

Исследование объявлений о продаже квартир

В вашем распоряжении данные сервиса Яндекс.Недвижимость — архив объявлений о продаже квартир в Санкт-Петербурге и соседних населённых пунктах 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.

Импортируем библиотеки pandas и matplotlib, сохраним датафрейм в переменной data и выведем первые 5 строк на экран.

In [1]:	<pre>import seaborn as sns import numpy as np</pre>																																																																																																
In [2]:	<pre>import pandas as pd import matplotlib.pyplot as plt data = pd.read_csv('/datasets/real_estate_data.csv', delimiter='\t') data.head()</pre>																																																																																																
Out[2]:	<table border="1"> <thead> <tr> <th></th><th>total_images</th><th>last_price</th><th>total_area</th><th>first_day_exposition</th><th>rooms</th><th>ceiling_height</th><th>floors_total</th><th>living_area</th><th>floor</th><th>is_apartment</th><th>...</th><th>kitchen_area</th><th>balcony</th><th>locality_name</th><th>airp</th></tr> </thead> <tbody> <tr> <td>0</td><td>20</td><td>13000000.0</td><td>108.0</td><td>2019-03-07T00:00:00</td><td>3</td><td>2.70</td><td>16.0</td><td>51.0</td><td>8</td><td>NaN</td><td>...</td><td>25.0</td><td>NaN</td><td>Санкт-Петербург</td><td></td></tr> <tr> <td>1</td><td>7</td><td>3350000.0</td><td>40.4</td><td>2018-12-04T00:00:00</td><td>1</td><td>NaN</td><td>11.0</td><td>18.6</td><td>1</td><td>NaN</td><td>...</td><td>11.0</td><td>2.0</td><td>посёлок Шушары</td><td></td></tr> <tr> <td>2</td><td>10</td><td>5196000.0</td><td>56.0</td><td>2015-08-20T00:00:00</td><td>2</td><td>NaN</td><td>5.0</td><td>34.3</td><td>4</td><td>NaN</td><td>...</td><td>8.3</td><td>0.0</td><td>Санкт-Петербург</td><td></td></tr> <tr> <td>3</td><td>0</td><td>64900000.0</td><td>159.0</td><td>2015-07-24T00:00:00</td><td>3</td><td>NaN</td><td>14.0</td><td>NaN</td><td>9</td><td>NaN</td><td>...</td><td>NaN</td><td>0.0</td><td>Санкт-Петербург</td><td></td></tr> <tr> <td>4</td><td>2</td><td>10000000.0</td><td>100.0</td><td>2018-06-19T00:00:00</td><td>2</td><td>3.03</td><td>14.0</td><td>32.0</td><td>13</td><td>NaN</td><td>...</td><td>41.0</td><td>NaN</td><td>Санкт-Петербург</td><td></td></tr> </tbody> </table> <p>5 rows × 22 columns</p>		total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcony	locality_name	airp	0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70	16.0	51.0	8	NaN	...	25.0	NaN	Санкт-Петербург		1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN	11.0	18.6	1	NaN	...	11.0	2.0	посёлок Шушары		2	10	5196000.0	56.0	2015-08-20T00:00:00	2	NaN	5.0	34.3	4	NaN	...	8.3	0.0	Санкт-Петербург		3	0	64900000.0	159.0	2015-07-24T00:00:00	3	NaN	14.0	NaN	9	NaN	...	NaN	0.0	Санкт-Петербург		4	2	10000000.0	100.0	2018-06-19T00:00:00	2	3.03	14.0	32.0	13	NaN	...	41.0	NaN	Санкт-Петербург	
	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcony	locality_name	airp																																																																																		
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70	16.0	51.0	8	NaN	...	25.0	NaN	Санкт-Петербург																																																																																			
1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN	11.0	18.6	1	NaN	...	11.0	2.0	посёлок Шушары																																																																																			
2	10	5196000.0	56.0	2015-08-20T00:00:00	2	NaN	5.0	34.3	4	NaN	...	8.3	0.0	Санкт-Петербург																																																																																			
3	0	64900000.0	159.0	2015-07-24T00:00:00	3	NaN	14.0	NaN	9	NaN	...	NaN	0.0	Санкт-Петербург																																																																																			
4	2	10000000.0	100.0	2018-06-19T00:00:00	2	3.03	14.0	32.0	13	NaN	...	41.0	NaN	Санкт-Петербург																																																																																			

Посмотрим общую информацию о данных

In [3]: `data.info()`

```
<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
```

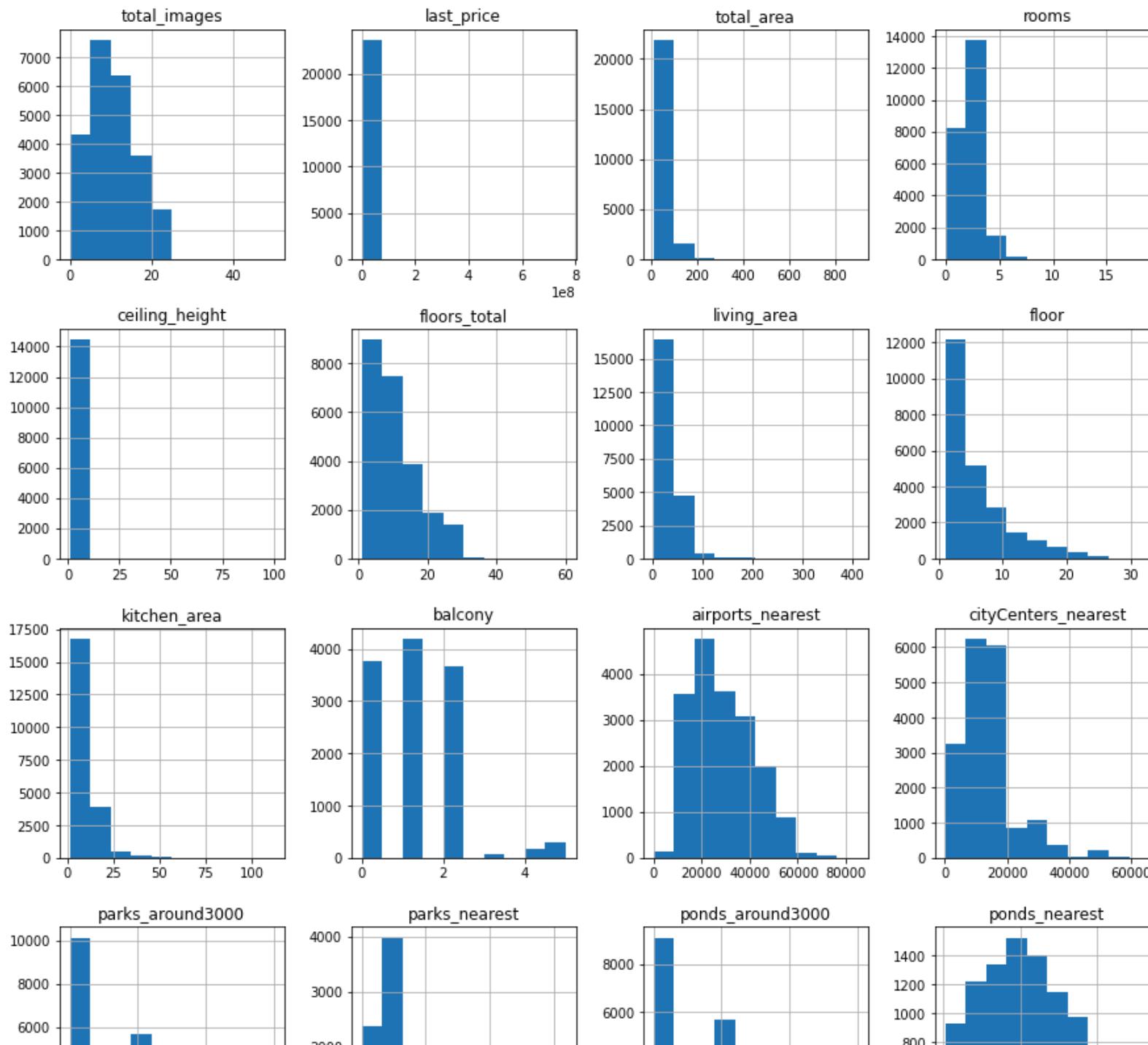
Смущает, что `first_day_exposition` - `object`, а не `datetime`. И ещё `is_apartment` должен быть `bool`. В остальном всё ок.

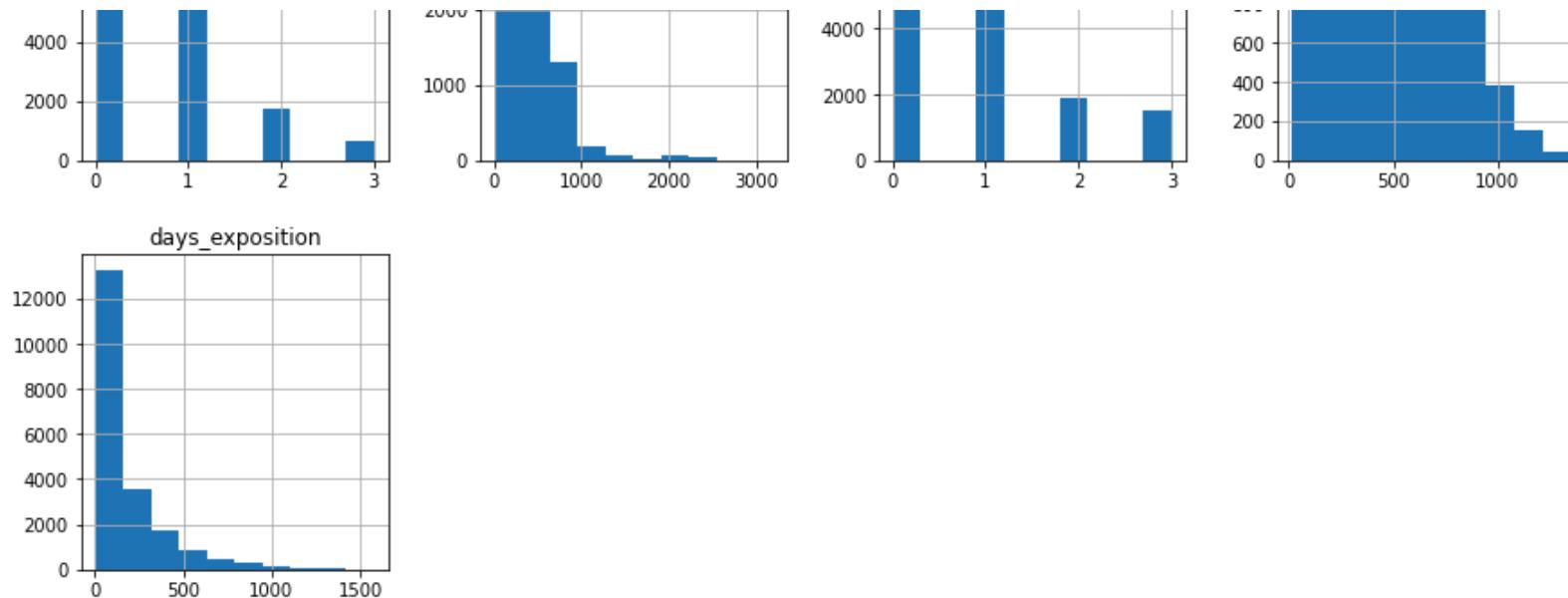
Построим гистограмму для всех числовых столбцов таблицы на одном графике.

In [4]: `data.hist(figsize=(15, 20))`

```
Out[4]: array([[<AxesSubplot:title={'center':'total_images'}>,
   <AxesSubplot:title={'center':'last_price'}>,
   <AxesSubplot:title={'center':'total_area'}>,
   <AxesSubplot:title={'center':'rooms'}>,
   [<AxesSubplot:title={'center':'ceiling_height'}>,
   <AxesSubplot:title={'center':'floors_total'}>,
   <AxesSubplot:title={'center':'living_area'}>,
   <AxesSubplot:title={'center':'floor'}>],
  [<AxesSubplot:title={'center':'kitchen_area'}>,
   <AxesSubplot:title={'center':'balcony'}>,
   <AxesSubplot:title={'center':'airports_nearest'}>,
```

```
<AxesSubplot:title={'center':'cityCenters_nearest'}>],  
[<AxesSubplot:title={'center':'parks_around3000'}>,  
<AxesSubplot:title={'center':'parks_nearest'}>,  
<AxesSubplot:title={'center':'ponds_around3000'}>,  
<AxesSubplot:title={'center':'ponds_nearest'}>],  
[<AxesSubplot:title={'center':'days_exposition'}>, <AxesSubplot:>,  
<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```





Некоторые графики информативны, а некоторые не особо. Не знаю, как их можно использовать сейчас, но, если сказали сделать, значит, вероятно, нужно было. Будем иметь в виду, что они у нас есть.

Предобработка данных

Начнём работы с пропусками

Определим в каких столбцах есть пропуски. Возьмём данные из таблицы data.info.

In [5]:

```
to_fill = data.info()

<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

```

Оставим только те столбцы, в которых есть пропуски.

In [6]:

```

try:
    to_fill = to_fill[to_fill['Non-Null'] != 23699]
except:
    print('Не получается\nЭх')

```

Не получается
Эх

Что ж. Тогда просто вручную запишем названия столбцов и количество имеющихся значений в них. Запомним, что всего в таблице 23699 строк.

Ой, а можете ещё даблкликтнуть на следующий блок?

```

1 ceiling_height 14504 2 floors_total 23613 3 living_area 21796 4 is_apartment 2775 5 kitchen_area 21421 6 balcony 12180 7 locality_name 23650 8 airports_nearest
18157 9 cityCenters_nearest 18180 10 parks_around3000 18181 11 parks_nearest 8079 12 ponds_around3000 18181 13 ponds_nearest 9110 14 days_exposition 20518

```

14 из 22. Не унываем! Если в сыре много дыр - значит, вкусным будет сыр! В конце концов нам платят за рабочие часы. Платят же?

Немного поплачали и можно возвращаться к работе. Начнём заполнять пропуски.

Больше всего пропусков в столбце `is_apartment`, там всего 2775 значений. Чем больше шкаф, тем громче падает. Попробуем разобраться, что к чему.

Обратимся к легенде. `is_apartment` — апартаменты (булев тип). Получается, тут содержится информация о том, является ли жилплощадь апартаментами или нет. Посмотрим на данные.

In [7]:

```
data['is_apartment'].value_counts()
```

```

Out[7]: False    2725
        True     50
Name: is_apartment, dtype: int64

```

Гмм. Нуууу. Я ожидал, что все или почти все значения окажутся `True` и можно будет сделать вывод, что пропуски связаны с тем, что для обычных квартир этот параметр просто не был указан.

Вряд ли на 23699 объявлений было всего 50 апартаментов. Вероятно, данные для этого пункта взяты из отдельной площадки для объявлений или просто были необязательными для заполнения.

Не думаю, что есть смысл пытаться заполнить этот столбец. Давайте просто избавимся от него.

P.S. Если бы мне приставили пистолет ко лбу и сказали, что мне необходимо провести анализ и по этому параметру, то можно было бы сделать отдельный срез данных по этому столбцу и провести сравнение по тому, что имеем. Но учитывая, что в группе 'True' всего 50 значений, а у нас куча других параметров, влияющих на ценообразование, скорее всего мы бы не смогли оценить, даже то, насколько это значение влияет на итоговую цену.

```
In [8]: data.drop('is_apartment', axis=1, inplace=True)
```

Извини, 'is_apartment', тебе не повезло. Ты, наверно, думаешь, что тебе выпало 18 карат невезения? Да нет, просто игра попалась нечестная.

Проверим, всё ли мы сделали правильно.

```
In [9]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 21 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   studio            23699 non-null   bool    
 10  open_plan         23699 non-null   bool    
 11  kitchen_area     21421 non-null   float64 
 12  balcony           12180 non-null   float64 
 13  locality_name    23650 non-null   object  
 14  airports_nearest  18157 non-null   float64 
 15  cityCenters_nearest 18180 non-null   float64 
 16  parks_around3000  18181 non-null   float64 
 17  parks_nearest     8079 non-null   float64 
 18  ponds_around3000  18181 non-null   float64 
 19  ponds_nearest      9110 non-null   float64 
 20  days_exposition    20518 non-null   float64 
dtypes: bool(2), float64(14), int64(3), object(2)
memory usage: 3.5+ MB
```

Отлично. Теперь посмотрим, что там с balcony (число балконов).

In [10]: `data['balcony'].value_counts()`

Out[10]:

1.0	4195
0.0	3758
2.0	3659
5.0	304
4.0	183
3.0	81
Name: balcony, dtype: int64	

Денис не согласен, но в задании сказано, что если не указано число балконов, то их скорее всего 0, и нужно так и заполнить. Сказано - сделано!

In [11]: `data['balcony'] = data['balcony'].fillna(0)`
`data['balcony'].value_counts()`

Out[11]:

0.0	15277
1.0	4195
2.0	3659
5.0	304
4.0	183
3.0	81
Name: balcony, dtype: int64	

Кто на новеньького?

В locality_name (название населенного пункта) не хватает всего 49 значений. Посмотрим, что это за строки и можно ли их заполнить.

In [12]: `data[data['locality_name'].isna() == True]`

Out[12]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
1097	3	8600000.0	81.70	2016-04-15T00:00:00	3	3.55	5.0	50.80	2	False	...	8.80	0.0	NaN	
2033	6	5398000.0	80.00	2017-05-30T00:00:00	3	NaN	4.0	42.60	2	False	...	18.60	0.0	NaN	
2603	20	3351765.0	42.70	2015-09-20T00:00:00	1	NaN	24.0	15.60	3	False	...	10.70	0.0	NaN	
2632	2	5130593.0	62.40	2015-10-11T00:00:00	2	NaN	24.0	33.10	21	False	...	8.20	0.0	NaN	
3574	10	4200000.0	46.50	2016-05-28T00:00:00	2	NaN	5.0	30.80	5	False	...	6.50	0.0	NaN	
4151	17	17600000.0	89.50	2014-12-09T00:00:00	2	3.00	8.0	39.62	7	False	...	13.38	0.0	NaN	
4189	7	9200000.0	80.00	2015-12-10T00:00:00	3	4.00	4.0	52.30	3	False	...	10.40	0.0	NaN	
4670	1	5500000.0	83.00	2015-08-14T00:00:00	3	NaN	7.0	NaN	6	False	...	NaN	0.0	NaN	
5343	19	13540000.0	85.50	2016-01-20T00:00:00	3	NaN	7.0	59.10	5	False	...	8.30	4.0	NaN	

Project_YP3

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
5707	7	3700000.0	30.00	2016-04-29T00:00:00	1	NaN	24.0	20.00	23	False	...	NaN	0.0	NaN	
6765	20	4895892.0	60.70	2015-03-12T00:00:00	2	NaN	24.0	31.90	3	False	...	12.20	0.0	NaN	
7114	5	4250000.0	56.00	2016-03-16T00:00:00	3	NaN	5.0	40.00	4	False	...	6.00	0.0	NaN	
7330	8	5100000.0	63.00	2015-01-27T00:00:00	3	NaN	5.0	42.00	1	False	...	7.50	0.0	NaN	
7600	8	6800000.0	70.00	2016-01-31T00:00:00	3	NaN	11.0	42.00	9	False	...	11.00	1.0	NaN	
8568	10	16000000.0	155.00	2016-05-09T00:00:00	3	NaN	6.0	94.00	3	False	...	23.00	0.0	NaN	
8986	10	4850000.0	103.10	2018-07-10T00:00:00	3	NaN	NaN	68.10	4	False	...	16.70	0.0	NaN	
9821	13	8000000.0	94.50	2015-01-21T00:00:00	4	3.00	2.0	57.80	2	False	...	11.30	0.0	NaN	
10122	5	8200000.0	83.00	2015-06-24T00:00:00	4	NaN	5.0	53.00	2	False	...	10.00	0.0	NaN	
11248	12	6300000.0	63.10	2015-01-16T00:00:00	4	NaN	8.0	44.00	7	False	...	8.70	0.0	NaN	
12879	12	4400000.0	39.20	2016-04-26T00:00:00	1	NaN	12.0	20.00	12	False	...	7.90	0.0	NaN	
12936	6	6800000.0	73.00	2015-11-01T00:00:00	3	NaN	5.0	53.10	2	False	...	8.20	0.0	NaN	
13223	1	2919911.0	29.40	2015-03-12T00:00:00	1	2.75	24.0	21.10	2	False	...	NaN	0.0	NaN	
13690	7	3500000.0	71.00	2016-06-23T00:00:00	3	2.75	2.0	45.60	1	False	...	8.00	2.0	NaN	
14273	2	4422000.0	60.00	2016-03-23T00:00:00	2	2.75	23.0	32.00	14	False	...	11.90	0.0	NaN	
14342	3	3611000.0	53.50	2017-04-27T00:00:00	1	NaN	4.0	25.80	3	False	...	NaN	1.0	NaN	
15686	13	4700000.0	44.00	2015-12-01T00:00:00	2	NaN	5.0	28.00	3	False	...	5.00	0.0	NaN	
15866	10	3950000.0	44.00	2016-04-16T00:00:00	2	2.70	5.0	28.50	5	False	...	5.50	1.0	NaN	
16499	2	4995573.0	56.90	2016-06-17T00:00:00	2	NaN	24.0	29.20	14	False	...	10.90	0.0	NaN	
16561	3	2450000.0	30.00	2016-06-02T00:00:00	1	NaN	4.0	17.00	2	False	...	6.00	1.0	NaN	
16610	11	11940000.0	112.00	2015-11-19T00:00:00	3	3.00	5.0	64.00	2	False	...	23.00	0.0	NaN	
17535	2	5985000.0	79.80	2018-07-30T00:00:00	3	NaN	9.0	NaN	2	False	...	NaN	0.0	NaN	
17764	9	8400000.0	94.00	2016-01-24T00:00:00	3	NaN	23.0	52.00	5	False	...	NaN	0.0	NaN	
18526	3	10800000.0	86.00	2016-06-24T00:00:00	4	3.20	7.0	48.00	2	False	...	12.00	0.0	NaN	
18917	3	2660000.0	37.99	2017-08-17T00:00:00	1	NaN	4.0	13.00	1	False	...	12.40	0.0	NaN	
19045	6	4650000.0	48.00	2016-01-25T00:00:00	2	3.12	5.0	26.20	1	False	...	8.00	0.0	NaN	
19972	20	4361004.0	62.40	2015-09-20T00:00:00	2	NaN	24.0	33.10	21	False	...	8.20	0.0	NaN	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
20057	13	11500000.0	102.00	2015-10-14T00:00:00	2	NaN	5.0	70.00	2	False	...	NaN	0.0	NaN	
20382	8	1750000.0	72.90	2018-10-27T00:00:00	3	NaN	5.0	47.30	2	False	...	8.30	0.0	NaN	
20590	7	3380000.0	56.00	2017-11-06T00:00:00	2	2.70	4.0	29.00	3	False	...	10.00	1.0	NaN	
20654	7	6100000.0	43.00	2016-01-13T00:00:00	1	NaN	5.0	21.00	3	False	...	12.00	1.0	NaN	
21119	8	3500000.0	43.20	2018-11-11T00:00:00	2	NaN	4.0	NaN	2	False	...	NaN	0.0	NaN	
21276	0	17122148.0	178.30	2017-02-10T00:00:00	1	NaN	3.0	NaN	1	False	...	41.60	1.0	NaN	
21333	10	5900000.0	58.00	2015-03-12T00:00:00	3	NaN	6.0	35.20	6	False	...	11.00	0.0	NaN	
21715	2	6047550.0	80.10	2018-07-30T00:00:00	2	NaN	9.0	30.50	2	False	...	29.20	0.0	NaN	
21898	2	5886750.0	83.50	2018-07-30T00:00:00	2	NaN	9.0	36.60	2	False	...	29.70	0.0	NaN	
22474	7	24000000.0	128.00	2015-07-24T00:00:00	4	2.75	6.0	68.40	6	False	...	16.50	0.0	NaN	
22717	9	3000000.0	35.00	2018-01-02T00:00:00	1	2.60	16.0	16.00	7	False	...	10.00	1.0	NaN	
22933	20	3176015.0	33.30	2015-04-22T00:00:00	1	NaN	23.0	15.40	22	False	...	9.00	0.0	NaN	
23214	3	7990000.0	56.00	2016-05-31T00:00:00	2	NaN	6.0	NaN	5	False	...	NaN	0.0	NaN	

49 rows × 21 columns

Посмотрим на last_price и total_area.

Алиса, какая была средняя цена за кв.м в питере в 2019 году?

Если коротко, то говорит, что около 115 000. Похоже на правду, видимо, это Дефолт-Сити 2.

Проверим, как в данных обозначен Санкт-Петербург.

In [13]: `data['locality_name'].value_counts()`

Out[13]:	Санкт-Петербург	15721
	посёлок Мурино	522
	посёлок Шушары	440
	Всеволожск	398
	Пушкин	369
	...	
	деревня Чудской Бор	1
	посёлок при железнодорожной станции Вещево	1
	посёлок Шугозеро	1
	коттеджный поселок Счастье	1

поселок Перово
Name: locality_name, Length: 364, dtype: int64

Разумно. Заполним пропуски и проверим результат.

```
In [14]: data['locality_name'] = data['locality_name'].fillna('Санкт-Петербург')
data['locality_name'].value_counts()
```

```
Out[14]: Санкт-Петербург    15770
посёлок Мурино        522
посёлок Шушары        440
Всеволожск            398
Пушкин                369
...
деревня Чудской Бор    1
посёлок при железнодорожной станции Вещево 1
посёлок Шугозеро      1
коттеджный поселок Счастье 1
поселок Перово        1
Name: locality_name, Length: 364, dtype: int64
```

Да я сегодня в ударе! Продолжим.

Заполним пропуски в city_center (расстояние до центра города в метрах). Там всего 18180 значений. Для начала посмотрим, какие названия населённых пунктов встречаются в пропущенных строках.

```
In [15]: center_range = data.loc[data['cityCenters_nearest'].isna() == True]
center_range['locality_name'].value_counts().head(20)
```

```
Out[15]: посёлок Мурино      522
          Всеволожск        398
          Гатчина           307
          деревня Кудрово    299
          Выборг             237
          Кудрово            173
          деревня Новое Девяткино 144
          Сертолово          142
          Кириши             125
          Сланцы              112
          Волхов               111
          Тосно                104
          Кингисепп          104
          Никольское          93
          Коммунар            89
          Сосновый Бор         87
          Кировск              84
          Отрадное             80
          посёлок Бугры         69
          Санкт-Петербург       69
          Name: locality_name, dtype: int64
```

Здесь я пропал из проектной работы почти на месяц. Я регулярно возвращался, но каждый раз не мог заставить себя просидеть достаточно долго, чтобы сдвинуться с мёртвой точки и реализовать сложное решение для этого столбца и floors_total. И не был готов использовать простое. Попробуем ещё раз сделать всё хорошо.

```
In [16]: center_range['locality_name'].value_counts().head(40)
```

```
Out[16]: посёлок Мурино      522
          Всеволожск        398
          Гатчина           307
          деревня Кудрово    299
          Выборг             237
          Кудрово            173
          деревня Новое Девяткино 144
          Сертолово          142
          Кириши             125
          Сланцы              112
          Волхов               111
          Тосно                104
          Кингисепп          104
          Никольское          93
          Коммунар            89
          Сосновый Бор         87
          Кировск              84
          Отрадное             80
          посёлок Бугры         69
          Санкт-Петербург       69
          Приозерск           66
          деревня Старая         64
```

городской посёлок Янино-1	61
Шлиссельбург	57
Луга	56
Тихвин	49
поселок Бугры	45
посёлок Тельмана	39
Волосово	36
поселок Романовка	36
Мурино	34
поселок Мурино	32
посёлок городского типа Сиверский	29
Ивангород	28
городской посёлок Новоселье	28
городской посёлок Мга	27
поселок городского типа имени Свердлова	25
Сясьстрой	24
посёлок Щеглово	22
посёлок городского типа Кузьмоловский	22
Name: locality_name, dtype: int64	

Очень много деревень и посёлков. Посмотрим, сколько вообще по ним строк с данными.

```
In [17]: data['locality_name'].value_counts().head(40)
```

Санкт-Петербург	15770
посёлок Мурино	522
посёлок Шушары	440
Всеволожск	398
Пушкин	369
Колпино	338
посёлок Парголово	327
Гатчина	307
деревня Кудрово	299
Выборг	237
Петергоф	201
Сестрорецк	183
Красное Село	178
Кудрово	173
деревня Новое Девяткино	144
Сертолово	142
Ломоносов	133
Кириши	125
Сланцы	112
Волхов	111
Тосно	104
Кингисепп	104
Кронштадт	96
Никольское	93
Коммунар	89
Сосновый Бор	87
Кировск	84

Отрадное	80
посёлок Бугры	69
посёлок Металлострой	66
Приозерск	66
деревня Старая	64
городской посёлок Янино-1	61
Шлиссельбург	57
Луга	56
Тихвин	49
поселок Бугры	45
посёлок Стрельна	44
посёлок Тельмана	39
Павловск	38

Name: locality_name, dtype: int64

Большинство пропусков соответствуют деревням, посёлкам и т.п. Скорее всего алгоритму просто не сказали, что для них нужно считать расстояние до центра. Мы не можем их заполнить, да и не считаем нужным. Но есть и пропуски в городах. Пока непонятно, с чем они связаны. Давайте посмотрим, сколько пропусков у нас в Питере.

```
In [18]: center_spb = center_range[center_range['locality_name'] == 'Санкт-Петербург']
center_spb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69 entries, 81 to 22717
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   total_images     69 non-null    int64  
 1   last_price       69 non-null    float64 
 2   total_area       69 non-null    float64 
 3   first_day_exposition  69 non-null  object  
 4   rooms            69 non-null    int64  
 5   ceiling_height   33 non-null    float64 
 6   floors_total     68 non-null    float64 
 7   living_area      52 non-null    float64 
 8   floor             69 non-null    int64  
 9   studio            69 non-null    bool   
 10  open_plan         69 non-null    bool   
 11  kitchen_area     51 non-null    float64 
 12  balcony           69 non-null    float64 
 13  locality_name    69 non-null    object  
 14  airports_nearest  0 non-null    float64 
 15  cityCenters_nearest 0 non-null    float64 
 16  parks_around3000  0 non-null    float64 
 17  parks_nearest    0 non-null    float64 
 18  ponds_around3000  0 non-null    float64 
 19  ponds_nearest    0 non-null    float64 
 20  days_exposition   32 non-null    float64 
dtypes: bool(2), float64(14), int64(3), object(2)
memory usage: 10.9+ KB
```

Всего 69. Удивительно малая доля для самого популярного города. Посмотрим внимательнее на этот срез.

In [19]: center_spb

Out[19]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpo
81	9	10949000.0	68.00	2018-12-27T00:00:00	1	NaN	10.0	NaN	6	False	...	NaN	0.0	Санкт-Петербург	
593	20	4200000.0	38.30	2019-04-16T00:00:00	1	2.6	16.0	19.8	10	False	...	10.1	2.0	Санкт-Петербург	
604	11	9899000.0	101.00	2018-12-12T00:00:00	3	3.0	5.0	62.0	3	False	...	15.0	0.0	Санкт-Петербург	
742	7	37000000.0	161.00	2016-05-22T00:00:00	3	NaN	9.0	84.0	8	False	...	NaN	0.0	Санкт-Петербург	
795	19	7100000.0	59.60	2019-02-13T00:00:00	2	3.4	5.0	36.6	2	False	...	12.5	0.0	Санкт-Петербург	
...	
21898	2	5886750.0	83.50	2018-07-30T00:00:00	2	NaN	9.0	36.6	2	False	...	29.7	0.0	Санкт-Петербург	
21955	19	130000000.0	431.00	2017-10-02T00:00:00	7	3.7	8.0	220.0	5	False	...	20.0	5.0	Санкт-Петербург	
22554	5	3415000.0	31.65	2019-04-16T00:00:00	1	NaN	24.0	NaN	14	False	...	8.7	2.0	Санкт-Петербург	
22628	8	8600000.0	74.90	2019-03-14T00:00:00	3	3.0	5.0	53.2	4	False	...	8.0	1.0	Санкт-Петербург	
22717	9	3000000.0	35.00	2018-01-02T00:00:00	1	2.6	16.0	16.0	7	False	...	10.0	1.0	Санкт-Петербург	

69 rows × 21 columns



Пётр его знает, откуда взяли эти пропуски. Количество этажей разное. Стоимость тоже сильно разная, как и её отношение к площади, так что мою идею с тем, что это дом, который попал в 0 метров до центра следует отвергнуть. Как и идею, что район новоприсоединённый, и его в геосервисе забыли пометить как Питер. Моё лучшее предположение - пользователь указал город неправильно и геосервис выдал ошибку в расстоянии до центра.

Давайте заполнять. Для начала проверим, есть ли в строках с пропусками другие показатели расстояния.

In [20]: center_range

Out[20]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airport
5	10	2890000.0	30.40	2018-09-10T00:00:00	1	NaN	12.0	14.40	5	False	...	9.10	0.0	городской посёлок Янино-1	
8	20	2900000.0	33.16	2018-05-23T00:00:00	1	NaN	27.0	15.43	26	False	...	8.81	0.0	посёлок Муриново	
12	10	3890000.0	54.00	2016-06-30T00:00:00	2	NaN	5.0	30.00	5	False	...	9.00	0.0	Сертолово	
22	20	5000000.0	58.00	2017-04-24T00:00:00	2	2.75	25.0	30.00	15	False	...	11.00	2.0	деревня Кудрово	
30	12	2200000.0	32.80	2018-02-19T00:00:00	1	NaN	9.0	NaN	2	False	...	NaN	0.0	Коммунар	
...
23683	16	2100000.0	62.80	2018-09-18T00:00:00	4	2.50	5.0	45.50	3	False	...	5.50	0.0	посёлок Дзержинского	
23692	2	1350000.0	30.00	2017-07-07T00:00:00	1	NaN	5.0	17.50	4	False	...	6.00	0.0	Тихвин	
23695	14	3100000.0	59.00	2018-01-15T00:00:00	3	NaN	5.0	38.00	4	False	...	8.50	0.0	Тосно	
23696	18	2500000.0	56.70	2018-02-11T00:00:00	2	NaN	3.0	29.70	1	False	...	NaN	0.0	село Рождествено	
23698	4	1350000.0	32.30	2017-07-21T00:00:00	1	2.50	5.0	12.30	1	False	...	9.00	0.0	поселок Новый Учхоз	

5519 rows × 21 columns



Везде всё по нулям. Нет, знаете... Я честно пытался. И я решил сдаться. Это явно не будет критичным для наших будущих рассчётов. Давайте просто заполним пропуски значением -1 и пойдём дальше.

In [21]: `data['cityCenters_nearest'].fillna(-1, inplace=True)`

In [22]: `data['cityCenters_nearest'].isna().sum()`

Out[22]: 0

Заполнили, проверили, осталось только прожить все стадии горя.

В floors_total (кол-во этажей в доме) 23613 значений. Пропусков мало. Сначала взглянем поближе.

In [23]:

```
data[data['floors_total'].isna() == True]
```

Out[23]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
186	12	11640000.0	65.2	2018-10-02T00:00:00	2	NaN	NaN	30.80	4	False	...	12.00	0.0	Санкт-Петербург	
237	4	2438033.0	28.1	2016-11-23T00:00:00	1	NaN	NaN	20.75	1	False	...	NaN	0.0	Санкт-Петербург	
457	4	9788348.0	70.8	2015-08-01T00:00:00	2	NaN	NaN	38.40	12	False	...	10.63	0.0	Санкт-Петербург	
671	4	6051191.0	93.6	2017-04-06T00:00:00	3	NaN	NaN	47.10	8	False	...	16.80	0.0	Санкт-Петербург	
1757	5	3600000.0	39.0	2017-04-22T00:00:00	1	NaN	NaN	NaN	9	False	...	NaN	0.0	Санкт-Петербург	
...	
22542	5	8500000.0	63.5	2017-05-24T00:00:00	2	2.8	NaN	NaN	3	False	...	NaN	0.0	Санкт-Петербург	
22656	4	4574160.0	64.5	2017-04-02T00:00:00	2	NaN	NaN	31.70	20	False	...	14.40	0.0	Санкт-Петербург	
22808	0	14569263.0	110.4	2016-11-20T00:00:00	3	NaN	NaN	45.38	6	False	...	23.42	0.0	Санкт-Петербург	
23590	0	21187872.0	123.3	2017-04-25T00:00:00	3	NaN	NaN	50.40	18	False	...	23.60	0.0	Санкт-Петербург	
23658	6	3063600.0	43.8	2016-11-28T00:00:00	1	2.7	NaN	14.00	8	False	...	15.50	2.0	Санкт-Петербург	

86 rows × 21 columns

Вроде бы ничего важного. Интересно, что на 10 строк один дом встретился 3 раза и ещё один 2 раза. Захотелось посмотреть ещё детальнее.

In [24]:

```
hmmm = data[data['floors_total'].isna() == True]
hmmm['airports_nearest'].nunique()
```

Out[24]: 41

41 дом на 86 строк. Чёрт, мне стало интересно.

In [25]: `data['airports_nearest'].nunique()`

Out[25]: 8275

С этими числами я обратился к дипсику. Если коротко, то он сказал, что вероятность такого количества повторов в выборке на 86 значений чрезвычайно мала, но для точного расчёта нужно больше информации, например о том, как именно распределены повторы в основной выборке.

Мне чертовски интересно. Однако, боюсь, что придётся убрать этот проект в нижний ящик стола и вернуться к работе.

Доизучим столбец. Проверим минимальное и максимальное значения.

In [26]: `data['floors_total'].min()`

Out[26]: 1.0

In [27]: `data['floors_total'].max()`

Out[27]: 60.0

Звучит реалистично. Может быть кто-то из пользователей поставил 0? Поскольку чётких взаимосвязей с пропусками в других столбцах не видно, да и самих пропусков мало, думаю, что вряд ли пользователям было необходимо заполнять данный пункт. Может быть данные потерялись во время выгрузки/обработки, но тогда, мне кажется, пропусков было бы больше.

Заполним пропуски в floors_total.

```
In [28]: for row in data['floors_total']:
    try:
        if row.isna() == True:
            df = data[data['cityCenters_nearest'] == row['cityCenters_nearest']]
            row = df['floors_total'].median()
    except AttributeError:
        continue
```

Эту часть я пытался сделать до упомянутого ранее месячного простоя. Этот кусок кода определённо не должен быть доведён до ума и запущен. Остановимся на том, что похвалим себя за изобретательность и вновь заменим пропуски на -1.

In [29]: `data['floors_total'].fillna(-1, inplace=True)`

In [30]: `data['floors_total'].isna().sum()`

Out[30]: 0

Тем временем я выучил новый метод вывода информации о пропусках.

In [31]:

```
data.isna().sum()
```

```
Out[31]: total_images          0
last_price           0
total_area           0
first_day_exposition 0
rooms                0
ceiling_height       9195
floors_total         0
living_area          1903
floor                0
studio               0
open_plan             0
kitchen_area         2278
balcony              0
locality_name        0
airports_nearest     5542
cityCenters_nearest   0
parks_around3000      5518
parks_nearest         15620
ponds_around3000      5518
ponds_nearest         14589
days_exposition       3181
dtype: int64
```

Давайте разберёмся с парками - столбцы parks_nearest и parks_around3000. Столбцы явно связаны. Насколько я понимаю, все parks_nearest будут в пределах 3 км. Давайте проверим это.

In [32]:

```
data[data['parks_nearest'] > 3000]
```

Out[32]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
1590	3	7500000.0	70.0	2017-07-11T00:00:00	2	NaN	18.0	37.0	13	False	...	10.0	0.0	Санкт-Петербург	
10959	9	3000000.0	31.1	2017-03-21T00:00:00	1	NaN	5.0	17.7	2	False	...	5.5	0.0	Санкт-Петербург	
19208	17	14950000.0	187.0	2017-10-12T00:00:00	5	3.0	2.0	80.0	2	False	...	17.0	1.0	Санкт-Петербург	
19430	9	3900000.0	30.5	2018-02-22T00:00:00	1	2.6	5.0	16.5	1	False	...	5.5	0.0	Санкт-Петербург	

4 rows × 21 columns

Ага, интересно. Получается, что геосервис для сбора инфы о ближайшем парке сделал пределом не 3км, а чуть большее расстояние. А ещё можно сделать вывод, что данные для этого столбца брались не из parks_around3000, а отдельно.

Давайте посмотрим parks_around3000.

In [33]:

```
data[data['parks_around3000'].isna()].head(10)
```

Out[33]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports_ne
5	10	2890000.0	30.40	2018-09-10T00:00:00	1	NaN	12.0	14.40	5	False	...	9.10	0.0	городской посёлок Янино-1	
8	20	2900000.0	33.16	2018-05-23T00:00:00	1	NaN	27.0	15.43	26	False	...	8.81	0.0	посёлок Мурино	
12	10	3890000.0	54.00	2016-06-30T00:00:00	2	NaN	5.0	30.00	5	False	...	9.00	0.0	Сертолово	
22	20	5000000.0	58.00	2017-04-24T00:00:00	2	2.75	25.0	30.00	15	False	...	11.00	2.0	деревня Кудрово	
30	12	2200000.0	32.80	2018-02-19T00:00:00	1	NaN	9.0	NaN	2	False	...	NaN	0.0	Коммунар	
37	10	1990000.0	45.80	2017-10-28T00:00:00	2	2.50	5.0	NaN	1	False	...	NaN	0.0	поселок городского типа Красный Бор	
38	10	3150000.0	40.00	2018-03-29T00:00:00	1	2.75	18.0	16.30	9	False	...	11.60	0.0	посёлок Мурино	
47	17	3600000.0	56.10	2018-10-18T00:00:00	3	NaN	4.0	42.50	3	False	...	5.70	1.0	Гатчина	
60	3	2740000.0	35.00	2018-01-01T00:00:00	1	NaN	12.0	NaN	8	False	...	NaN	0.0	посёлок Мурино	
62	0	4800000.0	78.60	2017-09-17T00:00:00	3	2.80	9.0	48.80	5	False	...	11.90	2.0	Сертолово	

10 rows × 21 columns



Если тут Nan, то и в nearest тоже Nan, логично. Попробую на всякий случай проверить. Сравним с пропусками в parks_around3000 и таблицу с пропусками в обоих столбцах.

In [34]:

```
data[data['parks_around3000'].isna()]
```

Out[34]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airport
5	10	2890000.0	30.40	2018-09-10T00:00:00	1	NaN	12.0	14.40	5	False	...	9.10	0.0	городской посёлок Янино-1	
8	20	2900000.0	33.16	2018-05-23T00:00:00	1	NaN	27.0	15.43	26	False	...	8.81	0.0	посёлок Мурино	
12	10	3890000.0	54.00	2016-06-30T00:00:00	2	NaN	5.0	30.00	5	False	...	9.00	0.0	Сертолово	
22	20	5000000.0	58.00	2017-04-24T00:00:00	2	2.75	25.0	30.00	15	False	...	11.00	2.0	деревня Кудрово	
30	12	2200000.0	32.80	2018-02-19T00:00:00	1	NaN	9.0	NaN	2	False	...	NaN	0.0	Коммунар	
...
23683	16	2100000.0	62.80	2018-09-18T00:00:00	4	2.50	5.0	45.50	3	False	...	5.50	0.0	посёлок Дзержинского	
23692	2	1350000.0	30.00	2017-07-07T00:00:00	1	NaN	5.0	17.50	4	False	...	6.00	0.0	Тихвин	
23695	14	3100000.0	59.00	2018-01-15T00:00:00	3	NaN	5.0	38.00	4	False	...	8.50	0.0	Тосно	
23696	18	2500000.0	56.70	2018-02-11T00:00:00	2	NaN	3.0	29.70	1	False	...	NaN	0.0	село Рождествено	
23698	4	1350000.0	32.30	2017-07-21T00:00:00	1	2.50	5.0	12.30	1	False	...	9.00	0.0	поселок Новый Учхоз	

5518 rows × 21 columns



In [35]:

```
data.loc[(data['parks_around3000'].isna()) & (data['parks_nearest'].isna())]
```

Out[35]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airport
5	10	2890000.0	30.40	2018-09-10T00:00:00	1	NaN	12.0	14.40	5	False	...	9.10	0.0	городской посёлок Янино-1	
8	20	2900000.0	33.16	2018-05-23T00:00:00	1	NaN	27.0	15.43	26	False	...	8.81	0.0	посёлок Мурино	
12	10	3890000.0	54.00	2016-06-30T00:00:00	2	NaN	5.0	30.00	5	False	...	9.00	0.0	Сертолово	
22	20	5000000.0	58.00	2017-04-24T00:00:00	2	2.75	25.0	30.00	15	False	...	11.00	2.0	деревня Кудрово	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airport
30	12	2200000.0	32.80	2018-02-19T00:00:00	1	NaN	9.0	NaN	2	False	...	NaN	0.0	Коммунар	
...	
23683	16	2100000.0	62.80	2018-09-18T00:00:00	4	2.50	5.0	45.50	3	False	...	5.50	0.0	посёлок Дзержинского	
23692	2	1350000.0	30.00	2017-07-07T00:00:00	1	NaN	5.0	17.50	4	False	...	6.00	0.0	Тихвин	
23695	14	3100000.0	59.00	2018-01-15T00:00:00	3	NaN	5.0	38.00	4	False	...	8.50	0.0	Тосно	
23696	18	2500000.0	56.70	2018-02-11T00:00:00	2	NaN	3.0	29.70	1	False	...	NaN	0.0	село Рождествено	
23698	4	1350000.0	32.30	2017-07-21T00:00:00	1	2.50	5.0	12.30	1	False	...	9.00	0.0	поселок Новый Учхоз	

5518 rows × 21 columns

Обе таблицы имеют одинаковое количество колонок. С этим разобрались. Теперь нужно понять, почему в parks_nearest больше пропусков.

In [36]:

```
data[data['parks_around3000'].isna() == False]
```

Out[36]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
0	20	13000000.0	108.00	2019-03-07T00:00:00	3	2.70	16.0	51.0	8	False	...	25.00	0.0	Санкт-Петербург	
1	7	3350000.0	40.40	2018-12-04T00:00:00	1	NaN	11.0	18.6	1	False	...	11.00	2.0	посёлок Шушары	
2	10	5196000.0	56.00	2015-08-20T00:00:00	2	NaN	5.0	34.3	4	False	...	8.30	0.0	Санкт-Петербург	
3	0	64900000.0	159.00	2015-07-24T00:00:00	3	NaN	14.0	NaN	9	False	...	NaN	0.0	Санкт-Петербург	
4	2	10000000.0	100.00	2018-06-19T00:00:00	2	3.03	14.0	32.0	13	False	...	41.00	0.0	Санкт-Петербург	
...	
23690	3	5500000.0	52.00	2018-07-19T00:00:00	2	NaN	5.0	31.0	2	False	...	6.00	0.0	Санкт-Петербург	
23691	11	9470000.0	72.90	2016-10-13T00:00:00	2	2.75	25.0	40.3	7	False	...	10.60	1.0	Санкт-Петербург	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
23693	9	4600000.0	62.40	2016-08-05T00:00:00	3	2.60	9.0	40.0	8	False	...	8.00	0.0	Петергоф	
23694	9	9700000.0	133.81	2017-03-21T00:00:00	3	3.70	5.0	73.3	3	False	...	13.83	0.0	Санкт-Петербург	
23697	13	11475000.0	76.75	2017-03-28T00:00:00	2	3.00	17.0	NaN	12	False	...	23.30	2.0	Санкт-Петербург	

18181 rows × 21 columns

Мы видим две строчки, в которых в parks_around3000 указано '0', а в parks_nearest стоит Nan. Логично. Жаль, что я не предложил это в качестве гипотезы ранее. Проверим, объясняется ли этим вся разница в пропусках. Сравним количество нулевых значений с разнице в количестве пропусков. Не забудем про 4 значения, из parks_nearest, где парки были дальше 3-х км.

```
In [37]: x = 15620 + 4 - 5518
x
```

Out[37]: 10106

```
In [38]: data['parks_around3000'].value_counts()
```

```
Out[38]: 0.0    10106
1.0    5681
2.0    1747
3.0    647
Name: parks_around3000, dtype: int64
```

Идеально. Давайте там, где в parks_around3000 стоит '0', заполним пропуски в parks_nearest значениями '-1'. Так мы обозначим, обозначим, что информация не пропущена, а должна отсутствовать. А там, где пропуски в обоих столбцах оставим всё как есть.

```
In [39]: data.loc[(data['parks_around3000'] == 0), 'parks_nearest'] = -1
```

```
In [40]: data['parks_nearest'].value_counts()
```

```
Out[40]: -1.0      10106
441.0      67
173.0      41
392.0      41
456.0      40
...
786.0      1
```

```
730.0      1
1186.0     1
726.0      1
929.0      1
Name: parks_nearest, Length: 993, dtype: int64
```

Проверили, всё правильно. Теперь попробуем аналогичным образом изучить ponds_around3000 и ponds_nearest (пруды), а затем заполнить их.

In [41]:

```
data[data['ponds_nearest'] > 3000]
```

Out[41]:

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports_nearby
--------------	------------	------------	----------------------	-------	----------------	--------------	-------------	-------	--------	-----	--------------	---------	---------------	-----------------

0 rows × 21 columns

Тут мне в голову пришла мысль, что парки, находящиеся дальше 3 км, могли попасть попасть из-за того, что расстояние считалось до центра парка, а алгоритм учитывал все парки, которые хотя бы краешком дотягивались до нужного радиуса.

Проверим, есть ли значения в ponds_nearest в строках, где в ponds_around3000 Nan.

In [42]:

```
data.loc[(data['ponds_around3000'].isna() == True) & (data['ponds_nearest'].isna() == False)]
```

Out[42]:

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports_nearby
--------------	------------	------------	----------------------	-------	----------------	--------------	-------------	-------	--------	-----	--------------	---------	---------------	-----------------

0 rows × 21 columns

Почему я не сделал так в первый раз? Хорошо, что я догадался сейчас!

На всякий случай проверим строки, где в ponds_around3000 указано '0'

In [43]:

```
data[data['ponds_around3000'] == 0]
```

Out[43]:

total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports_nearby
--------------	------------	------------	----------------------	-------	----------------	--------------	-------------	-------	--------	-----	--------------	---------	---------------	-----------------

1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN	11.0	18.6	1	False	...	11.0	2.0	посёлок Шушары
6	6	3700000.0	37.3	2017-11-02T00:00:00	1	NaN	26.0	10.6	6	False	...	14.4	1.0	посёлок Парголово
7	5	7915000.0	71.6	2019-04-18T00:00:00	2	NaN	24.0	NaN	22	False	...	18.9	2.0	Санкт-Петербург

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
9	18	5400000.0	61.0	2017-02-26T00:00:00	3	2.50	9.0	43.6	7	False	...	6.5	2.0	Санкт-Петербург	
11	9	3300000.0	44.0	2018-08-27T00:00:00	2	NaN	5.0	31.0	4	False	...	6.0	1.0	Ломоносов	
...	
23684	20	21400000.0	145.0	2018-11-02T00:00:00	4	3.00	26.0	71.4	17	False	...	15.6	0.0	Санкт-Петербург	
23685	15	2490000.0	31.0	2019-01-24T00:00:00	1	2.50	5.0	17.3	5	False	...	5.6	1.0	Ломоносов	
23687	6	3200000.0	39.0	2017-12-16T00:00:00	1	NaN	9.0	20.0	6	False	...	11.0	2.0	Санкт-Петербург	
23688	19	9200000.0	101.0	2019-04-01T00:00:00	3	3.05	15.0	63.0	12	False	...	15.0	0.0	Санкт-Петербург	
23690	3	5500000.0	52.0	2018-07-19T00:00:00	2	NaN	5.0	31.0	2	False	...	6.0	0.0	Санкт-Петербург	

9071 rows × 21 columns

Все как на подбор. Посчитаем *пропуски ponds_nearest* - пропуски *ponds_around3000* - *ponds_around3000 = 0*, чтобы окончательно убедиться в том, что всё просто и понятно.

```
In [44]: y = 14589 - 5518 - 9071
y
```

Out[44]: 0

Красота. Заполним пропуски в строчках с '0' значениями '-1'.

```
In [45]: data.loc[(data['ponds_around3000'] == 0), 'ponds_nearest'] = -1
```

Подводя итог по паркам и прудам, нужно обдумать, с чем могут быть связаны пропуски. Мы знаем, что данные получены из геосервисов. Я думаю, что часть пропусков снова связаны с деревнями и т.п. Давайте проверим это для парков.

```
In [46]: park_range = data.loc[data['parks_around3000'].isna() == True]
park_range['locality_name'].value_counts().head(20)
```

```
Out[46]: посёлок Мурино      522
Всеволожск                  398
```

```
Гатчина          307
деревня Кудрово   299
Выборг           237
Кудрово          173
деревня Новое Девяткино 144
Сертолово        142
Кириши           125
Сланцы            112
Волхов            111
Кингисепп       104
Тосно             104
Никольское        93
Коммунар         89
Сосновый Бор      87
Кировск          84
Отрадное          80
Санкт-Петербург    69
посёлок Бугры     69
Name: locality_name, dtype: int64
```

Предположение подтвердилось.

Сверимся со списком пропусков.

```
In [47]: data.isna().sum()
```

```
Out[47]: total_images          0
last_price            0
total_area            0
first_day_exposition  0
rooms                 0
ceiling_height        9195
floors_total          0
living_area           1903
floor                 0
studio                0
open_plan              0
kitchen_area          2278
balcony               0
locality_name          0
airports_nearest      5542
cityCenters_nearest    0
parks_around3000      5518
parks_nearest          5518
ponds_around3000       5518
ponds_nearest          5518
days_exposition        3181
dtype: int64
```

Осталось всего 5 столбцов. Уже даже дышится легко и свежо.

Давайте займёмся ceiling_height (высота потолков в метрах). Там больше всего пропусков.

```
In [48]: data['ceiling_height'].value_counts()
```

```
Out[48]: 2.50      3515
2.60      1646
2.70      1574
3.00      1112
2.80      993
...
4.25      1
3.39      1
3.93      1
10.30     1
4.65      1
Name: ceiling_height, Length: 183, dtype: int64
```

```
In [49]: data['ceiling_height'].value_counts().tail(20)
```

```
Out[49]: 3.58      1
4.30      1
26.00     1
2.49      1
3.84      1
2.89      1
5.20      1
4.90      1
3.76      1
22.60     1
14.00     1
2.25      1
3.88      1
24.00     1
8.30      1
4.25      1
3.39      1
3.93      1
10.30     1
4.65      1
Name: ceiling_height, dtype: int64
```

Популярнее всего округлённые до десятых значения. Наименее популярные - высокие значения, как правило неокруглённые. Видим много экстремально высоких значений. Можно будет на следующем этапе посмотреть на них детальнее, и, если поделить на 10 высоту для тех квартир, которые нестоят космических денег.

Можно было бы написать себе сложную функцию, которая бы считала медиану высоты потолков по группам с идентичным расстоянием до центра, назвать её boomstick и быть довольным собой. Но я буду больше доволен, если быстрее закончу этот проект на 4/5. Поэтому скажу, что данные этого столбца особо не связешь с другими и заполнять нам его особо нечем. Оставим пропуски как есть.

Перейдём к airports_nearest. Давайте сразу посмотрим список населённых пунктов.

```
In [50]: data[data['airports_nearest'].isna() == True]['locality_name'].value_counts()
```

```
Out[50]: посёлок Мурино      522
Всеволожск                 398
Гатчина                     307
деревня Кудрово             299
Выборг                      237
...
деревня Нижние Осельки      1
поселок Калитино            1
деревня Старое Хинколово    1
поселок Гладкое              1
деревня Русско                  1
Name: locality_name, Length: 344, dtype: int64
```

Ну да, всё те же знакомые лица. В эна этом столбце немного больше пропусков, чем осталось в парках и прудах. То есть тут есть ещё 24 пропуска, связанных с другими причинами. Но искать их и пытаться понять причину, вероятно, нет никакого смысла. Оставляем всё как есть.

Перейдём к days_exposition (сколько дней висело объявление). Там 3181 пропусков. Не очень много, но существенно.

```
In [51]: data[data['days_exposition'].isna()].head(15)
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70	16.0	51.00	8	False	...	25.0	0.0	Санкт-Петербург	
7	5	7915000.0	71.6	2019-04-18T00:00:00	2	NaN	24.0	NaN	22	False	...	18.9	2.0	Санкт-Петербург	
44	13	5350000.0	40.0	2018-11-18T00:00:00	1	NaN	22.0	NaN	3	False	...	NaN	1.0	Санкт-Петербург	
45	17	5200000.0	50.6	2018-12-02T00:00:00	2	2.65	9.0	30.30	7	False	...	7.0	0.0	Санкт-Петербург	
46	17	6600000.0	52.1	2019-01-31T00:00:00	2	2.60	24.0	29.70	9	False	...	8.3	2.0	Санкт-Петербург	
49	1	3050000.0	30.8	2018-11-22T00:00:00	1	2.50	9.0	18.00	7	False	...	6.0	0.0	Санкт-Петербург	
52	20	11795000.0	136.0	2017-09-22T00:00:00	6	3.00	2.0	94.00	2	False	...	11.0	1.0	Санкт-Петербург	
71	20	4850000.0	57.5	2019-03-27T00:00:00	3	2.50	9.0	39.00	8	False	...	6.2	0.0	Санкт-Петербург	

Project_YP3

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airports
81	9	10949000.0	68.0	2018-12-27T00:00:00	1	NaN	10.0	NaN	6	False	...	NaN	0.0	Санкт-Петербург	
84	11	2400000.0	43.9	2019-03-13T00:00:00	2	NaN	2.0	27.46	1	False	...	5.2	0.0	Кировск	
87	19	4999000.0	52.0	2019-04-23T00:00:00	2	2.61	9.0	30.00	6	False	...	9.0	2.0	Санкт-Петербург	
88	0	1700000.0	33.1	2019-04-09T00:00:00	1	NaN	5.0	18.40	1	False	...	6.4	0.0	посёлок городского типа Лебяжье	
101	4	3000000.0	37.0	2019-03-23T00:00:00	1	2.75	10.0	12.00	3	False	...	13.0	0.0	садовое товарищество Новая Ропша	
108	9	9490000.0	80.0	2017-10-30T00:00:00	3	2.55	10.0	46.00	7	False	...	11.0	1.0	Санкт-Петербург	
114	5	4000000.0	21.4	2019-02-07T00:00:00	1	2.90	8.0	8.30	7	False	...	6.3	0.0	Санкт-Петербург	

15 rows × 21 columns

В основном у объявлений в поле first_day_exposition стоят даты из начала 2019 года. Можно предположить, что примерно в это время и была сделана выгрузка. Но, если принять такую гипотезу, старых объявлений тоже хватает. Давайте посмотрим детальнее.

In [52]: `data['days_exposition'].value_counts()`

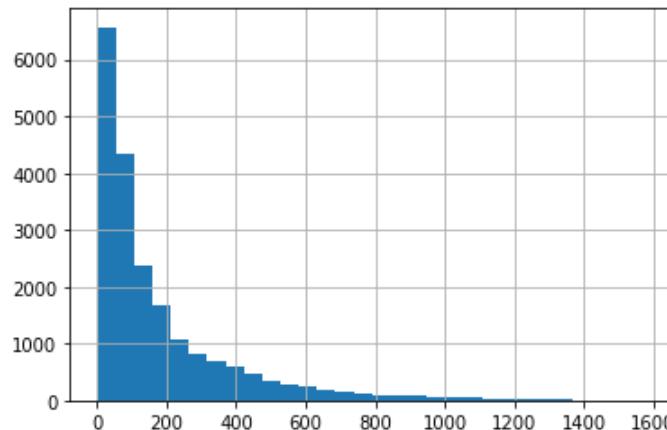
Out[52]:

45.0	880
60.0	538
7.0	234
30.0	208
90.0	204
...	
969.0	1
1.0	1
1147.0	1
1148.0	1
1174.0	1

Name: days_exposition, Length: 1141, dtype: int64

In [53]: `data['days_exposition'].hist(bins=30)`

Out[53]: <AxesSubplot:>



Нужно ещё назвать оси. Сделаем это позже. Как мы видим, объявления, которые висели более года хватает. Думаю, мы можем предположить, что Nan стоит у тех объявлений, которые ещё не сняты с публикации и косяков с этим столбцом не было, как и со столбцом first_day_exposition.

Остались kitchen_area (2278 пропусков) и living_area (1903 пропусков). Давайте посмотрим, насколько они пересекаются.

```
In [54]: data.loc[(data['kitchen_area'].isna() == True) & (data['living_area'].isna() == True)]
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
3	0	64900000.0	159.0	2015-07-24T00:00:00	3	NaN	14.0	NaN	9	False	...	NaN	0.0	Санкт-Петербург	
30	12	2200000.0	32.8	2018-02-19T00:00:00	1	NaN	9.0	NaN	2	False	...	NaN	0.0	Коммунар	
37	10	1990000.0	45.8	2017-10-28T00:00:00	2	2.50	5.0	NaN	1	False	...	NaN	0.0	поселок городского типа Красный Бор	
44	13	5350000.0	40.0	2018-11-18T00:00:00	1	NaN	22.0	NaN	3	False	...	NaN	1.0	Санкт-Петербург	
59	15	6300000.0	46.5	2017-03-06T00:00:00	1	NaN	13.0	NaN	13	False	...	NaN	0.0	Санкт-Петербург	
...
23632	20	5000000.0	38.0	2018-06-23T00:00:00	1	2.55	16.0	NaN	15	False	...	NaN	1.0	Санкт-Петербург	
23642	14	5950000.0	80.3	2018-07-03T00:00:00	3	2.70	12.0	NaN	5	False	...	NaN	0.0	Колпино	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	kitchen_area	balcony	locality_name	airpor
23663	12	6400000.0	88.0	2018-01-12T00:00:00	4	2.55	10.0	NaN	1	False	...	NaN	0.0	Санкт-Петербург	
23670	9	2450000.0	45.6	2018-02-08T00:00:00	1	2.80	9.0	NaN	2	False	...	NaN	0.0	поселок городского типа Синявино	
23679	0	2500000.0	35.0	2017-12-08T00:00:00	1	NaN	9.0	NaN	4	False	...	NaN	0.0	Сосновый Бор	

1464 rows × 21 columns

1464 пропусков. Много пересечений, но хватает и уникальных пропусков. Можно предположить, что эти поля не были обязательными для заполнения пользователями. Опять же, мы могли бы написать код, который бы достаточно неплохо предсказал пропущенные значения и заполнил их, но ведь это не поможет нам в анализе. Нам нужно оценить, как эти факторы влияют на стоимость, а значит усредненные показатели будут только ухудшать качество исследования. Оставим пропуски.

In [55]:

```
data.isna().sum()
```

Out[55]:

total_images	0
last_price	0
total_area	0
first_day_exposition	0
rooms	0
ceiling_height	9195
floors_total	0
living_area	1903
floor	0
studio	0
open_plan	0
kitchen_area	2278
balcony	0
locality_name	0
airports_nearest	5542
cityCenters_nearest	0
parks_around3000	5518
parks_nearest	5518
ponds_around3000	5518
ponds_nearest	5518
days_exposition	3181
	dtype: int64

Я доволен проделанной работой!

Разберёмся с типами данных.

Снова обратимся к инфе о таблице.

In [56]:

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 21 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     23699 non-null   float64 
 7   living_area      21796 non-null   float64 
 8   floor             23699 non-null   int64  
 9   studio            23699 non-null   bool    
 10  open_plan         23699 non-null   bool    
 11  kitchen_area     21421 non-null   float64 
 12  balcony           23699 non-null   float64 
 13  locality_name    23699 non-null   object  
 14  airports_nearest  18157 non-null   float64 
 15  cityCenters_nearest 23699 non-null   float64 
 16  parks_around3000  18181 non-null   float64 
 17  parks_nearest     18181 non-null   float64 
 18  ponds_around3000  18181 non-null   float64 
 19  ponds_nearest     18181 non-null   float64 
 20  days_exposition   20518 non-null   float64 
dtypes: bool(2), float64(14), int64(3), object(2)
memory usage: 3.5+ MB
```

Меняем last_price, floors_total, balcony, days_exposition на int, а first_day_exposition меняем на datetime и округляем до дат.

In [57]:

```
data['last_price'] = data['last_price'].astype(int)
data['floors_total'] = data['floors_total'].astype(int)
data['balcony'] = data['balcony'].astype(int)
data['first_day_exposition'] = pd.to_datetime(data['first_day_exposition'], format="%Y-%m-%dT%H:%M:%S")
data['first_day_exposition'] = data['first_day_exposition'].dt.floor('D')
```

Проверим.

In [58]:

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 21 columns):
```

```

#   Column           Non-Null Count Dtype  
---  --  
0   total_images    23699 non-null   int64  
1   last_price      23699 non-null   int64  
2   total_area      23699 non-null   float64 
3   first_day_exposition  23699 non-null   datetime64[ns] 
4   rooms           23699 non-null   int64  
5   ceiling_height  14504 non-null   float64 
6   floors_total    23699 non-null   int64  
7   living_area     21796 non-null   float64 
8   floor           23699 non-null   int64  
9   studio          23699 non-null   bool  
10  open_plan       23699 non-null   bool  
11  kitchen_area    21421 non-null   float64 
12  balcony         23699 non-null   int64  
13  locality_name   23699 non-null   object  
14  airports_nearest 18157 non-null   float64 
15  cityCenters_nearest 23699 non-null   float64 
16  parks_around3000 18181 non-null   float64 
17  parks_nearest   18181 non-null   float64 
18  ponds_around3000 18181 non-null   float64 
19  ponds_nearest   18181 non-null   float64 
20  days_exposition 20518 non-null   float64 

dtypes: bool(2), datetime64[ns](1), float64(11), int64(6), object(1)
memory usage: 3.5+ MB

```

In [59]: `data['first_day_exposition'].head()`

Out[59]: 0 2019-03-07
1 2018-12-04
2 2015-08-20
3 2015-07-24
4 2018-06-19
Name: first_day_exposition, dtype: datetime64[ns]

Блеск. Этой операцией мы упростили таблицу и будем тратить на меньше памяти. Также, с first_day_exposition будет проще работать.

Разберёмся с неявными дубликатами в столбце locality_name

Давайте изучим уникальные значения.

In [60]: `data['locality_name'].sort_values().unique()`

Out[60]: array(['Бокситогорск', 'Волосово', 'Волхов', 'Всеволожск', 'Выборг',
'Высоцк', 'Гатчина', 'Зеленогорск', 'Ивангород', 'Каменногорск',
'Кингисепп', 'Кириши', 'Кировск', 'Колпино', 'Коммунар',
'Красное Село', 'Кронштадт', 'Кудрово', 'Лодейное Поле',
'Ломоносов', 'Луга', 'Любань', 'Муриново', 'Никольское',
'Новая Ладога', 'Отрадное', 'Павловск', 'Петергоф', 'Пикалово',
'Подпорожье', 'Приморск', 'Приозерск', 'Пушкин', 'Санкт-Петербург',

'Светогорск', 'Сертолово', 'Сестрорецк', 'Сланцы', 'Сосновый Бор',
'Сясьстрой', 'Тихвин', 'Тосно', 'Шлиссельбург',
'городской поселок Большая Ижора', 'городской поселок Янино-1',
'городской посёлок Будогощь', 'городской посёлок Виллози',
'городской посёлок Лесогорский', 'городской посёлок Мга',
'городской посёлок Назия', 'городской посёлок Новоселье',
'городской посёлок Павлово', 'городской посёлок Рощино',
'городской посёлок Свиристрой', 'городской посёлок Советский',
'городской посёлок Фёдоровское', 'городской посёлок Янино-1',
'деревня Агалатово', 'деревня Аро', 'деревня Батово',
'деревня Бегуницы', 'деревня Белогорка', 'деревня Большая Вруда',
'деревня Большая Пустомержа', 'деревня Большие Колпаны',
'деревня Большое Рейзино', 'деревня Большой Сабск', 'деревня Бор',
'деревня Борисова Грива', 'деревня Ваганово', 'деревня Вартемяги',
'деревня Вахнова Кара', 'деревня Выскатка', 'деревня Гарболово',
'деревня Глинка', 'деревня Горбунки', 'деревня Гостилицы',
'деревня Заклинье', 'деревня Заневка', 'деревня Зимитицы',
'деревня Извара', 'деревня Иссад', 'деревня Калитино',
'деревня Кальтино', 'деревня Камышовка', 'деревня Каськово',
'деревня Келози', 'деревня Кипень', 'деревня Кисельня',
'деревня Колтуши', 'деревня Коркино', 'деревня Котлы',
'деревня Кривко', 'деревня Кудрово', 'деревня Кузьмолово',
'деревня Курковицы', 'деревня Куровицы', 'деревня Куттузи',
'деревня Лаврики', 'деревня Лаголово', 'деревня Лампово',
'деревня Лесколово', 'деревня Лопухинка', 'деревня Лупплово',
'деревня Малая Романовка', 'деревня Малое Верево',
'деревня Малое Карлино', 'деревня Малые Колпаны',
'деревня Мануйлово', 'деревня Меньково', 'деревня Мины',
'деревня Мистолово', 'деревня Ненимаки', 'деревня Нижние Осельки',
'деревня Нижняя', 'деревня Низино', 'деревня Новое Девяткино',
'деревня Новолисино', 'деревня Нурма', 'деревня Оржицы',
'деревня Парицы', 'деревня Пельгора', 'деревня Пеники',
'деревня Пижма', 'деревня Пикково', 'деревня Пудомяги',
'деревня Пустынка', 'деревня Пчева', 'деревня Рабитицы',
'деревня Разбегаево', 'деревня Раздолье', 'деревня Разметелево',
'деревня Рапполово', 'деревня Реброво', 'деревня Русско',
'деревня Сижно', 'деревня Снегирёвка', 'деревня Старая',
'деревня Старая Пустошь', 'деревня Старое Хинколово',
'деревня Старополье', 'деревня Старосиверская',
'деревня Старые Бегуницы', 'деревня Суоранда',
'деревня Сяськелево', 'деревня Тарасово', 'деревня Терпилицы',
'деревня Тихковицы', 'деревня Тойворово', 'деревня Торосово',
'деревня Торошковичи', 'деревня Трубников Бор',
'деревня Фалиеево', 'деревня Фёдоровское', 'деревня Хапо-Ое',
'деревня Хязельки', 'деревня Чудской Бор', 'деревня Шпаньково',
'деревня Щеглово', 'деревня Юкки', 'деревня Ялгино',
'деревня Яльгелево', 'деревня Ям-Тесово',
'коттеджный поселок Кивеннапа Север', 'коттеджный поселок Счастье',
'коттеджный посёлок Лесное', 'поселок Аннино', 'поселок Барышево',
'поселок Бугры', 'поселок Возрождение', 'поселок Войсковицы',
'поселок Володарское', 'поселок Гаврилово', 'поселок Гарболово',
'поселок Гладкое', 'поселок Глахово', 'поселок Глебычево',
'поселок Гончарово', 'поселок Громово', 'поселок Дружноселье',

'поселок Елизаветино', 'поселок Жилгородок', 'поселок Жилпосёлок',
'поселок Житково', 'поселок Заводской', 'поселок Запорожское',
'поселок Зимитицы', 'поселок Ильичёво', 'поселок Калитино',
'поселок Каложицы', 'поселок Кингисеппский', 'поселок Кирпичное',
'поселок Кобралово', 'поселок Кобринское', 'поселок Коммунары',
'поселок Коробицыно', 'поселок Котельский',
'поселок Красная Долина', 'поселок Красносельское',
'поселок Лесное', 'поселок Лисий Нос', 'поселок Лукаши',
'поселок Любань', 'поселок Мельниково', 'поселок Мичуринское',
'поселок Молодцово', 'поселок Мурино', 'поселок Новый Свет',
'поселок Новый Учхоз', 'поселок Оредеж',
'поселок Пансионат Зелёный Бор', 'поселок Первомайское',
'поселок Перово', 'поселок Петровское', 'поселок Победа',
'поселок Поляны', 'поселок Почап', 'поселок Починок',
'поселок Пущеное', 'поселок Пчевжа', 'поселок Рабитицы',
'поселок Романовка', 'поселок Ромашки', 'поселок Рябово',
'поселок Севастьяново', 'поселок Селезнёво', 'поселок Сельцо',
'поселок Семиозерье', 'поселок Семирено', 'поселок Серебрянский',
'поселок Совхозный', 'поселок Старая Малукса',
'поселок Стеклянный', 'поселок Сумино', 'поселок Суходолье',
'поселок Тельмана', 'поселок Терволово', 'поселок Торковичи',
'поселок Тёсово-4', 'поселок Углово', 'поселок Усть-Луга',
'поселок Ушаки', 'поселок Цвелодубово', 'поселок Цвылёво',
'поселок городского типа Большая Ижора',
'поселок городского типа Вырица',
'поселок городского типа Дружная Горка',
'поселок городского типа Дубровка',
'поселок городского типа Ефимовский',
'поселок городского типа Кондратьево',
'поселок городского типа Красный Бор',
'поселок городского типа Кузьмоловский',
'поселок городского типа Лебяжье',
'поселок городского типа Лесогорский',
'поселок городского типа Назия',
'поселок городского типа Никольский',
'поселок городского типа Приладожский',
'поселок городского типа Рахья', 'поселок городского типа Рошино',
'поселок городского типа Рябово',
'поселок городского типа Синявино',
'поселок городского типа Советский',
'поселок городского типа Токсово',
'поселок городского типа Форносово',
'поселок городского типа имени Свердлова',
'поселок станции Вещево', 'поселок станции Корнево',
'поселок станции Лужайка', 'поселок станции Приветнинское',
'посёлок Александровская', 'посёлок Алексеевка', 'посёлок Аннино',
'посёлок Белоостров', 'посёлок Бугры', 'посёлок Возрождение',
'посёлок Войсково', 'посёлок Высокоключевой',
'посёлок Гаврилово', 'посёлок Дзержинского', 'посёлок Жилгородок',
'посёлок Ильичёво', 'посёлок Кикерино', 'посёлок Кобралово',
'посёлок Коробицыно', 'посёлок Левашово', 'посёлок Ленинское',
'посёлок Лисий Нос', 'посёлок Мельниково', 'посёлок Металлострой',
'посёлок Мичуринское', 'посёлок Молодёжное', 'посёлок Мурино',

```
'посёлок Мыза-Ивановка', 'посёлок Новогорелово',
'посёлок Новый Свет', 'посёлок Пансионат Зелёный Бор',
'посёлок Парголово', 'посёлок Первово', 'посёлок Песочный',
'посёлок Петро-Славянка', 'посёлок Петровское',
'посёлок Платформа 69-й километр', 'посёлок Плодовое',
'посёлок Плоское', 'посёлок Победа', 'посёлок Поляны',
'посёлок Понтонный', 'посёлок Пригородный', 'посёлок Пудость',
'посёлок Репино', 'посёлок Ропша', 'посёлок Сапёрное',
'посёлок Сапёрный', 'посёлок Сосново', 'посёлок Старая Малукса',
'посёлок Стеклянный', 'посёлок Стрельна', 'посёлок Суида',
'посёлок Сумино', 'посёлок Тельмана', 'посёлок Терволово',
'посёлок Торфяное', 'посёлок Усть-Ижора', 'посёлок Усть-Луга',
'посёлок Форт Красная Горка', 'посёлок Шугозеро', 'посёлок Шушары',
'посёлок Щеглово', 'посёлок городского типа Важины',
'посёлок городского типа Вознесенье',
'посёлок городского типа Вырица',
'посёлок городского типа Красный Бор',
'посёлок городского типа Кузнечное',
'посёлок городского типа Кузьмоловский',
'посёлок городского типа Лебяжье', 'посёлок городского типа Мга',
'посёлок городского типа Павлово',
'посёлок городского типа Рошино', 'посёлок городского типа Рябово',
'посёлок городского типа Сиверский',
'посёлок городского типа Тайцы', 'посёлок городского типа Токсово',
'посёлок городского типа Ульяновка',
'посёлок городского типа Форносово',
'посёлок городского типа имени Морозова',
'посёлок городского типа имени Свердлова',
'посёлок при железнодорожной станции Вещево',
'посёлок при железнодорожной станции Приветнинское',
'посёлок станции Громово', 'посёлок станции Свирь',
'садоводческое некоммерческое товарищество Лесная Поляна',
'садовое товарищество Новая Ропша',
'садовое товарищество Приладожский', 'садовое товарищество Рахья',
'садовое товарищество Садко', 'село Копорье', 'село Никольское',
'село Павлово', 'село Паша', 'село Путилово', 'село Рождествено',
'село Русско-Высоцкое', 'село Старая Ладога', 'село Шум'],
dtype=object)
```

In [61]: `data['locality_name'].nunique()`

Out[61]: 364

Я думаю, сначала нужно заменить все вариации посёлков на "поселок".

In [62]: `def normalize_name(name):
 name = name.replace('посёлок', 'поселок')
 name = name.replace('поселок городского типа', 'поселок')
 name = name.replace('городской поселок', 'поселок')
 return name`

```
data['locality_name'] = data['locality_name'].apply(normalize_name)
```

```
In [63]: data['locality_name'].sort_values().unique()
```

```
Out[63]: array(['Бокситогорск', 'Волосово', 'Волхов', 'Всеволожск', 'Выборг',
 'Высоцк', 'Гатчина', 'Зеленогорск', 'Ивангород', 'Каменногорск',
 'Кингисепп', 'Кириши', 'Кировск', 'Колпино', 'Коммунар',
 'Красное Село', 'Кронштадт', 'Кудрово', 'Лодейное Поле',
 'Ломоносов', 'Луга', 'Любань', 'Мурино', 'Никольское',
 'Новая Ладога', 'Отрадное', 'Павловск', 'Петергоф', 'Пикалёво',
 'Подпорожье', 'Приморск', 'Приозерск', 'Пушкин', 'Санкт-Петербург',
 'Светогорск', 'Сертолово', 'Сестрорецк', 'Сланцы', 'Сосновый Бор',
 'Сясьстрой', 'Тихвин', 'Тосно', 'Шлиссельбург',
 'деревня Агалатово', 'деревня Аро', 'деревня Батово',
 'деревня Бегуницы', 'деревня Белогорка', 'деревня Большая Вруда',
 'деревня Большая Пустомержа', 'деревня Большие Колпаны',
 'деревня Большое Рейзино', 'деревня Большой Сабск', 'деревня Бор',
 'деревня Борисова Грива', 'деревня Ваганово', 'деревня Вартемяги',
 'деревня Вахнова Кара', 'деревня Выскатка', 'деревня Гарболово',
 'деревня Глинка', 'деревня Горбунки', 'деревня Гостилицы',
 'деревня Заклинье', 'деревня Заневка', 'деревня Зимитицы',
 'деревня Извара', 'деревня Иссад', 'деревня Калитино',
 'деревня Калътино', 'деревня Камышовка', 'деревня Каськово',
 'деревня Келози', 'деревня Кипень', 'деревня Кисельня',
 'деревня Колтуши', 'деревня Коркино', 'деревня Котлы',
 'деревня Кривко', 'деревня Кудрово', 'деревня Кузьмолово',
 'деревня Курковицы', 'деревня Куровицы', 'деревня Куттузи',
 'деревня Лаврики', 'деревня Лаголово', 'деревня Лампово',
 'деревня Лесколово', 'деревня Лопухинка', 'деревня Лупплолово',
 'деревня Малая Романовка', 'деревня Малое Верево',
 'деревня Малое Карлино', 'деревня Малые Колпаны',
 'деревня Мануйлово', 'деревня Меньково', 'деревня Мины',
 'деревня Мистолово', 'деревня Ненимяки', 'деревня Нижние Осельки',
 'деревня Нижняя', 'деревня Низино', 'деревня Новое Девяткино',
 'деревня Новолисино', 'деревня Нурма', 'деревня Оржицы',
 'деревня Парицы', 'деревня Пельгора', 'деревня Пеники',
 'деревня Пижма', 'деревня Пикколо', 'деревня Пудомяги',
 'деревня Пустынка', 'деревня Пчева', 'деревня Рабитицы',
 'деревня Разбегаево', 'деревня Раздолье', 'деревня Разметелево',
 'деревня Рапполово', 'деревня Реброво', 'деревня Русско',
 'деревня Сижно', 'деревня Снегирёвка', 'деревня Старая',
 'деревня Старая Пустошь', 'деревня Старое Хинколово',
 'деревня Старополье', 'деревня Старосиверская',
 'деревня Старые Бегуницы', 'деревня Суоранда',
 'деревня Сяськелево', 'деревня Тарасово', 'деревня Терпилицы',
 'деревня Тихковицы', 'деревня Тойворово', 'деревня Торосово',
 'деревня Торошковичи', 'деревня Трубников Бор',
 'деревня Фалилеево', 'деревня Фёдоровское', 'деревня Хапо-Ое',
 'деревня Хязельки', 'деревня Чудской Бор', 'деревня Шпаньково',
 'деревня Щеглово', 'деревня Юкки', 'деревня Ялгино',
```

'деревня Яльгелево', 'деревня Ям-Тесово',
'коттеджный поселок Кивеннапа Север', 'коттеджный поселок Лесное',
'коттеджный поселок Счастье', 'поселок Александровская',
'поселок Алексеевка', 'поселок Аннино', 'поселок Барышево',
'поселок Белоостров', 'поселок Большая Ижора', 'поселок Бугры',
'поселок Будогощь', 'поселок Важины', 'поселок Виллози',
'поселок Вознесенье', 'поселок Возрождение', 'поселок Войсковицы',
'поселок Войсково', 'поселок Володарское', 'поселок Вырица',
'поселок Высокоключевой', 'поселок Гаврилово', 'поселок Гарболово',
'поселок Гладкое', 'поселок Глахово', 'поселок Глебычево',
'поселок Гончарово', 'поселок Громово', 'поселок Дзержинского',
'поселок Дружная Горка', 'поселок Дружноселье', 'поселок Дубровка',
'поселок Елизаветино', 'поселок Ефимовский', 'поселок Жилгородок',
'поселок Жилпоселок', 'поселок Житково', 'поселок Заводской',
'поселок Запорожское', 'поселок Зимитицы', 'поселок Ильичёво',
'поселок Калитино', 'поселок Каложицы', 'поселок Кикерино',
'поселок Кингисеппский', 'поселок Кирпичное', 'поселок Кобралово',
'поселок Кобринское', 'поселок Коммунары', 'поселок Кондратьево',
'поселок Коробицыно', 'поселок Котельский',
'поселок Красная Долина', 'поселок Красносельское',
'поселок Красный Бор', 'поселок Кузнечное',
'поселок Кузьмоловский', 'поселок Лебяжье', 'поселок Левашово',
'поселок Ленинское', 'поселок Лесное', 'поселок Лесогорский',
'поселок Лисий Нос', 'поселок Лукаши', 'поселок Любань',
'поселок Мга', 'поселок Мельниково', 'поселок Металлострой',
'поселок Мичуринское', 'поселок Молодцово', 'поселок Молодёжное',
'поселок Мурино', 'поселок Мыза-Ивановка', 'поселок Назия',
'поселок Никольский', 'поселок Новогорелово', 'поселок Новоселье',
'поселок Новый Свет', 'поселок Новый Учхоз', 'поселок Оредеж',
'поселок Павлово', 'поселок Пансионат Зелёный Бор',
'поселок Парголово', 'поселок Первомайское', 'поселок Перово',
'поселок Песочный', 'поселок Петро-Славянка', 'поселок Петровское',
'поселок Платформа 69-й километр', 'поселок Плодовое',
'поселок Плоское', 'поселок Победа', 'поселок Поляны',
'поселок Понтонный', 'поселок Почап', 'поселок Починок',
'поселок Пригородный', 'поселок Приладожский', 'поселок Пудость',
'поселок Пушное', 'поселок Пчевжа', 'поселок Рабитицы',
'поселок Рахья', 'поселок Репино', 'поселок Романовка',
'поселок Ромашки', 'поселок Ропша', 'поселок Рошино',
'поселок Рябово', 'поселок Сапёрное', 'поселок Сапёрный',
'поселок Свирьстрой', 'поселок Севастьяново', 'поселок Селезнёво',
'поселок Сельцо', 'поселок Семиозерье', 'поселок Семирно',
'поселок Серебрянский', 'поселок Сиверский', 'поселок Синявино',
'поселок Советский', 'поселок Совхозный', 'поселок Сосново',
'поселок Старая Малукса', 'поселок Стеклянный', 'поселок Стрельна',
'поселок Судя', 'поселок Сумино', 'поселок Суходолье',
'поселок Тайцы', 'поселок Тельмана', 'поселок Терволово',
'поселок Токово', 'поселок Торковичи', 'поселок Торфяное',
'поселок Тёсово-4', 'поселок Углово', 'поселок Ульяновка',
'поселок Усть-Ижора', 'поселок Усть-Луга', 'поселок Ушаки',
'поселок Форносово', 'поселок Форт Красная Горка',
'поселок Фёдоровское', 'поселок Цвелодубово', 'поселок Цвылёво',
'поселок Шугозеро', 'поселок Шушары', 'поселок Щеглово',

```
'поселок Янино-1', 'поселок имени Морозова',
'поселок имени Свердлова',
'поселок при железнодорожной станции Вещево',
'поселок при железнодорожной станции Приветнинское',
'поселок станции Вещево', 'поселок станции Громово',
'поселок станции Корнево', 'поселок станции Лужайка',
'поселок станции Приветнинское', 'поселок станции Свирь',
'садоводческое некоммерческое товарищество Лесная Поляна',
'садовое товарищество Новая Ропша',
'садовое товарищество Приладожский', 'садовое товарищество Рахья',
'садовое товарищество Садко', 'село Копорье', 'село Никольское',
'село Павлово', 'село Паша', 'село Путилово', 'село Рождествено',
'село Русско-Высоцкое', 'село Старая Ладога', 'село Шум'],
dtype=object)
```

In [64]: `data['locality_name'].nunique()`

Out[64]: 322

Супер. Давайте ещё попробуем удалить дубликаты вида "Мурино" и "поселок Мурино".

```
def drop_names(series):
    names = series.tolist()
    normalized = []

    for name in names:
        # Разбиваем название на слова
        parts = name.split()

        # Ищем самое короткое название, которое является частью текущего
        shortest_match = None
        for other in names:
            if other == name:
                continue
            # Если текущее название содержит другое (например, "поселок Мурино" содержит "Мурино")
            if other in name:
                if shortest_match is None or len(other) < len(shortest_match):
                    shortest_match = other
            # Или если другое название содержит текущее (например, "Мурино" содержится в "поселок Мурино")
            elif name in other:
                if shortest_match is None or len(name) < len(shortest_match):
                    shortest_match = name

        # Если нашли более короткий вариант, берём его, иначе оставляем исходный
        normalized.append(shortest_match if shortest_match else name)

    return pd.Series(normalized, index=series.index)
```

```
# Применяем функцию
data['locality_name'] = drop_names(data['locality_name'])
```

Для написания этой функции я пользовался дипсиком. Думаю, это было удачное место для его помощи. Посмотрим на итог.

In [66]: `data['locality_name'].nunique()`

Out[66]: 311

ВЕ-ЛИ-КО-ЛЕП-НО!

С предобработкой закончили, едем дальше.

Посчитайте и добавьте в таблицу новые столбцы

Нам нужно добавить следующие столбцы:

- цена одного квадратного метра (нужно поделить стоимость объекта на его общую площадь, а затем округлить до двух знаков после запятой);
- день недели публикации объявления (0 — понедельник, 1 — вторник и так далее);
- месяц публикации объявления;
- год публикации объявления;
- тип этажа квартиры (значения — «первый», «последний», «другой»);
- расстояние до центра города в километрах (переведите из м в км и округлите до ближайших целых значений).

Начнём с цены за квадратный метр.

In [67]: `data['square_cost'] = data['last_price'] / data['total_area']
data['square_cost'] = data['square_cost'].round(2)
data['square_cost'].head()`

Out[67]: 0 120370.37
1 82920.79
2 92785.71
3 408176.10
4 100000.00
Name: square_cost, dtype: float64

Как два пальца... В общем, раз плюнуть.

In [68]: `data['weekday'] = data['first_day_exposition'].dt.weekday
data['weekday'].head()`

Out[68]: 0 3
1 1

```
2     3  
3     4  
4     1  
Name: weekday, dtype: int64
```

Добавили столбец. Но сейчас с нём порядковые числа дней недели (0,1,2...). Исправим на русскоязычные названия.

```
In [69]:  
weekday_map = {  
    0: 'понедельник',  
    1: 'вторник',  
    2: 'среда',  
    3: 'четверг',  
    4: 'пятница',  
    5: 'суббота',  
    6: 'воскресенье'  
}  
  
data['weekday'] = data['weekday'].map(weekday_map)  
  
data['weekday'].head()
```

```
Out[69]: 0    четверг  
1    вторник  
2    четверг  
3    пятница  
4    вторник  
Name: weekday, dtype: object
```

```
In [70]: data['month'] = data['first_day_exposition'].dt.month
```

```
In [71]: data['month'].head()
```

```
Out[71]: 0    3  
1    12  
2    8  
3    7  
4    6  
Name: month, dtype: int64
```

Меняем числа на названия.

```
In [72]: month_map = {  
    1: 'январь',  
    2: 'февраль',
```

```
3: 'март',
4: 'апрель',
5: 'май',
6: 'июнь',
7: 'июль',
8: 'август',
9: 'сентябрь',
10: 'октябрь',
11: 'ноябрь',
12: 'декабрь'
}

data['month'] = data['month'].map(month_map)

data['month'].head()
```

```
Out[72]: 0      март
1      декабрь
2      август
3      июль
4      июнь
Name: month, dtype: object
```

Супер. Теперь год.

```
In [73]: data['year'] = data['first_day_exposition'].dt.year
```

```
In [74]: data['year'].head()
```

```
Out[74]: 0    2019
1    2018
2    2015
3    2015
4    2018
Name: year, dtype: int64
```

А теперь сделаем столбец с типом этажа - «первый», «последний», «другой».

```
In [75]: data['floor_type'] = 'другой'
```

```
In [76]: data['floor_type'].head()
```

```
Out[76]: 0    другой
1    другой
2    другой
3    другой
```

```
4    другой
Name: floor_type, dtype: object
```

Не забываем про строки, где floors_total было пропущено и мы поставили -1. Присвоим floor_type значение 'неизвестно'.

```
In [77]: data.loc[data['floors_total'] == -1, 'floor_type'] = 'неизвестно'
```

Мы сделали столбец, в котором у всех строк значение типа этажа - другой. Теперь исправим это значение для строк, где этаж является первым или последним.

```
In [78]: data.loc[data['floor'] == 1, 'floor_type'] = 'первый'
data.loc[data['floor'] == data['floors_total'], 'floor_type'] = 'последний'
```

```
In [79]: data[['floor', 'floors_total', 'floor_type']].head(15)
```

Out[79]:

	floor	floors_total	floor_type
0	8	16	другой
1	1	11	первый
2	4	5	другой
3	9	14	другой
4	13	14	другой
5	5	12	другой
6	6	26	другой
7	22	24	другой
8	26	27	другой
9	7	9	другой
10	3	12	другой
11	4	5	другой
12	5	5	последний
13	5	5	последний
14	1	6	первый

```
In [80]: data['floor_type'].value_counts()
```

```
Out[80]: другой      17363  
последний    3361  
первый        2892  
неизвестно     83  
Name: floor_type, dtype: int64
```

Классно. Теперь округлённое расстояние до центра города в километрах.

Давайте, кстати, поменяем название столбца cityCenters_nearest на более адекватное. Лучше поздно, чем никогда.

```
In [81]: data = data.rename(columns={'cityCenters_nearest': 'centers_range'})
```

```
In [82]: data['centers_range_km'] = (data['centers_range']/1000).round(0)
```

```
In [83]: data[['centers_range', 'centers_range_km']].head()
```

```
Out[83]: centers_range  centers_range_km  
0      16028.0          16.0  
1      18603.0          19.0  
2      13933.0          14.0  
3       6800.0           7.0  
4      8098.0           8.0
```

Ай да я!

Проведите исследовательский анализ данных

Изучим перечисленные ниже параметры объектов и построим отдельные гистограммы для каждого из этих них. В некоторых параметрах встречаются редкие и выбивающиеся значения, обработаем их.

- общая площадь;
- жилая площадь;
- площадь кухни;
- цена объекта;
- количество комнат;
- высота потолков;
- тип этажа квартиры («первый», «последний», «другой»);

- общее количество этажей в доме;
- расстояние до центра города в метрах;
- расстояние до ближайшего парка

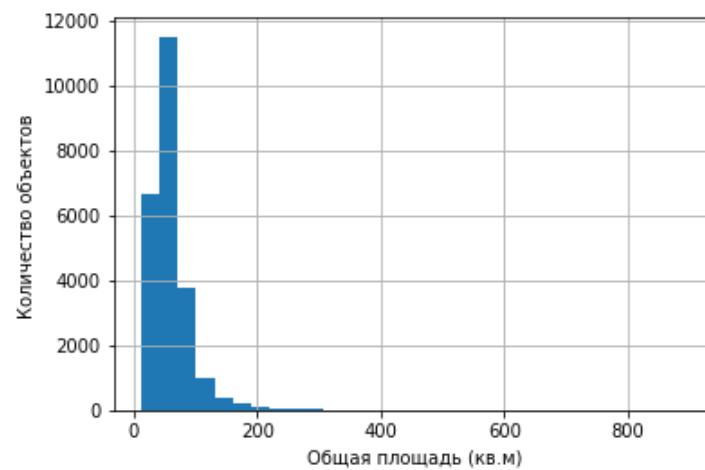
Общая площадь

Начнём с общей площади. Сначала посмотрим описание выборки, созданное методом describe, а затем построим гистограмму с названиями осей.

```
In [84]: data['total_area'].describe()
```

```
Out[84]: count    23699.000000
mean      60.348651
std       35.654083
min      12.000000
25%     40.000000
50%     52.000000
75%     69.900000
max     900.000000
Name: total_area, dtype: float64
```

```
In [85]: data['total_area'].hist(bins=30)
plt.xlabel('Общая площадь (кв.м)')
plt.ylabel('Количество объектов')
plt.show()
```



Здесь я названия расписал. Дальше не буду, т.к. и так всё понятно и, эта работа в первую очередь для меня. Не хочу тратить время.

Распределение похоже на нормальное. Медиана и среднее заметно отличаются. Есть выбросы в виде квартир с очень большой площадью. Имеет смысл избавиться от них. Но давайте сначала задумимся и проверим, нет ли у экстремальных значений оснований, чтобы мы посчитали, что кто-то поставил лишний

0.

```
In [86]: data.loc[data['total_area'] > 400, ['total_area', 'last_price', 'square_cost']]
```

```
Out[86]:
```

	total_area	last_price	square_cost
660	483.90	49950000	103223.81
3117	631.00	140000000	221870.05
3676	441.98	28789000	65136.43
4237	517.00	50000000	96711.80
5358	590.00	65000000	110169.49
5893	500.00	230000000	460000.00
6221	470.30	34000000	72294.28
6621	488.00	99000000	202868.85
8018	507.00	84000000	165680.47
9826	494.10	43185328	87402.00
12401	495.00	91500000	184848.48
12859	631.20	140000000	221799.75
13749	410.00	240000000	585365.85
14088	402.00	51000000	126865.67
14706	401.00	401300000	1000748.13
14991	413.50	45000000	108827.09
15016	500.00	150000000	300000.00
15651	618.00	300000000	485436.89
19540	900.00	420000000	466666.67
20273	460.80	20000000	43402.78
21955	431.00	130000000	301624.13
22131	422.20	27000000	63950.73
22494	491.00	91075000	185488.80

Проверим варианты с самой низкой ценой за кв.м.

In [87]: `data.loc[20273]`

```
Out[87]: total_images           12
last_price            20000000
total_area            460.8
first_day_exposition 2019-03-20 00:00:00
rooms                  6
ceiling_height         3.1
floors_total           3
living_area             279.6
floor                  1
studio                 False
open_plan               False
kitchen_area            55.5
balcony                 0
locality_name          Санкт-Петербург
airports_nearest        43756.0
centers_range           15459.0
parks_around3000         1.0
parks_nearest            852.0
ponds_around3000          3.0
ponds_nearest             122.0
days_exposition          21.0
square_cost              43402.78
weekday                среда
month                  март
year                   2019
floor_type              первый
centers_range_km         15.0
Name: 20273, dtype: object
```

Наверное частный дом. 3 этажа, большая площадь кухни, жилая площадь меньше 2/3. Всё в порядке.

In [88]: `data.loc[22131]`

```
Out[88]: total_images           14
last_price            27000000
total_area            422.2
first_day_exposition 2018-10-18 00:00:00
rooms                  5
ceiling_height         2.7
floors_total           13
living_area             NaN
floor                  11
studio                 False
open_plan               False
kitchen_area            NaN
balcony                 0
locality_name          Санкт-Петербург
```

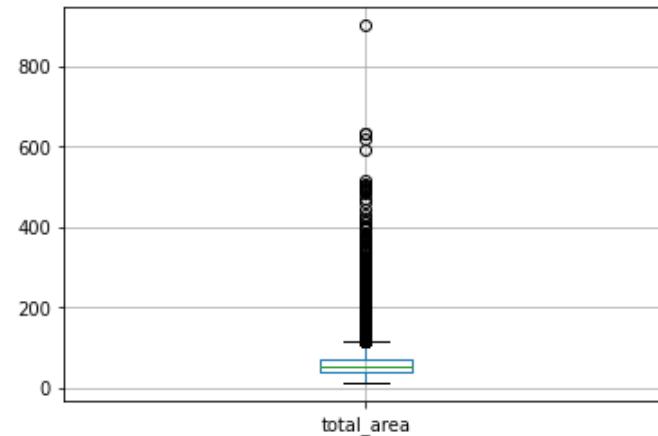
```
airports_nearest           46625.0
centers_range              16554.0
parks_around3000            0.0
parks_nearest                -1.0
ponds_around3000              0.0
ponds_nearest                -1.0
days_exposition                  NaN
square_cost                   63950.73
weekday                        четверг
month                           октябрь
year                            2018
floor_type                      другой
centers_range_km                 17.0
Name: 22131, dtype: object
```

Это вообще, видимо, не жилое помещение. Данных о жилой площади и площади кухни нет, комнат 5 (вряд ли они уместились на 42 метрах).

Значит исправлять нечего. Только удалять. Давайте определимся, где делать отрез. Посмотрим на ящик с усами.

```
In [89]: data.boxplot(column='total_area')
```

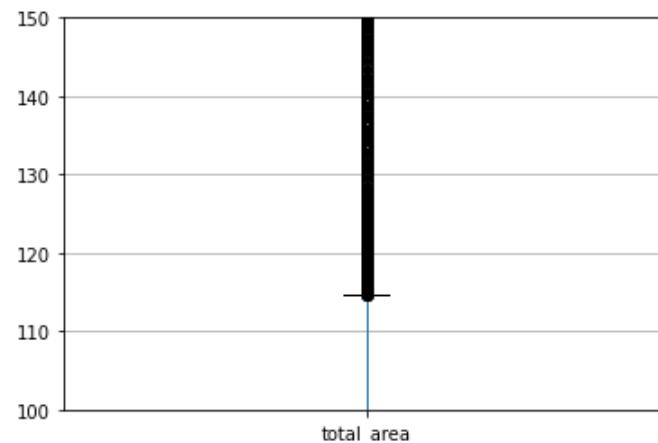
```
Out[89]: <AxesSubplot:>
```



Начиная с 400 кв.м. значений явно мало. Давайте посмотрим, где верхний ус.

```
In [90]: data.boxplot(column='total_area')
plt.ylim(100, 150)
```

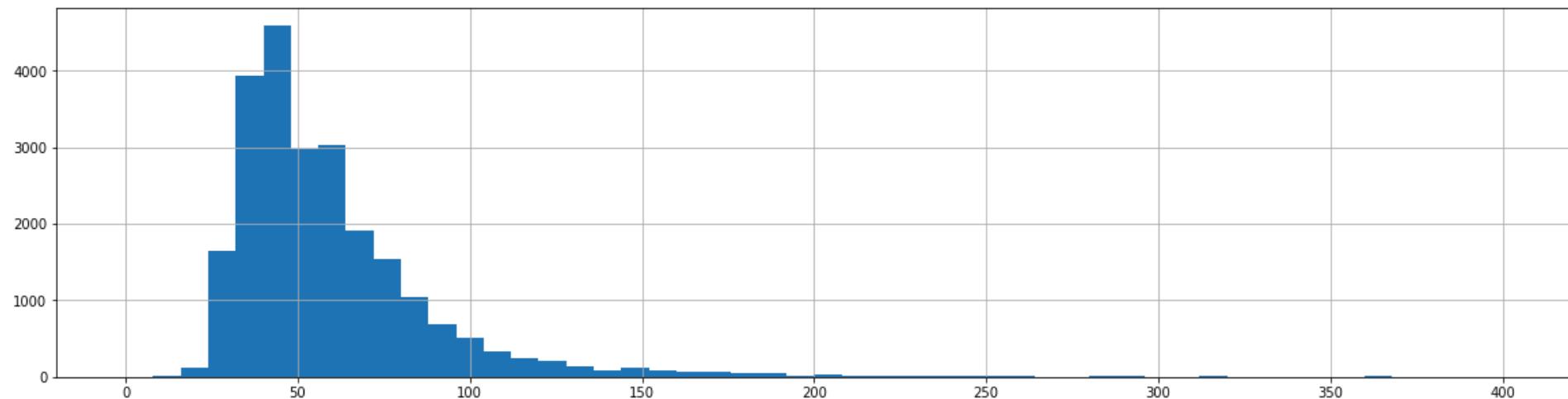
```
Out[90]: (100.0, 150.0)
```



Около 115. Но после этого значения ещё много выбросов. Посмотрим на гистограмму с распределением поближе.

```
In [91]: data['total_area'].hist(bins=50, range=(0,400), figsize=(20,5))
```

```
Out[91]: <AxesSubplot:>
```



После 150 значений мало, после 200 вообще почти нет. Давайте сделаем отсечку по 200. Посмотрим describe с этой границей.

```
In [92]: data[data['total_area'] < 200]['total_area'].describe()
```

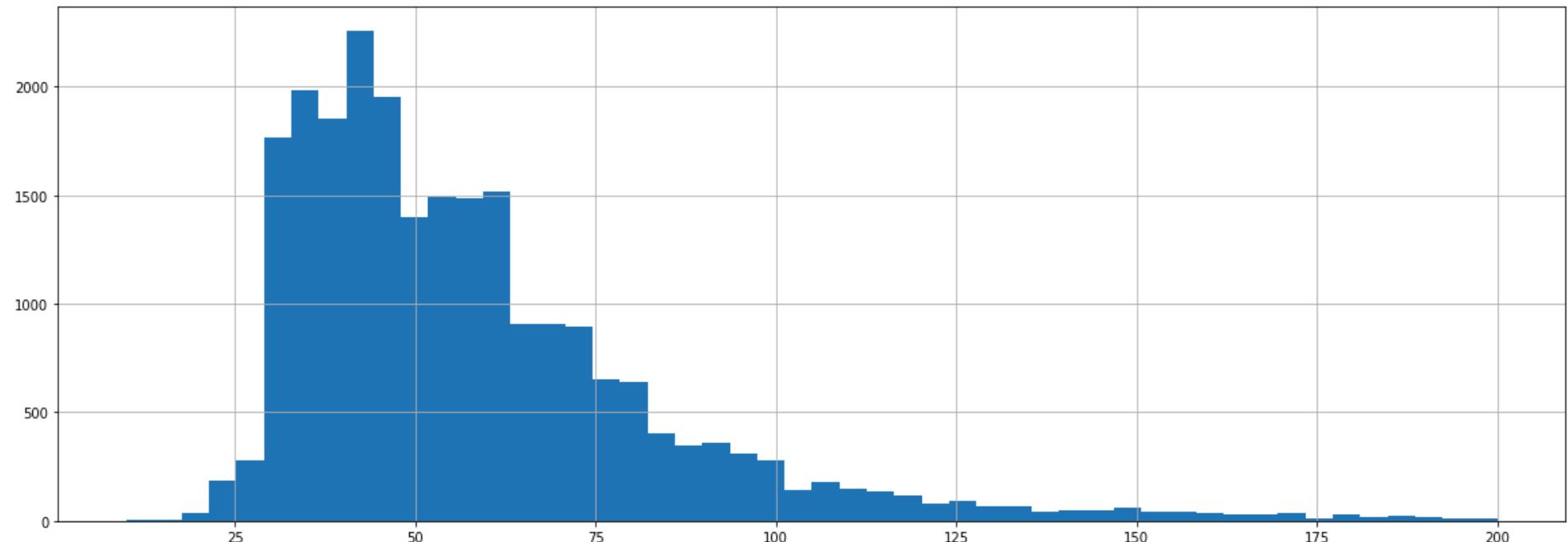
```
Out[92]: count    23468.000000
mean      58.155629
std       26.469479
```

```
min      12.000000
25%     40.000000
50%     51.900000
75%     68.900000
max    199.200000
Name: total_area, dtype: float64
```

Так получше. Среднее с медианой немного сблизились. Станд.отклон упал с ~35,5 до ~26,5.

```
In [93]: data['total_area'].hist(bins=50, range=(10,200), figsize=(20,7))
```

```
Out[93]: <AxesSubplot:>
```



Красота.

Итого. Анализ результата

Жилая площадь

Снова начнём с описания и гистограммы.

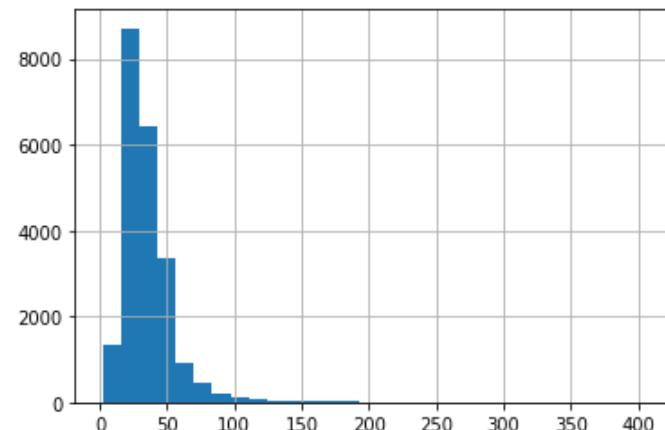
```
In [94]: data['living_area'].describe()
```

```
Out[94]: count    21796.000000
mean      34.457852
```

```
std      22.030445
min      2.000000
25%     18.600000
50%     30.000000
75%     42.300000
max    409.700000
Name: living_area, dtype: float64
```

In [95]: `data['living_area'].hist(bins=30)`

Out[95]: <AxesSubplot:>



Значения меньше, а общая картина идентична. Закономерно. Меня только смущает минимум. 2 кв.м? Это прихожая в студии? Я хочу изучить.

In [96]: `data[data['living_area'] < 10]['living_area'].count()`

Out[96]: 25

In [97]: `data[data['living_area'] < 10]`

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	parks_nearest	ponds_around3000	ponds_nea
114	5	4000000	21.40	2019-02-07	1	2.90	8	8.3	7	False	...	488.0	1.0	28
680	14	7200000	43.00	2017-10-31	1	NaN	8	9.0	2	False	...	2137.0	1.0	9:
1326	8	8100000	52.00	2017-01-29	2	2.70	25	9.0	15	False	...	-1.0	1.0	118
2309	10	4200000	62.40	2017-06-15	2	2.60	9	8.4	6	False	...	-1.0	0.0	
3242	7	4440000	41.00	2017-07-02	1	NaN	17	3.0	17	False	...	-1.0	0.0	

Project_YP3

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	parks_nearest	ponds_around3000	ponds_nea
4100	17	5300000	34.80	2017-06-19	1	2.70	23	9.0	15	False	...	-1.0	0.0	
4542	12	3300000	18.00	2018-11-07	1	3.50	5	9.0	2	False	...	349.0	1.0	90
7312	8	3400000	27.00	2018-02-21	2	2.50	15	9.0	4	False	...	617.0	0.0	
8325	9	4800000	52.00	2017-10-25	2	NaN	5	9.0	2	False	...	-1.0	2.0	1
13915	20	6350000	52.00	2018-02-06	2	3.00	6	2.0	2	False	...	-1.0	1.0	10
15833	20	4600000	33.00	2017-01-01	1	2.70	22	9.0	12	False	...	-1.0	0.0	
16431	13	3799000	31.00	2018-04-12	1	2.60	5	6.0	1	False	...	430.0	0.0	
17248	20	5300000	33.00	2017-04-14	1	2.70	22	8.9	16	False	...	-1.0	0.0	
17582	11	2680000	22.00	2018-08-11	0	NaN	25	5.0	8	True	...	835.0	1.0	6!
19251	19	4050000	33.00	2018-10-15	1	2.50	22	6.5	21	False	...	458.0	0.0	
19620	10	4300000	33.00	2018-02-01	1	NaN	5	9.0	1	False	...	251.0	3.0	2
20994	7	8900000	50.60	2018-10-22	2	2.50	7	8.0	3	False	...	173.0	3.0	17
21505	9	4100000	35.30	2018-01-10	1	2.75	27	8.5	11	False	...	-1.0	0.0	
21758	0	2330000	23.00	2018-01-01	0	NaN	24	2.0	22	True	...	NaN	NaN	1
21908	9	5300000	46.30	2018-03-20	1	2.75	7	9.8	3	False	...	-1.0	1.0	11
21943	15	6100000	77.60	2019-02-28	4	2.50	9	5.4	1	False	...	584.0	0.0	
22252	4	3340000	37.40	2018-02-08	1	2.80	8	9.0	4	False	...	365.0	0.0	
22473	0	3490304	33.26	2015-12-22	2	NaN	13	9.1	9	False	...	-1.0	0.0	
23208	12	4800000	37.00	2016-02-24	1	NaN	14	8.0	11	False	...	358.0	1.0	9!
23574	14	64990000	139.00	2015-11-24	3	3.00	8	3.0	8	False	...	630.0	3.0	10

25 rows × 27 columns

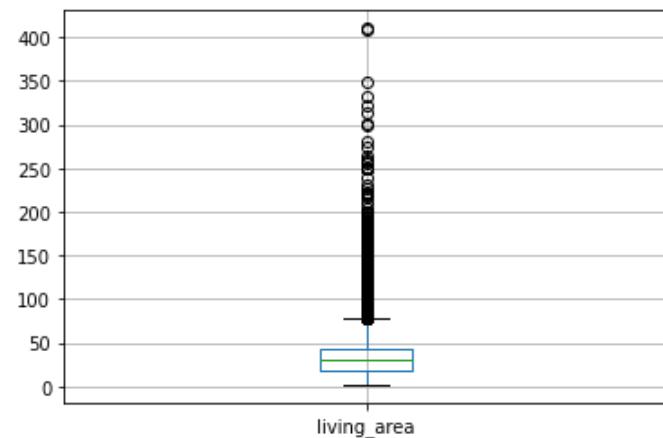
Какая-то шняга. Давайте не будем это учитывать.

Что там с усатиком?

In [98]:

data.boxplot(column='living_area')

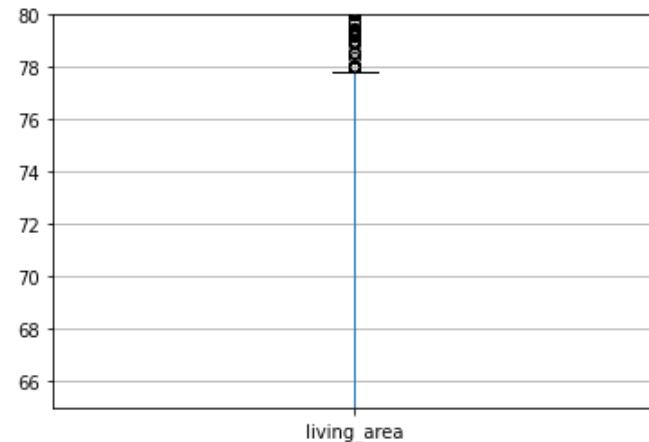
Out[98]: <AxesSubplot:>



После 200 выбросов сильно меньше. Построим гистограмму с этим ограничением, но сначала найдём верхний ус.

```
In [99]:  
data.boxplot(column='living_area')  
plt.ylim(65, 80)
```

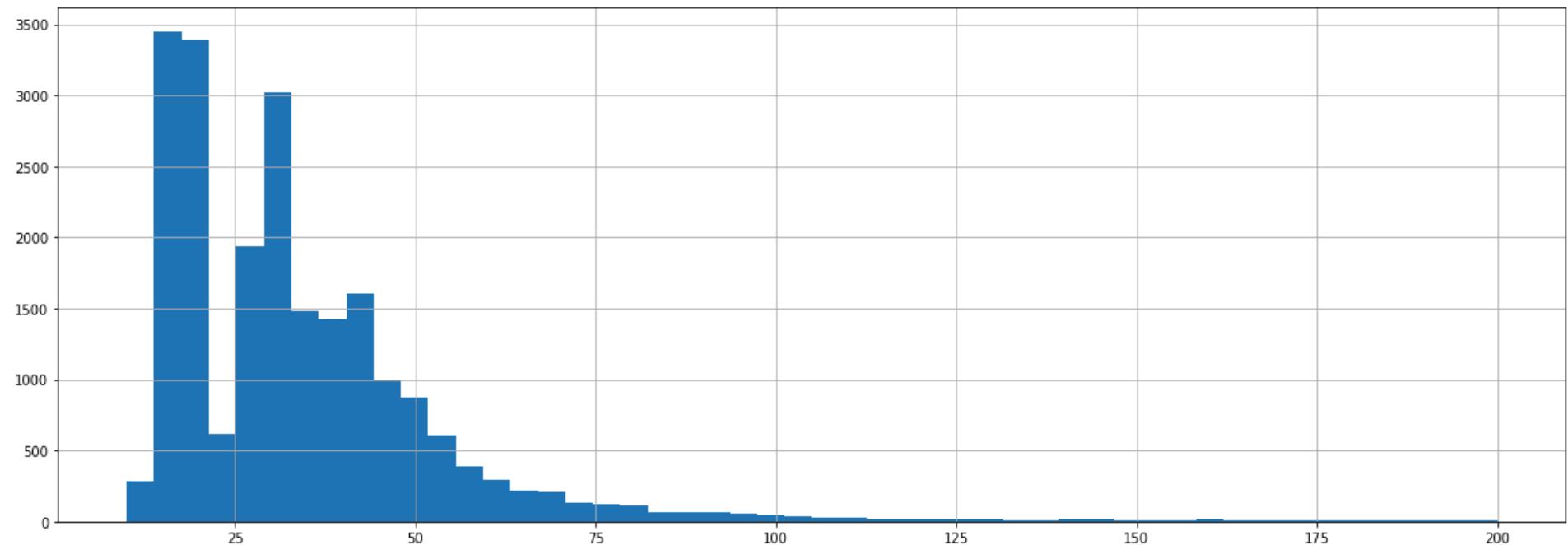
```
Out[99]: (65.0, 80.0)
```



Почти 78. Гистограмма?

```
In [100...]  
data['living_area'].hist(bins=50, range=(10,200), figsize=(20,7))
```

```
Out[100...]: <AxesSubplot:>
```

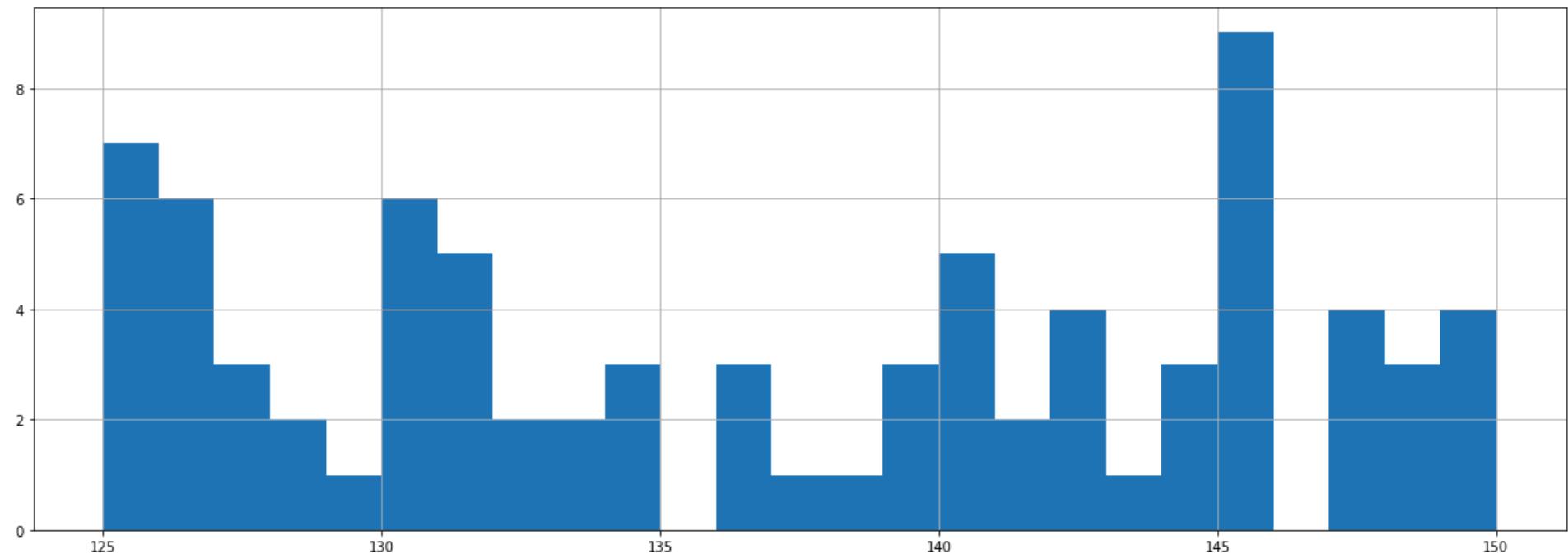


In [101...]

```
data['living_area'].hist(bins=25, range=(125,150), figsize=(20,7))
```

Out[101...]

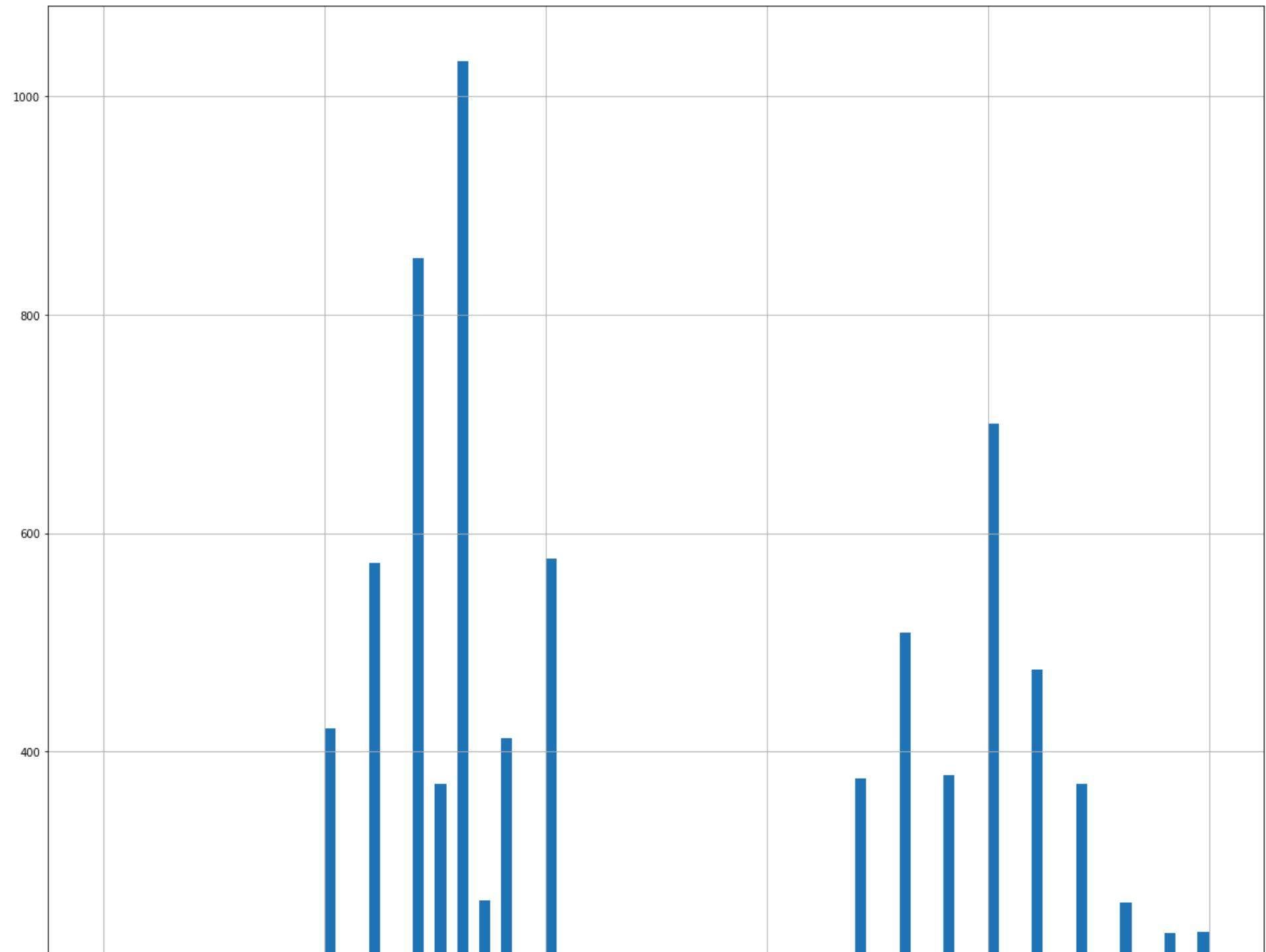
```
<AxesSubplot:>
```

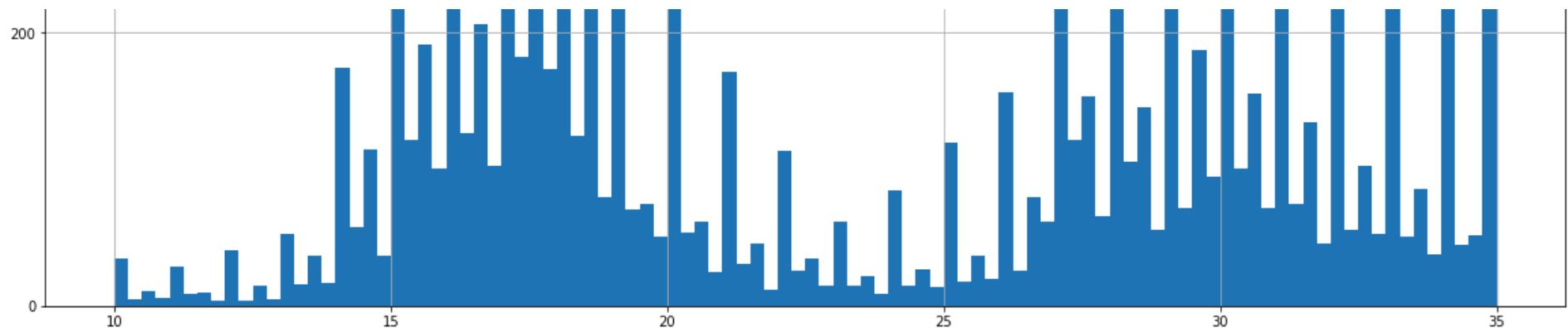


Я бы забил на это и остановился на 125.

In [102...]:
data['living_area'].hist(bins=100, range=(10,35), figsize=(20,20))

Out[102...]: <AxesSubplot:>





Я не уверен, как называется такое распределение. Нет, в целом распределение на общем графике выглядит как логнормальное с аномалиями, и к этим выбросами стоит отнестись спокойно. Наверняка это типичные показатели для однушек и двухшек. На общем графике видно и третий подъем, просто менее выраженный. Уверен, это типичная площадь для трешки. Комнат больше, следовательно больше гладкости в распределении, а пики короче. Не вижу смысла проверять детально. Чекнем медианы жилой площасти по количеству комнат.

In [103...]:

```
data.groupby('rooms')['living_area'].median()
```

Out[103...]:

rooms	living_area
0	18.000
1	17.600
2	30.500
3	45.000
4	63.600
5	91.785
6	123.900
7	145.000
8	161.000
9	179.200
10	161.400
11	133.900
12	409.700
14	195.150
15	409.000
16	180.000
19	264.500

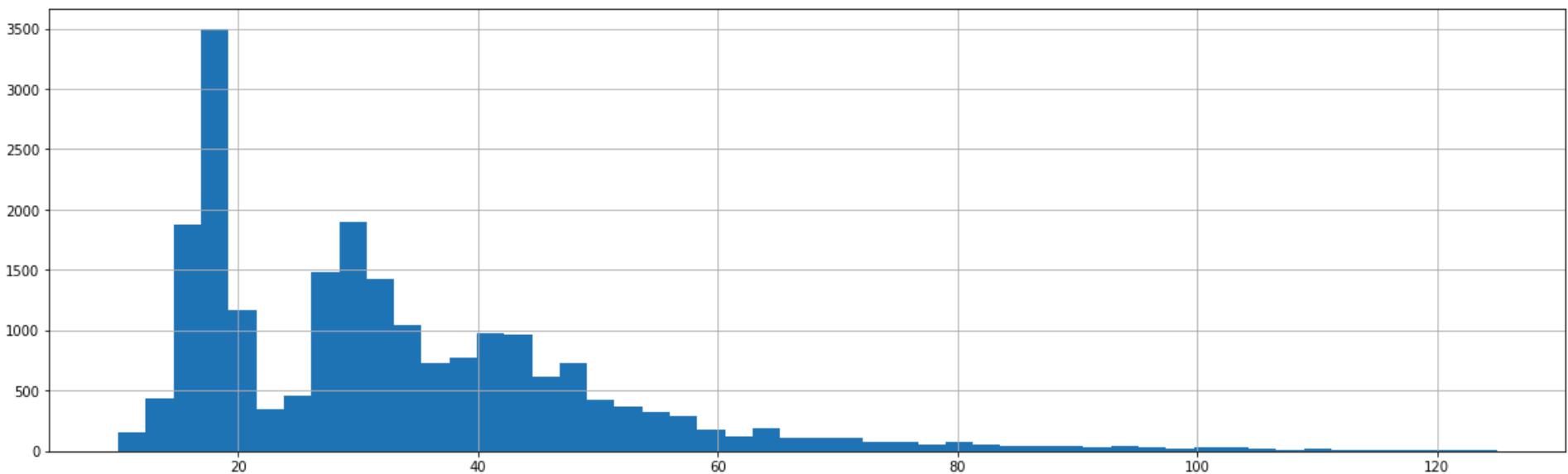
Name: living_area, dtype: float64

Сделаем срез до 125 и опишем результат.

In [104...]:

```
data['living_area'].hist(bins=50, range=(10,125), figsize=(20,6))
```

Out[104...]: <AxesSubplot:>



Не, немного кринжово выглядит, я согласен. Но моё объяснение всё объясняет.

Было:

```
In [105...]: data['living_area'].describe()
```

```
Out[105...]: count    21796.000000
mean        34.457852
std         22.030445
min         2.000000
25%        18.600000
50%        30.000000
75%        42.300000
max        409.700000
Name: living_area, dtype: float64
```

Стало:

```
In [106...]: data[(data['living_area'] >= 10) & (data['living_area'] < 125)]['living_area'].describe()
```

```
Out[106...]: count    21572.000000
mean        33.220543
std         17.057882
min         10.000000
25%        18.600000
50%        30.000000
75%        42.000000
```

```
max      124.900000
Name: living_area, dtype: float64
```

Покрасивше. До чего же устойчивые квартили. Это потому что данные тяготеют к типичным площадям 1, 2 и 3-х комнатных квартир. Больше всего снизился вновь станд.отклон.

Итого.

Площадь кухни

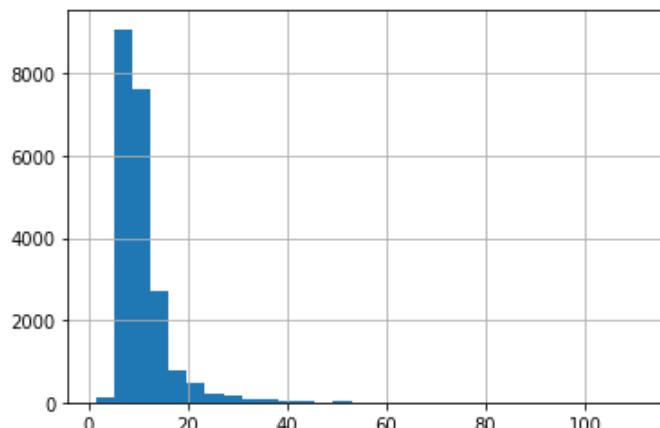
```
In [107...]: data['kitchen_area'].describe()
```

```
Out[107...]: count    21421.000000
mean      10.569807
std       5.905438
min      1.300000
25%      7.000000
50%      9.100000
75%     12.000000
max     112.000000
Name: kitchen_area, dtype: float64
```

Выглядит лучше, но это из-за того, что числа маленькие. Разница между средним и медианой больше, чем 1/10 от значений. Станд.отклон. больше, чем половина среднего. Так себе, в общем. Большой разброс.

```
In [108...]: data['kitchen_area'].hist(bins=30)
```

```
Out[108...]: <AxesSubplot:>
```



Кухня 1.3 метра - это конечно лучше, чем жилая зона 2 метра, но всё же плохо. Давайте посмотрим на квартиры с кухнями меньше 4 метров.

In [109...]

```
data[data['kitchen_area'] < 4][['last_price', 'locality_name', 'centers_range', 'total_area', 'living_area', 'kitchen_area']]
```

Out[109...]

	last_price	locality_name	centers_range	total_area	living_area	kitchen_area
906	2600000	Санкт-Петербург	17459.0	27.00	18.0	2.00
2165	3180000	Мурино	-1.0	25.00	17.0	3.00
3078	6000000	Санкт-Петербург	5247.0	43.00	29.3	3.20
3195	2690000	Санкт-Петербург	11281.0	25.60	16.2	3.80
6084	770000	Тихвин	-1.0	18.00	13.0	3.00
6262	3100000	Санкт-Петербург	4914.0	24.00	16.0	2.00
7047	2800000	Санкт-Петербург	3974.0	21.00	14.9	3.80
7950	5967734	Санкт-Петербург	20802.0	66.40	44.4	2.89
8712	14948000	Санкт-Петербург	3914.0	42.70	27.5	3.70
8729	2200000	Пушкин	30687.0	18.40	14.0	2.40
9138	2920000	Санкт-Петербург	4008.0	23.29	21.0	2.00
10058	3350000	поселок Шушары	-1.0	25.00	17.0	3.50
10208	1900000	Санкт-Петербург	5639.0	17.00	13.0	3.00
11033	5350000	Санкт-Петербург	3953.0	32.00	16.0	2.00
11475	2500000	деревня Яльгелево	-1.0	40.80	26.8	3.80
12370	7575000	Санкт-Петербург	14545.0	59.20	32.6	3.50
12595	3700000	Санкт-Петербург	13609.0	44.40	28.4	3.40
13814	3650000	Санкт-Петербург	16167.0	28.00	18.0	3.00
14183	5200000	Санкт-Петербург	2998.0	47.00	36.0	3.50
14836	3900000	поселок Стрельна	23654.0	45.00	28.4	3.30
15014	3350000	Санкт-Петербург	13871.0	28.30	20.2	3.00
16367	2720000	Санкт-Петербург	13268.0	25.70	22.6	2.00
17424	2700000	Санкт-Петербург	15319.0	26.69	17.4	3.00
17834	2940000	Мурино	-1.0	34.70	15.5	2.30
18066	3600000	Санкт-Петербург	11870.0	23.80	16.8	3.00

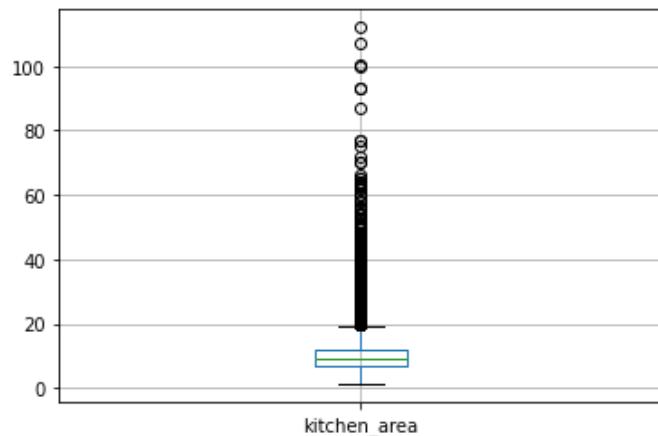
	last_price	locality_name	centers_range	total_area	living_area	kitchen_area
19642	1190000	Санкт-Петербург	11122.0	14.00	11.0	2.00
20217	4250000	Санкт-Петербург	12721.0	28.50	19.5	1.30
21419	1870000	Санкт-Петербург	15654.0	20.00	14.0	2.00
23498	1600000	Высоцк		-1.0	54.00	33.0

Хмм. Ну, ничего противозаконного тут нет. Наверное, будет правильным оставить эти данные. Посмотрим на диаграмму размаха.

In [110...]

```
data.boxplot(column='kitchen_area')
```

Out[110...]



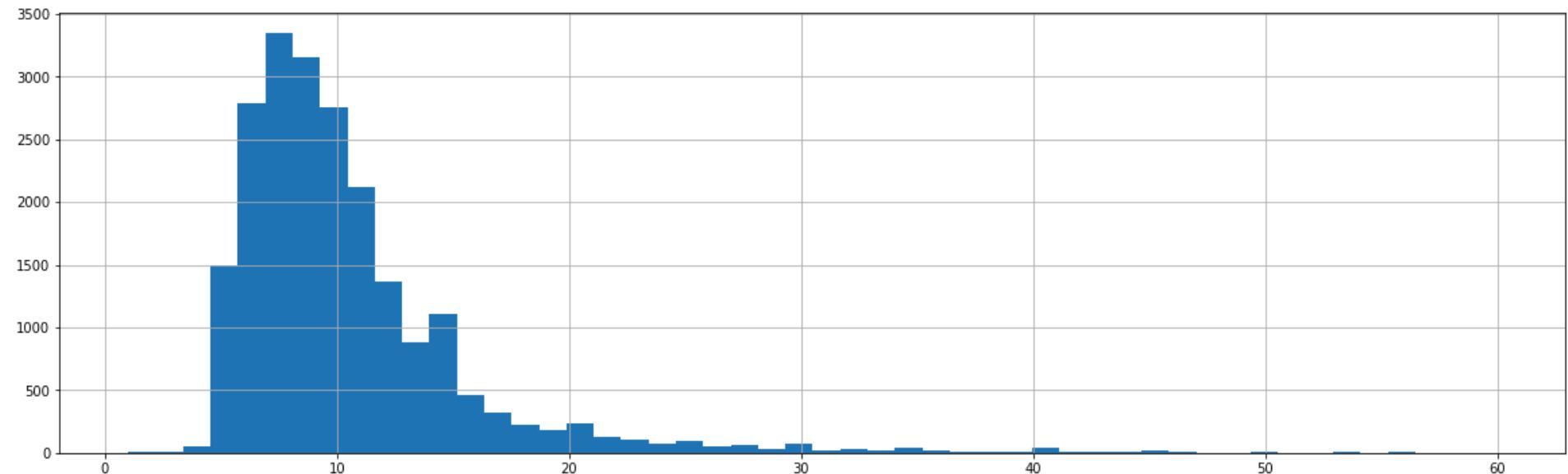
После 50 мало значений. Верхний ус упирается в 20, приближать не будем.

In [111...]

```
data['kitchen_area'].hist(bins=50, range=(1,60), figsize=(20,6))
```

Out[111...]

```
<AxesSubplot:>
```

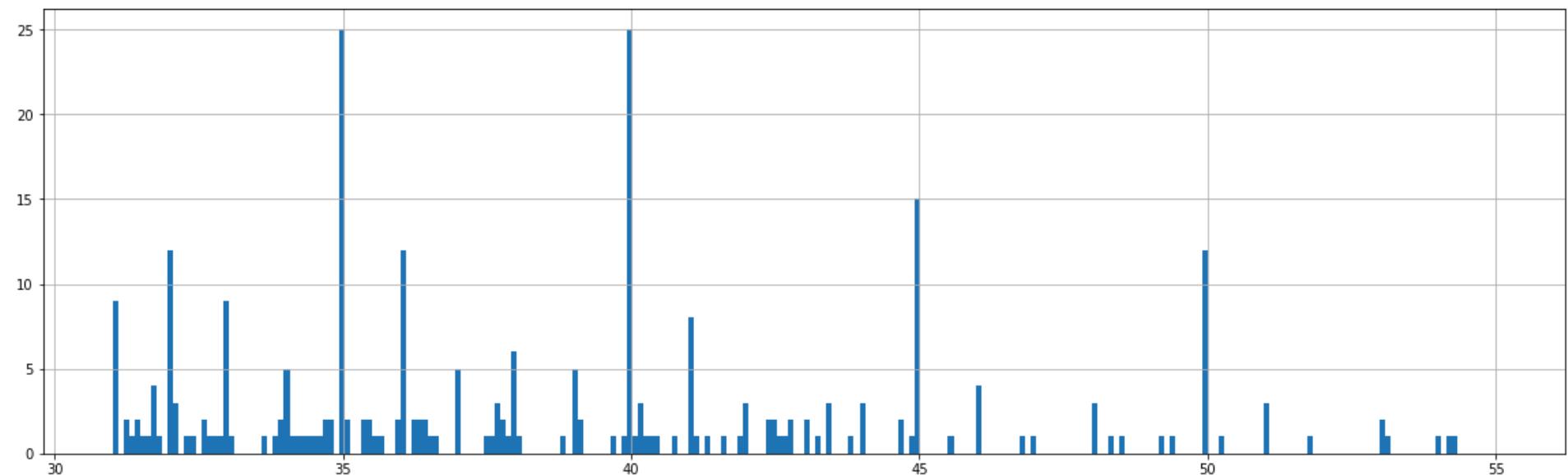


После 30 квартир мало, после 50 вообще ерунда. До 5 тоже очень мало.

In [112...]

```
data['kitchen_area'].hist(bins=250, range=(31,55), figsize=(20,6))
```

Out[112...]

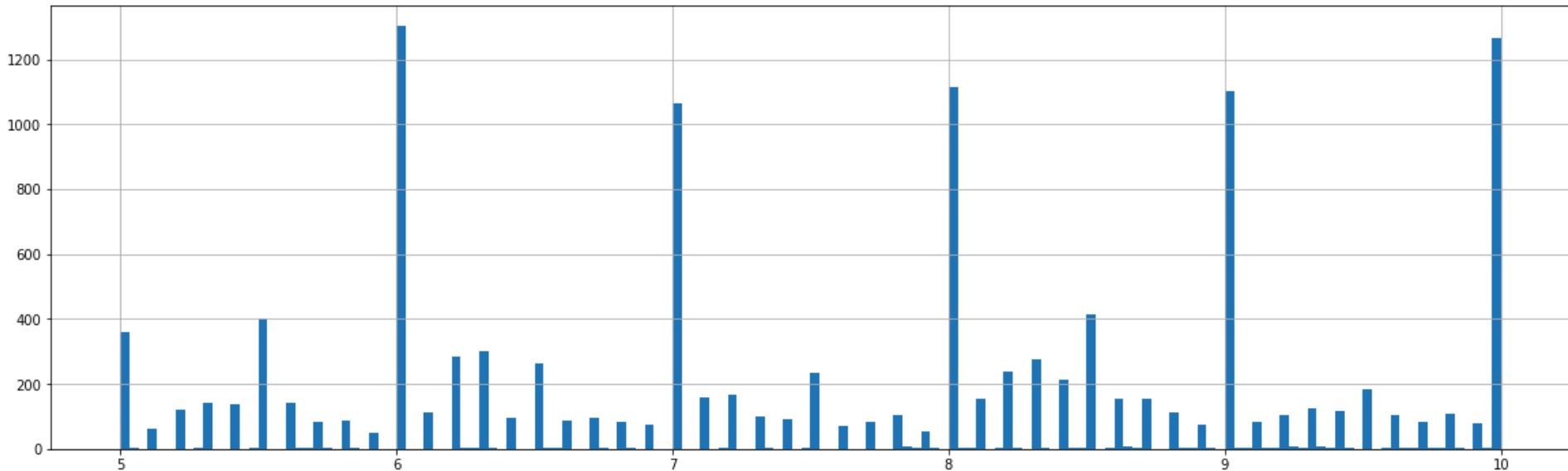


Бог с ним, оставим всё до 50. Пусть будет.

In [113...]

```
data['kitchen_area'].hist(bins=150, range=(5,10), figsize=(20,6))
```

Out[113...]



Люди прутся от целых чисел. Некоторые округляют до 0,5.

In [114...]

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

Out[114...]

```
count    21421.000000
mean     10.569807
std      5.905438
min      1.300000
25%      7.000000
50%      9.100000
75%      12.000000
max     112.000000
Name: kitchen_area, dtype: float64
```

In [115...]

```
data[data['kitchen_area'] <= 50]['kitchen_area'].describe()
```

Out[115...]

```
count    21375.000000
mean     10.450126
std      5.267284
min      1.300000
```

```
25%      7.000000
50%      9.100000
75%     12.000000
max     50.000000
Name: kitchen_area, dtype: float64
```

Не то чтобы стало сильно лучше. Ну, стандотклон пониже стал, славно.

Глянем ещё со строгими ограничениями.

```
In [116]: data[(data['kitchen_area'] <= 40) & (data['kitchen_area'] >= 5)]['kitchen_area'].describe()
```

```
Out[116]: count    21185.000000
mean      10.345124
std       4.801582
min       5.000000
25%      7.000000
50%      9.100000
75%     12.000000
max     40.000000
Name: kitchen_area, dtype: float64
```

Ну, стандотклон ещё пониже. Денис, что будет правильным выбрать? Денис сказал оставить до 50.

Итого. 75% квартир имеют кухни до 12 кв.м. Медиана 9.1 метров, средняя площадь 10,3 метра. Предел типичного значения 20 метров.

Цена объекта

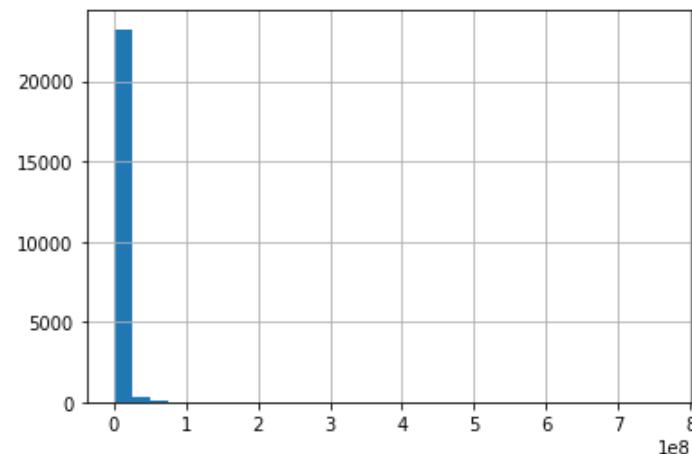
```
In [117]: data['last_price'].describe()
```

```
Out[117]: count    2.369900e+04
mean      6.541549e+06
std       1.088701e+07
min      1.219000e+04
25%      3.400000e+06
50%      4.650000e+06
75%      6.800000e+06
max     7.630000e+08
Name: last_price, dtype: float64
```

Долбаные порядки.

```
In [118]: data['last_price'].hist(bins=30)
```

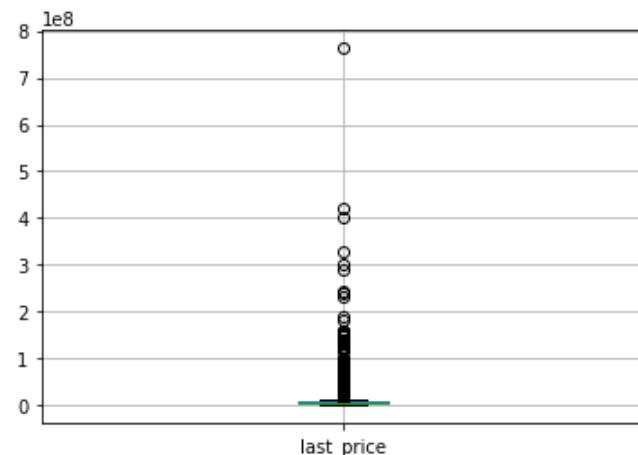
```
Out[118]: <AxesSubplot:>
```



Класс.

```
In [119... data.boxplot(column='last_price')
```

```
Out[119... <AxesSubplot:>
```



Ну, после 100кк значений мало. Давайте ещё раз, но с таким ограничением.

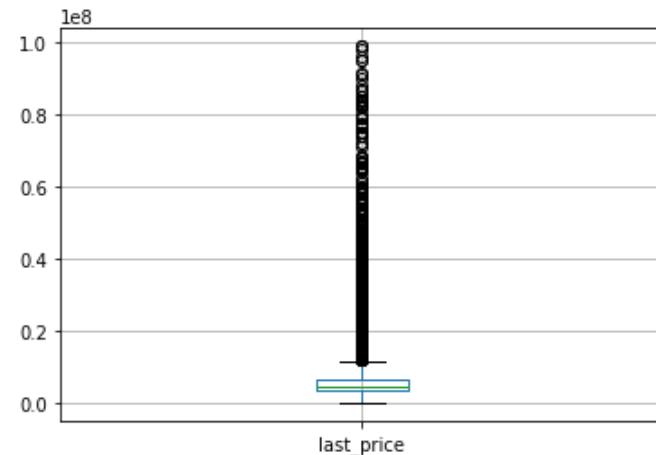
```
In [120... data[data['last_price'] > 100000000]['last_price'].count()
```

```
Out[120... 37
```

In [121...]

```
data.query('last_price < 100000000').boxplot(column='last_price')
```

Out[121...]



А если оставить только до 50кк.

In [122...]

```
data[data['last_price'] > 50000000]['last_price'].count()
```

Out[122...]

131

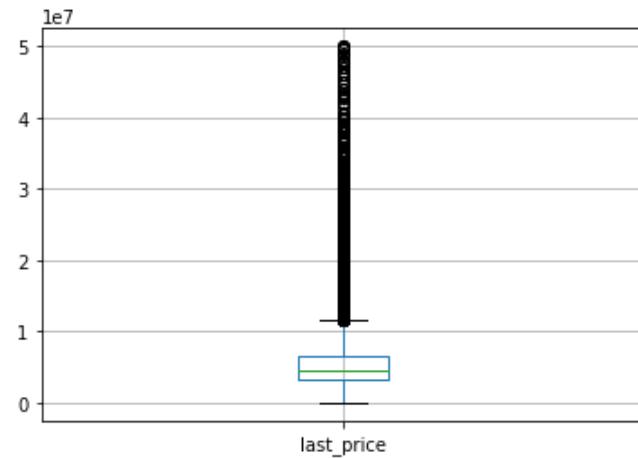
131 строка.

In [123...]

```
data.query('last_price < 50000000').boxplot(column='last_price')
```

Out[123...]

<AxesSubplot:>

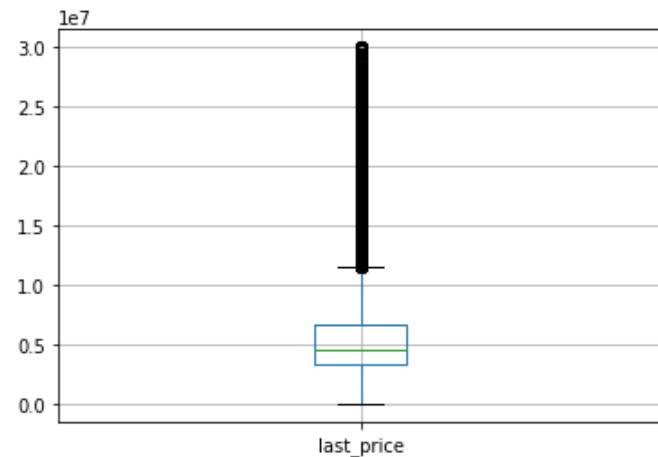


```
In [124...]: data[data['last_price'] > 30000000]['last_price'].count()
```

```
Out[124...]: 316
```

```
In [125...]: data.query('last_price < 30000000').boxplot(column='last_price')
```

```
Out[125...]: <AxesSubplot:>
```



На этом точно стоит остановиться. Вопрос в том, не нужно ли было остановиться раньше.

Поделим всё на лям?

```
In [126...]: data['price_kk'] = data['last_price'] / 1000000
```

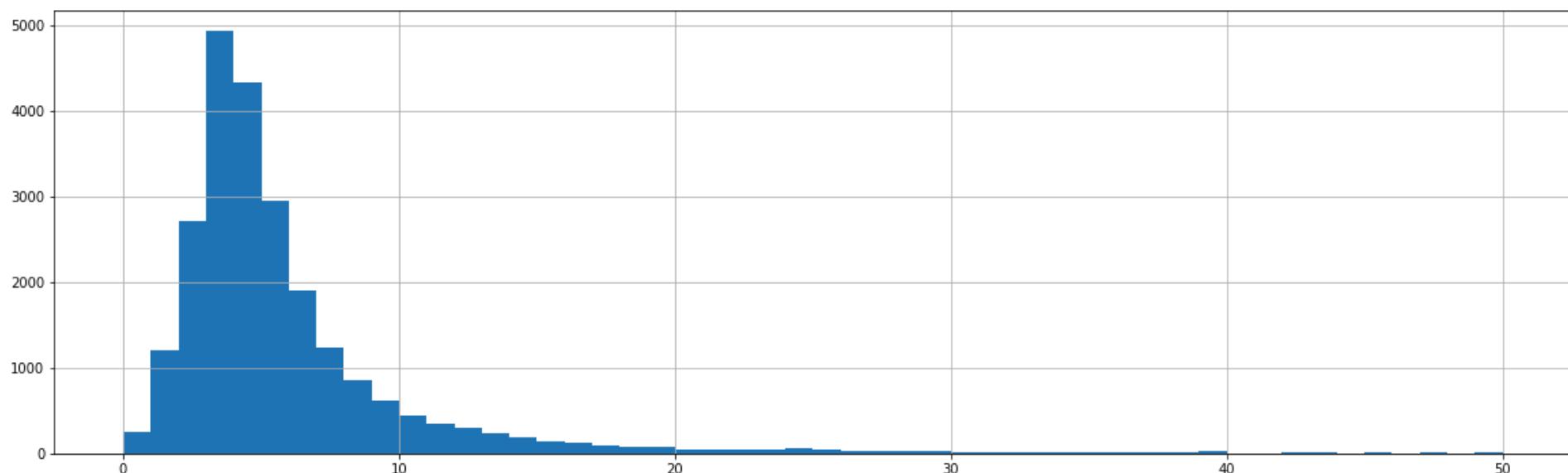
```
In [127...]: data['price_kk'] = data['price_kk'].round(3)
```

```
In [128...]: data['price_kk'].describe()
```

```
Out[128...]: count    23699.00000  
mean      6.541549  
std       10.887013  
min      0.012000  
25%      3.400000  
50%      4.650000  
75%      6.800000  
max     763.000000  
Name: price_kk, dtype: float64
```

```
In [129...]: data['price_kk'].hist(bins=50, range=(0.01, 50), figsize=(20,6))
```

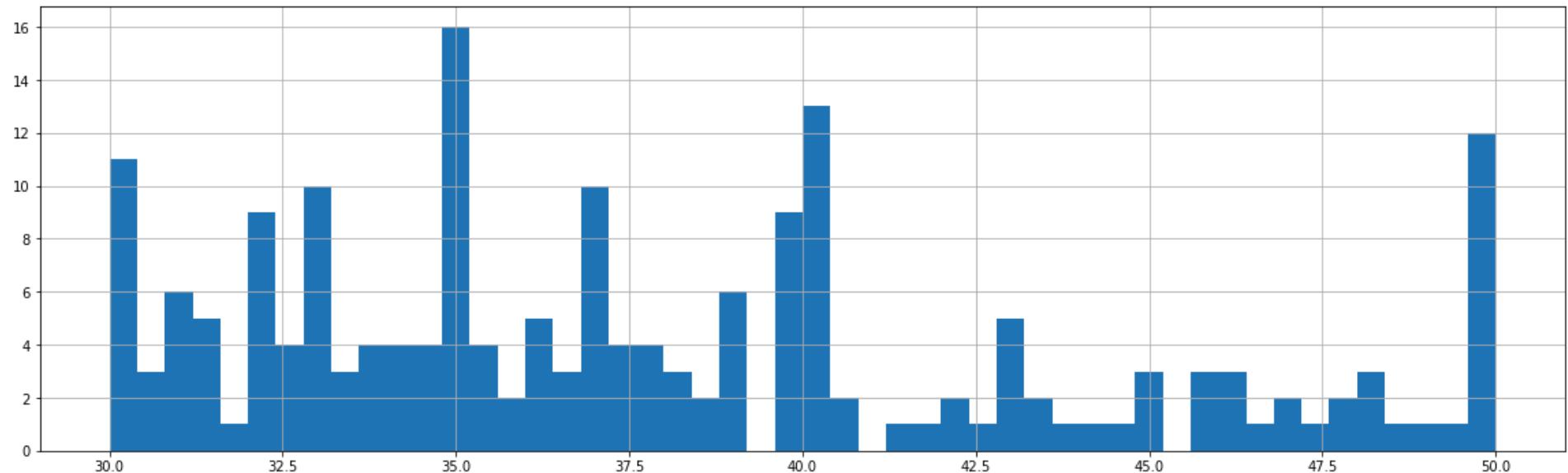
```
Out[129...]: <AxesSubplot:>
```



Выглядит сносно. Распределение уже похоже на логнормальное. Давайте посмотрим ближе на длинный хвост.

```
In [130...]: data['price_kk'].hist(bins=50, range=(30, 50), figsize=(20,6))
```

Out[130... <AxesSubplot:>



Сложно сказать. Я бы остановился на 40. После 35 ещё есть приличные данные. После 40 уже прям единицы, если не считать пика на 50.

In [131...]

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

Out[131...]

```
count    23699.000000
mean      6.541549
std       10.887013
min       0.012000
25%      3.400000
50%      4.650000
75%      6.800000
max      763.000000
Name: price_kk, dtype: float64
```

In [132...]

```
data[data['price_kk'] < 40]['price_kk'].describe()
```

Out[132...]

```
count    23505.000000
mean      5.895151
std       4.551259
min       0.012000
25%      3.400000
50%      4.600000
75%      6.700000
```

```
max      39.990000
Name: price_kk, dtype: float64
```

Класс. Медиана снизилась на 0.05, а среднее на 0.65. Это значит, что мы знатно почистили выбросы. Стандотклон уменьшился почти в 2.5 раза. Круто. Хотя разброс всё равно адовый. Стандотклон равен медиане. Среднее больше медианы на треть.

Итого. Медиана цены - 4.6кк. Средняя цена 5.9кк. 75% квартир стоят 6.7кк.

Количество комнат

In [133...]

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

Out[133...]

	count	mean	std	min	25%	50%	75%	max
	23699.000000	2.070636	1.078405	0.000000	1.000000	2.000000	3.000000	19.000000
Name:	rooms, dtype: float64							

0 комнат - это сильно. Типо в студии нет комнат, да? Смотрите, зато среднее и медиана почти равно! Это ли не прекрасно. Максимум комнат 19. Солидно.

In [134...]

```
data[data['rooms'] == 0][['rooms', 'studio', 'last_price', 'total_area', 'living_area', 'kitchen_area']]
```

Out[134...]

	rooms	studio	last_price	total_area	living_area	kitchen_area
144	0	True	2450000	27.00	15.50	NaN
349	0	False	2320000	25.00	17.00	NaN
440	0	True	2480000	27.11	24.75	NaN
508	0	False	3375000	34.40	24.30	NaN
608	0	True	1850000	25.00	NaN	NaN
...
23210	0	True	3200000	26.00	18.00	NaN
23442	0	False	2500000	27.70	18.00	NaN
23554	0	True	3350000	26.00	NaN	NaN
23592	0	False	1450000	29.00	18.00	NaN
23637	0	True	2350000	26.00	17.00	NaN

197 rows × 6 columns

Блин, 197. Придётся кодить.

In [135...]

```
data.query('rooms == 0').pivot_table(
    index='rooms', values=['studio', 'last_price', 'total_area', 'living_area', 'kitchen_area'],
    aggfunc=['mean'])
```

Out[135...]

	mean			
	last_price	living_area	studio	total_area
rooms				
0	3.337724e+06	18.865246	0.700508	29.279746

Нужно ли написать позапрошлую строку кода, но, только для НЕ студий?

In [136...]

```
data[(data['rooms'] == 0) & (data['studio'] == False)][['rooms', 'studio', 'last_price', 'total_area', 'living_area', 'kitchen_area']]
```

Out[136...]

	rooms	studio	last_price	total_area	living_area	kitchen_area
349	0	False	2320000	25.00	17.0	NaN
508	0	False	3375000	34.40	24.3	NaN
780	0	False	2600000	26.10	NaN	NaN
839	0	False	1900000	35.00	15.0	NaN
946	0	False	2200000	23.00	18.0	NaN
1574	0	False	2200000	22.00	15.0	NaN
1625	0	False	1980000	23.98	10.5	NaN
2532	0	False	3500000	27.10	18.7	NaN
3019	0	False	2100000	24.00	18.0	NaN
4115	0	False	2600000	24.00	18.0	NaN
4437	0	False	3200000	25.00	18.6	NaN
4683	0	False	3650000	35.00	23.4	NaN
4876	0	False	3000000	25.00	17.0	NaN
5749	0	False	3590000	25.00	NaN	NaN

	rooms	studio	last_price	total_area	living_area	kitchen_area
6472	0	False	3620000	28.00	18.0	NaN
6612	0	False	3590000	26.80	19.0	NaN
6805	0	False	1850000	31.00	18.0	NaN
7008	0	False	5200000	32.30	25.5	NaN
7237	0	False	2999000	42.63	25.7	NaN
7286	0	False	2580000	30.00	19.0	NaN
7818	0	False	3300000	27.30	NaN	NaN
9412	0	False	2100000	16.00	13.0	NaN
9586	0	False	1670000	28.30	18.2	NaN
9861	0	False	2350000	25.00	25.0	NaN
10284	0	False	2700000	24.00	15.5	NaN
10606	0	False	2950000	25.27	25.0	NaN
11035	0	False	2500000	26.00	14.0	NaN
11051	0	False	2200000	26.00	12.0	NaN
11157	0	False	2900000	27.30	20.0	NaN
11331	0	False	1315000	27.32	18.7	NaN
11692	0	False	2550000	23.06	18.0	NaN
11705	0	False	3380000	26.00	17.0	NaN
12412	0	False	3300000	27.00	25.0	NaN
12691	0	False	3700000	24.20	24.2	NaN
13953	0	False	3255000	28.20	20.0	NaN
15105	0	False	2500000	26.10	18.0	NaN
15273	0	False	2700000	25.20	15.2	NaN
15434	0	False	2200000	27.00	15.0	NaN
16429	0	False	2460000	28.01	18.1	NaN
17695	0	False	2550000	24.00	17.0	NaN
17729	0	False	2948000	28.05	21.8	NaN

	rooms	studio	last_price	total_area	living_area	kitchen_area
17805	0	False	2600000	31.10	21.4	NaN
17824	0	False	2300000	22.50	20.0	NaN
18042	0	False	2950000	25.90	16.9	NaN
18549	0	False	3699000	29.00	19.0	NaN
18782	0	False	2500000	25.00	17.0	NaN
19392	0	False	7100000	371.00	NaN	NaN
19477	0	False	2300000	20.00	16.0	NaN
19735	0	False	2800000	30.50	20.0	NaN
19917	0	False	2340000	25.41	18.5	NaN
20002	0	False	2700000	28.00	20.0	NaN
20045	0	False	3100000	25.00	18.0	NaN
20054	0	False	2730000	21.00	14.0	NaN
20793	0	False	3600000	27.50	19.0	NaN
21299	0	False	2450000	28.50	18.0	NaN
21628	0	False	3900000	34.00	24.0	NaN
22573	0	False	2250000	24.00	15.0	NaN
23442	0	False	2500000	27.70	18.0	NaN
23592	0	False	1450000	29.00	18.0	NaN

И что нам теперь делать? Менять почти во всех этих строках False на True? Ясно только, что париться о 0 комнат не нужно, т.к. это студии.

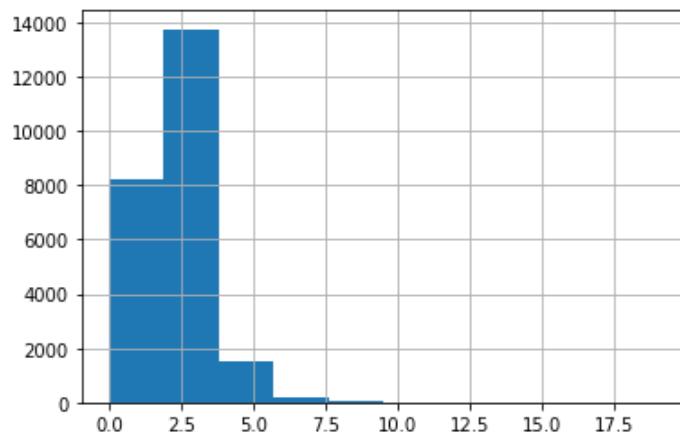
Мне кажется, нужно будет выкинуть квартиры с 0 комнат из этих рассчётов. Но я не уверен.

In [137...]

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

Out[137...]

```
<AxesSubplot:>
```



Мы видим, что у нас почти поровну однушек и двушек - около 8к. Поменьше трёшек - 5,8к где-то. Четырёшек немножко больше 1к, пятышек около 300. А где будем отрезать? Ксати, давайте выведем на экран нормальную статистику.

In [138]:
data.groupby('rooms')['rooms'].count()

Out[138]:
rooms
0 197
1 8047
2 7940
3 5814
4 1180
5 326
6 105
7 59
8 12
9 8
10 3
11 2
12 1
14 2
15 1
16 1
19 1
Name: rooms, dtype: int64

7 или 9? Давайте 9.

In [139]:
data['rooms'].describe()

Out[139]:
count 23699.000000
mean 2.070636
std 1.078405

```
min      0.000000
25%    1.000000
50%    2.000000
75%    3.000000
max   19.000000
Name: rooms, dtype: float64
```

In [140...]

```
data.loc[data['rooms'] < 10, 'rooms'].describe()
```

Out[140...]

```
count  23688.000000
mean   2.065603
std    1.051309
min    0.000000
25%    1.000000
50%    2.000000
75%    3.000000
max    9.000000
Name: rooms, dtype: float64
```

Ну, стало чуть однороднее, но существенной разницы нет, что было ожидаемо. У нас же не было квартиры с тысячей комнат или типа того.

Итого. 67% квартир - 1 и 2 комнаты, их поровну. 25% - 3 комнаты. Суммарно их 92%. Квартиры с 0 комнат - почти только студии.

Высота потолков

In [141...]

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

Out[141...]

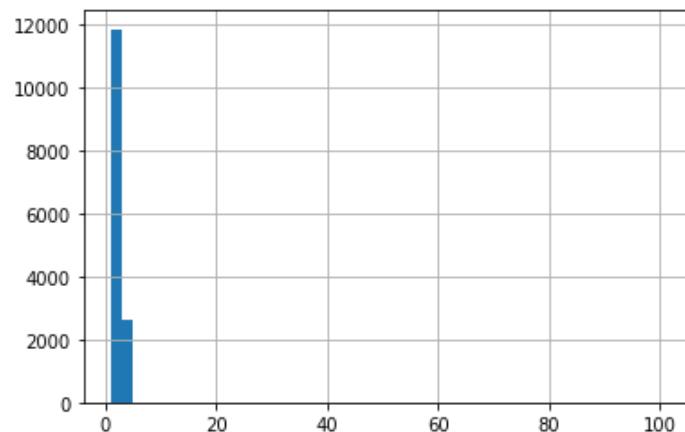
```
count  14504.000000
mean   2.771499
std    1.261056
min    1.000000
25%    2.520000
50%    2.650000
75%    2.800000
max   100.000000
Name: ceiling_height, dtype: float64
```

In [142...]

```
data['ceiling_height'].hist(bins=50)
```

Out[142...]

```
<AxesSubplot:>
```



Так, разобраться с этим можно было ещё в предобработке, но сейчас уже точно пора.

In [143...]

```
data[(data['ceiling_height'] > 9) | (data['ceiling_height'] < 2)]
```

Out[143...]

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	ponds_around3000	ponds_nearest	days_expo
355	17	3600000	55.2	2018-07-12	2	25.00	5	32.0	2	False	...	NaN	NaN	
3148	14	2900000	75.0	2018-11-12	3	32.00	3	53.0	2	False	...	NaN	NaN	
4643	0	4300000	45.0	2018-02-01	2	25.00	9	30.0	2	False	...	1.0	331.0	
4876	7	3000000	25.0	2017-09-27	0	27.00	25	17.0	17	False	...	NaN	NaN	
5076	0	3850000	30.5	2018-10-03	1	24.00	5	19.5	1	False	...	1.0	578.0	
5246	0	2500000	54.0	2017-10-13	2	27.00	5	30.0	3	False	...	NaN	NaN	
5669	4	4400000	50.0	2017-08-08	2	26.00	9	21.3	3	False	...	0.0	-1.0	
5712	5	1500000	42.8	2017-08-14	2	1.20	2	27.5	1	False	...	NaN	NaN	
5807	17	8150000	80.0	2019-01-09	2	27.00	36	41.0	13	False	...	3.0	80.0	
6246	6	3300000	44.4	2019-03-25	2	25.00	5	31.3	5	False	...	2.0	73.0	
9379	5	3950000	42.0	2017-03-26	3	25.00	5	30.0	2	False	...	0.0	-1.0	
10773	8	3800000	58.0	2017-10-13	2	27.00	10	30.1	3	False	...	NaN	NaN	
11285	0	1950000	37.0	2019-03-20	1	25.00	5	17.0	4	False	...	NaN	NaN	
14382	9	1700000	35.0	2015-12-04	1	25.00	5	20.0	2	False	...	NaN	NaN	
15061	19	5600000	56.4	2018-05-11	2	14.00	14	32.4	5	False	...	0.0	-1.0	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	ponds_around3000	ponds_nearest	days_expo
16934	5	4100000	40.0	2017-10-17	1	1.75	37	17.4	5	False	...	3.0	80.0	
17496	15	6700000	92.9	2019-02-19	3	20.00	17	53.2	14	False	...	0.0	-1.0	
17857	1	3900000	56.0	2017-12-22	3	27.00	5	33.0	4	False	...	0.0	-1.0	
18545	6	3750000	43.0	2019-03-18	2	25.00	5	29.0	3	False	...	0.0	-1.0	
20478	11	8000000	45.0	2017-07-18	1	27.00	4	22.0	2	False	...	3.0	449.0	
20507	12	5950000	60.0	2018-02-19	2	22.60	14	35.0	11	False	...	0.0	-1.0	
21377	19	4900000	42.0	2017-04-18	1	27.50	24	37.7	19	False	...	0.0	-1.0	
21824	20	2450000	44.0	2019-02-12	2	27.00	2	38.0	2	False	...	NaN	NaN	
22309	20	5300000	45.0	2017-09-30	1	10.30	16	15.5	15	False	...	2.0	450.0	
22336	19	9999000	92.4	2019-04-05	2	32.00	6	55.5	5	False	...	3.0	511.0	
22590	16	6000000	55.0	2018-10-31	2	1.00	12	32.4	7	False	...	2.0	289.0	
22869	0	15000000	25.0	2018-07-25	1	100.00	5	14.0	5	False	...	3.0	30.0	
22938	14	4000000	98.0	2018-03-15	4	27.00	2	73.0	2	False	...	NaN	NaN	

28 rows × 28 columns

Какие-то явно нужно разделить на 10. Какие-то, кажется, можно исправить вручную (например, 1.75 заменить на 2.75). С какими-то непонятно, что делать (например строка №15061 с высотой 14 метров, в доме с 14 этажами и на 5 этаже или строка №22869 с высотой 100м). Давайте поступим так: всё, что больше 20 и меньше 35 - поделим на 10. Остальное выкинем.

```
In [144]: data.loc[(data['ceiling_height'] > 20) & (data['ceiling_height'] < 35), 'ceiling_height'] /= 10
```

```
In [145]: data[(data['ceiling_height'] > 20) & (data['ceiling_height'] < 35)]
```

```
Out[145]: total_images  last_price  total_area  first_day_exposition  rooms  ceiling_height  floors_total  living_area  floor  studio  ...  ponds_around3000  ponds_nearest  days_exposition
```

0 rows × 28 columns

Отлично, теперь заменим аномальные значения на -1.

In [146...]

```
data.loc[(data['ceiling_height'] > 10) | (data['ceiling_height'] < 2), 'ceiling_height'] = -1
```

Снова обратимся к описательной статистике.

In [147...]

```
data[data['ceiling_height'] > 0]['ceiling_height'].describe()
```

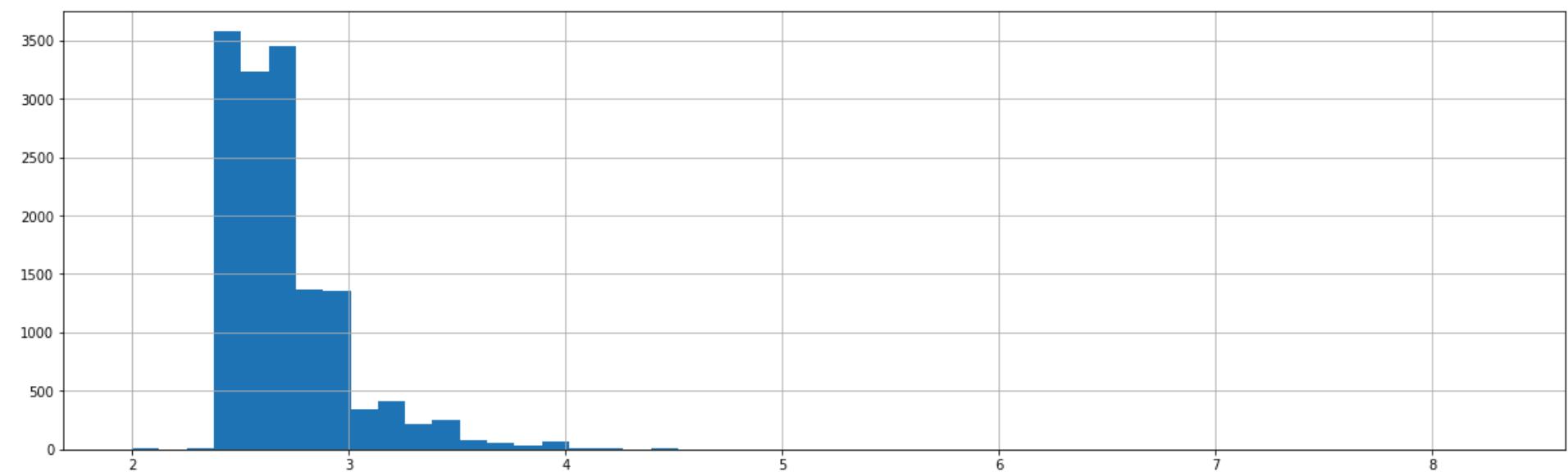
Out[147...]

```
count    14497.000000
mean      2.728149
std       0.292435
min      2.000000
25%     2.510000
50%     2.650000
75%     2.800000
max      8.300000
Name: ceiling_height, dtype: float64
```

In [148...]

```
data[data['ceiling_height'] > 0]['ceiling_height'].hist(bins=50, figsize=(20, 6))
```

Out[148...]

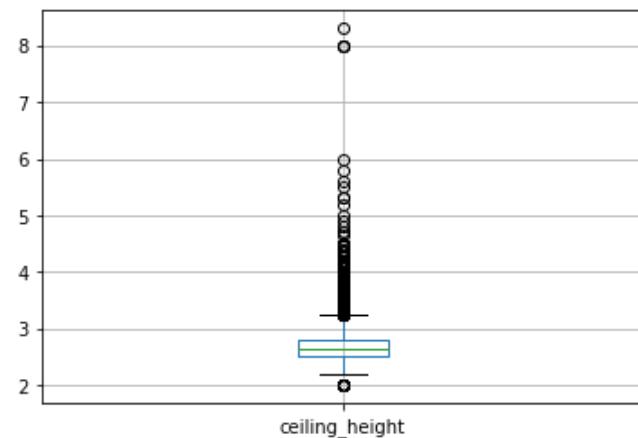


In [149...]

```
data.query('ceiling_height > 0').boxplot(column='ceiling_height')
```

Out[149...]

<AxesSubplot:>

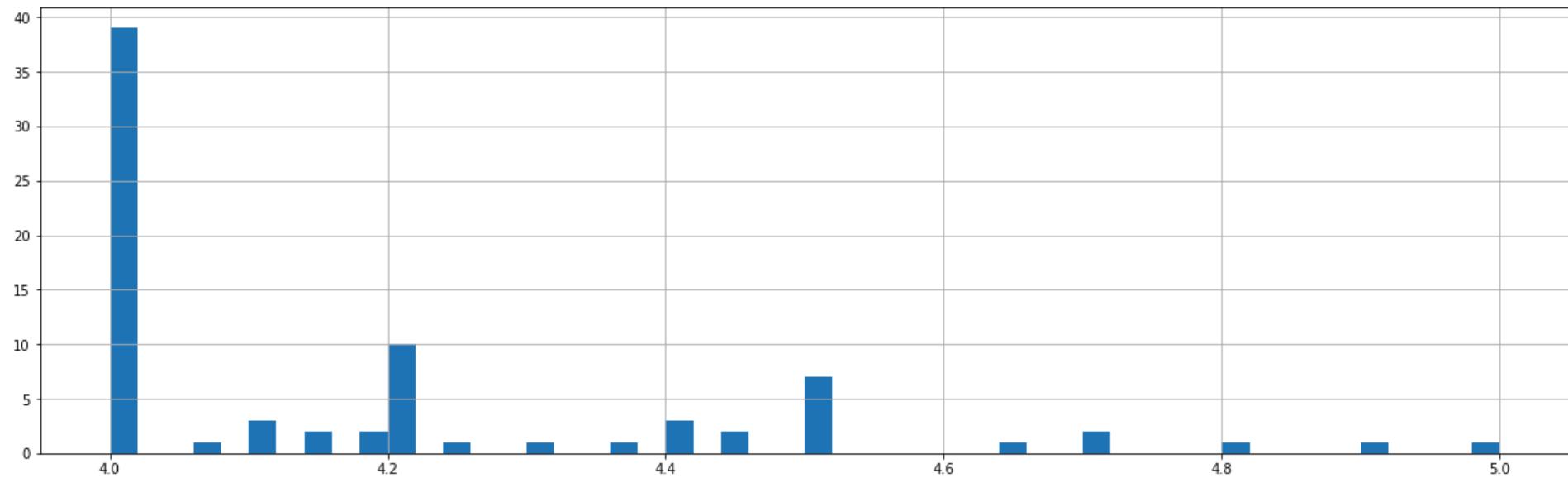


Гораздо лучше. Думаю, отрежем всё после 4. Посмотрим на этот участок на гистограмме.

In [150...]

```
data[data['ceiling_height'] > 0]['ceiling_height'].hist(bins=50, range=(4, 5), figsize=(20, 6))
```

Out[150...]

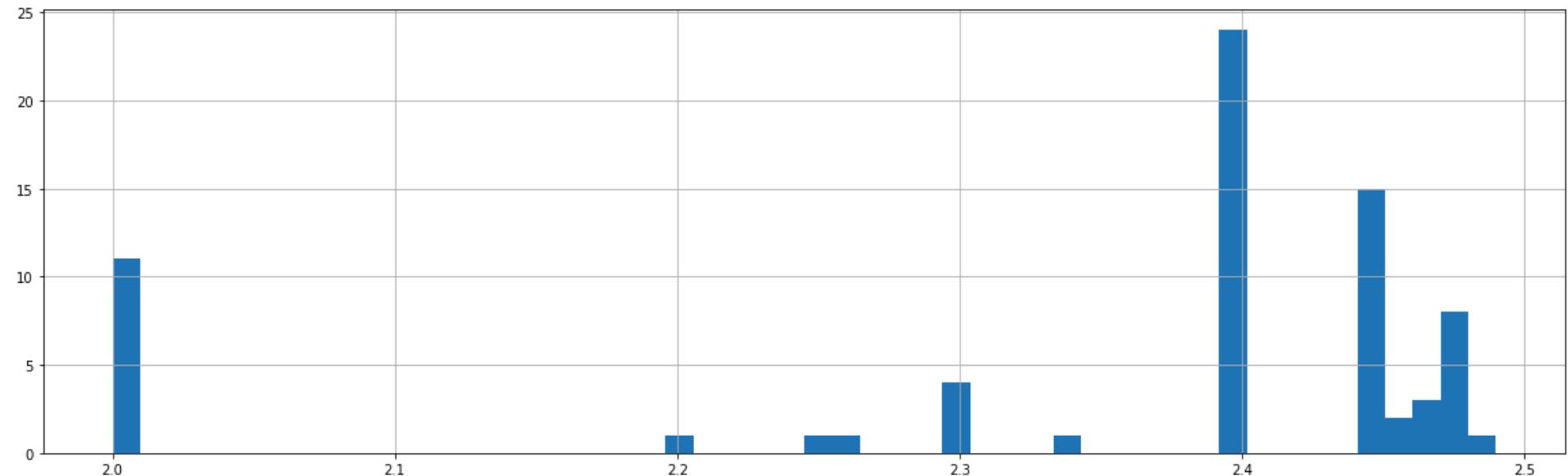


После 4 очень мало, обрезаем. А что по высоте меньше 2,5?

In [151...]

```
data[data['ceiling_height'] > 0]['ceiling_height'].hist(bins=50, range=(2, 2.49), figsize=(20, 6))
```

Out[151... <AxesSubplot:>

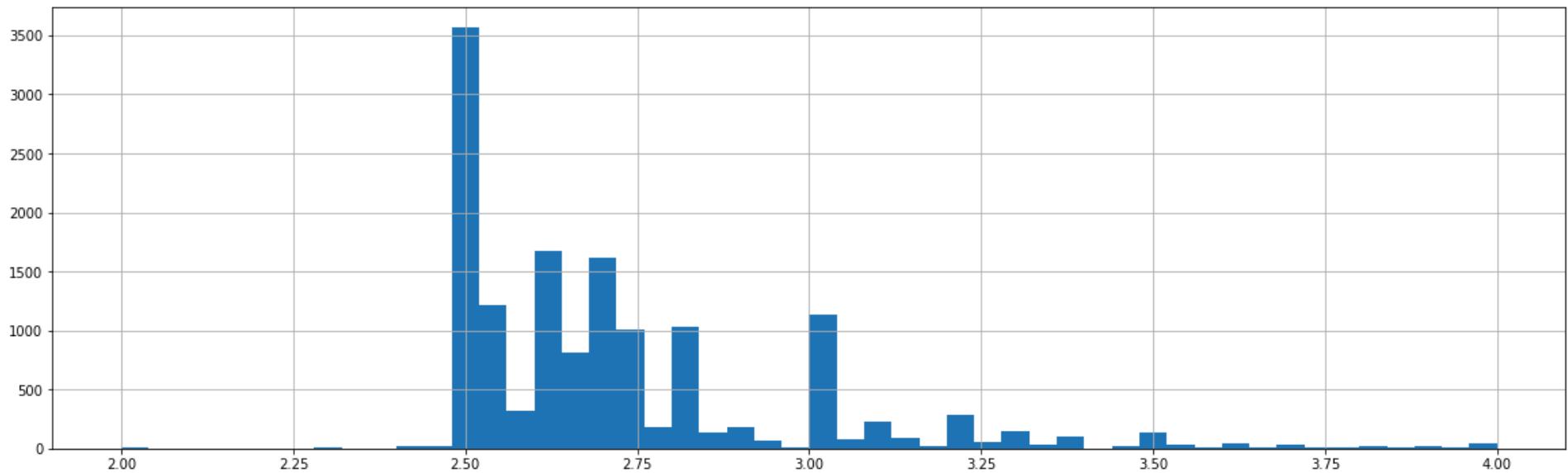


Мало, но что-то есть. Оставим. Денис сказал, что меньше 2,4 можно обрезать.

In [152...]

```
data[data['ceiling_height'] > 0]['ceiling_height'].hist(bins=50, range=(2, 4), figsize=(20, 6))
```

Out[152... <AxesSubplot:>



Почти всё в пределах 2,5м-3м. Давайте посчитаем, сколько точно.

```
In [153...]: data[(data['ceiling_height'] >= 2.5) & (data['ceiling_height'] <= 3)]['ceiling_height'].count()
```

```
Out[153...]: 12919
```

```
In [154...]: data[(data['ceiling_height'] > 0) & (data['ceiling_height'] <= 4)]['ceiling_height'].describe()
```

```
Out[154...]: count    14447.000000
mean        2.720889
std         0.258094
min         2.000000
25%        2.510000
50%        2.650000
75%        2.800000
max         4.000000
Name: ceiling_height, dtype: float64
```

Итого. Средняя высота = 2,72 метра. Медианная 2,65 метра. 90% квартир имеют потолки от 2,5 до 3 метров. Мы обработали аномальные значения и взяли за верхнюю границу высоты 4 метра.

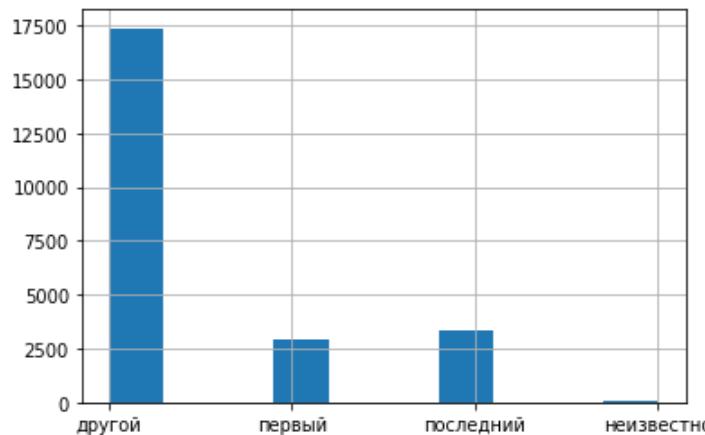
Тип этажа

```
In [155...]: data['floor_type'].value_counts()
```

```
Out[155...]: другой      17363  
последний     3361  
первый        2892  
неизвестно    83  
Name: floor_type, dtype: int64
```

```
In [156...]: data['floor_type'].hist()
```

```
Out[156...]: <AxesSubplot:>
```



Итого. 12% первых этажей. 14% последних. 74% других.

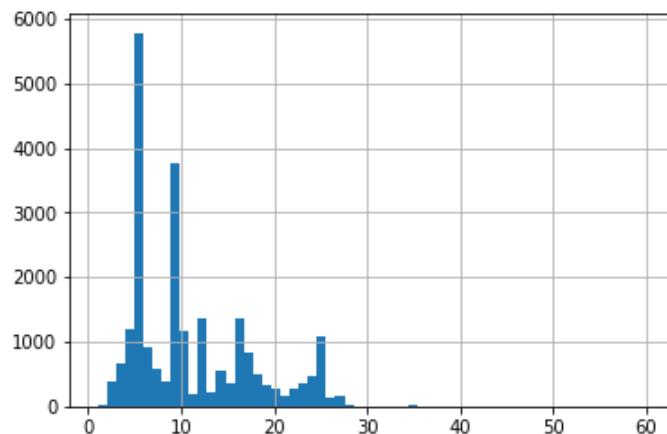
Количество этажей в доме

```
In [157...]: data[data['floors_total'] > 0]['floors_total'].describe()
```

```
Out[157...]: count    23613.000000  
mean       10.673824  
std        6.597173  
min        1.000000  
25%        5.000000  
50%        9.000000  
75%        16.000000  
max       60.000000  
Name: floors_total, dtype: float64
```

```
In [158...]: data[data['floors_total'] > 0]['floors_total'].hist(bins=60)
```

```
Out[158...]: <AxesSubplot:>
```

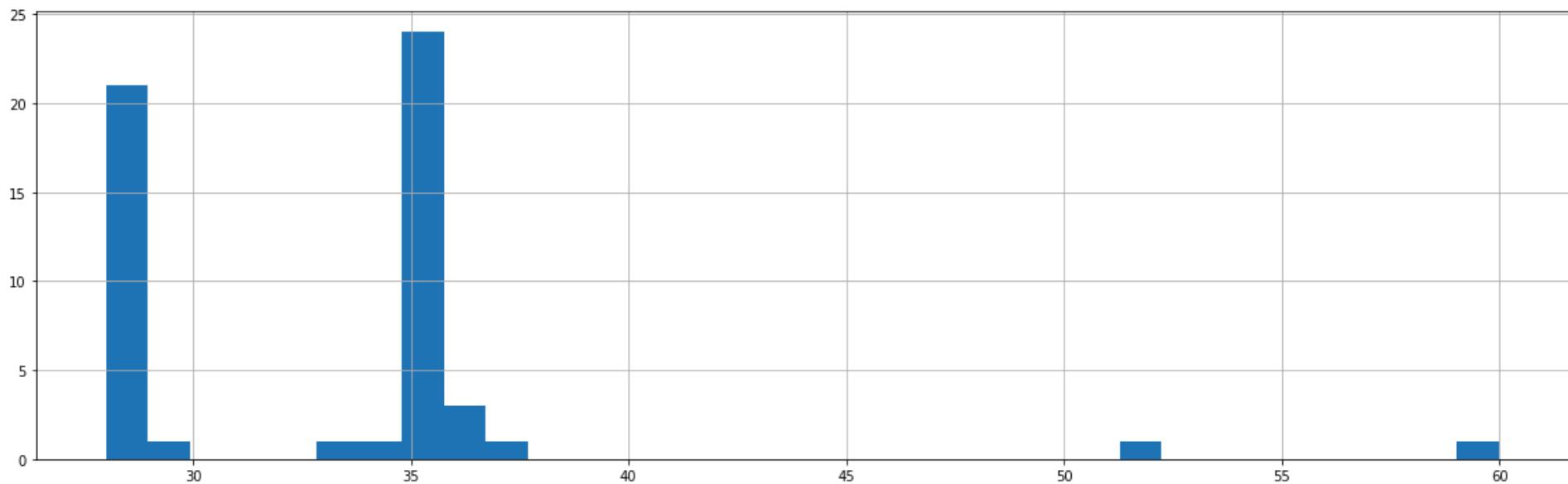


Ну, в целом, это похоже на отстойное распределение с поправкой на выбросы, связанные с большим количеством типовых домов.

In [159...]

```
data[data['floors_total'] > 0]['floors_total'].hist(bins=33, range=(28,60), figsize=(20, 6))
```

Out[159...]

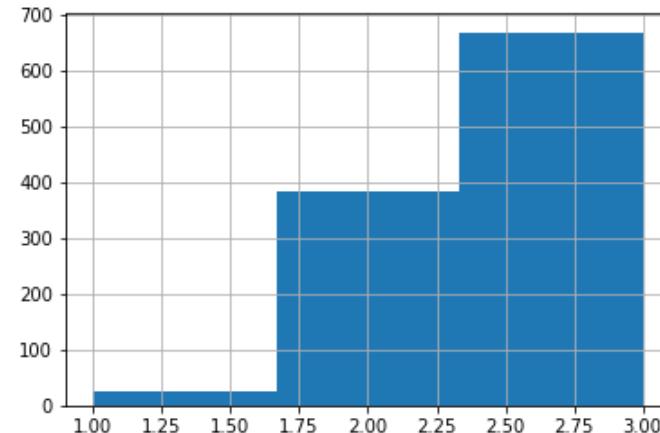


Обрежем после 27.

In [160...]

```
data[data['floors_total'] > 0]['floors_total'].hist(bins=3, range=(1,3))
```

Out[160... <AxesSubplot:>



Посмотрим на данные по одноэтажным домам.

In [161...]

```
data[data['floors_total'] == 1][['last_price', 'total_area', 'living_area', 'kitchen_area', 'locality_name', 'centers_range_km']]
```

Out[161...]

	last_price	total_area	living_area	kitchen_area	locality_name	centers_range_km
3076	2000000	80.0	48.5	10.0	Волосово	-0.0
4379	5300000	58.5	31.6	9.4	поселок Рошино	-0.0
5300	2990000	60.0	50.0	10.0	поселок станции Вещево	-0.0
5698	450000	42.0	23.0	5.8	поселок Будогощь	-0.0
5787	9000000	88.0	NaN	NaN	Санкт-Петербург	21.0
7962	550000	46.2	NaN	NaN	Луга	-0.0
8335	2700000	80.0	45.0	15.0	Луга	-0.0
8388	3550000	48.2	33.9	9.1	Сестрорецк	31.0
9517	1200000	39.4	NaN	NaN	поселок Гаврилово	-0.0
9752	850000	62.0	31.0	11.0	деревня Сижно	-0.0
10817	790000	50.6	32.0	NaN	поселок Ефимовский	-0.0
11641	3685000	44.7	23.5	15.0	Санкт-Петербург	15.0
11746	1070000	38.8	28.1	5.5	деревня Большие Колпаны	-0.0
14836	3900000	45.0	28.4	3.3	поселок Стрельна	24.0

	last_price	total_area	living_area	kitchen_area	locality_name	centers_range_km
15543	3500000	54.0	24.4	7.3	поселок Большая Ижора	-0.0
16444	990000	88.0	40.0	6.2	поселок Оредеж	-0.0
17020	3700000	100.0	35.0	22.0	деревня Каськово	-0.0
17744	2300000	50.9	30.0	7.0	поселок Суида	-0.0
18900	1800000	65.0	NaN	NaN	поселок Свирьстрой	-0.0
19590	6150000	95.0	65.0	21.0	поселок Рошино	-0.0
21603	1900000	40.0	19.2	6.5	поселок Сосново	-0.0
22550	5800000	115.0	40.0	14.0	Сестрорецк	35.0
22841	980000	54.0	NaN	NaN	поселок Пчевжа	-0.0
22855	2950000	31.0	16.4	5.6	Зеленогорск	54.0
23498	1600000	54.0	33.0	3.5	Высоцк	-0.0

Ладно, оставим их.

In [162...]

```
data[(data['floors_total'] > 4) & (data['floors_total'] < 26)]['floors_total'].count()
```

Out[162...]

20995

20995 домов в промежутке между 5 и 25 этажами включительно.

In [163...]

```
data[(data['floors_total'] > 0) & (data['floors_total'] < 28)]['floors_total'].describe()
```

Out[163...]

count	23559.000000
mean	10.622692
std	6.511834
min	1.000000
25%	5.000000
50%	9.000000
75%	16.000000
max	27.000000
Name:	floors_total, dtype: float64

Итого. Медианное значение 9 этажей. Минимум 1 этаж. Справедливо, меньше этажей строить смысла мало. Максимум этажей 60, но мы не будем брать в рассчёты все, что выше 27. 89% домов - от 5 до 25 этажей.

Расстояние до центра города в метрах

In [164...]

```
data[data['centers_range'] > 0]['centers_range'].describe()
```

Out[164...]

count	18180.000000
mean	14191.277833
std	8608.386210
min	181.000000
25%	9238.000000
50%	13098.500000
75%	16293.000000
max	65968.000000
Name:	centers_range, dtype: float64

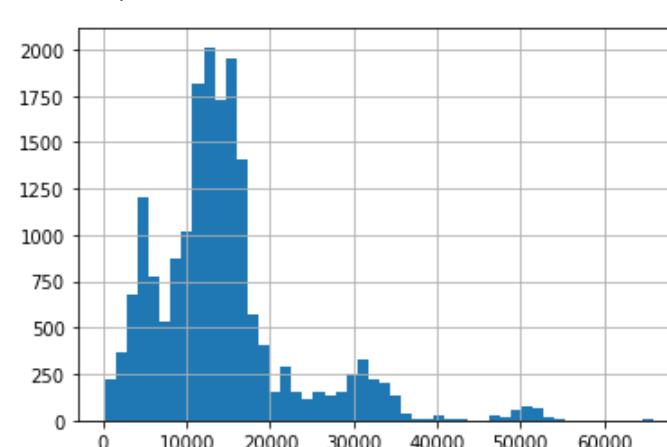
Диаметр Санкт-Петербурга в разных направлениях различается. Если говорить о протяжённости города с севера на юг, то в пределах КАД она составляет около 32 км, а за пределами КАД - 52 км. В направлении с северо-запада на юго-восток протяжённость города составляет около 90 км.

Чёрт, я не знал, что Питер так сильно растянут. Спасибо, гугл. Ой, яндекс, простите, пожалуйста.

In [165...]

```
data[data['centers_range'] > 0]['centers_range'].hist(bins=50)
```

Out[165...]



Начиная с 20 км объявлений сильно меньше. Почему? Во-первых, кроме как в Питере таких расстояний, вероятно, нет. Нужно ли во-вторых? Давайте посмотрим распределение по Питеру.

In [166...]

```
data[(data['centers_range'] > 0) & (data['locality_name'] == 'Санкт-Петербург')]['centers_range'].count()
```

Out[166...]

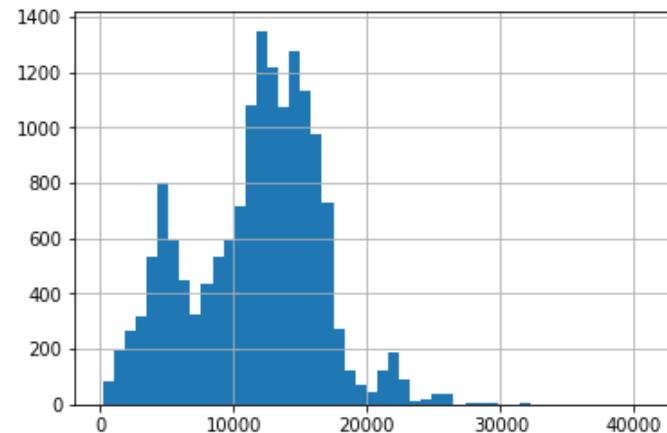
15701

Получается, из других городов только около 2500 объявлений.

In [167...]

```
data[(data['centers_range'] > 0) & (data['locality_name'] == 'Санкт-Петербург')]['centers_range'].hist(bins=50)
```

Out[167...]

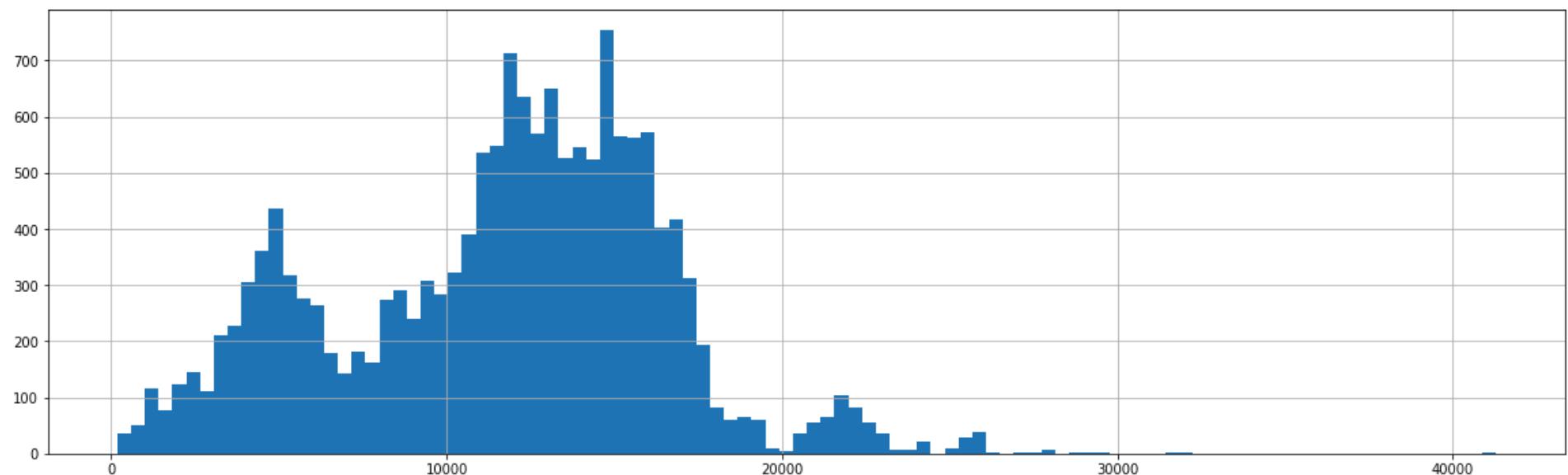


Хмм. Давайте улучшим отображение.

In [168...]

```
data[(data['centers_range'] > 0) & (data['locality_name'] == 'Санкт-Петербург')]['centers_range'].hist(bins=100, figsize=(20,6))
```

Out[168...]

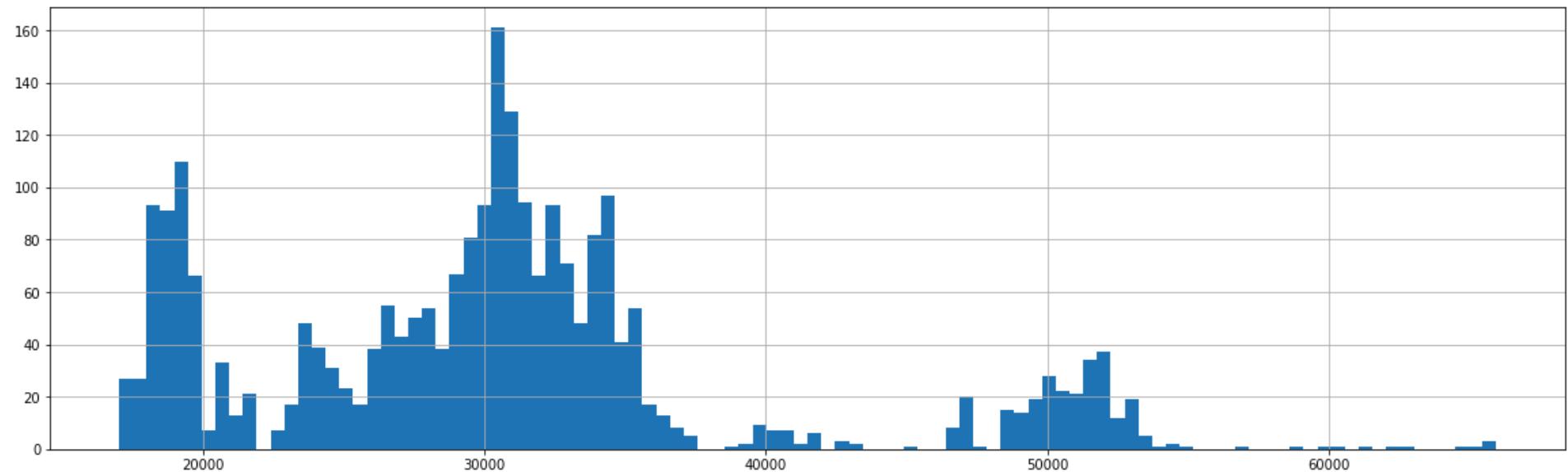


Something's wrong I can feel it. Посмотрим на непитерские объявления.

In [169...]

```
data[(data['centers_range'] > 0) & (data['locality_name'] != 'Санкт-Петербург')]['centers_range'].hist(bins=100, figsize=(20,6))
```

Out[169...]



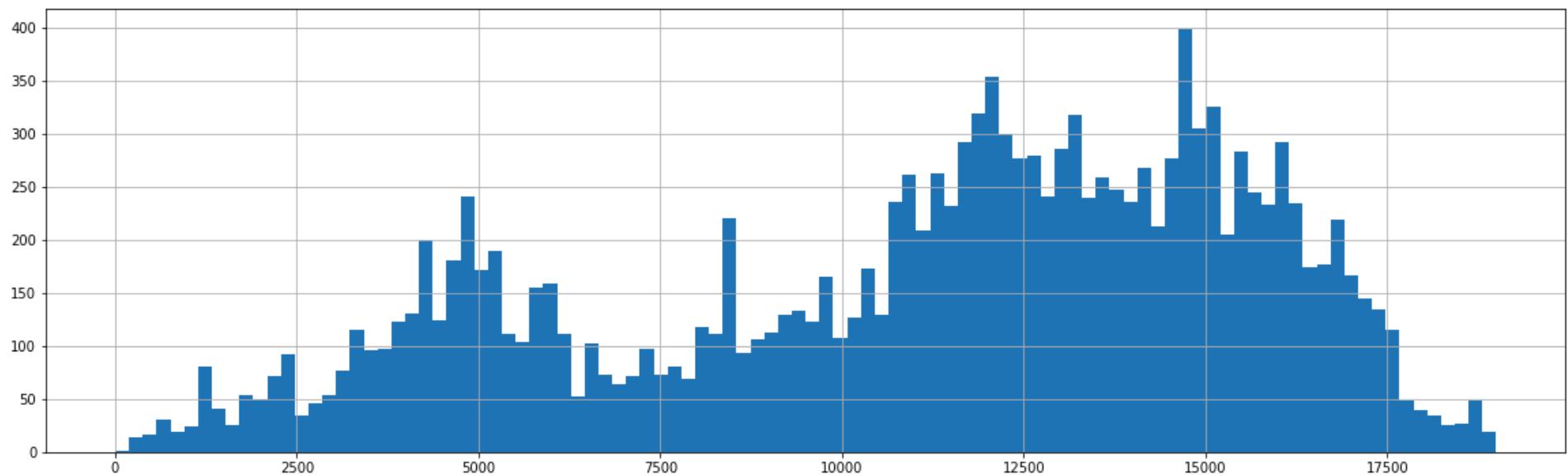
Так, я отказываюсь это понимать. Просто проигнорируем то, что мы сейчас увидели. Забудем как страшный сон. Этого не было, мы сюда не заглядывали.

In [170...]

```
data[(data['centers_range'] > 0) & (data['locality_name'] == 'Санкт-Петербург')]['centers_range'].hist(bins=100, range=(0, 19000), figsize=(20,6))
```

Out[170...]

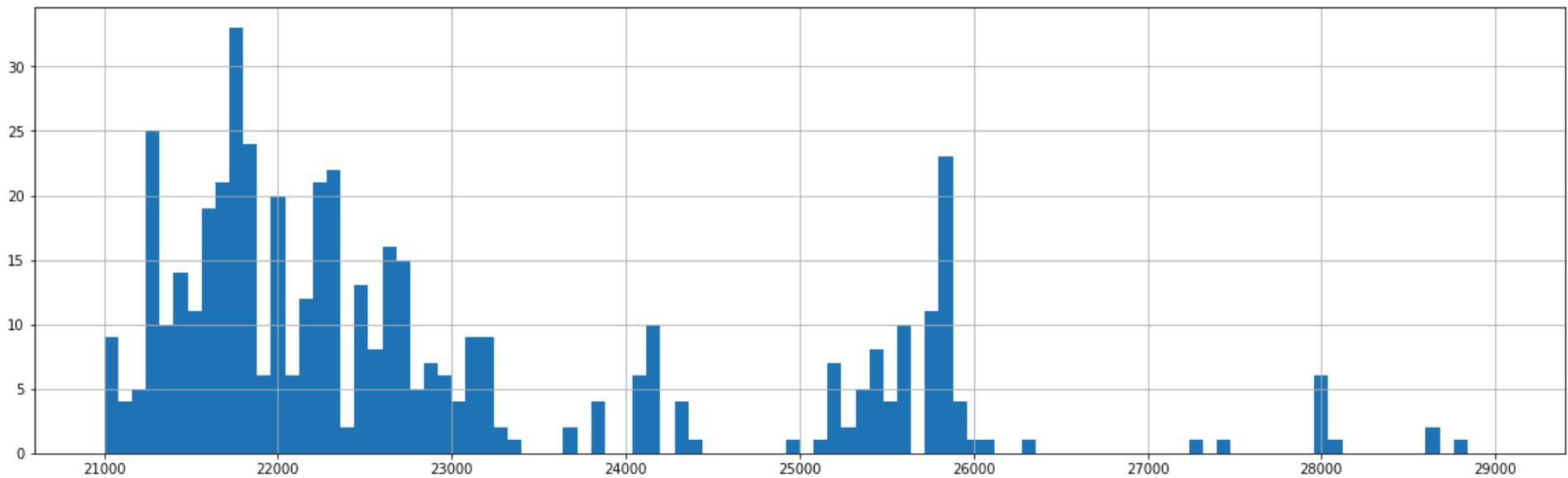
<AxesSubplot:>



Распределение выглядит не очень, но я думаю, что это связано с тем, что на расстоянии около 15 км в некоторых направлениях город заканчивается. Провал между 5 и 11 км тоже странен, но я не думаю, что нам следует исследовать это. Наверняка тоже какие-то особенности городского планирования, застройки и топографии.

```
In [171]: data[(data['centers_range'] > 0) & (data['locality_name'] == 'Санкт-Петербург')]['centers_range'].hist(bins=100, range=(21000, 29000), figsize=(20,
```

```
Out[171]: <AxesSubplot:>
```



In [172...]

```
data[(data['centers_range'] > 0) & (data['centers_range'] < 37000)]['centers_range'].describe()
```

Out[172...]

count	17863.000000
mean	13563.478531
std	7239.732257
min	181.000000
25%	9098.500000
50%	12995.000000
75%	16109.000000
max	36998.000000
Name:	centers_range, dtype: float64

Сложно рассматривать этот параметр для всех населённых пунктов сразу. Возможно стоило выкатить 2 анализа: отдельно для Питера и для всех остальных мест. Но это было бы актуально и для всех остальных столбцов. И вообще много как можно было бы разделить анализ по группам данных. Например, Денис сказал, что стоило бы убрать очень дорогие квартиры из общей группы и анализировать их отдельно. В общем, я не буду закапываться в этом всём и просто посчитаю мух вместе с котлетами.

Итого. После 37 км квартир почти нет, ограничили потолок этим уровнем. Медианной расстояние 13км. 75% квартир находятся в пределах 16км от центра населённого пункта.

Расстояние до ближайшего парка

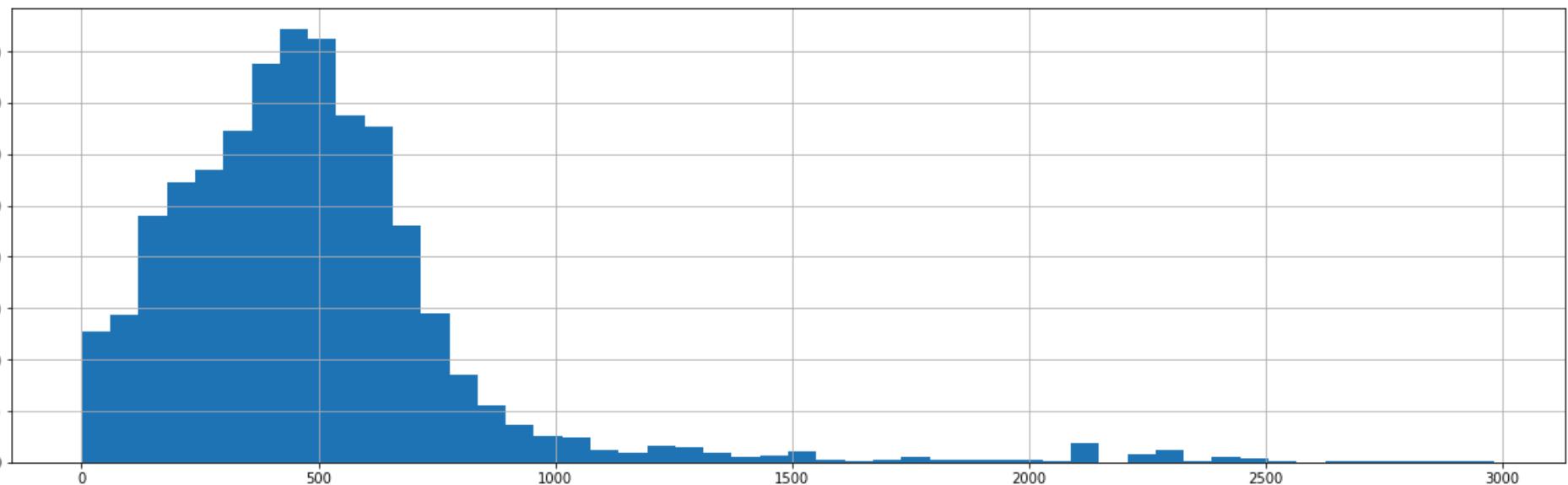
In [173...]

```
data[data['parks_nearest'] >= 0]['parks_nearest'].describe()
```

```
Out[173...]: count    8075.000000
          mean     489.505015
          std      337.380019
          min      1.000000
          25%    288.000000
          50%    454.000000
          75%    612.000000
          max    2984.000000
Name: parks_nearest, dtype: float64
```

```
In [174...]: data[data['parks_nearest'] >= 0]['parks_nearest'].hist(bins=50, figsize=(20,6))
```

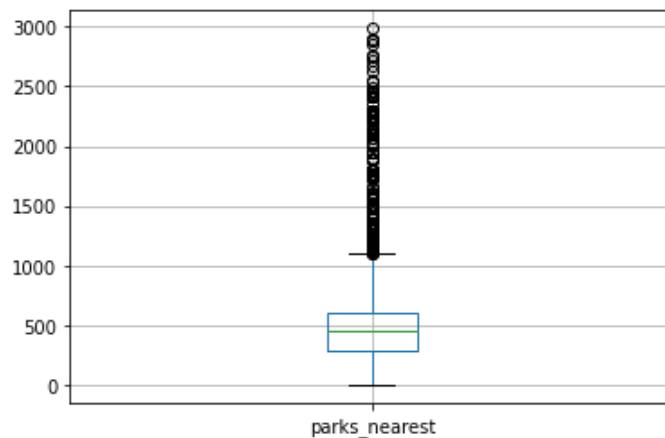
```
Out[174...]: <AxesSubplot:>
```



Распределение выглядит норм.

```
In [175...]: data[data['parks_nearest'] >= 0].boxplot(column='parks_nearest')
```

```
Out[175...]: <AxesSubplot:>
```



Отрезаем на 2500м.

```
In [176...]: data[(data['parks_nearest'] >= 0) & (data['parks_nearest'] < 2500)]['parks_nearest'].describe()
```

```
Out[176...]: count    8063.000000
mean      486.126504
std       326.012504
min       1.000000
25%     287.000000
50%     453.000000
75%     611.000000
max     2489.000000
Name: parks_nearest, dtype: float64
```

Итого. Данные есть только по 8000 квартир. Мы установили потолок на уровне 2500м. Медианное расстояние 450м. 75% объявлений в пределах 611м.

Изучим, как быстро продавались квартиры

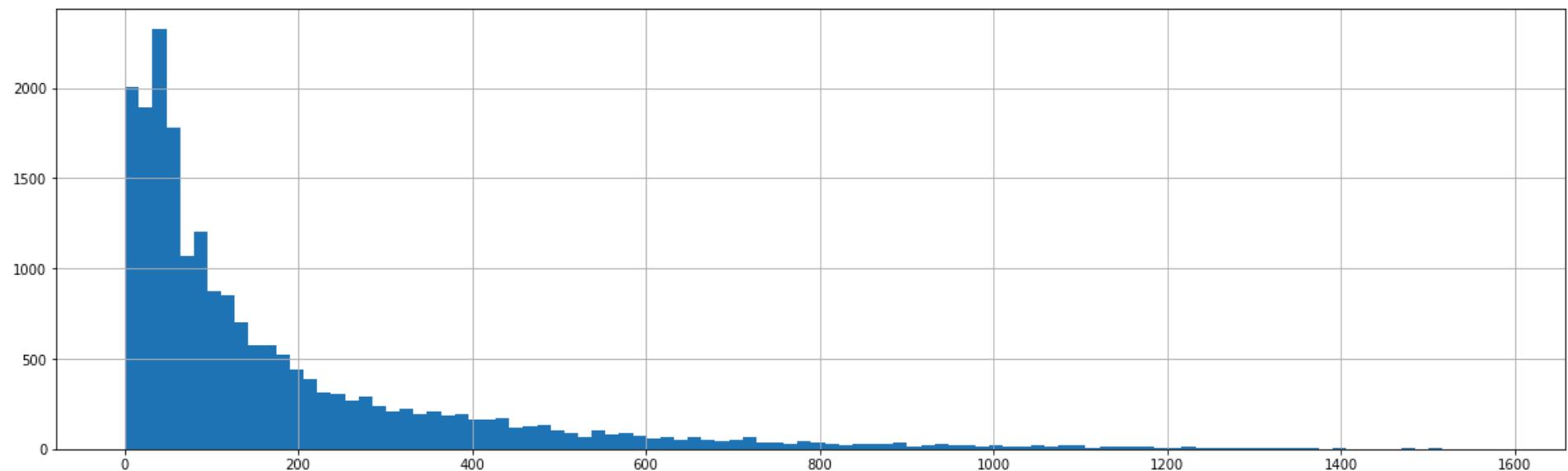
```
In [177...]: data['days_exposition'].describe()
```

```
Out[177...]: count    20518.000000
mean      180.888634
std       219.727988
min       1.000000
25%     45.000000
50%     95.000000
75%    232.000000
max     1580.000000
Name: days_exposition, dtype: float64
```

In [178...]

```
data['days_exposition'].hist(bins=100, figsize=(20, 6))
```

Out[178...]: <AxesSubplot:>

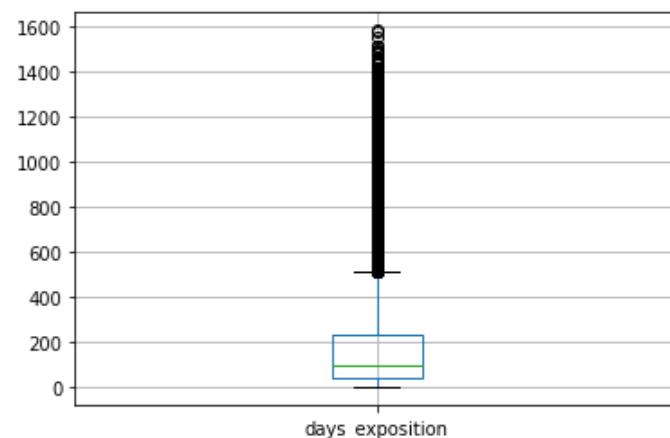


Распределение похоже на логнормальное.

In [179...]

```
data.boxplot(column='days_exposition')
```

Out[179...]: <AxesSubplot:>



Ящик говорит мне, что до 1400 значений ещё достаточно, а потом совсем мало.

In [180...]: `data[data['days_exposition'] < 1400]['days_exposition'].describe()`

Out[180...]:

	count	mean	std	min	25%	50%	75%	max
Name:	days_exposition	dtype:	float64					

```
count    20499.000000
mean     179.684277
std      216.232115
min      1.000000
25%     45.000000
50%     95.000000
75%    231.000000
max    1396.000000
```

Денис говорит мне, что можно не искать конец верхнего уса, а просто прибавить к третему квартилю 1.5 межквартильных размаха. Так и сделаем. $(232-95) * 1.5 + 232 = 437.5$

Итого. Обычное время продажи - 95 дней.

Среднее - 181 день.

Быстрая продажа - 45 дней и меньше.

Долгая - 232 дня и больше.

Необычно долгая - 438 дней и больше.

Давайте теперь уже избавимся от редких и выбивающихся значений, которые мы определили ранее.

In [181...]: `data['total_area'].count()`

Out[181...]: 23699

In [182...]: `data_new = data.query('total_area < 200 and total_area >= 20')`

In [183...]: `data_new['total_area'].count()`

Out[183...]: 23441

In [184...]: `data_new = data_new.query(
 '(living_area < 125 and living_area >= 10) or (living_area.isna())')`

```
In [185...]: data_new['total_area'].count()
```

```
Out[185...]: 23383
```

```
In [186...]: data_new = data_new.query(  
    '(kitchen_area <= 50 and kitchen_area >= 5) or (kitchen_area.isna())')
```

```
In [187...]: data_new['total_area'].count()
```

```
Out[187...]: 23255
```

```
In [188...]: data_new = data_new.query('price_kk < 40')
```

```
In [189...]: data_new['total_area'].count()
```

```
Out[189...]: 23174
```

```
In [190...]: data_new = data_new.query('rooms < 10')
```

```
In [191...]: data_new['total_area'].count()
```

```
Out[191...]: 23174
```

Забавно

```
In [192...]: data_new = data_new.query('ceiling_height <= 4 or ceiling_height.isna()')
```

```
In [193...]: data_new['total_area'].count()
```

```
Out[193...]: 23138
```

```
In [194...]: data_new = data_new.query('floors_total < 28')
```

```
In [195...]: data_new['total_area'].count()
```

```
Out[195... 23084
```

```
In [196... data_new = data_new.query('centers_range <= 37000 or centers_range.isna()')
```

```
In [197... data_new['total_area'].count()
```

```
Out[197... 22776
```

```
In [198... data_new = data_new.query('parks_nearest <= 2500 or parks_nearest.isna()')
```

```
In [199... data_new['total_area'].count()
```

```
Out[199... 22764
```

```
In [200... data_new = data_new.query('days_exposition <= 1400 or days_exposition.isna()')
```

```
In [201... data_new['total_area'].count()
```

```
Out[201... 22746
```

Мы потеряли меньше тысячи строк. Это ли не прекрасно?

Определите факторы, которые больше всего влияют на общую (полную) стоимость объекта.

Изучите, зависит ли цена от:

- общей площади;
- жилой площади;
- площади кухни;
- количества комнат;
- этажа, на котором расположена квартира (первый, последний, другой);
- даты размещения (день недели, месяц, год).

Постройте графики, которые покажут зависимость цены от указанных выше параметров. Для подготовки данных перед визуализацией вы можете использовать сводные таблицы.

In [202...]

data

Out[202...]

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	studio	...	ponds_around3000	ponds_nearest	days_expo
0	20	13000000	108.00	2019-03-07	3	2.70	16	51.0	8	False	...	2.0	755.0	
1	7	3350000	40.40	2018-12-04	1	NaN	11	18.6	1	False	...	0.0	-1.0	
2	10	5196000	56.00	2015-08-20	2	NaN	5	34.3	4	False	...	2.0	574.0	
3	0	64900000	159.00	2015-07-24	3	NaN	14	NaN	9	False	...	3.0	234.0	
4	2	10000000	100.00	2018-06-19	2	3.03	14	32.0	13	False	...	1.0	48.0	
...	
23694	9	9700000	133.81	2017-03-21	3	3.70	5	73.3	3	False	...	3.0	381.0	
23695	14	3100000	59.00	2018-01-15	3	NaN	5	38.0	4	False	...	NaN	NaN	
23696	18	2500000	56.70	2018-02-11	2	NaN	3	29.7	1	False	...	NaN	NaN	
23697	13	11475000	76.75	2017-03-28	2	3.00	17	NaN	12	False	...	3.0	196.0	
23698	4	1350000	32.30	2017-07-21	1	2.50	5	12.3	1	False	...	NaN	NaN	

23699 rows × 28 columns

In [204...]

```
data_price = data_new.loc[:, ['price_kk', 'total_area', 'living_area', 'kitchen_area',
                             'rooms', 'floor_type', 'weekday', 'month', 'year']]
```

In [205...]

data_price.head()

Out[205...]

	price_kk	total_area	living_area	kitchen_area	rooms	floor_type	weekday	month	year
0	13.000	108.0	51.0	25.0	3	другой	четверг	март	2019
1	3.350	40.4	18.6	11.0	1	первый	вторник	декабрь	2018
2	5.196	56.0	34.3	8.3	2	другой	четверг	август	2015
4	10.000	100.0	32.0	41.0	2	другой	вторник	июнь	2018

	price_kk	total_area	living_area	kitchen_area	rooms	floor_type	weekday	month	year
5	2.890	30.4	14.4	9.1	1	другой	понедельник	сентябрь	2018

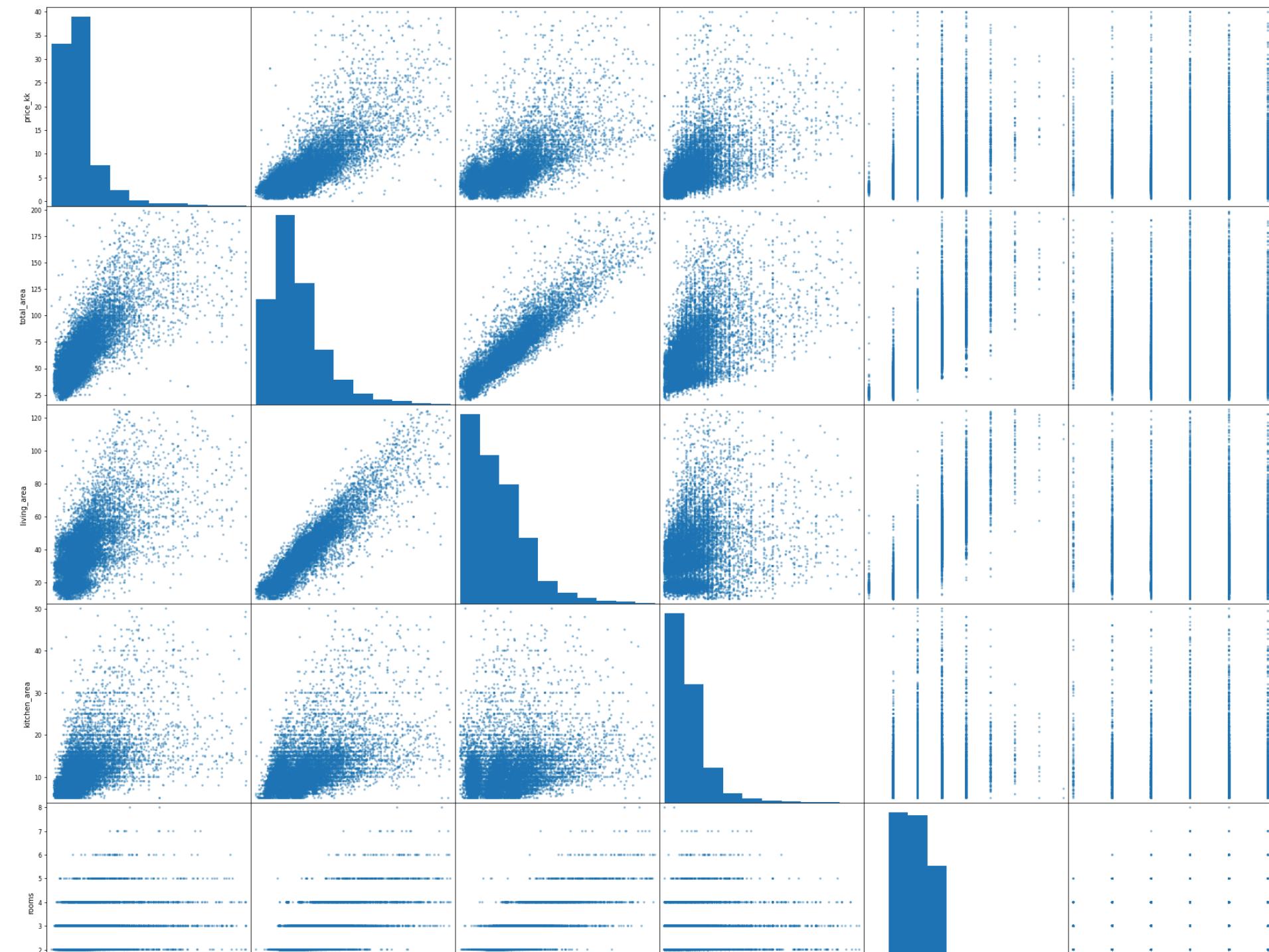
Построим матрицу диаграмм рассеяния.

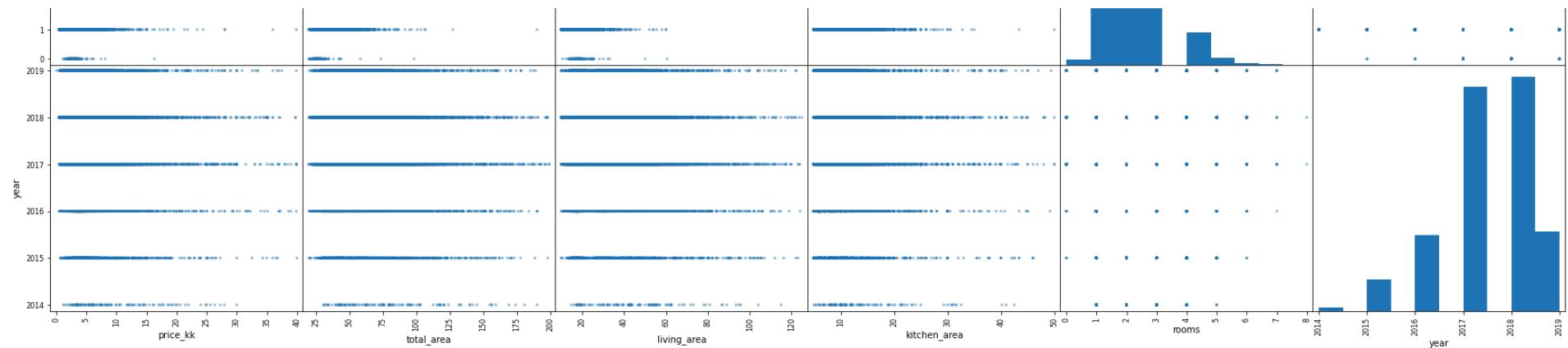
In [206...]

```
pd.plotting.scatter_matrix(data_price, figsize=(30, 30))
```

Out[206...]

```
array([[<AxesSubplot:xlabel='price_kk', ylabel='price_kk'>,
       <AxesSubplot:xlabel='total_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='living_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='rooms', ylabel='price_kk'>,
       <AxesSubplot:xlabel='year', ylabel='price_kk'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='total_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='total_area'>,
       <AxesSubplot:xlabel='year', ylabel='total_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='living_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='living_area'>,
       <AxesSubplot:xlabel='year', ylabel='living_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='year', ylabel='kitchen_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='rooms'>,
       <AxesSubplot:xlabel='total_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='living_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='rooms', ylabel='rooms'>,
       <AxesSubplot:xlabel='year', ylabel='rooms'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='year'>,
       <AxesSubplot:xlabel='total_area', ylabel='year'>,
       <AxesSubplot:xlabel='living_area', ylabel='year'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='year'>,
       <AxesSubplot:xlabel='rooms', ylabel='year'>,
       <AxesSubplot:xlabel='year', ylabel='year'>]], dtype=object)
```





Ну ёперый театр.

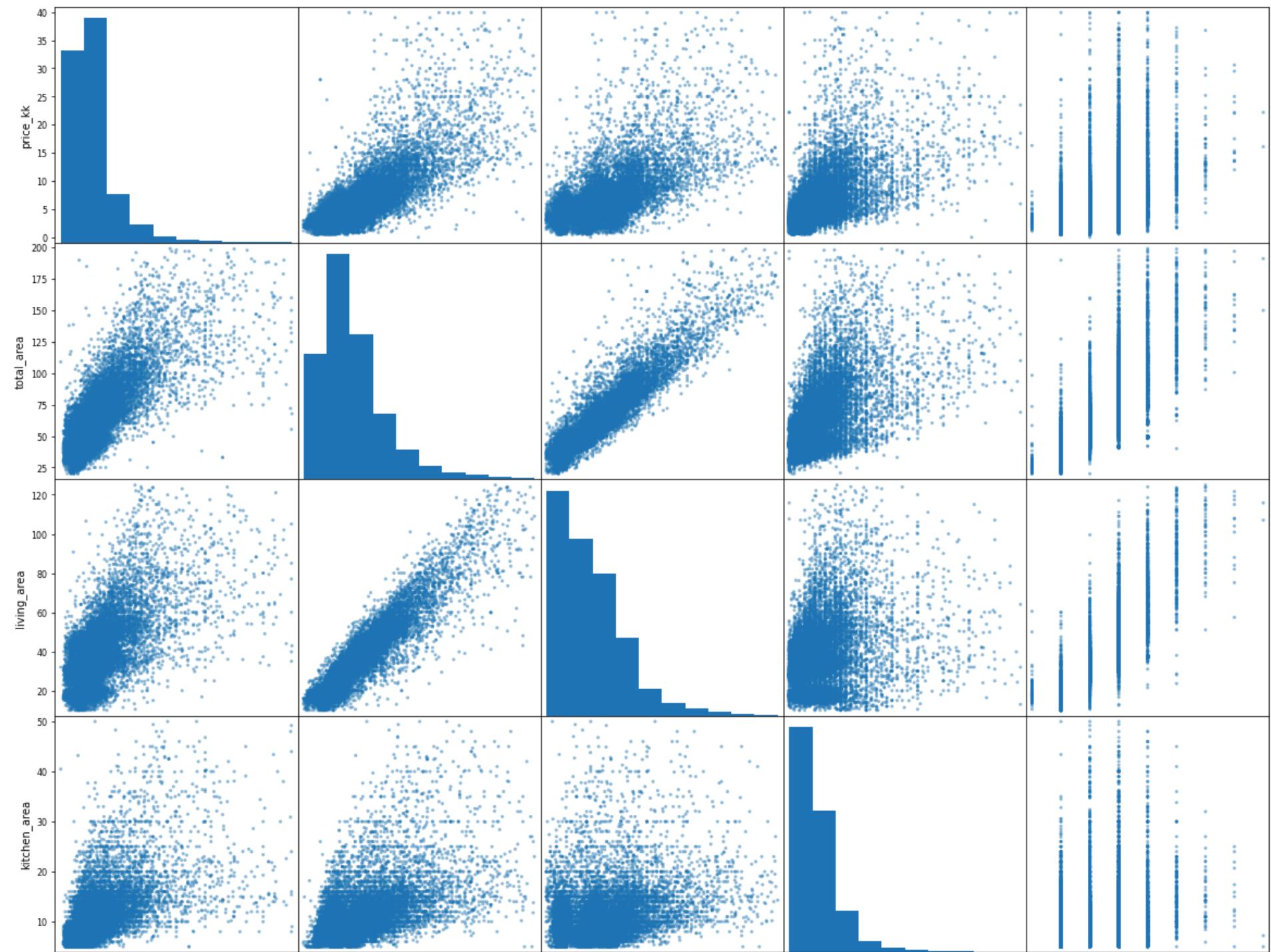
Ладно, меня уже так просто не сломать.

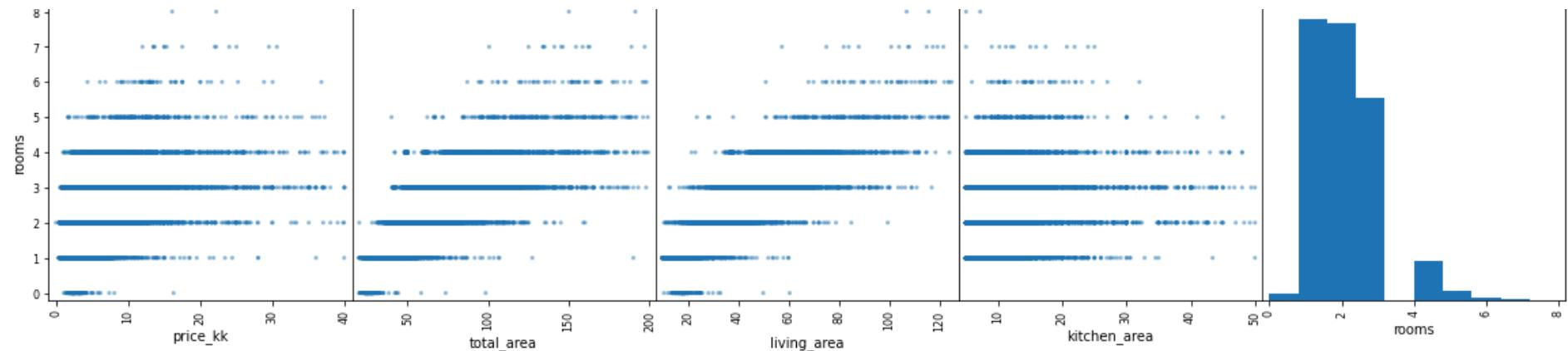
In [207...]

```
pd.plotting.scatter_matrix(data_new[['price_kk', 'total_area', 'living_area', 'kitchen_area', 'rooms', 'floor_type']]
                           , figsize=(20, 20))
```

Out[207...]

```
array([[<AxesSubplot:xlabel='price_kk', ylabel='price_kk'>,
       <AxesSubplot:xlabel='total_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='living_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='price_kk'>,
       <AxesSubplot:xlabel='rooms', ylabel='price_kk'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='total_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='total_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='total_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='living_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='living_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='living_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='total_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='living_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='kitchen_area'>,
       <AxesSubplot:xlabel='rooms', ylabel='kitchen_area'>],
      [<AxesSubplot:xlabel='price_kk', ylabel='rooms'>,
       <AxesSubplot:xlabel='total_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='living_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='kitchen_area', ylabel='rooms'>,
       <AxesSubplot:xlabel='rooms', ylabel='rooms'>]], dtype=object)
```

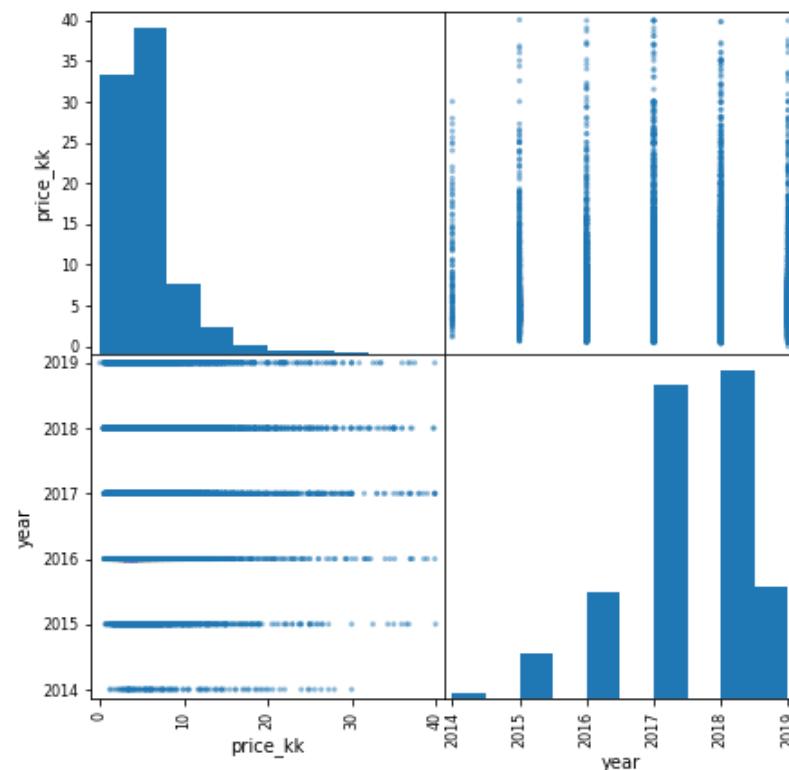




Блин, да, на что я надеялся, включая сюда floor_type, интересно. Пока самая сильная связь с total_area. Собственно, со всеми включенными параметрами есть явная связь. С rooms тоже явно есть, но метод визуализации не самый удачный.

```
In [258]: pd.plotting.scatter_matrix(data_new[['price_kk', 'weekday', 'month', 'year']], figsize=(7, 7))
```

```
Out[258]: array([[<AxesSubplot:xlabel='price_kk', ylabel='price_kk'>,
   <AxesSubplot:xlabel='year', ylabel='price_kk'>],
  [<AxesSubplot:xlabel='price_kk', ylabel='year'>,
   <AxesSubplot:xlabel='year', ylabel='year'>]], dtype=object)
```



Ладно, не обязательно же всегда думать перед тем, как делаешь, да? Есть связь, years но зависимость нелинейная. Стабильный подъем до 2017-2018 годов и потом спад. Или же нет? Объявлений то сильно меньше. Поэтому и прокрашено плохо. И что нам с этим делать? Вот сейчас самое время подумать.

In [209...]

```
data_price.corr()
```

Out[209...]

	price_kk	total_area	living_area	kitchen_area	rooms	year
price_kk	1.000000	0.772594	0.660506	0.586235	0.477476	-0.030711
total_area	0.772594	1.000000	0.924413	0.555772	0.790745	-0.075678
living_area	0.660506	0.924413	1.000000	0.312948	0.878049	-0.064594
kitchen_area	0.586235	0.555772	0.312948	1.000000	0.181169	-0.048978
rooms	0.477476	0.790745	0.878049	0.181169	1.000000	-0.043643
year	-0.030711	-0.075678	-0.064594	-0.048978	-0.043643	1.000000

Как я и сказал, самая сильная связь с общей площадью (total_area). С годом в итоге очень слабая обратная связь. А выглядело всё иначе. То-то же. Сложно сказать, почему связь обратная. Может недвижимость подешевела. А может больше людей с менее дорогой жилплощадью стали её продавать.

Посмотрим связь с типом этажа.

In [210...]

```
data_new.pivot_table(index='floor_type', values=['price_kk'],
                     aggfunc=['median', 'mean', 'count']).reset_index()
```

Out[210...]

	floor_type	median	mean	count
		price_kk	price_kk	price_kk
0	другой	4.800	6.033823	16747
1	неизвестно	5.249	6.594444	81
2	первый	3.950	4.567777	2767
3	последний	4.250	5.487222	3151

Дороже всего стоят квартиры с неизвестным этажом. На втором месте другой. Первый оказался последним. Ха-ха-ха.

Посмотрим на дни недели.

In [211...]

```
data_new.pivot_table(index='weekday', values=['price_kk'],
                     aggfunc=['median', 'mean', 'count']).reset_index()
```

Out[211...]

	weekday	median	mean	count
		price_kk	price_kk	price_kk
0	воскресенье	4.55	5.666129	1629
1	вторник	4.70	5.741725	4027
2	понедельник	4.60	5.791299	3466
3	пятница	4.55	5.703069	3864
4	среда	4.68	5.894072	3811
5	суббота	4.55	5.596038	1853
6	четверг	4.60	5.912780	4096

Реже всего объявления добавляют на выходных. Это все выводы, которые нам следует сделать из этой таблицы.

Теперь посмотрим связь с месяцем.

In [212...]

```
data_new.pivot_table(index='month', values=['price_kk'],
                     aggfunc=['median', 'mean', 'count']).reset_index()
```

Out[212...]

	month	median	mean	count
		price_kk	price_kk	price_kk
0	август	4.560	5.857026	1683
1	апрель	4.780	5.784410	2282
2	декабрь	4.680	5.831071	1565
3	июль	4.600	5.754225	1623
4	июнь	4.400	5.597253	1694
5	май	4.500	5.812328	1230
6	март	4.650	5.725737	2477
7	ноябрь	4.650	5.900285	2275
8	октябрь	4.550	5.650210	2056
9	сентябрь	4.675	5.882764	1904
10	февраль	4.600	5.767907	2519
11	январь	4.600	5.845656	1438

Ох, проиндексировать бы это по порядку месяцев.

Ладно, в общем. Больше всего любят выставлять к весне. С февраля по апрель прям много, потом на мае резкий провал. Осень тоже плотная, с сентября по ноябрь. Ну и ещё, чем больше объявлений в месяце, тем немного выше цены. Интересно было бы это сравнить с тем, как покупают квартиры по месяцам.

Год тоже глянем напоследок.

In [259...]

```
data_new.pivot_table(index='year', values=['price_kk'],
                     aggfunc=['median', 'mean', 'count']).reset_index()
```

Out[259...]

	year	median		mean		count	
		price_kk	total_area	price_kk	total_area	price_kk	total_area
0	2014	7.477	75.000	9.723807	80.969160	119	119
1	2015	5.200	59.000	6.650679	65.076250	1104	1104

year	median		mean		count	
	price_kk	total_area	price_kk	total_area	price_kk	total_area
2 2016	4.500	53.000	5.807142	59.157629	2661	2661
3 2017	4.500	51.135	5.671164	57.660901	7860	7860
4 2018	4.565	50.400	5.589891	56.201360	8218	8218
5 2019	5.010	51.400	6.123026	56.950841	2784	2784

В 2014 слишком мало данных, не берём в расчёт. Я не хочу описывать всё. Скажу так. У средней цены до 2019 явное падение. Она потянула за собой медиану. Скорее всего и правда стали продавать больше недорогих квартир и это сместило общую массу значений. Не похоже, чтобы жильё дешевело, скорее наоборот.

Итого. Цена за квартиру имеет высокую прямую зависимость от общей площади квартиры, а также среднюю прямую зависимость от числа комнат. Это значит, что чем выше площадь квартиры, тем выше цена за нее. Чем больше комнат, тем выше цена. Также если квартира находится на первом или последнем этаже, то ее стоимость ниже. Стоимость квартиры не зависит от дня недели публикации. Но немного зависит от месяца.

In [260...]

```
data_new.pivot_table(index='year', values=['price_kk', 'total_area'],
aggfunc=['median', 'mean', 'count']).reset_index()
```

Out[260...]

year	median		mean		count	
	price_kk	total_area	price_kk	total_area	price_kk	total_area
0 2014	7.477	75.000	9.723807	80.969160	119	119
1 2015	5.200	59.000	6.650679	65.076250	1104	1104
2 2016	4.500	53.000	5.807142	59.157629	2661	2661
3 2017	4.500	51.135	5.671164	57.660901	7860	7860
4 2018	4.565	50.400	5.589891	56.201360	8218	8218
5 2019	5.010	51.400	6.123026	56.950841	2784	2784

Квартиры всё меньше.

Цена квадратного метра в 10 населённых пунктах с наибольшим числом объявлений

Посчитайте среднюю цену одного квадратного метра в 10 населённых пунктах с наибольшим числом объявлений — постройте сводную таблицу с количеством объявлений и средней ценой квадратного метра для этих населенных пунктов. Выделите населённые пункты с самой высокой и низкой стоимостью квадратного метра.

In [214...]

```
data_new['locality_name'].value_counts().head(10)
```

Out[214...]

Санкт-Петербург	15232
Мурино	581
Кудрово	472
поселок Шушары	436
Всеволожск	396
Пушкин	356
Колпино	334
поселок Парголово	319
Гатчина	302
Выборг	235

Name: locality_name, dtype: int64

In [263...]

```
top_10_locs = ['Санкт-Петербург', 'Мурино', 'Кудрово', 'поселок Шушары', 'Всеволожск',
    'Пушкин', 'Колпино', 'поселок Парголово', 'Гатчина', 'Выборг']
```

In [266...]

```
data_new.query('locality_name == @top_10_locs').pivot_table(index='locality_name', values=('total_area', 'price_kk', 'square_cost'),
    aggfunc=['median', 'mean', 'count']).reset_index().sort_values(by=('median','square_cost'), ascending=False)
```

Out[266...]

	locality_name	median			mean			count		
		price_kk	square_cost	total_area	price_kk	square_cost	total_area	price_kk	square_cost	total_area
7	Санкт-Петербург	5.490	104251.845	54.30	6.928191	111375.337400	60.941864	15232	15232	15232
6	Пушкин	5.150	99865.715	53.00	6.092382	102913.258006	58.142697	356	356	356
4	Кудрово	3.890	95675.475	40.00	4.358689	95324.930508	46.395403	472	472	472
8	поселок Парголово	4.100	91516.560	42.90	4.455646	90209.996708	50.918401	319	319	319
5	Мурино	3.405	86119.400	37.70	3.691484	86012.196971	44.285663	581	581	581
9	поселок Шушары	3.950	76747.970	50.35	4.119553	78516.736972	54.018257	436	436	436
3	Колпино	3.600	74689.875	50.00	3.863353	75247.309491	52.367485	334	334	334
2	Гатчина	3.100	67925.850	45.60	3.503043	68846.422119	51.095629	302	302	302
0	Всеволожск	3.465	65789.470	53.55	3.791788	68621.569192	56.078359	396	396	396
1	Выборг	2.897	58158.320	50.20	3.243757	58140.851106	56.078979	235	235	235

Надо сравнить стоимость с площадью. Всё совпадает с установленным порядком. Питер самый дорогой, Выборг самый дешёвый.

Стоимость квартир в Питере по удалённости от центра.

Ранее вы посчитали расстояние до центра в километрах. Теперь выделите квартиры в Санкт-Петербурге с помощью столбца `locality_name` и вычислите их среднюю стоимость на разном удалении от центра. Учитывайте каждый километр расстояния: узнайте среднюю цену квартир в одном километре от центра, в двух и так далее. Опишите, как стоимость объектов зависит от расстояния до центра города — постройте график изменения средней цены для каждого километра от центра Петербурга.

In [217...]

```
data_spb = data_new.query('locality_name == "Санкт-Петербург"')
```

In [220...]

```
data_spb.pivot_table(index='centers_range_km', values='price_kk', aggfunc='mean').sort_values(by='centers_range_km')
```

Out[220...]

price_kk

centers_range_km	price_kk
-0.0	11.208038
1.0	12.079641
2.0	11.135828
3.0	9.681370
4.0	10.570031
5.0	11.017600
6.0	10.190862
7.0	10.506183
8.0	8.847796
9.0	6.858223
10.0	6.384820
11.0	6.122256
12.0	5.774808
13.0	6.062549
14.0	5.597547
15.0	5.789894
16.0	5.372317
17.0	5.184552
18.0	4.868167

price_kk**centers_range_km**

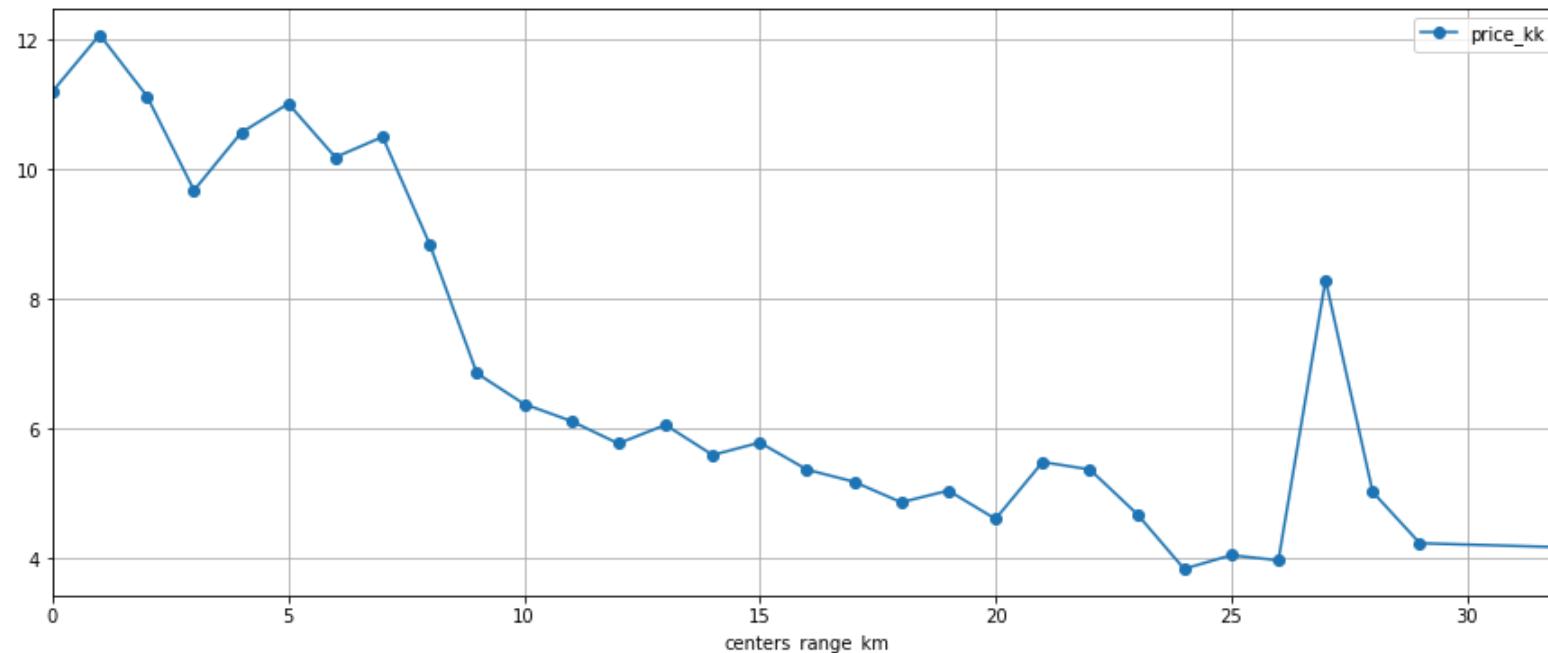
19.0	5.048558
20.0	4.611579
21.0	5.489379
22.0	5.374091
23.0	4.689843
24.0	3.841296
25.0	4.052846
26.0	3.973038
27.0	8.300000
28.0	5.026429
29.0	4.236667
32.0	4.175000

In [226...]

```
data_spb.pivot_table(index='centers_range_km', values='price_kk',
                      aggfunc='mean').plot(y='price_kk', style='o-', xlim=(0, 32), grid=True, figsize=(15, 6))
```

Out[226...]

<AxesSubplot:xlabel='centers_range_km'>



Оказывается, оси x нельзя передавать индекс. Во всяком случае без reset_index. Зато он берёт его сам по умолчанию.

По графику мы видим, что в центре средняя цена держится на высоком уровне, хоть и колеблется от 12 до 10 млн. Колебания могут быть связаны с тем, что мы обрезали очень дорогие квартиры, что сместило среднее в центре. После 7 км от центра начинается резкий спад, но после 9 км он замедляется и становится плавным и стабильным. Также мы наблюдаем аномальный пик цены на точке в 27 км. Давайте изучим его.

In [267...]

```
data_spb[data_spb['centers_range_km'] == 27][['last_price', 'total_area', 'living_area',
                                              'kitchen_area', 'rooms', 'square_cost']]
```

Out[267...]

	last_price	total_area	living_area	kitchen_area	rooms	square_cost
748	14350000	74.0	30.0	25.0	2	193918.92
5961	2250000	32.0	16.5	7.0	1	70312.50

Всего 2 квартиры. Одна дешёвая, а вторая ну очень дорогая. Особенно для такой удалённости. Дороже, чем средняя квартира в центре. Это статистически нормально. Она большая, в ней 2 комнаты и огромная кухня (возможно кухня-зал/гостиная). Скорее всего очень хорошая квартира. Возможно в дорогом доме - цена квадратного метра тоже вышла почти в 2 раза выше средней по Питеру. В общем, просто небольшая аномалия, ничего такого.

Итого. В центре средняя цена держится на высоком уровне, хоть и колеблется от 12 до 10 млн. После 7 км от центра резкий спад, до 6,4 млн на отметке в 9 км. Затем спад замедляется и становится плавным и стабильным. К отметке в 32 км средняя цена постепенно опускается до 4 млн.

Общий вывод

Стоимость квартиры существенно зависит от следующих параметров:

- **Площадь.** В первую очередь **общая площадь**, но также влияет и жилая площадь с площадью кухни (но мы не пытались выяснить независимость этих параметров от общей площади).
- **Близость к центру** города. В Питере самые дорогие квартиры в пределах 7 км от центра.
- **Количество комнат** (но тут снова может влиять то, что чем больше комнат, тем больше площадь).
- **Этаж.** Самые дорогие этажи промежуточные. Потом последние. Самые дешёвые - квартиры на первом этаже.
- **Сезон** публикации объявления. С **февраля по апрель** и с **сентября по ноябрь** публикуется больше объявлений и их средний ценник **немного** выше.
- **Время.** То есть год публикации объявления. Это спорный момент, но я делаю вывод, что с годами стоимость жилья **медленно** растёт.

Чек-лист готовности проекта

Поставьте 'x' в выполненных пунктах. Далее нажмите Shift+Enter.

- [x] открыт файл
- [x] файлы изучены (выведены первые строки, метод `info()`, гистограммы и т.д.)
- [x] определены пропущенные значения
- [x] заполнены пропущенные значения там, где это возможно
- [x] есть пояснение, какие пропущенные значения обнаружены
- [x] изменены типы данных
- [x] есть пояснение, в каких столбцах изменены типы и почему
- [x] устранены неявные дубликаты в названиях населённых пунктов
- [x] устранены редкие и выбивающиеся значения (аномалии) во всех столбцах
- [x] посчитано и добавлено в таблицу: цена одного квадратного метра
- [x] посчитано и добавлено в таблицу: день публикации объявления (0 - понедельник, 1 - вторник и т.д.)
- [x] посчитано и добавлено в таблицу: месяц публикации объявления
- [x] посчитано и добавлено в таблицу: год публикации объявления
- [x] посчитано и добавлено в таблицу: тип этажа квартиры (значения — «первый», «последний», «другой»)
- [x] посчитано и добавлено в таблицу: расстояние в км до центра города
- [x] изучены и описаны следующие параметры:

- общая площадь;
 - жилая площадь;
 - площадь кухни;
 - цена объекта;
 - количество комнат;
 - высота потолков;
 - этаж квартиры;
 - тип этажа квартиры («первый», «последний», «другой»);
 - общее количество этажей в доме;
 - расстояние до центра города в метрах;
 - расстояние до ближайшего аэропорта;
 - расстояние до ближайшего парка;
 - день и месяц публикации объявления
- [x] построены гистограммы для каждого параметра
 - [x] выполнено задание: "Изучите, как быстро продавались квартиры (столбец days_exposition). Этот параметр показывает, сколько дней «висело» каждое объявление.
 - Постройте гистограмму.
 - Посчитайте среднее и медиану.
 - В ячейке типа markdown опишите, сколько обычно занимает продажа. Какие продажи можно считать быстрыми, а какие — необычно долгими?"
 - [x] выполнено задание: "Какие факторы больше всего влияют на общую (полную) стоимость объекта? Постройте графики, которые покажут зависимость цены от указанных ниже параметров. Для подготовки данных перед визуализацией вы можете использовать сводные таблицы."
 - общей площади;
 - жилой площади;
 - площади кухни;
 - количество комнат;
 - типа этажа, на котором расположена квартира (первый, последний, другой);
 - даты размещения (день недели, месяц, год);
 - [x] выполнено задание: "Посчитайте среднюю цену одного квадратного метра в 10 населенных пунктах с наибольшим числом объявлений. Выделите населенные пункты с самой высокой и низкой стоимостью квадратного метра. Эти данные можно найти по имени в столбце locality_name ."
 - [x] выполнено задание: "Ранее вы посчитали расстояние до центра в километрах. Теперь выделите квартиры в Санкт-Петербурге с помощью столбца locality_name и вычислите среднюю цену каждого километра. Опишите, как стоимость объектов зависит от расстояния до центра города."
 - [x] в каждом этапе есть промежуточные выводы
 - [x] есть общий вывод