

**Привет!** Меня зовут Сороколетов Илья и я буду проверять твой проект. Предлагаю общаться на «\*\*ты\*\*», но если тебе будет комфортнее общаться на «\*\*вы\*\*», то сообщи об этом в комментариях) Для твоего удобства, я буду выделять свои комментарии следующим образом:

**✗ Комментарий ревьюера v1:** Самые важные замечания. Они указывают на ключевые моменты, которые влияют на конечный результат проекта.

**⚠ Комментарий ревьюера v1:** Советы или замечания, которые помогут сделать твою работу лучше, но необязательны к выполнению.

**✓ Комментарий ревьюера v1:** Так я выделяю все остальные комментарии.

Пометками **v1-v2-v3-...** я буду отмечать версию проверки. Так ты сможешь быстро найти мои новые комментарии.

Давай работать над проектом в диалоге: если ты что-то меняешь в проекте по моим рекомендациям — пиши об этом. Выбери для своих комментариев какой-то заметный цвет, так мне будет легче отследить изменения, например вот так:

#### Комментарий студента:

**Пожалуйста, не перемещай, не изменяй и не удаляй мои комментарии.** Если ты оставляешь свои комментарии, то делай это под моими, чтобы сообщения были расположены в хронологическом порядке. Всё это поможет выполнить повторную проверку твоего проекта быстрее.

Перед отправкой работы я рекомендую нажимать Kernel -> Restart & Run All. Это перезапустит ядро и по очереди выполнит все ячейки. Так ты сможешь проверить, что всё работает корректно. Кнопка Kernel находится в панели сверху

Обязательно задавай вопросы если они возникнут, а я перехожу к проверке)

P.S. На всякий случай, я оставлю пустой шаблон для твоих комментариев ниже. Кликни два раза на мой комментарий, скопируй последние четыре строчки кола и вставляй их в пустую ячейку там, где ты хочешь оставить комментарий. Не забудь только

Последние ячейки суть как когда-то вставили их в текущую ячейку или где-то еще оставил комментарий посреди текста перед этим сменить тип ячейки на Markdown. Быстро это можно сделать так: кликнуть на ячейку - нажать ESC - нажать M.

### Комментарий студента:

Удали этот текст и вместо него напиши свой комментарий 😊

## Исследование объявлений о продаже квартир

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

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

Откройте файл с данными и изучите общую информацию.

In [1]:

```
import pandas as pd # импорт библиотеки pandas
import matplotlib.pyplot as plt
```

In [2]:

```
data = pd.read_csv('https://code.s3.yandex.net/datasets/real_estate_data.csv', sep='\t')
```

Для ознакомления с данными, хранящимися в датафрейме, выводим его первые 10 строк методом head()

In [3]:

```
pd.set_option('display.max_columns', None)
```

In [4]:

```
data.head(50)
```

| Out[4]: |    | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|---------|----|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
|         | 0  | 20           | 13000000.0 | 108.00     | 2019-03-07T00:00:00  | 3     | 2.70           | 16.0         | 51.00       | 8     | NaN          | False  | False     |
|         | 1  | 7            | 3350000.0  | 40.40      | 2018-12-04T00:00:00  | 1     | NaN            | 11.0         | 18.60       | 1     | NaN          | False  | False     |
|         | 2  | 10           | 5196000.0  | 56.00      | 2015-08-20T00:00:00  | 2     | NaN            | 5.0          | 34.30       | 4     | NaN          | False  | False     |
|         | 3  | 0            | 64900000.0 | 159.00     | 2015-07-24T00:00:00  | 3     | NaN            | 14.0         | NaN         | 9     | NaN          | False  | False     |
|         | 4  | 2            | 10000000.0 | 100.00     | 2018-06-19T00:00:00  | 2     | 3.03           | 14.0         | 32.00       | 13    | NaN          | False  | False     |
|         | 5  | 10           | 2890000.0  | 30.40      | 2018-09-10T00:00:00  | 1     | NaN            | 12.0         | 14.40       | 5     | NaN          | False  | False     |
|         | 6  | 6            | 3700000.0  | 37.30      | 2017-11-02T00:00:00  | 1     | NaN            | 26.0         | 10.60       | 6     | NaN          | False  | False     |
|         | 7  | 5            | 7915000.0  | 71.60      | 2019-04-18T00:00:00  | 2     | NaN            | 24.0         | NaN         | 22    | NaN          | False  | False     |
|         | 8  | 20           | 2900000.0  | 33.16      | 2018-05-23T00:00:00  | 1     | NaN            | 27.0         | 15.43       | 26    | NaN          | False  | False     |
|         | 9  | 18           | 5400000.0  | 61.00      | 2017-02-26T00:00:00  | 3     | 2.50           | 9.0          | 43.60       | 7     | NaN          | False  | False     |
|         | 10 | 5            | 5050000.0  | 39.60      | 2017-11-16T00:00:00  | 1     | 2.67           | 12.0         | 20.30       | 3     | NaN          | False  | False     |
|         | 11 | 9            | 3300000.0  | 44.00      | 2018-08-27T00:00:00  | 2     | NaN            | 5.0          | 31.00       | 4     | False        | False  | False     |
|         | 12 | 10           | 3890000.0  | 54.00      | 2016-06-30T00:00:00  | 2     | NaN            | 5.0          | 30.00       | 5     | NaN          | False  | False     |
|         | 13 | 20           | 3550000.0  | 42.80      | 2017-07-01T00:00:00  | 2     | 2.56           | 5.0          | 27.00       | 5     | NaN          | False  | False     |
|         | 14 | 1            | 4400000.0  | 36.00      | 2016-06-23T00:00:00  | 1     | NaN            | 6.0          | 17.00       | 1     | NaN          | False  | False     |
|         | 15 | 16           | 4650000.0  | 39.00      | 2017-11-18T00:00:00  | 1     | NaN            | 14.0         | 20.50       | 5     | NaN          | False  | False     |

|    | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|----|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 16 | 11           | 6700000.0  | 82.00      | 2017-11-23T00:00:00  | 3     | 3.05           | 5.0          | 55.60       | 1     | NaN          | False  | False     |
| 17 | 6            | 4180000.0  | 36.00      | 2016-09-09T00:00:00  | 1     | NaN            | 17.0         | 16.50       | 7     | NaN          | False  | False     |
| 18 | 8            | 3250000.0  | 31.00      | 2017-01-27T00:00:00  | 1     | 2.50           | 5.0          | 19.40       | 2     | NaN          | False  | False     |
| 19 | 16           | 14200000.0 | 121.00     | 2019-01-09T00:00:00  | 3     | 2.75           | 16.0         | 76.00       | 8     | NaN          | False  | False     |
| 20 | 12           | 6120000.0  | 80.00      | 2017-09-28T00:00:00  | 3     | 2.70           | 27.0         | 48.00       | 11    | NaN          | False  | False     |
| 21 | 13           | 3200000.0  | 31.60      | 2018-03-14T00:00:00  | 1     | NaN            | 5.0          | 16.90       | 2     | NaN          | False  | False     |
| 22 | 20           | 5000000.0  | 58.00      | 2017-04-24T00:00:00  | 2     | 2.75           | 25.0         | 30.00       | 15    | NaN          | False  | False     |
| 23 | 11           | 2950000.0  | 32.00      | 2016-10-29T00:00:00  | 1     | 2.60           | 9.0          | 17.70       | 9     | NaN          | False  | False     |
| 24 | 8            | 6500000.0  | 97.20      | 2015-10-31T00:00:00  | 2     | NaN            | 3.0          | 46.50       | 1     | NaN          | False  | False     |
| 25 | 3            | 6800000.0  | 76.00      | 2015-10-01T00:00:00  | 2     | 2.75           | 23.0         | 39.00       | 18    | False        | False  | False     |
| 26 | 6            | 4050000.0  | 60.00      | 2017-04-28T00:00:00  | 4     | NaN            | 5.0          | 43.00       | 4     | NaN          | False  | False     |
| 27 | 20           | 7100000.0  | 70.00      | 2017-05-12T00:00:00  | 3     | 2.60           | 17.0         | 49.00       | 11    | NaN          | False  | False     |
| 28 | 8            | 4170000.0  | 44.00      | 2017-12-13T00:00:00  | 1     | 2.90           | 6.0          | 20.80       | 1     | NaN          | False  | False     |
| 29 | 9            | 8600000.0  | 100.00     | 2016-04-09T00:00:00  | 3     | NaN            | 19.0         | 52.00       | 15    | False        | False  | False     |
| 30 | 12           | 2200000.0  | 32.80      | 2018-02-19T00:00:00  | 1     | NaN            | 9.0          | NaN         | 2     | NaN          | False  | False     |

|    | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|----|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 31 | 8            | 7200000.0  | 67.90      | 2017-10-26T00:00:00  | 2     | 2.80           | 16.0         | 38.10       | 4     | NaN          | False  | False     |
| 32 | 7            | 4990000.0  | 60.00      | 2016-05-22T00:00:00  | 3     | NaN            | 5.0          | 39.00       | 4     | NaN          | False  | False     |
| 33 | 8            | 4800000.0  | 73.00      | 2018-10-15T00:00:00  | 4     | NaN            | 9.0          | 51.60       | 5     | NaN          | False  | False     |
| 34 | 3            | 3290000.0  | 33.00      | 2018-02-04T00:00:00  | 1     | 2.55           | 16.0         | 14.00       | 3     | NaN          | False  | False     |
| 35 | 6            | 15500000.0 | 149.00     | 2017-06-26T00:00:00  | 5     | NaN            | 5.0          | 104.00      | 4     | NaN          | False  | False     |
| 36 | 13           | 3790000.0  | 45.00      | 2017-01-25T00:00:00  | 2     | NaN            | 9.0          | 27.00       | 9     | False        | False  | False     |
| 37 | 10           | 1990000.0  | 45.80      | 2017-10-28T00:00:00  | 2     | 2.50           | 5.0          | NaN         | 1     | NaN          | False  | False     |
| 38 | 10           | 3150000.0  | 40.00      | 2018-03-29T00:00:00  | 1     | 2.75           | 18.0         | 16.30       | 9     | NaN          | False  | False     |
| 39 | 15           | 5200000.0  | 54.40      | 2018-11-29T00:00:00  | 2     | 2.75           | 9.0          | 29.70       | 2     | NaN          | False  | False     |
| 40 | 9            | 3590000.0  | 36.00      | 2017-03-15T00:00:00  | 1     | 2.60           | 26.0         | 15.00       | 22    | NaN          | False  | False     |
| 41 | 16           | 7900000.0  | 74.00      | 2016-05-04T00:00:00  | 3     | NaN            | 14.0         | 59.00       | 8     | False        | False  | False     |
| 42 | 13           | 22000000.0 | 161.80     | 2015-07-08T00:00:00  | 4     | 2.80           | 4.0          | 80.90       | 2     | False        | False  | False     |
| 43 | 13           | 9330000.0  | 48.00      | 2017-01-10T00:00:00  | 2     | 3.00           | 4.0          | 28.00       | 4     | NaN          | False  | False     |
| 44 | 13           | 5350000.0  | 40.00      | 2018-11-18T00:00:00  | 1     | NaN            | 22.0         | NaN         | 3     | NaN          | False  | False     |
| 45 | 17           | 5200000.0  | 50.60      | 2018-12-02T00:00:00  | 2     | 2.65           | 9.0          | 30.30       | 7     | NaN          | False  | False     |

|    | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|----|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 46 | 17           | 6600000.0  | 52.10      | 2019-01-31T00:00:00  | 2     | 2.60           | 24.0         | 29.70       | 9     | NaN          | False  | False     |
| 47 | 17           | 3600000.0  | 56.10      | 2018-10-18T00:00:00  | 3     | NaN            | 4.0          | 42.50       | 3     | NaN          | False  | False     |
| 48 | 10           | 3600000.0  | 33.83      | 2017-10-03T00:00:00  | 1     | NaN            | 24.0         | 15.35       | 6     | NaN          | False  | False     |
| 49 | 1            | 3050000.0  | 30.80      | 2018-11-22T00:00:00  | 1     | 2.50           | 9.0          | 18.00       | 7     | NaN          | False  | False     |

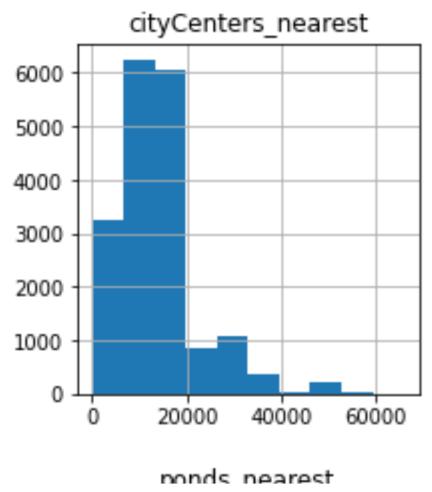
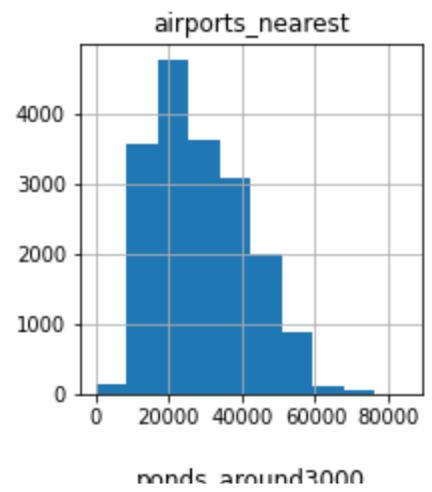
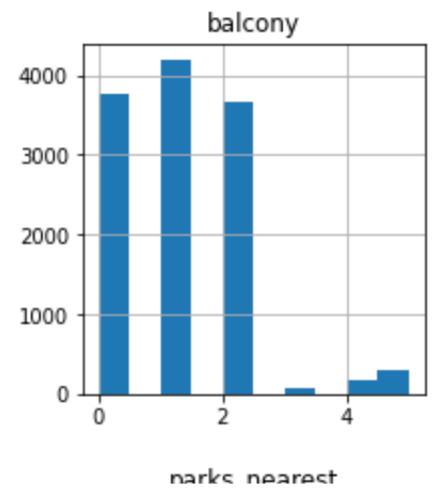
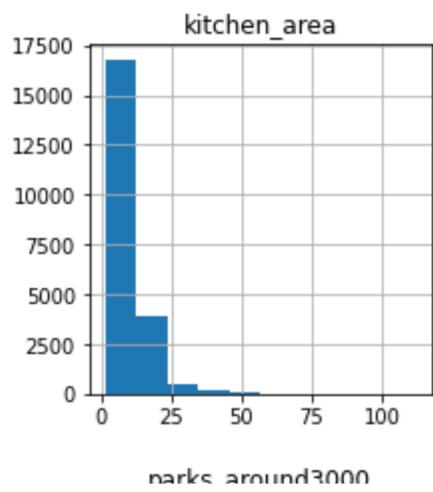
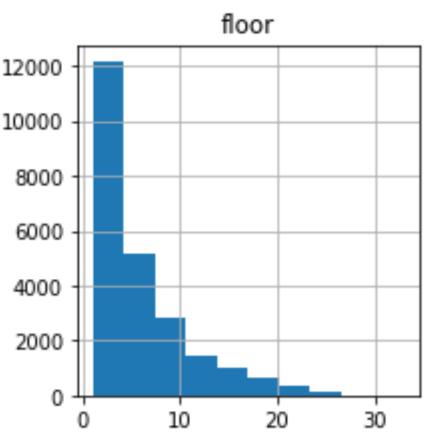
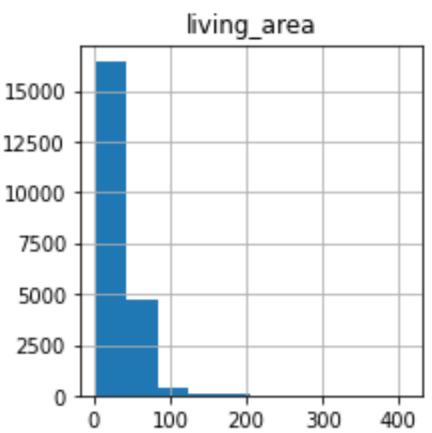
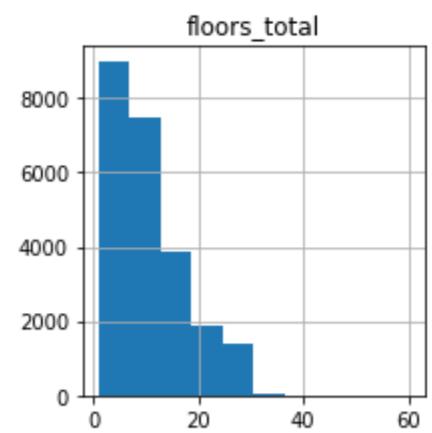
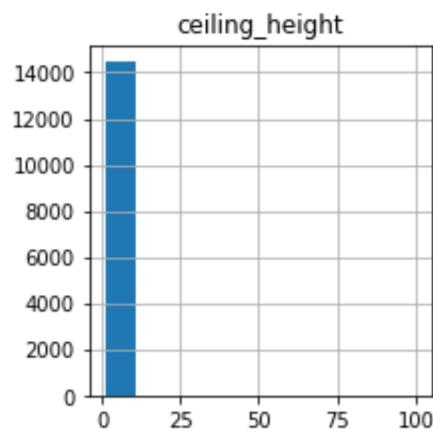
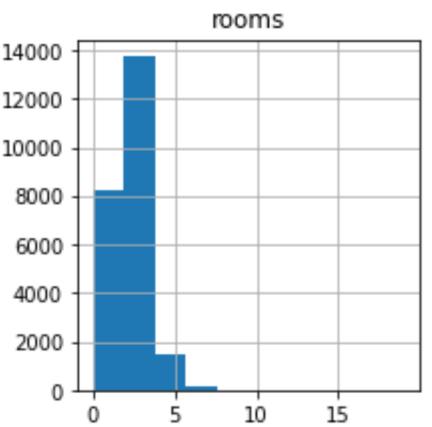
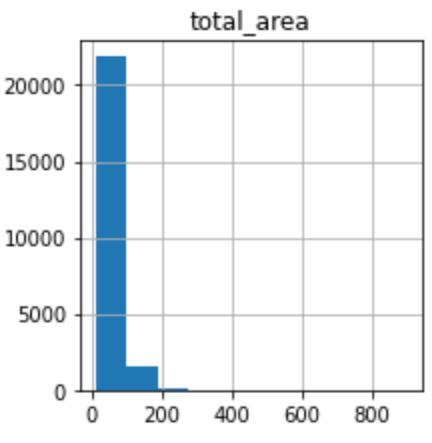
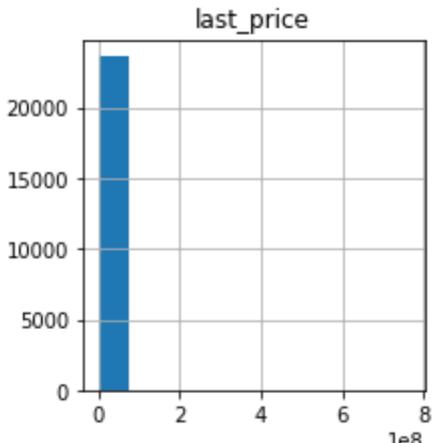
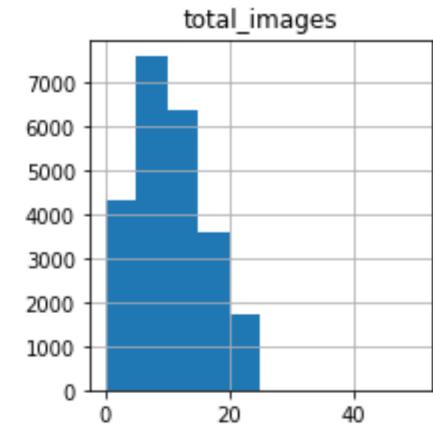
**Методом info() выводим основные данные о датафрейме**

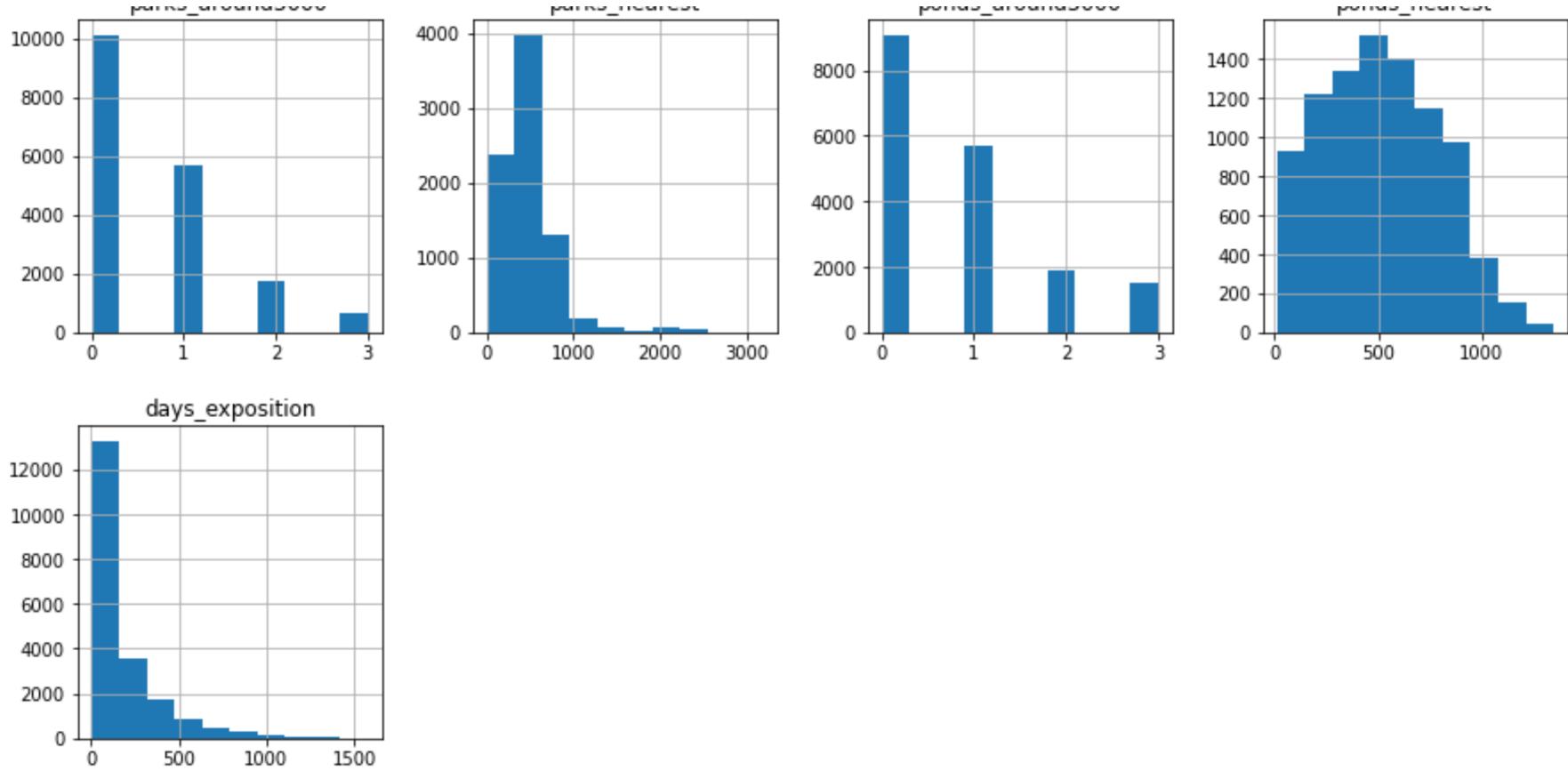
In [5]: `data.info()`

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

Для оценки значений в числовых столбцах датафрейма, строим гистограммы методом `hist()`

```
In [6]: data.hist(figsize=(15, 20));
```





✓ **Комментарий ревьюера v1:**

Графики построено верно!

Из полученных гистограмм, можно отметить выбросы в данных, которые в дальнейшем могут привести к ложным выводам при анализе. Явные выбросы можно видеть в столбцах `last_price`, `total_area`, `ceiling_height`.

### ✓ Комментарий ревьюера v1:

Целью первичного анализа данных является выявление проблем с данными, с которыми нужно будет поработать. Чем больше всего мы найдём в этом шаге, тем лучше сможем построить дальнейший план работы и понять с чем нам стоит работать, а на что можно закрыть глаза. Страйся указывать как можно больше проблем в данных, которые ты найдешь (пропуски, дубликаты, неверный тип данных, странные значения и т.д.)

## Предобработка данных

### Работа с пропусками

- Разберем каждый столбец, где есть пропуски, как заполнять эти пропуски и стоит ли их заполнять вообще

```
In [7]: data.isna().sum() # Общее количество пропусков в каждом столбце
```

```
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
```

```
In [8]: data['ceiling_height'].describe() # для оценки значений столбца `ceiling_height` воспользуемся методом describe()
```

```
Out[8]: count    14504.000000  
mean      2.771499  
std       1.261056  
min       1.000000  
25%      2.520000  
50%      2.650000  
75%      2.800000  
max      100.000000  
Name: ceiling_height, dtype: float64
```

- `ceiling_height` - высота потолков определяется строительными нормами и редко может выбиваться из стандартных значений. Применив метод `describe()` к столбцу `ceiling_height`, мы получаем, что в 75% случаев высота не превышает 2,8м. Но поскольку в разных объектах недвижимости высота отличается, пустые строки заполним медианным значением.

```
In [9]: data['ceiling_height'] = data['ceiling_height'].fillna(data['ceiling_height'].median())
```

✓ **Комментарий ревьюера v1:**

Согласен с таким решением, так как основная часть данных в этом столбце находится в небольшом диапазоне.

- `floors_total` - найти значение для заполнения пропусков в данном столбце довольно проблематично, так как данный столбец слабо коррелирует с другими столбцами датафрейма. Заполнять пропуски средним или медианным значением также некорректно и может привести лишь к искажению общей картины. К тому же количество пропусков составляет лишь доли процента от общего числа значений, поэтому данные пропуски оставим без изменений.
- `living_area` - значения данного столбца напрямую зависят от значений общей площади. В столбце `total_area` пропусков нет, отталкиваясь от этих значений заполним пропуски в столбце `living_area`. Используем отношение медианного значения `living_area` к `total_area` как коэффициент. Произведение `total_area` и этого коэффициента даст нам значение `living_area`. Столбец `living_area` играет существенную роль при анализе рынка недвижимости, поэтому, чтобы убедится в корректности такого метода заполнения пропусков, вызовем метод `describe()` до и после заполнения пропусков.

```
In [10]: data['living_area'].describe()
```

```
Out[10]: count    21796.000000
mean      34.457852
std       22.030445
min       2.000000
25%      18.600000
50%      30.000000
75%      42.300000
max      409.700000
Name: living_area, dtype: float64
```

```
In [11]: data.loc[data['living_area'].isna(), 'living_area'] = \
round(data['total_area']*(data['living_area'].median() / data['total_area'].median()), 2)
```

```
In [12]: data['living_area'].describe()
```

```
Out[12]: count    23699.000000
mean      34.624291
std       22.381876
min       2.000000
25%      19.000000
50%      30.000000
75%      42.170000
max     409.700000
Name: living_area, dtype: float64
```

Сравнивая полученные результаты, видим, что среднее значение и стандартное отклонение изменились на десятые доли, медиана не изменилась. Результат после заполнения пропусков - приемлемый.

- `is_apartment` - исходя из того, что апартаменты являются специфическим товаром на рынке недвижимости и количество пропусков составляет около 90% - предположим, что по умолчанию значение `False` (не апартаменты). Выяснить значение в данном столбце, опираясь на данные датафрейма вряд ли получится и может лишь привести к серьёзным ошибкам при анализе.

```
In [13]: data['is_apartment'] = data['is_apartment'].fillna(False)
```

✓ **Комментарий ревьюера v1:**



- `kitchen_area` - можно заполнить часть пропусков, опираясь на значения столбца `Studio`. Зная, что это студия, площадь кухни можем указать как ноль.

```
In [14]: data.query('studio == True') # смотрим, заполнены ли данные площади кухни в студиях
```

Out[14]:

|  |              | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_p |
|--|--------------|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|--------|
|  | <b>144</b>   | 1            | 2450000.0  | 27.00      | 2017-03-30T00:00:00  | 0     | 2.65           | 24.0         | 15.50       | 2     | False        | True   | Fa     |
|  | <b>440</b>   | 8            | 2480000.0  | 27.11      | 2018-03-12T00:00:00  | 0     | 2.65           | 17.0         | 24.75       | 4     | False        | True   | Fa     |
|  | <b>608</b>   | 2            | 1850000.0  | 25.00      | 2019-02-20T00:00:00  | 0     | 2.65           | 10.0         | 14.42       | 7     | False        | True   | Fa     |
|  | <b>697</b>   | 12           | 2500000.0  | 24.10      | 2017-12-01T00:00:00  | 0     | 2.75           | 25.0         | 17.50       | 21    | False        | True   | Fa     |
|  | <b>716</b>   | 5            | 1500000.0  | 17.00      | 2017-06-07T00:00:00  | 0     | 2.70           | 9.0          | 12.00       | 1     | False        | True   | Fa     |
|  | ...          | ...          | ...        | ...        | ...                  | ...   | ...            | ...          | ...         | ...   | ...          | ...    | ...    |
|  | <b>22867</b> | 8            | 3090000.0  | 30.00      | 2017-12-17T00:00:00  | 0     | 2.65           | 25.0         | 18.20       | 5     | False        | True   | Fa     |
|  | <b>22877</b> | 2            | 4280000.0  | 28.00      | 2017-10-26T00:00:00  | 0     | 2.70           | 19.0         | 18.00       | 10    | False        | True   | Fa     |
|  | <b>23210</b> | 7            | 3200000.0  | 26.00      | 2017-09-01T00:00:00  | 0     | 2.65           | 25.0         | 18.00       | 16    | False        | True   | Fa     |
|  | <b>23554</b> | 15           | 3350000.0  | 26.00      | 2018-09-07T00:00:00  | 0     | 2.65           | 19.0         | 15.00       | 8     | False        | True   | Fa     |
|  | <b>23637</b> | 8            | 2350000.0  | 26.00      | 2018-06-26T00:00:00  | 0     | 2.65           | 25.0         | 17.00       | 4     | False        | True   | Fa     |

149 rows × 22 columns

In [15]: `data.loc[data['studio'] == True, 'kitchen_area'] = 0`In [16]: `data['kitchen_area'].isna().sum() # Проверяем, что количество пропусков в столбце 'kitchen_area' уменьшилось`

Out[16]: 2129

- balcony - если не указаны данные о количестве балконов, то можно предположить, что их нет. Заполним пропуски нулями.

```
In [17]: data['balcony'] = data['balcony'].fillna(0)
```

✓ **Комментарий ревьюера v1:**



- parks\_around3000 , ponds\_around3000 - отсутствие данных в этих двух столбцах может говорить лишь о том, что в радиусе трёх километров нет парка, водоема. Заменим пропуски нулями.

```
In [18]: data['parks_around3000'] = data['parks_around3000'].fillna(0)
```

```
In [19]: data['ponds_around3000'] = data['ponds_around3000'].fillna(0)
```

- **столбцы, в которых указаны расстояния до ближайших парка, водоема, аэропорта, центра города.** В каждом из этих столбцов значительное количество пустых строк. Но заполнять эти пропуски одинаковыми значениями не корректно, поскольку квартиры могут находиться в разных частях города. Также, например, в столбце parks\_nearest пропусков больше половины от всех данных, соответственно нельзя найти значение для заполнения по меньшему количеству данных, и среднее, и медиана не подходят для этого. Поэтому столбцы о расстоянии оставим без изменений

✓ **Комментарий ревьюера:**

Очень важно обоснование. Иногда пропуски можно удалить (если их мало), либо вообще не трогать, если у нас нет возможности подобрать аргументированный способ замены. Обращай внимание на то, на сколько сильно ты искажешь данные при заполнении пропусков.

```
In [31]: data.head(10) # проверяем изменения в датасете
```

Out[31]:

|   | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|---|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 0 | 20           | 13000000.0 | 108.00     | 2019-03-07T00:00:00  | 3     | 2.70           | 16.0         | 51.00       | 8     | False        | False  | False     |
| 1 | 7            | 3350000.0  | 40.40      | 2018-12-04T00:00:00  | 1     | 2.65           | 11.0         | 18.60       | 1     | False        | False  | False     |
| 2 | 10           | 5196000.0  | 56.00      | 2015-08-20T00:00:00  | 2     | 2.65           | 5.0          | 34.30       | 4     | False        | False  | False     |
| 3 | 0            | 64900000.0 | 159.00     | 2015-07-24T00:00:00  | 3     | 2.65           | 14.0         | 91.73       | 9     | False        | False  | False     |
| 4 | 2            | 10000000.0 | 100.00     | 2018-06-19T00:00:00  | 2     | 3.03           | 14.0         | 32.00       | 13    | False        | False  | False     |
| 5 | 10           | 2890000.0  | 30.40      | 2018-09-10T00:00:00  | 1     | 2.65           | 12.0         | 14.40       | 5     | False        | False  | False     |
| 6 | 6            | 3700000.0  | 37.30      | 2017-11-02T00:00:00  | 1     | 2.65           | 26.0         | 10.60       | 6     | False        | False  | False     |
| 7 | 5            | 7915000.0  | 71.60      | 2019-04-18T00:00:00  | 2     | 2.65           | 24.0         | 41.31       | 22    | False        | False  | False     |
| 8 | 20           | 2900000.0  | 33.16      | 2018-05-23T00:00:00  | 1     | 2.65           | 27.0         | 15.43       | 26    | False        | False  | False     |
| 9 | 18           | 5400000.0  | 61.00      | 2017-02-26T00:00:00  | 3     | 2.50           | 9.0          | 43.60       | 7     | False        | False  | False     |

- `locality_name` - название населенного пункта - по сути, это один из ключевых факторов при выборе объекта недвижимости и пропуски здесь недопустимы. Восстановить эти пропуски вряд ли возможно, опираясь на датафрейм. Учитывая, что этих пропусков 0,2% от общего количества значений - удалим эти пропуски.

In [32]: `data = data.dropna(subset=['locality_name'])`

Пропуски в столбце `days_exposition` могут говорить лишь о том, что данное объявление еще актуально. Оставим данный столбец без изменений.

```
In [33]: data.isna().sum() # Проверяем, в каких столбцах еще остались пропуски
```

```
Out[33]: total_images          0  
last_price            0  
total_area            0  
first_day_exposition  0  
rooms                 0  
ceiling_height        0  
floors_total          85  
living_area           0  
floor                 0  
is_apartment          0  
studio                0  
open_plan              0  
kitchen_area          2120  
balcony               0  
locality_name         0  
airports_nearest      5534  
cityCenters_nearest   5511  
parks_around3000       0  
parks_nearest          15586  
ponds_around3000       0  
ponds_nearest          14565  
days_exposition        3180  
dtype: int64
```

**Небольшое резюме по поиску и устраниению пропусков в таблице. На данном этапе я руководствовался принципом "не навреди", т.е. там где зависимость прослеживается плохо - лучше оставить пропуски. При большом желании можно найти зависимость между данными, даже там где ее нет. Вот только в дальнейшем это может привести к ложным выводам. К тому же, неизвестно, а понадобятся ли нам эти данные в дальнейшем, т.е. можно потратить огромное количество времени на устранение пропусков, а по итогу эти пропуски ни на что не повлияли бы или они отвалятся сами в процессе построения различных выборок или сводных таблиц.**

Изменение типов данных

In [34]: `data.info() # Выведем еще раз общую информацию о датафрейме`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23650 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   total_images     23650 non-null   int64  
 1   last_price       23650 non-null   float64 
 2   total_area       23650 non-null   float64 
 3   first_day_exposition  23650 non-null   object  
 4   rooms            23650 non-null   int64  
 5   ceiling_height   23650 non-null   float64 
 6   floors_total     23565 non-null   float64 
 7   living_area      23650 non-null   float64 
 8   floor             23650 non-null   int64  
 9   is_apartment     23650 non-null   bool   
 10  studio            23650 non-null   bool   
 11  open_plan         23650 non-null   bool   
 12  kitchen_area      21530 non-null   float64 
 13  balcony           23650 non-null   float64 
 14  locality_name    23650 non-null   object  
 15  airports_nearest  18116 non-null   float64 
 16  cityCenters_nearest 18139 non-null   float64 
 17  parks_around3000  23650 non-null   float64 
 18  parks_nearest     8064 non-null   float64 
 19  ponds_around3000  23650 non-null   float64 
 20  ponds_nearest     9085 non-null   float64 
 21  days_exposition   20470 non-null   float64 
dtypes: bool(3), float64(14), int64(3), object(2)
memory usage: 3.7+ MB
```

Из данных видим, что столбцы с числовыми данными имеют тип `int` или `float`, что правильно. Также видим, что столбец `first_day_exposition` - `object`. День размещения объявления, все-таки, должен относится к типу `datetime`. Это необходимо для дальнейшего анализа, если нам понадобится извлечь день, месяц, год - столбец должен быть `datetime`. Преобразуем данный столбец.

In [35]: `data['first_day_exposition'] = pd.to_datetime(data['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S')`

In [36]: `data.head() # Проверяем изменения`

Out[36]:

|   | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|---|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 0 | 20           | 13000000.0 | 108.0      | 2019-03-07           | 3     | 2.70           | 16.0         | 51.00       | 8     | False        | False  | False     |
| 1 | 7            | 3350000.0  | 40.4       | 2018-12-04           | 1     | 2.65           | 11.0         | 18.60       | 1     | False        | False  | False     |
| 2 | 10           | 5196000.0  | 56.0       | 2015-08-20           | 2     | 2.65           | 5.0          | 34.30       | 4     | False        | False  | False     |
| 3 | 0            | 64900000.0 | 159.0      | 2015-07-24           | 3     | 2.65           | 14.0         | 91.73       | 9     | False        | False  | False     |
| 4 | 2            | 10000000.0 | 100.0      | 2018-06-19           | 2     | 3.03           | 14.0         | 32.00       | 13    | False        | False  | False     |

In [37]: `data.info() # Проверяем изменения`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23650 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   total_images     23650 non-null   int64  
 1   last_price       23650 non-null   float64 
 2   total_area       23650 non-null   float64 
 3   first_day_exposition  23650 non-null   datetime64[ns] 
 4   rooms            23650 non-null   int64  
 5   ceiling_height   23650 non-null   float64 
 6   floors_total     23565 non-null   float64 
 7   living_area      23650 non-null   float64 
 8   floor             23650 non-null   int64  
 9   is_apartment     23650 non-null   bool   
 10  studio            23650 non-null   bool   
 11  open_plan         23650 non-null   bool   
 12  kitchen_area     21530 non-null   float64 
 13  balcony           23650 non-null   float64 
 14  locality_name    23650 non-null   object  
 15  airports_nearest  18116 non-null   float64 
 16  cityCenters_nearest 18139 non-null   float64 
 17  parks_around3000  23650 non-null   float64 
 18  parks_nearest    8064 non-null    float64 
 19  ponds_around3000  23650 non-null   float64 
 20  ponds_nearest    9085 non-null    float64 
 21  days_exposition  20470 non-null   float64 

dtypes: bool(3), datetime64[ns](1), float64(14), int64(3), object(1)
memory usage: 3.7+ MB
```

### ✖ Комментарий ревьюера v1:

Ещё здесь нужно изменить тип данных в столбце balcony

```
In [38]: data = data.astype({'balcony':'int64'})
```

### ✓ Комментарий ревьюера v2:

В данном случае это ещё и вопрос внешнего вида. 1 балкон смотрится лучше чем 1.0 балкон. Это уменьшает количество отображаемых знаков и делает таблицу проще для восприятия

In [39]: `data.info() # Проверяем изменения`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23650 entries, 0 to 23698
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   total_images      23650 non-null   int64  
 1   last_price        23650 non-null   float64 
 2   total_area        23650 non-null   float64 
 3   first_day_exposition  23650 non-null   datetime64[ns]
 4   rooms             23650 non-null   int64  
 5   ceiling_height    23650 non-null   float64 
 6   floors_total      23565 non-null   float64 
 7   living_area       23650 non-null   float64 
 8   floor              23650 non-null   int64  
 9   is_apartment      23650 non-null   bool   
 10  studio            23650 non-null   bool   
 11  open_plan          23650 non-null   bool   
 12  kitchen_area      21530 non-null   float64 
 13  balcony           23650 non-null   int64  
 14  locality_name     23650 non-null   object  
 15  airports_nearest  18116 non-null   float64 
 16  cityCenters_nearest  18139 non-null   float64 
 17  parks_around3000  23650 non-null   float64 
 18  parks_nearest     8064 non-null    float64 
 19  ponds_around3000  23650 non-null   float64 
 20  ponds_nearest     9085 non-null    float64 
 21  days_exposition   20470 non-null   float64 
dtypes: bool(3), datetime64[ns](1), float64(13), int64(4), object(1)
memory usage: 3.7+ MB
```

Обработка дубликатов

### Устранение неявных дубликатов в столбце locality\_name

Для начала выведем уникальные значения и их количество в столбце `locality_name`, используя методы `unique()` и `count_values()`.

```
In [40]: data['locality_name'].unique()
```

```
Out[40]: array(['Санкт-Петербург', 'посёлок Шушары', 'городской посёлок Янино-1',  
   'посёлок Парголово', 'посёлок Мурино', 'Ломоносов', 'Сертолово',  
   'Петргоф', '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 Кузьмоловский',  
   'садовое товарищество Новая Ропша', 'Павловск',  
   'деревня Пикколо', '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 Рошино', 'Кириши', 'Луга', 'Волосово',  
   'Отрадное', 'село Павлово', 'поселок Оредеж', 'село Копорье',  
   'посёлок городского типа Красный Бор', 'посёлок Молодёжное',  
   'Тихвин', 'посёлок Победа', 'деревня Нурма',  
   'поселок городского типа Синявино', 'Тосно',  
   'посёлок городского типа Кузьмоловский', 'посёлок Стрельна',  
   'Бокситогорск', 'посёлок Александровская', 'деревня Лопухинка',  
   'Пикалёво', 'поселок Терволово',  
   'поселок городского типа Советский', 'Подпорожье',  
   'посёлок Петровское', 'посёлок городского типа Токсово',  
   'поселок Сельцо', 'посёлок городского типа Вырица',  
   'деревня Кипень', 'деревня Келози', 'деревня Вартемяги',  
   'посёлок Тельмана', 'поселок Севастьяново',  
   'городской поселок Большая Ижора', 'городской посёлок Павлово',  
   'деревня Агалатово', 'посёлок Новогорелово',  
   'городской посёлок Лесогорский', 'деревня Лаголово',  
   'поселок Цвелодубово', 'поселок городского типа Рахья',  
   'поселок городского типа Вырица', 'деревня Белогорка',  
   'поселок Заводской', 'городской посёлок Новоселье',  
   'деревня Большие Колпаны', 'деревня Горбунки', 'деревня Батово',  
   'деревня Заневка', 'деревня Иссад', 'Приморск',  
   'городской посёлок Фёдоровское', 'деревня Мистолово',
```

'Новая Ладога', 'поселок Зимитицы', 'поселок Барышево',  
'деревня Разметелево', 'поселок городского типа имени Свердлова',  
'деревня Пеники', 'поселок Рябово', 'деревня Пудомяги',  
'поселок станции Корнево', 'деревня Низино', 'деревня Бегуницы',  
'посёлок Поляны', 'городской посёлок Мга', 'поселок Елизаветино',  
'посёлок городского типа Кузнечное', 'деревня Колтуши',  
'поселок Запорожское', 'посёлок городского типа Рошино',  
'деревня Гостилицы', 'деревня Малое Карлино',  
'посёлок Мичуринское', 'посёлок городского типа имени Морозова',  
'посёлок Песочный', 'посёлок Сосново', 'деревня Аро',  
'поселок Ильичёво', 'посёлок городского типа Тайцы',  
'деревня Малое Верево', 'деревня Извара', 'поселок станции Вещево',  
'село Паша', 'деревня Калитино',  
'посёлок городского типа Ульяновка', 'деревня Чудской Бор',  
'поселок городского типа Дубровка', 'деревня Мины',  
'поселок Войсковицы', 'посёлок городского типа имени Свердлова',  
'деревня Коркино', 'посёлок Ропша',  
'поселок городского типа Приладожский', 'посёлок Щеглово',  
'посёлок Гаврилово', 'Лодейное Поле', 'деревня Рабитицы',  
'поселок городского типа Никольский', 'деревня Куэймолово',  
'деревня Малые Колпаны', 'поселок Тельмана',  
'посёлок Петро-Славянка', 'городской посёлок Назия',  
'посёлок Репино', 'посёлок Ильичёво', 'поселок Углово',  
'поселок Старая Малукса', 'садовое товарищество Рахья',  
'поселок Аннино', 'поселок Победа', 'деревня Меньково',  
'деревня Старые Бегуницы', 'посёлок Сапёрный', 'поселок Семрино',  
'поселок Гаврилово', 'поселок Глажево', 'поселок Кобринское',  
'деревня Гарболово', 'деревня Юкки',  
'поселок станции Приветнинское', 'деревня Мануйлово',  
'деревня Пчева', 'поселок Поляны', 'поселок Цвылёво',  
'поселок Мельниково', 'посёлок Пудость', 'посёлок Усть-Луга',  
'Светогорск', 'Любань', 'поселок Селезнёво',  
'поселок городского типа Рябово', 'Каменногорск', 'деревня Кривко',  
'поселок Глебычево', 'деревня Парицы', 'поселок Жилпосёлок',  
'посёлок городского типа Мга', 'городской поселок Янино-1',  
'посёлок Войскорово', 'село Никольское', 'посёлок Терволово',  
'поселок Стеклянный', 'посёлок городского типа Важины',  
'посёлок Мыза-Ивановка', 'село Русско-Высоцкое',  
'поселок городского типа Лебяжье',  
'поселок городского типа Форносово', 'село Старая Ладога',  
'поселок Житково', 'городской посёлок Виллози', 'деревня Лампово',  
'деревня Шпаньково', 'деревня Лаврики', 'посёлок Сумино',

'посёлок Возрождение', 'деревня Старосиверская',  
'посёлок Кикерино', 'поселок Возрождение',  
'деревня Старое Хинково', 'посёлок Пригородный',  
'посёлок Торфяное', 'городской посёлок Будогощь',  
'поселок Суходолье', 'поселок Красная Долина', 'деревня Хапо-Ое',  
'поселок городского типа Дружная Горка', 'поселок Лисий Нос',  
'деревня Яльгелево', 'посёлок Стеклянный', 'село Рождествено',  
'деревня Старополье', 'посёлок Левашово', 'деревня Сяськелево',  
'деревня Камышовка',  
'садоводческое некоммерческое товарищество Лесная Поляна',  
'деревня Хязельки', 'поселок Жилгородок',  
'посёлок городского типа Павлово', 'деревня Ялгино',  
'поселок Новый Учхоз', 'городской посёлок Рошино',  
'поселок Гончарово', 'поселок Почап', 'посёлок Сапёрное',  
'посёлок Платформа 69-й километр', 'поселок Каложицы',  
'деревня Фалилеево', 'деревня Пельгора',  
'поселок городского типа Лесогорский', 'деревня Торошковичи',  
'посёлок Белоостров', 'посёлок Алексеевка', 'поселок Серебрянский',  
'поселок Лукаши', 'поселок Петровское', 'деревня Щеглово',  
'поселок Мичуринское', 'деревня Тарасово', 'поселок Кингисеппский',  
'посёлок при железнодорожной станции Вещево', 'поселок Ушаки',  
'деревня Котлы', 'деревня Сижно', 'деревня Торосово',  
'посёлок Форт Красная Горка', 'поселок городского типа Токсово',  
'деревня Новолисино', 'посёлок станции Громово', 'деревня Глинка',  
'посёлок Мельниково', 'поселок городского типа Назия',  
'деревня Старая Пустошь', 'поселок Коммунары', 'поселок Починок',  
'посёлок городского типа Вознесенье', 'деревня Разбегаево',  
'посёлок городского типа Рябово', 'поселок Гладкое',  
'посёлок при железнодорожной станции Приветнинское',  
'поселок Тёсово-4', 'посёлок Жилгородок', 'деревня Бор',  
'посёлок Коробицыно', 'деревня Большая Вруда', 'деревня Курковицы',  
'посёлок Лисий Нос', 'городской посёлок Советский',  
'посёлок Кобралово', 'деревня Суоранда', 'поселок Кобралово',  
'поселок городского типа Кондратьево',  
'коттеджный поселок Счастье', 'поселок Любань', 'деревня Реброво',  
'деревня Зимитицы', 'деревня Тойворово', 'поселок Семиозерье',  
'поселок Лесное', 'поселок Совхозный', 'поселок Усть-Луга',  
'посёлок Ленинское', 'посёлок Суйда',  
'посёлок городского типа Форносово', 'деревня Нижние Осельки',  
'посёлок станции Свирь', 'поселок Перово', 'Высоцк',  
'поселок Гарболово', 'село Шум', 'поселок Котельский',  
'поселок станции Лужайка', 'деревня Большая Пустомержа',

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

```
In [41]: data['locality_name'].value_counts()
```

```
Out[41]:
```

|                            |       |
|----------------------------|-------|
| Санкт-Петербург            | 15721 |
| посёлок Мурино             | 522   |
| посёлок Шушары             | 440   |
| Всеволожск                 | 398   |
| Пушкин                     | 369   |
| ...                        |       |
| садовое товарищество Рахья | 1     |
| поселок Жилпосёлок         | 1     |
| деревня Сижно              | 1     |
| деревня Лаврики            | 1     |
| посёлок Петро-Славянка     | 1     |

```
Name: locality_name, Length: 364, dtype: int64
```

Изучив полученные данные, выполним следующее:

- приведем все данные в нижний регистр, чтобы выглядели одинаково - на случай если название населенного пункта написали с маленькой буквы или же его тип с большой;
- из списка значений видно, что использование букв ё/е искажает картину. Приведем к одному типу - заменим "ё" на "е";
- название одного и того же населенного пункта указано и как "поселок городского типа" и как "городской поселок". (Рябово, Большая Ижора, Павлово - как примеры). Укажем их просто как "поселок"
- Мурино также указан как поселок Мурино, Кудрово также указан как деревня Кудрово. Википедия говорит, что данные населенные пункты относятся к городским поселениям. Доля объявлений из Кудрово и Мурино довольно существенная и такие данные могут привести к искажениям. Изменим данные в столбце на Кудрово и Мурино.

Для измененных значений создадим новый столбец locality\_name\_new

```
In [42]: data['locality_name_new'] = data['locality_name'].str.lower()

In [43]: data['locality_name_new'] = data['locality_name_new'].str.replace("ё", "е")

In [44]: data['locality_name_new'] = \
data['locality_name_new'].replace([r"\городской поселок", r"\поселок городского типа"], ["поселок", "поселок"], regex=True)

In [45]: data = data.replace({'locality_name_new':{"поселок мурино": "мурино", "деревня кудрово": "кудрово"}})

In [46]: data['locality_name_new'].value_counts() # проверяем изменения в столбце
```

```
Out[46]:
```

| locality_name_new     | Count |
|-----------------------|-------|
| санкт-петербург       | 15721 |
| мурино                | 590   |
| Кудрово               | 472   |
| поселок шушары        | 440   |
| всеволожск            | 398   |
| ...                   |       |
| поселок ромашки       | 1     |
| поселок дзержинского  | 1     |
| деревня большой сабск | 1     |
| деревня ялгино        | 1     |
| деревня хязельки      | 1     |

```
Name: locality_name_new, Length: 320, dtype: int64
```

Проверив изменения в датафрейме можно заметить, что изначально было 364 уникальных значений в столбце. После изменений их стало 320

```
In [47]: data['locality_name_new'].sort_values().unique()
```

```
Out[47]: array(['бокситогорск', 'волосово', 'волхов', 'всеволожск', 'выборг',  
   'высоцк', 'гатчина', 'деревня агалатово', 'деревня аро',  
   'деревня батово', 'деревня бегуницы', 'деревня белогорка',  
   'деревня большая вруда', 'деревня большая пустомержа',  
   'деревня большие колпаны', 'деревня большое рейзино',  
   'деревня большой сабск', 'деревня бор', 'деревня борисова грива',  
   'деревня ваганово', 'деревня вартемяги', 'деревня вахнова кара',  
   'деревня выскатка', 'деревня гарболово', 'деревня глинка',  
   'деревня горбунки', 'деревня гостилицы', 'деревня заклинье',  
   'деревня заневка', 'деревня зимитицы', 'деревня извара',  
   'деревня иссад', 'деревня калитино', 'деревня кальтино',  
   'деревня камышовка', 'деревня каськово', 'деревня келози',  
   'деревня кипень', 'деревня кисельня', 'деревня колтуши',  
   'деревня коркино', 'деревня котлы', 'деревня кривко',  
   'деревня кузьмолово', 'деревня курковицы', 'деревня куровицы',  
   'деревня куттузи', 'деревня лаврики', 'деревня лаголово',  
   'деревня лампово', 'деревня лесково', 'деревня лопухинка',  
   'деревня лупплово', 'деревня малая романовка',  
   'деревня малое верево', 'деревня малое карлино',  
   'деревня малые колпаны', 'деревня мануйлово', 'деревня меньково',  
   'деревня мины', 'деревня мистолово', 'деревня ненимяки',  
   'деревня нижние осельки', 'деревня нижняя', 'деревня низино',  
   'деревня новое девяткино', 'деревня новолисино', 'деревня нурма',  
   'деревня оржицы', 'деревня парицы', 'деревня пельгора',  
   'деревня пеники', 'деревня пижма', 'деревня пикколово',  
   'деревня пудомяги', 'деревня пустынка', 'деревня пчева',  
   'деревня рабитицы', 'деревня разбегаево', 'деревня раздолье',  
   'деревня разметелево', 'деревня раполово', 'деревня реброво',  
   'деревня русско', 'деревня сижно', 'деревня снегиревка',  
   'деревня старая', 'деревня старая пустошь',  
   'деревня старое хинколо', 'деревня старополье',  
   'деревня старосиверская', 'деревня старые бегуницы',  
   'деревня суоранда', 'деревня сяськелево', 'деревня тарасово',  
   'деревня терпилицы', 'деревня тихковицы', 'деревня тойворово',  
   'деревня торосово', 'деревня торошковичи', 'деревня трубников бор',  
   'деревня фалилеево', 'деревня федоровское', 'деревня хапо-ое',  
   'деревня хязельки', 'деревня чудской бор', 'деревня шпаньково',  
   'деревня щеглово', 'деревня юкки', 'деревня ялгин',  
   'деревня яльгелево', 'деревня ям-тесово', 'зеленогорск',  
   'ивангород', 'каменногорск', 'кингисепп', 'кириши', 'кировск',  
   'колпино', 'коммунар', 'коттеджный поселок кивеннапа север',  
   'коттеджный поселок лесное', 'коттеджный поселок счастье',
```

'красное село', 'кронштадт', 'кудрово', 'лодейное поле',  
'ломоносов', 'луга', 'любань', 'муриново', 'никольское',  
'новая ладога', 'отрадное', 'павловск', 'петергоф', 'пикалево',  
'подпорожье', 'поселок александровская', 'поселок алексеевка',  
'поселок аннино', 'поселок барышево', 'поселок белоостров',  
'поселок большая ижора', 'поселок бугры', 'поселок будогощь',  
'поселок важины', 'поселок виллози', 'поселок вознесенье',  
'поселок возрождение', 'поселок войсковицы', 'поселок войскорово',  
'поселок володарское', 'поселок вырица', 'поселок высокоключевой',  
'поселок гаврилово', 'поселок гарболово', 'поселок гладкое',  
'поселок глахово', 'поселок глебычево', 'поселок гончарово',  
'поселок громово', 'поселок дзержинского', 'поселок дружная горка',  
'поселок дружноселье', 'поселок дубровка', 'поселок елизаветино',  
'поселок ефимовский', 'поселок жилгородок', 'поселок жилпоселок',  
'поселок житково', 'поселок заводской', 'поселок запорожское',  
'поселок зимитицы', 'поселок ильичево', 'поселок имени морозова',  
'поселок имени свердлова', 'поселок калитино', 'поселок каложицы',  
'поселок кикерино', 'поселок кингисеппский', 'поселок кирпичное',  
'поселок кобралово', 'поселок кобринское', 'поселок коммунары',  
'поселок кондратьево', 'поселок коробицыно', 'поселок котельский',  
'поселок красная долина', 'поселок красносельское',  
'поселок красный бор', 'поселок кузнечное',  
'поселок кузьмоловский', 'поселок лебяжье', 'поселок левашово',  
'поселок ленинское', 'поселок лесное', 'поселок лесогорский',  
'поселок лисий нос', 'поселок лукаши', 'поселок любань',  
'поселок мга', 'поселок мельниково', 'поселок металлострой',  
'поселок мичуринское', 'поселок молодежное', 'поселок молодцово',  
'поселок мыза-ивановка', 'поселок назия', 'поселок никольский',  
'поселок новогорелово', 'поселок новоселье', 'поселок новый свет',  
'поселок новый учхоз', 'поселок оредеж', 'поселок павлово',  
'поселок пансионат зеленый бор', 'поселок парголово',  
'поселок первомайское', 'поселок первово', 'поселок песочный',  
'поселок петро-славянка', 'поселок петровское',  
'поселок платформа 69-й километр', 'поселок плодовое',  
'поселок плоское', 'поселок победа', 'поселок поляны',  
'поселок понтонный', 'поселок почап', 'поселок починок',  
'поселок при железнодорожной станции вешево',  
'поселок при железнодорожной станции приветнинское',  
'поселок пригородный', 'поселок приладожский', 'поселок пудость',  
'поселок пушное', 'поселок пчевжа', 'поселок рабитицы',  
'поселок ракхья', 'поселок репино', 'поселок романовка',  
'поселок ромашки', 'поселок ропша', 'поселок рощино',

```
'поселок рябово', 'поселок саперное', 'поселок саперный',
'поселок свирьстрой', 'поселок севастьяново', 'поселок селезнево',
'поселок сельцо', 'поселок семиозерье', 'поселок семрино',
'поселок серебрянский', 'поселок сиверский', 'поселок синявино',
'поселок советский', 'поселок совхозный', 'поселок сосново',
'поселок станции вещево', 'поселок станции громово',
'поселок станции корнево', 'поселок станции лужайка',
'поселок станции приветнинское', 'поселок станции свирь',
'поселок старая малукса', 'поселок стеклянный', 'поселок стрельна',
'поселок суда', 'поселок сумино', 'поселок суходолье',
'поселок тайцы', 'поселок тельмана', 'поселок терволово',
'поселок тесово-4', 'поселок токсово', 'поселок торковичи',
'поселок торфяное', 'поселок углово', 'поселок ульяновка',
'поселок усть-ижора', 'поселок усть-луга', 'поселок ушки',
'поселок федоровское', 'поселок форносово',
'поселок форт красная горка', 'поселок цвелодубово',
'поселок цылево', 'поселок шугозеро', 'поселок шушары',
'поселок щеглово', 'поселок янино-1', 'приморск', 'приозерск',
'пушкин',
'садоводческое некоммерческое товарищество лесная поляна',
'садовое товарищество новая ропша',
'садовое товарищество приладожский', 'садовое товарищество рапхья',
'садовое товарищество садко', 'санкт-петербург', 'светогорск',
'село копорье', 'село никольское', 'село павлово', 'село паша',
'село путилово', 'село рождествено', 'село русско-высоцкое',
'село старая ладога', 'село шум', 'сертолово', 'сестрорецк',
'сланцы', 'сосновый бор', 'сясьстрой', 'тихвин', 'тосно',
'шлиссельбург'], dtype=object)
```

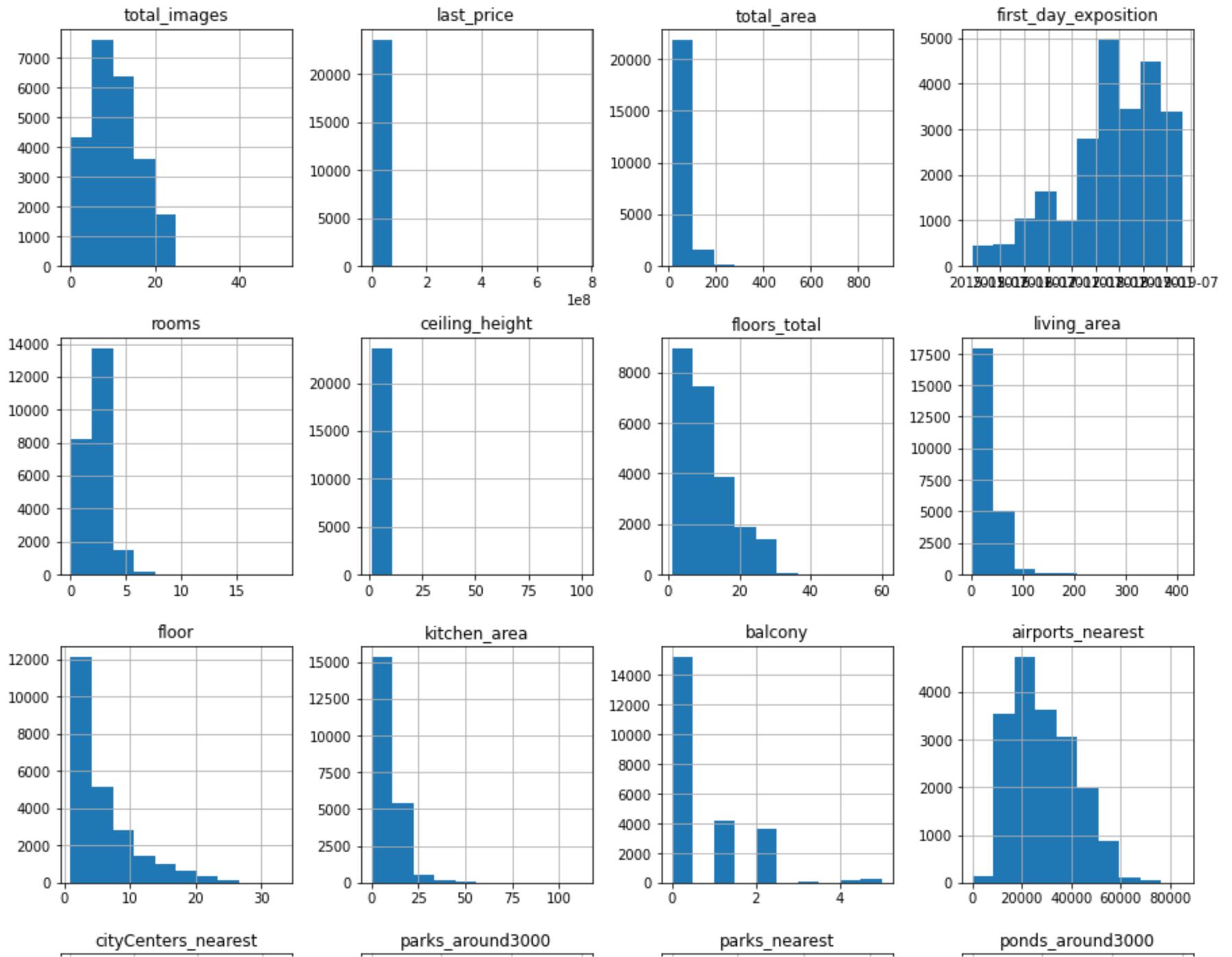
#### ✓ Комментарий ревьюера v1:

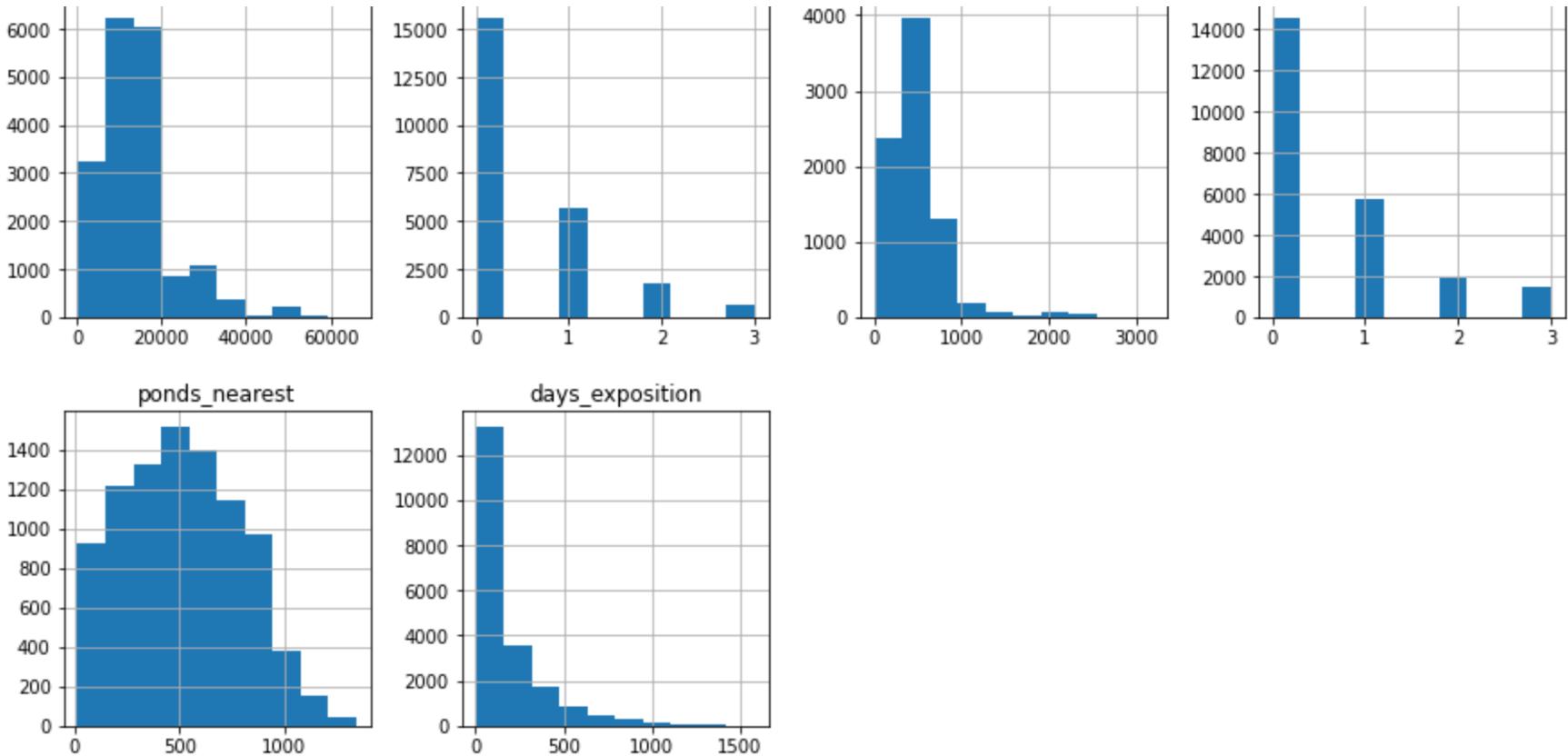
Супер! Здесь всё верно)

## Поиск и устранение редких и выбивающихся значений

Для поиска и устранения редких и выбывающихся значений обратимся к уже построенным гистограммам, по которым можно определить столбцы, в которых явно присутствуют выбросы. К таковым можно отнести: `last_price`, `total_area`, `rooms`, `ceiling_height`, `floors_total`, `living_area`, `kitchen_area`, `balcony`.

In [48]: `data.hist(figsize=(15, 20));`





Для начала рассмотрим столбцы датафрейма с выбивающимися значениями, где могли быть допущены ошибки при вводе данных. К таковым может относиться `ceiling_height`. На гистограмме видно, что значения идут к 100м. - трудно представить себе такой потолок. Для начала выведем список уникальных значений столбца.

```
In [49]: data['ceiling_height'].sort_values().unique()
```

```
Out[49]: array([ 1. ,  1.2 ,  1.75,  2. ,  2.2 ,  2.25,  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,  2.55,  2.56,  2.57,  2.58,  2.59,
   2.6 ,  2.61,  2.62,  2.63,  2.64,  2.65,  2.66,  2.67,
   2.68,  2.69,  2.7 ,  2.71,  2.72,  2.73,  2.74,  2.75,
   2.76,  2.77,  2.78,  2.79,  2.8 ,  2.81,  2.82,  2.83,
   2.84,  2.85,  2.86,  2.87,  2.88,  2.89,  2.9 ,  2.91,
   2.92,  2.93,  2.94,  2.95,  2.96,  2.97,  2.98,  2.99,
   3. ,  3.01,  3.02,  3.03,  3.04,  3.05,  3.06,  3.07,
   3.08,  3.09,  3.1 ,  3.11,  3.12,  3.13,  3.14,  3.15,
   3.16,  3.17,  3.18,  3.2 ,  3.21,  3.22,  3.23,  3.24,
   3.25,  3.26,  3.27,  3.28,  3.29,  3.3 ,  3.31,  3.32,
   3.33,  3.34,  3.35,  3.36,  3.37,  3.38,  3.39,  3.4 ,
   3.42,  3.43,  3.44,  3.45,  3.46,  3.47,  3.48,  3.49,
   3.5 ,  3.51,  3.52,  3.53,  3.54,  3.55,  3.56,  3.57,
   3.58,  3.59,  3.6 ,  3.62,  3.63,  3.65,  3.66,  3.67,
   3.68,  3.69,  3.7 ,  3.75,  3.76,  3.78,  3.8 ,  3.82,
   3.83,  3.84,  3.85,  3.86,  3.87,  3.88,  3.9 ,  3.93,
   3.95,  3.98,  4. ,  4.06,  4.1 ,  4.14,  4.15,  4.19,
   4.2 ,  4.25,  4.3 ,  4.37,  4.4 ,  4.45,  4.5 ,  4.65,
   4.7 ,  4.8 ,  4.9 ,  5. ,  5.2 ,  5.3 ,  5.5 ,  5.6 ,
   5.8 ,  6. ,  8. ,  8.3 ,  10.3 ,  14. ,  20. ,  22.6 ,
  24. ,  25. ,  26. ,  27. ,  27.5 ,  32. ,  100. ])
```

Учитывая значения 32м., 26м., 24м, можно предположить ошибку при вводе данных и это 3,2м, 2,6м., 2,4м. Для исправления этой ошибки сделаем выборку, в которой значения превышают 20м. и разделим на 10.

Кроме слишком высоких потолков, можно заметить и слишком низкие - 1м., 1.2м. Найдем количество объявлений, где высота потолков меньше 2м.

```
In [50]: len(data.loc[data['ceiling_height'] < 2, 'ceiling_height'])
```

```
Out[50]: 3
```

Объявлений со слишком низкими потолками всего 3. Удаление этих данных не изменит общей картины при анализе. (сложно определить вероятную причину появления подобных данных)

Далее изменим выбивающиеся значения, такие как 32м. и 25м. и избавимся от объявлений с потолками ниже 2м.

```
In [51]: data.loc[data['ceiling_height'] >= 20, 'ceiling_height'] = data['ceiling_height']/10
```

```
In [52]: data = data.query('ceiling_height >= 2')
```

```
In [53]: data.loc[(data['ceiling_height'] < 2) & (data['ceiling_height'] >= 20), 'ceiling_height']  
# Потолков ниже 2м. и выше 20м. в таблице больше нет
```

```
Out[53]: Series([], Name: ceiling_height, dtype: float64)
```

Определить по гистограммам аналогичные ошибки в других столбцах проблематично. Например, `living_area` также имеет выбивающиеся значения. Можно посмотреть подробнее и выяснить нет ли там ошибок. Для этого возьмем жилую площадь более 80 м<sup>2</sup>, и выведем данные об общей площади, чтобы убедиться, не превышает ли жилая площадь общую площадь помещения и данные правдоподобны.

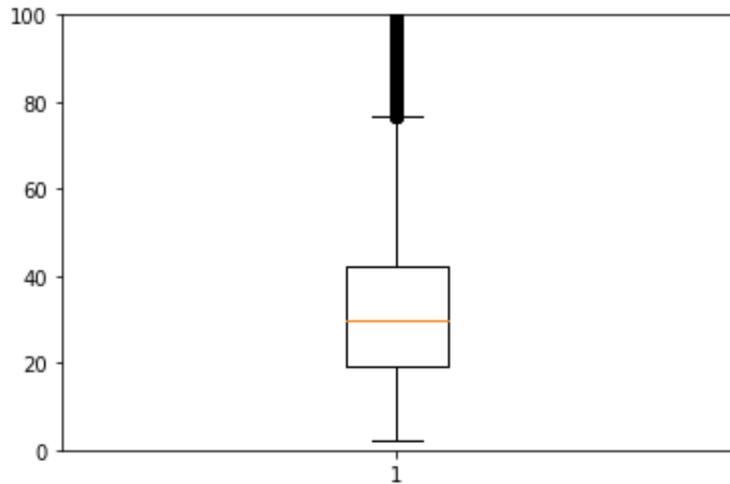
```
In [54]: data.loc[data['living_area'] >= 80, 'total_area']
```

```
Out[54]: 3      159.0  
35     149.0  
42     161.8  
52     136.0  
121    180.0  
...  
23448   180.0  
23491   250.0  
23514   136.5  
23516   139.5  
23622   114.0  
Name: total_area, Length: 797, dtype: float64
```

В целом, данные выглядят правдоподобно, но здесь стоит посмотреть на общее количество данных в выборке - от общего числа строк датафрейма это около 3 процентов. По сути мы можем отнести эти данные к выбросам. Для подтверждения построим диаграмму размаха.

```
In [55]: plt.boxplot(x=data['living_area'])  
plt.ylim(0,100)
```

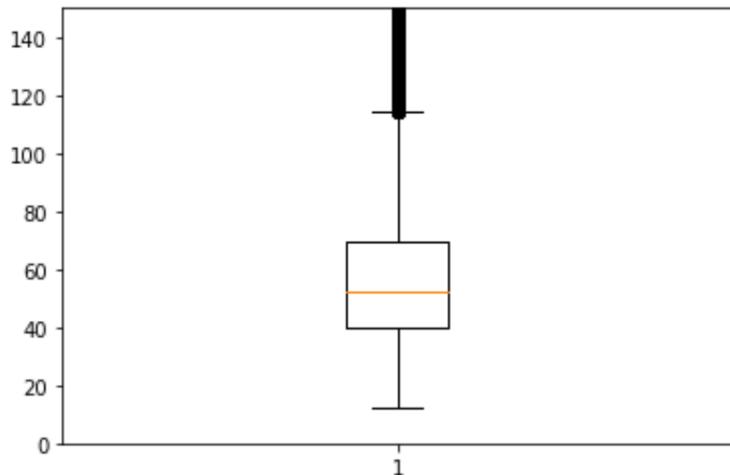
```
Out[55]: (0.0, 100.0)
```



Из диаграммы видно, что данные выше 80 уже можно отнести к выбросам. Аналогично посмотрим данные об общей площади.

```
In [56]: plt.boxplot(x=data['total_area'])
plt.ylim(0,150)
```

```
Out[56]: (0.0, 150.0)
```



Все что выше 120 можно относить к выбросам.

Теперь, чтобы не избавиться от слишком большого количества данных, определим столбцы, в которых выбросы существенно могут повлиять на анализ данных. В первую очередь это цена объекта недвижимости, также площади: общая, жилая и кухня и количество комнат.

Наблюдение можно считать выбросом, если оно больше, чем значение верхнего квартиля плюс 1.5 межквартильного размаха.

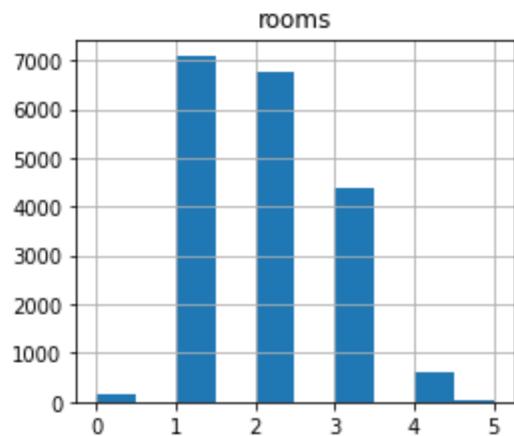
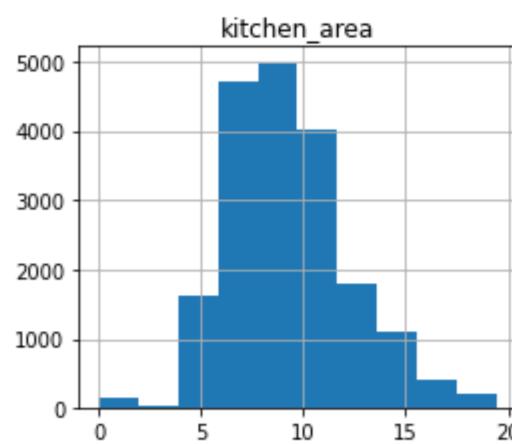
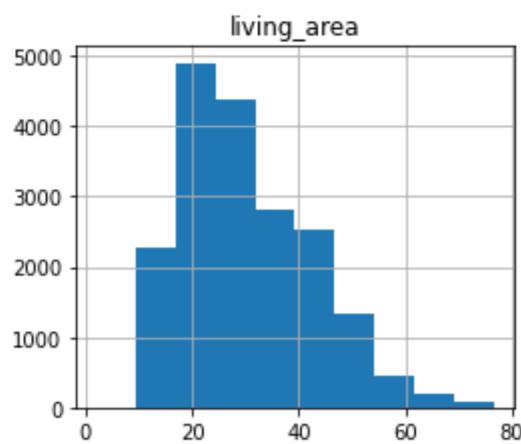
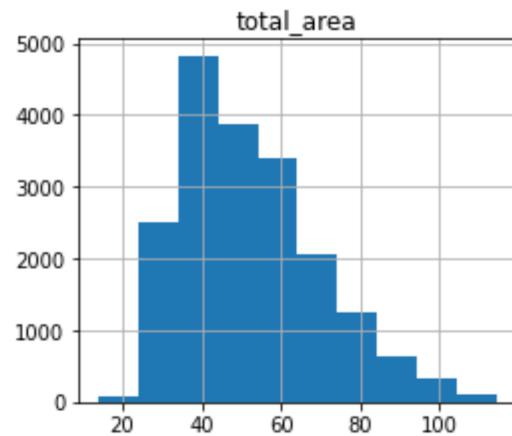
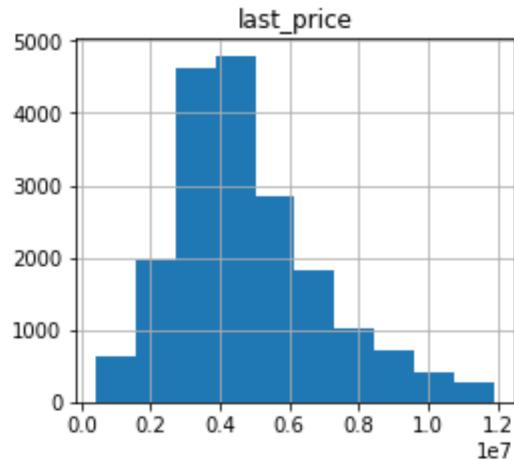
Воспользуемся данным определением, чтобы избавиться от выбросов в датафрейме. Функция ниже находит значение для столбца, за пределами которого данные в этом столбце будут относится к выбросам.

```
In [57]: def too_much(column):
    x = 1.5*(column.quantile(0.75) - column.quantile(0.25)) + column.quantile(0.75)
    return x
```

Используя эту функцию, удалим значения в датафрейме, не удовлетворяющие данным условиям

```
In [58]: data = (
    data.loc[(data['last_price'] < too_much(data['last_price'])) &
            (data['total_area'] < too_much(data['total_area'])) &
            (data['living_area'] < too_much(data['living_area'])) &
            (data['kitchen_area'] < too_much(data['kitchen_area'])) &
            (data['rooms'] < too_much(data['rooms']))]
)
```

```
In [59]: data.loc[:,['last_price', 'total_area', 'living_area', 'kitchen_area', 'rooms']].hist(figsize=(9,12));
# Построение гистограмм для столбцов, в которых были удалены выбросы
```



Построив гистограммы после удаления выбросов, можно увидеть изменения в распределении значений - данные стремятся к нормальному распределению.

✓ **Комментарий ревьюера v1:**

Диапазоны для аномальных значений подобраны и удалены верно!

Посчитайте и добавьте в таблицу новые столбцы

```
In [60]: data['sq_m_price'] = round(data['last_price'] / data['total_area'], 0)
# Создаем столбец, в котором указана цена квадратного метра объекта недвижимости
```

```
In [61]: data['day_exposition'] = data['first_day_exposition'].dt.weekday
# Создаем столбец, в котором указан день недели, когда было размещено объявление
```

```
In [62]: data['month_exposition'] = pd.DatetimeIndex(data['first_day_exposition']).month
# Создаем столбец, в котором указан месяц, когда было размещено объявление
```

```
In [63]: data['year_exposition'] = pd.DatetimeIndex(data['first_day_exposition']).year
# Создаем столбец, в котором указан год размещения объявления
```

```
In [64]: def floor_type(row):
    floor = row['floor']
    total = row['floors_total']
    if floor == 1:
        return 'первый'
    if floor == total:
        return 'последний'
    return 'другой'
```

Написанная выше функция `floor_type` возвращает тип этажа (первый, последний, другой). Аргумент функции `row` - строка датафрейма целиком, переменными `floor` и `total` обращаемся к конкретным столбцам. Далее создаем новый столбец `floor_type`, куда будут записываться результаты и применяем метод `apply` к датафрейму. Параметром `axis=1` указываем, что на вход нужно отправить строки датафрейма.

```
In [65]: data['floor_type'] = data.apply(floor_type, axis=1)
```

```
In [66]: data['km_to_cityCenter'] = round(data['cityCenters_nearest'] / 1000, 0)
# Создаем столбец, в котором указано расстояние до центра города в километрах
```

```
In [67]: data.head(10) # Выводим первые 10 строк датафрейма для проверки
```

|    | total_images | last_price | total_area | first_day_exposition | rooms | ceiling_height | floors_total | living_area | floor | is_apartment | studio | open_plan |
|----|--------------|------------|------------|----------------------|-------|----------------|--------------|-------------|-------|--------------|--------|-----------|
| 1  | 7            | 3350000.0  | 40.40      | 2018-12-04           | 1     | 2.65           | 11.0         | 18.60       | 1     | False        | False  | False     |
| 2  | 10           | 5196000.0  | 56.00      | 2015-08-20           | 2     | 2.65           | 5.0          | 34.30       | 4     | False        | False  | False     |
| 5  | 10           | 2890000.0  | 30.40      | 2018-09-10           | 1     | 2.65           | 12.0         | 14.40       | 5     | False        | False  | False     |
| 6  | 6            | 3700000.0  | 37.30      | 2017-11-02           | 1     | 2.65           | 26.0         | 10.60       | 6     | False        | False  | False     |
| 7  | 5            | 7915000.0  | 71.60      | 2019-04-18           | 2     | 2.65           | 24.0         | 41.31       | 22    | False        | False  | False     |
| 8  | 20           | 2900000.0  | 33.16      | 2018-05-23           | 1     | 2.65           | 27.0         | 15.43       | 26    | False        | False  | False     |
| 9  | 18           | 5400000.0  | 61.00      | 2017-02-26           | 3     | 2.50           | 9.0          | 43.60       | 7     | False        | False  | False     |
| 10 | 5            | 5050000.0  | 39.60      | 2017-11-16           | 1     | 2.67           | 12.0         | 20.30       | 3     | False        | False  | False     |
| 11 | 9            | 3300000.0  | 44.00      | 2018-08-27           | 2     | 2.65           | 5.0          | 31.00       | 4     | False        | False  | False     |
| 12 | 10           | 3890000.0  | 54.00      | 2016-06-30           | 2     | 2.65           | 5.0          | 30.00       | 5     | False        | False  | False     |

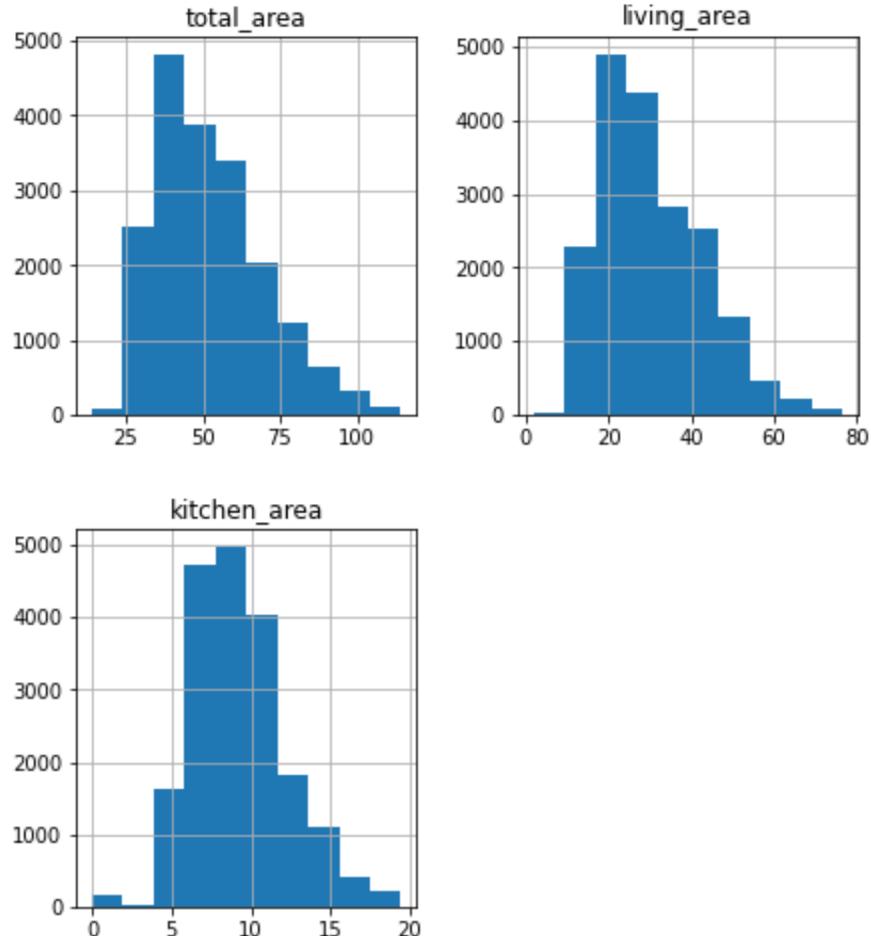
#### ✓ Комментарий ревьюера v1:

Все необходимые колонки добавлены. Идём дальше)

## Проведите исследовательский анализ данных

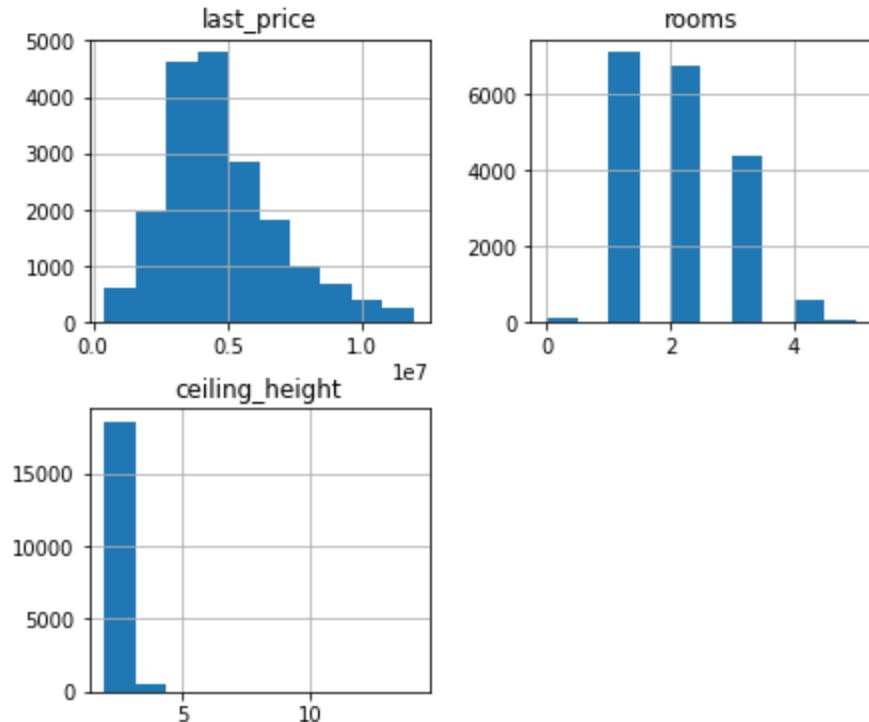
### Изучение отдельных данных датасета

```
In [68]: data.loc[:,['total_area', 'living_area', 'kitchen_area']].hist(figsize=(7,8));  
# гистограммы для общей площади, жилой площади и площади кухни
```



Полученные гистограммы можно описать как тяготеющие к нормальному. То, что гистограммы имеют резкий рост значений - пик - и более плавный спад, можно объяснить, что площадь имеет какие-то минимальные значения, т.е. крайне редко встречаются жилые помещения меньше 25 кв.м. общей площади или 10 кв.м. жилой. На каждой гистограмме можно увидеть пик наиболее часто встречающихся значений. Чаще всего продается недвижимость с общей площадью около 40 кв.м., жилой площадью 20 кв.м. и площадью кухни около 9 кв.м. Площадь кухни равная нулю говорит о количестве студий среди объявлений.

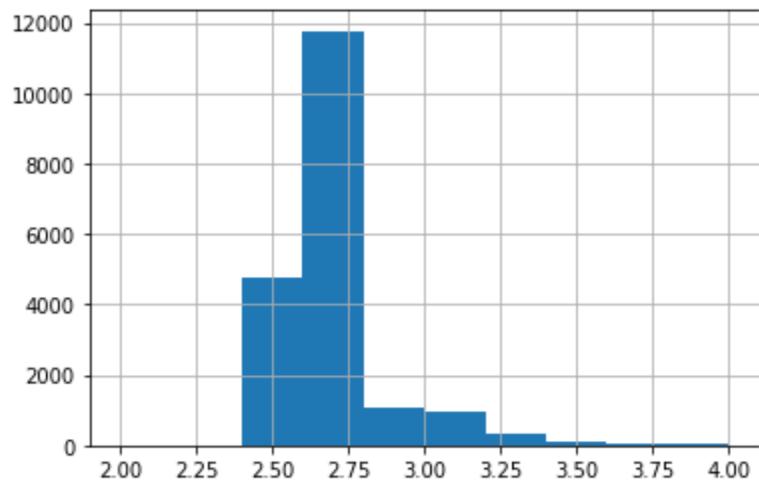
```
In [69]: data.loc[:,['last_price', 'rooms', 'ceiling_height']].hist(figsize=(7,6));  
# гистограммы для окончательной цены недвижимости, количества комнат и высоты потолков
```



Гистограмма, отображающая цену указывает, что в большинстве объявлений цена указана в районе 4 млн., найти что-то дешевле 2.5 млн. будет проблематично. Также на рынке есть редкие объекты с ценой выше 10 млн. В абсолютном большинстве случаев это 1-2х комнатные помещения. Для высоты потолков построим отдельную гистограмму, т.к. в датафрейме остались уникальные объекты с 10-метровыми потолками.

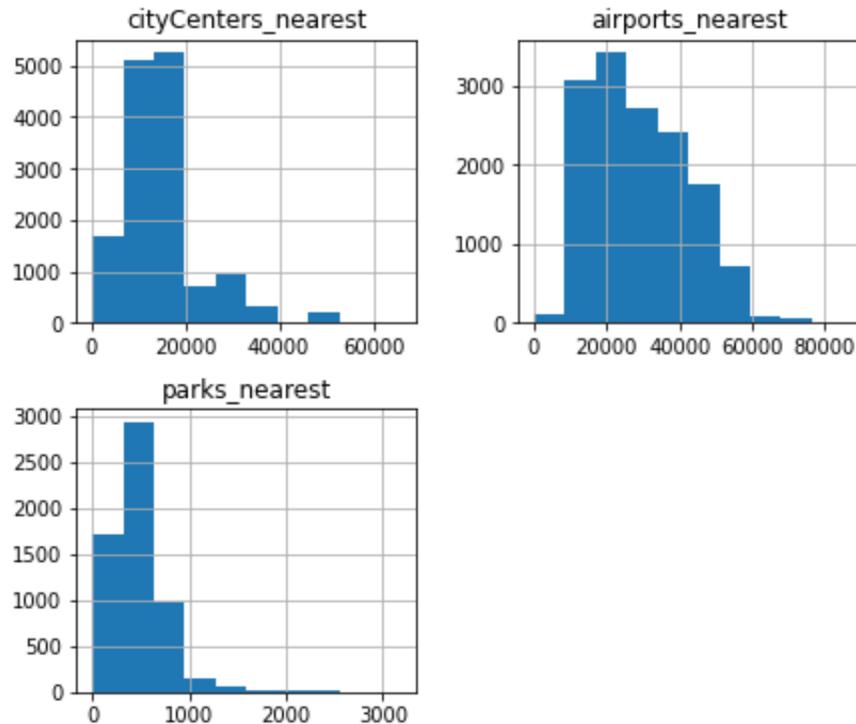
```
In [70]: data['ceiling_height'].hist(range=(2, 4))
```

```
Out[70]: <AxesSubplot:>
```



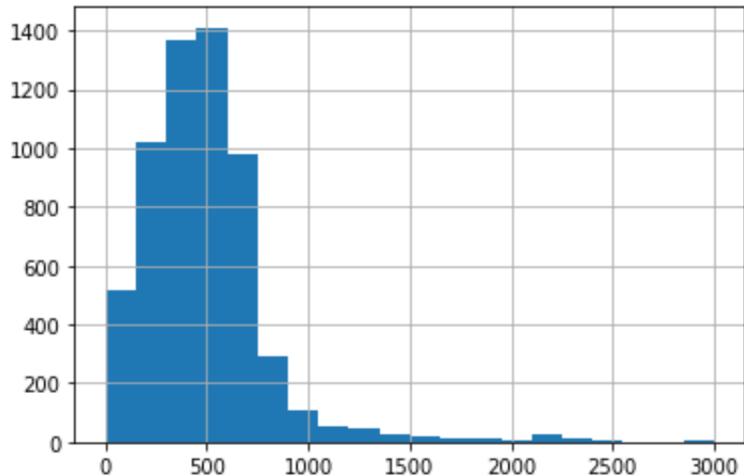
Большинство помещений с потолками высотой в районе 2.6 - 2.7 метра.

```
In [71]: data.loc[:,['cityCenters_nearest', 'airports_nearest', 'parks_nearest']].hist(figsize=(7,6));  
# гистограммы для расстояния до центра города, расстояния до аэропорта, расстояния до ближайшего парка
```

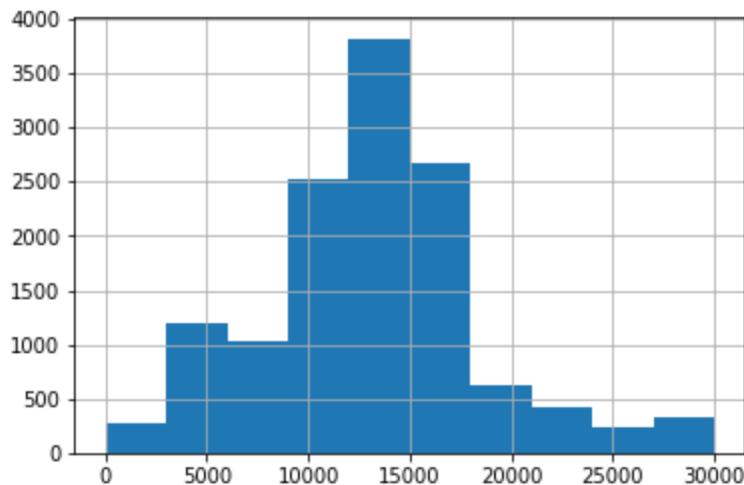


Для анализа построенных выше гистограмм нужно помнить, что в этих столбцах указаны большие значения взамен пустых строк, как обозначения, что эти объекты очень далеко. Т.е. в большинстве объявлений нет proximity парка. До аэропорта в значительной части объявлений также довольно далеко и это, кстати, коррелирует с гистограммой о расстоянии до центра города, где в большинстве объявлений указано расстояние до 25 км., т.е. чем ближе к центру города, тем дальше до аэропорта.

```
In [72]: data['parks_nearest'].hist(bins=20, range=(0,3000))
plt.show()
data['cityCenters_nearest'].hist(range=(0,30000))
```

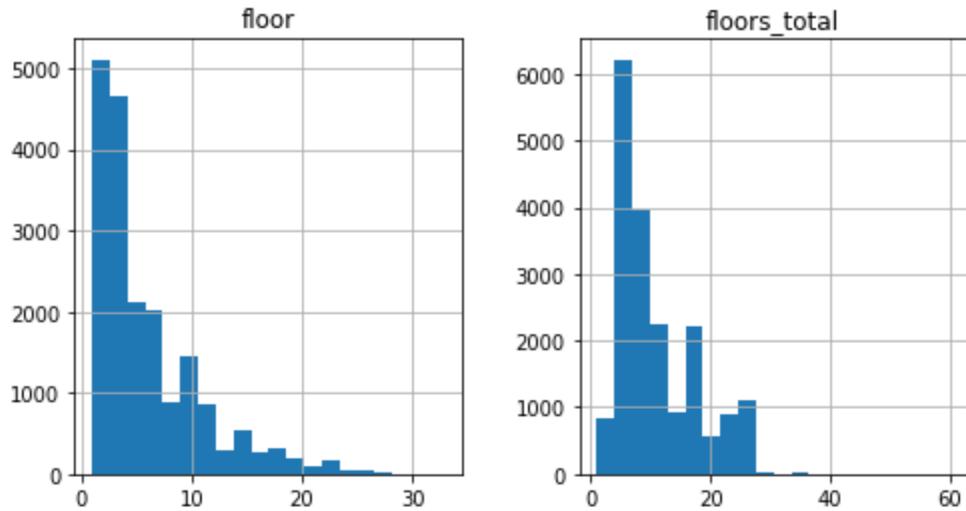


Out[72]: <AxesSubplot:>

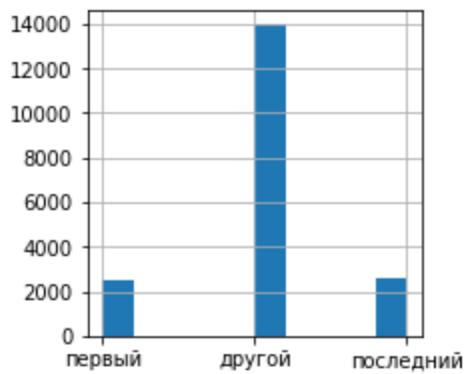


Построив гистограммы с количеством объектов непосредственно рядом с парками или недалеко от центра города, можно увидеть, что в целом таких объектов не так много.

```
In [73]: data.loc[:,['floor', 'floors_total']].hist(bins=20, figsize=(8,4))
# гистограммы для описания, на каком этаже находятся квартиры и общее количество этажей в доме
plt.show()
data['floor_type'].hist(figsize=(3,3)) # гистограмма, отображающая тип этажа
```

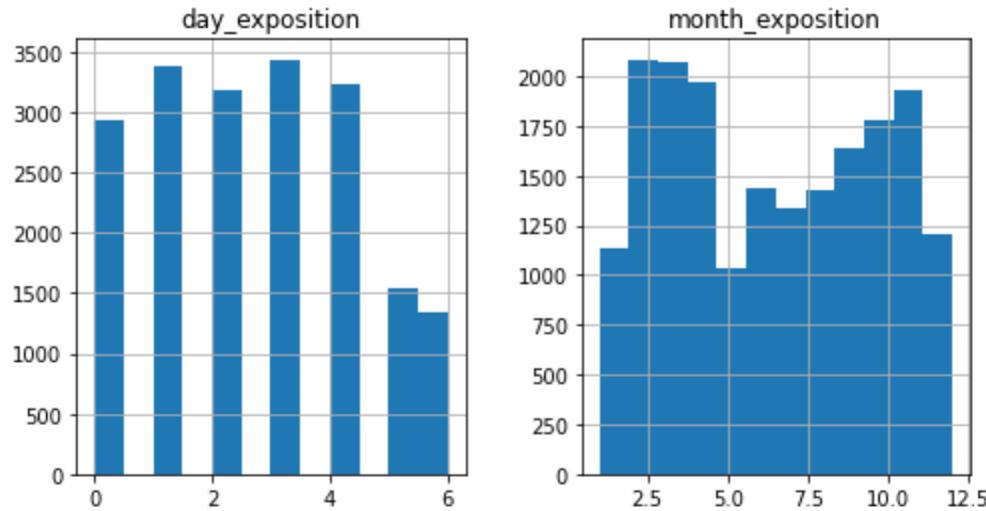


Out[73]: <AxesSubplot:>



Построенные выше гистограммы указывают, что среди объявлений лидируют дома с этажностью менее 10, сами объекты распределены в основном между 2 и 5 этажами.

In [74]: `data.loc[:,['day_exposition', 'month_exposition']].hist(bins=12, figsize=(8,4));  
# гистограммы, отображающие день и месяц размещения объявления`



На гистограммах выше четко видно, что большинство объявлений размещается в будние дни. Относительно календарного года - чаще всего это в период февраль-апрель. Провал в мае можно объяснить праздниками и длинными выходными, когда люди включаются в садово-огородный сезон. Декабрь и январь - новый год, подготовка и отдых - также не самый популярный период для размещения объявлений

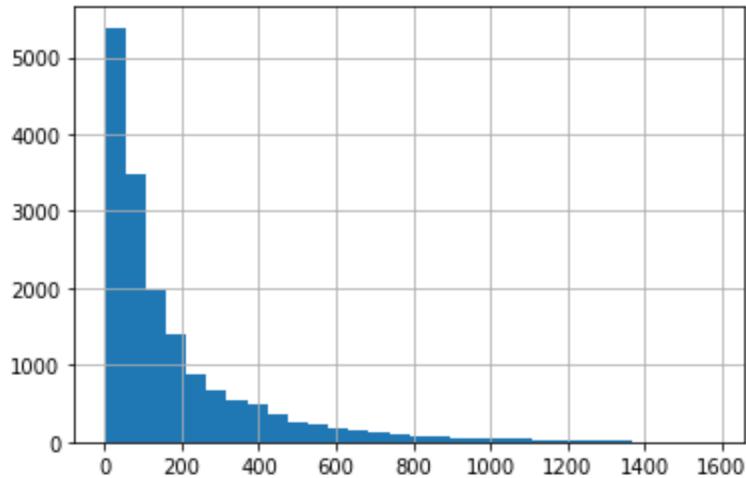
✓ **Комментарий ревьюера v1:**

Ты корректно подбираешь основные диапазоны, а также верно читаешь графики!

### Длительность размещения объявления

Далее проанализируем, сколько в среднем размещены объявления о продаже - построим гистограмму и найдем среднее значение и медиану.

```
In [75]: data['days_exposition'].hist(bins=30); # гистограмма, отображающая сколько дней были размещены объявления
```



```
In [76]: data['days_exposition'].describe()
```

```
Out[76]: count    16645.000000
mean      175.758486
std       212.609561
min       1.000000
25%      42.000000
50%      95.000000
75%     224.000000
max     1580.000000
Name: days_exposition, dtype: float64
```

По гистограмме можно увидеть, что абсолютное большинство объявлений удаляются через полгода после размещения. После вызова метода `describe()` можно увидеть среднее значение и медиану и насколько они отличаются. В данном случае оценивать по среднему будет не корректно, поскольку мы также видим, что минимальное значение - 1, и трудно сказать, было ли объявление удалено из-за быстрой продажи или по другим причинам. Аналогично с максимальным значением - более 4 лет, и возможно объявление было продано быстрее, но объявление забыли удалить. Поэтому оценим по медиане - 95 дней. Меньшие значения можно отнести к необычайно быстрым продажам. Что касается необычайно долгих продаж, здесь можно вернуться к гистограмме и сказать, что это все, что более полугода. И, подводя итог, можно сказать, что в среднем объявление размещено 3-6 месяцев.

## Поиск зависимости стоимости недвижимости от различных факторов

Проведем анализ зависимости цены от площади помещений (общей, жилой, кухни), сколько комнат, на каком этаже находится объект, когда разместили объявление.

```
In [77]: data.pivot_table(index='last_price', values=[ 'total_area', 'living_area', 'kitchen_area'])
```

```
Out[77]:
```

| last_price        | kitchen_area | living_area | total_area |
|-------------------|--------------|-------------|------------|
| <b>430000.0</b>   | 6.00         | 16.000000   | 30.400000  |
| <b>450000.0</b>   | 6.60         | 23.533333   | 40.633333  |
| <b>470000.0</b>   | 8.25         | 30.000000   | 42.750000  |
| <b>490000.0</b>   | 5.50         | 17.000000   | 29.100000  |
| <b>500000.0</b>   | 7.80         | 21.033333   | 38.900000  |
| ...               | ...          | ...         | ...        |
| <b>11858000.0</b> | 13.80        | 36.200000   | 76.100000  |
| <b>11866860.0</b> | 10.90        | 67.800000   | 107.800000 |
| <b>11879250.0</b> | 12.35        | 36.150000   | 70.500000  |
| <b>11880000.0</b> | 15.20        | 51.500000   | 92.800000  |
| <b>11894400.0</b> | 10.63        | 38.400000   | 70.800000  |

1929 rows × 3 columns

Из сводной таблицы выше сложно провести адекватный анализ - мы получили около 2000 строк, и сложно проанализировать их между собой. Но даже сейчас можно заметить, что не всегда цена выше у объектов с большей площадью (как пример 4 строка, 4.9 млн.). Это может говорить о влиянии других факторов - например близость к центру города. Но все же для более удобного анализа разобъем объявления на категории, в зависимости от цены объекта. И построенная ранее гистограмма показывает, на какие группы стоит разделить объявления.

```
In [78]: def price_cat(col):
    if col < 3000000:
        return '0 - 3 млн'
    if col > 3000000 and col < 4000000:
        return '03 - 4 млн'
    if col > 4000000 and col < 5000000:
        return '04 - 5 млн'
    if col > 5000000 and col < 6000000:
        return '05 - 6 млн'
    if col > 6000000 and col < 8000000:
        return '06 - 8 млн'
    if col > 8000000 and col < 10000000:
        return '08 - 10 млн'
    if col > 10000000 and col < 12000000:
        return '10 - 12 млн'
    return 'Более 12 млн'
```

Функция `price_cat` делит объявления на категории, в зависимости от стоимости объекта. После чего создаем новый столбец, куда помещаем полученные результаты, применив эту функцию к столбцу `last_price`

```
In [79]: data['price_cat'] = data['last_price'].apply(price_cat)
```

Далее создадим сводную таблицу по новому столбцу и построим график к этой таблице

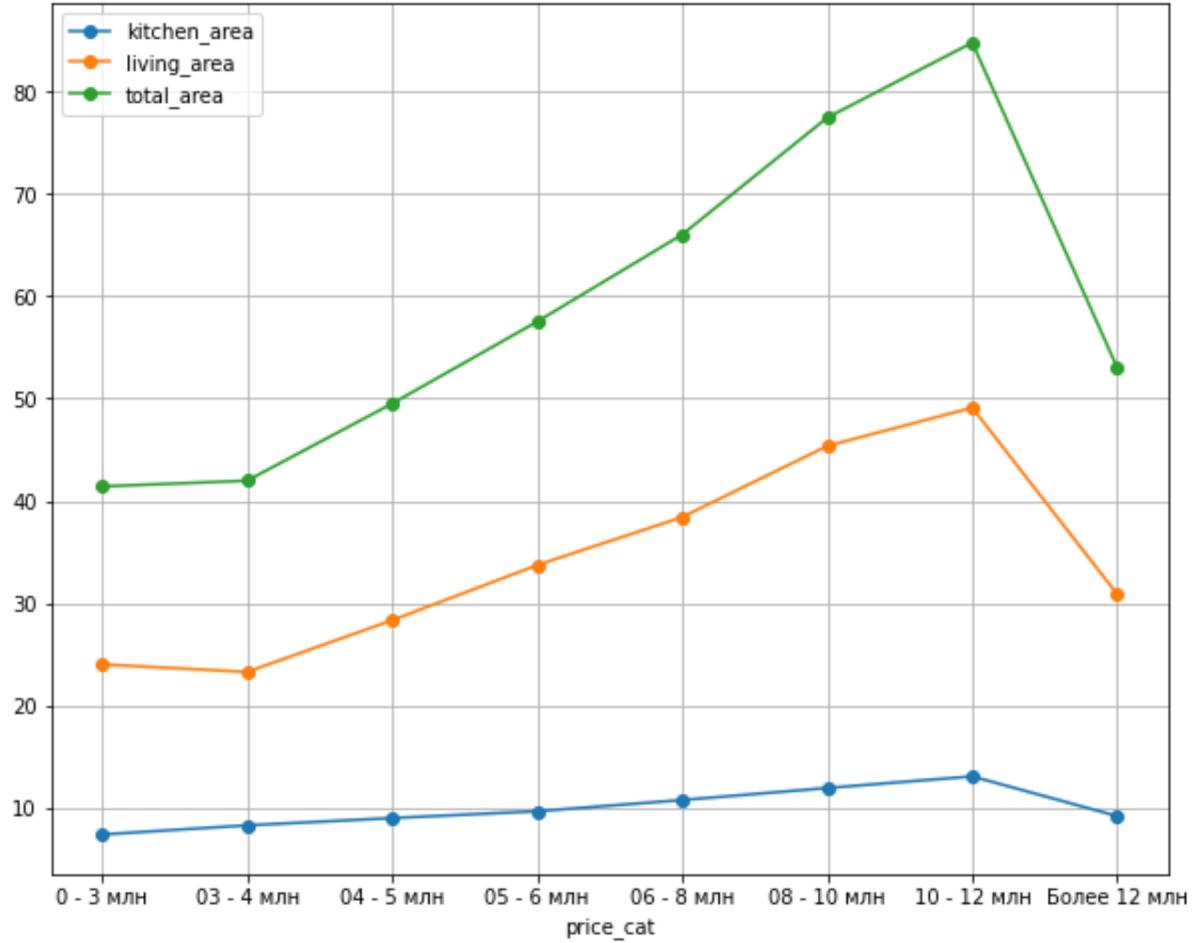
```
In [80]: price_tab = data.pivot_table(index='price_cat', values=['total_area', 'living_area', 'kitchen_area'])
```

```
In [81]: price_tab
```

Out[81]:

| price_cat           | kitchen_area | living_area | total_area |
|---------------------|--------------|-------------|------------|
| <b>0 - 3 млн</b>    | 7.356806     | 24.000277   | 41.400975  |
| <b>03 - 4 млн</b>   | 8.255015     | 23.256072   | 41.963235  |
| <b>04 - 5 млн</b>   | 8.963654     | 28.313363   | 49.504818  |
| <b>05 - 6 млн</b>   | 9.630103     | 33.703165   | 57.522431  |
| <b>06 - 8 млн</b>   | 10.727313    | 38.403475   | 66.037585  |
| <b>08 - 10 млн</b>  | 11.907531    | 45.359180   | 77.474407  |
| <b>10 - 12 млн</b>  | 13.039934    | 49.103560   | 84.807516  |
| <b>Более 12 млн</b> | 9.154911     | 30.846399   | 53.004357  |

In [82]: `price_tab.plot(style='o-', grid=True, figsize=(10,8));`



На графике хорошо видна зависимость цены от площади в интервале от 4 до 12 млн., причем наибольший коэффициент этой линейной зависимости у кривой общей площади, наименьший - у кривой площади кухни (т.е. цена слабо зависит от площади кухни). В первом интервале (до 3 млн - 3-4 млн) можно увидеть, что с ростом цены площадь увеличивается слабо, а в случае с жилой площадью даже немного падает. То есть в ценовом сегменте до 4 млн., увеличение стоимости не говорит об обязательном увеличении площади помещения, здесь больше влияют какие-то другие факторы - где находится объект, близость инфраструктуры. Причем что-то похожее мы видим и со стоимостью более 12 млн. Когда с ростом цены, площадь начинает резко падать. По сути 12 млн - как отсечка для рынка недвижимости. Если до этого цена зависела непосредственно от самого объекта недвижимости, его площади, то после 12 млн гораздо больше влияют какие-то третий факторы, например, близость к центру, или даже какие-то субъективные факторы (элитный район, дом, подъезд...)

#### Комментарий студента:

В дополнение к графику, найдем числовые эквиваленты взаимосвязи между ценой и общей площадью, жилой и площадью кухни. Рассчитаем коэффициенты корреляции для каждой пары.

```
In [83]: data['last_price'].corr(data['total_area'])
```

```
Out[83]: 0.689230270459117
```

```
In [84]: data['last_price'].corr(data['living_area'])
```

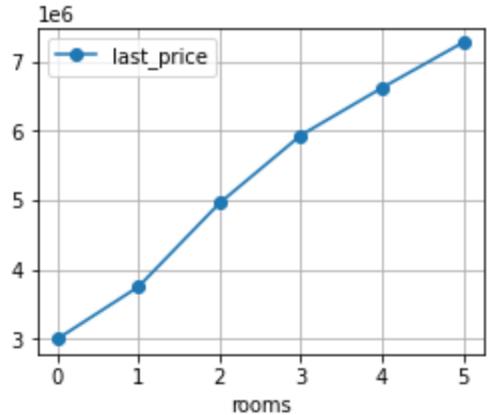
```
Out[84]: 0.569266843631126
```

```
In [85]: data['last_price'].corr(data['kitchen_area'])
```

```
Out[85]: 0.4786216962401706
```

Рассчитанные коэффициенты корреляции как подтверждение вышесказанного - между ценой и площадью есть прямая зависимость. Чем больше площадь - выше стоимость. Наибольшая взаимосвязь между ценой и общей площадью помещения. Наименьшая - между ценой и площадью кухни.

```
In [86]: data.pivot_table(index='rooms', values='last_price').plot(style='o-', grid=True, figsize=(4,3));
```



Построив график зависимости стоимости от количества комнат, можно увидеть практически линейную зависимость - с увеличение количества комнат, растет и стоимость.

```
In [87]: data['last_price'].corr(data['rooms'])
```

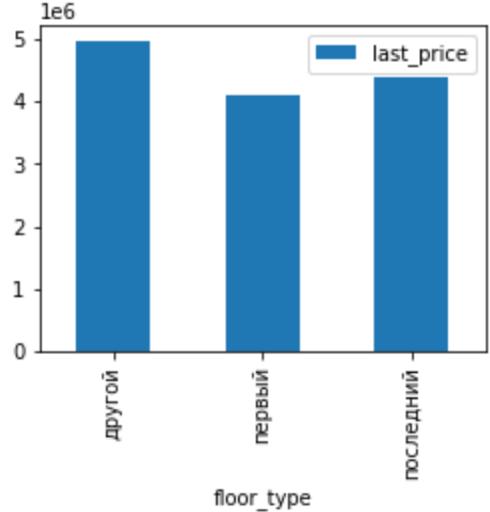
```
Out[87]: 0.43474549714305655
```

Рассчитанный коэффициент корреляции подтверждает описанный график - с ростом числа комнат растет и стоимость. Но здесь стоит также отметить - из-за графика может сложиться ложное впечатление, что стоимость растет прямо пропорционально увеличению количества комнат, и двухкомнатная квартира будет в два раза дороже однокомнатной. Коэффициент корреляции же нам говорит - зависимость безусловно есть, но, скажем так, средняя.

### ✖ Комментарий ревьюера v1:

Давай ещё добавим расчет корреляции для общей цены, общей площади, жилой площади, площади кухни и количества комнат используя .corr()

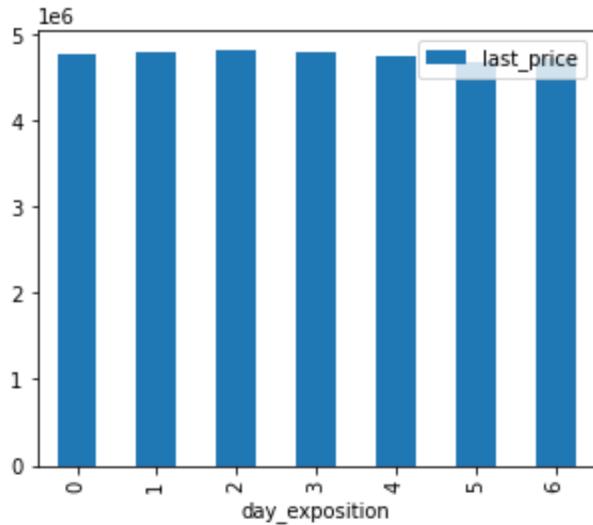
```
In [88]: data.pivot_table(index='floor_type', values='last_price').plot(kind='bar', figsize=(4,3));
```



При сравнении стоимости на разных типах этажей, получаем, что дешевле всего покупать недвижимость на первом этаже. На последнем также дешевле, чем недвижимость между первым и последним этажами.

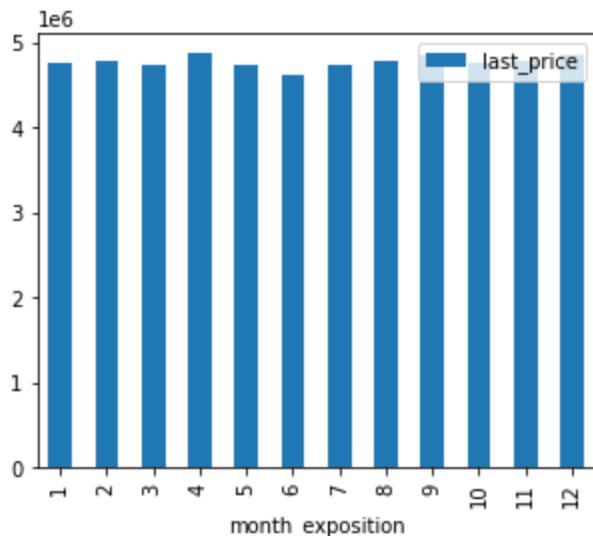
Далее оценим, зависит ли цена от даты размещения объявления. Для этого построим три гистограммы - зависимость стоимости от дня недели, месяца, года.

```
In [89]: data.pivot_table(index='day_exposition', values='last_price').plot(kind='bar', figsize=(5,4));
```



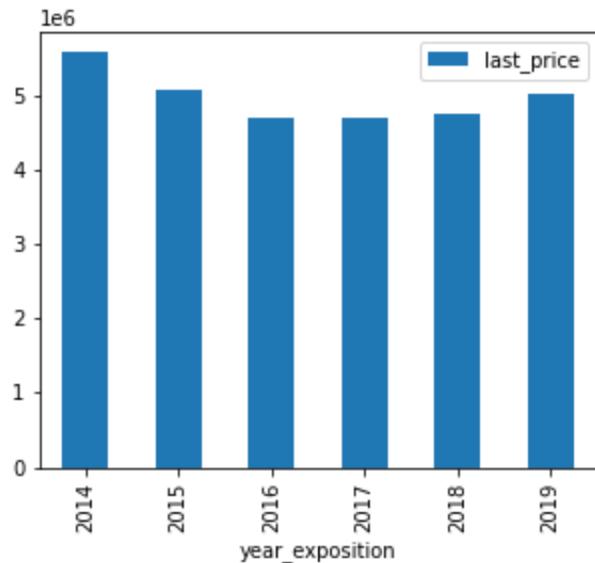
Из гистограммы видим, что день недели слабо влияет на цену недвижимости.

```
In [90]: data.pivot_table(index='month_exposition', values='last_price').plot(kind='bar', figsize=(5,4));
```



Месяц, как и день недели, не оказывает существенное влияние на ценообразование на рынке недвижимости.

```
In [91]: data.pivot_table(index='year_exposition', values='last_price').plot(kind='bar', figsize=(5,4));
```



Гистограмма зависимости цены от года размещения объявления дала интересный результат. Вопреки популярному мнению, что цена на недвижимость неуклонно растет и это удачный вариант для инвестиций - после 2014 года цены падали следующие 2 года и даже спустя пять лет, в 2019 году не достигли уровня 2014 года.

Средняя цена одного квадратного метра в 10 населённых пунктах с наибольшим числом объявлений.

Создадим сводную таблицу по населенным пунктам, где укажем количество объявлений в этом населенном пункте и среднюю цену квадратного метра там.

```
In [92]: sq_m_compare = data.pivot_table(index='locality_name_new', values=['sq_m_price'], aggfunc=['count', 'mean'])
```

```
In [93]: sq_m_compare.columns=[ 'Количество объявлений', 'Средняя цена кв.м.' ] # переименовываем названия столбцов для надлежащего вида
```

```
In [94]: sq_m_compare.sort_values('Количество объявлений', ascending=False).head(10)  
# выводим первые десять населенных пунктов с наибольшим количеством объявлений
```

Out[94]:

**Количество объявлений Средняя цена кв.м.**

| locality_name_new | Количество объявлений | Средняя цена кв.м. |
|-------------------|-----------------------|--------------------|
| санкт-петербург   | 12176                 | 105197.308640      |
| мурино            | 492                   | 86380.546748       |
| поселок шушары    | 395                   | 78882.331646       |
| кудрово           | 374                   | 95699.334225       |
| всеволожск        | 352                   | 67206.738636       |
| колпино           | 315                   | 75025.657143       |
| пушкин            | 296                   | 100196.648649      |
| поселок парголово | 287                   | 90451.111498       |
| гатчина           | 287                   | 68481.393728       |
| выборг            | 187                   | 58242.925134       |

Из таблицы выше получаем, что Санкт-Петербург лидирует как по количеству объявлений, так и по стоимости одного квадратного метра. Самый дешевый квадратный метр недвижимости, среди этой десятки - в Выборге.

Стоимость каждого километра до центра Санкт-Петербурга при формировании цены на недвижимость

В дополнение к ранее найденному расстоянию до центра в километрах, теперь необходимо выделить квартиры в Санкт-Петербурге с помощью столбца `locality_name` и вычислить среднюю цену каждого километра.

In [99]: `s_pb = data.query('locality_name_new == "санкт-петербург"')`In [100...]: `s_pb_pivot = s_pb.pivot_table(index='km_to_cityCenter', values='sq_m_price')`In [101...]: `s_pb_pivot.head(10)`

Out[101]:

**sq\_m\_price****km\_to\_cityCenter**

|            |               |
|------------|---------------|
| <b>0.0</b> | 126848.000000 |
| <b>1.0</b> | 120564.636364 |
| <b>2.0</b> | 114690.541353 |
| <b>3.0</b> | 109992.292308 |
| <b>4.0</b> | 115955.245383 |
| <b>5.0</b> | 121742.565130 |
| <b>6.0</b> | 120860.029326 |
| <b>7.0</b> | 116333.606299 |
| <b>8.0</b> | 115913.911548 |
| <b>9.0</b> | 105753.140998 |

**✓ Комментарий ревьюера v2:**

Всё верно, но лучше вывести все данные, а не ограничивать первыми 10 км

In [102...]

`s_pb_pivot.describe()`

Out[102]:

|              | sq_m_price    |
|--------------|---------------|
| <b>count</b> | 30.000000     |
| <b>mean</b>  | 101582.752001 |
| <b>std</b>   | 14228.349250  |
| <b>min</b>   | 70312.000000  |
| <b>25%</b>   | 92639.800021  |
| <b>50%</b>   | 102776.190711 |
| <b>75%</b>   | 113515.979092 |
| <b>max</b>   | 126848.000000 |

Из сводной таблицы можно увидеть, что с увеличением расстояния от центра города, стоимость недвижимости падает. В центре города цена квадратного метра - 127 тысяч, на окраине же минимальная цена -70 тысяч. Используя метод describe() находим среднее значение квадратного метра - 101,5 тысячи. При этом можно обратить внимание, что медиана и среднее практически равны, что говорит о том, что в целом по всей выборке нет каких-либо резко выбивающихся значений.

## Общий вывод

Цель данного проекта - анализ данных о продаже квартир в Санкт-Петербурге и соседних населенных пунктах. В ходе работы, на начальном этапе был проведен первичный анализ полученного датасета - выведена таблица для оценки хранящихся данных, выведена общая информация (количество столбцов, их названия, количество строк в каждом столбце, тип данных) Также построены гистограммы для всех числовых столбцов датасета. Результаты первичного анализа:

- были обнаружены пропуски в столбцах датасета (в некоторых из них, как в `parks_nearest` и `is_apartment` число пропусков составляло более половины от общего числа строк)
- в числовых столбцах присутствуют выбросы, искажающие общую картину

Далее была выполнена предобработка данных. Для заполнения пропусков было выполнено следующее:

- где было возможно выявить зависимость между данными датасета, данные были заполнены в соответствии с этой зависимостью (столбец `living_area`)
- была выдвинута гипотеза, что незаполненные данные говорят об отсутствии таких параметров. Например, в столбцах о балконах или наличии парка в радиусе 3000м - если пропуск, значит этого нет, можно менять на ноль.
- столбцы `cityCenters_nearest`, `airports_nearest`, `parks_nearest`, `ponds_nearest` остались без изменений. Заполнение пропусков могло привести к искажению результатов при анализе.
- в столбцах `kitchen_area`, `total_floors` пропуски составляли незначительную долю, столбцы остались без изменений.

Пропуски в столбце `days_exposition` означают, что объявление еще не снято, поэтому были оставлены без изменений.

Также была проведена работа по устранению неявных дубликатов. В результате, в столбце с названиями населенных пунктов количество уникальных значений сократилось с 365 до 320. Измененные названия населенных пунктов добавлены в новый столбец `locality_name_new`.

Была проведена работа с аномалиями и выбивающимися значениями в столбцах. Опираясь на данные диаграммы размаха были удалены выбросы в столбцах `last_price`, `total_area`, `living_area`, `kitchen_area`.

Далее, для последующего анализа, в таблицу были добавлены следующие столбцы:

- цена одного квадратного метра недвижимости;
- день публикации объявления (0 - понедельник, 1 - вторник и т.д.);
- месяц публикации объявления;
- год публикации объявления;
- тип этажа квартиры (значения — «первый», «последний», «другой»);
- посчитано и добавлено в таблицу: расстояние в км до центра города;

Далее был выполнен исследовательский анализ данных, по результатам которого можно сделать следующие выводы:

1. Среди продаваемых квартир, наиболее часто встречаются одно-двухкомнатные квартиры, с общей площадью в интервале 40 - 50

кв.м., жилой площадью около 20 кв.м. и площадью кухни 8 кв.м., с высотой потолка около 2,75м. В основном цены варьируются в интервале 4-5 млн. Наиболее часто встречающаяся этажность домов - менее десяти, квартиры преимущественно расположены не на первом и последнем этажах. Доля квартир с близлежащими парками (в радиусе 3 км) - не более трети от всех объявлений, в половине случаев удаленность от центра - более 20 км, т.е. объекты расположены на окраине или за пределами Санкт-Петербурга.

2. Объявления в большинстве случаев выкладываются в будние дни. Относительно календарного года, меньше всего новых объявлений появляется в январе, мае и декабре, больше всего - февраль-апрель.

3. Что касается продолжительности размещения объявления - медианное значение составляет 95 дней, в среднем можно говорить, что объявление висит 3-6 месяцев.

4. При поиске зависимости цены от различных факторов, было обнаружено следующее:

4.1. Зависимость цены от площади явно проявляется в ценовом сегменте от 4 млн. до 12 млн. - когда цена растет с увеличением площади квартиры. Также, был обнаружен интересный факт по объектам дороже 12 млн - цена перестает зависеть непосредственно от самого объекта - его площади. Начинают больше влиять какие-то третий факторы, например, близость к центру, или даже какие-то субъективные факторы (элитный район, дом, подъезд...). Что касается "дешевых" квартир, также было замечено, что все что до 4 млн - также больше формируются больше как субъективная оценка хозяина недвижимости и зависимость от площади квартиры низкая.

4.2. Количество комнат напрямую влияет на цену - с ростом количества комнат растет и цена квартиры

4.3. Квартиры на первом этаже уступают в цене квартирам, расположенным на верхних этажах

4.4. Влияние дня недели или месяца, в котором выкладывалось объявление не оказывает существенной роли на цену. При этом было замечено общее падение цен после 2014 года.

5. Были найдены первые десять населенных пунктов по количеству объявлений. Лидирует Санкт-Петербург, где доля объявлений составляет около 50% от общего числа. В первых 10 населенных пунктах по количеству объявлений, дороже всего квадратный метр квартиры стоит в Санкт-Петербурге - 105197, дешевле всего в Выборге - 58243.

6. Чем ближе квартира к центру города, тем выше стоимость квадратного метра. Максимальная стоимость квадратного метра, то есть непосредственно в центре города, составляет 127 тысяч, в среднем же цена равна 101,5 тысячи

#### ✓ Итоговый комментарий рецензента v2:

Хорошей практикой является написание в общем выводе всех твоих действий (кратко) по проекту. А после этого нужно написать общий вывод используя информацию из промежуточных выводов после каждого раздела. Твой вывод раскрывает для заказчика всю твою проделанную работу и за что он платит деньги, даже если ему не интересен весь процесс исследования, то в отчет он заглянет обязательно.

Я рад был поработать над проверкой твоей работы) В качестве дополнительного материала для изучения могу

порекомендовать следующий ресурс: <https://www.python-graph-gallery.com/>

В нем содержится большая библиотека графиков с готовым кодом, который можно использовать при работе.

Поздравляю со сдачей проекта и желаю удачи в дальнейшем обучении! 😊