

Исследование объявлений о продаже квартир

В нашем 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. Наша 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.

1 Изучение данных из файла

```
In [1]: 1 import pandas as pd
        2 import seaborn as sns
        3 import matplotlib.pyplot as plt
```

```
In [2]: 1 # отключаем некритичные уведомления
        2 import warnings
        3 warnings.filterwarnings('ignore')
        4 # показывать до 40ка колонок
        5 pd.set_option('display.max.columns', 40)
        6 # установка формата вывода на дисплей численных значений
        7 pd.options.display.float_format = '{:,.2f}'.format
        8 # размер графика по умолчанию
        9 plt.figure(figsize=(6, 5));
```

<Figure size 432x360 with 0 Axes>

```
In [3]: 1 try:
        2     data = pd.read_csv('/datasets/real_estate_data.csv', sep='\t') # Yandex pat
        3 except:
        4     data = pd.read_csv(r"C:\Users\eddyd\Downloads\Praktikum\Datas_Thried_Projec
```

Первые строки

```
In [4]: 1 data.head() # смотрим первые строки
```

Out[4]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living
0	20	13,000,000.00	108.00	2019-03-07T00:00:00	3	2.70	16.00	
1	7	3,350,000.00	40.40	2018-12-04T00:00:00	1	NaN	11.00	
2	10	5,196,000.00	56.00	2015-08-20T00:00:00	2	NaN	5.00	
3	0	64,900,000.00	159.00	2015-07-24T00:00:00	3	NaN	14.00	
4	2	10,000,000.00	100.00	2018-06-19T00:00:00	2	3.03	14.00	

```
In [5]: 1 data.info() # сводка по данным данных
```

```
<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
```

Информация о численных данных

```
In [6]: 1 data.describe() # сведения о численных данных в колонках
```

Out[6]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	1
count	23,699.00	23,699.00	23,699.00	23,699.00	14,504.00	23,613.00	21,796.00	23,69
mean	9.86	6,541,548.77	60.35	2.07	2.77	10.67	34.46	
std	5.68	10,887,013.27	35.65	1.08	1.26	6.60	22.03	
min	0.00	12,190.00	12.00	0.00	1.00	1.00	2.00	
25%	6.00	3,400,000.00	40.00	1.00	2.52	5.00	18.60	
50%	9.00	4,650,000.00	52.00	2.00	2.65	9.00	30.00	
75%	14.00	6,800,000.00	69.90	3.00	2.80	16.00	42.30	
max	50.00	763,000,000.00	900.00	19.00	100.00	60.00	409.70	3

Есть ли пропуски значений

```
In [7]: 1 data.isna().sum() # смотрим пропуски
```

```
Out[7]: total_images      0
        last_price       0
        total_area       0
        first_day_exposition  0
        rooms            0
        ceiling_height    9195
        floors_total      86
        living_area       1903
        floor             0
        is_apartment      20924
        studio            0
        open_plan         0
        kitchen_area      2278
        balcony          11519
        locality_name     49
        airports_nearest  5542
        cityCenters_nearest 5519
        parks_around3000  5518
        parks_nearest     15620
        ponds_around3000  5518
        ponds_nearest     14589
        days_exposition   3181
        dtype: int64
```

Доли пропусков

```
In [8]: 1 round(data.isna().sum() * 100 / len(data), 2) # доля пропущенных значений
```

```
Out[8]: total_images      0.00
        last_price       0.00
        total_area       0.00
        first_day_exposition  0.00
        rooms            0.00
        ceiling_height    38.80
        floors_total      0.36
        living_area       8.03
        floor             0.00
        is_apartment      88.29
        studio            0.00
        open_plan         0.00
        kitchen_area      9.61
        balcony          48.61
        locality_name     0.21
        airports_nearest  23.38
        cityCenters_nearest 23.29
        parks_around3000  23.28
        parks_nearest     65.91
        ponds_around3000  23.28
        ponds_nearest     61.56
        days_exposition   13.42
        dtype: float64
```

1.1 Дубликаты

```
In [9]: 1 data.duplicated().sum()
```

```
Out[9]: 0
```

Дубликатов **нет**

1.2 Вывод

2 Описание данных

- `airports_nearest` — расстояние до ближайшего аэропорта в метрах (м) есть пропуски в данных. примерно в четверти из всего массива интересно посмотреть, какова корреляция с другими колонками
- `balcony` — число балконов есть пропуски значений (около половины от всей базы) и есть значения "0" Считаю, что пропуски значений в количестве балконов могут означать отсутствие балкона. то есть присвоить значение "0" Таким образом, 50% с пропусками + 25% со значением "0" получение очень много квартир без балконов
- `ceiling_height` — высота потолков (м) очень много пропусков. почти 40% от общей массы значений считаю, что высота потолков указывается в основном тогда, когда она выгодно отличается от обычного значения. поэтому можно подставить высоту потолков значением, наиболее часто встречающимся также есть странные значения 1м 100м следует проанализировать эти значения и решить, как с ними поступить
- `cityCenters_nearest` — расстояние до центра города (м) примерно столько же пропусков, как в `airports_nearest` проверить, есть ли соответствие или пропуски не согласованы друг с другом. так как у нас в данных нет адресов, то мы не сможем при всём желании самостоятельно рассчитать этот параметр
- `days_exposition` — сколько дней было размещено объявление (от публикации до снятия) странно, что в этом показателе есть пропуски. предлагаю проверить связку этого параметра с **`first_day_exposition`** и рассчитать самостоятельно недостающие данные
- `first_day_exposition` — дата публикации прекрасно! пропусков нет. данные пришли в строковом формате. 2018-12-04T00:00:00 переведем в питоновский формат даты с сохранением в новый столбец
- `floor` — этаж супер! пропусков нет нет странных значений небоскрёбы есть в городе (33 этаж)
- `floors_total` — всего этажей в доме есть чуть пропусков. меньше одного процента. проверим, может быть это просто одноэтажные дома? выведем эти значения в корреляции с этажом
- `is_apartment` — апартаменты (булев тип) есть немного пропусков. думаю, что пропуски - это значит, что это не апартаменты, а полноценные квартиры заменим пропуск на "не апартамент"
- `kitchen_area` — площадь кухни в квадратных метрах (м²) есть квартиры без данных в этом параметре может быть, это квартиры-студии? проверим
- `last_price` — цена на момент снятия с публикации хорошая новость: пропуски и странные значения отсутствуют но стоит проверить, нет ли странных значений в корреляции с метражом объектов
- `living_area` — жилая площадь в квадратных метрах (м²) есть и пропуски, и странные значения. вероятно, это связано с квартирами формата студия или ошибки заполнения данных проверим корреляцию со Студия, Общая площадь, Площадь кухни
- `locality_name` — название населённого пункта интересно. что даже здесь есть пропуски. странно, не правда ли? думаю, что этот параметр не существенен. у нас есть расстояния до знаковых мест
- `open_plan` — свободная планировка (булев тип) чудесно! пропусков нет
- `parks_around3000` — число парков в радиусе 3 км здесь примерно столько же пропусков, как и расстояниях до центра, до аэропорта кроме того, очень много строк со значением "0" основное значение - 1 бывают места с количеством парков 3 хочу туда!
- `parks_nearest` — расстояние до ближайшего парка (м) здесь больше пропусков. видимо, кроме тех строк, где нет данных по количеству парков поблизости, добавились ещё строки с пропусками
- `ponds_around3000` — число водоёмов в радиусе 3 км здесь примерно столько же пропусков, как и расстояниях до центра, до аэропорта кроме того, очень много строк со значением "0"
- `ponds_nearest` — расстояние до ближайшего водоёма (м) здесь больше пропусков. видимо, кроме тех строк, где нет данных по количеству парков поблизости, добавились ещё строки с

- пропусками
- rooms — число комнат пропусков нет, но есть "0" м б это студии? проверим...
- studio — квартира-студия (булев тип) ответственно подошли к заполнению этого параметра. пропусков нет
- total_area — площадь квартиры в квадратных метрах (м²) пожалуй, самый полно заполненный параметр. ни одного пропуска и ни одного странного значения!
- total_images — число фотографий квартиры в объявлении встречаются объявления без фотографий. это нормальная практика. к тому же, эта колонка нам навряд ли понадобится для ответов на поставленные перед исследованием вопросы.

3 Предобработка данных

3.1 airports_nearest

все строки с пропусками значений

```
In [10]: 1 data[data['airports_nearest'].isna()].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5542 entries, 5 to 23698
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   total_images                          5542 non-null   int64
1   last_price                           5542 non-null   float64
2   total_area                           5542 non-null   float64
3   first_day_exposition                 5542 non-null   object
4   rooms                                5542 non-null   int64
5   ceiling_height                       2964 non-null   float64
6   floors_total                         5532 non-null   float64
7   living_area                          4917 non-null   float64
8   floor                                5542 non-null   int64
9   is_apartment                         631 non-null    object
10  studio                               5542 non-null   bool
11  open_plan                            5542 non-null   bool
12  kitchen_area                         4825 non-null   float64
13  balcony                             2825 non-null   float64
14  locality_name                        5534 non-null   object
15  airports_nearest                     0 non-null      float64
16  cityCenters_nearest                  24 non-null     float64
17  parks_around3000                     24 non-null     float64
18  parks_nearest                        20 non-null     float64
19  ponds_around3000                     24 non-null     float64
20  ponds_nearest                        8 non-null      float64
21  days_exposition                      4675 non-null   float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 920.1+ KB
```

```
In [11]: 1 data[data['airports_nearest'].isna()].head(10)
```

Out[11]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area
5	10	2,890,000.00	30.40	2018-09-10T00:00:00	1	NaN	12.00	
8	20	2,900,000.00	33.16	2018-05-23T00:00:00	1	NaN	27.00	
12	10	3,890,000.00	54.00	2016-06-30T00:00:00	2	NaN	5.00	
22	20	5,000,000.00	58.00	2017-04-24T00:00:00	2	2.75	25.00	
30	12	2,200,000.00	32.80	2018-02-19T00:00:00	1	NaN	9.00	
37	10	1,990,000.00	45.80	2017-10-28T00:00:00	2	2.50	5.00	
38	10	3,150,000.00	40.00	2018-03-29T00:00:00	1	2.75	18.00	
47	17	3,600,000.00	56.10	2018-10-18T00:00:00	3	NaN	4.00	
60	3	2,740,000.00	35.00	2018-01-01T00:00:00	1	NaN	12.00	
62	0	4,800,000.00	78.60	2017-09-17T00:00:00	3	2.80	9.00	

из первых строк видно, что это - не Питер, а какие-то другие населённые пункты. сгруппируем данные по locality_name и подсчитаем количество строк

```
In [12]: 1 data[data['airports_nearest'].isna()].groupby(by='locality_name')['locality_name']
```

Out[12]:

locality_name	
Бокситогорск	16
Волосово	36
Волхов	111
Всеволожск	398
Выборг	237
...	
село Путилово	2
село Рождествено	3
село Русско-Высоцкое	9
село Старая Ладога	2
село Шум	1
Name: locality_name, Length: 344, dtype: int64	

В результате получили 344 названия. Выведем полный список

```
In [13]: 1 data[data['airports_nearest'].isna()][['locality_name']].unique()
```

```
Out[13]: array(['городской посёлок Янино-1', 'посёлок Мурино', 'Сертолово',  
               'деревня Кудрово', 'Коммунар',  
               'поселок городского типа Красный Бор', 'Гатчина', 'поселок Мурино',  
               'деревня Фёдоровское', 'Выборг', 'Кировск',  
               'деревня Новое Девяткино', 'Санкт-Петербург',  
               'посёлок городского типа Лебяжье',  
               'посёлок городского типа Сиверский', 'поселок Молодцово',  
               'поселок городского типа Кузьмоловский',  
               'садовое товарищество Новая Ропша', 'деревня Пикколово',  
               'Всеволожск', 'Волхов', 'Кингисепп', 'Приозерск',  
               'деревня Куттузи', 'посёлок Аннино',  
               'поселок городского типа Ефимовский', 'посёлок Плодовое',  
               'деревня Заклинье', 'поселок Торковичи', 'поселок Первомайское',  
               'Сясьстрой', 'деревня Старая', 'деревня Лесколово',  
               'посёлок Новый Свет', 'Сланцы', 'село Путилово', 'Ивангород',  
               'Мурино', 'Шлиссельбург', 'Никольское', 'Сосновый Бор',  
               'поселок Новый Свет', 'деревня Оржицы', 'деревня Кальтино',  
               'Кудрово', 'поселок Романовка', 'посёлок Бугры', 'поселок Бугры',  
               'поселок городского типа Рощино', 'Кириши', 'Луга', 'Волосово',  
               ...])
```

Вот так новость! Пропуски есть и в Санкт-Петербург. А я то думал, что только вне города пропущены эти сведения. Что же делать с этими пропусками? По сведениям от заказчика, эти данные сгенерированы автоматически. Внести их вручную не получится быстро. Их очень много... Значит, оставим "КАК ЕСТЬ". И при исследовании влияния этих данных строки с пропусками будем исключать из анализа.

3.2 cityCenters_nearest

все строки с пропусками значений

Из общей справки мы знаем, что количество пропусков значений в cityCenters_nearest похоже на количество пропусков в airports_nearest Проверим совпадение строк с пропусками в обоих столбцах

Пропуски в cityCenters_nearest

In [14]:

1 data[data['cityCenters_nearest'].isna()].info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5519 entries, 5 to 23698
Data columns (total 22 columns):
Column Non-Null Count Dtype

0 total_images 5519 non-null int64
1 last_price 5519 non-null float64
2 total_area 5519 non-null float64
3 first_day_exposition 5519 non-null object
4 rooms 5519 non-null int64
5 ceiling_height 2944 non-null float64
6 floors_total 5509 non-null float64
7 living_area 4896 non-null float64
8 floor 5519 non-null int64
9 is_apartment 628 non-null object
10 studio 5519 non-null bool
11 open_plan 5519 non-null bool
12 kitchen_area 4804 non-null float64
13 balcony 2809 non-null float64
14 locality_name 5511 non-null object
15 airports_nearest 1 non-null float64
16 cityCenters_nearest 0 non-null float64
17 parks_around3000 1 non-null float64
18 parks_nearest 1 non-null float64
19 ponds_around3000 1 non-null float64
20 ponds_nearest 0 non-null float64
21 days_exposition 4653 non-null float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 916.2+ KB

In [15]:

1 data[data['cityCenters_nearest'].isna()].head(10)

Out[15]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area
5	10	2,890,000.00	30.40	2018-09-10T00:00:00	1	NaN	12.00	
8	20	2,900,000.00	33.16	2018-05-23T00:00:00	1	NaN	27.00	
12	10	3,890,000.00	54.00	2016-06-30T00:00:00	2	NaN	5.00	
22	20	5,000,000.00	58.00	2017-04-24T00:00:00	2	2.75	25.00	
30	12	2,200,000.00	32.80	2018-02-19T00:00:00	1	NaN	9.00	
37	10	1,990,000.00	45.80	2017-10-28T00:00:00	2	2.50	5.00	
38	10	3,150,000.00	40.00	2018-03-29T00:00:00	1	2.75	18.00	
47	17	3,600,000.00	56.10	2018-10-18T00:00:00	3	NaN	4.00	
60	3	2,740,000.00	35.00	2018-01-01T00:00:00	1	NaN	12.00	
62	0	4,800,000.00	78.60	2017-09-17T00:00:00	3	2.80	9.00	

У нас есть 5519 строк с пропусками в cityCenters_nearest А в airports_nearest 5542 пропуска Всего записей 23699 строк Значит, строк безз пропусков в обоих колонках ожидается


```
In [16]: 1 23699-5542
```

```
Out[16]: 18157
```

```
In [17]: 1 airports_and_cityCenters_nearest_drop_na = data.dropna(subset= (  
2     ['airports_nearest',  
3     'cityCenters_nearest',  
4     ]  
5 )  
6 )
```

Срез данных без пропусков в расстояниях

```
In [18]: 1 airports_and_cityCenters_nearest_drop_na.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 18156 entries, 0 to 23697  
Data columns (total 22 columns):  
#   Column                                Non-Null Count  Dtype  
---  -  
0   total_images                          18156 non-null  int64  
1   last_price                            18156 non-null  float64  
2   total_area                            18156 non-null  float64  
3   first_day_exposition                 18156 non-null  object  
4   rooms                                18156 non-null  int64  
5   ceiling_height                       11539 non-null  float64  
6   floors_total                         18080 non-null  float64  
7   living_area                          16878 non-null  float64  
8   floor                                18156 non-null  int64  
9   is_apartment                         2144 non-null   object  
10  studio                               18156 non-null  bool  
11  open_plan                            18156 non-null  bool  
12  kitchen_area                         16595 non-null  float64  
13  balcony                              9354 non-null   float64  
14  locality_name                        18115 non-null  object  
15  airports_nearest                     18156 non-null  float64  
16  cityCenters_nearest                  18156 non-null  float64  
17  parks_around3000                     18156 non-null  float64  
18  parks_nearest                        8058 non-null   float64  
19  ponds_around3000                     18156 non-null  float64  
20  ponds_nearest                        9102 non-null   float64  
21  days_exposition                      15843 non-null  float64  
dtypes: bool(2), float64(14), int64(3), object(3)  
memory usage: 2.9+ MB
```

3.2.1 ИТОГ

Гипотеза подтвердилась. Данные по расстояниям до аэропорта и до центра города пропущены практически в одних и тех же строках. Срез данных без этих пропусков `airports_and_cityCenters_nearest_drop_na` содержит 18156 строк.

3.3 parks_around3000 и ponds_around3000

пропуски значений составляют ~24% это совпадает по количеству строк с пропусками в `airports_and_cityCenters_nearest` Проверим, сохранились ли эти пропуски в `airports_and_cityCenters_nearest_drop_na`

In [19]:

▶

1 airports_and_cityCenters_nearest_drop_na[airports_and_cityCenters_nearest_drop_

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   total_images                          0 non-null      int64
1   last_price                            0 non-null      float64
2   total_area                            0 non-null      float64
3   first_day_exposition                 0 non-null      object
4   rooms                                0 non-null      int64
5   ceiling_height                       0 non-null      float64
6   floors_total                         0 non-null      float64
7   living_area                          0 non-null      float64
8   floor                                0 non-null      int64
9   is_apartment                         0 non-null      object
10  studio                               0 non-null      bool
11  open_plan                            0 non-null      bool
12  kitchen_area                         0 non-null      float64
13  balcony                              0 non-null      float64
14  locality_name                        0 non-null      object
15  airports_nearest                     0 non-null      float64
16  cityCenters_nearest                  0 non-null      float64
17  parks_around3000                     0 non-null      float64
18  parks_nearest                        0 non-null      float64
19  ponds_around3000                     0 non-null      float64
20  ponds_nearest                        0 non-null      float64
21  days_exposition                      0 non-null      float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 0.0+ bytes
```

```
In [20]: ▶ 1 airports_and_cityCenters_nearest_drop_na[airports_and_cityCenters_nearest_drop_
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   total_images                          0 non-null      int64
1   last_price                            0 non-null      float64
2   total_area                            0 non-null      float64
3   first_day_exposition                  0 non-null      object
4   rooms                                 0 non-null      int64
5   ceiling_height                        0 non-null      float64
6   floors_total                          0 non-null      float64
7   living_area                           0 non-null      float64
8   floor                                 0 non-null      int64
9   is_apartment                          0 non-null      object
10  studio                                0 non-null      bool
11  open_plan                             0 non-null      bool
12  kitchen_area                          0 non-null      float64
13  balcony                               0 non-null      float64
14  locality_name                         0 non-null      object
15  airports_nearest                      0 non-null      float64
16  cityCenters_nearest                   0 non-null      float64
17  parks_around3000                      0 non-null      float64
18  parks_nearest                         0 non-null      float64
19  ponds_around3000                      0 non-null      float64
20  ponds_nearest                         0 non-null      float64
21  days_exposition                       0 non-null      float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 0.0+ bytes
```

3.3.1 Итак, в airports_and_cityCenters_nearest_drop_na нет пропусков в рассматриваемых колонках.

Можно использовать базу airports_and_cityCenters_nearest_drop_na и для анализа влияния на цену количества парков и прудов вблизи от объекта недвижимости

3.3.2 Заменять пропуски тоже не будем

3.4 parks_nearest и ponds_nearest

большое количество пропусков в данных выдвигаю гипотезу о том, что расстояния отсутствуют в строках с пропусками и в строках, где значение параметра равно "0". То есть не до чего измерять расстояние.

In [21]:

1 data[data['parks_nearest'].isna()].describe()

Out[21]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	
count	15,620.00	15,620.00	15,620.00	15,620.00	9,210.00	15,568.00	14,294.00	15,620
mean	9.69	5,243,464.04	55.98	1.96	2.72	11.26	31.67	
std	5.64	7,222,437.85	29.65	0.99	1.05	6.97	17.81	
min	0.00	430,000.00	13.00	0.00	1.20	1.00	2.00	
25%	6.00	3,093,750.00	39.00	1.00	2.50	5.00	18.00	
50%	9.00	4,150,000.00	49.90	2.00	2.60	9.00	29.30	
75%	14.00	5,800,000.00	64.60	3.00	2.75	16.00	40.00	
max	50.00	420,000,000.00	900.00	19.00	32.00	37.00	409.70	3

Итак, для всех строк с пропусками данных в parks_nearest значение parks_around3000 == 0 **Гипотеза подтвердилась.**

In [22]:

1 data[data['ponds_nearest'].isna()].describe()

Out[22]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	
count	14,589.00	14,589.00	14,589.00	14,589.00	8,772.00	14,552.00	13,381.00	14,589
mean	9.80	5,148,266.26	55.93	1.99	2.73	10.65	31.96	
std	5.61	4,956,858.23	28.61	1.00	1.04	6.70	17.98	
min	0.00	12,190.00	13.00	0.00	1.20	1.00	2.00	
25%	6.00	3,000,000.00	39.00	1.00	2.50	5.00	18.00	
50%	9.00	4,150,000.00	50.00	2.00	2.60	9.00	29.60	
75%	14.00	5,748,000.00	64.00	3.00	2.75	16.00	40.00	
max	50.00	150,000,000.00	590.00	16.00	32.00	52.00	409.00	2

Итак, для всех строк с пропусками данных в ponds_nearest значение ponds_around3000 == 0 **Гипотеза подтвердилась.**

3.4.1 Вывод

Для анализа влияния на цену близости парков и водоёмов следует исследовать только строки со значениями больше "0"

3.5 total_images

эта колонка не имеет значения для этого исследования

3.6 last_price

Сведения о данных:

```
In [23]: 1 data['last_price'].describe()
```

```
Out[23]: count      23,699.00
         mean      6,541,548.77
         std      10,887,013.27
         min       12,190.00
         25%      3,400,000.00
         50%      4,650,000.00
         75%      6,800,000.00
         max      763,000,000.00
         Name: last_price, dtype: float64
```

3.6.1 В данных отсутствуют пропуски и странные значения (≤ 0) на первый взгляд.

Возможные аномалии могут быть выявлены при последующем анализе в совокупности с прочими параметрами

3.7 total_area

из общей справки по пропускам, в этой колонке их нет общие сведения о данных:

```
In [24]: 1 data['total_area'].describe()
```

```
Out[24]: count      23,699.00
         mean         60.35
         std         35.65
         min         12.00
         25%         40.00
         50%         52.00
         75%         69.90
         max         900.00
         Name: total_area, dtype: float64
```

3.7.1 В данных отсутствуют пропуски и странные значения (≤ 0) на первый взгляд.

Возможные аномалии могут быть выявлены при последующем анализе в совокупности с прочими параметрами Есть прекрасный объект с огромной площадью

3.8 first_day_exposition

из общей справки по пропускам, в этой колонке их нет

- данные представлены в строковом формате.
- переводим их в питоновский формат с сохранением в этой же колонке

Исходный формат 2018-05-23T00:00:00

```
In [25]: 1 data['first_day_exposition'] = pd.to_datetime(data['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S')
```

```
In [26]: 1 data['first_day_exposition'].head()
```

```
Out[26]: 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]
```

```
In [27]: 1 data['first_day_exposition'].describe()
```

```
Out[27]: count                23699
         unique                1491
         top      2018-02-01 00:00:00
         freq                 368
         first    2014-11-27 00:00:00
         last     2019-05-03 00:00:00
         Name: first_day_exposition, dtype: object
```

3.8.1 Итог

Успешно даты переведены в правильный формат.

- Самая давняя дата - 2014 год
- Самая поздняя дата - 2019 год, май месяц
- Самая частая дата - 2018 год

3.9 rooms и ... потом studio

пропуски отсутствуют какие данные встречаются:

```
In [28]: 1 data['rooms'].describe()
```

```
Out[28]: count    23,699.00
         mean       2.07
         std        1.08
         min        0.00
         25%        1.00
         50%        2.00
         75%        3.00
         max       19.00
         Name: rooms, dtype: float64
```

0 комнат. Что это? Студии? ПРОВЕРИМ

```
In [29]: 1 data_zero_rooms = data[data['rooms'] == 0]
```

```
In [30]: 1 data_zero_rooms['studio'].value_counts()
```

```
Out[30]: True      138
         False     59
         Name: studio, dtype: int64
```

Есть объекты с количеством комнат "0", но не студии. А что же это такое?

```
In [31]: 1 data_zero_rooms[data_zero_rooms['studio'] == False]
```

Out[31]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_tot
349	4	2,320,000.00	25.00	2017-09-27	0	NaN	14.0
508	0	3,375,000.00	34.40	2017-03-28	0	NaN	26.0
780	9	2,600,000.00	26.10	2018-02-20	0	NaN	18.0
839	14	1,900,000.00	35.00	2017-04-14	0	2.70	5.0
946	5	2,200,000.00	23.00	2016-09-27	0	NaN	27.0
1574	0	2,200,000.00	22.00	2017-11-03	0	NaN	18.0
1625	7	1 980 000.00	23.98	2018-02-01	0	NaN	4.0

```
In [32]: 1 data_zero_rooms.describe()
```

Out[32]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floor_k
count	197.00	197.00	197.00	197.00	82.00	194.00	183.00	197.00
mean	7.78	3,337,724.11	29.28	0.00	3.10	16.90	18.87	8.58
std	5.08	5,046,021.69	25.86	0.00	2.70	6.93	6.55	6.29
min	0.00	945,750.00	15.50	0.00	2.50	3.00	2.00	1.00
25%	4.00	2,300,000.00	24.05	0.00	2.70	12.00	16.00	3.00
50%	7.00	2,700,000.00	26.05	0.00	2.75	17.00	18.00	7.00
75%	11.00	3,380,000.00	28.40	0.00	2.80	23.00	19.95	13.00
max	21.00	71,000,000.00	371.00	0.00	27.00	35.00	68.00	26.00

Итак

- у всех объектов с числом комнат "0", также отсутствует информация о площади кухни
- есть объекты малой площади
- есть объекты с большой площадью ТАк получается, что здесь есть и квартиры малой площади, которые можно отнести к классу Студия, и есть квартиры большой площади. Их можно отнести к классу "Свободная планировка"
- что даёт это пониманимание нам в контексте исследования? Проверим, какие значения "комнат" встречаются для "Студия"

```
In [33]: 1 data_studio_True = data[data['studio'] == True]
```

```
In [34]: 1 data_studio_True['rooms'].value_counts()
```

Out[34]: 0 138
1 11
Name: rooms, dtype: int64

3.9.1 Вывод

надо для всех "Студия" выставить значение "Комнат" == 0

```
In [35]: 1 data.loc[data['studio'] == True, 'rooms'] = 0
```

Проверка

```
In [36]: 1 data[data['studio'] == True]['rooms'].value_counts()
```

```
Out[36]: 0    149
         Name: rooms, dtype: int64
```

3.9.2 Готово

3.10 ceiling_height

много (~40%) строк с пропусками значений

- в предоставленном данных отсутствуют колонки с косвенными параметрами для определения высоты потолков. Это могли бы быть
- год постройки дома
- серия дома Поэтому оставляем пропуски значений как есть.
- сводка по данным

```
In [37]: 1 data['ceiling_height'].describe()
```

```
Out[37]: count    14,504.00
         mean         2.77
         std         1.26
         min         1.00
         25%         2.52
         50%         2.65
         75%         2.80
         max         100.00
         Name: ceiling_height, dtype: float64
```

Присутствуют странные значения "1" и "100"

- исследуем их

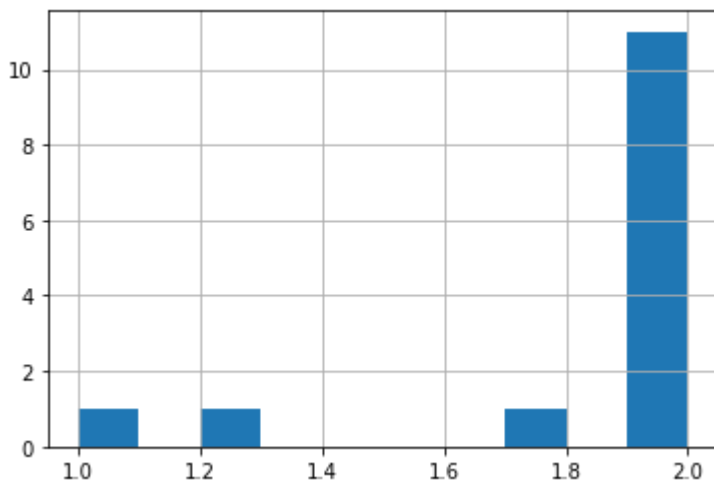
Высота потолков 1 метр

Посмотрим распределение значений высоты потолка до 2х метров включительно

```
In [38]: 1 data_ceiling_height_2 = data[data['ceiling_height'] <= 2]
```



```
In [39]: 1 data_ceiling_height_2['ceiling_height'].hist(color='C0');
```



1.80 - ужасно низкий потолок. Согласны? Интересно будет посмотреть на цены таких объектов.

Итак. Какие встречаются значения меньше 2,00?:

```
In [40]: 1 data_ceiling_height_2['ceiling_height'].unique()
```

```
Out[40]: array([2. , 1.2 , 1.75, 1.  ])
```

Это

- 1
- 1,2
- 1,75

2,00 примем за крайний вариант приемлемого значения высоты потолка. А вот значения меньше этой цифры заменим на NaN

- замену выполним в основной базе данных

```
In [41]: 1 import numpy as np # библиотека для применения операций с NaN
2
```

```
In [42]: 1 # Сначала присвою значение "0" всем ячейкам, в которых надо проставить NaN
2 data.loc[(data['ceiling_height'] < 2 ), 'ceiling_height'] = 0
```

Проверка

```
In [43]: 1 sorted(data.ceiling_height.unique())
```

```
Out[43]: [0.0,  
          2.0,  
          2.3,  
          2.4,  
          2.45,  
          2.46,  
          2.48,  
          2.53,  
          2.59,  
          2.62,  
          2.63,  
          2.7,  
          nan,  
          2.2,  
          2.25,  
          2.34,  
          2.47,  
          2.49,  
          2.5,  
          ~ ~]
```

Успешно!

- теперь заменим все "0" на NaN

```
In [44]: 1 data['ceiling_height'].replace(0, np.NaN, inplace = True)
```

Проверка

```
In [45]: 1 sorted(data.ceiling_height.unique())
```

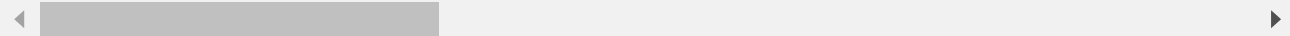
```
Out[45]: [2.0,  
          2.2,  
          2.3,  
          2.4,  
          2.45,  
          2.46,  
          2.48,  
          2.53,  
          2.59,  
          2.62,  
          2.63,  
          2.7,  
          nan,  
          2.25,  
          2.34,  
          2.47,  
          2.49,  
          2.5,  
          2.51,  
          ~ ~]
```

Интересно .что это за объект, высота потолка в котором 100 м

```
In [46]: 1 data[data['ceiling_height'] == 100 ]
```

Out[46]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total
22869	0	15,000,000.00	25.00	2018-07-25	1	100.00	5.00



Однокомнатная квартира средней площади на 5 этаже с пятью балконами и высотой потолков 100 метров)))

- заменю высоту на NaN

```
In [47]: 1 data['ceiling_height'].replace(100, np.NaN, inplace = True)
```

Посмотрим сводку по данным высоты потолка сейчас

```
In [48]: 1 data['ceiling_height'].describe()
```

```
Out[48]: count    14,500.00
mean         2.77
std          0.97
min          2.00
25%          2.52
50%          2.65
75%          2.80
max          32.00
Name: ceiling_height, dtype: float64
```

Можно смело считать, что высота потолка больше 3х метров - необычное явление.

- рассмотрим внимательнее

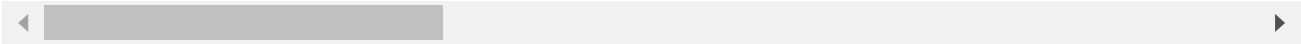
In [49]:

1 data[data['ceiling_height'] > 3].sort_values('ceiling_height')

Out[49]:

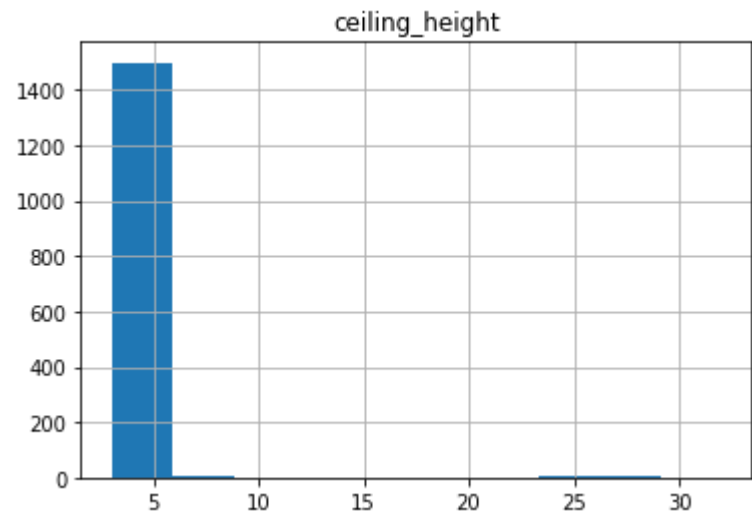
	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total
5892	2	5,200,000.00	79.10	2016-06-29	3	3.01	5.00
1351	14	8,500,000.00	66.20	2019-02-27	3	3.01	4.00
1606	2	11,390,000.00	68.80	2016-09-22	2	3.01	12.00
17837	6	35,900,000.00	110.00	2017-08-09	3	3.01	6.00
14694	20	31,750,000.00	124.50	2016-05-16	3	3.01	7.00
...
10773	8	3,800,000.00	58.00	2017-10-13	2	27.00	10.00
20478	11	8,000,000.00	45.00	2017-07-18	1	27.00	4.00
21377	19	4,900,000.00	42.00	2017-04-18	1	27.50	24.00
3148	14	2,900,000.00	75.00	2018-11-12	3	32.00	3.00
22336	19	9,999,000.00	92.40	2019-04-05	2	32.00	6.00

1528 rows × 22 columns



In [50]:

1 data[data['ceiling_height'] > 3].hist('ceiling_height');



In [51]:

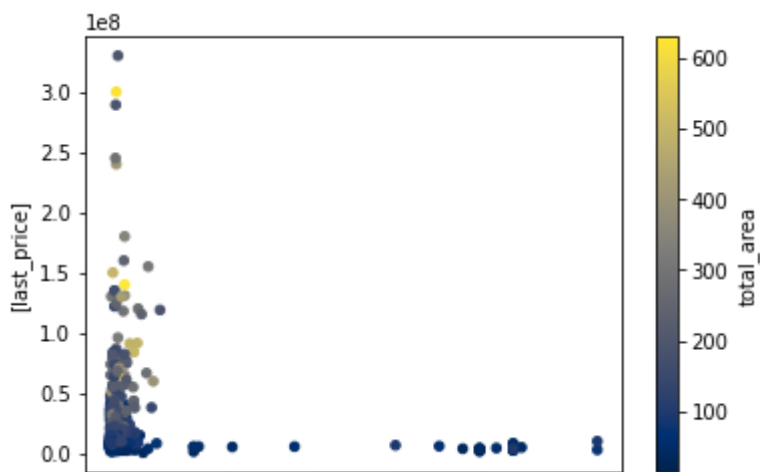
1 data_3_up = data[data['ceiling_height'] > 3]

In [52]: 1 data_3_up.describe()

Out[52]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floor
count	1,528.00	1,528.00	1,528.00	1,528.00	1,528.00	1,528.00	1,445.00	1,528.00
mean	10.72	16,058,118.11	103.79	3.09	3.71	6.25	62.53	3.81
std	6.16	23,669,769.24	66.62	1.53	2.78	3.53	42.23	2.71
min	0.00	550,000.00	13.00	0.00	3.01	2.00	9.00	1.00
25%	6.00	6,900,000.00	64.38	2.00	3.15	5.00	35.70	2.00
50%	10.00	9,700,000.00	86.00	3.00	3.26	5.00	52.21	3.00
75%	16.00	15,500,000.00	121.43	4.00	3.50	6.00	74.99	5.00
max	50.00	330,000,000.00	631.20	15.00	32.00	36.00	409.00	26.00

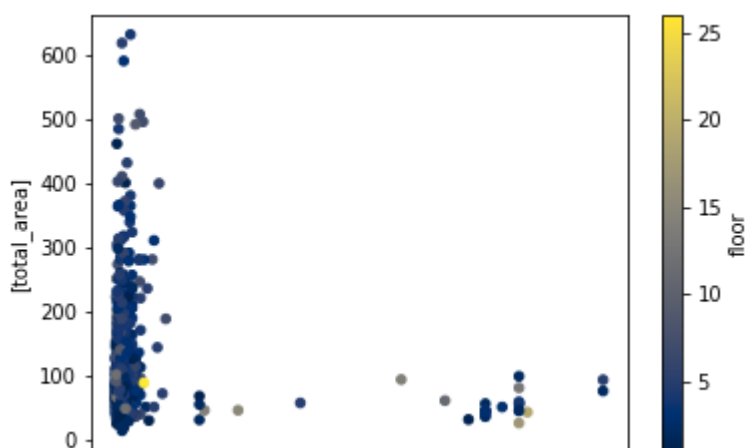
In [53]: 1 data_3_up.plot.scatter(x=['ceiling_height'], y=['last_price'], c='total_area', cmap=



Интересный феномен. Высота потолка выше ~7 метров, а цена очень маленькая.

- похоже, что при вводе данных ошиблись запятой.

In [54]: 1 data_3_up.plot.scatter(x=['ceiling_height'], y=['total_area'], c='floor', cmap=



С параметром "площадь" картина похожа

Так. Посоветовался с риелторами. Ещё раз посмотрим сводку данных по высоте потолков

```
In [55]: 1 data['ceiling_height'].describe()
```

```
Out[55]: count    14,500.00
         mean       2.77
         std        0.97
         min        2.00
         25%        2.52
         50%        2.65
         75%        2.80
         max        32.00
         Name: ceiling_height, dtype: float64
```

- Среднее 2,73
- больше всего значений 2,8
- std отклонение ,28
- самое большое значение 32 м
- ИТАК. Все значения больше 17 метров ЗАМЕНЯЮ на их же, но поделённые на 10.
- значения до 17 метров оставлю как есть.

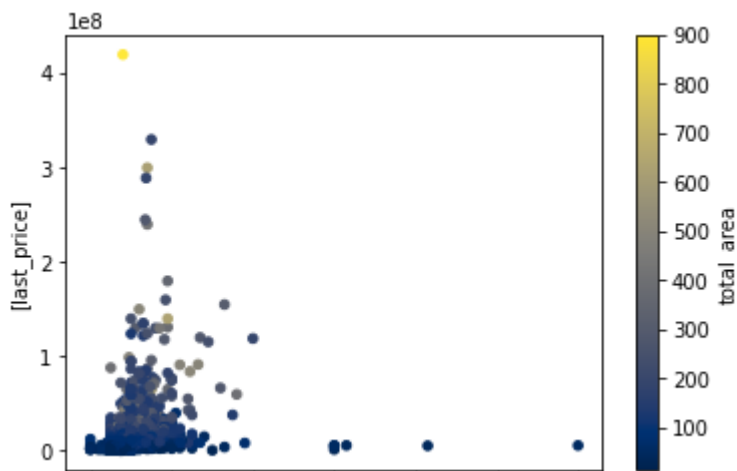
```
In [56]: 1 data.loc[(data['ceiling_height'] > 17 ), 'ceiling_height'] = data['ceiling_heig
```

Проверка

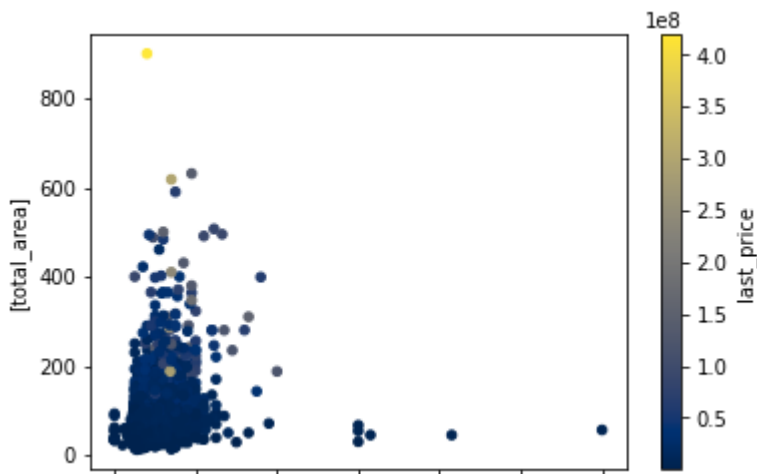
```
In [57]: 1 data['ceiling_height'].describe()
```

```
Out[57]: count    14,500.00
         mean       2.73
         std        0.31
         min        2.00
         25%        2.51
         50%        2.65
         75%        2.80
         max        14.00
         Name: ceiling_height, dtype: float64
```

```
In [58]: 1 data.plot.scatter(x=['ceiling_height'], y=['last_price'], c='total_area', cmap=
```



```
In [59]: 1 data.plot.scatter(x=['ceiling_height'], y=['total_area'], c='last_price', cmap=
```



Посмотрел я на эту картину и принял ещё одно волонтаристское решение

- значения высоты потолка более 7 метров заменяю на NaN через промежуточный "0"

```
In [60]: 1 data.loc[(data['ceiling_height'] > 7 ), 'ceiling_height'] = 0
```

```
In [61]: 1 data['ceiling_height'].replace(0, np.NaN, inplace = True)
```

Проверка

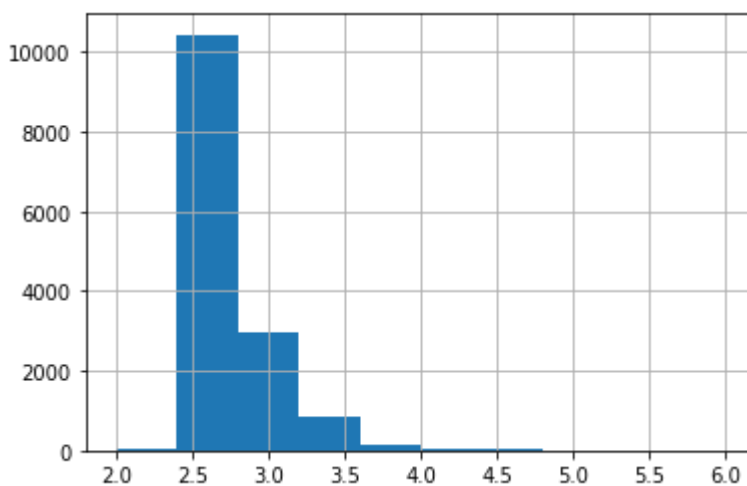
```
In [62]: 1 data['ceiling_height'].describe()
```

```
Out[62]: count    14,494.00
         mean       2.73
         std        0.28
         min        2.00
         25%        2.51
         50%        2.65
         75%        2.80
         max         6.00
         Name: ceiling_height, dtype: float64
```

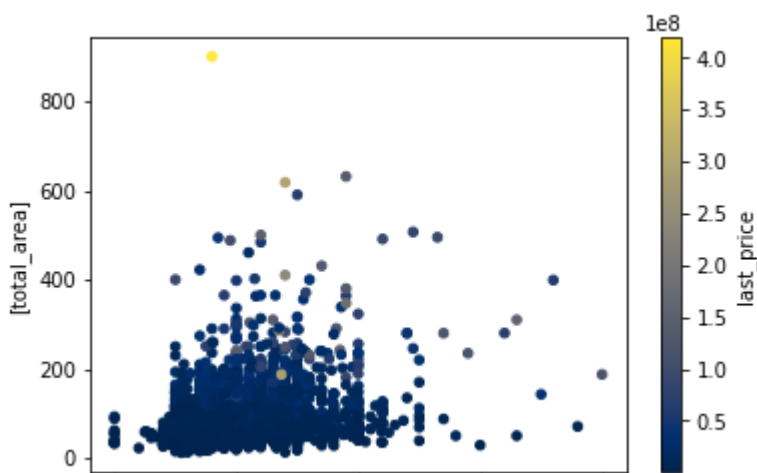
```
In [63]: 1 sorted(data.ceiling_height.unique())
```

```
Out[63]: [2.0,  
2.2,  
2.7,  
nan,  
2.25,  
2.2600000000000002,  
2.3,  
2.34,  
2.4,  
2.45,  
2.46,  
2.47,  
2.48,  
2.49,  
2.5,  
2.51,  
2.52,  
2.53,  
2.54,  
~ ~ ~
```

```
In [64]: 1 data['ceiling_height'].hist();
```



```
In [65]: 1 data.plot.scatter(x=['ceiling_height'], y=['total_area'], c='last_price', cmap=
```



3.10.1 данные по высоте потолка готовы к дальнейшей работе

3.11 floors_total

Из общей информации о качестве данных:

- пропуски есть. но меньше одного процента
- есть подозрительные значения
- ▪ 1
- ▪ 60
- смогу ли я их проверить, и надо ли это делать?

Проверяем данные со значением floors_total == 1

Какие значения "Этаж" у объектов с "floors_total" == 1

```
In [66]: 1 data[data['floors_total'] == 1].floor.unique()
```

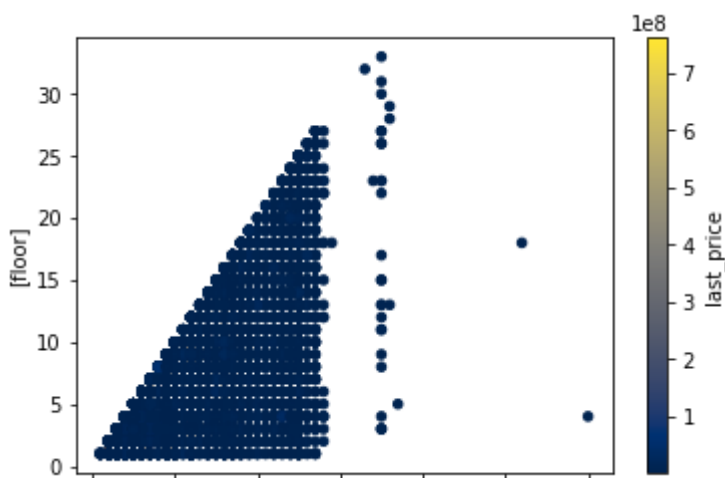
```
Out[66]: array([1], dtype=int64)
```

Что ж. Во всех строках с количеством этажей "1", этаж также указан как "1".

- данные верны

Посмотрим, как соотносятся данные по этажу и количеству этажей

```
In [67]: 1 data.plot.scatter(x=['floors_total'], y=['floor'], c='last_price', cmap="cividib")
```



В городе много объектов в зданиях до 30-ти этажей

- в них плотное распределение предложений по всей этажности
- ▪ встречается немного зданий 35 - 38 этажей с небольшим количеством вариантов по этажу объекта
- есть два уникальных здания 50 -60 этажей.
- ▪ предложения в них отличаются малой вариативностью
- -- предполагаю ошибку в данных.
- -- из общего анализа такие объекты можно исключить, так как в отсутствии дополнительных сведений (адрес), нет возможности проверить данные.

3.11.1 floors_total - вывод

При анализе зависимостей с этажностью можно исключить

- строки с пропусками

- строки с этажностью более 45 этажей

3.12 living_area

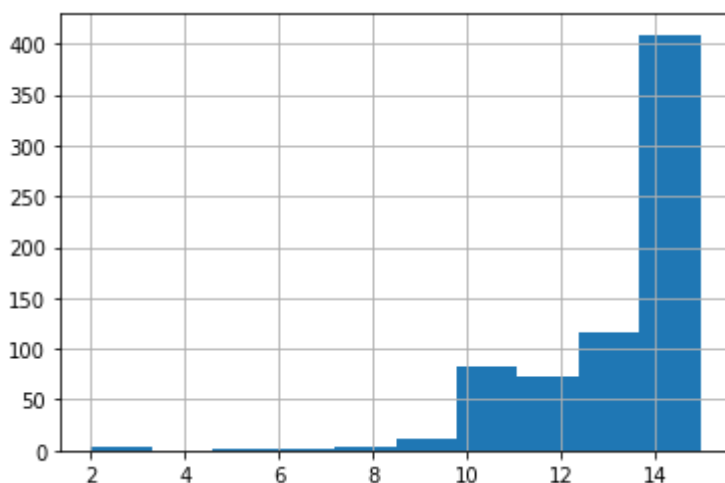
```
In [68]: 1 data['living_area'].describe()
```

```
Out[68]: count    21,796.00
         mean      34.46
         std       22.03
         min        2.00
         25%       18.60
         50%       30.00
         75%       42.30
         max      409.70
         Name: living_area, dtype: float64
```

Есть странные значения - малая площадь - 2 метра.

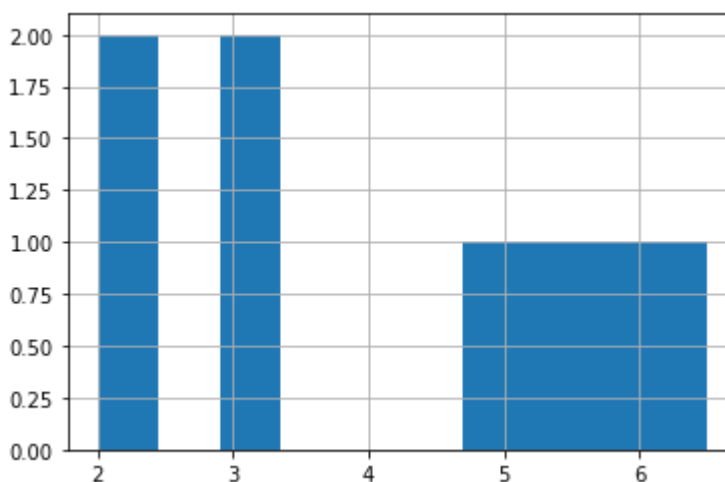
- посмотрим, какие значения встречаются малые

```
In [69]: 1 data[data['living_area'] < 15]['living_area'].hist();
```



Точку отсечения поставим 8 метров

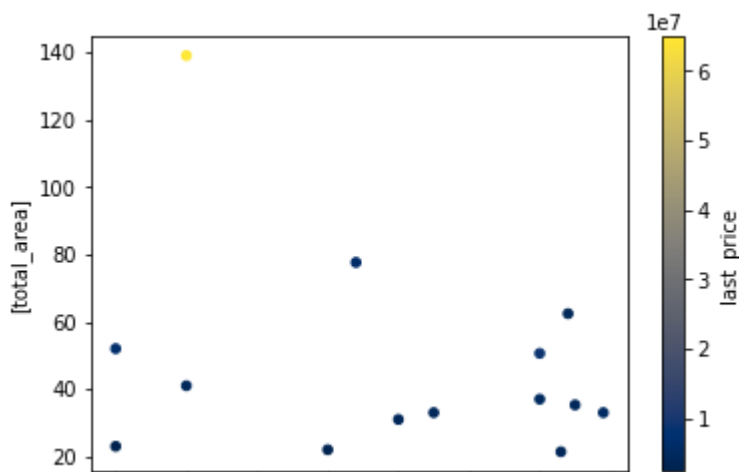
```
In [70]: 1 data[data['living_area'] < 8]['living_area'].hist();
```



Значения "жилая площадь" перекликаются со значениями "общая площадь"

- проверим для малых значений

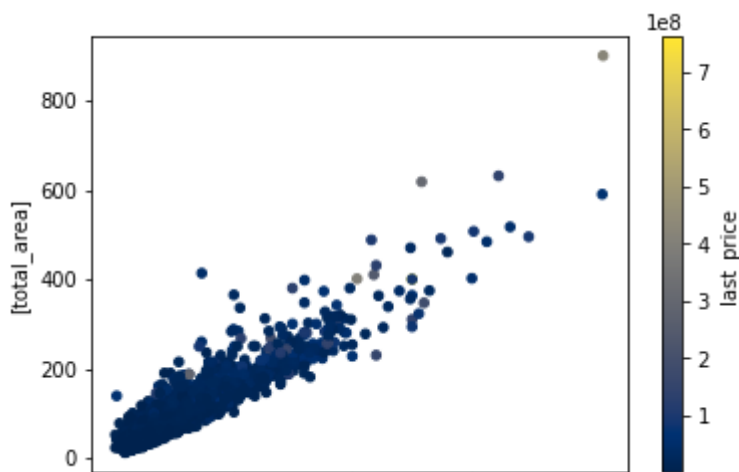
```
In [71]: 1 data[data['living_area'] < 9].plot.scatter(x=['living_area'], y=['total_area'],
```



Значения меньше 9 метров встречаются редко и подчас в объектах с большей общей площадью.

- из анализа такие строки можно исключить

```
In [72]: 1 data.plot.scatter(x=['living_area'], y=['total_area'], c='last_price', cmap="ci
```



В целом же данные по жилой площади имеют линейную зависимость от размера общей площади.

- сомневаюсь, имеет ли значительное влияние этот параметр на стоимость жилья

3.12.1 living_area - вывод

При анализе зависимостей с размером жилой площади можно исключить

- строки с пропусками
- строки со значениями менее равно 9 метров
- строки, в которых соотношение жилой и общей площади значительно отличается от общего коэффициента.

3.13 floor

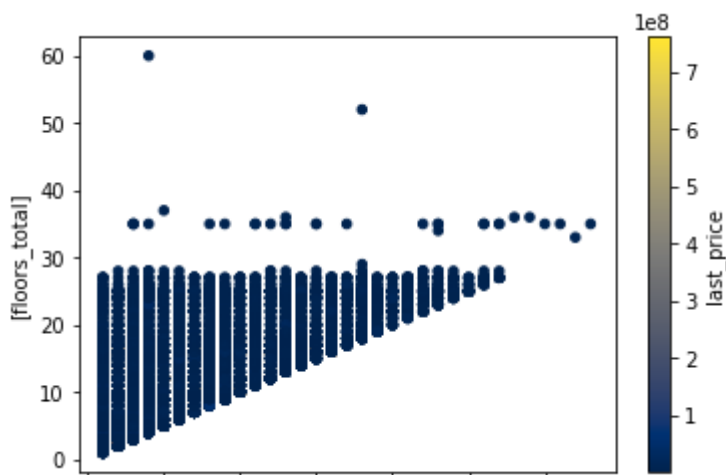
```
In [73]: 1 data['floor'].describe()
```

```
Out[73]: count    23,699.00
         mean       5.89
         std        4.89
         min        1.00
         25%        2.00
         50%        4.00
         75%        8.00
         max       33.00
         Name: floor, dtype: float64
```

Отлично!

- нет странных значений (0)
- нет пропусков данных
- - проверим соотношение Этаж / Этажей

```
In [74]: 1 data.plot.scatter(x=['floor'], y=['floors_total'], c='last_price', cmap="cividib")
```



Отличные новости!

- нет случаев, когда этаж больше, чем этажность.

3.13.1 floor - вывод

Данные хорошие.

- можно смело использовать при анализе

3.14 is_apartment

```
In [75]: 1 data['is_apartment'].describe()
```

```
Out[75]: count      2775
         unique        2
         top      False
         freq      2725
         Name: is_apartment, dtype: object
```

Данных мало, и при этом, к тому же, большинство значений - False

- посмотрим, сколько значений True

```
In [76]: 1 data[data['is_apartment'] == True]['is_apartment'].count()
```

```
Out[76]: 50
```

50 (!) объектов в статусе "апартаменты"

- это мизерный процент и не подходит для анализа влияния на цену
- -- вероятно, этот параметр просто не выставлен во всех объектах, которые на самом деле являются апартаментами.
- в реальном исследовании можно запросить дополнительные данные.

3.14.1 is_apartment - вывод

Данные плохие.

- при анализе не будем использовать

3.15 open_plan

```
In [77]: 1 data['open_plan'].describe()
```

```
Out[77]: count      23699
         unique        2
         top      False
         freq     23632
         Name: open_plan, dtype: object
```

```
In [78]: 1 data[data['open_plan'] == True]['open_plan'].count()
```

```
Out[78]: 67
```

67 (!) объектов в статусе "свободная планировка"

- это мизерный процент и не подходит для анализа влияния на цену
- -- вероятно, этот параметр просто не выставлен во всех объектах, которые на самом деле являются такими.
- в реальном исследовании можно запросить дополнительные данные.

3.15.1 open_plan - вывод

Данные плохие.

- при анализе не будем использовать

3.16 kitchen_area

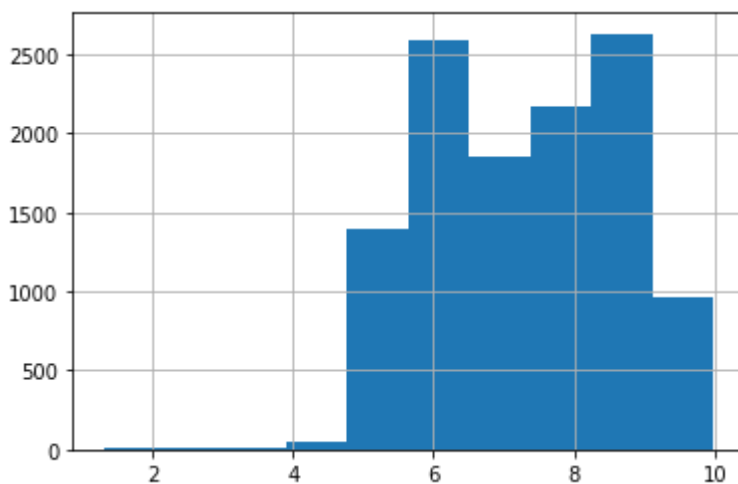
- *** - kitchen_area = общая площадь - жилая площадь - санузел - прихожая

```
In [79]: 1 data['kitchen_area'].describe()
```

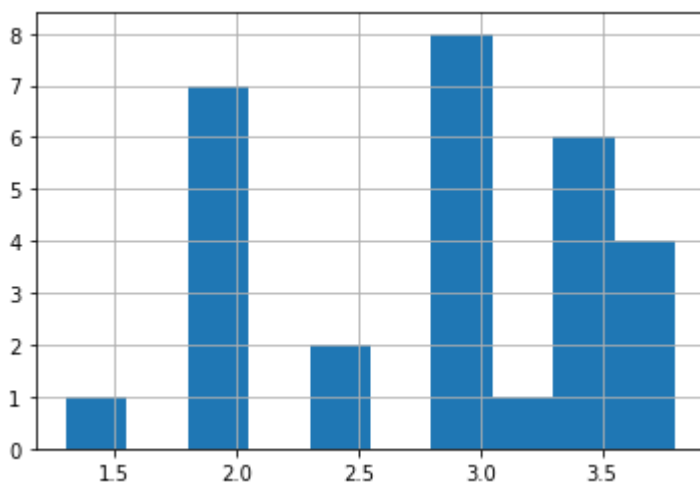
```
Out[79]: count    21,421.00  
mean         10.57  
std           5.91  
min           1.30  
25%           7.00  
50%           9.10  
75%          12.00  
max          112.00  
Name: kitchen_area, dtype: float64
```

- есть пропуски значений
- есть странно маленькие значения - проверить в соотношении с общей и жилой площадями
- есть странно большие значения - проверить в соотношении с общей и жилой площадями

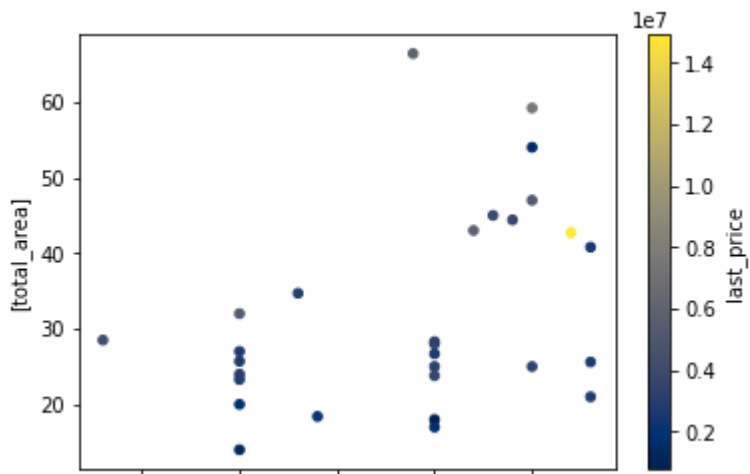
```
In [80]: 1 # посмотрим значения менее среднего  
2 data[data['kitchen_area'] < 10]['kitchen_area'].hist();
```



```
In [81]: 1 # посмотрим значения менее 4 метров  
2 data[data['kitchen_area'] < 4]['kitchen_area'].hist();
```



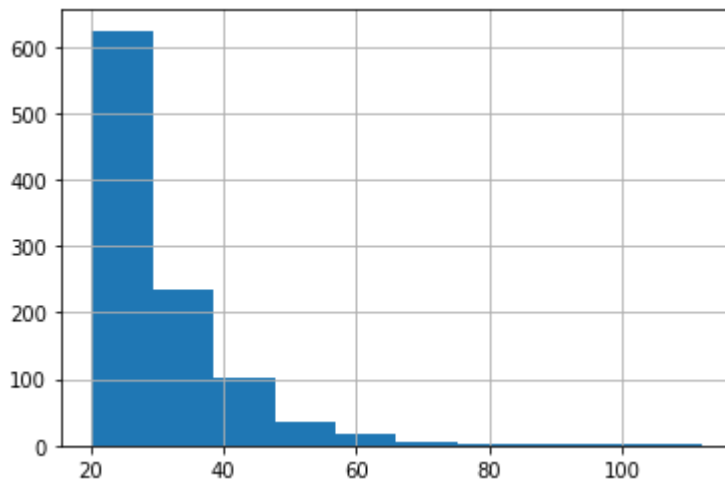
In [82]: `1 data[data['kitchen_area'] < 4].plot.scatter(x=['kitchen_area'], y=['total_area'])`



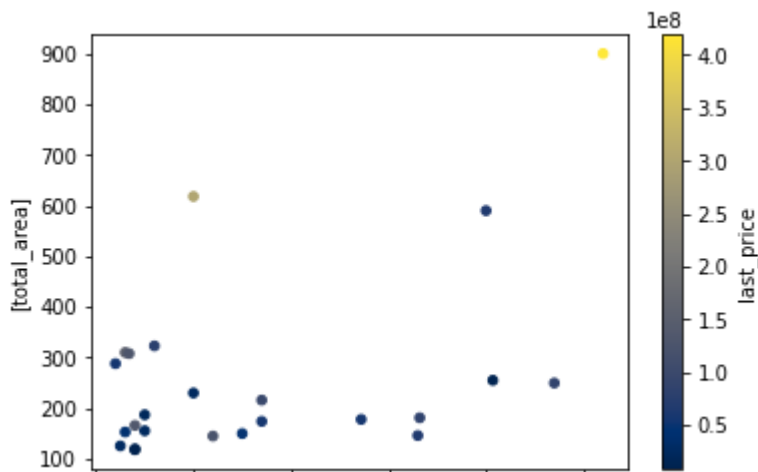
3.16.1 Вывод1. Значения менее 4 метров можно смело исключить из анализа

3.16.2 Рассмотрим очень большие значения

In [83]: `1 # посмотрим значения более 20 метров
2 data[data['kitchen_area'] > 20]['kitchen_area'].hist();`



```
In [84]: 1 # распределение размера кухни более 60 метров к общей площади
2 data[data['kitchen_area'] > 60].plot.scatter(x=['kitchen_area'], y=['total_area'])
```



3.16.3 Вывод2: вариантов, в которых размер кухни не был бы частью общей площади, не видно.

Значения большой площади можно использовать в анализе

3.16.4 kitchen_area - вывод

значения с малой (менее 4х метров) величиной, исключаем из анализа

3.17 balcony

```
In [85]: 1 data['balcony'].describe()
```

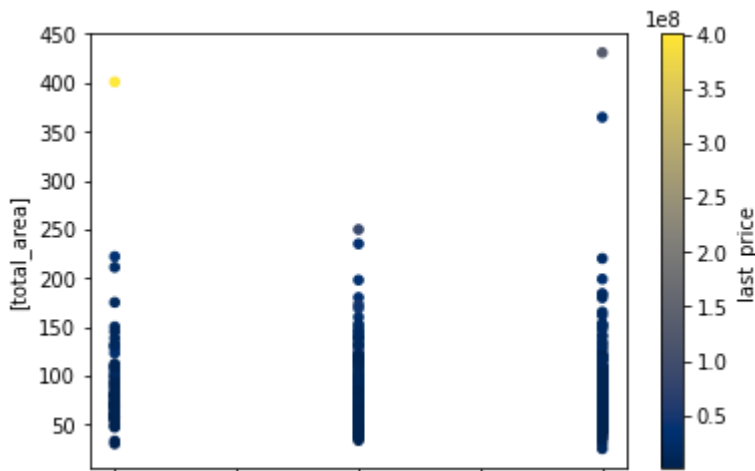
```
Out[85]: count    12,180.00
mean         1.15
std          1.07
min           0.00
25%           0.00
50%           1.00
75%           2.00
max           5.00
Name: balcony, dtype: float64
```

- нет отрицательных значений
- нет очень больших значений
- ▪ посмотрим в каких объектах 3 и более балконов
- есть пропуски значений

- есть нулевые значения
- пропуск значения не обязательно равен нулевому значению
- ▪ тип данных Float64, но по сути это int
- изменим тип данных

In [86]:

```
1
2 data[data['balcony'] > 2].plot.scatter(x=['balcony'], y=['total_area'], c='last_price')
```



- ИЗМЕНИМ ТИП ДАННЫХ

тип данных изменить на INT не получается, так как есть пропуски значений. заполнять их нечем, потому что пропуски невозможно соотнести с наличием или отсутствием балкона

3.17.1 Здравый смысл подсказывает, что у объектов малой площади навряд ли может быть более 3х балконов.

ВЫВОД: для анализа рассматривать будем лишь "нет данных", "без балкона", "есть балкон", "больше одного балкона"

In [87]:

```
1 data['balcony'] = data['balcony'].fillna(0)
```

In [88]:

```
1 sorted(data.balcony.unique())
```

Out[88]: [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

С балконами закончили

3.18 locality_name

In [89]:

```
1 data['locality_name'].describe()
```

```
Out[89]: count          23650
unique           364
top      Санкт-Петербург
freq           15721
Name: locality_name, dtype: object
```

In [90]: ▶

```
1 # посмотрим уникальные значения на предмет неявных дубликатов
2 print(data['locality_name'].unique())
```

```
['Санкт-Петербург' 'посёлок Шушары' 'городской посёлок Янино-1'
 'посёлок Парголово' 'посёлок Мурино' 'Ломоносов' 'Сертолово' 'Петергоф'
 'Пушкин' 'деревня Кудрово' 'Коммунар' 'Колпино'
 'поселок городского типа Красный Бор' 'Гатчина' 'поселок Мурино'
 'деревня Фёдоровское' 'Выборг' 'Кронштадт' 'Кировск'
 'деревня Новое Девяткино' 'посёлок Металлострой'
 'посёлок городского типа Лебяжье' 'посёлок городского типа Сиверский'
 'поселок Молодцово' 'поселок городского типа Кузьмолровский'
 'садовое товарищество Новая Ропша' 'Павловск' 'деревня Пикколово'
 'Всеволожск' 'Волхов' 'Кингисепп' 'Приозерск' 'Сестрорецк'
 'деревня Куттузи' 'посёлок Аннино' 'поселок городского типа Ефимовский'
 'посёлок Плодовое' 'деревня Заклинье' 'поселок Торковичи'
 'поселок Первомайское' 'Красное Село' 'посёлок Понтонный' 'Сясьстрой'
 'деревня Старая' 'деревня Лесколово' 'посёлок Новый Свет' 'Сланцы'
 'село Путилово' 'Ивангород' 'Мурино' 'Шлиссельбург' 'Никольское'
 'Зеленогорск' 'Сосновый Бор' 'поселок Новый Свет' 'деревня Оржицы'
 'деревня Кальтино' 'Кудрово' 'поселок Романовка' 'посёлок Бугры'
 'поселок Бугры' 'поселок городского типа Рожино' 'Кириши' 'Луга'
 'Волосово' 'Отрадное' 'село Павлово' 'поселок Оредеж' 'село Копорье'
 '...']
```

- посёлок Бугры - поселок Бугры'
- посёлок городского типа Мга - поселок городского типа Мга
- ▪ видим то, что встречается два написания слова "посёлок".
- ▪ через "е" и через "ё"
- для целей исследования это не существенная особенность данных.
- ▪ главное для анализа - Санкт-Петербург это или нет.
- посмотрел впереди задачи. да. надо поменять посёлок на поселок

In [91]: ▶

```
1 # выяснилось то, что в работе нам мешают значения nan
2 # заменим их на "не указан"
3 data['locality_name'] = data['locality_name'].fillna('не указан')
```

In [92]: ▶

```
1 # меняем "посёлок" на "поселок"
2 # променяем все буквы ё и Ё на е и Е
3
4 #data['locality_name'] = data['locality_name'].replace('ё', 'е')
5 def replacer(string):
6
7     string = string.replace('ё', 'е').replace('Ё', 'Е')
8     return string
9
10 data['locality_name'] = data['locality_name'].apply(replacer)
```

In [93]:

```
1 # посмотрим уникальные значения на предмет неявных дубликатов
2 print(sorted(data['locality_name'].unique()))
```

```
['Бокситогорск', 'Волосово', 'Волхов', 'Всеволожск', 'Выборг', 'Высоцк', 'Гатчин
а', 'Зеленогорск', 'Ивангород', 'Каменногорск', 'Кингисепп', 'Кириши', 'Кировс
к', 'Колпино', 'Коммунар', 'Красное Село', 'Кронштадт', 'Кудрово', 'Лодейное Пол
е', 'Ломоносов', 'Луга', 'Любань', 'Мурино', 'Никольское', 'Новая Ладога', 'Отра
дное', 'Павловск', 'Петергоф', 'Пикалево', 'Подпорожье', 'Приморск', 'Приозерс
к', 'Пушкин', 'Санкт-Петербург', 'Светогорск', 'Сертолово', 'Сестрорецк', 'Сланц
ы', 'Сосновый Бор', 'Сясьстрой', 'Тихвин', 'Тосно', 'Шлиссельбург', 'городской п
оселок Большая Ижора', 'городской поселок Будогощь', 'городской поселок Виллоз
и', 'городской поселок Лесогорский', 'городской поселок Мга', 'городской поселок
Назия', 'городской поселок Новоселье', 'городской поселок Павлово', 'городской п
оселок Рощино', 'городской поселок Свирьстрой', 'городской поселок Советский',
'городской поселок Федоровское', 'городской поселок Янино-1', 'деревня Агалатов
о', 'деревня Аро', 'деревня Батово', 'деревня Бегуницы', 'деревня Белогорка', 'д
еревня Большая Вруда', 'деревня Большая Пустомержа', 'деревня Большие Колпаны',
'деревня Большое Рейзино', 'деревня Большой Сабск', 'деревня Бор', 'деревня Бори
сова Грива', 'деревня Ваганово', 'деревня Вартемяги', 'деревня Вахнова Кара', 'д
еревня Вискатка', 'деревня Гарболово', 'деревня Глинка', 'деревня Горбунки', 'де
ревня Гостилицы', 'деревня Заклинье', 'деревня Заневка', 'деревня Зимитицы', 'де
ревня Извара', 'деревня Иссад', 'деревня Калитино', 'деревня Кальтино', 'деревня
```

In [94]:

```
1 # сколько строк с "не указан"
2 data['locality_name'].value_counts().head(50)
```

```
Out[94]: Санкт-Петербург      15721
поселок Мурино              556
поселок Шушары              440
Всеволожск                  398
Пушкин                      369
Колпино                     338
поселок 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
поселок Бугры               114
```

3.18.1 "не указан" менее 50 штук

это мало и на статистику по Петербургу не окажет существенного влияния.

- с заменой на значение, похожее по расстояниям, пока менять не буду.

3.18.2 locality_name - вывод

Для анализа рассматриваем вопрос

- это Санкт-Петербург или нет

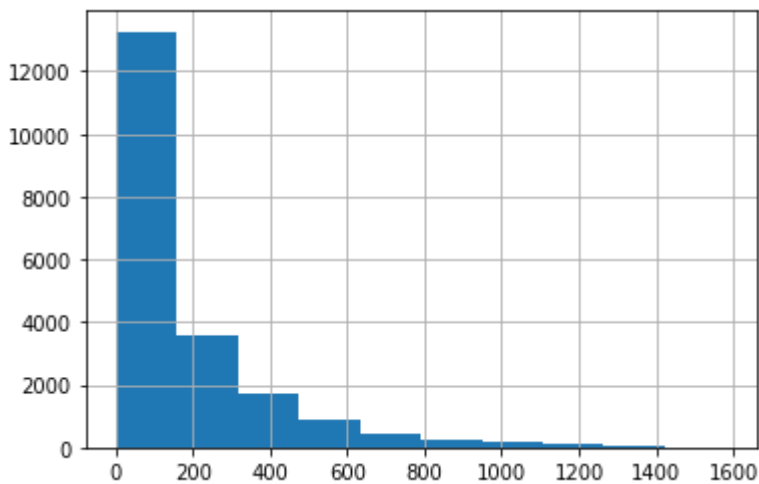
3.19 days_exposition

```
In [95]: 1 data['days_exposition'].describe()
```

```
Out[95]: count    20,518.00
         mean      180.89
         std       219.73
         min        1.00
         25%       45.00
         50%       95.00
         75%      232.00
         max     1,580.00
         Name: days_exposition, dtype: float64
```

- Есть пропуски значений
- - ничего с ними не сделать. просто отметить при анализе
- интересно будет посмотреть влияние этого параметра на стоимость по сравнению похожими объектами.
- - посмотрим как распределены значения

```
In [96]: 1 data['days_exposition'].hist();
```



Дела на рынке недвижимости вполне себе хорошо идут.

- график имеет красивый вид
- - и за полгода большинство объектов уходят с рынка

3.19.1 days_exposition - вывод

интересно рассмотреть зависимость цены от срока экспозиции

4 Вывод по данным

- airports_nearest — расстояние до ближайшего аэропорта в метрах (м) есть пропуски в данных. -- Вот так новость! Пропуски есть и в Санкт-Петербург. А я то думал, что только вне города пропущены эти сведения. Что же делать с этими пропусками? По сведениям от заказчика, эти данные сгенерированы автоматически. Внести их вручную не получится быстро. Их очень много... Значит, оставим "КАК ЕСТЬ". И при исследовании влияния этих данных строки с пропусками будем исключать из анализа.
- cityCenters_nearest — расстояние до центра города (м) примерно столько же пропусков, как в airports_nearest проверить, есть ли соответствие или пропуски не согласованы друг с другом. так как у нас в данных нет адресов, то мы не сможем при всём желании самостоятельно оассчитать этот параметр

-- Гипотеза подтвердилась. Данные по расстояниям до аэропорта и до центра города пропущены практически в одних и тех же строках. Срез данных без этих пропусков `airports_and_cityCenters_nearest_drop_na` содержит 18156 строк.

- `balcony` — число балконов есть пропуски значений (около половины от всей базы) и есть значения "0". Считаю, что пропуски значений в количестве балконов могут означать отсутствие балкона. то есть присвоить значение "0". Таким образом, 50% с пропусками + 25% со значением "0" получение очень много квартир без балконов -- **ВЫВОД:** для анализа рассматривать будем лишь "нет данных", "без балкона", "есть балкон", "больше одного балкона"

- `ceiling_height` — высота потолков (м) очень много пропусков. почти 40% от общей массы значений считаю, что высота потолков указывается в основном тогда, когда она выгодно отличается от обычного значения. поэтому можно подставить высоту потолков значением, наиболее часто встречающимся также есть странные значения 1м 100м следует проанализировать эти значения и решить, как с ними поступить

-- данные по высоте потолка готовы к дальнейшей работе

- `days_exposition` — сколько дней было размещено объявление (от публикации до снятия) странно, что в этом показателе есть пропуски. предлагаю проверить связку этого параметра с **`first_day_exposition`** и рассчитать самостоятельно недостающие данные

-- нет. привязать к **`first_day_exposition`**, потому что недостаёт данных по дате снятия объекта с экспозиции

- `first_day_exposition` — дата публикации прекрасно! пропусков нет. данные пришли в строковом формате. 2018-12-04T00:00:00

переведём в питоновский формат даты -- данные готовы к работе

- `floor` — этаж супер! пропусков нет нет странных значений небоскрёбы есть в городе (33 этаж)

-- Данные хорошие.

----можно смело использовать при анализе

- `floors_total` — всего этажей в доме есть чуть пропусков. меньше одного процента. проверим, может быть это просто одноэтажные дома? выведем эти значения в корреляции с этажом

-- При анализе зависимостей с этажностью можно исключить

--- строки с пропусками --- строки с этажностью более 45 этажей

- `is_apartment` — апартаменты (булев тип) есть немного пропусков. думаю, что пропуски - это значит, что это не апартаменты, а полноценные квартиры заменим пропуск на "не апартамент"

-- 50 (!) объектов в статусе "апартаменты"

--- это мизерный процент и не подходит для анализа влияния на цену -- вероятно, этот параметр просто не выставлен во всех объектах, которые на самом деле являются апартаментами. --- в реальном исследовании можно запросить дополнительные данные.

- `kitchen_area` — площадь кухни в квадратных метрах (м²) есть квартиры без данных в этом параметре может быть, это квартиры-студии? проверим

-- значения с малой (менее 4х метров) величиной, исключаем из анализа

- `last_price` — цена на момент снятия с публикации хорошая новость: пропуски и странные значения отсутствуют но стоит проверить, нет доли странных значений в корреляции с метражом объектов

-- В данных отсутствуют пропуски и странные значения (≤ 0) на первый взгляд.

- `living_area` — жилая площадь в квадратных метрах (м²) есть и пропуски, и странные значения. вероятно, это связано с квартирами формата студия или ошибки заполнения данных проверим корреляцию со Студия, Общая площадь, Площадь кухни

-- При анализе зависимостей с размером жилой площади можно исключить

--- строки с пропусками --- строки со значениями менее равно 9 метров --- строки, в которых соотношение жилой и общей площади значительно отличается от общего коэффициента.

- `locality_name` — название населённого пункта интересно .что даже здесь есть пропуски. странно, не правда ли? думаю, что этот параметр не существенен. -- заменил на "не указан" у нас есть расстояния до знаковых мест

-- главное для анализа - Санкт-Петербург это или нет. -- заменил все включения ё и Ё на е и Е

- `open_plan` — свободная планировка (булев тип) чудесно! пропусков нет

-- данных очень мало

- `parks_around3000` — число парков в радиусе 3 км здесь примерно столько же пропусков, как и расстояниях до центра, до аэропорта кроме того, очень много строк со значением "0" основное значение - 1 бывают места с количеством парков 3 хочу туда!
- `parks_nearest` — расстояние до ближайшего парка (м) здесь больше пропусков. видимо, кроме тоех строк, где нет данных по количеству парков поблизости, добавились ещё строки с пропусками

-- Для анализа влияния на цену близости парков и водоёмов следует исследовать только строки со значениями больше "0"

- `ponds_around3000` — число водоёмов в радиусе 3 км здесь примерно столько же пропусков, как и расстояниях до центра, до аэропорта кроме того, очень много строк со значением "0"
- `ponds_nearest` — расстояние до ближайшего водоёма (м) здесь больше пропусков. видимо, кроме тоех строк, где нет данных по количеству парков поблизости, добавились ещё строки с пропусками

-- Для анализа влияния на цену близости парков и водоёмов следует исследовать только строки со значениями больше "0"

- `rooms` — число комнат пропусков нет, но есть "0" м б это студии? проверим...
- `studio` — квартира-студия (булев тип) ответственно подошли к заполнению этого параметра. пропусков нет

-- для всех "Студия" выставить значение "Комнат" == 0 --- готово

- `total_area` — площадь квартиры в квадратных метрах (м²) пожалуй, самый полно заполненный параметр. ни одного пропуска и ни одного странного значения!
- `total_images` — число фотографий квартиры в объявлении встречаются объявления без фотографий. это нормальная практика. к тому же, эта колонка нам навряд ли понадобится для ответов на поставленные перед исследованием вопросы.

5 Расчёты и добавление результатов в таблицу

5.1 Цена квадратного метра

Стоимость / площадь

`price_per_square_meter`

```
In [97]: 1 data['price_per_square_meter'] = data['last_price'] / data['total_area']
```

```
In [98]: 1 data['price_per_square_meter'].describe()
```

```
Out[98]: count      23,699.00  
mean       99,421.66  
std        50,306.80  
min         111.83  
25%        76,585.47  
50%        95,000.00  
75%       114,256.33  
max       1,907,500.00  
Name: price_per_square_meter, dtype: float64
```

```
In [99]: 1 # настройка отображения численных значений  
2 pd.set_option('display.float_format', '{:,.2f}'.format)
```

```
In [100]: 1 data['price_per_square_meter'].describe()
```

```
Out[100]: count      23699.00  
mean       99421.66  
std        50306.80  
min         111.83  
25%        76585.47  
50%        95000.00  
75%       114256.33  
max       1907500.00  
Name: price_per_square_meter, dtype: float64
```

5.2 день недели, месяц и год публикации объявления

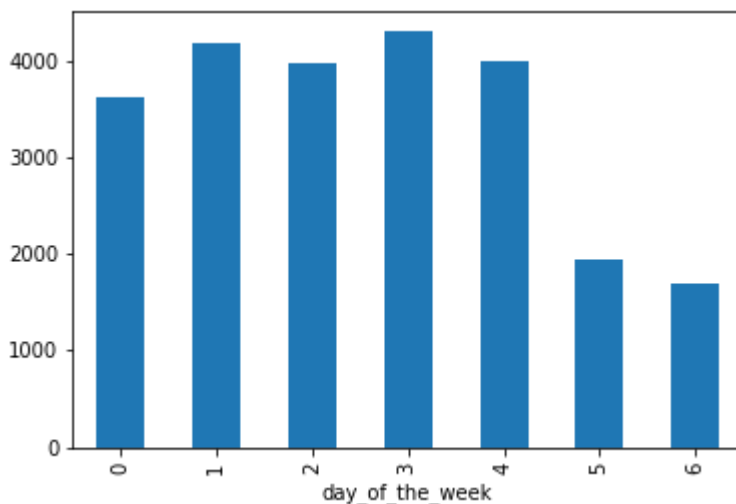
5.2.1 день недели

day_of_the_week

```
In [101]: 1 data['day_of_the_week'] = data['first_day_exposition'].dt.weekday
```

5.3 Посмотрим на графике

```
In [102]: 1 data.groupby(['day_of_the_week'])['last_price'].count().plot(kind='bar');
```



5.3.0.1 Забавный факт:

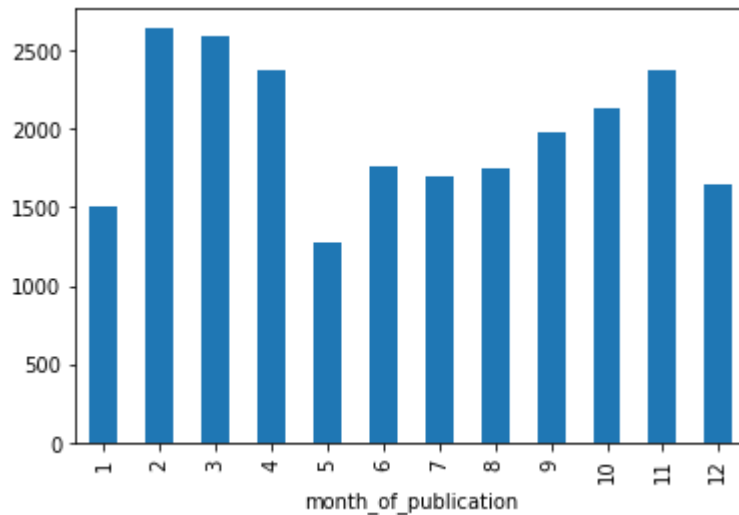
- в будние дни объявлений было размещено больше, чем выходные...

5.3.1 месяц

month_of_publication

```
In [103]: 1 data['month_of_publication'] = data['first_day_exposition'].dt.month
```

```
In [104]: 1 data.groupby(['month_of_publication'])['last_price'].count().plot(kind='bar');
```

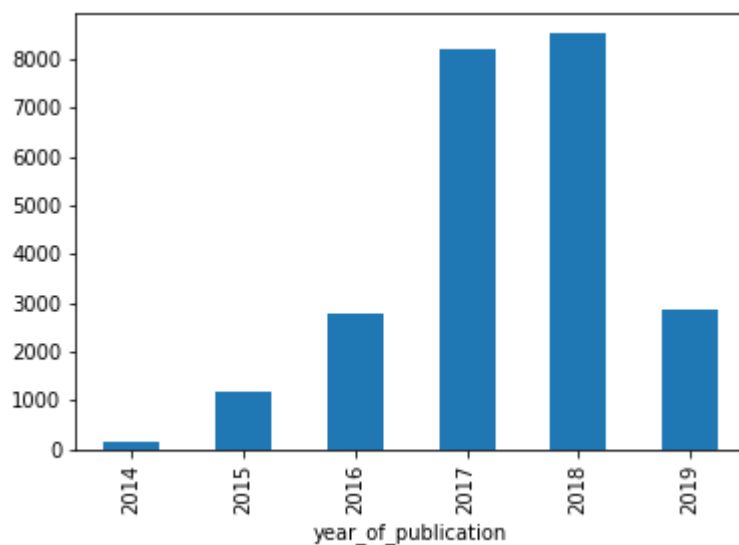


5.3.2 год

year_of_publication

```
In [105]: 1 data['year_of_publication'] = data['first_day_exposition'].dt.year
```

```
In [106]: 1 data.groupby(['year_of_publication'])['last_price'].count().plot(kind='bar');
```



5.4 этаж квартиры

apartment_floor

- первый
- последний

- другой
- если floor == 1
 - то первый,
- если floor == floors_total -- то последний
- иначе другой

определим функцию и применим к данным

```
In [107]: 1 def which_floor(dataset):
          2     if dataset['floor'] == 1:
          3         return 'первый'
          4     elif dataset['floor'] == dataset['floors_total']:
          5         return 'последний'
          6     else:
          7         return 'другой'
          8
```

```
In [108]: 1 # применим функцию к датасету
          2 # результат запишем в новую колонку
          3 data['apartment_floor'] = data.apply(which_floor, axis=1)
```

```
In [109]: 1 data['apartment_floor'].value_counts()
```

```
Out[109]: другой      17446
          последний    3336
          первый      2917
          Name: apartment_floor, dtype: int64
```

5.4.1 Забавный факт

- количество первых и последних этажей почти одинаковое

5.5 соотношение жилой и общей площади

ratio_living_total_area

```
In [110]: 1 # living_area
          2 data['ratio_living_total_area'] = data['living_area'] / data['total_area']
```

```
In [111]: 1 data.ratio_living_total_area.describe()
```

```
Out[111]: count    21796.00
          mean       0.56
          std       0.11
          min       0.02
          25%       0.50
          50%       0.57
          75%       0.64
          max       1.00
          Name: ratio_living_total_area, dtype: float64
```

5.6 соотношение кухни и общей площади

ratio_kitchen_total_area

```
In [112]: 1 # kitchen_area
          2 data['ratio_kitchen_total_area'] = data['kitchen_area'] / data['total_area']
```

```
In [113]: 1 data.ratio_kitchen_total_area.describe()
```

```
Out[113]: count    21421.00
          mean       0.19
          std        0.07
          min        0.03
          25%        0.13
          50%        0.17
          75%        0.23
          max        0.79
          Name: ratio_kitchen_total_area, dtype: float64
```

5.7 Данные добавлены

```
In [114]: 1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 29 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  datetime64[ns]
 4   rooms                                23699 non-null  int64
 5   ceiling_height                       14494 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                              23699 non-null  float64
14   locality_name                        23699 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
22   price_per_square_meter                23699 non-null  float64
23   day_of_the_week                      23699 non-null  int64
24   month_of_publication                  23699 non-null  int64
25   year_of_publication                   23699 non-null  int64
26   apartment_floor                      23699 non-null  object
27   ratio_living_total_area               21796 non-null  float64
28   ratio_kitchen_total_area              21421 non-null  float64
dtypes: bool(2), datetime64[ns](1), float64(17), int64(6), object(3)
memory usage: 4.9+ MB
```

6 Исследовательский анализ данных

6.1 цена метра и площадь

```
In [115]: 1 # цена метра
          2 data.price_per_square_meter.describe()
```

```
Out[115]: count    23699.00
          mean     99421.66
          std      50306.80
          min       111.83
          25%      76585.47
          50%      95000.00
          75%     114256.33
          max     1907500.00
          Name: price_per_square_meter, dtype: float64
```

```
In [116]: 1 # цена метра
          2 data.price_per_square_meter.plot(kind='hist', bins=50, legend=True);
```

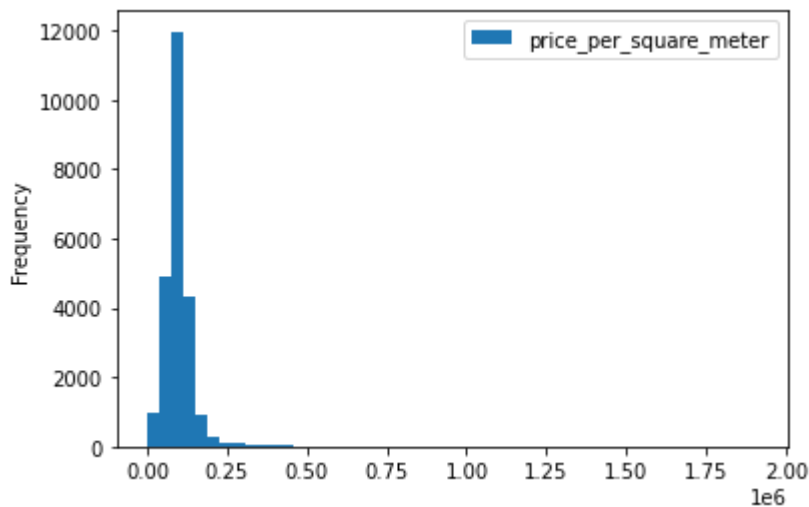
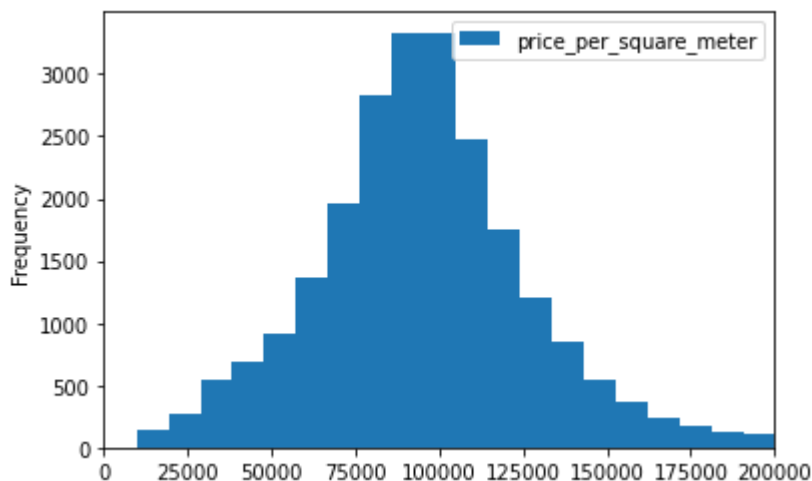


График распределения количества объявлений в зависимости от цены квадратного метра. Мы видим "длинный хвост" вправо. С малым количеством объявлений с аномально высокой стоимостью.

Ограничим график предельным значением 200.000 р

```
In [117]: 1 plt.xlim(0, 200000)
          2 data.price_per_square_meter.plot(kind='hist', bins=200, legend=True);
```



Здесь мы видим медианное значение соответствует 95000 руб

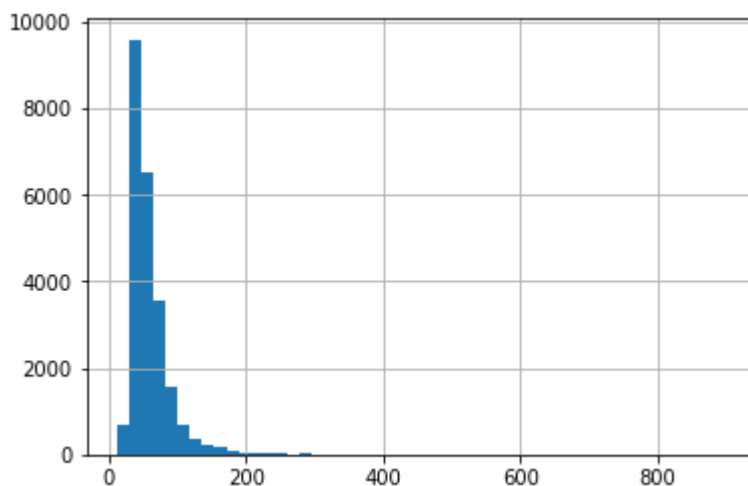
И оно совпадает со справкой перед построением графиков

Большинство цен располагаются в интервале от 50.000 руб до 140.000 руб.

```
In [118]: 1 # площадь
          2 data.total_area.describe()
```

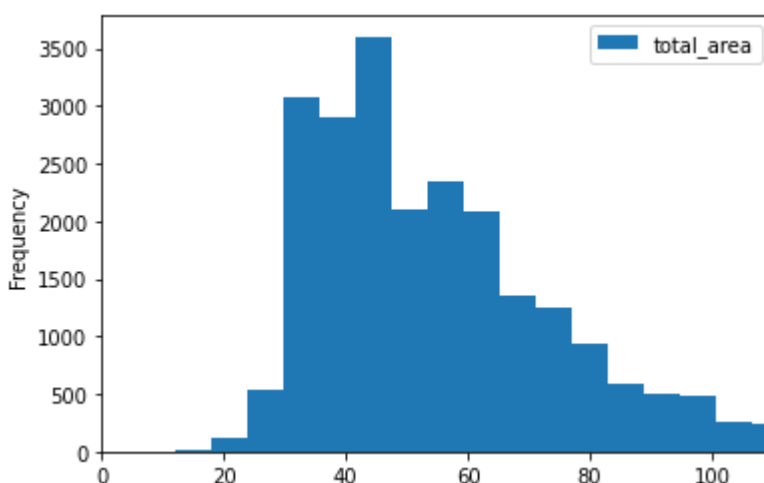
```
Out[118]: count    23699.00
          mean       60.35
          std        35.65
          min        12.00
          25%        40.00
          50%        52.00
          75%        69.90
          max        900.00
          Name: total_area, dtype: float64
```

```
In [119]: 1 # площадь
          2 data.total_area.hist(bins=50);
```



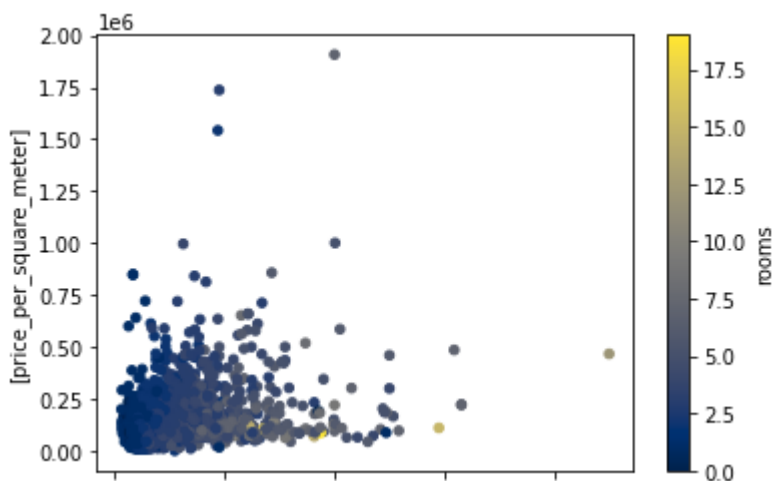
распределение цена метра / площадь
и рассмотрим более детально, обрезав "хвост"

```
In [120]: 1 plt.xlim(0, 110)
          2 data.total_area.plot(kind='hist', bins=150, legend=True);
```



И верно! Общая площадь квартир изменяется "ступеньками" с различными выбросами в областях X-комнатных квартир серий домов массовой застройки.
Серийные дома - одинаковый метраж похожих квартир.

```
In [121]: 1 data.plot.scatter(x=['total_area'], y=['price_per_square_meter'], c='rooms', cm
```



Это - зависимость между ценой квадратного метра и общей площадью квартиры

Как здесь видно ,нет линейной зависимости.

Нельзя сказать, что при увеличении общей площади, цена за квадратный метр уменьшается либо увеличивается.

ТО есть прямой зависимости **нет**.

6.2 число комнат

```
In [122]: 1 # сводка  
2 data.rooms.describe()
```

```
Out[122]: count    23699.00  
mean         2.07  
std          1.08  
min          0.00  
25%          1.00  
50%          2.00  
75%          3.00  
max          19.00  
Name: rooms, dtype: float64
```

In [123]:

```
1 # гистограмма
2 data.rooms.hist(bins=50);
```

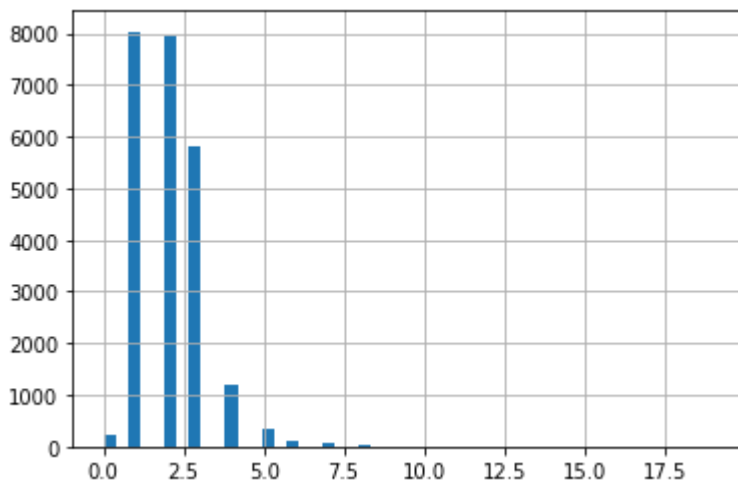


График распределения количества объявлений в зависимости от числа комнат.

Лидеры предложения - Одно и Двух- комнатные квартиры.

Примерно на четверть на рынке меньше предложений Трёх- комнатных квартир.

Объекты свободной планировки попрежнему - экзотика, как и Пяти- и более - комнатные предложения.

Цифры можем посмотреть ниже:

In [124]:

```
1 # встречаемость значений
2 data.rooms.value_counts()
```

```
Out[124]: 1      8036
          2      7940
          3      5814
          4      1180
          5       326
          0       208
          6       105
          7        59
          8        12
          9         8
         10         3
         11         2
         14         2
         15         1
         19         1
         16         1
         12         1
          Name: rooms, dtype: int64
```

```
In [125]: 1 data.plot.scatter(x=['rooms'], y=['price_per_square_meter'], c='rooms', cmap='c
```

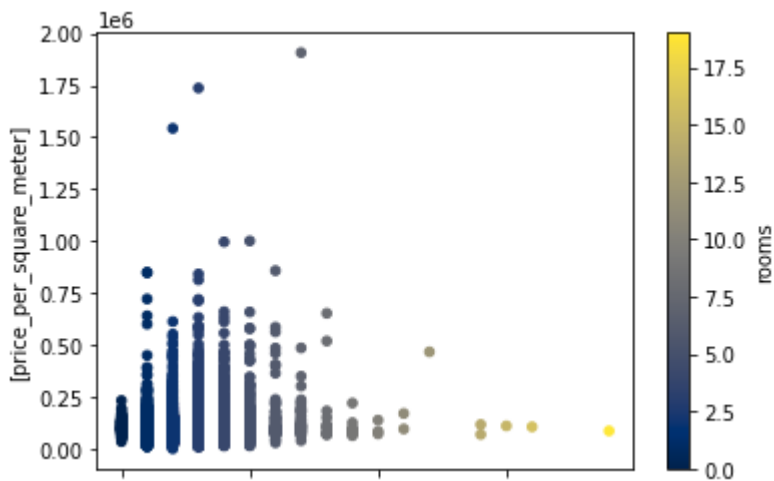


График зависимости цены квадратного метра от количества комнат.

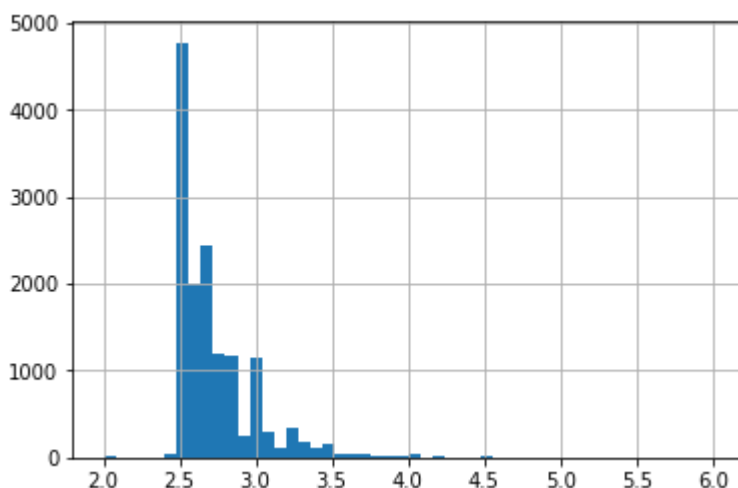
- Объекты свободной планировки отличаются низкой ценой (0 комнат)
- Более 5-комнатные объекты - это, похоже, отдельные дома за пределами городской черты. Поэтому там тоже цена квадратного метра ниже.

6.3 ВЫСОТА ПОТОЛКОВ

```
In [126]: 1 # сводка  
2 data.ceiling_height.describe()
```

```
Out[126]: count    14494.00  
mean         2.73  
std          0.28  
min          2.00  
25%         2.51  
50%         2.65  
75%         2.80  
max          6.00  
Name: ceiling_height, dtype: float64
```

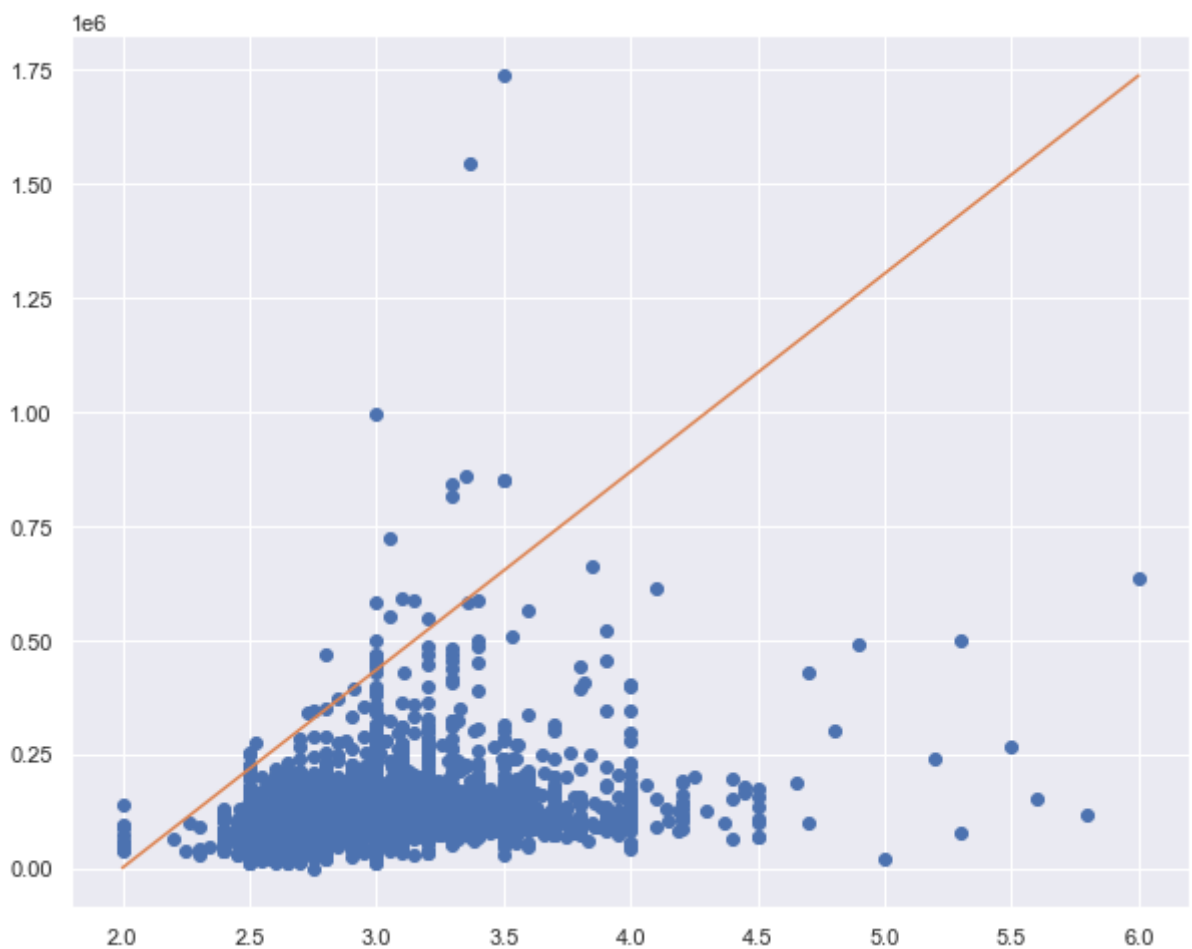
```
In [127]: 1 # гистограмма  
2 data.ceiling_height.hist(bins=50);
```



Здесь наглядно видно преобладание высоты ~2,65.
Есть интересный всплеск по значению ~3,0 метра

In [128]:

```
1 # setup size plot
2 sns.set(rc = {'figure.figsize':(10,8)})
3
4 x_values1=data['ceiling_height']
5 y_values1=data['price_per_square_meter']
6
7 x_values2=[2.25,3.0]
8 y_values2=[15000,200000]
9
10 fig=plt.figure()
11
12 ax=fig.add_subplot(111, label="1")
13 ax2=fig.add_subplot(111, frame_on=False)
14
15 ax.scatter(x_values1, y_values1, color="C0")
16 ax2.plot(x_values2, y_values2, color="C1")
17 ax2.axis('off')
18 plt.show()
```



Зависимость цены за квадратный метр от высоты потолков.

В интервале от ~2,25 до ~3,35 проглядывает положительная зависимость.

(красная линия)

При увеличении высоты потолка увеличивается цена квю метра.

Но эта зависимость не является преобладающей.

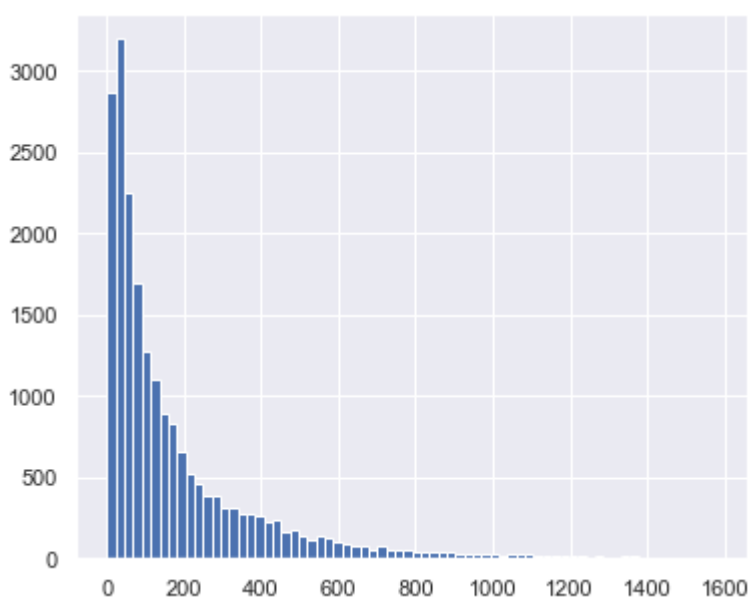
6.4 время продажи

days_exposition


```
In [129]: 1 # сводка
          2 data.days_exposition.describe()
```

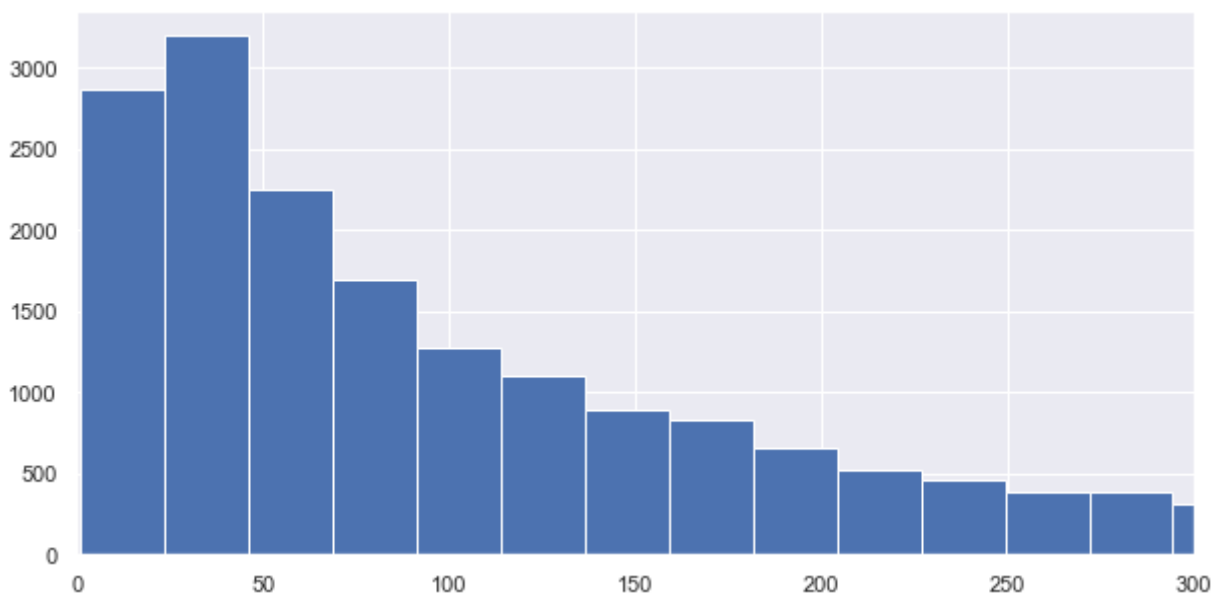
```
Out[129]: count    20518.00
          mean      180.89
          std       219.73
          min        1.00
          25%       45.00
          50%       95.00
          75%      232.00
          max      1580.00
          Name: days_exposition, dtype: float64
```

```
In [130]: 1 # гистограмма
          2 sns.set(rc = {'figure.figsize':(6,5)})
          3 data.days_exposition.hist(bins=70);
```



Отбросим правый "хвост"

```
In [131]: 1 # setup size plot
          2 sns.set(rc = {'figure.figsize':(10,5)})
          3 plt.xlim(0, 300)
          4 data.days_exposition.hist(bins=70);
```



Взаимосвязь времени экспозиции и количества объявлений.

Видно, что рынок весьма динамичный
Объявления в основном держатся до трёх месяцев

```
In [132]: 1 data.days_exposition.median()
```

```
Out[132]: 95.0
```

Это подтверждает **медианное** значение = 95 дней.

```
In [133]: 1 round(data.days_exposition.mean(), 2)
```

```
Out[133]: 180.89
```

А вот среднее арифметическое уехало далеко к полугоду.
Из-за рекордсменов - долгожителей.

6.5 Продавать квартиру можно и быстро и долго

Всех интересуют ответы на вопросы:

- быстро - это сколько дней
- обычно (нормально) - это сколько дней
- долго - это сколько дней
- через сколько дней задуматься, почему не продаётся квартира
- какие факторы и как влияют на скорость продажи

6.5.1 быстро

за месяц

6.5.2 хорошо

за квартал

6.5.3 обычно

дольше, чем полгода, но быстрее, чем ребёнок рождается

6.5.4 долго

больше года

6.5.5 задуматься

это зависит от потребностей продавца.

- какой ориентир для него важен
- быстро
- хорошо
- обычно

7 Отсечём выбивающиеся значения

Основная метрика - цена квадратного метра

7.1 Отсечение сверху

```
In [134]: 1 # цена метра
          2 data.price_per_square_meter.describe()
```

```
Out[134]: count    23699.00
          mean     99421.66
          std      50306.80
          min       111.83
          25%      76585.47
          50%      95000.00
          75%     114256.33
          max     1907500.00
          Name: price_per_square_meter, dtype: float64
```

Из этой справки видно, что максимальное значение цены за квадратный метр превышает наиболее распространённое значение почти в 20 раз.

Для задач анализа сделаем отсечение слишком больших значений.

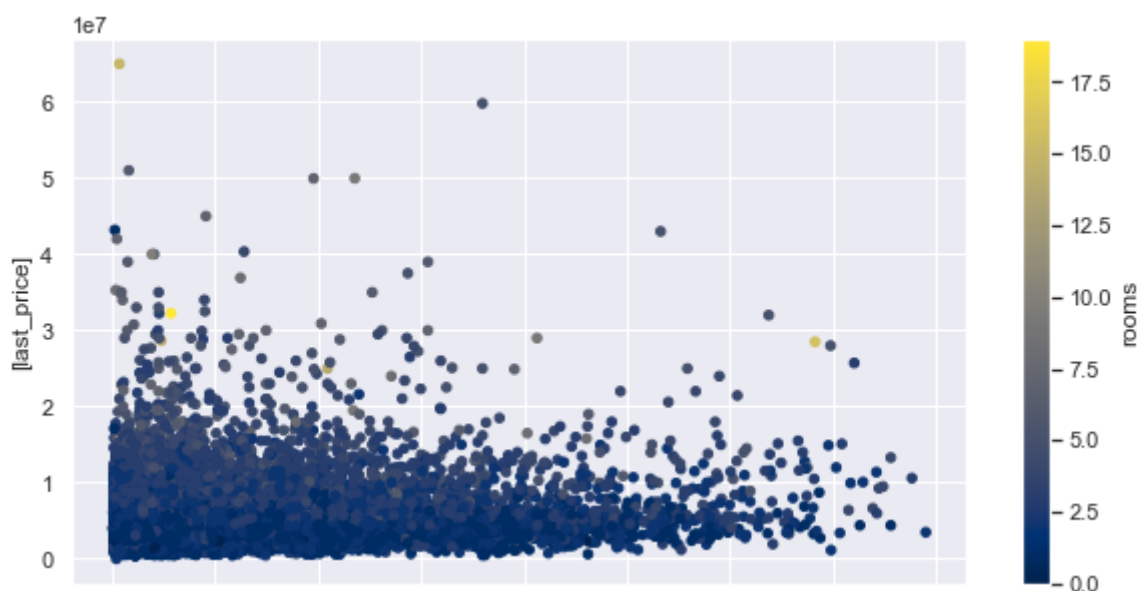
Границу определим как сумму максимальной цифры из часто встречающихся значений (114256) и стандартного отклонения (50306)

назовём скорректированные данные

data_short

```
In [135]: 1 data_short = data.query('price_per_square_meter <= (114256+50306)')
```

```
In [136]: 1 data_short.plot.scatter(x=['days_exposition'], y=['last_price'], c='rooms', cma
```



Связь времени продажи и стоимости

Удивительно, но дорогие объекты не висят на рынке "вечно"

а вполне себе динамично двигаются.

In [137]:

```
1 # setup size plot
2 sns.set(rc = {'figure.figsize':(10,8)})
3
4 x_values1=data_short['days_exposition']
5 y_values1=data_short['price_per_square_meter']
6
7 x_label="Время экспозиции"
8 y_label="Цена кв.м"
9
10 x_values2=[0,1600.0]
11 y_values2=[95000,95000]
12
13 fig=plt.figure()
14
15 ax=fig.add_subplot(111, label="1")
16 ax2=fig.add_subplot(111, frame_on=False, alpha=0.7)
17
18 ax.scatter(x_values1, y_values1, color="C0")
19 ax.set_xlabel("Время экспозиции", color="C0")
20 ax.set_ylabel("Цена кв.м", color="C0")
21
22 ax2.plot(x_values2, y_values2, color="C1", linewidth=8, alpha=0.5, label='Медиа')
23 ax2.axis('off')
24 plt.show()
```



Распределение стоимости квадратного метра и времени экспозиции.
Видна ярковыраженная **медиана** на 95.000 рублей

Но при этом, дольше по времени на рынке задерживаются всё-таки объекты с более высокой ценой квадратного метра.

In [138]:

```
1 data_short.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22625 entries, 0 to 23698
Data columns (total 29 columns):
 #   Column                                Non-Null Count  Dtype  
---  -
 0   total_images                         22625 non-null  int64  
 1   last_price                           22625 non-null  float64
 2   total_area                           22625 non-null  float64
 3   first_day_exposition                 22625 non-null  datetime64[ns]
 4   rooms                                22625 non-null  int64  
 5   ceiling_height                       13771 non-null  float64
 6   floors_total                         22544 non-null  float64
 7   living_area                          20838 non-null  float64
 8   floor                                22625 non-null  int64  
 9   is_apartment                         2650 non-null   object  
10  studio                               22625 non-null  bool    
11  open_plan                            22625 non-null  bool    
12  kitchen_area                         20467 non-null  float64
13  balcony                              22625 non-null  float64
14  locality_name                        22625 non-null  object  
15  airports_nearest                     17109 non-null  float64
16  cityCenters_nearest                  17121 non-null  float64
17  parks_around3000                     17122 non-null  float64
18  parks_nearest                         7332 non-null   float64
19  ponds_around3000                     17122 non-null  float64
20  ponds_nearest                        8349 non-null   float64
21  days_exposition                      19716 non-null  float64
22  price_per_square_meter                22625 non-null  float64
23  day_of_the_week                       22625 non-null  int64  
24  month_of_publication                  22625 non-null  int64  
25  year_of_publication                   22625 non-null  int64  
26  apartment_floor                       22625 non-null  object  
27  ratio_living_total_area               20838 non-null  float64
28  ratio_kitchen_total_area              20467 non-null  float64
dtypes: bool(2), datetime64[ns](1), float64(17), int64(6), object(3)
memory usage: 4.9+ MB
```

In [139]:

```
1 # цена метра
2 data_short.price_per_square_meter.describe()
```

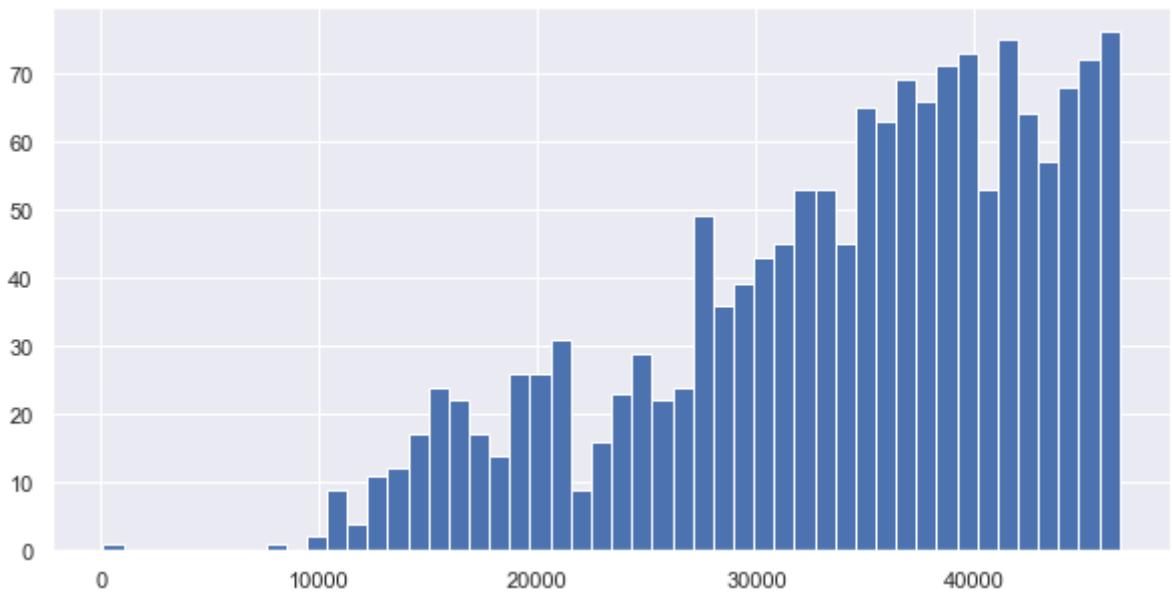
```
Out[139]: count    22625.00
mean      92673.80
std       28750.33
min        111.83
25%       75384.62
50%       93439.36
75%      110807.11
max      164549.65
Name: price_per_square_meter, dtype: float64
```

7.2 мы отрезали меньше 5 % данных

рассмотрим данные снизу

In [140]:

```
1 # цена метра
2 sns.set(rc = {'figure.figsize':(10,5)})
3 data_short.query('price_per_square_meter <= (75384-28750)').price_per_square_me
```



7.2.1 Оставим данные с ценой квадратного метра более 15000

In [141]:

```
1 data_short = data_short.query('price_per_square_meter >= 15000')
```

In [142]:

```
1 # цена метра
2 data_short.price_per_square_meter.describe()
```

```
Out[142]: count    22568.00
mean      92875.97
std       28503.19
min       15000.00
25%       75557.75
50%       93520.94
75%      110833.33
max      164549.65
Name: price_per_square_meter, dtype: float64
```

7.2.2 Отрезалось совсем чуть данных

8 Какие факторы больше всего влияют на стоимость квартиры?

In [143]: ▶

```
1 data_short.info()
```

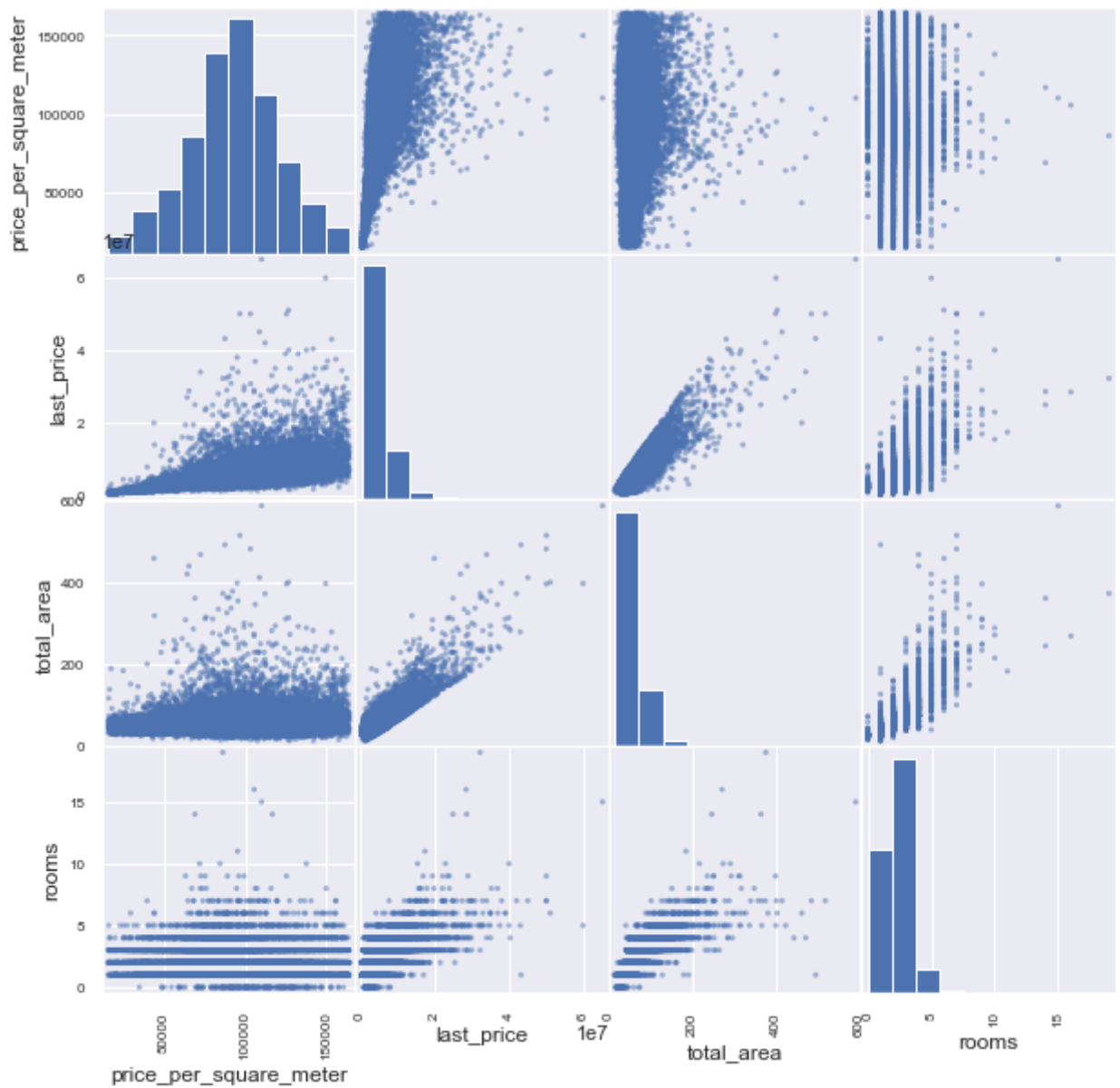
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22568 entries, 0 to 23698
Data columns (total 29 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   total_images                          22568 non-null  int64
1   last_price                            22568 non-null  float64
2   total_area                            22568 non-null  float64
3   first_day_exposition                 22568 non-null  datetime64[ns]
4   rooms                                22568 non-null  int64
5   ceiling_height                       13754 non-null  float64
6   floors_total                         22487 non-null  float64
7   living_area                          20794 non-null  float64
8   floor                                22568 non-null  int64
9   is_apartment                         2648 non-null   object
10  studio                               22568 non-null  bool
11  open_plan                            22568 non-null  bool
12  kitchen_area                         20424 non-null  float64
13  balcony                              22568 non-null  float64
14  locality_name                        22568 non-null  object
15  airports_nearest                    17108 non-null  float64
16  cityCenters_nearest                 17120 non-null  float64
17  parks_around3000                    17121 non-null  float64
18  parks_nearest                       7331 non-null   float64
19  ponds_around3000                    17121 non-null  float64
20  ponds_nearest                       8349 non-null   float64
21  days_exposition                     19669 non-null  float64
22  price_per_square_meter               22568 non-null  float64
23  day_of_the_week                     22568 non-null  int64
24  month_of_publication                 22568 non-null  int64
25  year_of_publication                  22568 non-null  int64
26  apartment_floor                     22568 non-null  object
27  ratio_living_total_area              20794 non-null  float64
28  ratio_kitchen_total_area             20424 non-null  float64
dtypes: bool(2), datetime64[ns](1), float64(17), int64(6), object(3)
memory usage: 4.9+ MB
```

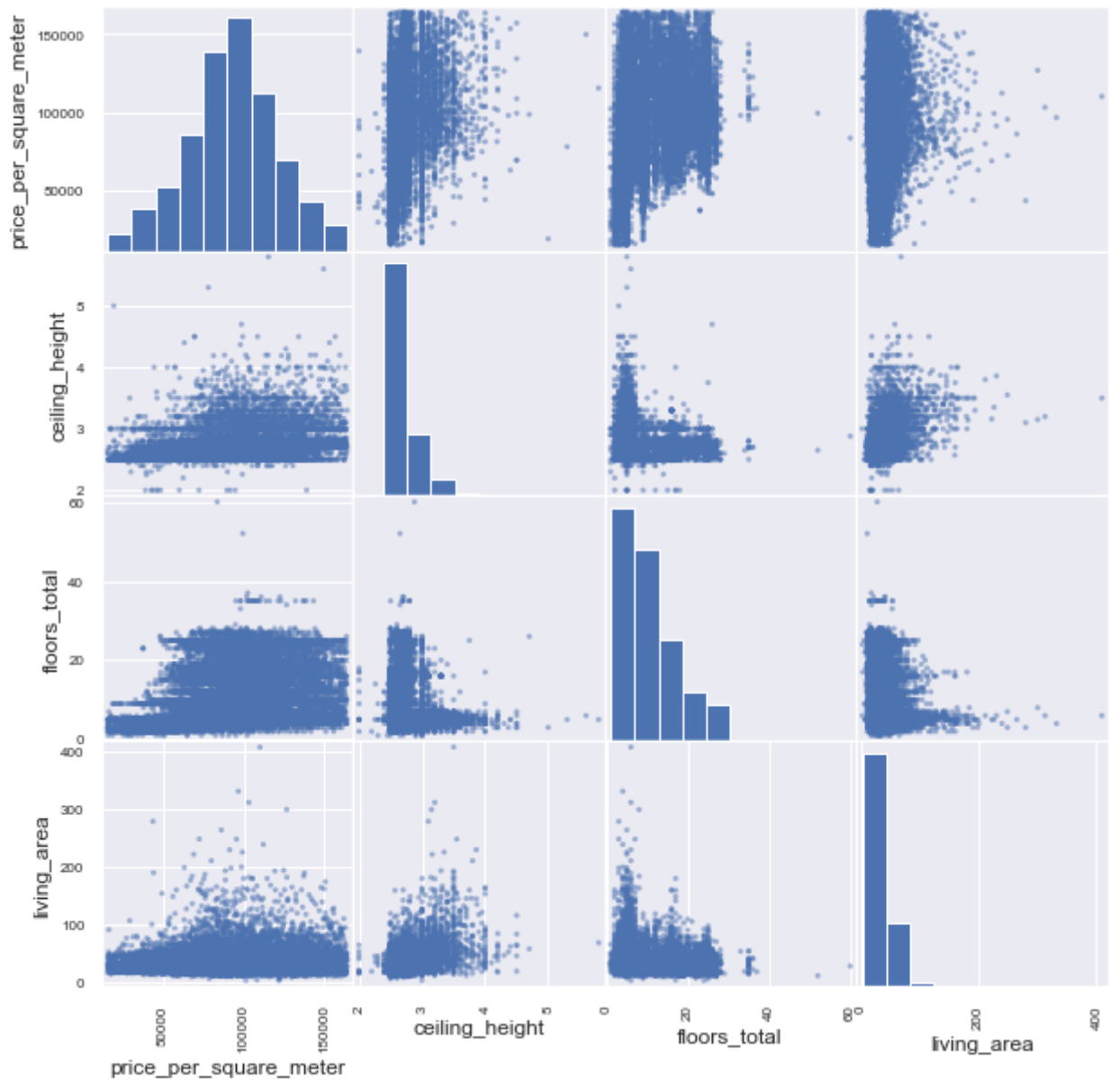
Какие параметры имеют линейные зависимости между собой?

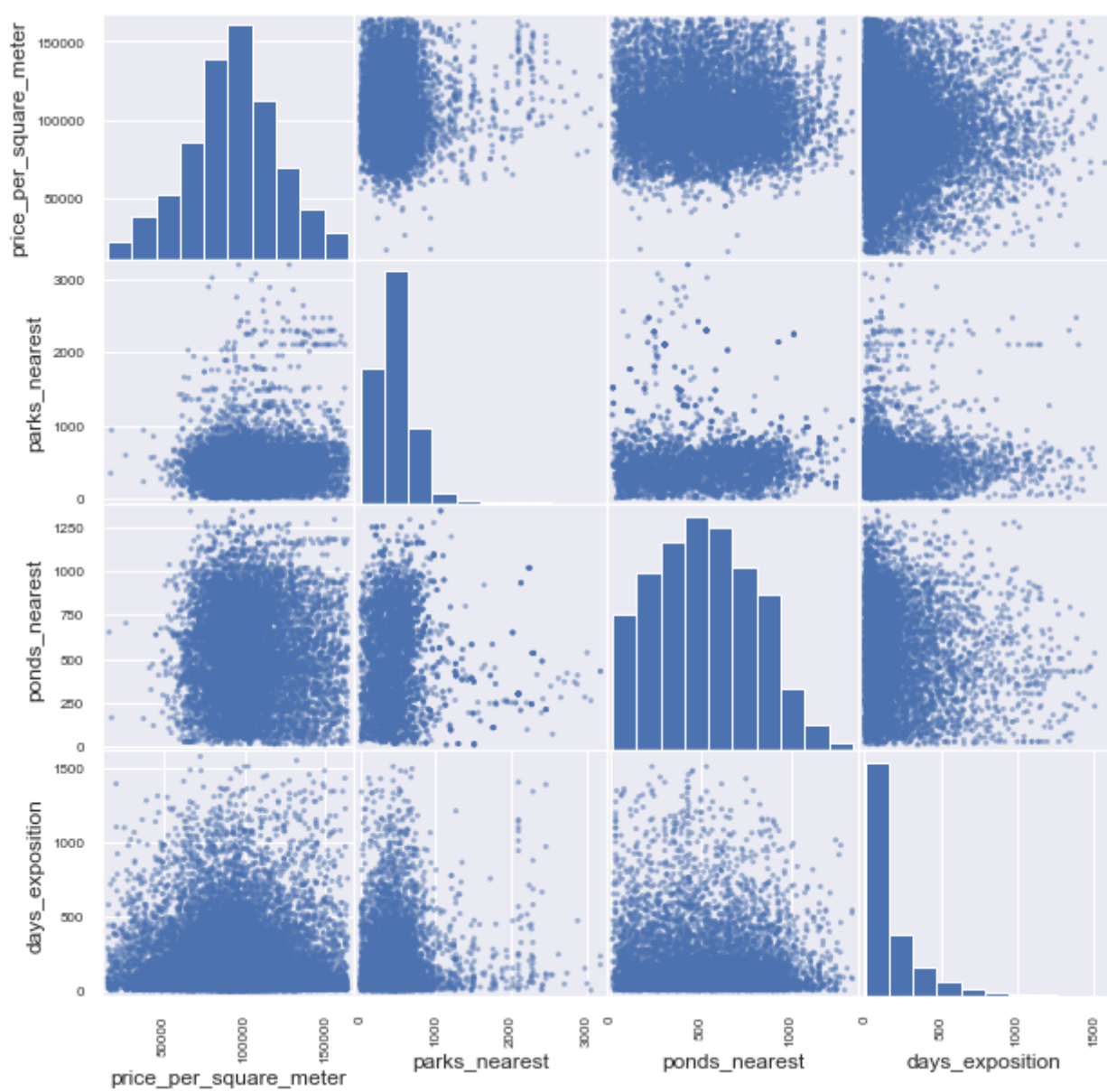
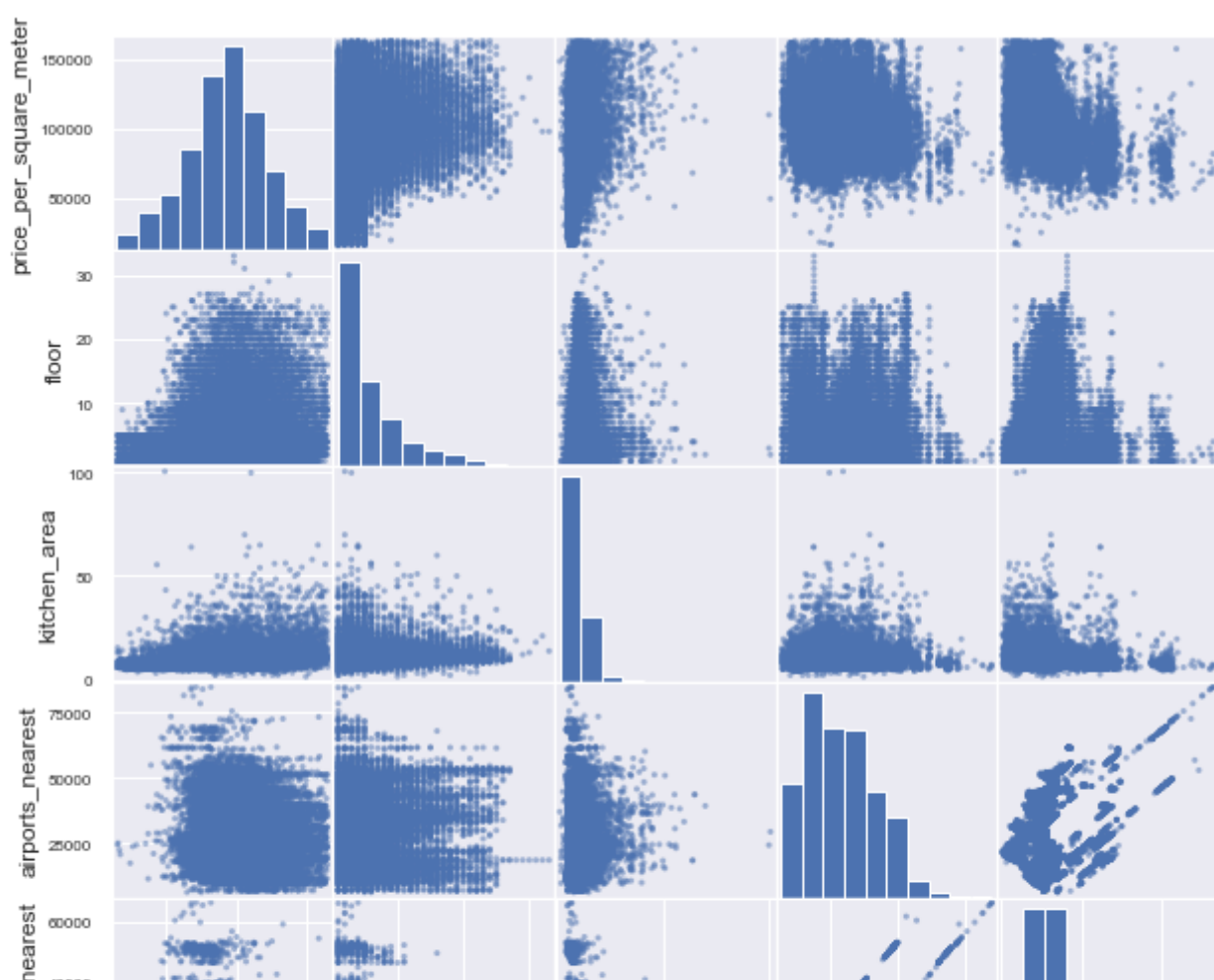
Посмотрим на матрицы:

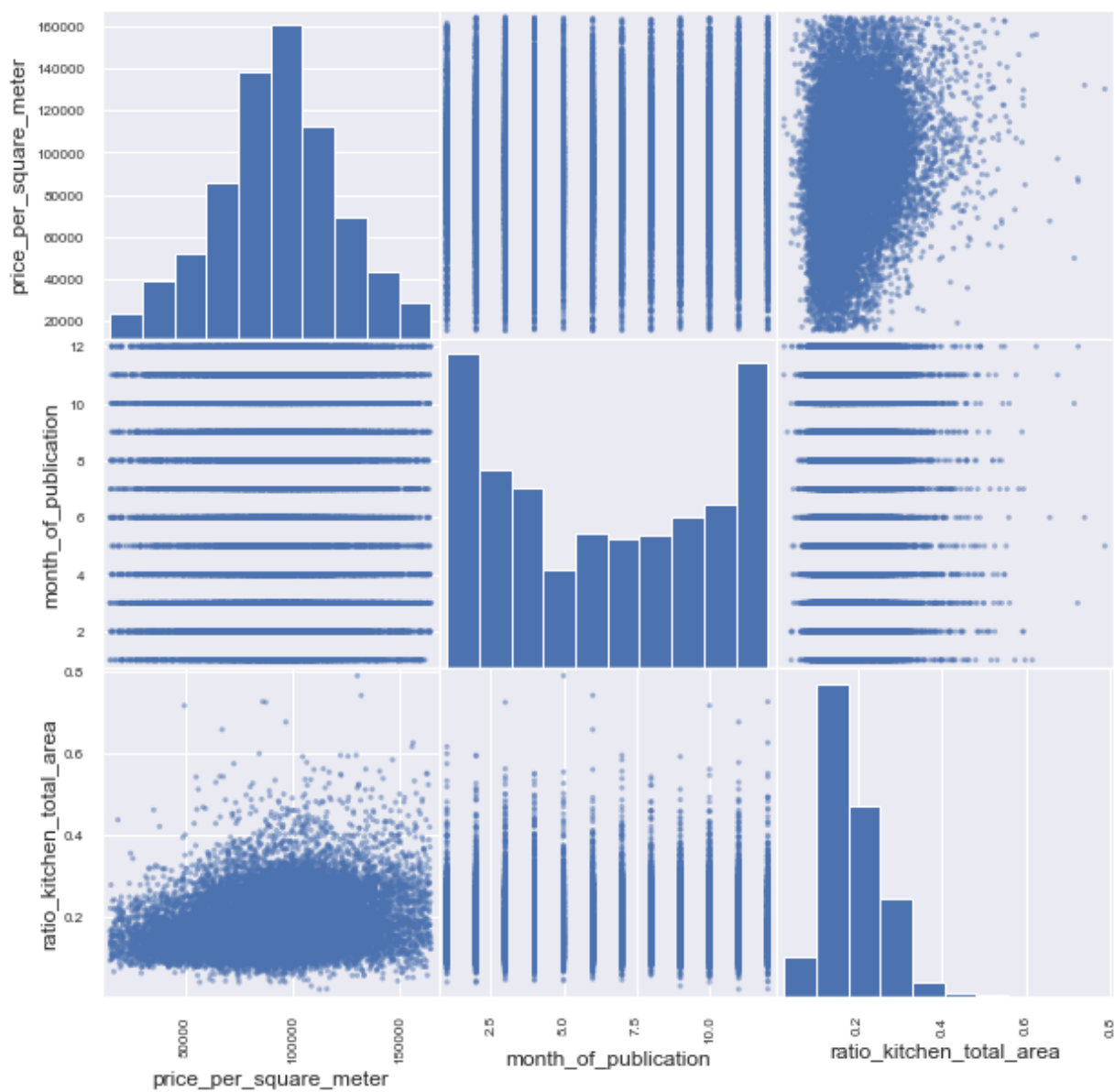
In [144]:

```
1 sns.set(rc = {'figure.figsize':(10,10)})
2 pd.plotting.scatter_matrix(data_short[['price_per_square_meter', 'last_price',
3 pd.plotting.scatter_matrix(data_short[['price_per_square_meter', 'ceiling_height',
4 pd.plotting.scatter_matrix(data_short[['price_per_square_meter', 'floor', 'kitchen_area',
5 pd.plotting.scatter_matrix(data_short[['price_per_square_meter', 'parks_nearest',
6 pd.plotting.scatter_matrix(data_short[['price_per_square_meter', 'apartment_floor',
```









Визуальный анализ выявил параметры, умеющие прямое (или размытое прямое) влияние на цену квадратного метра:

- ceiling_height
- floors_total
- floor
- last_price
- kitchen_area
- cityCenters_nearest
- airports_nearest

8.0.1 * Расстояния переведем в КМ и математически округлим

```
In [145]: ► 1 data_short['airports_nearest'] = round(data_short['airports_nearest'] / 1000)
          2 data_short['cityCenters_nearest'] = round(data_short['cityCenters_nearest'] / 1000)
```

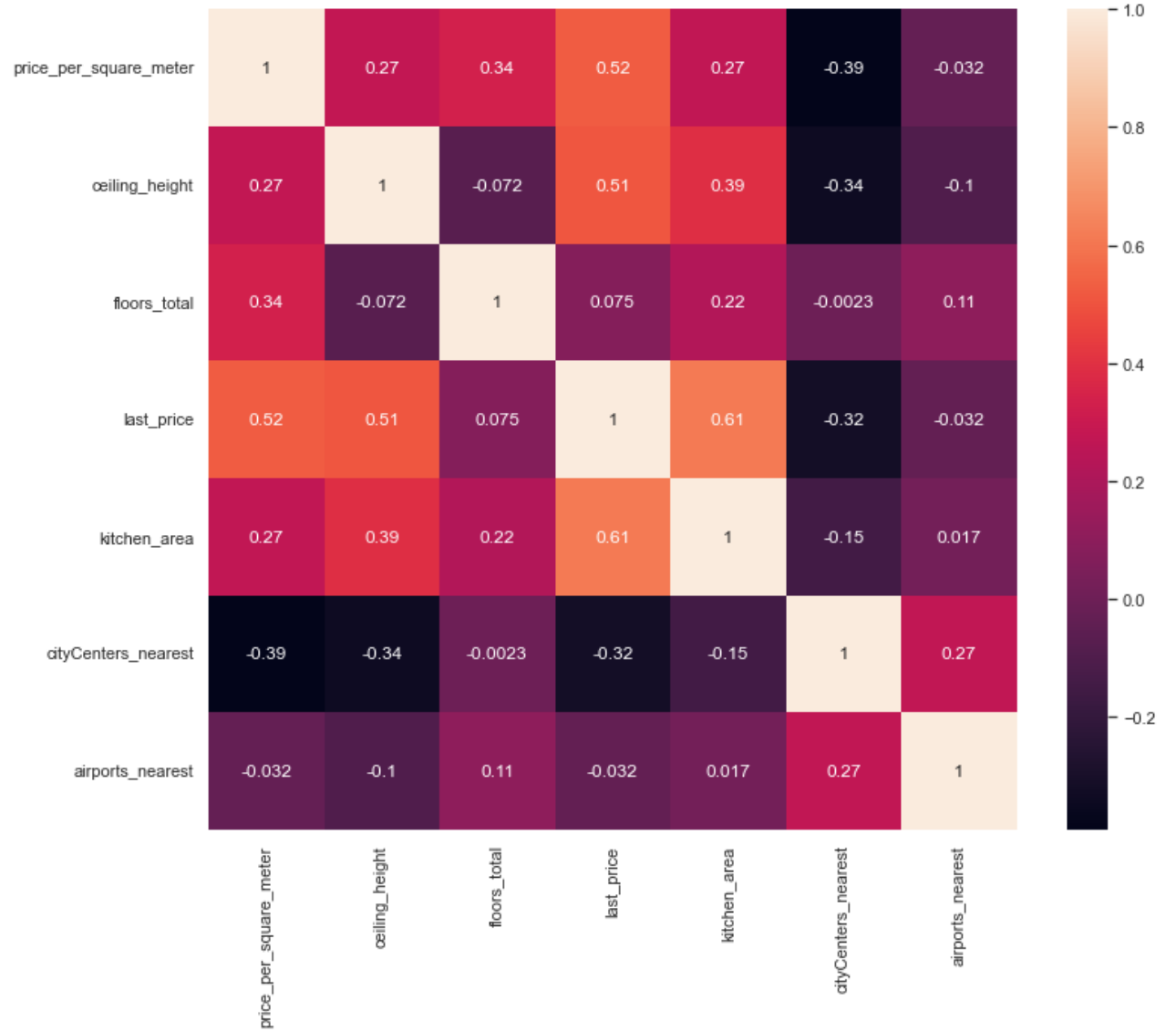
Посмотрим на эти параметры сквозь призму корреляции

```
In [146]: 1 data_corr = data_short[['price_per_square_meter', 'ceiling_height', 'floors_total', 'last_price', 'kitchen_area', 'cityCenters_nearest', 'airports_nearest']]
          2 data_corr
```

Out[146]:

	price_per_square_meter	ceiling_height	floors_total	last_price	kitchen_area	cityCenters_nearest	airports_nearest
price_per_square_meter	1.00	0.27	0.34	0.52	0.27	-0.39	-0.03
ceiling_height	0.27	1.00	-0.07	0.51	0.39	-0.34	-0.10
floors_total	0.34	-0.07	1.00	0.07	0.22	-0.00	0.11
last_price	0.52	0.51	0.07	1.00	0.61	-0.32	-0.03
kitchen_area	0.27	0.39	0.22	0.61	1.00	-0.15	0.02
cityCenters_nearest	-0.39	-0.34	-0.00	-0.32	-0.15	1.00	-0.27
airports_nearest	-0.03	-0.10	0.11	-0.03	0.02	-0.27	1.00

```
In [147]: 1 sns.set(rc = {'figure.figsize':(12,10)})
          2 sns.heatmap(data_corr, annot=True);
```



Сильных зависимостей не выявлено.

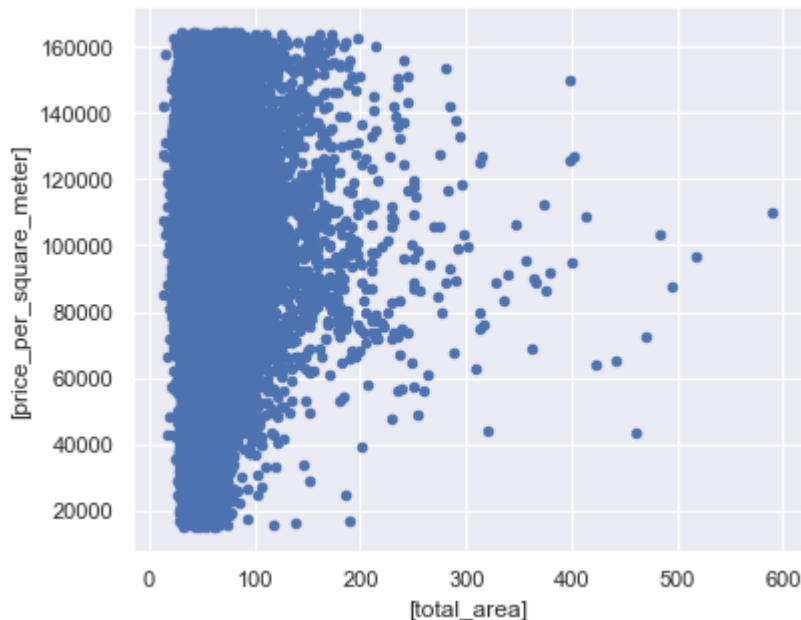
Средняя зависимость - Цена объекта last_price

Слабая обратная зависимость - расстояние от Центра.
На таком же уровне - положительная зависимость от этажности здания.

8.1 площадь квартиры

```
In [148]: 1 plt.figure(figsize=(6, 5));  
2 sns.set(rc = {'figure.figsize':(6,5)});  
3 data_short.plot.scatter(x=['total_area'], y=['price_per_square_meter'], color='')
```

<Figure size 432x360 with 0 Axes>



```
In [149]: 1 data_short[['total_area', 'price_per_square_meter']].corr()
```

Out[149]:

	total_area	price_per_square_meter
total_area	1.00	0.08
price_per_square_meter	0.08	1.00

Математическая прямая зависимость **отсутствует**

8.2 более 100 метров

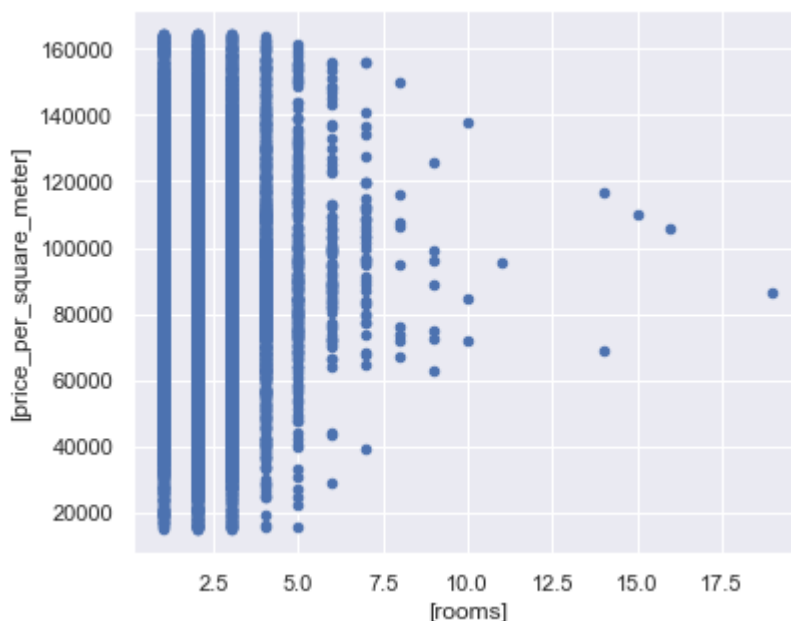
для квартир более 100 метров
постепенно повышается порог нижнего значения
цены квадратного метра

- предложений квартир более 200 метров становится меньше
- их цены чаще в сегменте средней и высокой стоимости

8.3 число комнат

In [150]:

```
1 # выведем на экран значения без "0"  
2 data_short.query('rooms > 0').plot.scatter(x=['rooms'], y=['price_per_square_me
```



In [151]:

```
1 data_short[['rooms', 'price_per_square_meter']].corr()
```

Out[151]:

	rooms	price_per_square_meter
rooms	1.00	-0.10
price_per_square_meter	-0.10	1.00

Математическая прямая зависимость **отсутствует**

8.4 есть небольшая зависимость от числа комнат

цена квадратного метра для квартир
с количеством комнат более ДВУХ
имеет тенденцию постепенного роста
засчет того, что поднимается нижний порог в выборке
вероятно, это также связано с тем, что квартиры с большим количеством комнат
чаще располагаются в более престижных домах
и в более престижных районах

8.5 удалённость от центра

Сгруппируем данные по каждому километру.

In [152]:

```
1 data_group_km_center = data_short.groupby(['cityCenters_nearest'])['price_per_s
```

Посмотрим рапределение значений price_per_square_meter и линию медианных значений

In [153]:

```
1 # setup size plot
2 sns.set(rc = {'figure.figsize':(15,10)})
3
4 x_values1=data_short['cityCenters_nearest']
5 y_values1=data_short['price_per_square_meter']
6
7 x_label="Расстояние до центра"
8 y_label="Цена кв.м"
9
10 fig=plt.figure()
11
12 ax=fig.add_subplot(111, label="1")
13 ax2=fig.add_subplot(111, frame_on=False, alpha=0.7)
14
15 ax.scatter(x_values1, y_values1, color="C0")
16 ax.set_xlabel("Время экспозиции", color="C0")
17 ax.set_ylabel("Цена кв.м", color="C0")
18
19 ax2.plot(data_group_km_center, color="C1", linewidth=5, label='Медиана')
20 ax2.axis('off')
21 plt.show()
```



На графике видно два резких спада.
Совсем рядом с центром и чуть по-дальше.
Посмотрим на медиану поближе.
Ограничим интервал до 10 км.

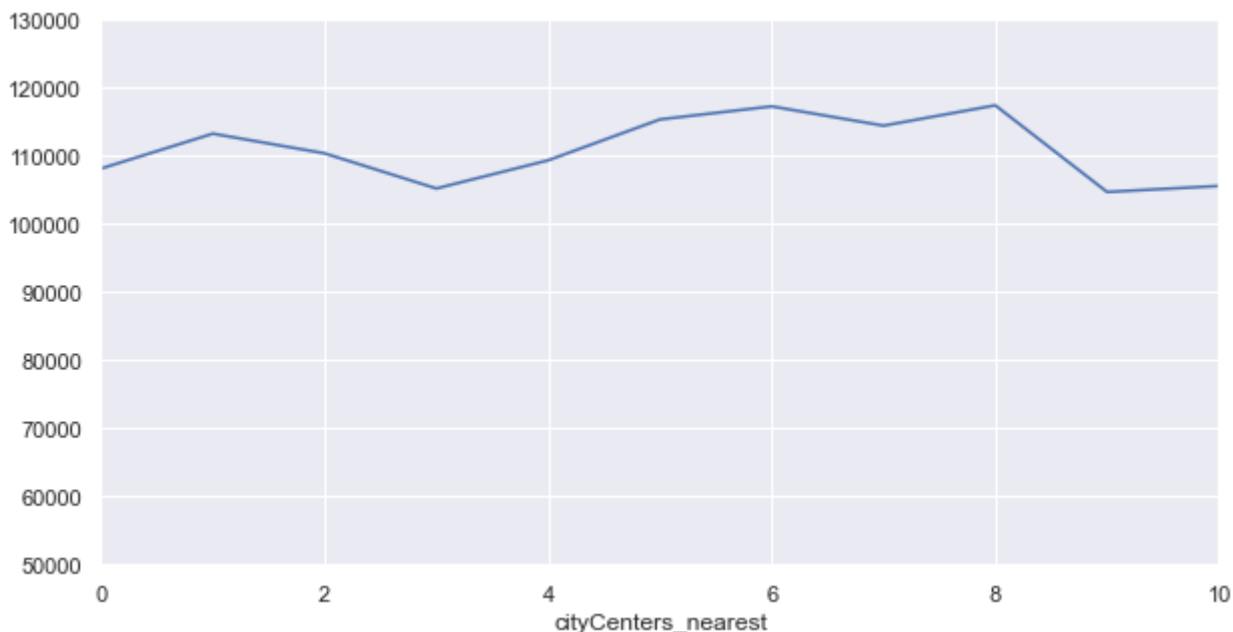
Также явно видна граница **города** ~23 км.

Затем следует всплеск ближнего загорода.

А потом - скачки и падения загородной недвижимости.

In [154]:

```
1 sns.set(rc = {'figure.figsize':(10,5)})
2 plt.xlim(0, 10)
3 plt.ylim(50000, 130000)
4 data_group_km_center.plot();
```



Центр города принимаем за 9 км

8.6 Центр города <= 8,5 км

8.6.1 Также явно видна граница города ~23 км.

Затем следует всплеск ближнего загорода l_j **~27 км.**

А потом - скачки и падения загородной недвижимости.

8.7 на каком этаже

In [155]:

```
1 data_short.apartment_floor.describe()
```

```
Out[155]: count      22568
unique         3
top            другой
freq          16576
Name: apartment_floor, dtype: object
```

среднее

In [156]:

```
1 round(data_short.query('apartment_floor == "первый"]').price_per_square_meter.mean())
```

```
Out[156]: 79555.8
```

```
In [157]: 1 round(data_short.query('apartment_floor == "последний").price_per_square_meter
Out[157]: 84972.79
```

```
In [158]: 1 round(data_short.query('apartment_floor == "другой").price_per_square_meter.me
Out[158]: 96661.62
```

медиана

```
In [159]: 1 round(data_short.query('apartment_floor == "первый").price_per_square_meter.me
Out[159]: 80952.38
```

```
In [160]: 1 round(data_short.query('apartment_floor == "последний").price_per_square_meter
Out[160]: 86124.86
```

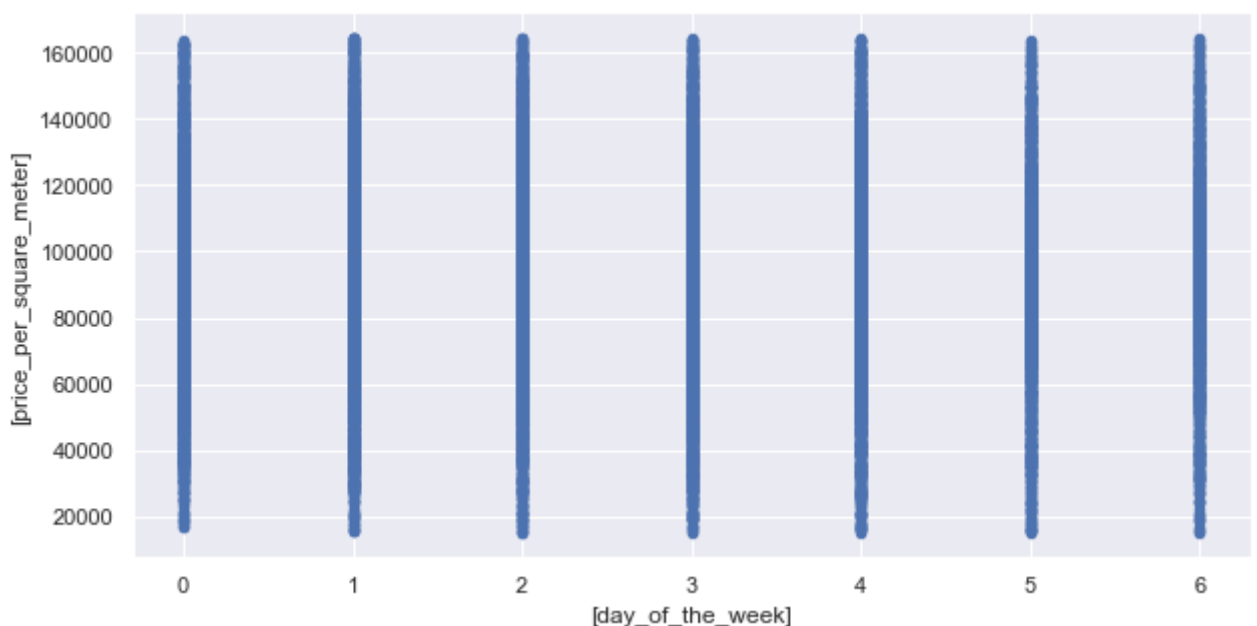
```
In [161]: 1 round(data_short.query('apartment_floor == "другой").price_per_square_meter.me
Out[161]: 96781.01
```

8.7.1 этаж влияет на цену

- первый этаж - дешевле
- другой этаж - дороже
- последний этаж дешевле другого, но дороже первого

8.8 день недели

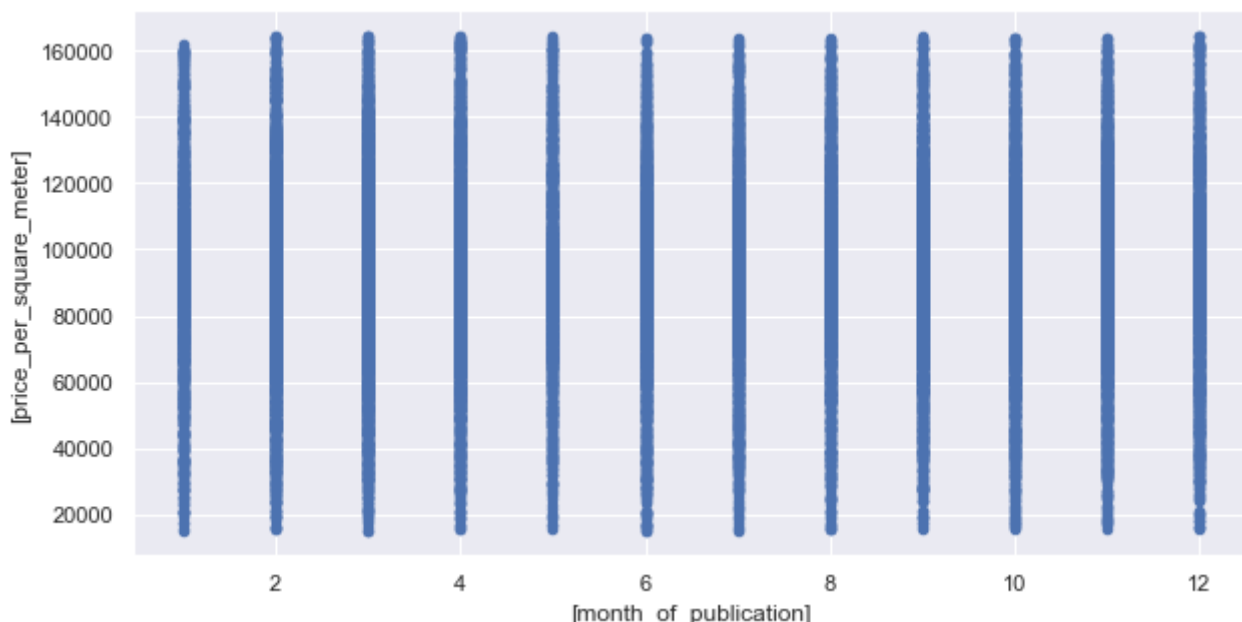
```
In [162]: 1 data_short.plot.scatter(x=['day_of_the_week'], y=['price_per_square_meter'], co
```



8.9 не влияет

8.10 месяц

In [163]: `data_short.plot.scatter(x=['month_of_publication'], y=['price_per_square_meter'])`



8.10.1 не влияет

8.11 год

In [164]: `data_short.plot.scatter(x=['year_of_publication'], y=['price_per_square_meter'])`



8.12 в древности на электронных площадках размещали более дорогие объекты

для анализа динамики цен с 2016 по 2019 года надо более детально сегментировать выборки
однако у нас нет точной адресной информации по объявлениям
поэтому корректного сравнения мы не получим
а в среднем по рынку видимых изменений не заметно

8.13 10 населённых пунктов с наибольшим числом объявлений

```
In [165]: 1 data_short['locality_name'].value_counts().head(10)
```

```
Out[165]: Санкт-Петербург      14664
           поселок Мурино        556
           поселок Шушары        440
           Всеволожск           397
           Пушкин                363
           Колпино               338
           поселок Парголово     327
           Гатчина               305
           деревня Кудрово       299
           Выборг                237
           Name: locality_name, dtype: int64
```

```
In [166]: 1 # 6 списков
           2 locality_short = data_short['locality_name'].value_counts()
```

```
In [167]: 1 locality_short
```

```
Out[167]: Санкт-Петербург      14664
           поселок Мурино        556
           поселок Шушары        440
           Всеволожск           397
           Пушкин                363
           ...
           деревня Курковицы      1
           деревня Пчева          1
           деревня Мануйлово      1
           деревня Бор            1
           поселок Дзержинского   1
           Name: locality_name, Length: 324, dtype: int64
```

```
In [168]: 1 # срез 10 значений
           2 locality_short = locality_short[0:10]
```

```
In [169]: 1 locality_short
```

```
Out[169]: Санкт-Петербург      14664
           поселок Мурино        556
           поселок Шушары        440
           Всеволожск           397
           Пушкин                363
           Колпино               338
           поселок Парголово     327
           Гатчина               305
           деревня Кудрово       299
           Выборг                237
           Name: locality_name, dtype: int64
```

средняя цена квадратного метра по списку

```
In [170]: ► 1 reiting_locality = pd.DataFrame()
2 for i in range(len(locality_short)):
3     name= locality_short.index[i]
4     qty= locality_short[i]
5     mean= data_short.query('locality_name == @name').price_per_square_meter.mea
6     print()
7     print('B ', name, "всего", qty, "объявлений со средней ценой метра", mean)
8
9     new_row = {'city':name, 'qty':qty, 'mean':mean}
10
11     reiting_locality = reiting_locality.append(new_row, ignore_index=True)
12
13
14
```

```
B Санкт-Петербург всего 14664 объявлений со средней ценой метра 105708.7404492590
9
B поселок Мурино всего 556 объявлений со средней ценой метра 85681.76260114645
B поселок Шушары всего 440 объявлений со средней ценой метра 78677.36421675135
B Всеволожск всего 397 объявлений со средней ценой метра 67214.25263468112
B Пушкин всего 363 объявлений со средней ценой метра 101607.98440469746
B Колпино всего 338 объявлений со средней ценой метра 75424.57909803944
B поселок Парголово всего 327 объявлений со средней ценой метра 90175.91345801107
B Гатчина всего 305 объявлений со средней ценой метра 69126.76188219966
B деревня Кудрово всего 299 объявлений со средней ценой метра 92473.54757579978
B Выборг всего 237 объявлений со средней ценой метра 58141.909153318615
```

```
In [171]: ► 1 reiting_locality
```

Out[171]:

	city	qty	mean
0	Санкт-Петербург	14664	105708.74
1	поселок Мурино	556	85681.76
2	поселок Шушары	440	78677.36
3	Всеволожск	397	67214.25
4	Пушкин	363	101607.98
5	Колпино	338	75424.58
6	поселок Парголово	327	90175.91
7	Гатчина	305	69126.76
8	деревня Кудрово	299	92473.55
9	Выборг	237	58141.91

In [172]:

1 reiting_locality.describe()

Out[172]:

	qty	mean
count	10.00	10.00
mean	1792.60	82423.28
std	4523.40	15429.79
min	237.00	58141.91
25%	310.50	70701.22
50%	350.50	82179.56
75%	429.25	91899.14
max	14664.00	105708.74

8.14 Выборг - с самой низкой ценой

58141.91 за квадратный метр

8.15 Санкт-Петербург - с самой высокой ценой

105708.74 за квадратный метр

8.16 Определение центральной зоны

In [173]:

1 *# только Питер*
2 piter = data_short.query('locality_name == "Санкт-Петербург"')

In [174]:

1 piter.describe()

Out[174]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floo
count	14664.00	14664.00	14664.00	14664.00	9386.00	14599.00	13637.00	14664.00
mean	10.04	6498151.28	61.54	2.14	2.74	11.52	35.45	6.20
std	5.67	3946677.03	33.81	1.12	0.28	6.42	21.64	4.90
min	0.00	1190000.00	13.00	0.00	2.00	1.00	2.00	1.00
25%	6.00	4150000.00	41.30	1.00	2.55	5.00	19.30	3.00
50%	10.00	5300000.00	53.80	2.00	2.65	9.00	31.00	5.00
75%	14.00	7500000.00	71.40	3.00	2.80	16.00	43.10	9.00
max	50.00	65000000.00	590.00	19.00	5.80	52.00	409.00	33.00

Расстояния переводим в километры и округляем математически

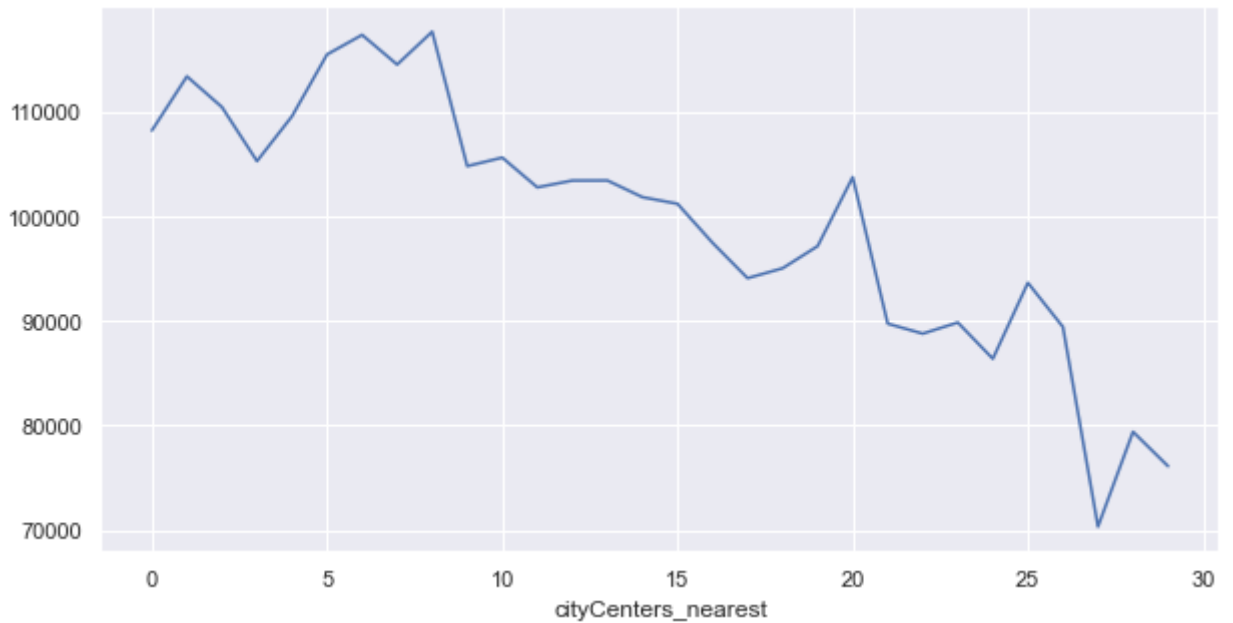
```
In [175]: 1 # округляем до целых значений расстояние до центра
2 piter = piter.reset_index(drop=True)
3 piter['parks_nearest'] = round(piter['parks_nearest'] / 1000)
4 piter['ponds_nearest'] = round(piter['ponds_nearest'] / 1000)

In [176]: 1 # медианная цена для каждого километра
2 piter_group_distance = piter.groupby('cityCenters_nearest')['price_per_square_m

In [177]: 1 piter_group_distance
```

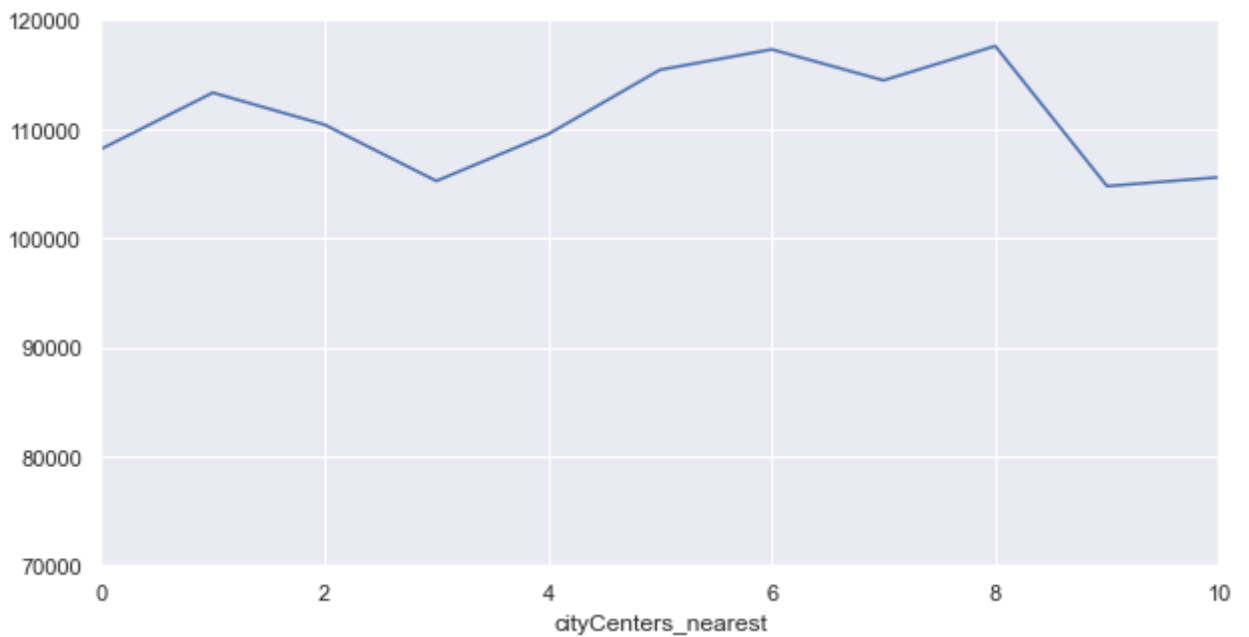
```
Out[177]: cityCenters_nearest
0.00      108163.27
1.00      113333.33
2.00      110397.70
3.00      105236.84
4.00      109530.81
5.00      115431.03
6.00      117300.00
7.00      114465.76
8.00      117607.31
9.00      104761.90
10.00     105574.37
11.00     102731.98
12.00     103383.46
13.00     103389.83
14.00     101793.73
15.00     101162.14
16.00      97435.71
17.00      94051.45
18.00      95000.00
19.00      97117.96
20.00     103703.69
21.00      89705.88
22.00      88753.00
23.00      89815.53
24.00      86353.29
25.00      93612.16
26.00      89393.94
27.00      70312.50
28.00      79372.09
29.00      76084.73
Name: price_per_square_meter, dtype: float64
```

```
In [178]: 1 piter_group_distance.plot(kind='line');
```



```
In [179]: 1 data['day_of_the_week'] = data['first_day_exposition'].dt.weekday
```

```
In [180]: 1 sns.set(rc = {'figure.figsize':(10,5)})  
2 plt.xlim(0,10)  
3 plt.ylim(70000, 120000)  
4 piter_group_distance.plot();
```



8.17 центр до 8500 м (8,5 км)

```
In [181]: 1 # центр города  
2 centre = piter.query('cityCenters_nearest <= 8500')
```



```
In [182]: 1 data_short.price_per_square_meter.describe()
```

```
Out[182]: count      22568.00  
mean       92875.97  
std        28503.19  
min        15000.00  
25%        75557.75  
50%        93520.94  
75%       110833.33  
max       164549.65  
Name: price_per_square_meter, dtype: float64
```

```
In [183]: 1 centre.price_per_square_meter.describe()
```

```
Out[183]: count      14617.00  
mean     105665.70  
std       21608.86  
min       15345.27  
25%       90000.00  
50%      102631.58  
75%      119411.76  
max      164549.65  
Name: price_per_square_meter, dtype: float64
```

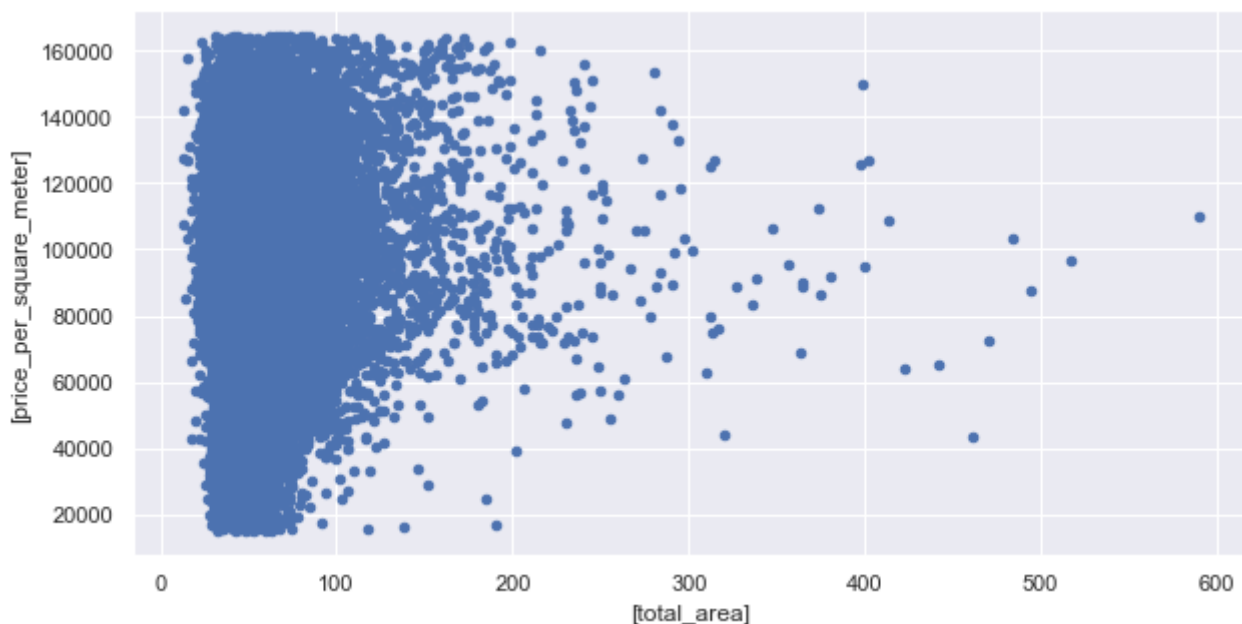
8.18 стоимость квадратного метра

в целом выше в центре города

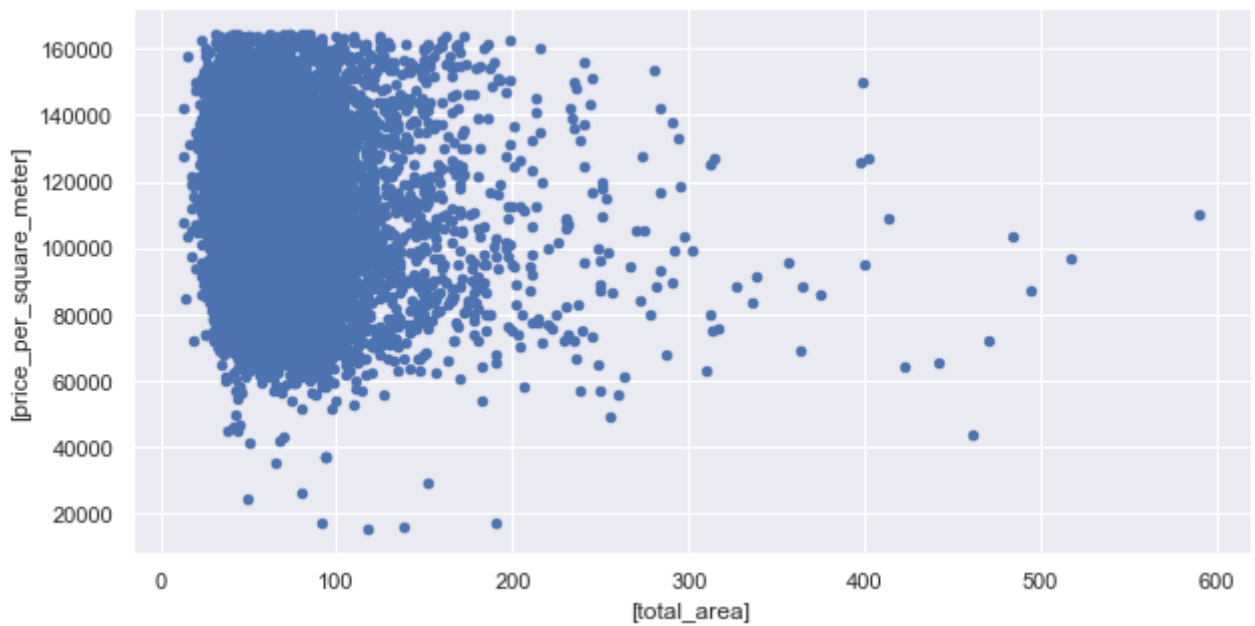
- больше среднее
- больше максимальная
- в целом все точки больше

8.19 площадь квартиры

```
In [184]: 1 # no всей базе  
2 data_short.plot.scatter(x=['total_area'], y=['price_per_square_meter'], color='')
```



```
1 # по центру
2 centre.plot.scatter(x=['total_area'], y=['price_per_square_meter'], color='C0')
```



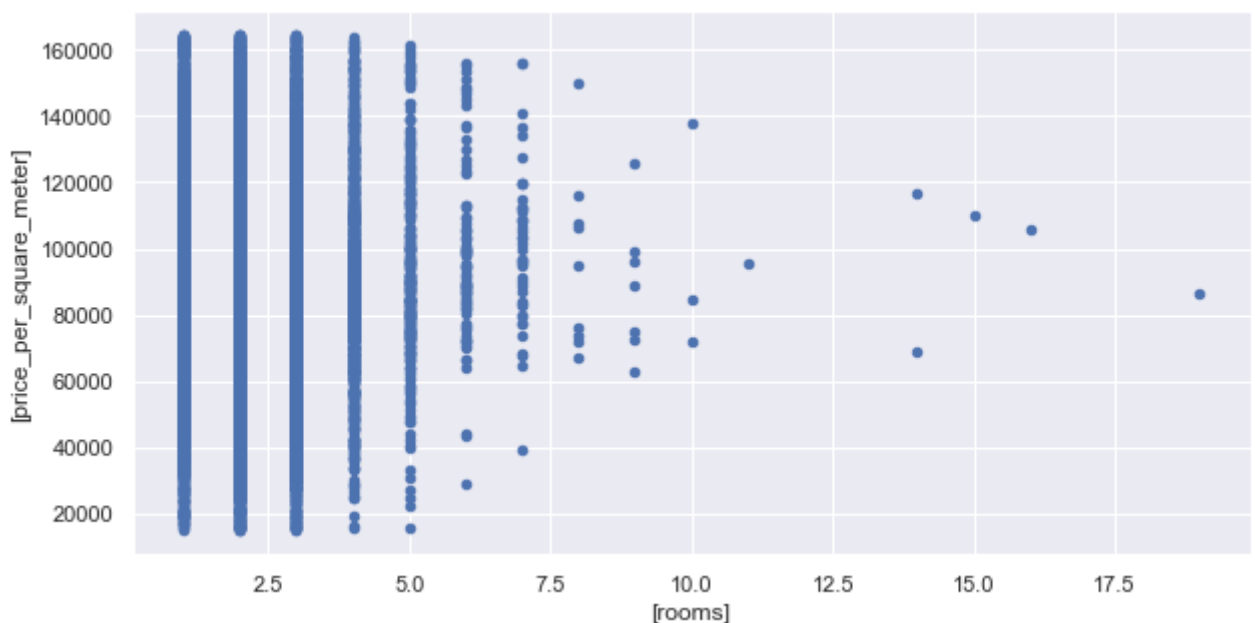
8.20 более 60.000 нижний порог

стоимости квадратного метра в центре города

- при этом зависимость между общей площадью квартиры и ценой квадратного метра вполне похожа на то же самое в основной базе (за вычетом дешевых вариантов)

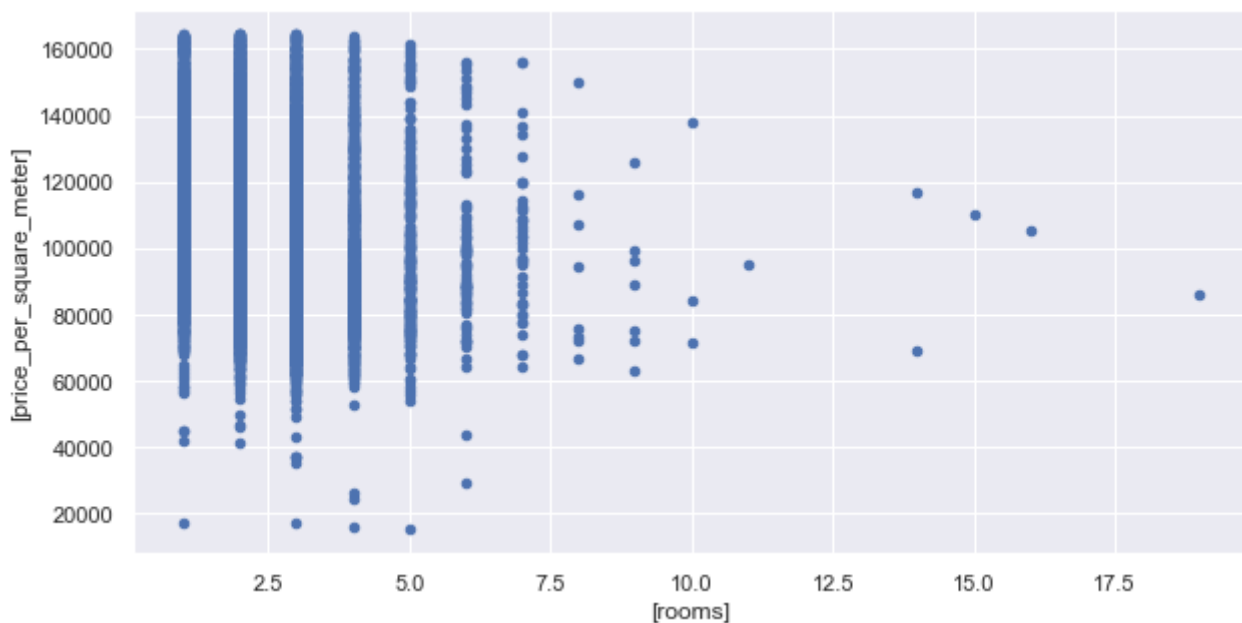
8.21 число комнат

```
1 # вся база
2 sns.set(rc = {'figure.figsize':(10,5)})
3 data_short.query('rooms > 0').plot.scatter(x=['rooms'], y=['price_per_square_me
```



In [187]:

```
1 # центр
2 centre.query('rooms > 0').plot.scatter(x=['rooms'], y=['price_per_square_meter'])
```



8.22 более 60.000 нижний порог

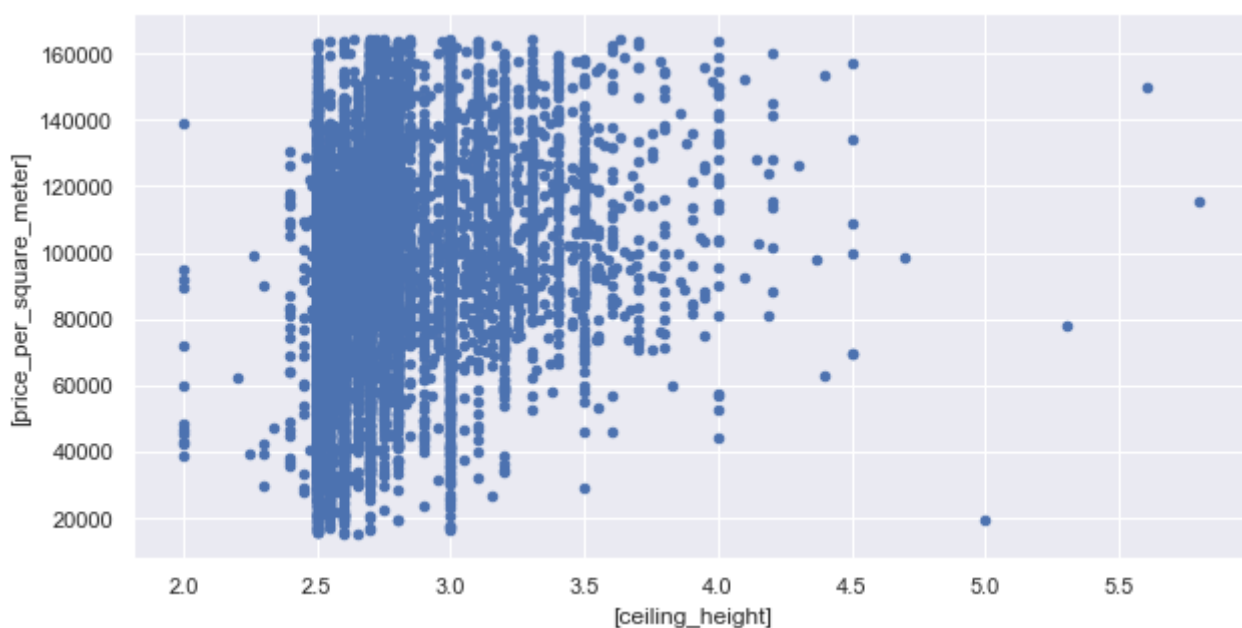
стоимости квадратного метра в центре города

- при этом зависимость между числом комнат и ценой квадратного метра вполне похожа на то же самое в основной базе (за вычетом дешевых вариантов)

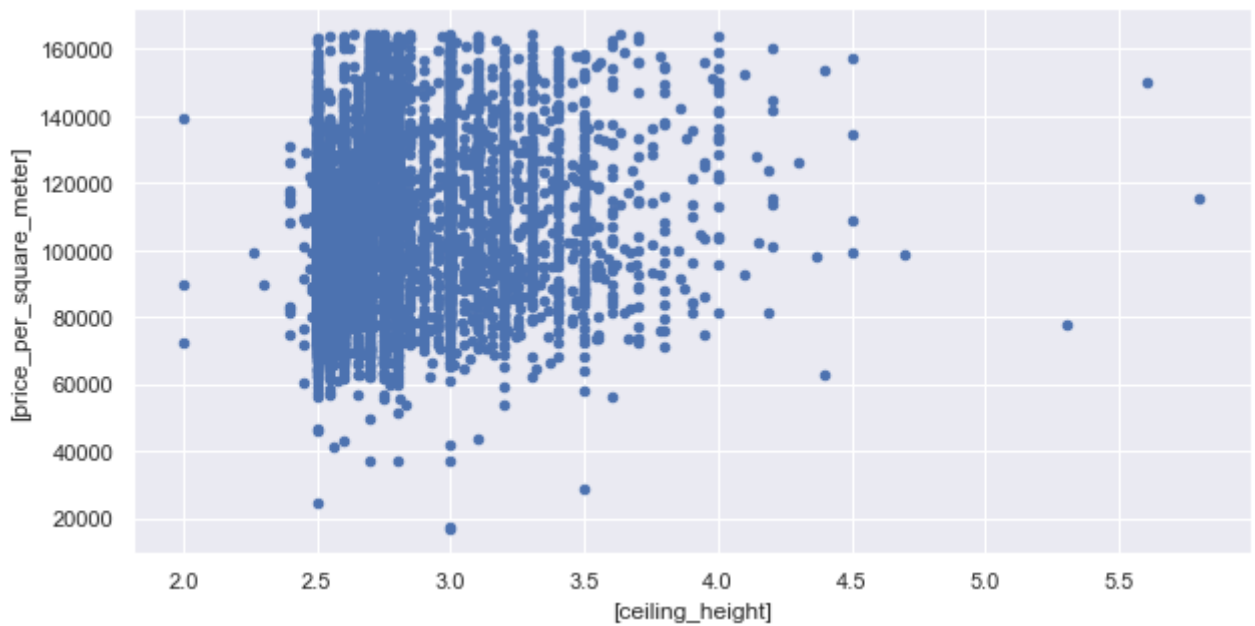
8.23 ВЫСОТА ПОТОЛКОВ

In [188]:

```
1 # по всей базе
2 data_short.plot.scatter(x=['ceiling_height'], y=['price_per_square_meter'], col
```



```
1 # по центру
2 centre.plot.scatter(x=['ceiling_height'], y=['price_per_square_meter'], color='')
```



8.24 Высота потолков влияет на цену квадратного метра

- в основной выборке после высоты 2,8м, снижается количество дешевых вариантов
- и далее наблюдается линейный рост минимального порога
- для центра засчет отсутствия низких цен ,параметр высоты потолка уже не имеет такого влияния

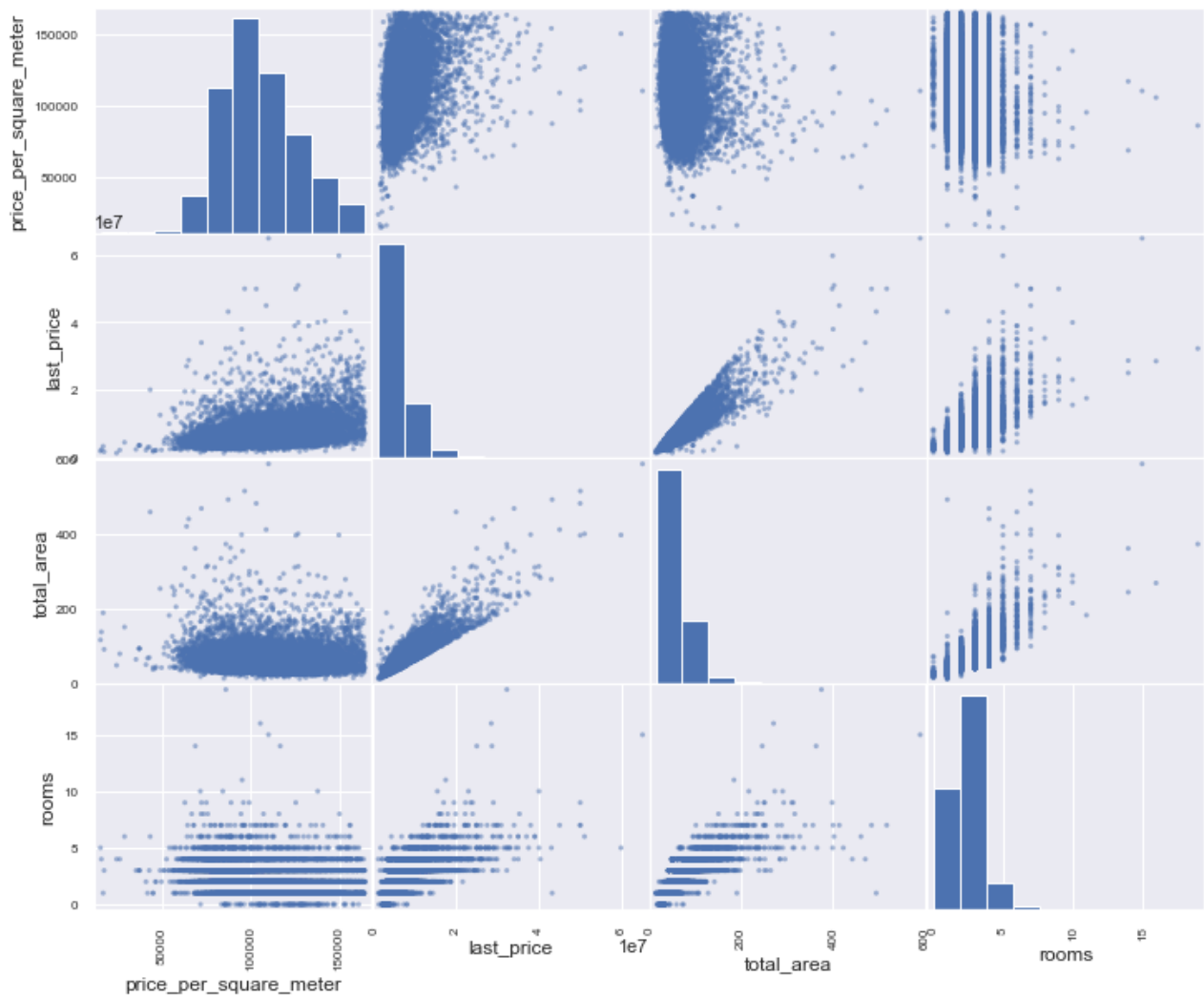
In [190]: 

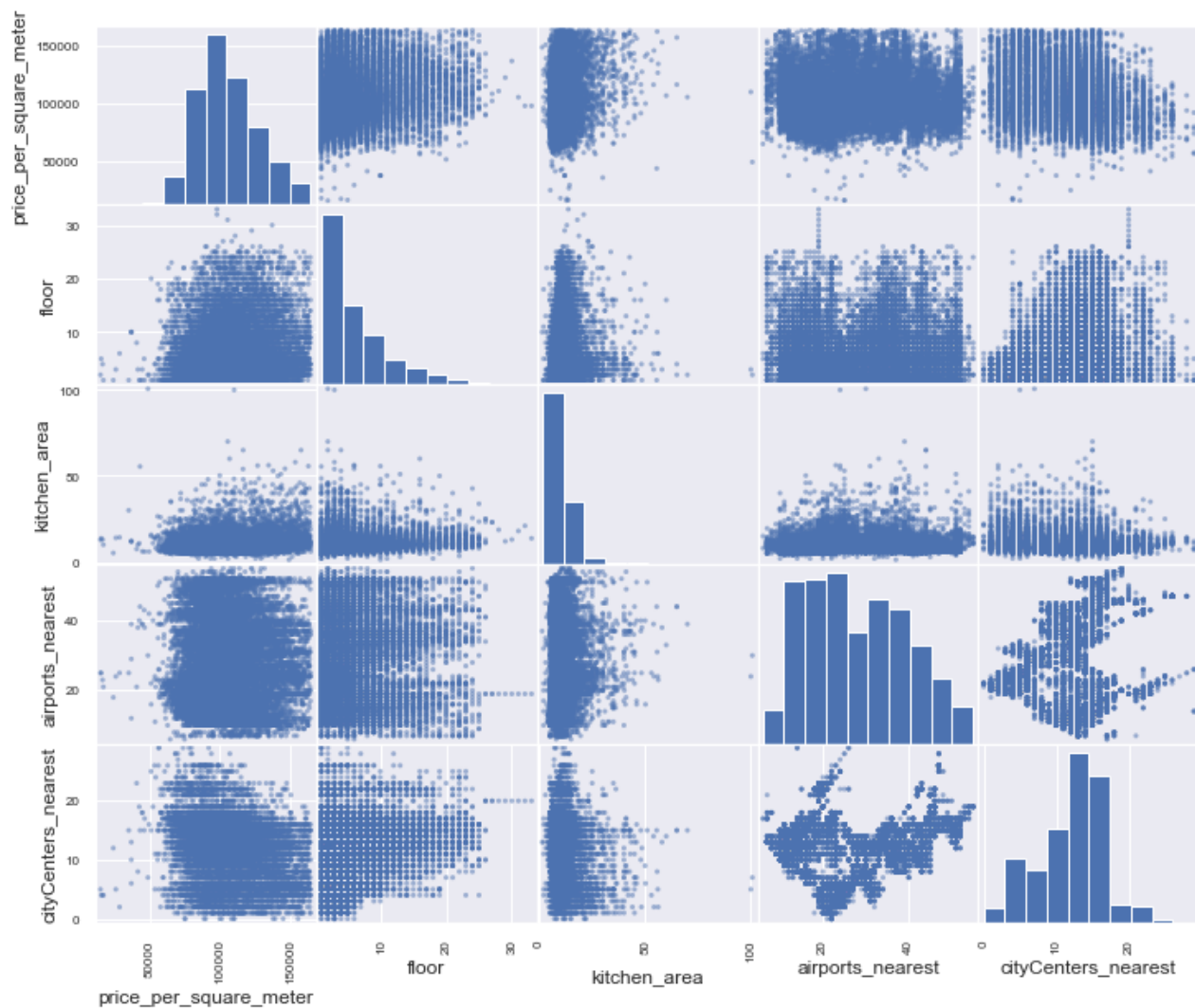
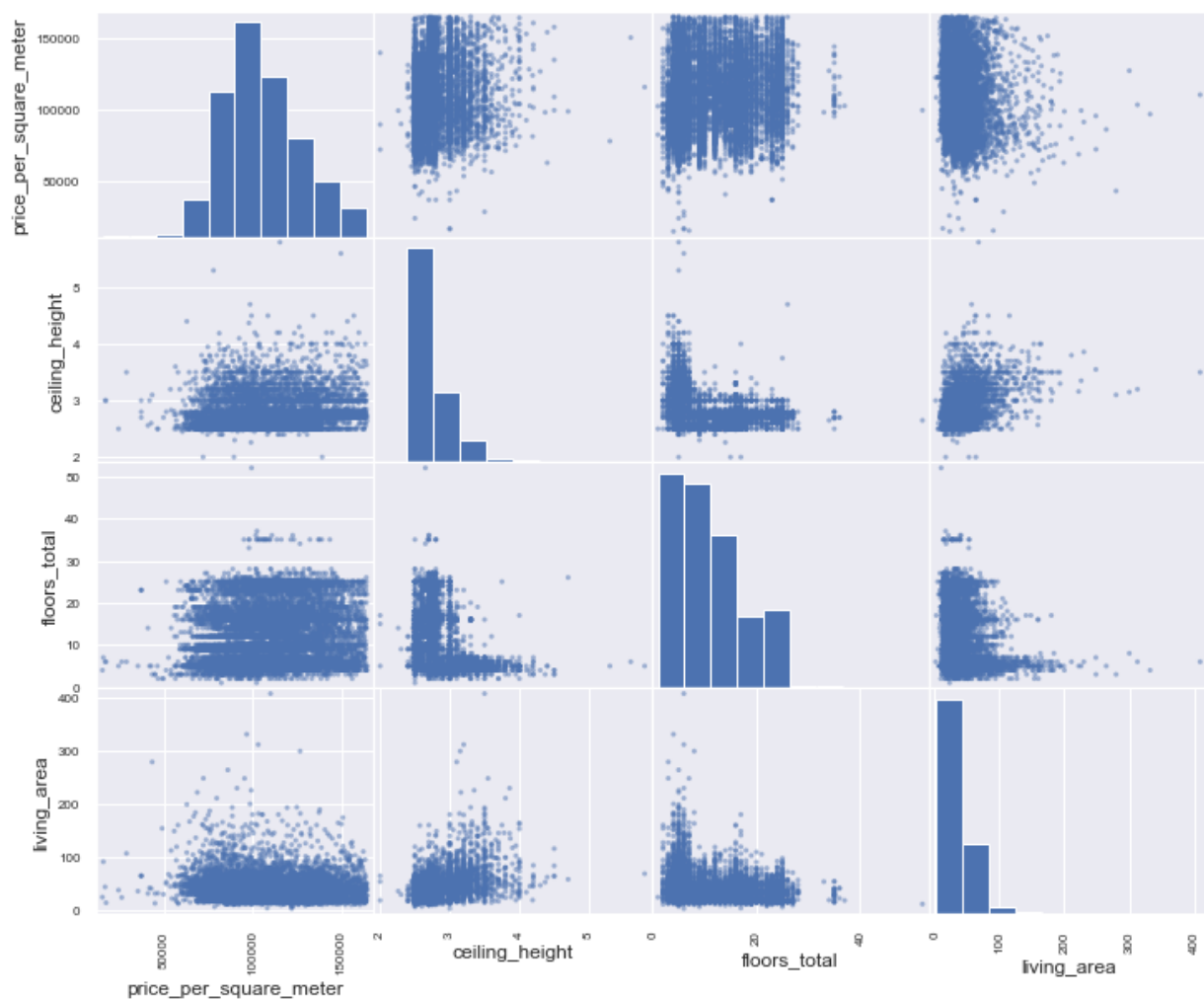
```
1 centre.info()
```

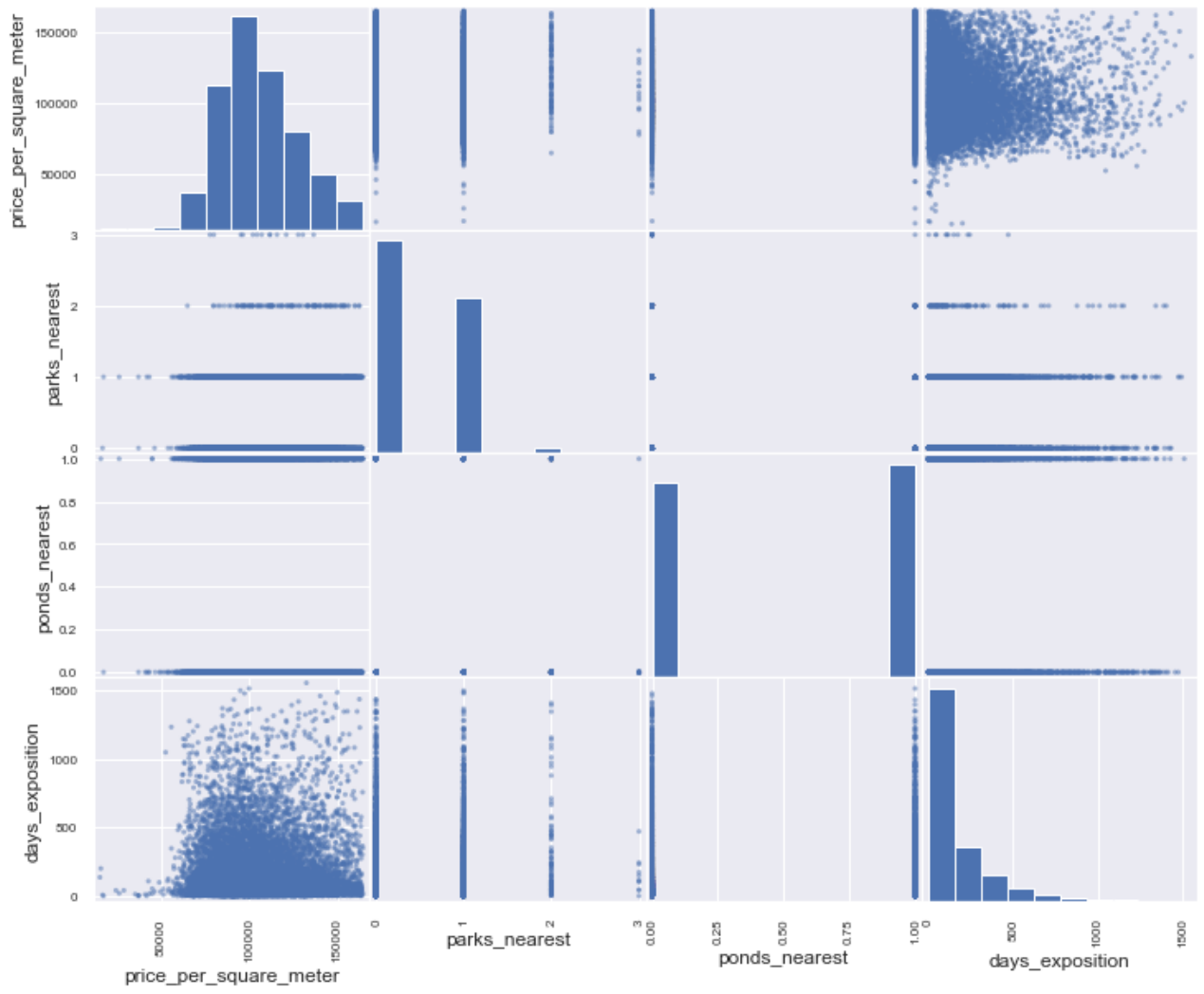
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14617 entries, 0 to 14663
Data columns (total 29 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   total_images                          14617 non-null  int64
1   last_price                            14617 non-null  float64
2   total_area                            14617 non-null  float64
3   first_day_exposition                 14617 non-null  datetime64[ns]
4   rooms                                14617 non-null  int64
5   ceiling_height                       9364 non-null   float64
6   floors_total                         14553 non-null  float64
7   living_area                          13603 non-null  float64
8   floor                                14617 non-null  int64
9   is_apartment                         1707 non-null   object
10  studio                               14617 non-null  bool
11  open_plan                            14617 non-null  bool
12  kitchen_area                         13367 non-null  float64
13  balcony                              14617 non-null  float64
14  locality_name                        14617 non-null  object
```

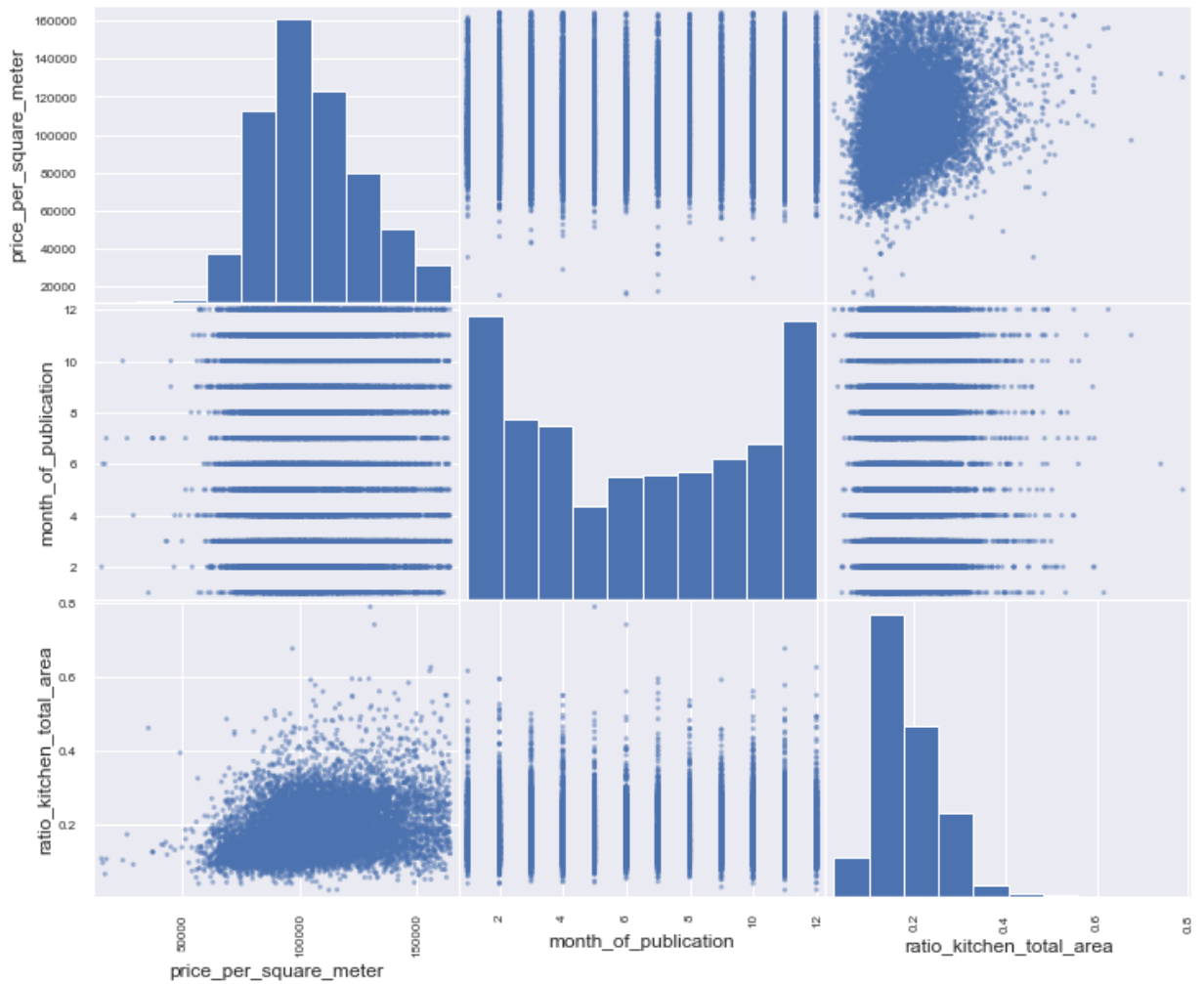
In [191]:

```
1 sns.set(rc = {'figure.figsize':(12,10)})
2 pd.plotting.scatter_matrix(centre[['price_per_square_meter', 'last_price', 'tot
3 pd.plotting.scatter_matrix(centre[['price_per_square_meter', 'ceiling_height',
4 pd.plotting.scatter_matrix(centre[['price_per_square_meter', 'floor', 'kitchen_
5 pd.plotting.scatter_matrix(centre[['price_per_square_meter', 'parks_nearest', '
6 pd.plotting.scatter_matrix(centre[['price_per_square_meter', 'apartment_floor',
```









In [192]:

```
1 centre_corr = data_short[['price_per_square_meter', 'ceiling_height', 'floors_total', 'last_price', 'kitchen_area', 'cityCenters_nearest', 'airports_nearest']]
2 centre_corr
```

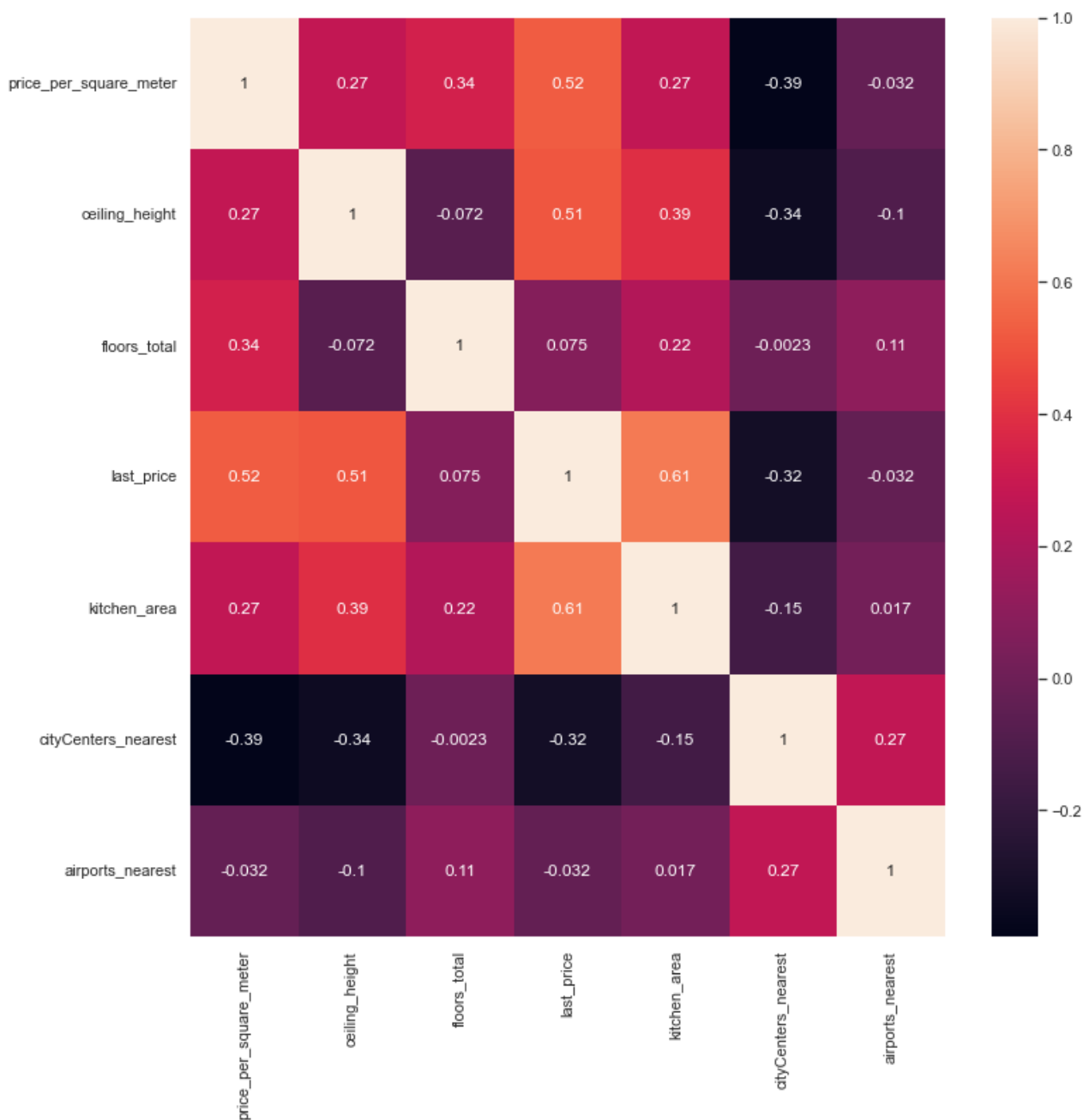
Out[192]:

	price_per_square_meter	ceiling_height	floors_total	last_price	kitchen_area	cityCenters_nearest	airports_nearest
price_per_square_meter	1.00	0.27	0.34	0.52	0.27	-0.39	-0.03
ceiling_height	0.27	1.00	-0.07	0.51	0.39	-0.34	-0.10
floors_total	0.34	-0.07	1.00	0.07	0.22	-0.00	0.11
last_price	0.52	0.51	0.07	1.00	0.61	-0.32	-0.03
kitchen_area	0.27	0.39	0.22	0.61	1.00	-0.15	0.02
cityCenters_nearest	-0.39	-0.34	-0.00	-0.32	-0.15	1.00	-0.03
airports_nearest	-0.03	-0.10	0.11	-0.03	0.02	-0.03	1.00

In [193]:



```
1 sns.set(rc = {'figure.figsize':(12,12)})
2 sns.heatmap(centre_corr, annot=True);
```



Математически существенного влияния на цену квадратного метра объектов недвижимости внутри центра города не выявлено ни для одного параметра.

Единственно, **среднее** влияние имеет общая цена на квартиру. Более дорогие квартиры имеют тенденцию продаваться с более высокой ценой за кв. м

Чуть выше слабого имеет влияние расстояние до нулевой отметки.

- Чем ближе, тем дороже.

9 Общий вывод

10 На цену квадратного метра влияют:

- общая площадь квартиры: после 100 метров, стабильно исчезают дешевые варианты
- число комнат: после 4х комнат, квартиры постепенно переходят в дорогой сегмент
- удалённость от центра
- до 8500 м практически нет дешевых вариантов
- после 17000 начинается линейный спад
- Также явно видна граница города ~23 км.
- Затем следует всплеск ближнего загорода до ~27 км.
- после 35000 общественное снижение цены
- в ареоле 50000 анклав недорогой массовой застройки
- Этаж
- первые этажи дешевле всего

10.1 Продавать квартиру можно и быстро и долго

10.1.1 быстро

за месяц

10.1.2 хорошо

за квартал

10.1.3 обычно

дольше, чем полгода, но быстрее, чем ребёнок родится

10.1.4 долго

больше года

10.1.5 задуматься

это зависит от потребностей продавца.

- какой ориентир для него важен
- быстро
- хорошо
- обычно