22933	20	3176015.0	33.30	2015-04-22T00:00:00	1
23214	3	7990000.0	56.00	2016-05-31T00:00:00	2

49 rows × 22 columns

```
data = data.dropna(subset = ['locality_name']) #ничего выделяющегося на первый взгляд в э
```

Расстояние до аэропорта airports_nearest, расстояние до ценнтра cityCenters_nearest

> Отсутствующие значения расстояния до аэропорта и до центра города имеет смысл заменить на медианные значения, сгруппировав по населенному пункту

```
#for row in data['locality_name'].unique():
 #   data.loc[(data['locality_name'] == row) & (data['airports_nearest'].isna()), 'airpo
  #  data.loc[(data['locality_name'] == row), 'airports_nearest'].median()
```

При проверке, получаем предупреждение от Python, что заставляет задуматься, а есть ли аэропорт в тех локациях, где ячейки со значением NaN. Т.к. оценить наверняка очень сложно, будем отталкиваться от того, что аэропорт находится в Санкт-Петербурге. Проверим, есть ли незаполненные ячейки с локацией там.

```
data.query('airports_nearest.isna() and airports_nearest == "Санкт-Петербург"')
```

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height

0 rows × 22 columns

Нет. Таких строк нет. Будем иметь в виду, что если значение не заполнено, значит аэропорт где-то далеко. Значения менять не будем.

Апартаменты is_apartment

> Заменим все недостающие знгачения на False

```
pd.set_option('mode.chained_assignment', None)
#уберем предупреждение, связанное с особенностью библиотеки, оно не влияет на результат
data['is_apartment'] = data['is_apartment'].fillna(False)
```

Парки поблизости parks_around3000, водоемы поблизости ponds_around3000

> Заменим эти значения на 0, т.к. скорее всего парков и водоемов поблизости просто нет

```
data['parks_around3000'] = data['parks_around3000'].fillna(0)
data['ponds_around3000'] = data['ponds_around3000'].fillna(0)
```

Расстояние до ближайшего парка parks_nearest, расстояние до ближайшего водоема ponds_nearest

> Замениить отсутствующие значения на 0 мы не можем, т.к. в этом случае получится, что квартира находится в непосредственной близости от парка/водоема. Поэтому эти значения оставим как есть/

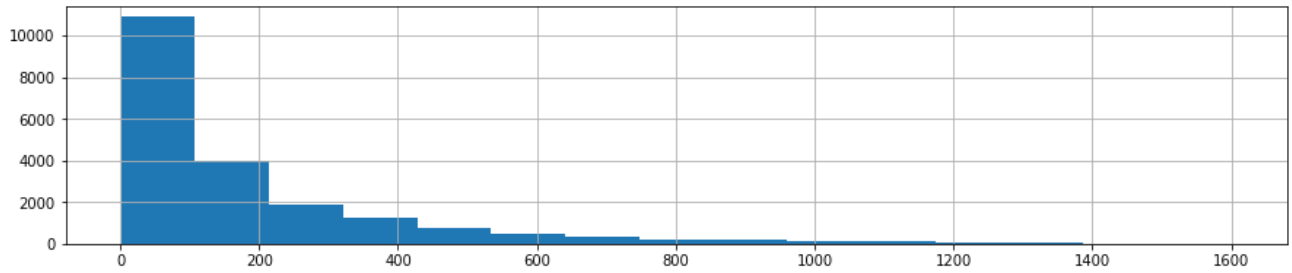
Количество дней, на которе было размещено объявление days_exposition

> Пропуски в ['days_exposition'] говорят нам, что квартиры еще не проданы. Оставляем.

```
data[['days_exposition']].apply (['count', 'min', 'max']).style.format("{:,.2f}")
```

	days_exposition
count	20,285.00
min	1.00
max	1,580.00

```
data.days_exposition.hist(bins = 15, figsize = (15,3), range = (1,1600))
plt.show()
```



Видим низкое количесвто значений после 1150. Проверим сколько это строк

```
data[data['days_exposition'] > 1000].count()
```

```
total_images            266
last_price              266
total_area              266
first_day_exposition    266
rooms                   266
ceiling_height          180
floors_total            266
living_area             242
floor                   266
is_apartment            266
studio                  266
open_plan               266
kitchen_area            251
balcony                 266
locality_name           266
airports_nearest        225
cityCenters_nearest     225
parks_around3000        266
parks_nearest           112
ponds_around3000        266
ponds_nearest           145
days_exposition         266
dtype: int64
```

256 строк - это около 1%, можем удалить

```
data = data.loc[(data['days_exposition'] < 1000)|(data['days_exposition'].isna())]
```

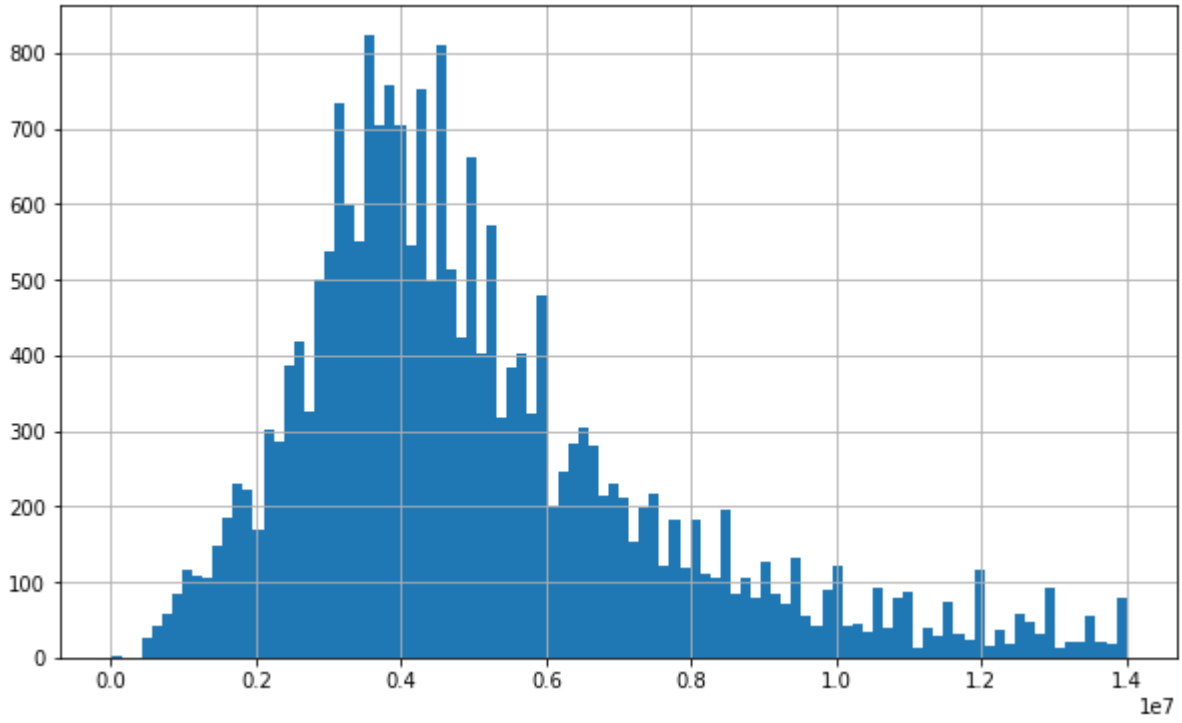
Цена

```
data['last_price'] = data['last_price'].astype(int)
```

```
data[['last_price']].apply (['count', 'min', 'max']).style.format("{:,.2f}")
```

	last_price
count	23,149.00
min	12,190.00
max	330,000,000.00

```
data.last_price.hist(bins = 100, figsize = (10,6), range = (0, 1.4e+07))
plt.show()
```




```
data['last_price'].describe()
```

```
count    2.314900e+04
mean     6.032660e+06
std      6.555309e+06
min      1.219000e+04
25%      3.400000e+06
50%      4.600000e+06
75%      6.650000e+06
max      3.300000e+08
Name: last_price, dtype: float64
```

```
data[data['last_price'] > 3.0e+07].count() # квартиры стоимостью выше 30 млн
```

```
total_images            190
last_price              190
```

```
total_area              190
first_day_exposition    190
rooms                   190
ceiling_height          124
floors_total            190
living_area             155
floor                   190
is_apartment            190
studio                  190
open_plan               190
kitchen_area            163
balcony                 190
locality_name           190
airports_nearest        184
cityCenters_nearest     184
parks_around3000        190
parks_nearest           138
ponds_around3000        190
ponds_nearest           148
days_exposition         136
dtype: int64
```

```
data = data.loc[(data['last_price'] < 3.0e+07)|(data['last_price'].isna())]
```

Дата публикации first_day_exposition

> В столбце Дата публикации поменяем тип данных на datetime и проверим

```
data['first_day_exposition'] = pd.to_datetime(data['first_day_exposition'], format = '%Y
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22952 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   total_images          22952 non-null  int64
 1   last_price            22952 non-null  int64
 2   total_area            22952 non-null  float64
 3   first_day_exposition  22952 non-null  datetime64[ns]
 4   rooms                 22952 non-null  int64
 5   ceiling_height        14015 non-null  float64
 6   floors_total          22952 non-null  int64
 7   living_area           21115 non-null  float64
 8   floor                 22952 non-null  int64
 9   is_apartment          22952 non-null  bool
 10  studio                22952 non-null  bool
 11  open_plan             22952 non-null  bool
 12  kitchen_area          20739 non-null  float64
 13  balcony               22952 non-null  int64
 14  locality_name         22952 non-null  object
 15  airports_nearest      17475 non-null  float64
 16  cityCenters_nearest   17495 non-null  float64
 17  parks_around3000      22952 non-null  float64
 18  parks_nearest         7648 non-null   float64
```

```
 19  ponds_around3000      22952 non-null  float64
 20  ponds_nearest         8629 non-null   float64
 21  days_exposition       19877 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(11), int64(6), object(1)
memory usage: 3.6+ MB
```

Теперь разберемся с дубликатами.

```
duplicated_data  = data[data.duplicated()]
duplicated_data
```

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height

0 rows × 22 columns

Полных дубликатов нет. Поищем неявные дубликаты

```
data['locality_name'].unique()
```

```
       деревня Бор ',  поселок станции Свирь ',  поселок Перово ',  Высоцк ',
       'поселок Гарболово', 'село Шум', 'поселок Котельский',
       'поселок станции Лужайка', 'посёлок Стеклянный',
       'деревня Большая Пустомержа', 'поселок Красносельское',
       'деревня Вахнова Кара', 'деревня Пижма',
       'коттеджный поселок Кивеннапа Север', 'поселок Коробицыно',
       'поселок Ромашки', 'посёлок Перово', 'деревня Каськово',
       'деревня Куровицы', 'посёлок Плоское', 'поселок Сумино',
       'поселок городского типа Большая Ижора', 'поселок Кирпичное',
       'деревня Ям-Тесово', 'деревня Раздолье', 'деревня Терпилицы',
       'посёлок Шугозеро', 'деревня Ваганово', 'поселок Пушное',
       'садовое товарищество Садко', 'посёлок Усть-Ижора',
       'деревня Выскатка', 'городской посёлок Свирьстрой',
       'поселок Громово', 'деревня Кисельня', 'посёлок Старая Малукса',
       'деревня Трубников Бор', 'поселок Калитино',
       'посёлок Высокоключевой', 'садовое товарищество Приладожский',
       'посёлок Пансионат Зелёный Бор', 'деревня Ненимяки',
       'поселок Пансионат Зелёный Бор', 'деревня Снегирёвка',
       'деревня Рапполово', 'деревня Пустынка', 'поселок Рабитицы',
       'деревня Большой Сабск', 'деревня Русско', 'посёлок Лисий Нос',
       'деревня Лупполово', 'деревня Большое Рейзино',
       'деревня Малая Романовка', 'поселок Дружноселье', 'поселок Пчевжа',
       'поселок Володарское', 'деревня Нижняя',
       'коттеджный посёлок Лесное', 'деревня Тихковицы',
       'деревня Борисова Грива', 'посёлок Дзержинского'], dtype=object)
```

```
# check
data['locality_name'].nunique()
```

```
363
```

```
# Используем метод replace()
data['locality_name'] = data['locality_name'].str.replace('посёлок','поселок')
data['locality_name'] = (
    data['locality_name'].str.replace('поселок Мурино','Мурино')
    .str.replace('деревня Кудрово','Кудрово')
    .str.replace('поселок городского типа Рябово','Рябово')
)
```

```
#проверим
pd.set_option('display.max_columns', None) #строчки, чтобы Юпитер полностью вывел таблицу
pd.set_option('display.max_rows', None)
data['locality_name'].value_counts()
```

```
Санкт-Петербург                                    15109
Мурино                                               584
Кудрово                                              466
поселок Шушары                                       436
Всеволожск                                           394
Пушкин                                               356
Колпино                                              335
поселок Парголово                                    326
Гатчина                                              307
Выборг                                               236
Петергоф                                             195
Сестрорецк                                           181
Красное Село                                         175
```

```
деревня Новое Девяткино                          142
Сертолово                                        138
Ломоносов                                        132
Кириши                                           124
поселок Бугры                                    112
Сланцы                                           112
Волхов                                           111
Кингисепп                                        104
Тосно                                            104
Кронштадт                                         94
Никольское                                        93
Сосновый Бор                                      87
Коммунар                                          86
Кировск                                           84
Отрадное                                          79
городской поселок Янино-1                         68
поселок Металлострой                              66
Приозерск                                         66
деревня Старая                                    64
Шлиссельбург                                      56
Луга                                              56
Тихвин                                            49
поселок Стрельна                                  43
поселок Тельмана                                  40
Павловск                                          36
Волосово                                          36
поселок Романовка                                 36
поселок городского типа имени Свердлова          36
поселок городского типа Кузьмоловский            35
поселок городского типа Рощино                    34
поселок городского типа Сиверский                 29
Ивангород                                         28
городской поселок Новоселье                       28
городской поселок Мга                             26
Сясьстрой                                         24
Зеленогорск                                       24
поселок Щеглово                                   23
поселок Новый Свет                                22
поселок городского типа Синявино                  21
поселок городского типа Вырица                    21
деревня Вартемяги                                 20
поселок городского типа Токсово                   20
поселок Понтонный                                 20
поселок Новогорелово                              20
деревня Десколово                                 20
```

```
data['locality_name'].nunique()
```

```
327
```

Найдем аномальные значения. Выведем таблицу методом describe

```
data.describe()
```

	total_images	last_price	total_area	rooms	ceiling_height	floors_
count	22952.000000	2.295200e+04	22952.000000	22952.000000	14015.000000	22952.0
mean	9.844284	5.645989e+06	57.383432	2.019127	2.713378	10.6
std	5.653254	3.865348e+06	25.779422	0.983738	0.258572	6.6
min	0.000000	1.219000e+04	12.000000	0.000000	2.000000	1.0
25%	6.000000	3.400000e+06	40.000000	1.000000	2.500000	5.0
50%	9.000000	4.590000e+06	51.000000	2.000000	2.650000	9.0
75%	14.000000	6.550000e+06	68.000000	3.000000	2.800000	16.0
max	50.000000	2.999900e+07	441.980000	9.000000	5.800000	60.0

Если посмотреть на минимальные, максимальные значения, становятся очевидными выбросы в столбцах floors_total (максимальное значение 60 этажей, тогда как в Санкт-Петербурге самое высокое здание Лахта-центр имеет 35 этажей) rooms (минммальное значение 0)

```
data = data.loc[(data['rooms'] >= 1)|(data['rooms'].isna())]
                                    #отметаем нулевые значения количества комнат
```

```
data.query('floors_total > 35') #проверим много ли выбросов по этажности
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_he
397	15	5990000	54.0	2018-03-22	2	
2253	12	3800000	45.5	2018-06-28	2	
5807	17	8150000	80.0	2019-01-09	2	
11079	16	9200000	75.0	2019-02-22	2	
16731	9	3978000	40.0	2018-09-24	1	

6 строк имееют значение этажности здания более 35 эт. 36 и 37 этажей значение очень близкое к 35, поэтому их оставим. Вдруг где-то в пригороде и правда уже построили здание выше Лахта-центра. Строки со значением 52 и 60 явно ошибочные. Удалим их.

```
data = data.loc[(data['floors_total'] <= 37)|(data['last_price'].isna())]
#оставляем в таблице значения меньше 37 этажей
```

```
# check
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22754 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   total_images          22754 non-null  int64
 1   last_price            22754 non-null  int64
 2   total_area            22754 non-null  float64
 3   first_day_exposition  22754 non-null  datetime64[ns]
 4   rooms                 22754 non-null  int64
 5   ceiling_height        13932 non-null  float64
 6   floors_total          22754 non-null  int64
 7   living_area           20930 non-null  float64
 8   floor                 22754 non-null  int64
 9   is_apartment          22754 non-null  bool
 10  studio                22754 non-null  bool
 11  open_plan             22754 non-null  bool
 12  kitchen_area          20737 non-null  float64
 13  balcony               22754 non-null  int64
 14  locality_name         22754 non-null  object
 15  airports_nearest      17356 non-null  float64
 16  cityCenters_nearest   17376 non-null  float64
 17  parks_around3000      22754 non-null  float64
 18  parks_nearest         7618 non-null   float64
 19  ponds_around3000      22754 non-null  float64
 20  ponds_nearest         8560 non-null   float64
 21  days_exposition       19693 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(11), int64(6), object(1)
memory usage: 3.5+ MB
```

```
# check

# Показатели о кол-ве объявлений в датасете, минимальных и максимальных значениях
# в выбранных параметрах о продаже квартир
# сырые данные

(
    data[['rooms', 'total_area', 'ceiling_height', 'days_exposition', 'last_price', 'liv
          'floor', 'floors_total']]
    .apply (['count', 'min', 'max'])
    .style.format("{:,.2f}")
)
```

	rooms	total_area	ceiling_height	days_exposition	last_price	living_area	kitchen_area
count	22,754.00	22,754.00	13,932.00	19,693.00	22,754.00	20,930.00	20,737.00
min	1.00	12.00	2.00	1.00	12,190.00	2.00	1.30
max	9.00	441.08	5.30	999.00	29,999,000.00	138.00	49.40

```
# check
data.rooms.value_counts().to_frame()
```

	rooms
1	7959
2	7802
3	5610
4	1067
5	242
6	53
7	18
8	2
9	1

```
data.query('rooms > 7').count()
```

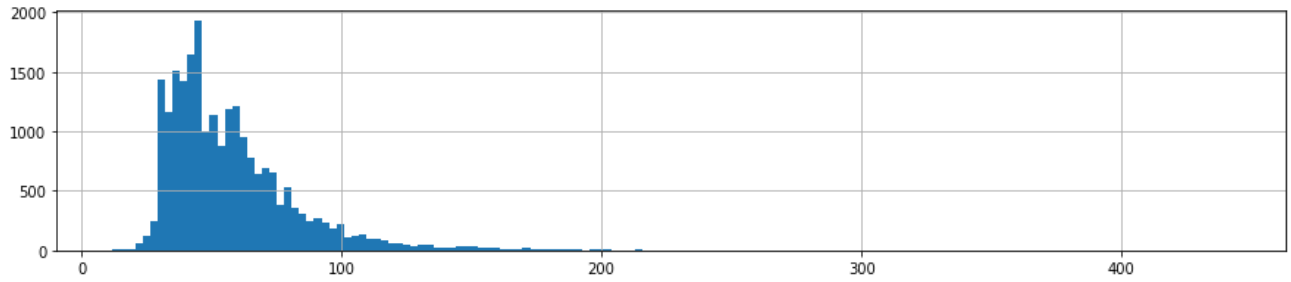
```
total_images            3
last_price              3
total_area              3
first_day_exposition    3
rooms                   3
ceiling_height          3
floors_total            3
living_area             2
floor                   3
is_apartment            3
studio                  3
open_plan               3
kitchen_area            2
balcony                 3
locality_name           3
airports_nearest        3
cityCenters_nearest     3
parks_around3000        3
parks_nearest           3
ponds_around3000        3
ponds_nearest           1
days_exposition         2
dtype: int64
```

```
data[['rooms']].apply (['count', 'min', 'max']).style.format("{:,.2f}")
```

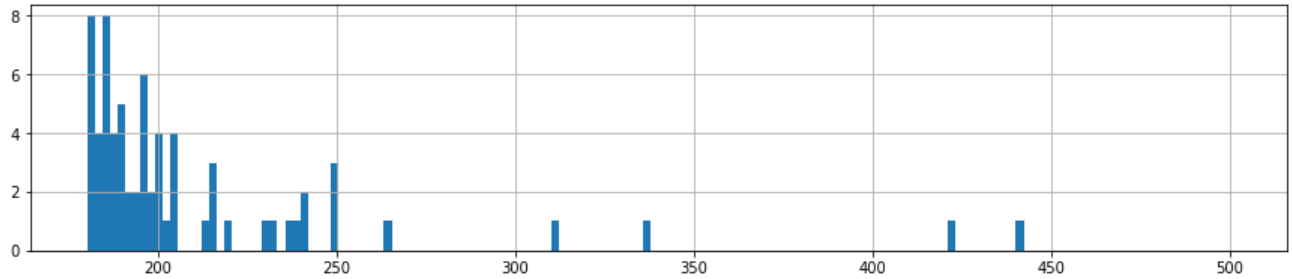
	rooms
count	22,754.00
min	1.00
max	9.00

3 строк - это почти ничего, можем их удалить

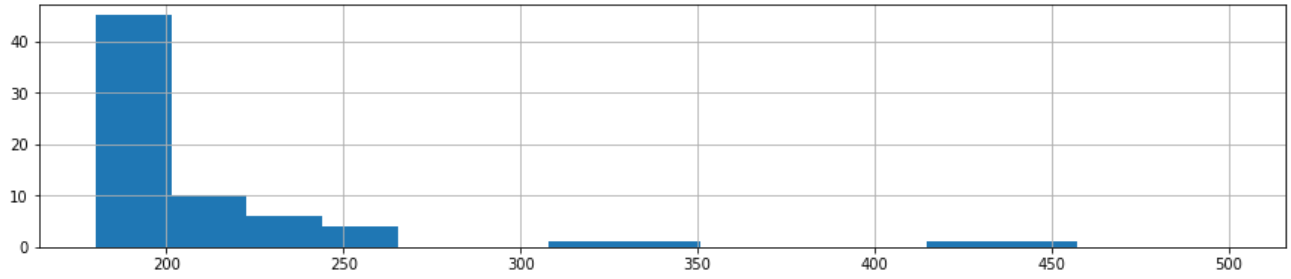
```
data = data.loc[(data['rooms'] <= 7)|(data['rooms'].isna())]


# check
data.total_area.hist(bins = 150, figsize = (15,3));
```




```
# check
data.total_area.hist(bins = 150, figsize = (15,3), range = (180,500));
```




```
# check
data.total_area.hist(bins = 15, figsize = (15,3), range = (180,500));
```




```
data.query('total_area > 250').count()
```

```
total_images           6
last_price             6
total_area             6
first_day_exposition   6
rooms                  6
ceiling_height         3
```

```
floors_total           6
living_area            3
floor                  6
is_apartment           6
studio                 6
open_plan              6
kitchen_area           5
balcony                6
locality_name          6
airports_nearest       6
cityCenters_nearest    6
parks_around3000       6
parks_nearest          2
ponds_around3000       6
ponds_nearest          2
days_exposition        4
dtype: int64
```

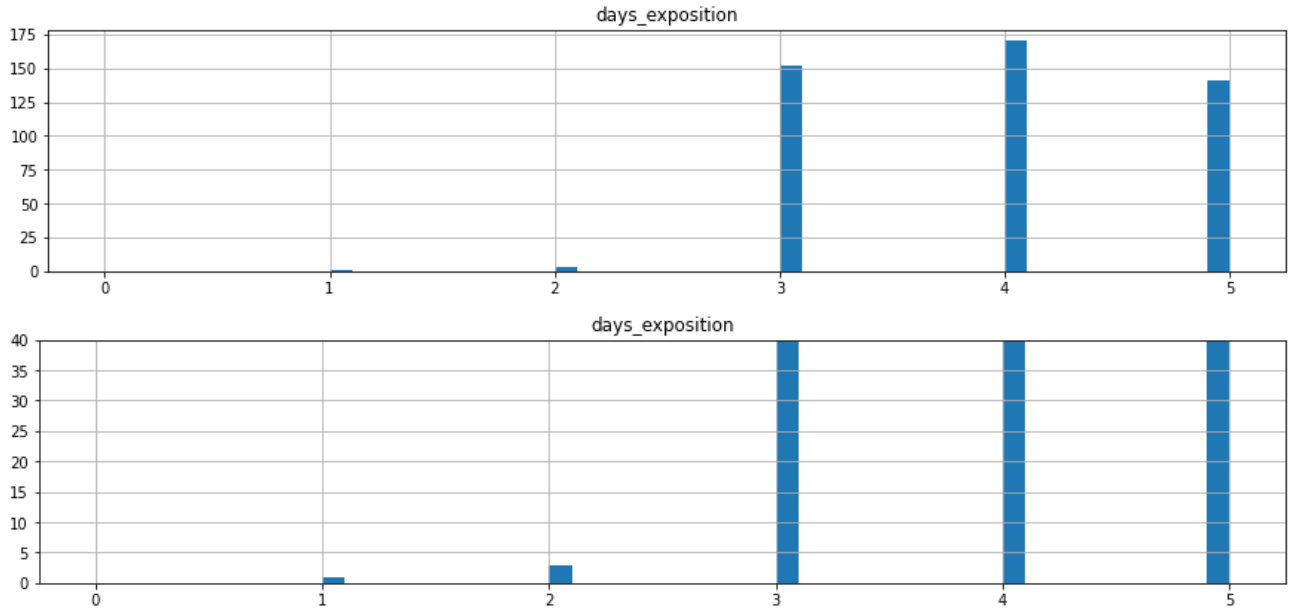
```
data = data.loc[(data['total_area'] <= 250)|(data['total_area'].isna())]


# check

# Значения параметров объектов недвижимости на разных квантилях

(
    data[['rooms', 'total_area', 'ceiling_height', 'days_exposition', 'last_price', 'liv
        'kitchen_area', 'floor',   'floors_total']]
    .quantile([0.0012, 0.01, .5, .99, .9988]) # выбираем размах в 0,9976 квантилей
    .style.format("{:,.2f}")
)
```

	rooms	total_area	ceiling_height	days_exposition	last_price	living_area	kitchen_area	fl
0.0012	1.00	20.62	2.30	3.00	560,000.00	10.00	3.67	1.
0.01	1.00	27.00	2.50	4.00	1,000,000.00	13.00	5.00	1.
0.5	2.00	51.40	2.65	93.00	4,600,000.00	30.00	9.00	4.
0.99	5.00	151.56	3.66	863.00	22,000,000.00	92.33	30.00	2:

```
# check

# Показатели о кол-ве объявлений в датасете, минимальных и максимальных значениях
# в выбранных параметрах о продаже квартир

(
    data[['rooms', 'total_area', 'ceiling_height', 'days_exposition', 'last_price', 'liv
          'floor', 'floors_total']]
    .apply (['count', 'min', 'max'])
    .style.format("{:,.2f}")
)
```

	rooms	total_area	ceiling_height	days_exposition	last_price	living_area	kitchen_area
count	22,745.00	22,745.00	13,926.00	19,687.00	22,745.00	20,925.00	20,730.00
min	1.00	12.00	2.00	1.00	12,190.00	2.00	1.30
max	7.00	250.00	5.30	999.00	29,999,000.00	128.00	49.40

```
# check
data.hist(column = 'days_exposition', bins = 50, figsize = (15,3), range = (0,5));

data.hist(column = 'days_exposition', bins = 50, figsize = (15,3), range = (0,5))
plt.ylim(0, 40);
```




```
data.query('days_exposition < 3').count()
```

```
total_images            4
last_price              4
total_area              4
first_day_exposition    4
rooms                   4
ceiling_height          3
floors_total            4
living_area             3
floor                   4
is_apartment            4
studio                  4
open_plan               4
kitchen_area            3
balcony                 4
locality_name           4
airports_nearest        4
cityCenters_nearest     4
parks_around3000        4
```

```
parks_nearest           2
ponds_around3000        4
ponds_nearest           1
days_exposition         4
dtype: int64
```

```
data = data.loc[(data['days_exposition'] > 3)|(data['days_exposition'].isna())]
```

Посчитайте и добавьте в таблицу новые столбцы

Посчитаем среднюю цену за квадратный метр и внесем данные в отдельный столбец.

```
data['price_m'] = data['last_price']/data['total_area']  # добавим столбец с ценой за ме
data['price_m'] = data['price_m'].astype('int') # поменяем тип данных на целочисленный
data.head() #проверим
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height
0	20	13000000	108.0	2019-03-07	3	2.70
1	7	3350000	40.4	2018-12-04	1	NaN
2	10	5196000	56.0	2015-08-20	2	NaN
4	2	10000000	100.0	2018-06-19	2	3.03
5	10	2890000	30.4	2018-09-10	1	NaN

Добавим новые столбцы со значениями дня недели, мексяца и года размещения объявления на сайте

```
data['weekday'] = data['first_day_exposition'].dt.weekday
#определяем день недели размещеничя объявления, помещаем в отдельный столбец
data['month'] = data['first_day_exposition'].dt.month # месяц
data['year'] = data['first_day_exposition'].dt.year # год
data.head()
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height
0	20	13000000	108.0	2019-03-07	3	2.70
1	7	3350000	40.4	2018-12-04	1	NaN
2	10	5196000	56.0	2015-08-20	2	NaN
4	2	10000000	100.0	2018-06-19	2	3.03
5	10	2890000	30.4	2018-09-10	1	NaN

Сделаем категоризацию этажей: "первый", "порследний", "другой"

```
# создаем функцию для категоризации типов этажей
def type_floor(data):
    if data['floor'] == 1:
        return 'первый'
    elif data['floor'] < data['floors_total']:
        return 'другой'
    elif data['floor'] == data['floors_total']:
        return 'последний'
    elif data['floor'] <= 0 :
        return 'ошибка'
    else:
        return 'не найдено'

data['type_floor'] = data.apply(type_floor,axis=1) # вызываем функцию и складываем значе
```

Переведем значения расстояния до центра города в километры и округлим до целого

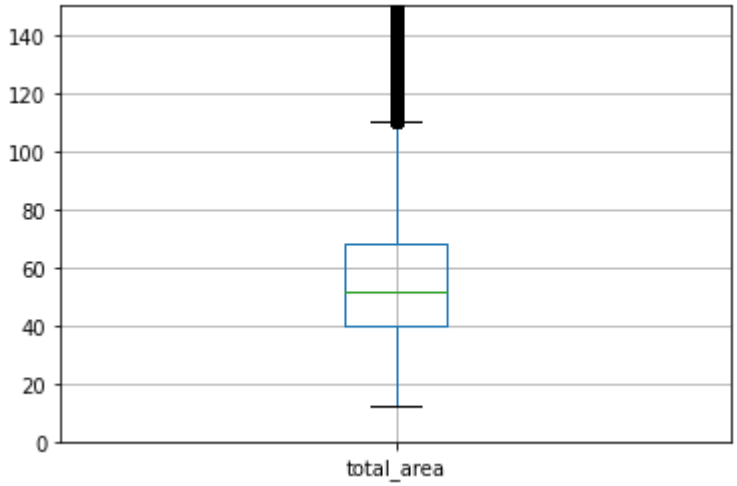
```
data['cityCenters_nearest'] = round(data['cityCenters_nearest'] /1000)
data.tail(10) #проверим
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_he
23689	13	3550000	35.30	2018-02-28	1	
23690	3	5500000	52.00	2018-07-19	2	
23691	11	9470000	72.90	2016-10-13	2	
23692	2	1350000	30.00	2017-07-07	1	
23693	9	4600000	62.40	2016-08-05	3	
23694	9	9700000	133.81	2017-03-21	3	
23695	14	3100000	59.00	2018-01-15	3	
23696	18	2500000	56.70	2018-02-11	2	
23697	13	11475000	76.75	2017-03-28	2	
23698	4	1350000	32.30	2017-07-21	1	

Проведем исследовательский анализ данных

Общая площадь

```
plt.ylim(0, 150) #выставляем значения по оси y
data.boxplot('total_area') # строим диаграмму размаха
plt.show()
```

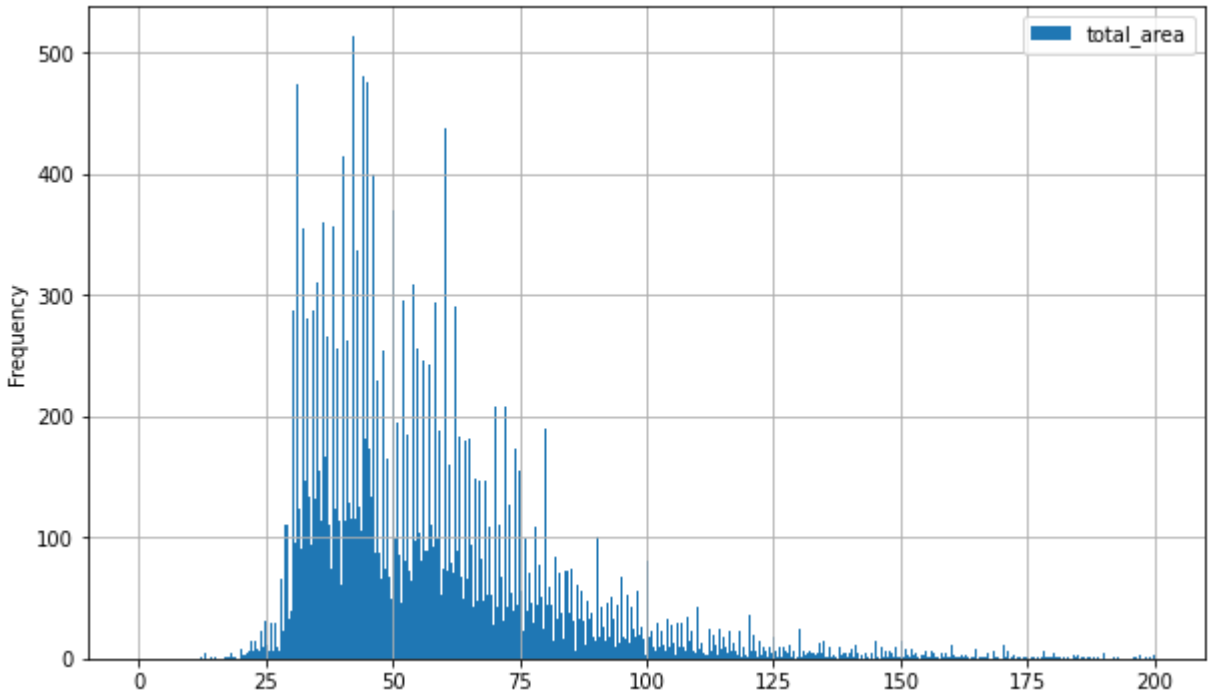


```
data['total_area'].describe()
```

```
count    22741.000000
mean        57.552796
std         25.252299
min         12.000000
25%         40.000000
50%         51.400000
75%         68.000000
max        250.000000
Name: total_area, dtype: float64
```

Стандартное отклонение в 2 раза ниже среднего. Это говорит о том, что значения набора данных имеют большой разброс. Это действительно так, учитывая, что у нас есть квартира 12 квадратов и 250 квадратов. По диаграмме размаха можем определить, что большинство квартир имеют площадь от 40 до 70 кв. м. - это значения, которые входят в "ящик". Минимальное значение 12 кв.м., верхний ус примерно на 110 кв.м. Остальные значения считаем выбросами. Медиана проходит примерно на 50

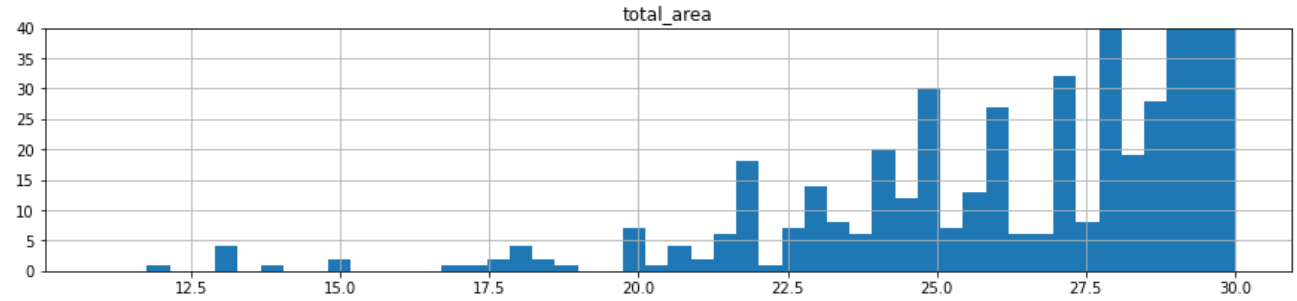
```
data.plot(y = 'total_area', kind = 'hist', bins = 500, grid=True, range = (0,200), figsi
plt.show()
```



Ориентируясь на гистограмму и диаграмму размаха возьмем значения выше 120 за выбросы

Построим гистограмму минимальных значений

```
data.hist(column = 'total_area', bins = 50, figsize = (15,3), range = (11,30))
plt.ylim(0, 40);
```




```
data.query('total_area < 20').count()
```

```
total_images            19
last_price              19
total_area              19
first_day_exposition    19
rooms                   19
ceiling_height          10
floors_total            19
living_area             11
floor                   19
is_apartment            19
studio                  19
open_plan               19
kitchen_area             4
balcony                 19
locality_name           19
airports_nearest        14
cityCenters_nearest     14
parks_around3000        19
parks_nearest            6
ponds_around3000        19
ponds_nearest            9
days_exposition         19
price_m                 19
weekday                 19
month                   19
year                    19
type_floor              19
dtype: int64
```

Т.к строк со значениями ниже 20 всего 19, можем взять 20 за минимальное значение

```
data = data.loc[(data['total_area'] >=20)&(data['total_area'] <=120)|(data['total_area']
```

Жилая площадь

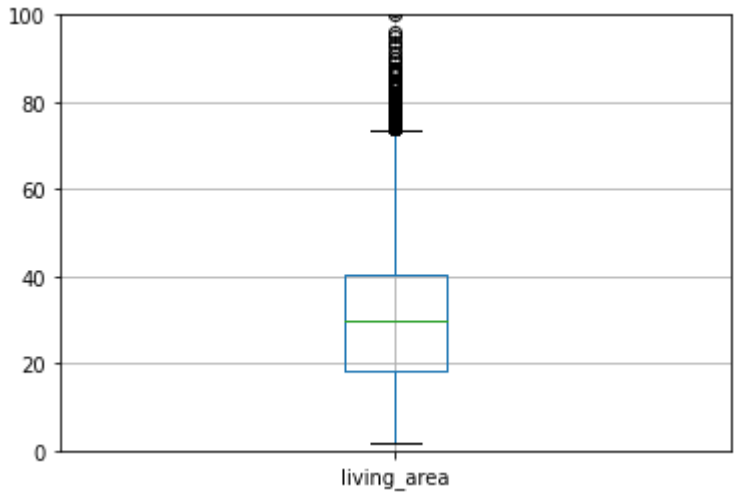
Проверим сразу нет ли значений жилой площади кухни превышающих значенияе общей площади квартиры

```
data.query( 'living_area > total_area or kitchen_area > total_area').count()
```

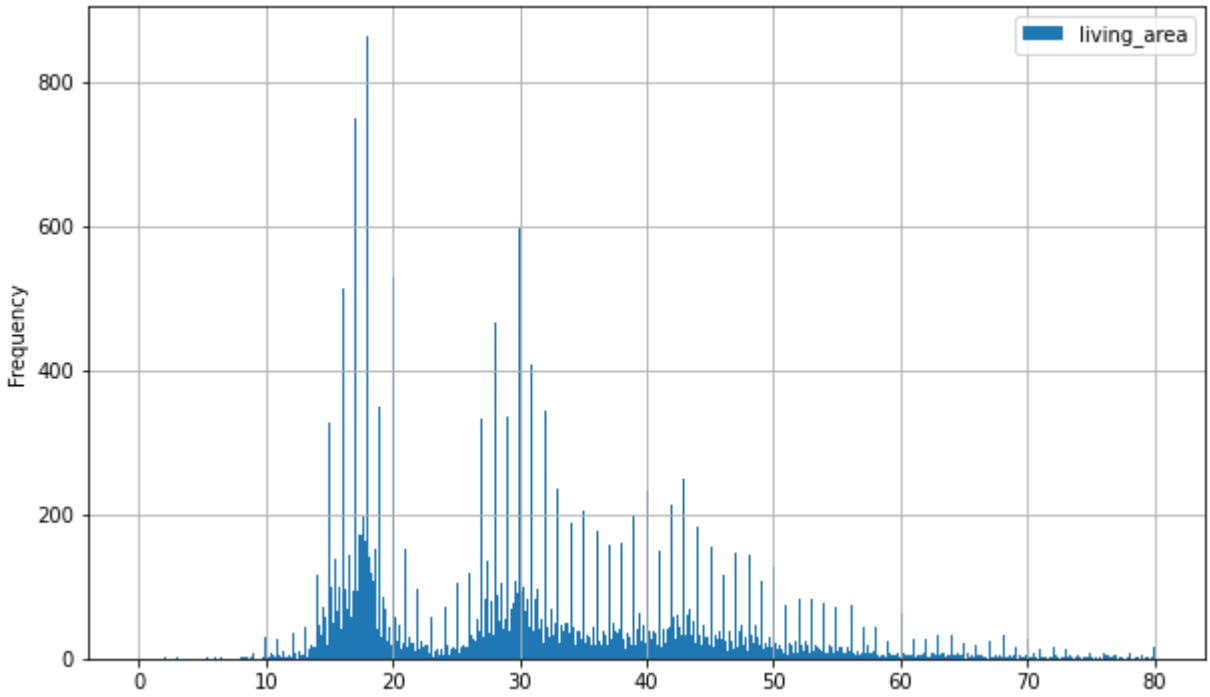
```
total_images            0
last_price              0
total_area              0
first_day_exposition    0
rooms                   0
ceiling_height          0
floors_total            0
living_area             0
floor                   0
is_apartment            0
studio                  0
open_plan               0
kitchen_area            0
balcony                 0
locality_name           0
airports_nearest        0
cityCenters_nearest     0
parks_around3000        0
parks_nearest           0
ponds_around3000        0
ponds_nearest           0
days_exposition         0
price_m                 0
weekday                 0
month                   0
year                    0
type_floor              0
dtype: int64
```

Таких значений нет

```
plt.ylim(0, 100) #выставляем значения по оси y
data.boxplot('living_area') # строим диаграмму размаха
plt.show()
```




```
data.plot(y = 'living_area', kind = 'hist', bins = 500, grid=True, range = (0, 80), figs
#построим гистограмму
plt.show()
```




```
data['living_area'].describe()
```

```
count    20333.000000
mean        31.380894
std         13.799247
min          2.000000
25%         18.400000
50%         30.000000
75%         40.500000
max        101.000000
Name: living_area, dtype: float64
```

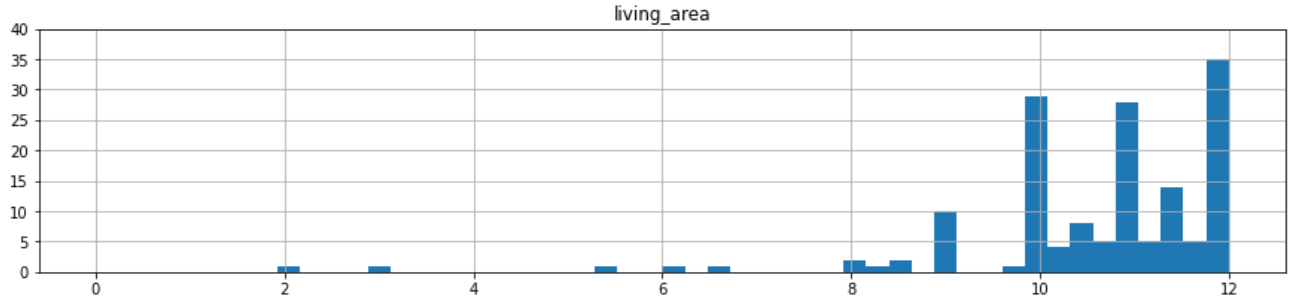
Правильно, что мы не стали менять пропуски на среднее значение. Среднее значение выше, чем общая площадь самой маленькой квартиры. Верний ус примерно 78. Нижний - минимум - 2 кв.м., это очень мало, но и такое может быть, если это студия. Большая часть квартир имеет жилую площадь от 19 до 42 кв.м. - вполне сравнимо с общей площадью (от 40 до 70 кв. м.)

```
# check
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22069 entries, 0 to 23698
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   total_images          22069 non-null  int64         
 1   last_price            22069 non-null  int64         
 2   total_area            22069 non-null  float64       
 3   first_day_exposition  22069 non-null  datetime64[ns]
 4   rooms                 22069 non-null  int64         
 5   ceiling_height        13461 non-null  float64       
 6   floors_total          22069 non-null  int64         
```

```
 7   living_area           20333 non-null  float64
 8   floor                 22069 non-null  int64
 9   is_apartment          22069 non-null  bool
 10  studio                22069 non-null  bool
 11  open_plan             22069 non-null  bool
 12  kitchen_area          20128 non-null  float64
 13  balcony               22069 non-null  int64
 14  locality_name         22069 non-null  object
 15  airports_nearest      16703 non-null  float64
 16  cityCenters_nearest   16722 non-null  float64
 17  parks_around3000      22069 non-null  float64
 18  parks_nearest         7204 non-null   float64
 19  ponds_around3000      22069 non-null  float64
 20  ponds_nearest         8145 non-null   float64
 21  days_exposition       19195 non-null  float64
 22  price_m               22069 non-null  int64
 23  weekday               22069 non-null  int64
 24  month                 22069 non-null  int64
 25  year                  22069 non-null  int64
 26  type_floor            22069 non-null  object
dtypes: bool(3), datetime64[ns](1), float64(11), int64(10), object(2)
memory usage: 4.3+ MB
```

```
data.hist(column = 'living_area', bins = 50, figsize = (15,3), range = (0,12))
plt.ylim(0, 40);
```




```
data.query('living_area < 10').count()
```

```
total_images            21
last_price              21
total_area              21
first_day_exposition    21
rooms                   21
ceiling_height          15
floors_total            21
living_area             21
floor                   21
is_apartment            21
studio                  21
open_plan               21
kitchen_area            21
balcony                 21
locality_name           21
airports_nearest        21
cityCenters_nearest     21
```

```
parks_around3000        21
parks_nearest           10
ponds_around3000        21
ponds_nearest            9
days_exposition         18
price_m                 21
weekday                 21
month                   21
year                    21
type_floor              21
dtype: int64
```

Возьмем значение за минимум и избавимся от 21 строки с меньшими значениями

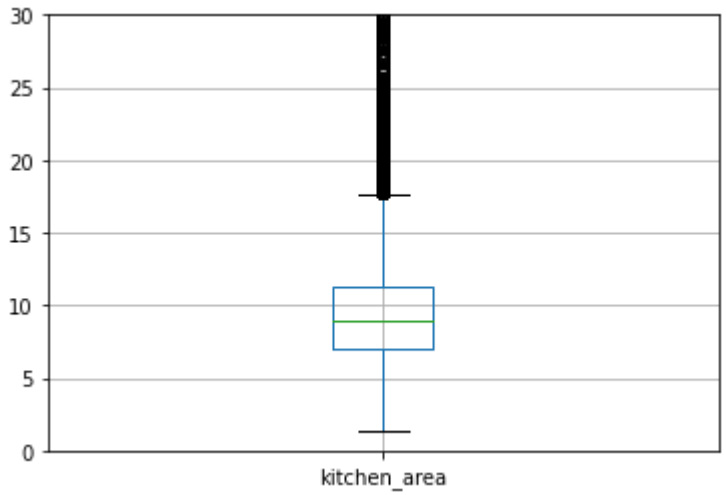
```
data = data.loc[(data['living_area'] >=10)&(data['living_area'] <=78)|(data['living_area
#оставляем лишь значения в пределах нормальных значений
```

```
# check
data.shape[0]
```

```
21973
```

Площадь кухни

```
plt.ylim(0, 30) #выставляем значения по оси y
data.boxplot('kitchen_area') # строим диаграмму размаха
plt.show()
```



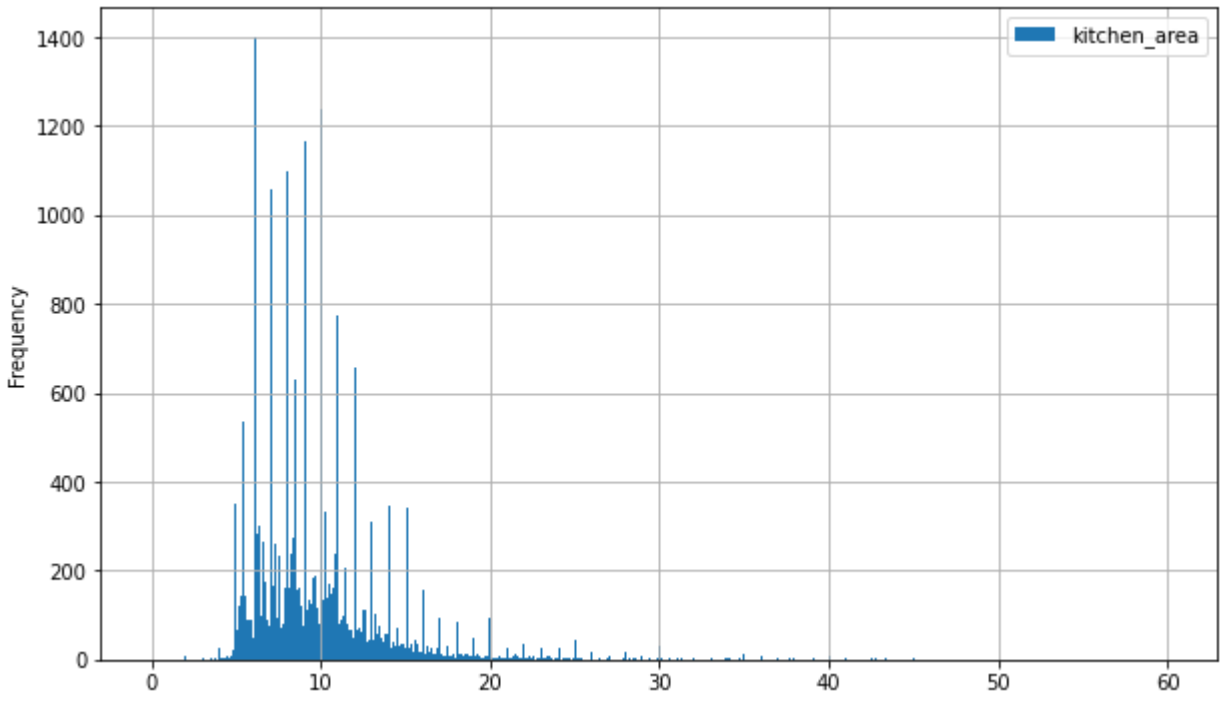

```
data['kitchen_area'].describe()
```

```
count    20040.000000
mean         9.911968
std          4.339390
min          1.300000
25%          7.000000
50%          9.000000
```

```
75%         11.300000
max         49.400000
Name: kitchen_area, dtype: float64
```

История похожа на предыдущие. Разброс данных от 1кв.м до 30. Стандартное отклонение высокое. Выбросы за пределами 17кв.м. В основном кухни 6- 12 кв.м

```
data.plot(y = 'kitchen_area', kind = 'hist', bins = 500, grid=True, range = (0,60), figs
#построим гистограмму
plt.show()
```

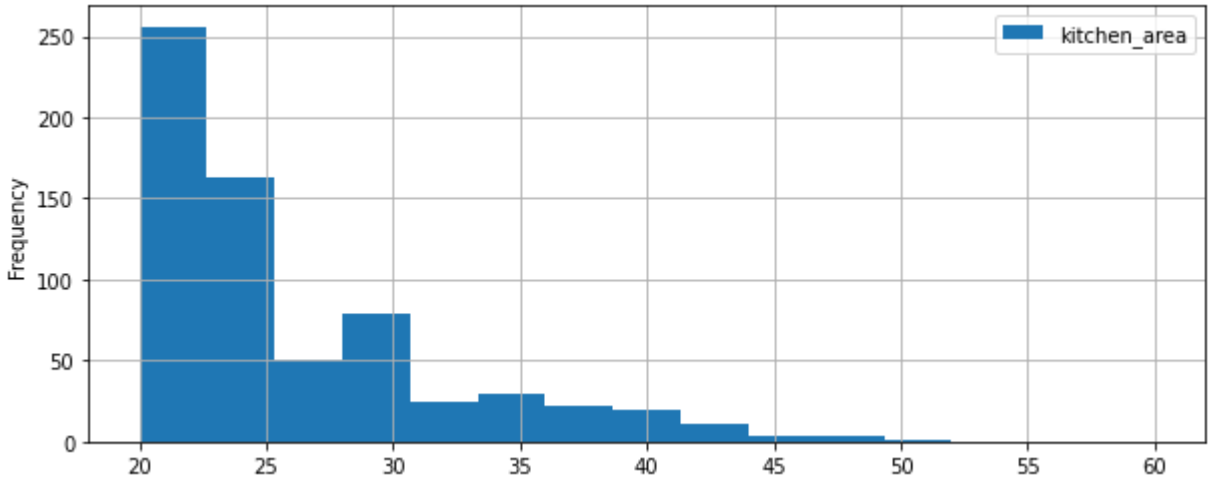



```
# check
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21973 entries, 0 to 23698
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   total_images          21973 non-null  int64         
 1   last_price            21973 non-null  int64         
 2   total_area            21973 non-null  float64       
 3   first_day_exposition  21973 non-null  datetime64[ns]
 4   rooms                 21973 non-null  int64         
 5   ceiling_height        13394 non-null  float64       
 6   floors_total          21973 non-null  int64         
 7   living_area           20237 non-null  float64       
 8   floor                 21973 non-null  int64         
 9   is_apartment          21973 non-null  bool          
 10  studio                21973 non-null  bool          
 11  open_plan             21973 non-null  bool          
 12  kitchen_area          20040 non-null  float64       
 13  balcony               21973 non-null  int64         
 14  locality_name         21973 non-null  object
```

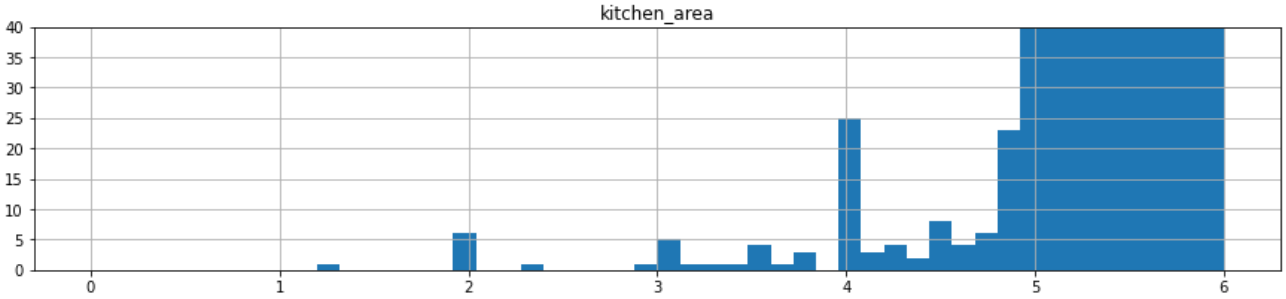
```
 15  airports_nearest      16615 non-null  float64
 16  cityCenters_nearest   16633 non-null  float64
 17  parks_around3000      21973 non-null  float64
 18  parks_nearest         7153 non-null   float64
 19  ponds_around3000      21973 non-null  float64
 20  ponds_nearest         8093 non-null   float64
 21  days_exposition       19120 non-null  float64
 22  price_m               21973 non-null  int64
 23  weekday               21973 non-null  int64
 24  month                 21973 non-null  int64
 25  year                  21973 non-null  int64
 26  type_floor            21973 non-null  object
dtypes: bool(3), datetime64[ns](1), float64(11), int64(10), object(2)
memory usage: 4.3+ MB
```

```
# check
data.plot(y = 'kitchen_area', kind = 'hist', bins = 15, grid=True, range = (20,60), figs
```



Все, что больше 40 возьмем за выбросы

```
data.hist(column = 'kitchen_area', bins = 50, figsize = (15,3), range = (0,6))
plt.ylim(0, 40);
```




```
data.query('kitchen_area < 4').count()
```

```
total_images           25
last_price             25
```

```
total_area              25
first_day_exposition    25
rooms                   25
ceiling_height          17
floors_total            25
living_area             25
floor                   25
is_apartment            25
studio                  25
open_plan               25
kitchen_area            25
balcony                 25
locality_name           25
airports_nearest        20
cityCenters_nearest     20
parks_around3000        25
parks_nearest            5
ponds_around3000        25
ponds_nearest           12
days_exposition         23
price_m                 25
weekday                 25
month                   25
year                    25
type_floor              25
dtype: int64
```

Все, что меньше 4 возьмем за выбросы

```
data = data.loc[(data['kitchen_area'] >=4)&(data['kitchen_area'] <=40)|(data['kitchen_ar
```

```
# check
data.shape[0]
```

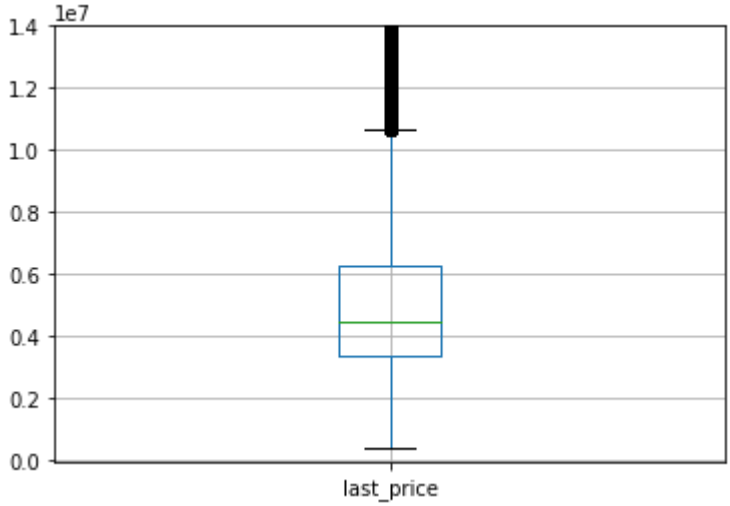
```
21924
```

Цена

```
data['last_price'].describe()
```

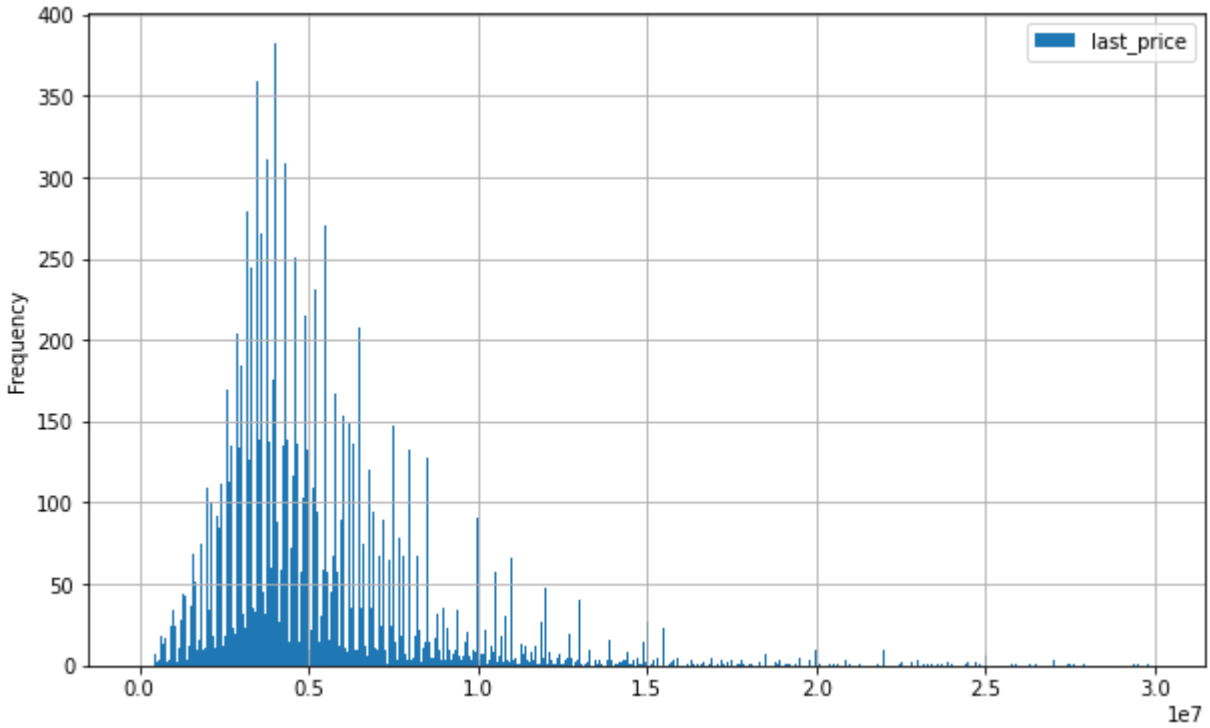
```
count    2.192400e+04
mean     5.288666e+06
std      3.143623e+06
min      4.300000e+05
25%      3.400000e+06
50%      4.500000e+06
75%      6.300000e+06
max      2.990000e+07
Name: last_price, dtype: float64
```

```
plt.ylim(-3e+04, 1.4e+07)
data.boxplot('last_price')
plt.show()
```



Разброс цен огромен. Не удивительно, ведь расположены они в разных местах. Большая часть квартир стоит 3,5 - 7 млн. руб.

```
data.plot(y = 'last_price', kind = 'hist', bins = 1000, grid=True, range = (0,3.0e+07), 
#построим гистограмму
plt.show()
```



По гистограмме видно, что самая частая цена около 4 млн.руб

```
data = data.loc[(data['last_price'] <=2.0e+07)] #избавляемся от выбросов
```

```
# check
data.shape[0]
```

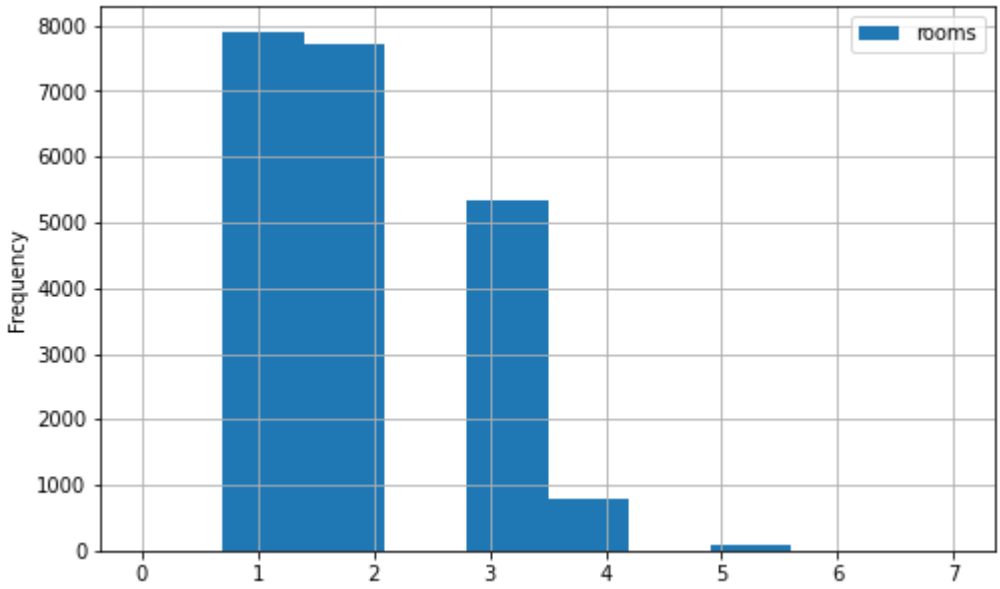
```
21816
```

Количество комнат

```
data['rooms'].describe()
```

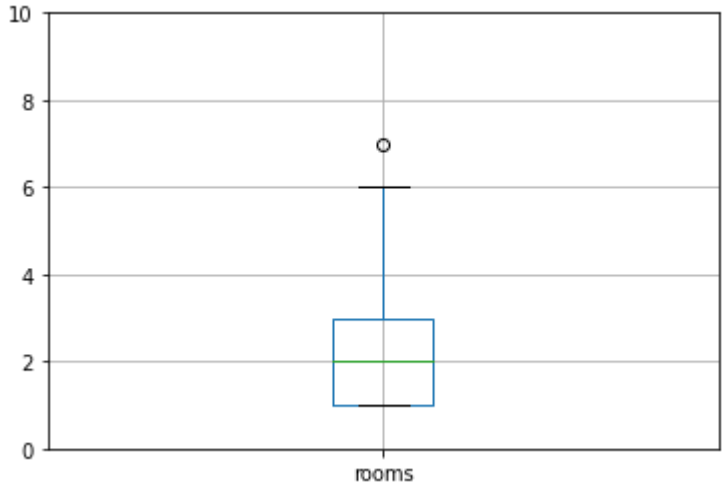
```
count    21816.000000
mean         1.966355
std          0.887515
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max          7.000000
Name: rooms, dtype: float64
```

```
data.plot(y = 'rooms', kind = 'hist', bins = 10, grid=True, range = (0,7), figsize = (8,
plt.show()
```



По гистограмме видим, что чаще всего продают однокомнатные и двухкомнатные квартиры.

```
plt.ylim(0,10)
data.boxplot('rooms')
plt.show()
```



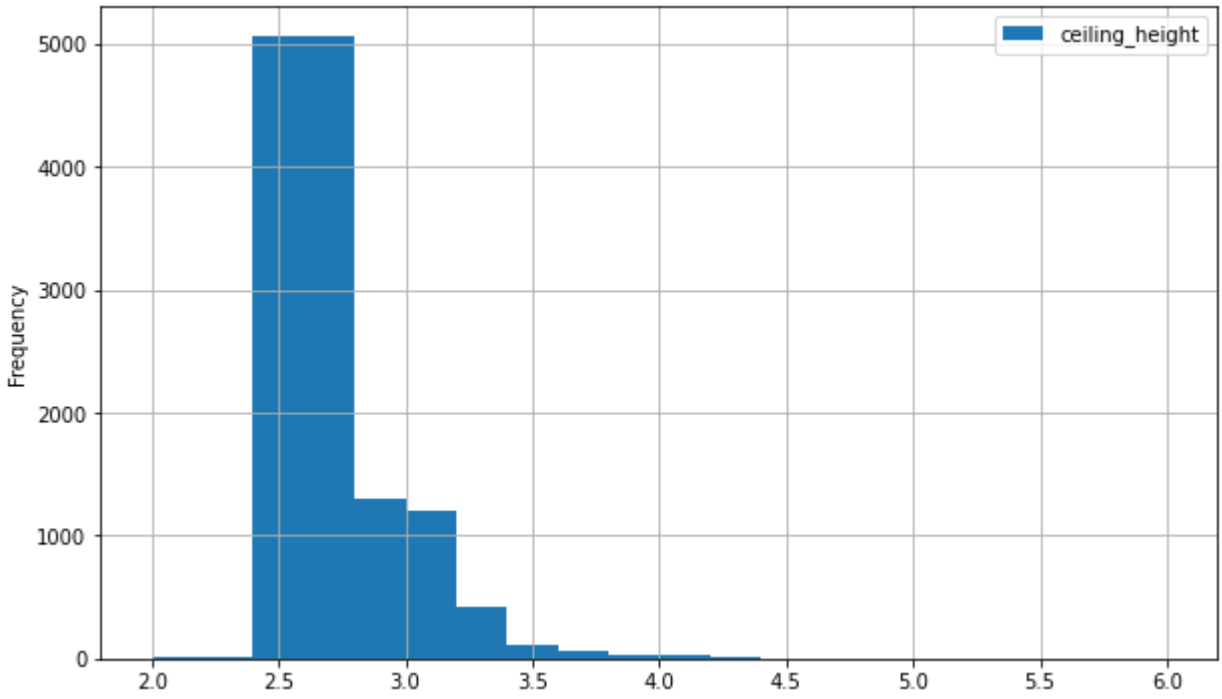
Семикомнатные попали в выбросы. Но мы их оставим, как интересные для исследования

Высота потолков

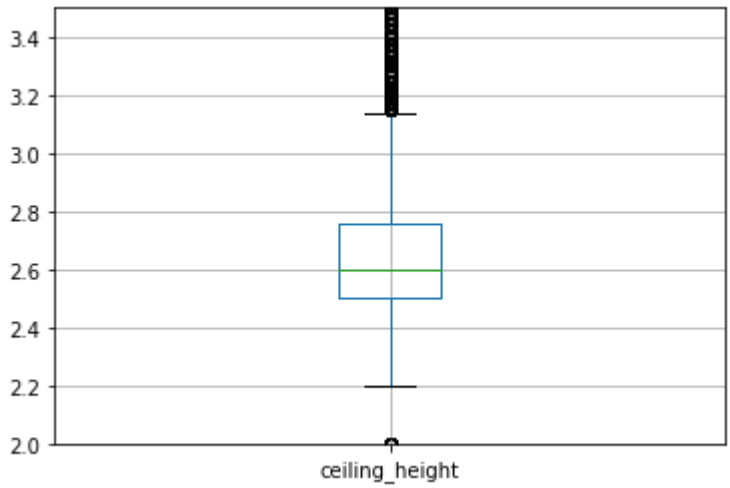
```
data['ceiling_height'].describe()
```

```
count    13289.000000
mean         2.695765
std          0.240063
min          2.000000
25%          2.500000
50%          2.600000
75%          2.760000
max          5.300000
Name: ceiling_height, dtype: float64
```

```
data.plot(y = 'ceiling_height', kind = 'hist', bins = 20, grid=True, range = (2,6), figs
#построим гистограмму
plt.show()
```




```
plt.ylim(2, 3.5)
data.boxplot('ceiling_height')
plt.show()
```



Средняя высота потолка 2,6 м. Минимальная высота сейчас 2,2 м., а максимум 5,3 м.. Т.к такие значения имеют место быть в реальности, не будем удалять их из таблицы, как выбросы.

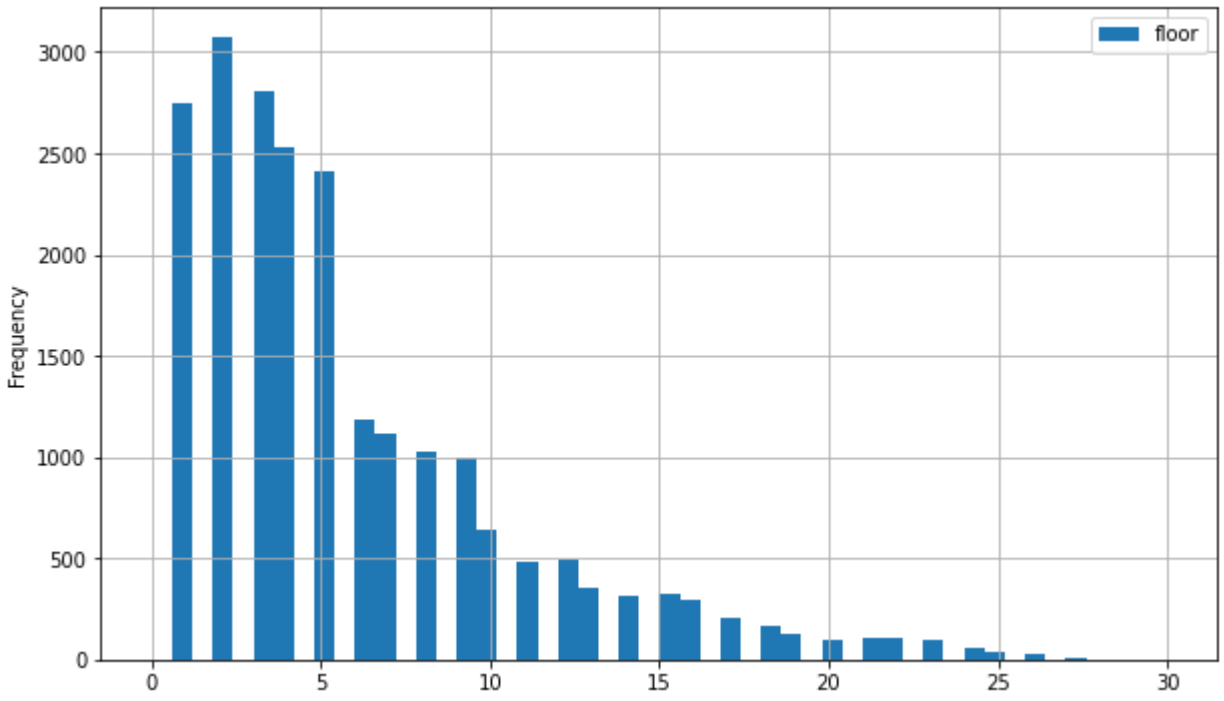
Этаж

```
data['floor'].describe()
```

```
count    21816.000000
mean         5.899936
std          4.885771
min          1.000000
25%          2.000000
```

```
50%          4.000000
75%          8.000000
max         33.000000
Name: floor, dtype: float64
```

```
data.plot(y = 'floor', kind = 'hist', bins = 50, grid=True, range = (0,30), figsize = (1(
plt.show()
```



Самые часто встречающиеся значения 1-5. Очень похоже на правду. Стандартное отклонение небольшое, что говорит о небольшом разбросе значений.

Проверим на аномалии значения количества этажей в здании. Нет ли таких строк, где этаж больше количества этажей в здании.

```
data.query( 'floor > floors_total').count()
```

```
total_images            71
last_price              71
total_area              71
first_day_exposition    71
rooms                   71
ceiling_height           7
floors_total            71
living_area             47
floor                   71
is_apartment            71
studio                  71
open_plan               71
kitchen_area            35
balcony                 71
locality_name           71
airports_nearest        65
```

```
cityCenters_nearest     65
parks_around3000        71
parks_nearest           31
ponds_around3000        71
ponds_nearest           43
days_exposition         63
price_m                 71
weekday                 71
month                   71
year                    71
type_floor              71
dtype: int64
```

```
data.loc[data['floor'] > data['floors_total'], 'floors_total'] = data['floor']
```

Такие строки есть. Поменяем значения общего количества этажей на этаж, приняв его за последний. 72 строки - это меньше 1%.

Тип этажа

```
data['type_floor'].value_counts().plot(kind='pie')
plt.show()
```

Квартир на первом и последнем этажах практически поровну.

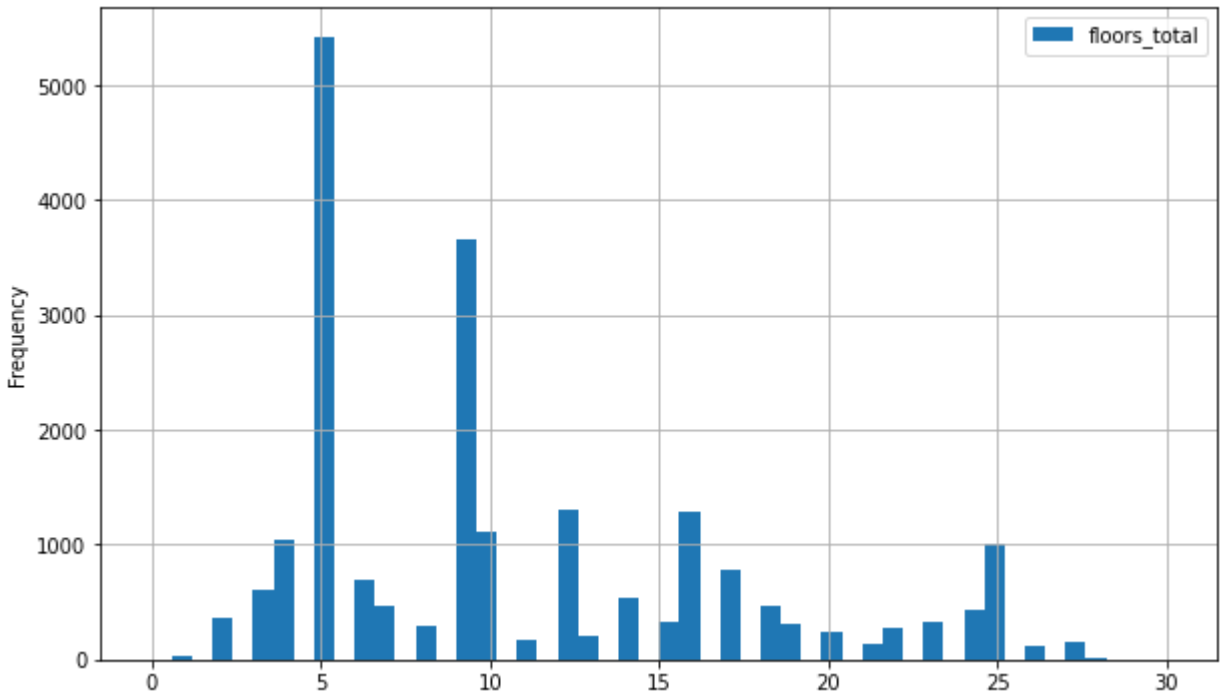
Общее количество этажей в доме

```
data['floors_total'].describe()
```

```
count    21816.000000
mean        10.749908
std          6.597222
min          1.000000
25%          5.000000
50%          9.000000
```

```
75%         16.000000
max         36.000000
Name: floors_total, dtype: float64
```

```
data.plot(y = 'floors_total', kind = 'hist', bins = 50, grid=True, range = (0,30), figsi
#построим гистограмму
plt.show()
```



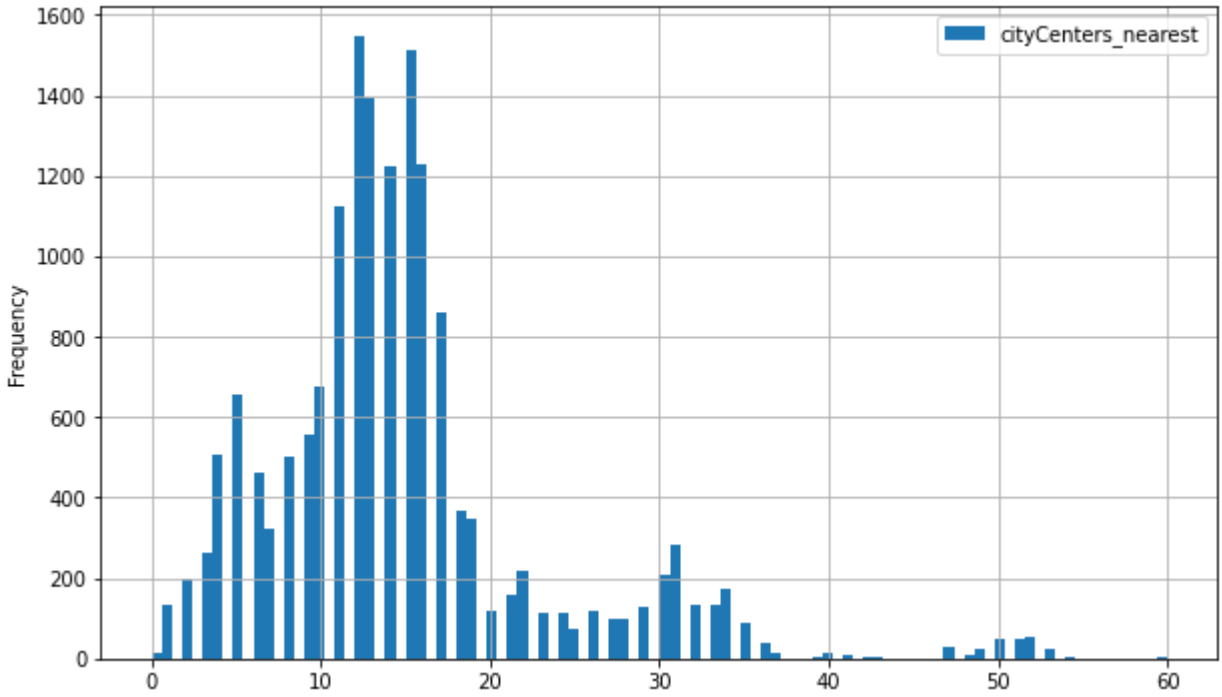
Чаще всего дома имеют 5 и 9 этажей. Так и есть.

Расстояние до центра города

```
data['cityCenters_nearest'].describe()
```

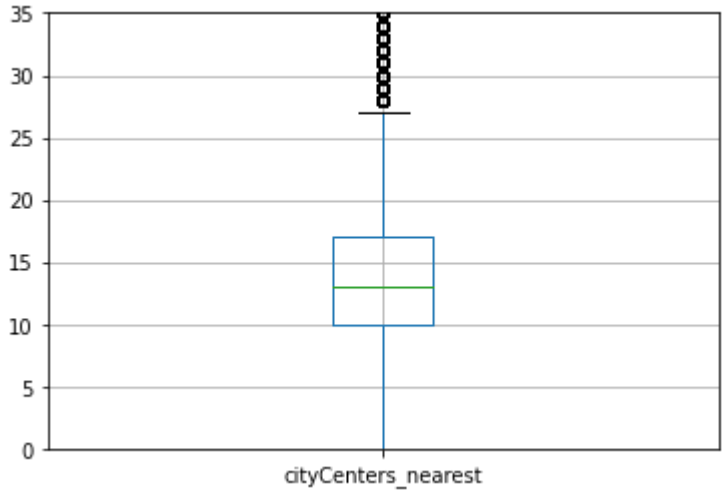
```
count    16485.000000
mean        14.722414
std          8.511493
min          0.000000
25%         10.000000
50%         13.000000
75%         17.000000
max         66.000000
Name: cityCenters_nearest, dtype: float64
```

```
data.plot(y = 'cityCenters_nearest', kind = 'hist', bins = 100, grid=True, range = (0,60
#построим гистограмму
plt.show()
```



Меньше всего квартир находится в центре города. А некоторые расположены аж в 60 км.

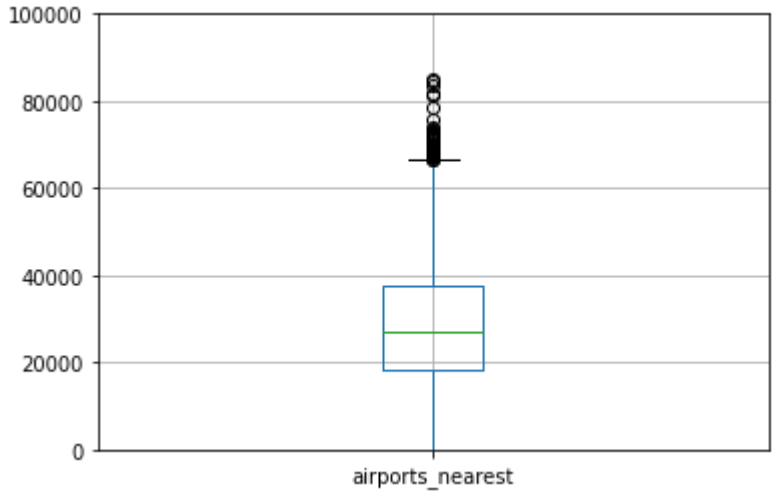
```
plt.ylim(0, 35)
data.boxplot('cityCenters_nearest')
plt.show()
```



Большая часть квартир расположена на расстоянии 11-18 км от центра города. Нижний ус на отметке 2. Значит кто-то живет в центре. Верхний ус на 26 км. Остальное считаем выбросами

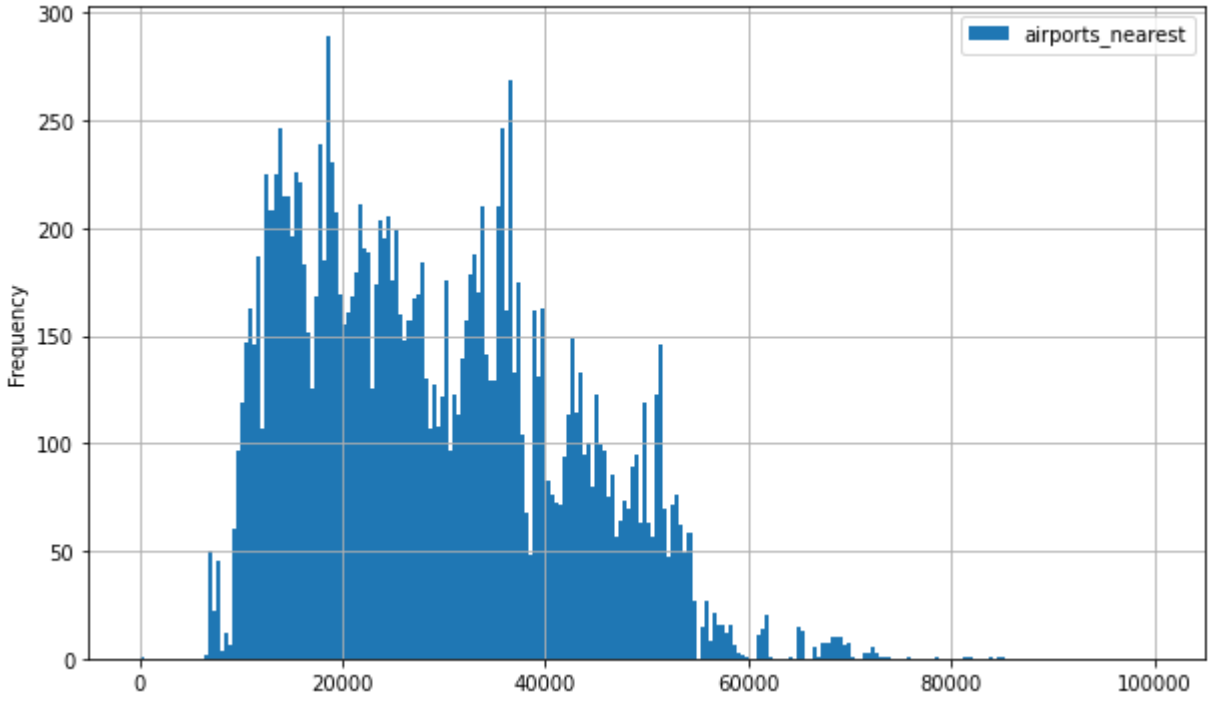
Расстояние до аэропорта

```
plt.ylim(0, 100000)
data.boxplot('airports_nearest')
plt.show()
```



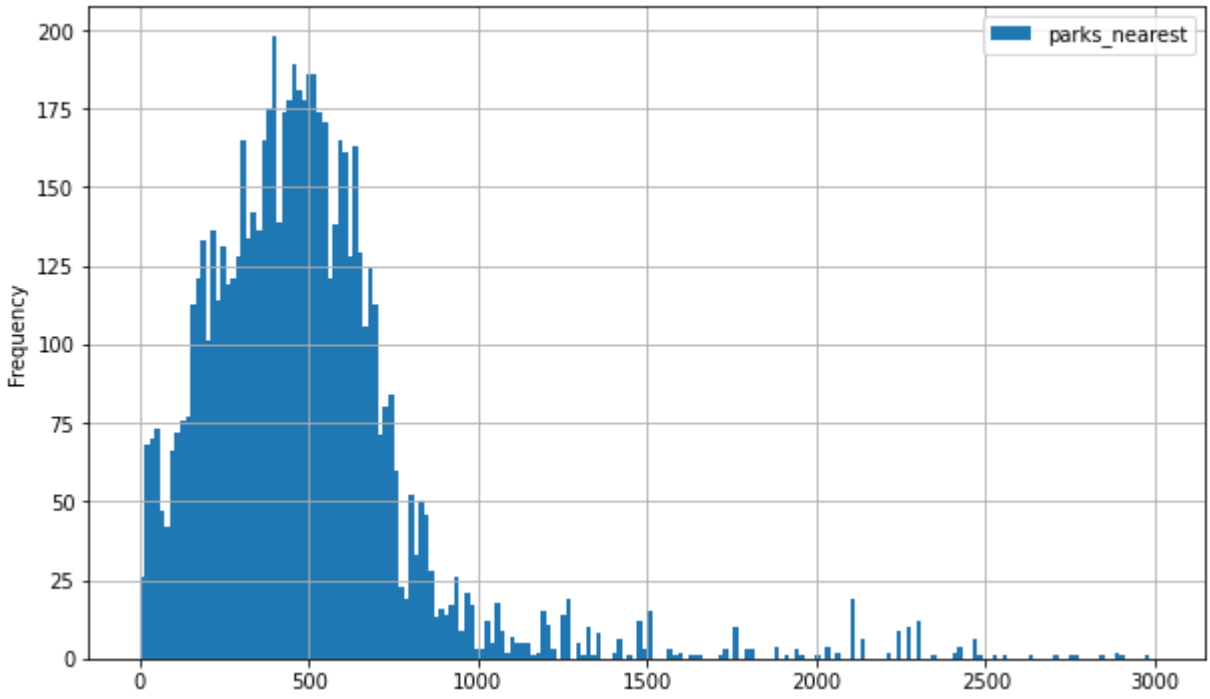
В основном расстояние до аэропорта в значениях от 18 до 39 км. Но мы помним, что во многих строках значения просто отсутствовали

```
data.plot(y = 'airports_nearest', kind = 'hist', bins = 250, grid=True, range = (0,10000
#построим гистограмму
plt.show()
```

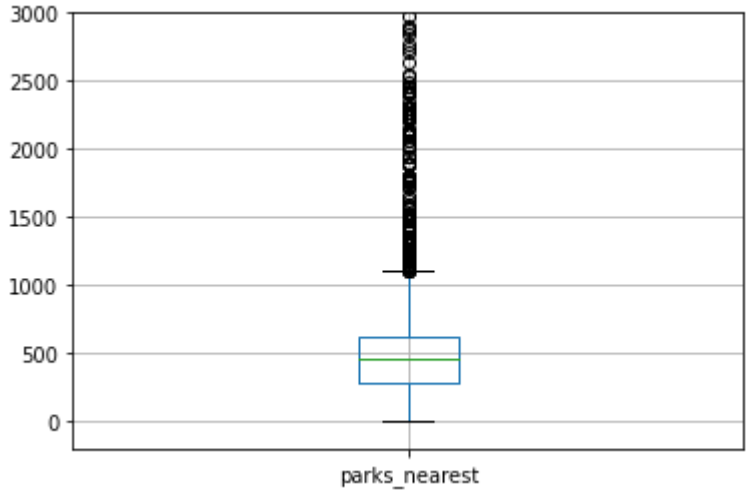


Расстояние до ближайшего парка

```
data.plot(y = 'parks_nearest', kind = 'hist', bins = 200, grid=True, range = (0,3000), f
#построим гистограмму
plt.show()
```




```
plt.ylim(-200, 3000)
data.boxplot('parks_nearest')
plt.show()
```



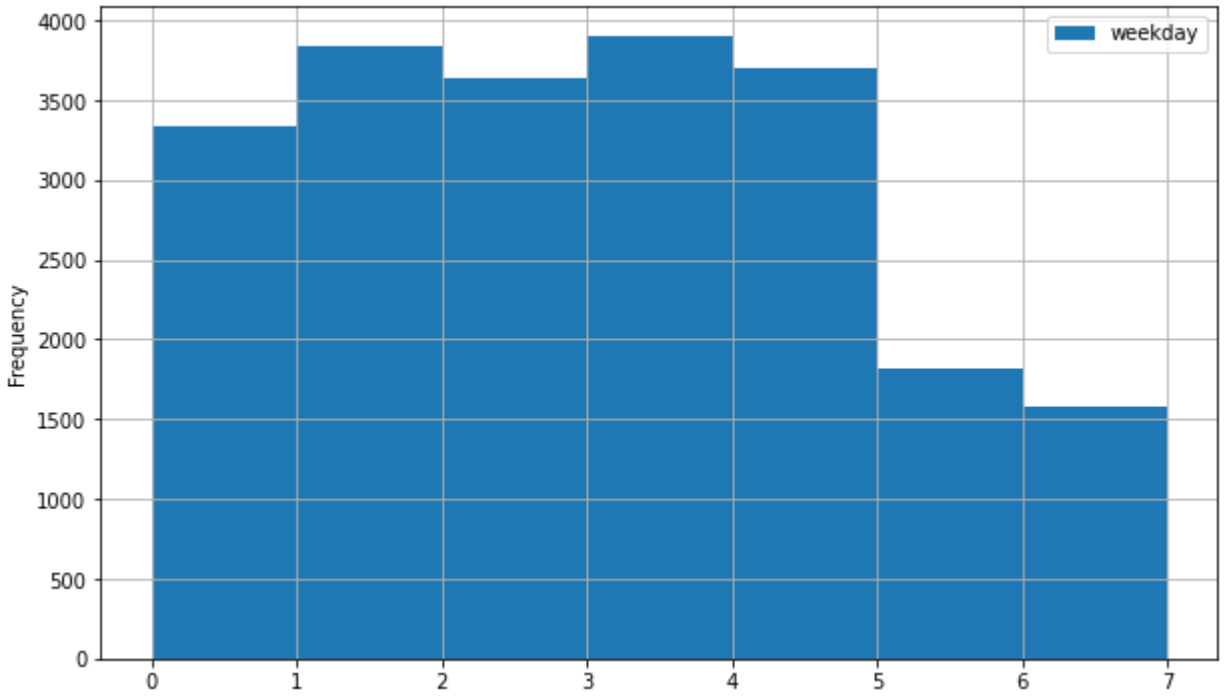

```
data['parks_nearest'].describe()
```

```
count    7066.000000
mean      493.430795
std       339.088390
min         1.000000
25%       289.000000
50%       458.000000
75%       616.000000
max      3190.000000
Name: parks_nearest, dtype: float64
```

В основном в продаже квартиры недалеко от парка, в 300-600 метрах. Но кому-то очень повезет и до парка будет 1 м.

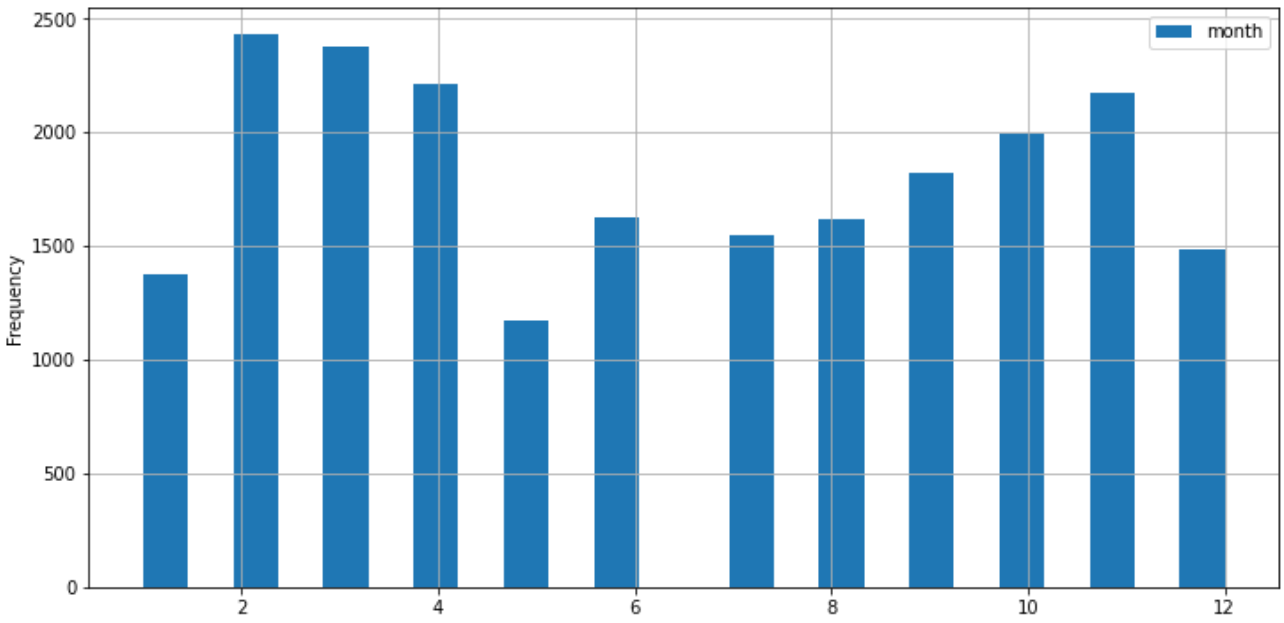
День и месяц публикации

```
data.plot(y = 'weekday', kind = 'hist', bins = 7, grid=True, range = (0,7), figsize = (1(
plt.show()
```



Чаще всего размещают объявления по четвергам, а реже всего по воскресеньям

```
data.plot(y = 'month', kind = 'hist', bins = 24, grid=True, range = (1,12), figsize = (1:
plt.show()
```



Самый насыщенный на объявления месяц февраль. В мае размещают объявления реже всего

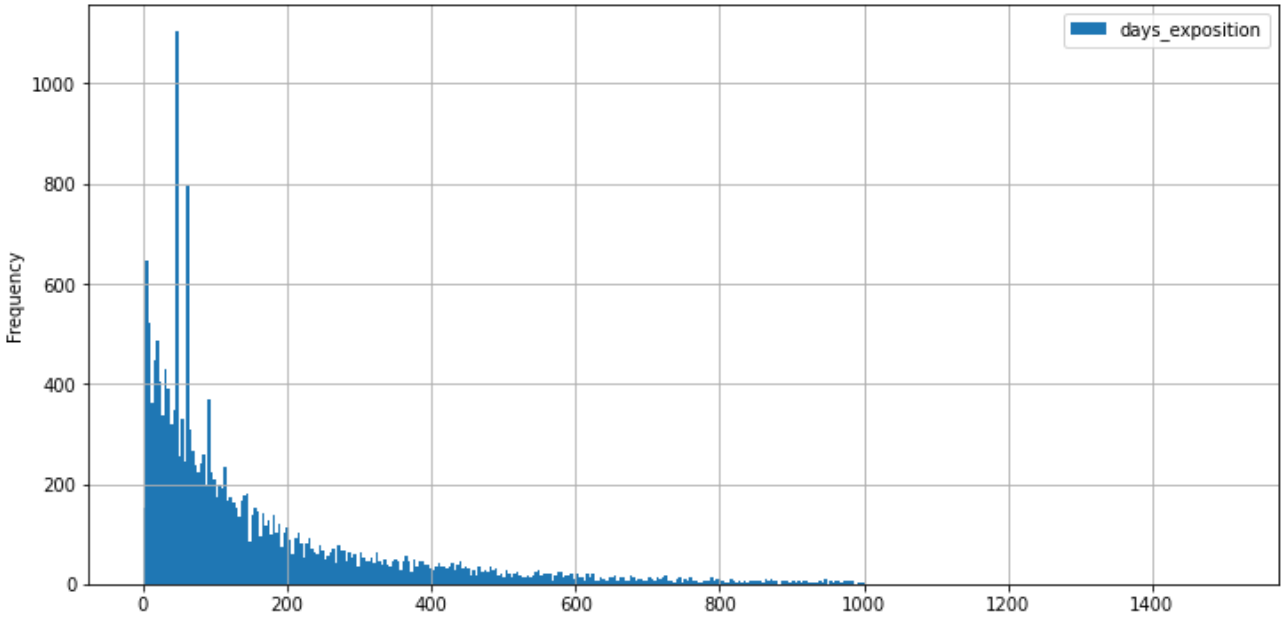
```
# check

# Показатели о кол-ве объявлений в датасете, минимальных и максимальных значениях
# в выбранных параметрах о продаже квартир

(
    data[['rooms', 'total_area', 'ceiling_height', 'days_exposition', 'last_price', 'liv
          'floor', 'floors_total']]
    .apply (['count', 'min', 'max'])
    .style.format("{:,.2f}")
)
```

	rooms	total_area	ceiling_height	days_exposition	last_price	living_area	kitchen_area
count	21,816.00	21,816.00	13,289.00	19,009.00	21,816.00	20,097.00	19,897.00
min	1.00	20.00	2.00	3.00	430,000.00	10.00	4.00
max	7.00	120.00	5.30	999.00	20,000,000.00	78.00	40.00

```
# check
data.hist(column = 'kitchen_area', bins = 50, figsize = (15,3), range = (0,5));
```

Как быстро продавались квартиры

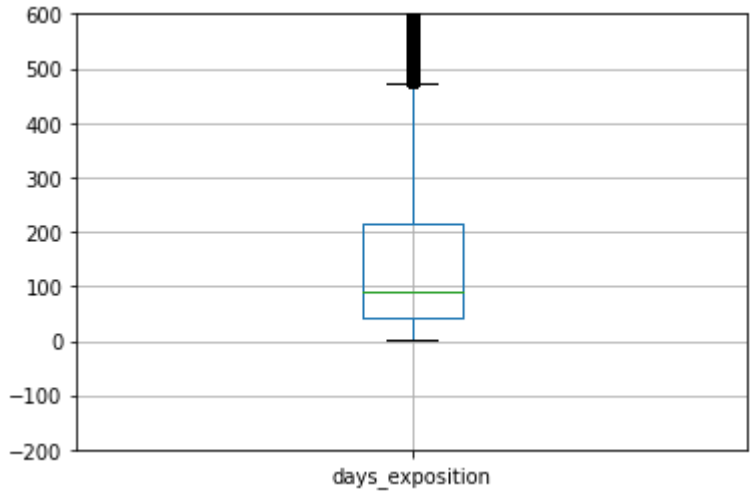
```
data_corr_days = data.query('~days_exposition.isna() and days_exposition >= 1')
# делаем выборку без нулевых значений и пустых строк


data_corr_days.plot(y = 'days_exposition', kind = 'hist', bins = 400, grid=True, range =
#построим гистограмму
plt.show()
```




```
data_corr_days['days_exposition'].describe()
```

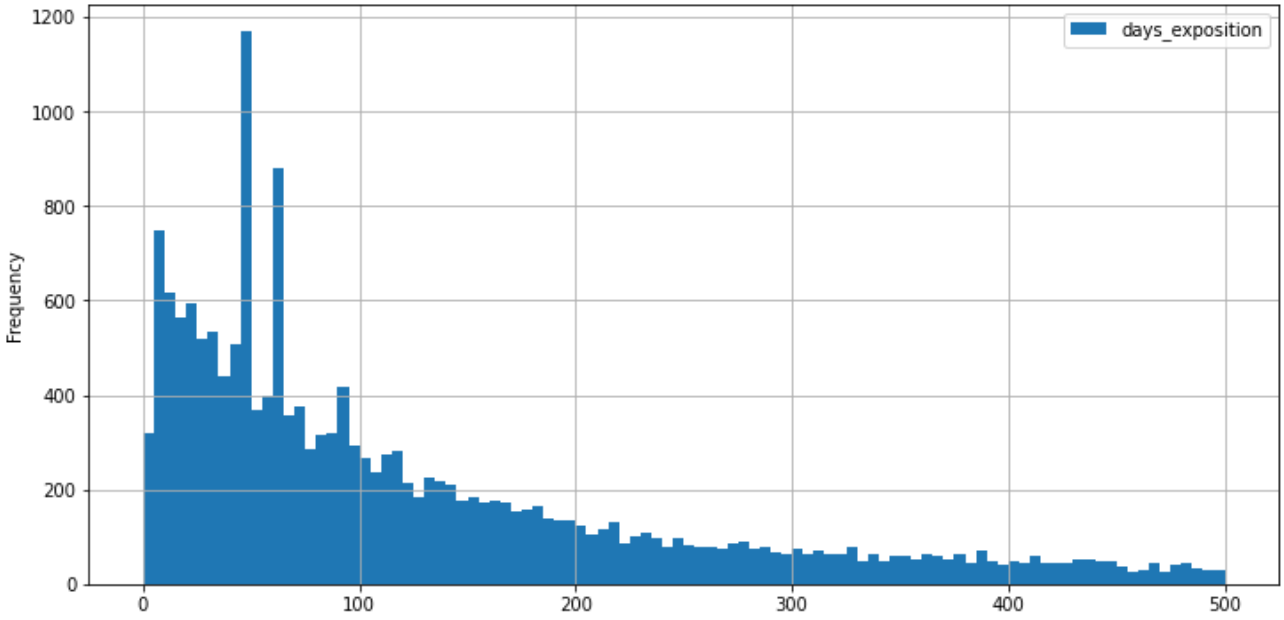
```
count    19009.000000
mean       163.562944
std        184.226479
min          3.000000
25%         44.000000
50%         91.000000
75%        216.000000
max        999.000000
Name: days_exposition, dtype: float64
```

```
plt.ylim(-200, 600)
data_corr_days.boxplot('days_exposition')
plt.show()
```



Большая часть объявлений висела на сайте от 50 до 200 дней. Все, что больше 500 возьмем за выбросы. Построим гистограмму без них

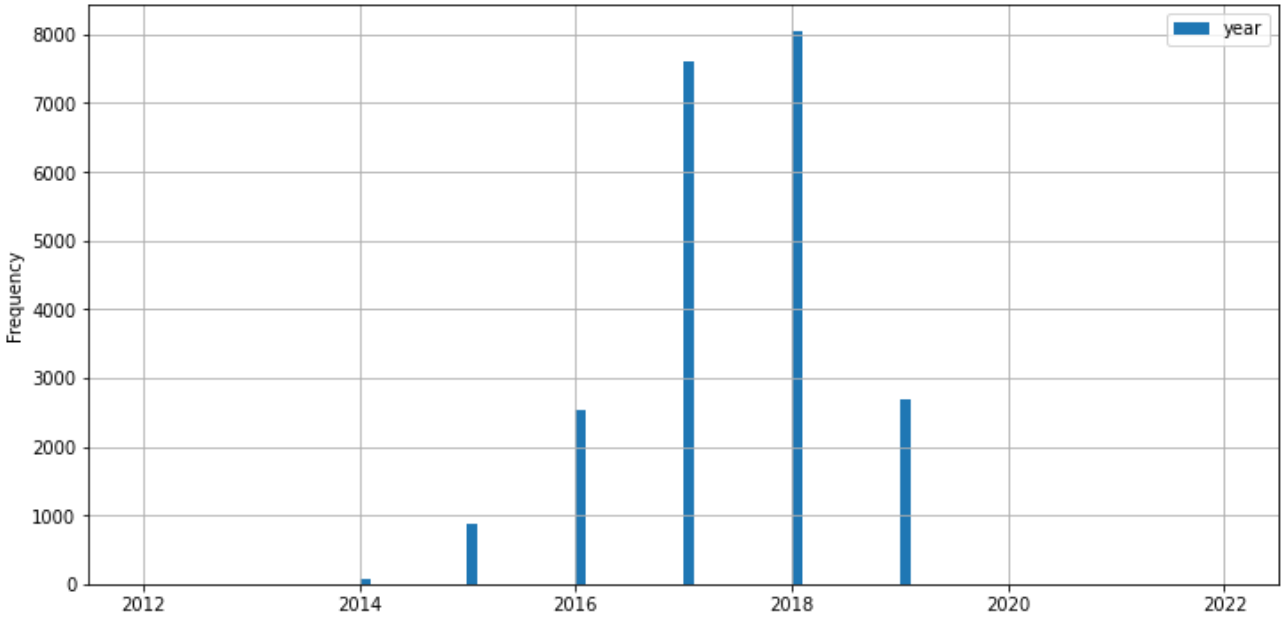
```
#построим гистограмму
data_corr_days.plot(y = 'days_exposition', kind = 'hist', bins = 100, grid=True, range =
plt.show()
```



Чаще всего объявления висят около трех месяцев, но бывает, что квартиры продаются за 1 день или, наоборот, очень долго не продаются, как та, что висела дольше 4 лет.

Изучив сайт Яндекс.Недвижимость, видим, что срок публикации объявления для квартир от 10 млн — 90 дней. Поэтому на графике мы видим скачок. Это не значит, что квартиры проданы. Скорее всего у них просто закнчился срок объявления. Тоже самое происходит и с квартирами до 4,5 млн. — 45 дней, от 4,5 до 10 млн — 60 дней

```
data.plot(y = 'year', kind = 'hist', bins = 100, grid=True, range = (2012,2022), figsize
plt.show()
```



Либо продажи росли по годам, либо данные вводили регулярнее. Скорее всего данные в таблице не до конца 2019 г.

Изучим какие факторы больше всего влияют на общую стоимость объекта

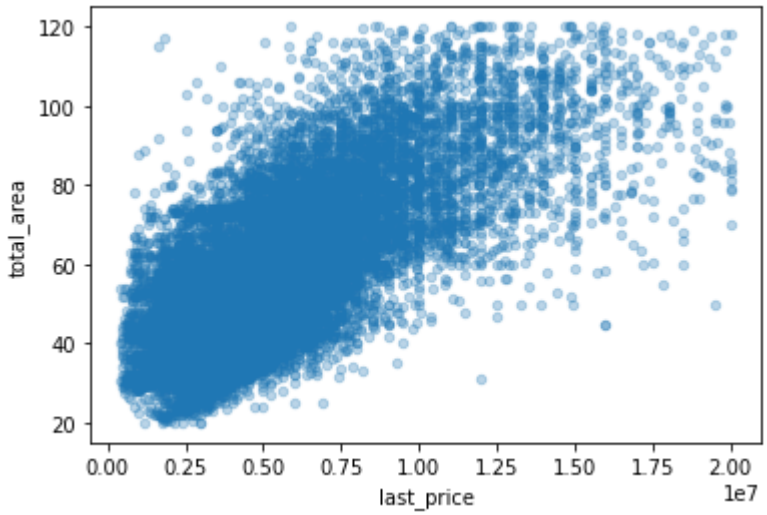
Зависимость цены от общей площади

```
data['last_price'].corr(data['total_area'])
```

```
0.730607090018265
```

Корреляция положительная. Коэффициент Пирсона в 0.73 говорит о наличии связи, и довольно сильной. Выходит, увеличение площади сопровождается увеличением цены, но так бывает не всегда.

```
data.plot(x = 'last_price', y = 'total_area', kind = 'scatter',  alpha = 0.3)
plt.show()
```



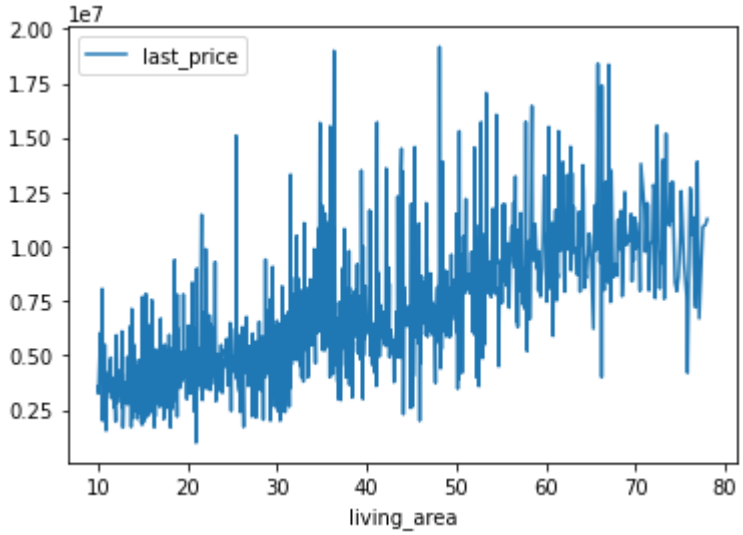
Есть основная масса точек с наиболее частыми сочетаниями цены и площади. При этом с увеличением площади увеличивается и цена. Это мы можем увидеть на графике. Но лишь в среднем. Можно найти уникальные примеры квартир с высокой ценой и не очень большой площадью.

Завсисимость цены от жилой площади

```
data['last_price'].corr(data['living_area'])
```

```
0.5935695614314986
```

```
data.pivot_table(index = 'living_area', values  = 'last_price').plot()
plt.show()
```



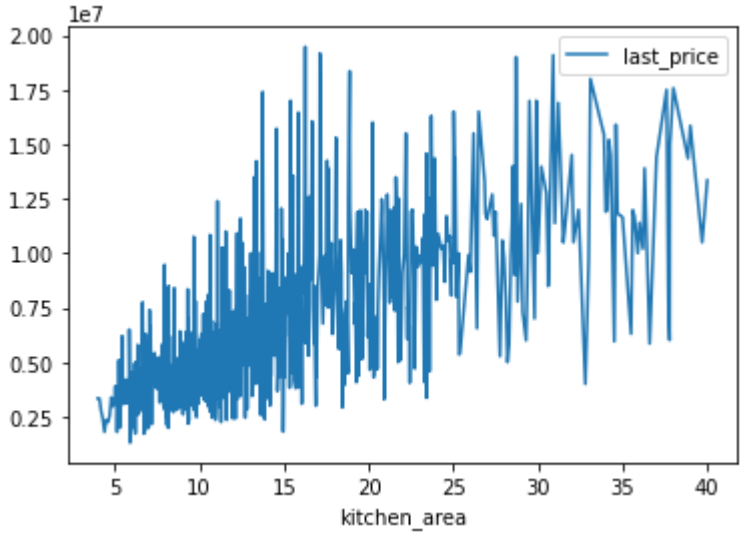
Очень похоже на общую площадь. Коэффициент Пирсона еще ниже, 0,59. Значит зависимость цены от жилой площади ниже, чем от общей

Завсисимость цены от площади кухни

```
data['last_price'].corr(data['kitchen_area'])
```

```
0.5464135120612365
```

```
data.pivot_table(index = 'kitchen_area', values  = 'last_price').plot()
plt.show()
```



А зависимость цены от площади кухни еще ниже. *Хочется посмотреть как зависит жилая площадь и площадь кухни от общей площади*

```
#создадим перемеенную для столбцов общая площадь, жилая площадь и площадь кухни
data_corr_area = data.filter(['total_area','kitchen_area','living_area'], axis=1)
#корреляция по трем столбцам
data_corr_area.corr()
```

	total_area	kitchen_area	living_area
total_area	1.000000	0.482685	0.909565
kitchen_area	0.482685	1.000000	0.191732
living_area	0.909565	0.191732	1.000000

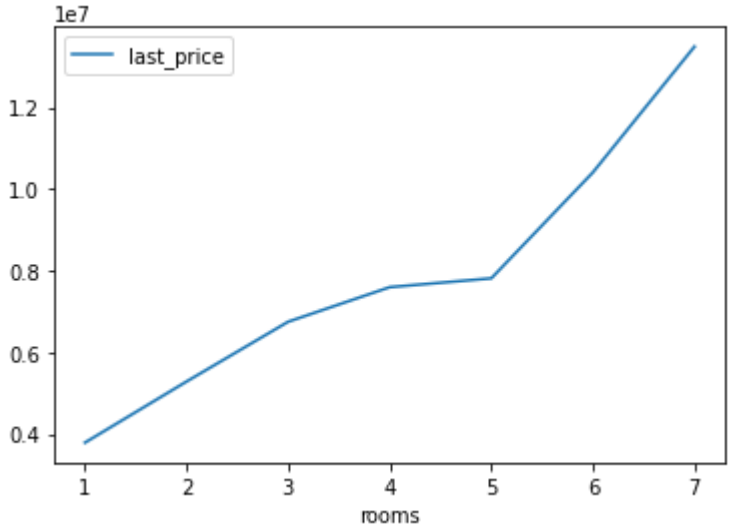
Видим сильную связь между общей и жилой площадью. А вот связь между общей площадью и кухней низкая. Это значит, что и в больших квартирах часто встречаются маленькие кухни.

Завсисимость цены от количества комнат

```
data['last_price'].corr(data['rooms'])
```

```
0.43332704396276617
```

```
data.pivot_table(index = 'rooms', values  = 'last_price').plot()
plt.show()
```



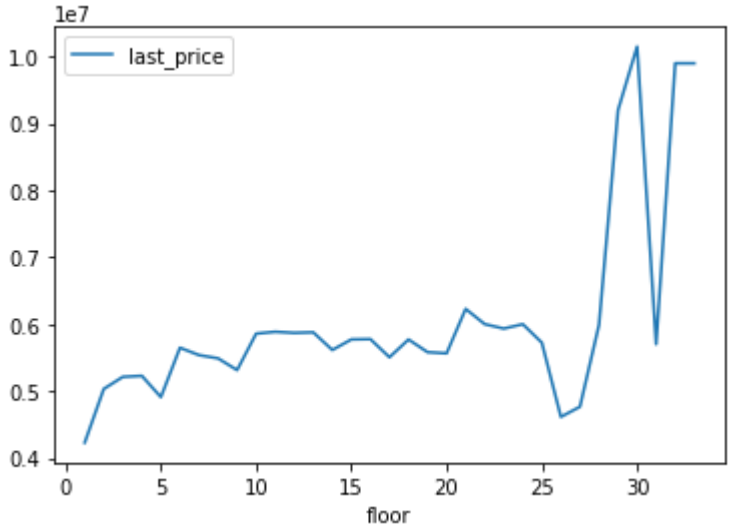
Коэффициент корреляции низкий, значит зависимость цены от количества комнат слабая

Завсисимость цены от этажа, на котором расположена квартира

```
data['last_price'].corr(data['floor'])
```

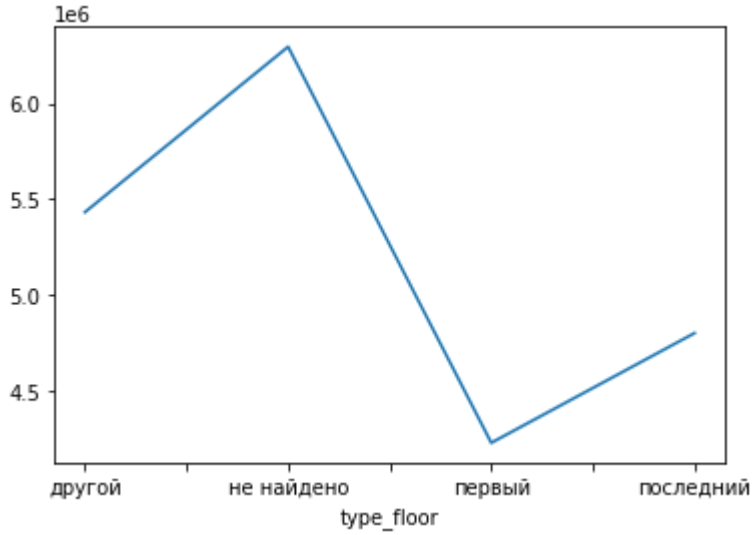
```
0.1218105319163859
```

```
data.pivot_table(index = 'floor', values  = 'last_price').plot()
plt.show()
```



Корреляция цены и этажа очень слабая, всего 0,12.

```
data_floor = data.groupby('type_floor')['last_price'].mean()
data_floor.plot()
plt.show()
```




```
data_floor
```

```
type_floor
другой        5.432199e+06
не найдено    6.294832e+06
первый        4.228888e+06
последний     4.800969e+06
Name: last_price, dtype: float64
```

Самые дешевые квартиры на первом этаже, далее на последнем, остальные дороже.

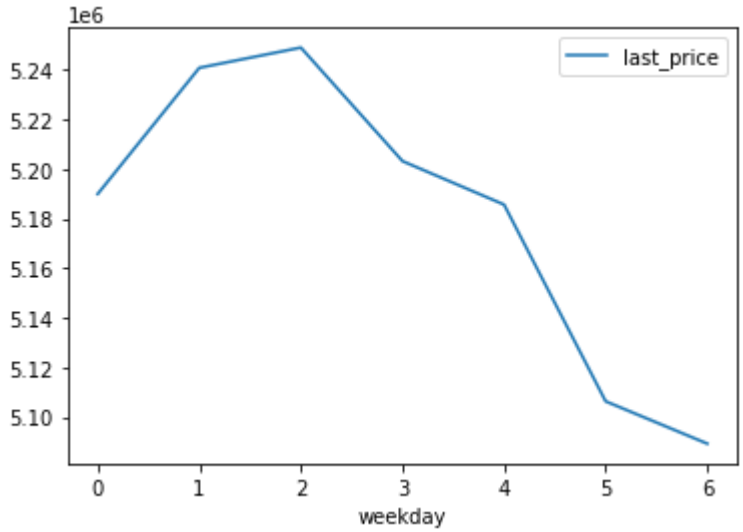
Завсисимость цены от даты размещения

```
#создадим перемеенную для столбцов даты и цены.
#Т.к столбец Дата публикации имеет тип данных datetime и корреляцию по нему сделать нево
data_corr_date = data.filter(['last_price','weekday','month','year'], axis=1)
#корреляция по четырем столбцам
data_corr_date.corr()
```

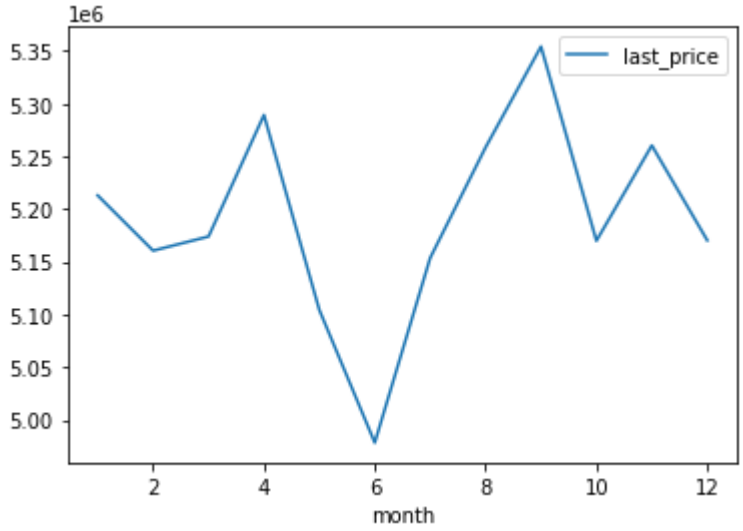
	last_price	weekday	month	year
last_price	1.000000	-0.012034	0.006185	0.000945
weekday	-0.012034	1.000000	0.009439	-0.006376
month	0.006185	0.009439	1.000000	-0.276299
year	0.000945	-0.006376	-0.276299	1.000000

Корреляция цены и даты начинается с сотых значений, значит она очень слабая

```
data.pivot_table(index = 'weekday', values  = 'last_price').plot()
plt.show()
```




```
data.pivot_table(index = 'month', values  = 'last_price').plot()
plt.show()
```



Посчитаем среднюю цену за квадратный метр в 10 населенных пунктах с наибольшим числом объявлений

```
top10_price_m = data.pivot_table(index = 'locality_name', values = 'price_m', aggfunc=['
top10_price_m.columns = ['count', 'mean']
top10_price_m = top10_price_m.sort_values('count', ascending = False)
top10_price_m.head(10)
```

locality_name	count	mean
Санкт-Петербург	14146	108603.074085
Мурино	553	85498.848101
Кудрово	445	95142.062921
поселок Шушары	429	78224.370629
Всеволожск	384	67194.148438
Пушкин	341	101938.712610
Колпино	334	75316.068862
поселок Парголово	321	90374.937695
Гатчина	304	68919.016447
Выборг	229	58243.454148

Выведем минимальное и максимальное значение стоимости квадратного метра.

```
top10_price_m.sort_values(by = 'mean').head(1)
```

locality_name	count	mean
деревня Старополье	3	11206.0

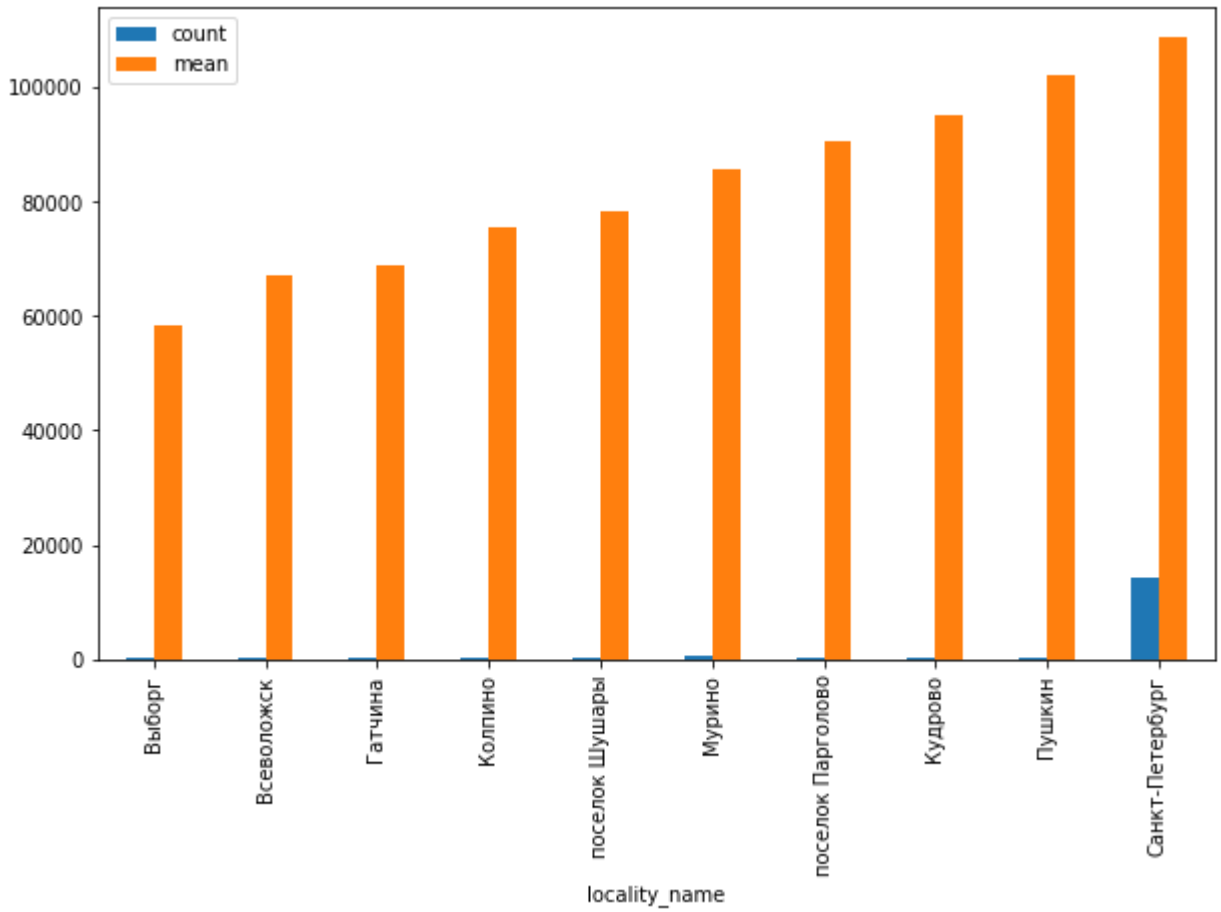
```
top10_price_m.sort_values(by = 'mean', ascending = False).head(1)
```

locality_name	count	mean
поселок Лисий Нос	2	113728.0

Самый дешевый квадратный метр в деревне Старополье 10368 руб. Самый дорогой в Санкт-Петербурге 103796 руб.

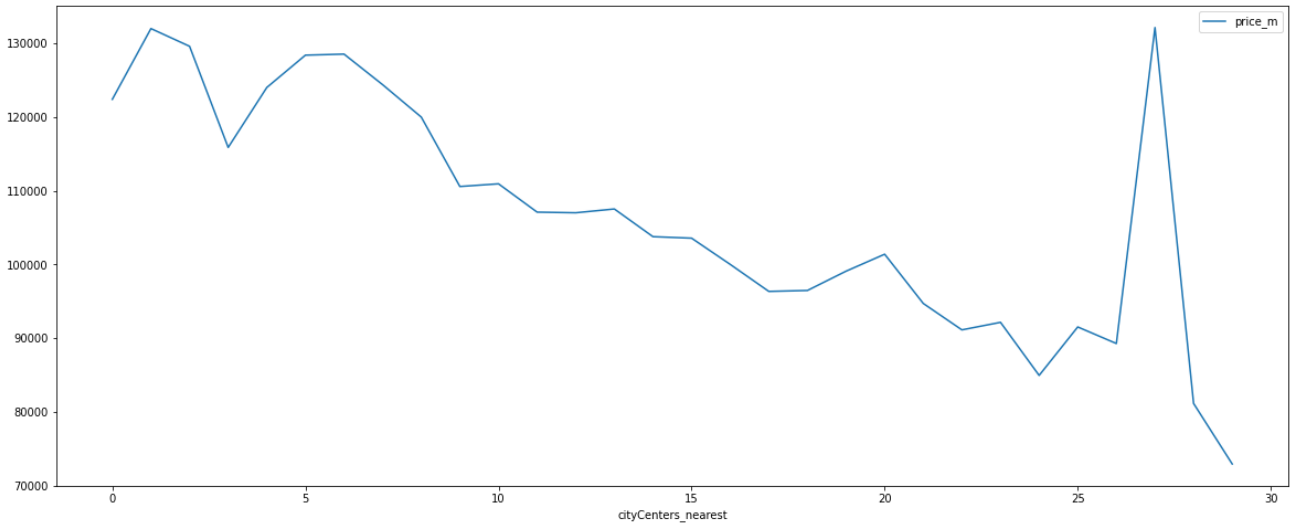
```
top10_price_m = top10_price_m.head(10)
```

```
top10_price_m.sort_values(by = 'mean').plot(kind= 'bar', figsize = (10,6))
plt.show()
```



Изучим зависимость стоимости объектов от расстояния до центра города

```
data.query('locality_name == "Санкт-Петербург"').pivot_table(index = 'cityCenters_neares
plt.show()
```



По графику видно, как средняя цена за квадратный метр уменьшается с отдалением от центра. Выведем сводную таблицу со средними значениями цен за каждый километр.

Изучим причину выбросов на 27 км.

```
data.query('cityCenters_nearest < 28 and cityCenters_nearest > 26')
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_
140	8	16912000	105.70	2016-12-09	2	
439	9	8570000	72.00	2018-08-11	3	
556	0	3500000	28.50	2018-06-06	1	
558	13	4500000	65.50	2017-10-27	3	
748	13	14350000	74.00	2017-11-28	2	
931	8	6650000	69.00	2017-06-20	3	
1138	1	8000000	84.40	2017-08-22	3	
1675	4	3300000	31.00	2017-02-20	1	
1719	12	4200000	38.00	2018-02-12	1	
1904	14	5150000	50.00	2018-11-11	2	
2104	11	3150000	32.00	2018-02-25	1	
2460	15	7500000	78.00	2018-09-19	3	
2776	8	10500000	105.00	2017-12-06	4	
2929	0	5830000	50.00	2017-11-11	2	
2948	23	11350000	75.00	2017-08-15	3	
2953	13	4500000	61.30	2017-10-05	3	
3185	19	4700000	74.20	2017-11-27	4	
3575	7	3680000	42.00	2017-10-31	2	
3579	18	2400000	31.40	2017-09-15	1	
3858	8	5200000	56.00	2019-03-16	2	
3961	11	3500000	31.10	2019-03-13	1	
4400	4	12300000	78.65	2017-09-09	3	
4676	18	3700000	48.10	2018-12-18	2	
4678	11	4300000	59.00	2018-03-14	3	
4822	10	3100000	30.00	2018-09-18	1	
4842	10	3900000	52.40	2019-01-18	2	
4911	14	2999000	36.00	2018-02-12	1	

5168	7	3650000	33.80	2018-01-21	1
5261	8	3200000	31.60	2016-04-02	1
5961	6	2250000	32.00	2018-02-27	1
5968	1	2800000	31.00	2017-12-26	1
6028	5	3350000	30.00	2016-06-26	1
6079	2	3600000	35.20	2019-01-06	1
6223	9	3730000	57.70	2017-06-02	3
6316	5	3200000	30.00	2017-08-07	1
6676	12	3300000	45.00	2018-07-02	2
6872	9	6000000	56.50	2018-12-12	3
7103	10	3650000	35.20	2018-10-15	1
7128	8	3180000	30.00	2018-12-18	1
7295	14	7950000	50.00	2017-07-06	1
7555	4	4700000	63.00	2018-10-23	3
7793	10	3500000	43.00	2018-10-05	2
7936	20	5650000	48.60	2018-10-03	2
7996	17	16600000	106.00	2017-12-02	4
8213	0	2800000	29.54	2016-09-30	1
8262	11	3900000	44.00	2019-02-14	2
8281	17	2290000	31.70	2017-05-29	1
8285	1	3550000	30.00	2017-11-16	1
8466	20	4800000	60.00	2017-12-08	3
9147	5	3500000	42.00	2018-07-18	2
9241	14	7990000	68.00	2017-10-09	2
9883	0	4800000	60.00	2016-05-26	3
10098	9	7400000	70.00	2016-01-27	3
10117	0	4250000	67.00	2015-07-09	3
10162	11	8000000	56.79	2017-10-01	2
10968	4	2850000	39.00	2016-01-12	1

11002	20	5500000	51.00	2019-03-12	2
11230	8	3080000	32.00	2018-03-08	1
11311	14	5790000	50.60	2016-11-08	2
11352	4	3300000	36.00	2017-06-18	1
12022	7	3500000	30.50	2019-03-18	1
12144	10	6300000	52.80	2018-03-06	2
12179	9	4240000	55.30	2017-09-20	3
12466	11	15000000	89.60	2017-01-31	3
12886	5	2450000	35.00	2017-05-26	1
13375	3	3650000	31.00	2016-07-01	1
13620	9	3500000	32.60	2019-01-28	1
15019	1	2870000	38.80	2018-06-30	1
15379	12	3900000	43.50	2019-01-27	2
15415	11	3800000	43.00	2018-03-29	2
15578	20	16000000	101.90	2018-01-08	2
15696	1	5350000	66.10	2018-06-01	3
15721	17	7500000	70.00	2019-01-22	2
15918	9	7600000	63.00	2017-08-03	1
15939	6	3850000	44.20	2018-08-31	2
16097	12	4800000	75.00	2016-05-26	4
16233	10	7700000	63.60	2014-12-10	3
16303	0	4350000	58.00	2017-09-10	3