

✓ Комментарий ревьюера ✓

Привет, Ирина! Меня зовут Михаил, я буду проверять твой проект. Предлагаю общаться на 'ты' :) Однако, если это неудобно - сообщи, и мы перейдем на 'Вы'. Моя главная цель — поделиться с тобой своим опытом и помочь тебе стать аналитиком данных, а не только указать на совершенные тобой ошибки.

Видно, что к проекту приложен большой труд. Все ключевые этапы в работе выполнены, статистическое исследование проведено качественно. Так что в целом справиться с задачей тебе удалось.

Есть несколько аспектов, которые **требуют** твоего **внимания**. Комментарии по ним помечены красным цветом и символами ✗.

После их доработки проект будет принят, осталось совсем немного :)

Давай работать над проектом в диалоге: если **ты что-то меняешь** в проекте по моим рекомендациям — **пиши об этом**. Мне будет легче отследить изменения, если ты будешь использовать синюю форму, которую я прикреплю ниже. Пожалуйста, **не перемещай, не изменяй и не удаляй мои комментарии**. Всё это поможет выполнить повторную проверку твоего проекта оперативнее.

Жду проект на повторное ревью. Успехов в изучении!

✗ **Комментарий ревьюера** ✗ Так я выделяю моменты, которые требуют особого внимания. Нужно будет учесть их и внести корректировки в свою работу.

⚠ **Комментарий ревьюера** ⚠ Желтым я отмечу рекомендации, которые, могут быть полезными при твоей работе. Они носят рекомендационный характер, но будет классно, если ты будешь учитывать их при работе.

✓ **Комментарий ревьюера** ✓ Так я выделяю удачные и элегантные решения, на которые можно опираться в будущих проектах.

А в таком блоке ты можешь оставить комментарии для меня

Комментарий И 1

Привет, Михаил. Спасибо за качественное ревью и вдохновляющие комментарии, слова поддержки. Начинаю исправлять с более позитивным настроем.

✓ **Комментарий ревьюера v2** ✓

Привет, Ирина. Давай посмотрим, что получилось :)

▼ 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.

✓ **Комментарий ревьюера** ✓

Хорошее введение в проект - пригодится в будущем, когда соберешь большое портфолио и откроешь проект вновь👉

▼ Откройте файл с данными и изучите общую информацию.

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('/datasets/real_estate_data.csv', sep = '\t')
display(data.head(10))
data_new = data.copy() # сделаем копию данных и сюда будем помещать восстановленные данные без пропусков
```

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

10 rows × 22 columns

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   14584 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
```

Данные, требующие преобразования типа:

- first_day_exposition - дата публикации - дата
- floors_total - количество этажей в доме - в целочисленный
- is_apartment - апартаменты - в булев тип
- balcony - число балконов - в целочисленный
- parks_around3000 - число парков в радиусе 3 км - в целочисленный
- parks_nearest - расстояние до парка - можно округлить до целых значений
- ponds_around3000 - число водоёмов в радиусе 3 км - в целочисленный
- ponds_nearest - расстояние до водоема можно округлить до целых значений
- airports_nearest - можно округлить до целых значений
- cityCenters_nearest - расстояние до центра города можно округлить до целых значений
- days_exposition - сколько дней было размещено объявление (от публикации до снятия) - целочисленный
- площади так же можно округлить до целых

✖ Комментарий ревьюера ✖

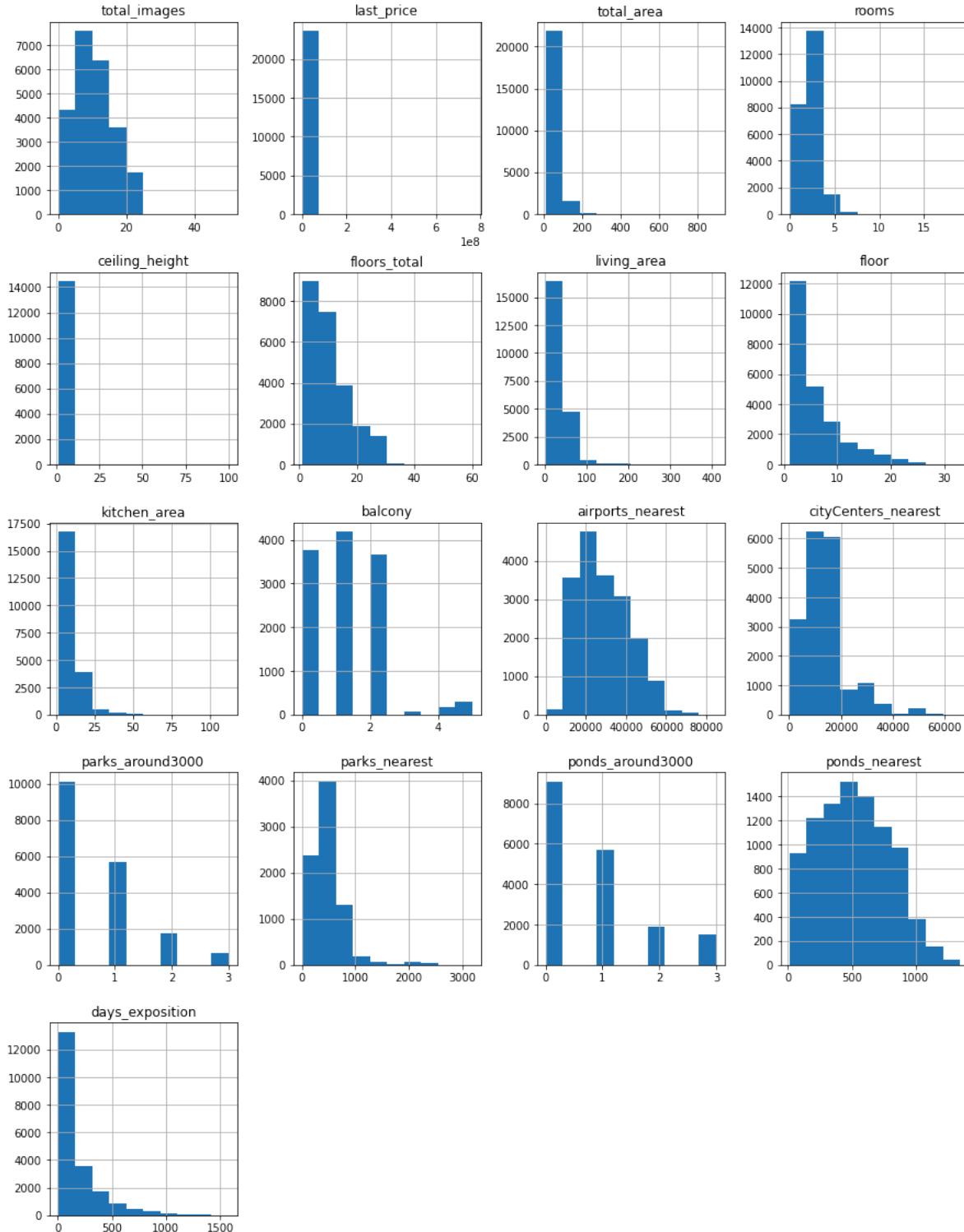
На данном этапе нас также просят построить общую гистограмму для всех числовых столбцов таблицы.

Комментарий И1

Гистограммы для всех числовых столбцов таблицы

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

```
array([[<AxesSubplot:title={'center':'total_images'}>,
       <AxesSubplot:title={'center':'last_price'}>,
       <AxesSubplot:title={'center':'total_area'}>,
       <AxesSubplot:title={'center':'rooms'}>],
      [<AxesSubplot:title={'center':'ceiling_height'}>,
       <AxesSubplot:title={'center':'floors_total'}>,
       <AxesSubplot:title={'center':'living_area'}>,
       <AxesSubplot:title={'center':'floor'}>],
      [<AxesSubplot:title={'center':'kitchen_area'}>,
       <AxesSubplot:title={'center':'balcony'}>,
       <AxesSubplot:title={'center':'airports_nearest'}>,
       <AxesSubplot:title={'center':'cityCenters_nearest'}>],
      [<AxesSubplot:title={'center':'parks_around3000'}>,
       <AxesSubplot:title={'center':'parks_nearest'}>,
       <AxesSubplot:title={'center':'ponds_around3000'}>,
       <AxesSubplot:title={'center':'ponds_nearest'}>],
      [<AxesSubplot:title={'center':'days_exposition'}>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



✓ Комментарий ревьюера v2 ✓

Хорошо👍 Гистограммы построены

▼ Предобработка данных

План:

1. Поиск и заполнение пропусков
2. Обработка аномальных значений
3. Поиск и удаление явных и неявных дубликатов
4. Поиск и удаление неявных дубликатов
5. Изменение типов данных
6. Проверка необходимости категоризации данных

В таблице 23699 записей о продажах квартир

В следующих столбцах наблюдаются пропуски:

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

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

▼ Высота потолков ceiling_height - пропуски и аномалии

Посчитаем количество пропусков

```
print('Количество пропусков в данных о высоте потолков:', data['ceiling_height'].isna().sum() )
print(f'Доля пропусков: { data["ceiling_height"].isna().mean():.0%}' )
```

```
Количество пропусков в данных о высоте потолков: 9195
Доля пропусков: 39%
```

Выведем основные числовые характеристики

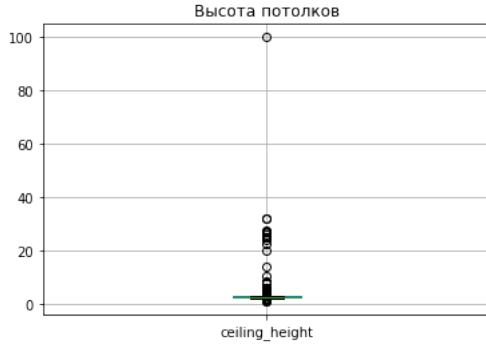
```
data['ceiling_height'].describe()

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

Построим график с квартилями, чтобы посмотреть, где сосредоточены основные значения и есть ли выбросы

```
data[['ceiling_height']].boxplot()
plt.title('Высота потолков')
```

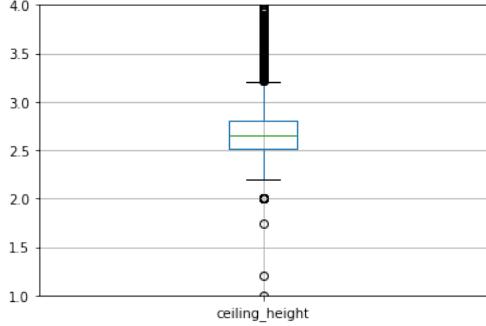
Text(0.5, 1.0, 'Высота потолков')



Зададим диапазон, от 1 до 4, чтобы лучше рассмотреть результаты

```
data[['ceiling_height']].boxplot()
plt.title('Высота потолков')
plt.ylim(1, 4)
plt.show()
```

Text(0.5, 1.0, 'Высота потолков')



Основные данные сосредоточились от 2,5 до 2.8. Значения меньше 2.5 и более 3.2 редкие.

Получаем, что максимальная высота потолков 100 м, а минимальная 1м. Очень сомнительно, что существуют квартиры с высотой потолков более 4м и ниже 2,4м, значит в данных ошибки. Но 75% данных выглядят правдоподобно. Высота потолков не более 2.8м.

Среднее значение 2,77 и медиана 2,65 близки, но лучше взять медиану для заполнения пропусков. На нее в меньшей степени влияют аномальные значения, такие как 1м и 100м.

Но прежде, чем использовать медиану для заполнения пропусков в высоте потолка, хорошо бы разобраться с аномальными значениями и рассчитать новое значение медианы и им заполнить пропуски.

Взглянем на гистограммы распределения квартир по высоте потолков

```
data.plot(
    kind='hist',
    y='ceiling_height',
    bins=10,
    range=(1, 2.4),
    label='Высота потолков от 1 до 2.4м'
)
plt.xlabel('Высота потолка')
plt.ylabel('Количество квартир в продаже')
plt.show()

data.plot(
    kind='hist',
    y='ceiling_height',
    bins=110,
    range=(2.4, 5),
    label='Высота потолков от 2.4 до 5м'
)
plt.xlabel('Высота потолка')
plt.ylabel('Количество квартир в продаже')
plt.show()

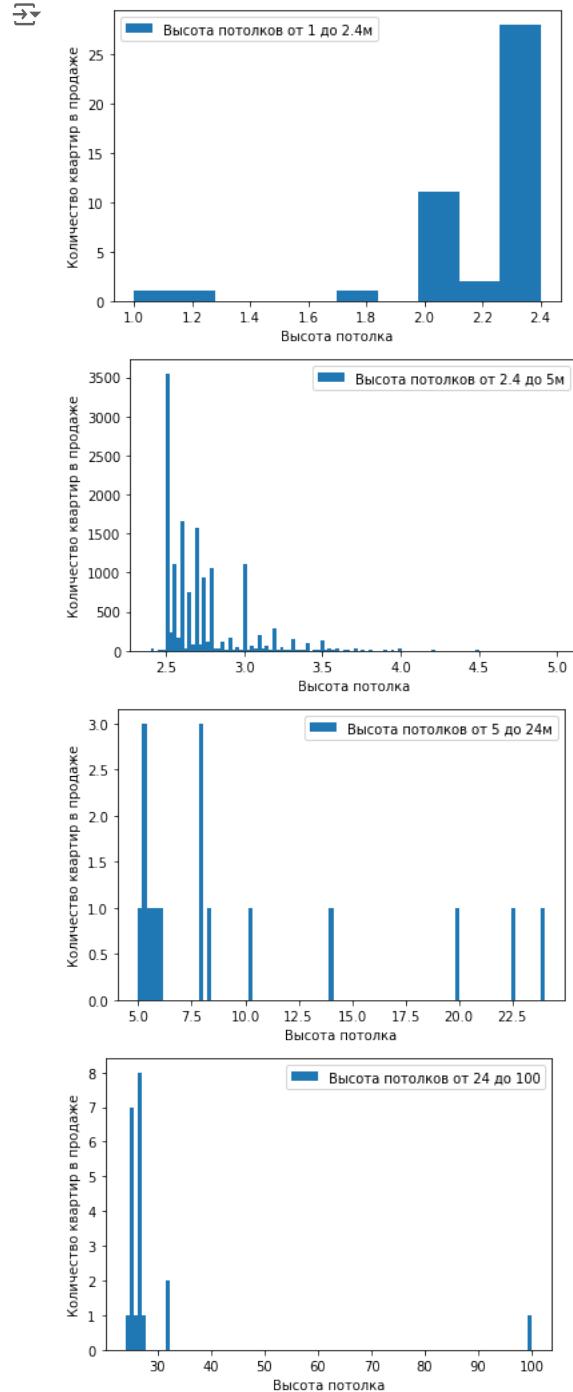
data.plot(
    kind='hist',
    y='ceiling_height',
    bins=100,
```

```

range=(5,24),
label='Высота потолков от 5 до 24м'
)
plt.xlabel('Высота потолка')
plt.ylabel('Количество квартир в продаже')
plt.show()

data.plot(
    kind='hist',
    y='ceiling_height',
    bins=100,
    range=(24,100),
    label='Высота потолков от 24 до 100'
)
plt.xlabel('Высота потолка')
plt.ylabel('Количество квартир в продаже')
plt.show()

```



Разобъем значения высот потолков на интервалы и посчитаем количество значений в каждом.

За аномально высокие потолки примем выше 4м, за аномально низкие примем ниже 2,4м

```
print('Число аномально высоких потолков, выше 4м:',
      data.query('ceiling_height > 4')['ceiling_height'].count())
print('Число аномально низких потолков, ниже 2.4м:',
      data.query('ceiling_height <= 2.4')['ceiling_height'].count())
```

➡ Число аномально высоких потолков, выше 4м: 75
 Число аномально низких потолков, ниже 2.4м: 44

Высота потолков [1; 2.4]: все значения ниже 2.4 скорее всего имеют ошибочные значения.

Высота потолков (2.4; 4]: оставляем

Высота потолков (4; 24]: ошибочные значения

Высота потолков (24; 40]: имеют ошибку в десятках и их можно заменить на значения в диапазоне 2,4 - 4,0м

Высота потолков более 40м: ошибочные значения

Для начала заменим данные, в которых есть ошибка в десятках: высота потолков (24; 40]

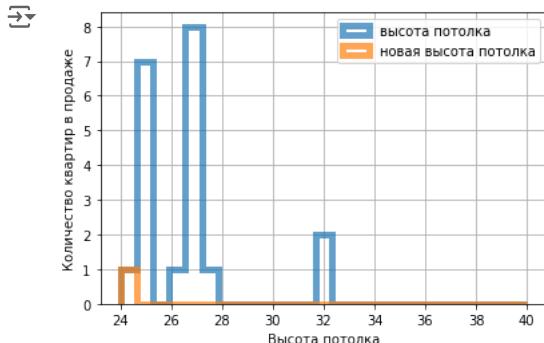
```
data_new['ceiling_height'] = \
data['ceiling_height'].\
where((data['ceiling_height']>24) & (data['ceiling_height']<=40), data['ceiling_height']/10 )
```

✓ Комментарий ревьюера ✓

Хорошее решение👍

Проверим, исчезли ли значения от 24 до 40 с гистограммы

```
ax = data.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(24, 40),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='высота потолка'
)
data_new.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(24, 40),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='новая высота потолка',
    ax=ax,
    grid=True,
    legend=True,
)
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Высота потолка')
plt.show()
```



Аномальные значения в диапазоне (24 - 40] убраны

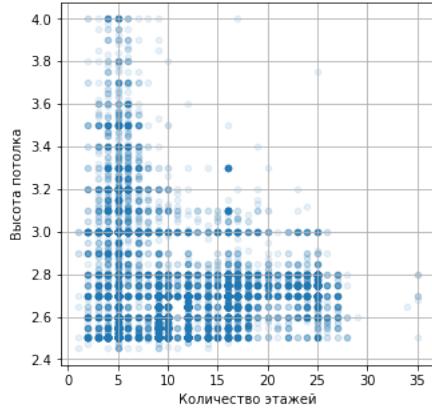
Хорошо бы понять от какого параметра зависит высота потолков и в каждой группе рассчитать медианное значение и им заполнить пропуски данных и другие аномальные значения. Можно посмотреть, есть ли связь между количеством этажей в доме и высотой

потолков и рассчитать медиану потолков для домов разной высотности, если связь есть.

Проверим зависимость высоты потолков от количества этажей в доме

```
data.\nquery('2.4<ceiling_height<=4 and floors_total<=35').\nplot(x='floors_total', y='ceiling_height', \n    kind='scatter', grid=True, figsize = (5,5), alpha=0.1,\n    xlabel='Количество этажей', ylabel = 'Высота потолка')
```

→ <AxesSubplot:xlabel='Количество этажей', ylabel='Высота потолка'>



В старых, низковысотных домах (до 10 этажей) наблюдается тенденция к наличию широкого разброса высот потолков от стандартных 2.5 до высоких 4 м. А в высотных домах данные сосредоточены более узко и высота потолков не так сильно колеблется и составляет от 2.5 до 2.8м.

Разобъем данные на две группы и рассчитаем для этих групп отдельно медианные значения высоты потолков.

Дома от 1 до 9 этажей

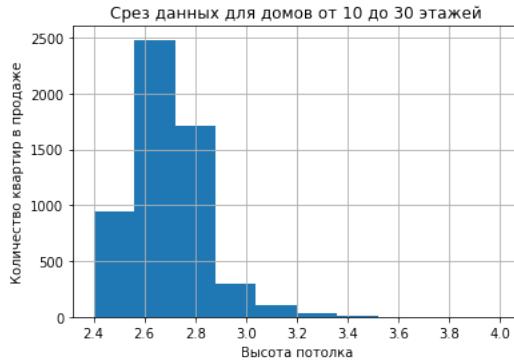
Дома от 10 до 30 этажей

Построим гистограммы распределения высоты потолка для двух срезов данных по количеству этажей

Гистограмма высот потолков для срезанных от 10 до 30 этажей

```
data.query('9 < floors_total < 31 and 2.4 < ceiling_height < 4' )['ceiling_height'].hist(range=(2.4,4), bins=10)\nplt.xlabel('Высота потолка')\nplt.ylabel('Количество квартир в продаже')\nplt.title('Срез данных для домов от 10 до 30 этажей')
```

→ Text(0.5, 1.0, 'Срез данных для домов от 10 до 30 этажей')

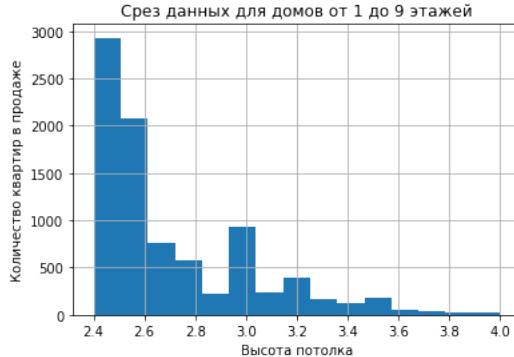


Гистограмма имеет один пик и нет длинного хвоста, для домов высотой от 10 до 30 этажей подойдет заполнение пропусков и ошибочных значений медианым значением высоты потолка.

Гистограмма высоты потолков для среза данных от 1 до 9 этажей

```
data.\nquery('0 < floors_total < 10 and 2.4 < ceiling_height < 4' )['ceiling_height'].\nhist(range=(2.4,4), bins=15)\nplt.xlabel('Высота потолка')\nplt.ylabel('Количество квартир в продаже')\nplt.title('Срез данных для домов от 1 до 9 этажей')
```

Срез данных для домов от 1 до 9 этажей



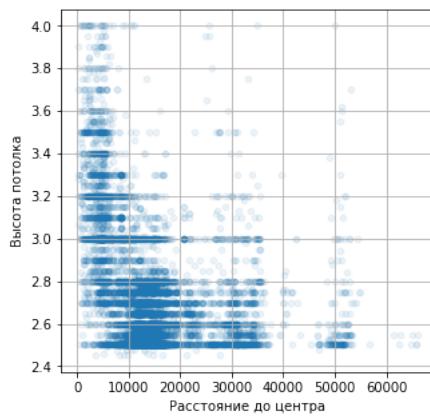
На распределении видны два пика. Большинство квартир с высотой потолков 2.5м. Но наблюдается всплеск на высоте 3м и длинный хвост до 4м. Что это за дома с высотой потолков в 3м?

Для домов высотой от 1 - 9 этажей медианное значение высоты потолка для пропусков значений явно не подходит и нужно поискать другой параметр, от чего зависит высота потолков.

Попробуем поискать зависимость высоты потолков от расстояния от центра города.

```
data.\nquery('2.4<ceiling_height<=4').\nplot(x='cityCenters_nearest', y='ceiling_height', \n    kind='scatter', grid=True, figsize = (5,5), alpha=0.08,\n    xlabel='Расстояние до центра', ylabel = 'Высота потолка')
```

Срез данных для домов от 1 до 9 этажей



Дома с самыми высокими потолками от 2.7 - 3.5 располагаются в промежутке 0-10км от центра, дома с высотой 2.5-2.8 чаще встречаются дальше от центра (более 10км от центра). Центр города всегда характеризовался более элитным, более уникальным по своим характеристикам жильем, чем периферия. На периферии строятся однотипные жилые массивы. Поэтому разобъем все данные на две группы:

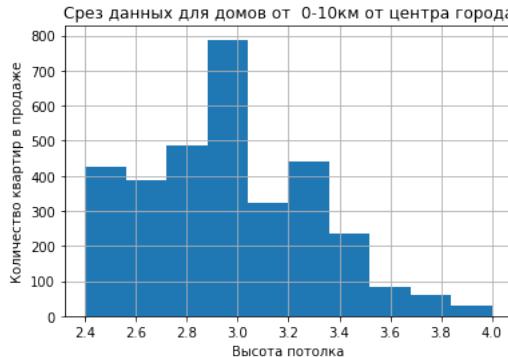
- дома на расстоянии 0-10км до центра СПБ
- дома дальше 10км до центра СПБ

Для двух разных групп посчитаем медиану и этой медианой заполним пропуски и на нее заменим аномальные значения.

Гистограмма высоты потолков для среза данных от 0-10км от центра города

```
data.\nquery('0 <= cityCenters_nearest <= 10000 and 2.4 < ceiling_height < 4' )['ceiling_height'].\nhist(range=(2.4,4), bins=10)\nplt.xlabel('Высота потолка')\nplt.ylabel('Количество квартир в продаже')\nplt.title('Срез данных для домов от 0-10км от центра города')
```

Text(0.5, 1.0, 'Срез данных для домов от 0-10км от центра города')



Высота потолков в центре тяготеет к 3м. Рассчитаем медиану для этого диапазона

```
print('Медианное значение высоты потолков для домов в центре города')
data.query('0 <= cityCenters_nearest <= 10000 and 2.4 < ceiling_height < 4')[['ceiling_height']].median()
```

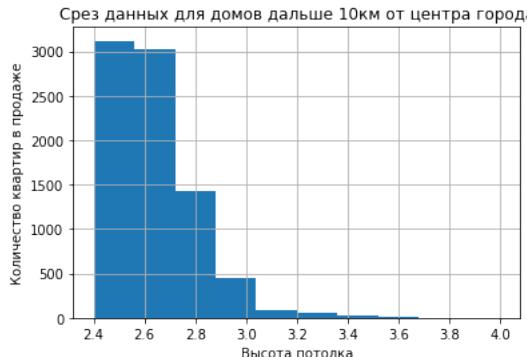
Text(0.5, 1.0, 'Медианное значение высоты потолков для домов в центре города')

3.0

Гистограмма высоты потолков для среза данных более 10км от центра города

```
data.\nquery('cityCenters_nearest > 10000 and 2.4 < ceiling_height < 4')[['ceiling_height']].\nhist(range=(2.4,4), bins=10)\nplt.xlabel('Высота потолка')\nplt.ylabel('Количество квартир в продаже')\nplt.title('Срез данных для домов дальше 10км от центра города')
```

Text(0.5, 1.0, 'Срез данных для домов дальше 10км от центра города')



```
print('Медианное значение высоты потолков для домов расположенных более 10км от центра города')
data.query('cityCenters_nearest > 10000 and 2.4 < ceiling_height < 4')[['ceiling_height']].median()
```

Text(0.5, 1.0, 'Медианное значение высоты потолков для домов расположенных более 10км от центра города')

2.6

Заполним высоту потолков

Медианное значение высоты потолков для домов до 10км от центра города: 3м

Медианное значение высоты потолков для домов расположенных более 10км от центра города: 2.6м

Высота потолков [1; 2.4]: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м

Высота потолков (2.4; 4]: оставляем

Высота потолков (4; 24]: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м

Высота потолков (24; 40]: исправили 2,4 - 4,0м

Высота потолков более 40м: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м

```
#создадим фильтр значений потолка (2.4;4], остальные значения будем считать аномальными
#будем заменять на медиану в зависимости от расстояния от центра
#заметим, что в фильтр ~filter_data войдут еще и пропуски в данных для потолка
filter_data = (data['ceiling_height']>2.4) & (data['ceiling_height']<=4)

#фильтр центра города (0;10000], близко к центру города
filter_center = (data['cityCenters_nearest']>0)&(data['cityCenters_nearest']<=10000)
#далеко от центра города, заметим, что в этот фильтр войдут и пропуски в данных расстояния
#следовательно, если для дома не указано расстояние до центра, то аномальное значение высоты потолка
#будет считаться по медиане для домов на периферии
filter_far = ~filter_center

#рассчитаем медианное значение высоты потолков близко к центру
median_ceil_center = data[(filter_center) & (filter_data)]['ceiling_height'].median()
print(median_ceil_center)

# рассчитаем медианное значение высоты потолков для домов далеко от центра
median_ceil_far = data[(filter_far)&(filter_data)]['ceiling_height'].median()
print(median_ceil_far)

#запишем в data_new['ceiling_height'] медианные значения высоты потолка
#характерные для центра города вместо аномальных значений высот потолка
data_new['ceiling_height'] = data_new['ceiling_height'].where(~((filter_center)&(~(filter_data))), median_ceil_center)

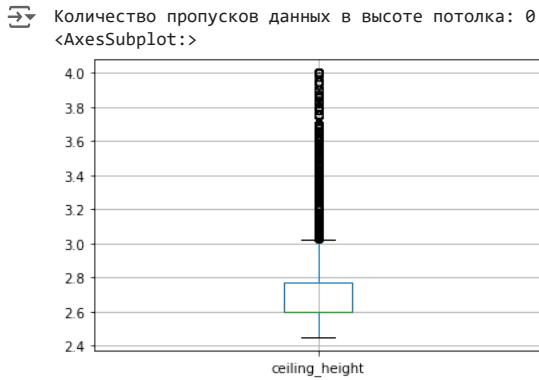
#характерные для периферии города вместо аномальных значений высот потолка
data_new['ceiling_height'] = data_new['ceiling_height'].where(~((filter_far)&(~(filter_data))), median_ceil_far)
```

3.0
2.6

Проверим, что аномальные значения и пустые все перезаписались в новый столбец new_ceiling_height

```
#посчитаем пропуски в данных
print('Количество пропусков данных в высоте потолка:', \
      data_new['ceiling_height'].isna().sum() )

#выведем "ящик с усами и посмотрим на выбросы"
data_new[['ceiling_height']].boxplot()
```



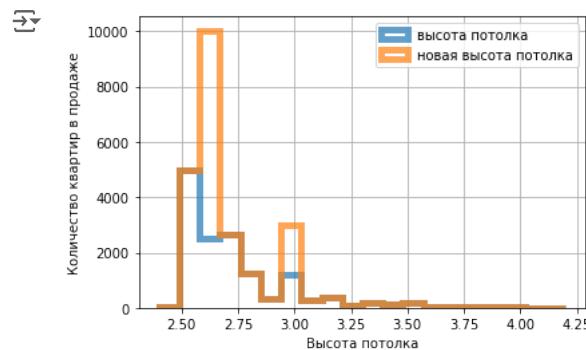
Данные уместились в желаемом диапазоне: высота потолков (2.4; 4]

Посмотрим вид гистограммы высоты потолка

```

ax = data.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(2.4, 4.2),
    bins=20,
    linewidth=5,
    alpha=0.7,
    label='высота потолка'
)
data_new.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(2.4, 4.2),
    bins=20,
    linewidth=5,
    alpha=0.7,
    label='новая высота потолка',
    ax=ax,
    grid=True,
    legend=True,
)
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Высота потолка')
plt.show()

```



Выведем основные числовые характеристики:

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

	count	mean	std	min	25%	50%	75%	max
Name:	ceiling_height	dtype: float64						
count	23699.000000							
mean	2.704788							
std	0.224681							
min	2.450000							
25%	2.600000							
50%	2.600000							
75%	2.770000							
max	4.000000							

Выход

Доля пропусков в значениях высоты потолков: 39%

Была произведена замена в данных с ошибкой в десятках: высота потолков (24; 40] заменена на (2.4; 4]

Было проведено исследование от какого параметра зависит высота потолков. Был произведен анализ зависимости от высоты потолков и от расстояния от центра. Были рассчитаны медианные значения высоты потолков в зависимости от расположения от центра

Дома с самыми высокими потолками от 2.7 - 3.5 располагаются в промежутке 0-10км от центра, дома с высотой 2,5-2,8 чаще встречаются дальше от центра (более 10км от центра). Центр города всегда характеризовался более элитным, более уникальным по своим характеристикам жильем, чем периферия. На периферии строятся однотипные жилые массивы.

Для двух разных групп рассчитаны медианные значения и ими заполнены пропуски: дома на расстоянии 0-10км до центра СПб дома дальше 10км до центра СПб

- Медианное значение высоты потолков для домов до 10км от центра города: 3м
- Медианное значение высоты потолков для домов расположенных более 10км от центра города: 2.6м
- Высота потолков [1; 2.4]: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м
- Высота потолков (2.4; 4]: оставляем
- Высота потолков (4; 24]: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м
- Высота потолков (24; 40]: исправили 2,4 - 4,0м

- Высота потолков более 40м: медианным значением в зависимости от расстояния от ц.г. 2.6м или 3м

"Хвосты" аномальных значений убраны и заменены медианными значениями и они дали новый пик в распределении высоты потолков в районе 2.6м Отчетливее стал виден пик в 3м для квартир в центре города.

✓ Комментарий ревьюера ✓

Проведена отличная работа! В целом, после устранения аномалий, связанных с ошибкой в десятках, мы могли заполнить пропуски медианой по столбцу - размах выборки небольшой, значения были корректны

✗ Количество этажей floors_total и пропуски, аномалии, тип данных

Для начала проверим данные об общем количестве этажей в домах

```
display(data['floors_total'].describe())
print('Количество пропусков в данных:', data['floors_total'].isna().sum() )
print(f'Доля пропусков: { data["floors_total"].isna().mean():.2%}' )
```

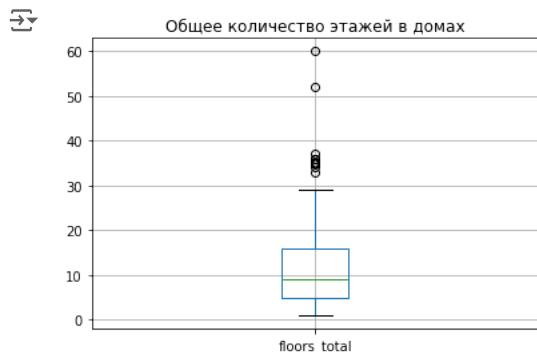
```
→ count    23613.000000
mean      10.673824
std       6.597173
min       1.000000
25%      5.000000
50%      9.000000
75%     16.000000
max      60.000000
Name: floors_total, dtype: float64
Количество пропусков в данных: 86
Доля пропусков: 0.36%
```

Какие сразу видны проблемы в данных?

- Есть пропуски, но их немного.
- Тип данных нужно изменить на целочисленный (очень сложно представить себе этаж 5 и 3/4)
- Количество этажей 60 выглядит неправдоподобным. Самое высокое жилое здание в СПБ имеет 35 этажей

Посмотрим на характер выбросов и самих данных

```
data[['floors_total']].boxplot()
plt.title('Общее количество этажей в домах')
plt.show()
```



В данных о количестве этажей присутствуют выбросы и аномальные значения. Основные значения сосредоточены в диапазоне от 5 до 16 этажей. Но есть выбросы аж до 60 этажей

Будем считать аномальным высоту дома более 35 этажей. В СПБ есть жилой дом в 35 этажей и он всего один.

Для пропущенных данных посмотрим данные из столбца - этаж на каком располагается продаваемая квартира. Значит в доме не меньше этажей, чем этаж, на котором продается квартира. Значит этими данными можно заполнить пропуск в этажности

```
data[data['floors_total'].isna()]['floor'].head(5)
```

```
→ 186      4
  237      1
  457     12
  671      8
  1757     9
Name: floor, dtype: int64
```

Количество этажей не может быть меньше этажа на котором располагается квартира. И следовательно не сильно ошибемся, если пропуски в этажности дома заполним этажем на котором располагается квартира.

Проверим нет ли аномалий в этаже квартиры

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

	count	mean	std	min	25%	50%	75%	max	Name: floor, dtype: float64
count	86.000000								
mean	10.023256								
std	6.210005								
min	1.000000								
25%	5.000000								
50%	8.000000								
75%	13.750000								
max	24.000000								

Для пропущенных данных этажностей домов максимальный этаж квартиры 24.

Проверим, нет ли пропусков в данных этажей

```
data[data['floors_total'].isna()]['floor'].isna().sum()
```

0

В срезе данных этажей квартир нет аномальных и пропусков. Можно этими данными заполнять пропуски этажности домов

```
data_new['floors_total'] = data_new['floors_total'].fillna(data['floor'])
print('Осталось пропусков:', data_new['floors_total'].isna().sum() )
```

Осталось пропусков: 0

Посмотрим, записались те значения, что мы и хотели записать

```
display (data_new.loc[data['floors_total'].isna(),['floors_total', 'floor']].head(5))
```

	floors_total	floor
186	4.0	4
237	1.0	1
457	12.0	12
671	8.0	8
1757	9.0	9

Изменим аномальные значения этажности домов.

Посчитаем, сколько квартир продается в 35 этажном доме:

```
print('Количество квартир в продаже в 35 этажном доме:', data[data['floors_total']==35]['floor'].count())
```

Количество квартир в продаже в 35 этажном доме: 24
--

Будем считать количество этажей более 35 неправдоподобным.

Посмотрим на каких этажах в этих домах продаются квартиры

```
data.query('floors_total > 35')[['floors_total','floor']]
```

	floors_total	floor
397	36.0	28
2253	60.0	4
5807	36.0	13
11079	36.0	29
16731	52.0	18
16934	37.0	5

Для домов количество этажей в которых больше 35 заменим количество этажей максимальным значением этажа на котором продается квартира, например 36 заменим на 29.

```
new_tmp = data_new.pivot_table(index='floors_total', values='floor', aggfunc='max')
new_tmp['floors_total'] = new_tmp.index
max_floor=0

for f in data_new.loc[data_new['floors_total']>35, 'floors_total'].unique():
    max_floor = new_tmp.loc[f, 'floor']
    data_new.loc[(data_new['floors_total'] == f), 'floors_total'] = max_floor
```

Проверим, те ли данные мы записали в новые значения этажности домов

```
filter_floors_total = data.query('floors_total > 35')
print('новые значения')
display(data_new.query('index in @filter_floors_total.index')[['floors_total','floor']])
print('старые значения')
display(data.query('floors_total > 35')[['floors_total','floor']])
```

→ новые значения

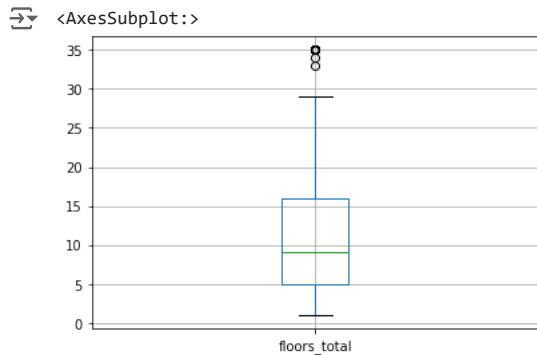
	floors_total	floor
397	29.0	28
2253	4.0	4
5807	29.0	13
11079	29.0	29
16731	18.0	18
16934	5.0	5

старые значения

	floors_total	floor
397	36.0	28
2253	60.0	4
5807	36.0	13
11079	36.0	29
16731	52.0	18
16934	37.0	5

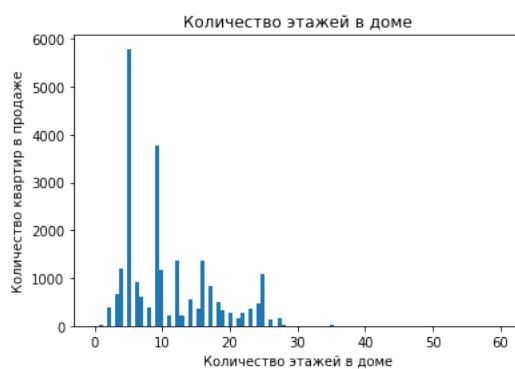
Снова построим ящик с усами.

```
data_new[['floors_total']].boxplot()
```



Выбросов после 35 этажей больше нет. Построим гистограмму количества этажей в домах

```
plt.hist(data_new['floors_total'], bins=100, range=(0,60))
plt.title('Количество этажей в доме')
plt.xlabel('Количество этажей в доме')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Теперь, когда пропусков в данных нет и аномальных значений нет, можно изменить тип данных для количества этажей на int

```
data_new['floors_total'] = data_new['floors_total'].astype(int)
data_new['floors_total'].head(5)
```

```
0    16
1    11
2     5
3    14
4    14
Name: floors_total, dtype: int64
```

Вывод

Доля пропусков в этажности домов 0.36% - незначительна

Тип данных изменен на целочисленный (очень сложно представить себе этаж 5 и 3/4)

Основные значения этажностей домов сосредоточены в диапазоне от 5 до 16 этажей. Но есть выбросы аж до 60 этажей. Известно, что самое высокое жилое здание в СПБ имеет 35 этажей. Принято решение считать аномальной высоту дома более 35 этажей.

Для аномальных данных этажности дома подобрана замена пропуска максимальным значением этажем из объявления о продаже квартиры в этом и аналогичном доме.

⚠️ Комментарий рецензента ⚠️

В данном случае, ввиду незначительной доли пропусков, мы могли удалить их без потери качества данных

✓ Названия населенных пунктов locality_name - пропуски и аномалии

```
print('Количество пропусков в данных:', data['locality_name'].isna().sum())
```

```
Количество пропусков в данных: 49
```

Всего то 49 населенных пунктов без названий. Запишем в них пустые строки

```
data_new['locality_name'] = data_new['locality_name'].fillna("")
```

⚠️ Комментарий рецензента ⚠️

NaN сам по себе является хорошим маркером - здесь мы также могли удалить пропуски без вреда для датасета

```
print('Количество пропусков в данных:', data_new['locality_name'].isna().sum())
```

```
Количество пропусков в данных: 0
```

```
print('Количество уникальных названий:', len(data['locality_name'].unique()))
```

```
Количество уникальных названий: 365
```

Если внимательно присмотреться к данным, то можно обнаружить что одни и теже названия типов населенных пунктов сформулированы по-разному и их следует заменить на единые названия

Заменим в названиях:

- приведем к нижнему регистру
- удалить пробелы в начале и в конце строки

- посёлок на поселок
- городской поселок на поселок
- поселок городского типа на поселок
- поселок при железнодорожной станции на поселок станции
- деревня кудрово заменить на кудрово

и снова посчитаем количество уникальных названий

```
data_new['locality_name'] = data_new['locality_name'].str.lower()
data_new['locality_name'] = data_new['locality_name'].str.strip()
data_new['locality_name'] = data_new['locality_name'].str.replace('посёлок','поселок')
data_new['locality_name'] = data_new['locality_name'].str.replace('городской поселок','поселок')
data_new['locality_name'] = data_new['locality_name'].str.replace('поселок городского типа','поселок')
data_new['locality_name'] = data_new['locality_name'].str.replace('поселок при железнодорожной станции','поселок станции')
data_new['locality_name'] = data_new['locality_name'].str.replace('ё','е')
data_new['locality_name'] = data_new['locality_name']\
    .str.replace('садоводческое некоммерческое товарищество','садовое товарищество')
data_new['locality_name'] = data_new['locality_name'].str.replace('деревня кудрово','кудрово')

print('Количество уникальных названий:',len(data_new['locality_name'].unique()))
print('Число переименованных названий', \
    len(data['locality_name'].unique())-len(data_new['locality_name'].unique()) )
```

Количество уникальных названий: 320
Число переименованных названий 45

Категоризируем данные по типу населенного пункта:

- 1 - города и без указания населенного пункта
- 2 - деревня
- 3 - поселок
- 4 - село
- 5 - товарищество

```
def typeLocality(name):
    if ('деревня' in name):
        return 2
    if ('поселок' in name):
        return 3
    if ('село' in name):
        return 4
    if ('товарищество' in name):
        return 5
    return 1
data_new['type_locality_name'] = data_new['locality_name'].apply(typeLocality)
display(data_new[['type_locality_name','locality_name','cityCenters_nearest']].head(5))
```

	type_locality_name	locality_name	cityCenters_nearest
0	1	санкт-петербург	16028.0
1	3	поселок шушары	18603.0
2	1	санкт-петербург	13933.0
3	1	санкт-петербург	6800.0
4	1	санкт-петербург	8098.0

Выводы

Количество пропусков в данных названий населенных пунктов: 49 Пропуски были заполнены пустыми строками Данные были проверены на неявные дубликаты и дубликаты были удалены Замены проводившиеся в данных:

приведем к нижнему регистру удалить пробелы в начале и в конце строки посёлок на поселок городской поселок на поселок поселок городского типа на поселок поселок при железнодорожной станции на поселок станции деревня кудрово заменить на кудрово

После удаления 45 дубликатов получено 320 уникальных названий Была проведена категоризация данных:

1 - города и без указания населенного пункта 2 - деревня 3 - поселок 4 - село 5 - товарищество

✓ Комментарий ревьюера ✓

Хорошая работа. Также рекомендую хорошую статью по предобработке, в ней есть интересные способы обработки неявных дубликатов: <https://proglib.io/p/moem-dataset-rukovodstvo-po-ochistke-dannih-v-python-2020-03-27> В пункте 8 содержится информация о

работе с опечатками в названиях населенных пунктов

✓ Расстояние до центра cityCenters_nearest - пропуски и аномалии

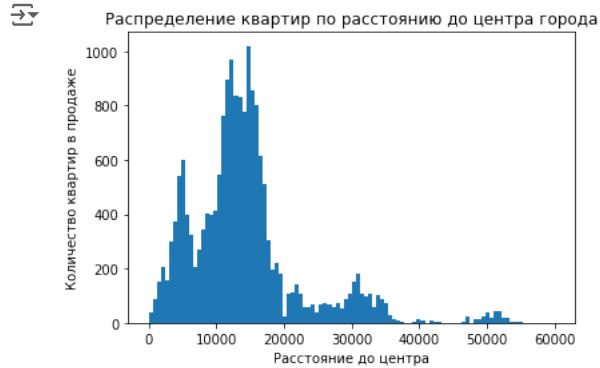
```
display(data['cityCenters_nearest'].describe())
print('Количество пропусков в данных:', data['cityCenters_nearest'].isna().sum())
print(f'Доля пропусков: { data["cityCenters_nearest"].isna().mean():.0%}')
```

	count	mean	std	min	25%	50%	75%	max
cityCenters_nearest	18180.000000	14191.277833	8608.386210	181.000000	9238.000000	13098.500000	16293.000000	65968.000000

Name: cityCenters_nearest, dtype: float64
Количество пропусков в данных: 5519
Доля пропусков: 23%

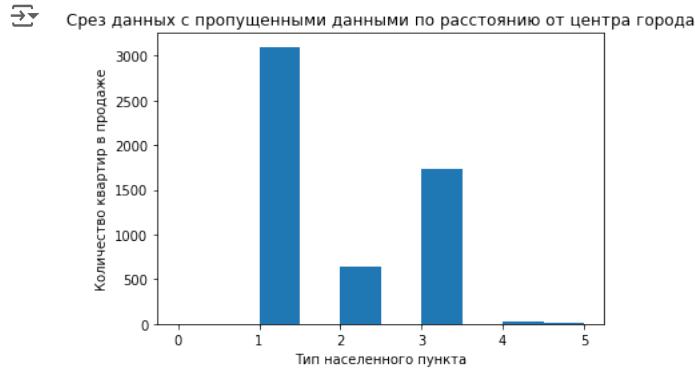
Посмотрим на гистограмму распределения квартир в продаже по расстоянию от центра

```
plt.hist( data['cityCenters_nearest'], range=(0,60000), bins=100)
plt.title('Распределение квартир по расстоянию до центра города')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Посмотрим для каких type_locality_name есть больше всего пропусков в cityCenters_nearest

```
plt.hist((data_new.loc[data_new['cityCenters_nearest'].isna(),'type_locality_name']),\
         bins=10, range=(0,5,))
plt.title('Срез данных с пропущенными данными по расстоянию от центра города')
plt.xlabel('Тип населенного пункта')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Построим гистограммы расстояний для срезов: для города и для всех остальных населенных пунктов

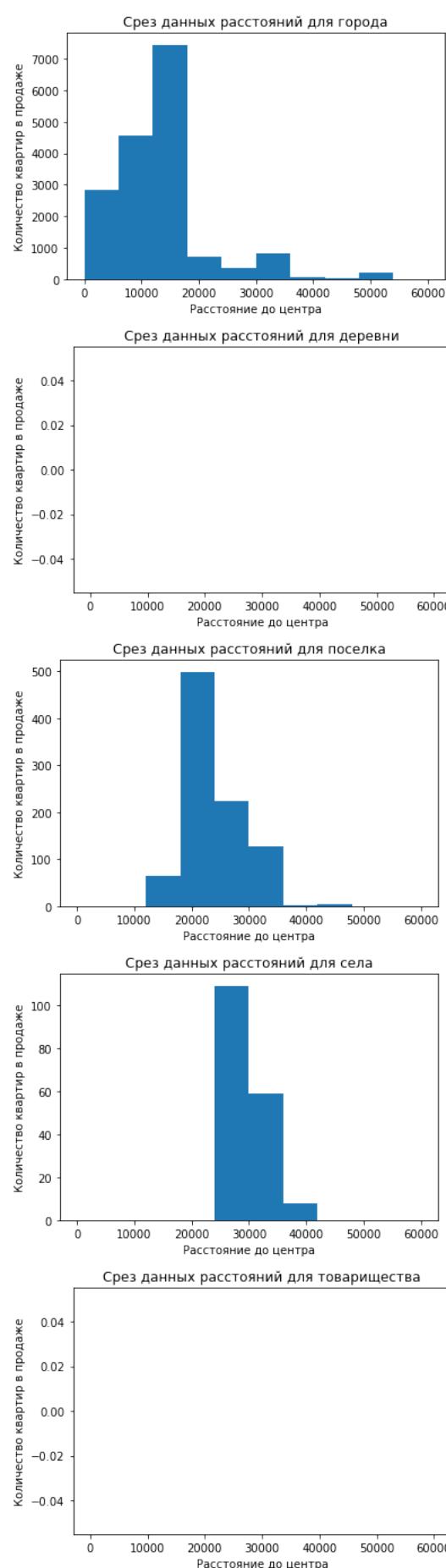
```
plt.hist( data_new.query('type_locality_name == 1')['cityCenters_nearest'] , range=(0,60000))
plt.title('Срез данных расстояний для города')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()

plt.hist( data_new.query('type_locality_name == 2')['cityCenters_nearest'] , range=(0,60000))
plt.title('Срез данных расстояний для деревни')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()

plt.hist( data_new.query('type_locality_name == 3')['cityCenters_nearest'] , range=(0,60000))
plt.title('Срез данных расстояний для поселка')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()

plt.hist( data_new.query('type_locality_name == 4')['cityCenters_nearest'] , range=(0,60000))
plt.title('Срез данных расстояний для села')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()

plt.hist( data_new.query('type_locality_name == 5')['cityCenters_nearest'] , range=(0,60000))
plt.title('Срез данных расстояний для товарищества')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Рассчитаем медианы и заполним медианными значениями пропуски. Пропуски для деревни заполним медианой для села, пропуски расстояний для товарищества заполним данными для поселка

- 1 - города и без указания населенного пункта
- 2 - деревня
- 3 - поселок

- 4 - село
- 5 - товарищество

```
median_city = data_new.query('type_locality_name == 1')['cityCenters_nearest'].median()
median_poselok = data_new.query('type_locality_name == 3')['cityCenters_nearest'].median()
median_selo = data_new.query('type_locality_name == 4')['cityCenters_nearest'].median()
print('Медиана расстояний для города:', median_city)
print('Медиана расстояний для поселка:', median_poselok)
print('Медиана расстояний для села:', median_selo)
```

→ Медиана расстояний для города: 12702.0
 Медиана расстояний для поселка: 21291.0
 Медиана расстояний для села: 29140.5

Заполним пропуски медианными значениями в зависимости от категории

```
data_new.loc[ (data_new['type_locality_name']==1)&(data_new['cityCenters_nearest'].isna()), ['cityCenters_nearest']] = \
    median_city
data_new.loc[ (data_new['type_locality_name']==2)&(data_new['cityCenters_nearest'].isna()), ['cityCenters_nearest']] = \
    median_selo
data_new.loc[ (data_new['type_locality_name']==3)&(data_new['cityCenters_nearest'].isna()), ['cityCenters_nearest']] = \
    median_poselok
data_new.loc[ (data_new['type_locality_name']==4)&(data_new['cityCenters_nearest'].isna()), ['cityCenters_nearest']] = \
    median_selo
data_new.loc[ (data_new['type_locality_name']==5)&(data_new['cityCenters_nearest'].isna()), ['cityCenters_nearest']] = \
    median_poselok

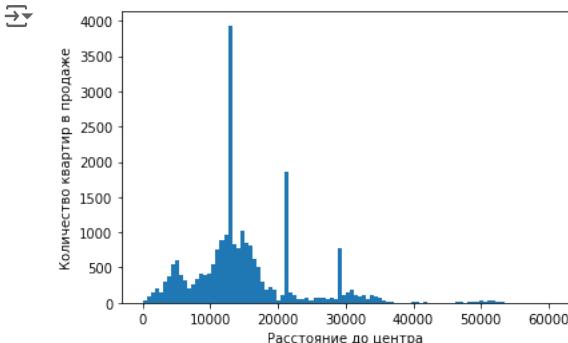
display(data_new['cityCenters_nearest'].describe())
print('Количество пропусков в данных:', data_new['cityCenters_nearest'].isna().sum())
print(f'Доля пропусков: { data_new["cityCenters_nearest"].isna().mean():.0% }')
```

→ count 23699.000000
 mean 14946.740749
 std 8165.730378
 min 181.000000
 25% 10927.000000
 50% 12863.000000
 75% 17417.000000
 max 65968.000000
 Name: cityCenters_nearest, dtype: float64
 Количество пропусков в данных: 0
 Доля пропусков: 0%

Посмотрим конечную гистограмму распределения

```
data_new['cityCenters_nearest'] = data_new['cityCenters_nearest'].astype(int)
```

```
plt.hist( data_new['cityCenters_nearest'], range=(0,60000), bins=100)
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Выходы

Доля пропусков: 23%

Больше всего пропусков в расстоянии для городов. Это связано с тем, что и объявлений для городов больше Пропуски для деревни заполнены медианой, расчетаной для типа населенного пункта село Пропуски расстояний для товарищества заполнены медианными значениями для поселка

На гистограмме числа квартир в продаже от расстояния от центра наблюдаются пики на 13, 21 и 29км от города. Это медианные значения:13 - для города, 21 - поселки, товарищества, 29 - села, деревни

❖ Апартаменты is_apartment пропуски, тип данных

Посмотрим, что за данные в столбце апартаменты

```
print('Количество пропусков в данных:', data['is_apartment'].isna().sum())
print(f'Доля пропусков: {data["is_apartment"].isna().mean():.0%}'')
```

Количество пропусков в данных: 20924
Доля пропусков: 88%

```
data['is_apartment'].unique()
array([nan, False, True], dtype=object)
```

```
data['is_apartment'].value_counts()
False    2725
True     50
Name: is_apartment, dtype: int64
```

Будем считать, что пропуски в данных обозначают, что объявление не относится к апартаментам и пропуски нужно заполнить значением False

```
data_new.loc[data_new['is_apartment'].isna(), 'is_apartment'] = False
data_new['is_apartment'].isna().sum()
```

0

Преобразуем тип к логическому

```
data_new['is_apartment'] = data_new['is_apartment'].astype('bool')
```

Вывод

Доля пропусков: 88% Будем считать, что пропуски в данных обозначают, что объявление не относится к апартаментам и пропуски нужно заполнить значением False

✓ Комментарий рецензента ✓

Согласен

❖ Балконы balcony пропуски, аномалии, тип данных

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

	count	mean	std	min	25%	50%	75%	max
count	12180.000000	1.150082	1.071300	0.000000	0.000000	1.000000	2.000000	5.000000
mean								
std								
min								
25%								
50%								
75%								
max								

Name: balcony, dtype: float64

```
print('Количество пропусков в данных:', data['balcony'].isna().sum())
print(f'Доля пропусков: {data["balcony"].isna().mean():.0%}'')
```

Количество пропусков в данных: 11519
Доля пропусков: 49%

Всего 25% квартир без балкона, слабо верится. Скорее всего все пропуски в данных - это и есть квартиры без балкона.

Пропуски скорее всего связаны с тем, что просто не указали, что балконов ноль.

Заменим все пропуски на ноль

```
data_new.loc[data['balcony'].isna(),'balcony'] = 0
data_new['balcony'].isna().sum()
```

0

```
data_new['balcony'].describe()
```

	count	mean	std	min	25%	50%	75%	max	Name:	dtype:
	23699.000000	0.591080	0.959298	0.000000	0.000000	0.000000	1.000000	5.000000	balcony	float64

Более 50 квартир без балкона. Да, скорее всего так и есть.

Приведем данные о балконах к целочисленному типу данных.

```
data_new['balcony']= data_new['balcony'].astype(int)
```

Вывод

Доля пропусков: 49% Пропуски заполнены нулями - это значит, что балконов нет

✓ Комментарий ревьюера ✓

Верно

▼ Парки parks_nearest parks_around3000

```
print('Количество пропусков в данных:', data['parks_nearest'].isna().sum())
print(f'Доля пропусков: { data["parks_nearest"].isna().mean():.0%}' )
```

Количество пропусков в данных: 15620
Доля пропусков: 66%

Комментарий И1

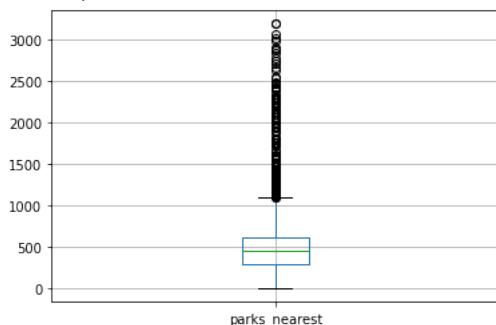
Выведу сразу долю пропусков для parks_around3000

```
print('Количество пропусков в данных:', data['parks_around3000'].isna().sum())
print(f'Доля пропусков: { data["parks_around3000"].isna().mean():.0%}' )
```

Количество пропусков в данных: 5518
Доля пропусков: 23%

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

<AxesSubplot:>



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

   count    8079.000000
   mean     490.804555
   std      342.317995
   min       1.000000
   25%    288.000000
   50%    455.000000
   75%    612.000000
   max    3190.000000
Name: parks_nearest, dtype: float64
```

Посмотрим пропуски в данных о расстоянии до ближайшего парка.

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

   array([ 0., nan])
```

Получается, что если расстояние пропущено, то и в данных о количестве парков рядом или ноль или пропуски.

Получается, пропущенные значения это не что иное, как отсутствие парков рядом. Следовательно их как-то нужно отметить

По основным числовым характеристикам в parks_nearest нет нулевых значений. Значит отсутствие парков рядом никак не помечается, а делаются пропуски в данных.

Заполним пропуски нулем, например 0 - это будет обозначать, что парков рядом нет. Запомним это.

Комментарий И1

Так как дополнительных картографических данных нет, а данные о парках в дальнейшем исследовании не участвуют, то оставим данные как есть, не будем удалять пропуски

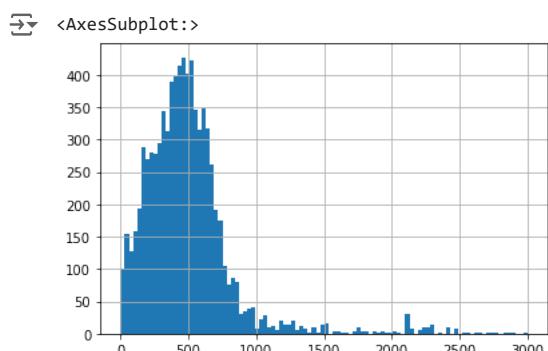
Удалим код

```
data_new.loc[ data_new['parks_nearest'].isna(), 'parks_nearest' ] = 0
data_new['parks_nearest'] = data_new['parks_nearest'].astype(int)
```

✓ Комментарий ревьюера v2 ✓

Верное решение

```
data_new['parks_nearest'].hist(bins=100, range=(1,3000))
```



Посмотрим числовые характеристики для количества парков рядом

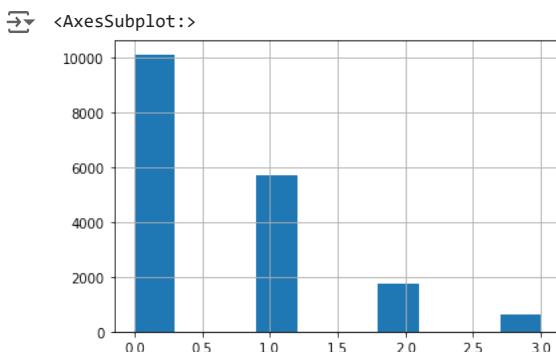
```
data[['parks_around3000']].describe()
```

parks_around3000	
count	18181.000000
mean	0.611408
std	0.802074
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	3.000000

Комментарий И1

Построим гистограмму для parks_around3000

```
data_new['parks_around3000'].hist()
```



Комментарий И1

Удалим следующий код и внесем изменения в вывод

```
data_new.loc[ data_new['parks_around3000'].isna(), 'parks_around3000' ] = 0
data_new['parks_around3000'].isna().sum()
data_new['parks_around3000'] = data_new['parks_around3000'].astype(int)
```

Вывод

- Доля пропусков в расстоянии до ближайшего парка: 66%
- Доля пропусков в количестве парков рядом: 23%

Оставим картографические данные, как есть с пропусками, тем более, что в дальнейшем исследовании они не участвуют

✖ Комментарий ревьюера ✖

Касательно этих и остальных картографических данных (ponds_nearest ponds_around3000 airports_nearest и т.п.) - при работе с ними нужно быть очень осторожным - размах выборки большой, поэтому заполнение пропусков приведет к тому, что данные окажутся неактуальными - к тому же, эти данные в дальнейшем анализе не участвуют - стоит оставить пропуски без изменений

✓ Комментарий ревьюера v2 ✓

Замечание исправлено

- ✓ Пруды ponds_nearest ponds_around3000 пропуски, типы данных

```
print('Количество пропусков в данных:', data['ponds_nearest'].isna().sum())
print(f'Доля пропусков: { data["ponds_nearest"].isna().mean():.0% }')
```

→ Количество пропусков в данных: 14589
Доля пропусков: 62%

Комментарий И1

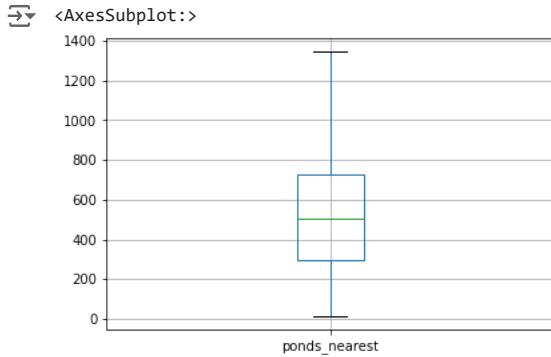
Выведу сразу долю пропусков для ponds_around3000

```
print('Количество пропусков в данных:', data['ponds_nearest'].isna().sum())
print(f'Доля пропусков: { data["ponds_nearest"].isna().mean():.0%}' )
```

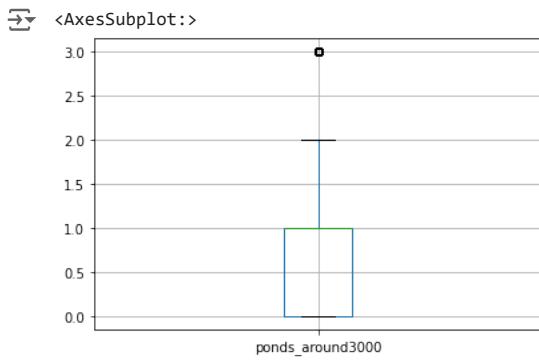
→ Количество пропусков в данных: 5518
Доля пропусков: 23%

По тому же принципу обработаем данные о количестве прудов

```
data[['ponds_nearest']].boxplot()
```



```
data[['ponds_around3000']].boxplot()
```



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

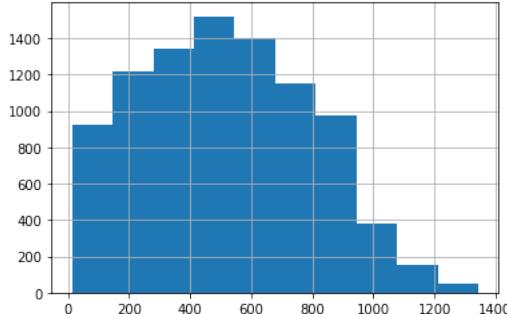
→ count 9110.000000
mean 517.980900
std 277.720643
min 13.000000
25% 294.000000
50% 502.000000
75% 729.000000
max 1344.000000
Name: ponds_nearest, dtype: float64

```
data['ponds_around3000'].describe()
```

→ count 18181.000000
mean 0.770255
std 0.938346
min 0.000000
25% 0.000000
50% 1.000000
75% 1.000000
max 3.000000
Name: ponds_around3000, dtype: float64

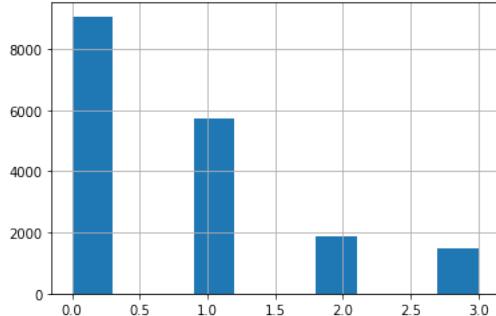
```
data['ponds_nearest'].hist()
```

<AxesSubplot:>



data['ponds_around3000'].hist()

<AxesSubplot:>



Комментарий И1

Так же как и в предыдущем блоке удалю обработку пропусков. Их количество слишком значительное, а в анализе данные не участвуют. Пропуски можно оставить как есть.

Удалю код ниже

```
data_new.loc[ data_new['ponds_around3000'].isna(), 'ponds_around3000' ] = 0
print('Количество пропусков в данных:', data_new['ponds_around3000'].isna().sum())
data_new['ponds_around3000'] = data_new['ponds_around3000'].astype(int)
data_new.loc[ data_new['ponds_nearest'].isna(), 'ponds_around3000' ].unique()
data_new.loc[ data_new['ponds_nearest'].isna(), 'ponds_nearest' ] = 0
data_new['ponds_nearest'] = data_new['ponds_nearest'].astype(int)
```

Выводы

- Доля пропусков в расстоянии до прудов: 62%
- Доля пропусков в количестве ближайших прудов: 23%

Данные оставлены без заполнения пропусков, т.к. данные картографические, количество пропусков значительное и не известно, какими данными их заполнять.

✓ Комментарий ревьюера v2 ✓

Верно. Теория обработки пропусков хорошо изложена в статье: <https://loginom.ru/blog/missing>. Особенno хорошо описаны виды пропусков и их влияние на целесообразность и выбор способа заполнения. Действительно, решая аналитические задачи, заполнение пропусков может быть необязательно, если объем достоверных данных достаточен для выявления закономерностей, а риск изменить распределение данных существенен. Не зря в задании просят: "Заполните пропущенные значения там, где это возможно". При ответе на вопросы исследования можно взять только достоверно известные значения, а пропуски проигнорировать, если надежного способа их заполнить нет.

❖ Дата и количество дней публикации first_day_exposition days_exposition пропуски, тип данных

Всего записей 23699

first_day_exposition - нет пропусков

days_exposition - 3181 пропусков

Посмотрим на характер данных

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

→ 0    2019-03-07T00:00:00
  1    2018-12-04T00:00:00
  2    2015-08-20T00:00:00
  3    2015-07-24T00:00:00
  4    2018-06-19T00:00:00
Name: first_day_exposition, dtype: object
```

Преобразуем тип данных, соответствующий дате

```
data_new['first_day_exposition'] = pd.to_datetime(data_new['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S')
display(data_new['first_day_exposition'].head())
```

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

```
data_new['first_day_exposition'].dtype
```

```
→ dtype('<M8[ns]')
```

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

```
→ 0      NaN
  1     81.0
  2    558.0
  3    424.0
  4    121.0
Name: days_exposition, dtype: float64
```

Проверим какой дате соответствуют пропуски в количестве дней публикации

```
print('Количество пропусков в данных:', data['days_exposition'].isna().sum())
print(f'Доля пропусков: {data["days_exposition"].isna().mean():.0%}')
```

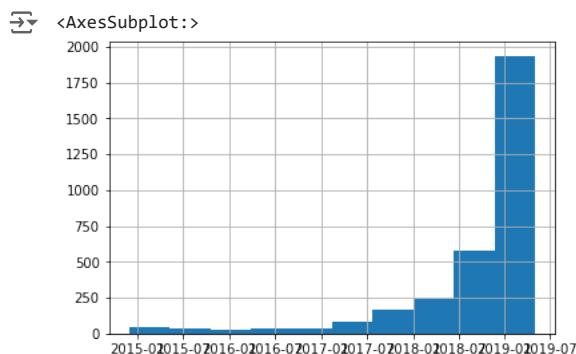
```
→ Количество пропусков в данных: 3181
Доля пропусков: 13%
```

```
min_date_exposition = data_new['first_day_exposition'].dt.date.min()
max_date_exposition = data_new['first_day_exposition'].dt.date.max()
print(min_date_exposition)
print(max_date_exposition)
```

```
→ 2014-11-27
2019-05-03
```

Вот и получили промежуток, за который есть данные о продаже квартир. С 2014 по 2019 года

```
data_new.loc[ data_new['days_exposition'].isna(), 'first_day_exposition'].hist()
```



Пропуски в количестве дней публикации увеличивается к 2019 году. Возможно это объявления, которые не сняты с публикации на момент 2019-05-03. Пропуски можно заполнить, посчитав разницу между датой публикации и 2019-05-03.

Есть квартиры, которые аномально долго не продаются. Объявления о них висят годами. То ли люди не спешат продать и цена на квартиры завышена, толи ошибки в системе, квартира давно продана, а на ресурсе данные о ней сохранились из-за какой-нибудь

системной ошибки и не могут удалиться.

Комментарий И1

Пропуски в данных не следует удалять. Они будут помечать, какие квартиры не продались на момент выгрузки данных

Удаляю код ниже

```
# создаем колонку с числом дней, которыми нужно заполнить данные
# они будут в формате timedelta, например "5 days"
days_data = max_date_exposition - data_new.loc[data['days_exposition'].isna(), 'first_day_exposition'].dt.date

# переведем формат timedelta в int
days_data[days_data.index] = days_data[days_data.index].dt.days
# заполним пропуски по индексам
data_new['days_exposition'] = data_new['days_exposition'].where(~data_new.index.isin(days_data.index), days_data[days_data.index])
# тип float преобразуем в int
data_new['days_exposition'] = data_new['days_exposition'].astype(int)
```

Комментарий И1

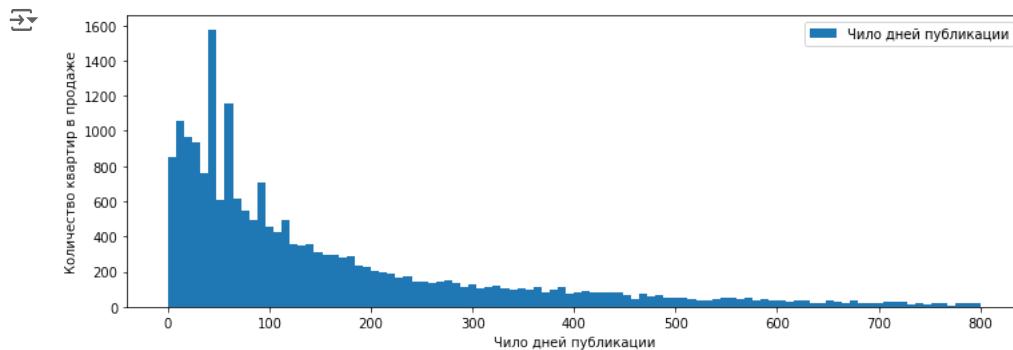
Хочу отдельно заметить, что невозможно преобразовать тип данных days_exposition, если в колонке есть пропуски и невозможно применить метод astype()

Ниже выведу гистограмму для числа дней публикаций без удаленных пропусков

✓ Комментарий ревьюера v2 ✓

Да, не стоит жертвовать качеством данных ради приведения к другому типу. Но мы можем попробовать использовать, например, Int64 (именно с большой буквы) - он допускает наличие NaN: https://pandas.pydata.org/pandas-docs/stable/user_guide/integer_na.html

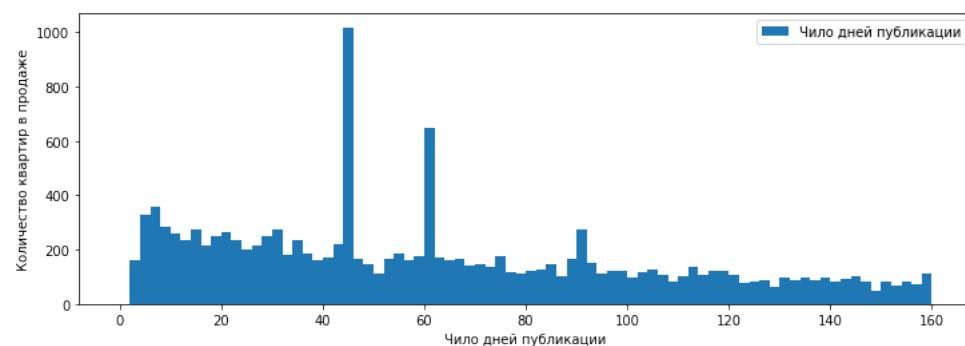
```
data.plot(
    kind='hist',
    y='days_exposition',
    bins=100,
    range = (0, 800),
    label='Число дней публикации',
    figsize=(12,4)
)
plt.xlabel('Число дней публикации')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Комментарий И1

Рассмотрим данные за полгода поближе

```
data.plot(
    kind='hist',
    y='days_exposition',
    bins=80,
    range = (0, 160),
    label='Число дней публикации',
    figsize=(12,4)
)
plt.xlabel('Число дней публикации')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Комментарий И1

Видим пики на 45 дне, на 60 дне и на 90 дней публикации. Это связано с условиями размещения объявлений на площадке Яндекс.Недвижимость. Срок размещения кратен 45 дней, 60 дней, 90 дней. В эти дни происходит обновление и если срок размещения квартиры закончился, объявление заново выставляется в продажу.

Получается в эти дни происходит задвоение данных о квартирах и для этих квартир нужно увеличить срок размещения в два раза. Таких случаев всего 7%. Может их просто удалить

Следовательно эти нужно удалить дубликаты объявлений на 45, 60 и 90 днях

⚠ Комментарий ревьюера v2 ⚡

Нам не обязательно удалять эти данные - на этапе оценки скорости продажи мы можем использовать диаграмму размаха

```
data_days = data.copy()
data_days = data_days.dropna(subset=['days_exposition'])
data_days['days_exposition'] = data_days['days_exposition'].astype(int)

#данные квартир со сроком публикации 45 60 90 дней
data_days_45_60_90 = data_days.loc[(data_days['days_exposition']==90) | 
                                    (data_days['days_exposition']==60) | 
                                    (data_days['days_exposition']==45) ]

#фильтр находит данные квартир, которые дублируются
#дубликатами считаем квартиры со сроком размещения 45,60,90 дней
#и расположенные в одном и том же месте и на одинаковых этажах
# другие данные в объявлении могли поменяться
# такие как цена, может быть уточнена площадь
# в фильтр могут попасть и лишние квартиры, но очень незначительная доля
filter_dupl = data_days_45_60_90[['cityCenters_nearest','floor']].duplicated()
data_new = data_new.query('index not in @filter_dupl.index')

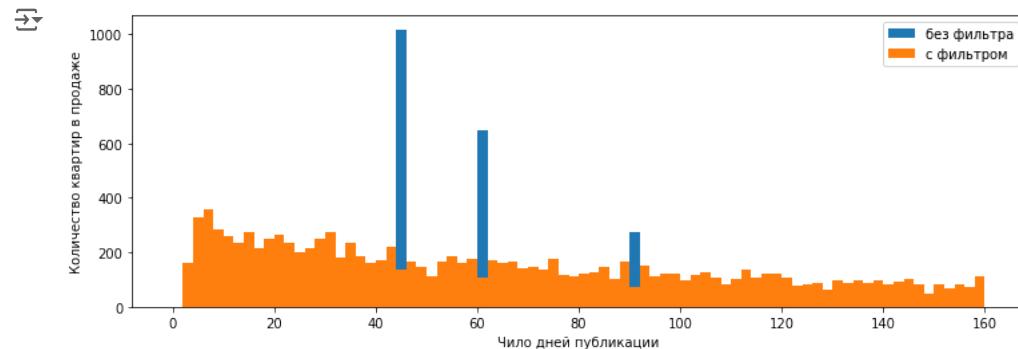
print('Всего объявлений:',len( data))
print('Количество квартир в продаже на момент выгрузки данных', data['days_exposition'].isna().sum())
print('Всего проданных квартир:', len(data_days))
print('Всего дублей:',len(filter_dupl))
print('Всего удалено дублей:', len(filter_dupl)/ len(data))
print('Осталось квартир после удаления дублей:', len(data_new))
```

→ Всего объявлений: 23699
 Количество квартир в продаже на момент выгрузки данных 3181
 Всего проданных квартир: 20518
 Всего дублей: 1622
 Всего удалено дублей: 0.06844170640111397
 Осталось квартир после удаления дублей: 22077

```

ax=data_days.plot(
    kind='hist',
    y='days_exposition',
    bins=80,
    range = (0, 160),
    label='без фильтра',
    figsize=(12,4)
)
data_new.plot(
    kind='hist',
    y='days_exposition',
    bins=80,
    range = (0, 160),
    ax=ax,
    label='с фильтром',
)
plt.xlabel('Число дней публикации')
plt.ylabel('Количество квартир в продаже')
plt.show()

```



Комментарий И1

Всего 7% данных с выбросами были удалены из засмотрения

Осталось квартир после удаления дублей: 22077

Выводы

Доля пропусков данных в количестве дней публикации 13%. В дате публикации пропусков в данных нет. Тип данных был преобразован в соответствующий типу даты format='%Y-%m-%dT%H:%M:%S'. Получен промежуток, за который есть данные о продаже квартир. С 2014 по 2019 года (с 2014-11-27 по 2019-05-03)

Установлено, что пропуски в количестве дней публикации увеличивается к 2019 году. Возможно это объявления, которые не сняты с публикации на момент 2019-05-03. Пропуски можно заполнить, посчитав разницу между датой публикации и 2019-05-03.

Есть квартиры, которые аномально долго не продаются. Объявления о них висят годами. То ли люди не спешат продать и цена на квартиры завышена, то ли ошибки в системе, квартира давно продана, а на ресурсе данные о ней сохранились из-за какой-нибудь системной ошибки и не могут удалиться.

✖ Комментарий рецензента ✖

Если рассматривать распределение пропусков относительно года выставления на продажу, можно заметить, что большая часть пропущенных значений - в 2019 году. Больше половины объектов, выставленных на продажу в 2019 году не имеет рассчитанного "времени продажи квартиры". Также высокая доля пропусков, хотя и меньшая - в 2018 году. Получается, чем свежее объявление - тем выше вероятность пропуска в этой колонке.

Пропуск в этой колонке означает, что объект не продан и объявление не закрыто. Поэтому заполнять пропуски в этом столбце не надо - они не случайные и существенно влияют на статистику. Мы можем заполнять только те данные, которые пропущены совершенно случайно. К тому же, у нас нет гарантий, что на момент выгрузки данных эти квартиры продались

Комментарий И1

В работе как раз и было проведено исследование как пропуски распределяются по годам. И сделан правильный вывод, что пропуск обозначает, что квартира до сих пор в продаже.

Согласна, что пропуски не следовало заполнять, разницей: "день выгрузки данных - день выставления в продажу". Чтобы отметить, что квартира все еще в продаже находится пропуски следует оставить. Ведь нет гарантии, что квартира продалась в день выгрузки данных.

Новые выводы

- Всего объявлений: 23699
- Количество квартир все еще в продаже на момент выгрузки данных 3181

- Всего проданных квартир: 20518

В объявлениях обнаружены дубли на сроке размещения объявления 45 60 и 90 дней. Это связано с условиями размещения объявлений на площадке Яндекс.Недвижимость. Срок размещения кратен 45 дней, 60 дней, 90 дней. В эти дни происходит обновление и если срок размещения квартиры закончился, объявление заново выставляется в продажу.

Получается в эти дни происходит задвоение данных о квартирах. Таких случаев всего 7%. Такие дублирующиеся данные были удалены из рассмотрения.

- Всего дублей: 1662
- Всего удалено дублей: 7%
- Осталось объявлений о продаже квартир после удаления дублей: 22077

▼ Аэропорт airports_nearest пропуски тип данных

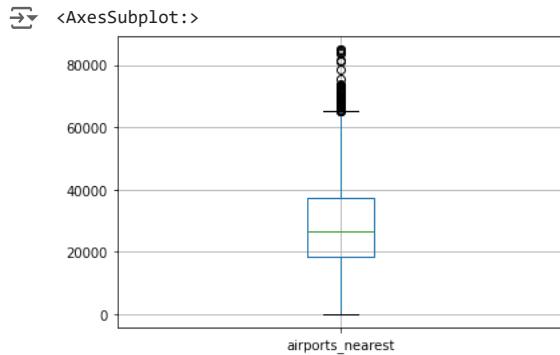
```
print('Количество пропусков в данных:', data['airports_nearest'].isna().sum())
print(f'Доля пропусков: { data["airports_nearest"].isna().mean():.0%}')
```

→ Количество пропусков в данных: 5542
Доля пропусков: 23%

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

→ count 18157.000000
mean 28793.672193
std 12630.880622
min 0.000000
25% 18585.000000
50% 26726.000000
75% 37273.000000
max 84869.000000
Name: airports_nearest, dtype: float64

```
data[['airports_nearest']].boxplot()
```



Комментарий И1

Данные с расстоянием для аэропорта не участвуют в анализе. Так же это картографические данные. Не будем их трогать.

Возможно пропуски в расстоянии до аэропорта связаны с отсутствием вблизи аэропортов.

Другая причина пропусков, это не владение продовцом квартиры данными о расстоянии до ближайшего аэропорта

Посчитаем количество квартир вблизи 6км от аэропорта

```
data.loc[ data['airports_nearest'] < 6000 , 'airports_nearest'].count()
```

→ 1

Всего одно значение и оно для расстояния в 0. Можно посмотреть данные этой квартиры.

```
data.query('airports_nearest == 0')
```

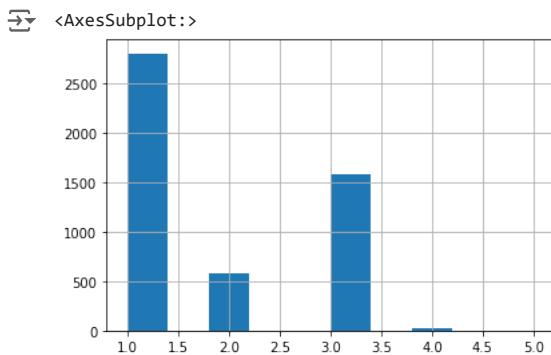
	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apart
21085	0	7000000.0	34.7	2018-09-23T00:00:00	1	2.7	9.0	19.8	3	

1 rows × 22 columns

Ничего особенного в данных этой квартиры нет. Расстояние от центра СПБ 22 км. Расстояние аэропорта Пулково от СПБ как раз 21 км. Возможно есть жилой массив рядом с аэропортом и поэтому расстояние 0

Посмотрим для каких населенных пунктов отсутствуют данные о расстоянии до ближайшего аэропорта.

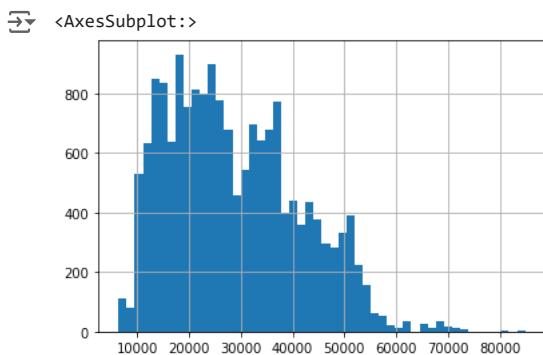
```
data_new.loc[ data_new['airports_nearest'].isna(), 'type_locality_name'].hist()
```



Больше всего данных пропущено для квартир в городе и для поселков, значит пропуски можно заполнить медианным значением расстояния до аэропорта рассчитанным для квартир в городе

Гистограмма распределения удаленности от аэропорта

```
data_new[ 'airports_nearest'].hist(bins=50)
```



Комментарий И1

Выводы

Доля пропусков в данных : 23%

Возможно пропуски в расстоянии до аэропорта связаны с отсутствием вблизи аэропортов. Другая причина пропусков, это не владение продовцом квартиры данными о расстоянии до ближайшего аэропорта

Т.к. расстояние до аэропорта - это картографические данные и данные не участвуют в анализе, пропуски не будем заполнять.

Есть квартира расстояние до аэропорта Ноль. Анализ данных этой квартиры не выявил никаких особенностей. Расстояние от центра СПБ 22 км. Расстояние аэропорта Пулково от СПБ как раз 21 км. Возможно есть жилой массив рядом с аэропортом и поэтому расстояние 0

Больше всего данных пропущено для квартир в городе и для поселков.

✓ Комментарий ревьюера v2 ✓

Хорошо

❖ Площадь: общая, жилая и площадь кухни living_area kitchen_area total_area

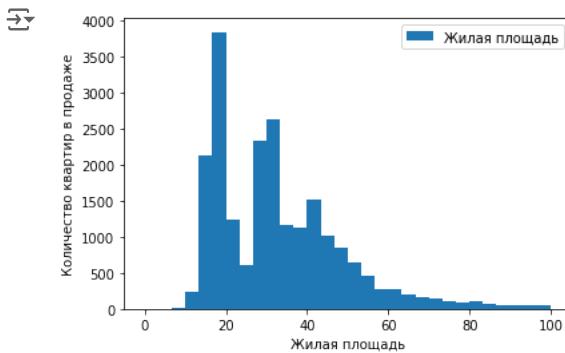
```
print('Количество пропусков в данных жилой площади:', data['living_area'].isna().sum())
print(f'Доля пропусков жилой площади: { data["living_area"].isna().mean():.0%}')

print('Количество пропусков в данных площади кухни:', data['kitchen_area'].isna().sum())
print(f'Доля пропусков в площади кухни: { data["kitchen_area"].isna().mean():.0%}')

print('Количество пропусков в данных общая площадь:', data['total_area'].isna().sum())
print(f'Доля пропусков в общей площади: { data["total_area"].isna().mean():.0%}')
```

→ Количество пропусков в данных жилой площади: 1903
Доля пропусков жилой площади: 8%
Количество пропусков в данных площади кухни: 2278
Доля пропусков в площади кухни: 10%
Количество пропусков в данных общая площадь: 0
Доля пропусков в общей площади: 0%

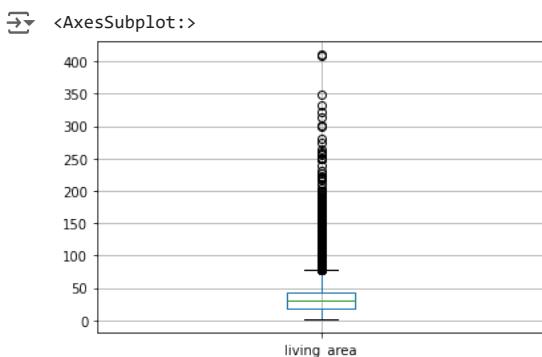
```
data.plot(
    kind='hist',
    y='living_area',
    bins=30,
    range=(0, 100),
    label='Жилая площадь'
)
plt.xlabel('Жилая площадь')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



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

→ 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

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



Комментарий И1

Посчитаем долю данных для жилой площади более 100 кв.м

```
print('Число квартир с аномально большой жилой площадью', \
      data.loc[data['living_area'] >= 100, 'living_area'].count())
data.loc[data['living_area'] >= 100, 'living_area'].count() / len(data)
```

→ Число квартир с аномально большой жилой площадью 363
0.01531710198742563

Комментарий И1

1,5% данных можно удалить с жилой площадью более 100 кв.м

```
data_new = data_new.loc[~(data['living_area'] >= 100)]
```

```
len(data_new)
```

→ 21725

Комментарий И1

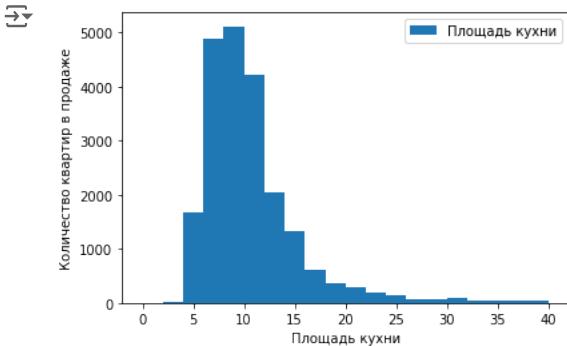
21725 записи осталось

Жилая площадь 2кв.м. Сомнительно, но таких данных немного. Так же можно предположить, что существуют квартиры с жилой площадью 409кв.м. 25% данных- 18.6кв.м. - как раз размер одной жилой комнаты. Данные похожи на правду.

Есть провал в данных жилой площади в районе 22-23кв.м.

На гистограмме видны три пика, как раз характеризующие однокомнатные, двухкомнатные и трехкомнатные квартиры.

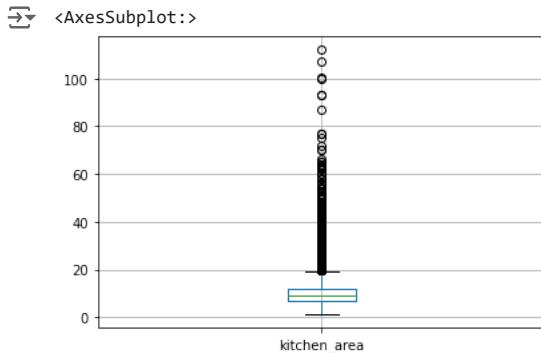
```
data.plot(
    kind='hist',
    y='kitchen_area',
    bins=20,
    range=(0, 40),
    label='Площадь кухни'
)
plt.xlabel('Площадь кухни')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



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

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

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



```
data.loc[ data['kitchen_area']>=35 , 'kitchen_area'].count()/len(data)
```

0.010169205451706824

Комментарий И1

Расстанемся с 1% данных без сожаления Удалим все данные с площадью кухни более 35м

```
data_new = data_new.loc[~((data_new['kitchen_area']>=35))]
len(data_new)
```

21539

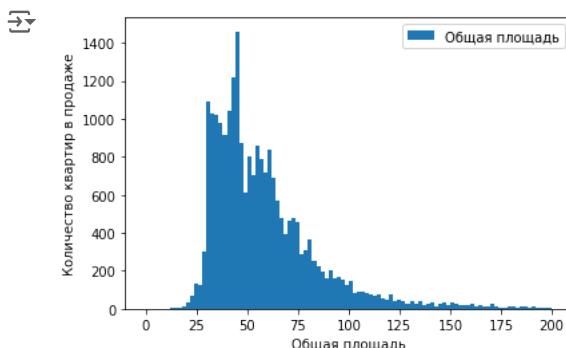
Комментарий И1

21539 записей осталось

Кухня 1.3кв.м.? Сомнительно, но опять же таких данных немного. 25% -кухонь с 7кв.м. - похоже на правду. И кому-то досталась широкая кухня 112 кв.м

Гистограмма более гладкая, соответственно на ней наблюдается один пик, т.к. в квартирах одна только кухня.

```
data.plot(
    kind='hist',
    y='total_area',
    bins=100,
    range=(0, 200),
    label='Общая площадь'
)
plt.xlabel('Общая площадь')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



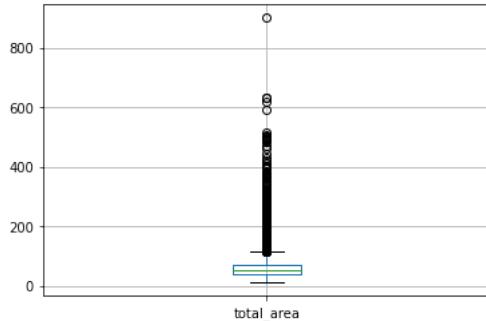
```
data['total_area'].describe()
```

count	23699.000000
mean	60.348651
std	35.654083
min	12.000000
25%	40.000000
50%	52.000000
75%	69.900000
max	900.000000

Name: total_area, dtype: float64

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

↳ <AxesSubplot:>



```
data.loc[ data['total_area']>=200, 'total_area'].count()/len(data)
```

↳ 0.009747246719270856

Комментарий И1

Расстанемся с 1% данных общей площади без сожаления

```
data_new = data_new.loc[~(data['total_area']>=200)]
print(len(data_new))
print('Доля удаленных данных площадей+дубликатов:', (len(data)-len(data_new))/len(data))
data_new[['kitchen_area', 'total_area', 'living_area']].describe()
```

↳ 21506

Доля удаленных данных площадей+дубликатов: 0.09253555002320774

	kitchen_area	total_area	living_area
count	19836.000000	21506.000000	20227.000000
mean	10.032437	56.964076	32.496139
std	4.230037	24.180389	15.643663
min	1.300000	12.000000	2.000000
25%	7.000000	40.000000	18.500000
50%	9.000000	51.000000	30.000000
75%	11.600000	68.000000	41.900000
max	34.700000	199.200000	99.900000

Комментарий И1

После удаления всех аномально больших площадей

- жилых помещений площадью более 100кв.м
- кухонь площадью более 35кв.м
- общих площадей более 200кв.м.

Осталось записей: 21507

- Доля удаленной информации о площадях около 1,5%
- Доля всей удаленной информации 9%

12кв.м. - общая площадь квартиры. Опять же сомнительно. 25% - общая площадь 40кв.м. - соответствует однокомнатной квартире. Каких-то сильно выдающихся пиков или провалов нет в гистограмме. Есть небольшие пики, соответствующие одно-, дву- трехкомнатным квартирам, но они более сглаженные, т.к. в общей площади добавляются площади нежелых помещений и накладываются на гистограмму жилой площади.

Пропусков в жилой площади и кухни всего 8% и 10%.

- Пропуски в площади кухни можно рассчитать: пл.кухни = общ.пл.- жил.пл
- Пропуски в жилой площаади: жил.пл. = общ.пл. - пл.кухни
- после этих операций посчитать долю пропусков

✖ Комментарий ревьюера ✖

Это неточный метод, так как помимо площади кухни и жилой площаади, общая включает также санузлы, кладовые и т.п. Мы можем найти соотношение медианной общей площаади к медианной жилой. А затем, заполнить пропуски в столбце living_area отношением общей площаади к найденному соотношению.

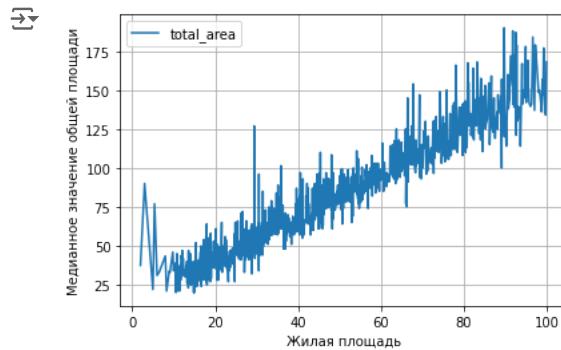
Со столбцом kitchen_area можно поступить так же, как со столбцом living_area.

Комментарий И1

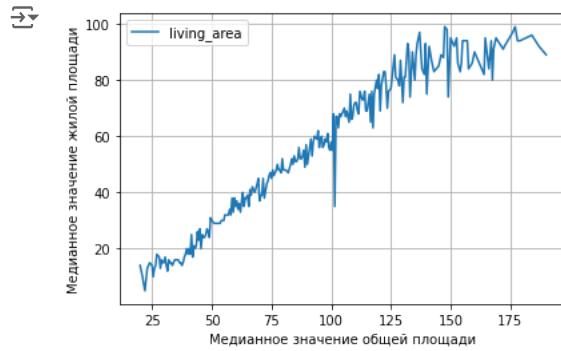
Согласна, что метод не точный, а приближенный.

Попробую применить рекомендованный и посмотрим, что получается

```
#сразу преобразуем тип общей площади в int, раз пропусков в данных нет
data_new['total_area'] = data_new['total_area'].astype(int)
#найдем для каждой жилой площади медиану общей площади и округлим значения
median_total = data_new.pivot_table(index='living_area', values='total_area', aggfunc='median')
median_total['living_area']=median_total.index
median_total.plot(
    x='living_area',
    y='total_area',
    grid=True)
plt.ylabel('Медианное значение общей площади')
plt.xlabel('Жилая площадь')
plt.show()
```



```
#для каждой медианы общей площади найдем медиану жилой площади и округлим значения
#назовем это отношение медианной общей с медианной жилой
med_tot_liv=median_total.pivot_table(index='total_area', values='living_area', aggfunc='median')
med_tot_liv['living_area'] = med_tot_liv['living_area'].astype(int)
med_tot_liv.plot(
    grid=True)
plt.xlabel('Медианное значение общей площади')
plt.ylabel('Медианное значение жилой площади')
plt.show()
med_tot_liv['total_area'] = med_tot_liv.index
```



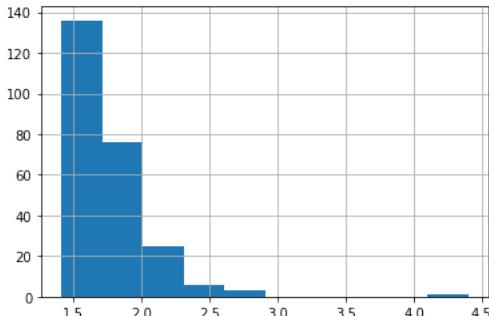
Комментарий И1

Посчитаем отношение медианных значений общей площади к медианным значениям жилой площади

Построим гистограмму и возьмем медианное значение отношения

```
delta = med_tot_liv['total_area'] / med_tot_liv['living_area']
delta.hist()
delta.median()
```

1.6891891891891893



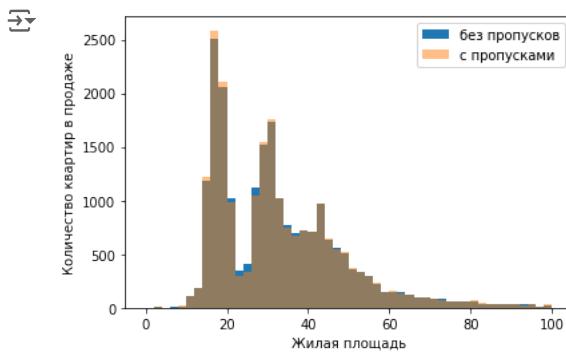
Комментарий И1

Заполним жилую площадь

```
data_nan= data_new.loc[data_new['living_area'].isna()].copy()
data_nan['living_area']=data_nan['total_area']/delta.median()
data_nan['living_area'] = data_nan['living_area'].astype(int)
data_new['living_area'] = data_new['living_area'].\
where(~data_new.index.isin(data_nan.index), data_nan['living_area'])
```

Сравним полученные результаты

```
ax = data_new.plot(
    kind='hist',
    y='living_area',
    bins=50,
    range=(0, 100),
    label='без пропусков'
)
data.plot(
    kind='hist',
    y='living_area',
    ax=ax,
    alpha=0.5,
    bins=50,
    range=(0, 100),
    label='с пропусками'
)
plt.xlabel('Жилая площадь')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Комментарий И1

Повторим все для кухни

```
#найдем для каждой площади кухни медиану общей площади и округлим значения
median_total_k = data_new.pivot_table(index='kitchen_area',values='total_area', aggfunc='median')
median_total_k['kitchen_area']=median_total_k.index
#для каждой медианы общей площади найдем медиану площади кухни и округлим значения
#назовем это отношение медианной общей с медианной кухни
med_tot_k=median_total_k.pivot_table(index='total_area', values='kitchen_area', aggfunc='median')
med_tot_k['kitchen_area'] = med_tot_k['kitchen_area'].astype(int)
med_tot_k['total_area'] = med_tot_k.index
```

```
#соотношение медианы общей площади к медиане площади кухни
delta_k = med_tot_k['total_area'] / med_tot_k['kitchen_area']
delta_k.median()

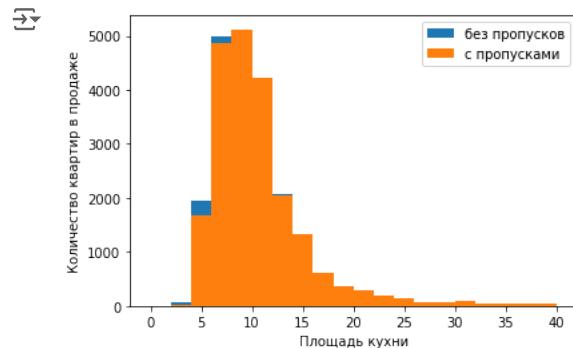
→ 5.333333333333333
```

#заполним пропуски площади кухни

```
data_nan_k= data_new.loc[data_new['kitchen_area'].isna()].copy()
data_nan_k['kitchen_area']=data_nan_k['total_area']/delta_k.median()
data_nan_k['kitchen_area'] = data_nan_k['kitchen_area'].astype(int)
data_new['kitchen_area'] = data_new['kitchen_area'].\\
where(~data_new.index.isin(data_nan_k.index), data_nan_k['kitchen_area'])
data_new['kitchen_area'].isna().sum()
```

→ 0

```
ax=data_new.plot(
    kind='hist',
    y='kitchen_area',
    bins=20,
    range=(0, 40),
    label='без пропусков'
)
data.plot(
    kind='hist',
    y='kitchen_area',
    bins=20,
    ax=ax,
    alpha=1,
    range=(0, 40),
    label='с пропусками'
)
plt.xlabel('Площадь кухни')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



```
data_new[['kitchen_area','total_area','living_area']].describe()
```

	kitchen_area	total_area	living_area
count	21506.000000	21506.000000	21506.000000
mean	9.979886	56.732261	32.650024
std	4.332646	24.188784	15.782640
min	1.300000	12.000000	2.000000
25%	7.000000	40.000000	18.700000
50%	9.000000	51.000000	30.000000
75%	11.600000	68.000000	41.900000
max	37.000000	199.000000	117.000000

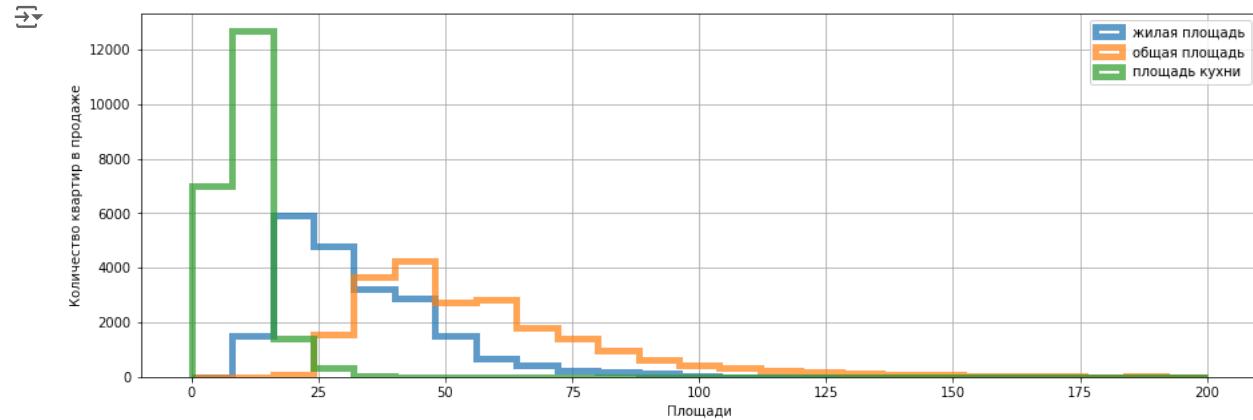
Данные площадей можно округлить до целых. Изменим их тип данных

```
data_new['total_area'] = data_new['total_area'].astype(int)
data_new['living_area'] = data_new['living_area'].astype(int)
data_new['kitchen_area'] = data_new['kitchen_area'].astype(int)
```

```

ax = data_new.plot(
    kind='hist',
    y='living_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='жилая площадь'
)
data_new.plot(
    kind='hist',
    y='total_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='общая площадь',
    ax=ax,
    grid=True,
    legend=True
)
data_new.plot(
    kind='hist',
    y='kitchen_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='площадь кухни',
    ax=ax,
    grid=True,
    legend=True,
    figsize=(15,5)
)
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Площади')
plt.show()

```



Комментарий И1 Вывод

На гистограмме жилой площади видны три пика, как раз характеризующие однокомнатные, двухкомнатные и трехкомнатные квартиры. Гистограмма площади кухни более гладкая, соответственно на ней наблюдается один пик, т.к. в квартирах одна только кухня.

В данных площадей есть аномально малые значения, и аномально большие. После удаления всех аномально больших площадей

- жилых помещений площадью более 100кв.м
- кухонь площадью более 35кв.м
- общих площадей более 200кв.м.
- Осталось записей : 21507
- Удалено: 570 записей
- Доля удаленной информации по площадям около 2,4%

Каких-то сильно выдающихся пиков или провалов нет в гистограмме общей площади нет. Есть небольшие пики, соответствующие одно-, дву- трех- комнатным квартирам, но они более сглаженные, т.к. в общей площади добавляются площади нежелых помещений и накладываются на гистограмму жилой площади.

Пропусков в жилой площади и кухни всего 8% и 10%.

Для устранения пропусков в жилой площиди было рассчитано соотношение медианного значения общей площаи к медианному значению жилой площаи и пропуски заполнены как пл.жил = общ.пл./соотношение

Тоже самое было проделано и для пропусков для кухни

✓ Комментарий реевьюера v2 ✓

Получилось отлично! 👍

✗ Проверка аномалий в данных last_price

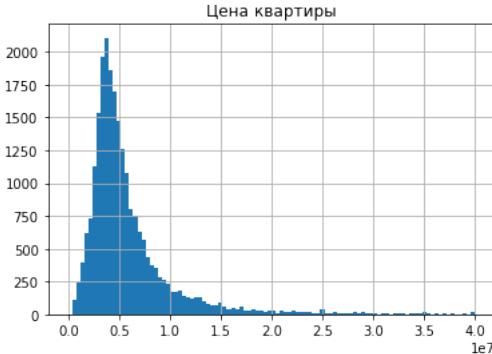
Рассмотрим данные по цене квартиры

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

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

```
data[['last_price']].hist(bins = 100, range = (0,40000000))
plt.title('Цена квартиры')
```

→ Text(0.5, 1.0, 'Цена квартиры')



Вывод

Распределение цены квартир выглядит вполне правдоподобно. Основной пик на 6.5млн. Далее большой "хвост" для экстрадорогих квартир.

75% данных умещается в цену до 6.8 млн В анализ можно будет включить квартиры с ценой до 15млн

Посмотрим распределение цены квартиры от площаи

✗ Проверка аномалий open_plan

Рассмотрим данный по свободной планировке

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

count	23699
unique	2
top	False
freq	23632
Name:	open_plan, dtype: object

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

False	23632
True	67
Name:	open_plan, dtype: int64

Вывод

Всего 67 квартир с открытой планировкой. Тоже вполне себе правдоподобно

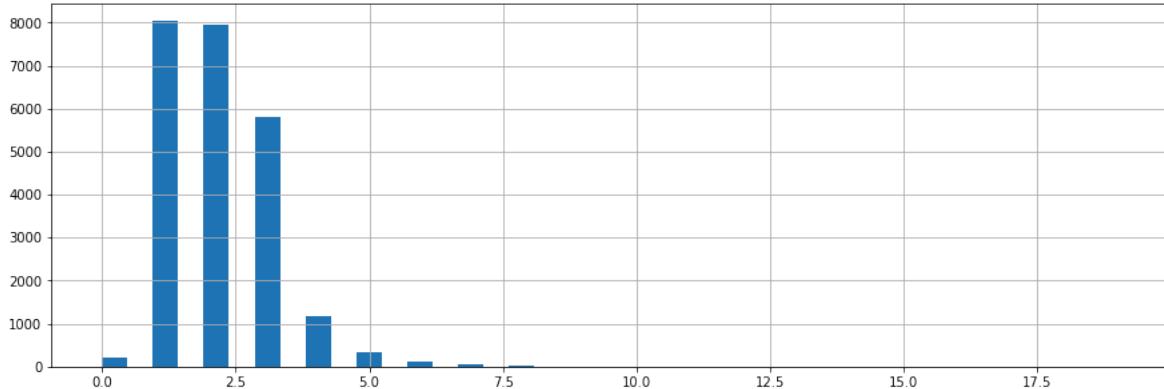
✓ Проверка аномалий rooms

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

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

```
data['rooms'].hist(bins=40, range=(0,19), figsize=(15,5))
```

```
→ <AxesSubplot:>
```



Однокомнатных и двухкомнатных квартир больше всего. Чуть меньше трехкомнатных. Квартир с 4 и более комнатами продается совсем мало.

Выведем информацию о квартирах с ноль комнат

```
print('Количество пропусков в данных о количестве комнат:', data_new.query('rooms==0')['rooms'].count() )
print(f'Доля пропусков: { data_new.query("rooms==0")["rooms"].count()/len(data_new["rooms"]):.0%}')
```

```
→ Количество пропусков в данных о количестве комнат: 181
Доля пропусков: 1%
```

Всего в 1% данных не указано количество комнат. Заменим, что комнат 1

```
data_new.loc[data['rooms']==0, 'rooms'] = 1
```

Вывод

В данных о числе комнат заменили аномальное значение ноль на одну

✓ Проверка аномалий studio

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

```
→ False    23550
  True     149
Name: studio, dtype: int64
```

На первый взгляд все нормально в данных

✓ Проверка аномалий в total_images

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

```
count    23699.000000
mean     9.858475
std      5.682529
min      0.000000
25%     6.000000
50%     9.000000
75%    14.000000
max     50.000000
Name: total_images, dtype: float64
```

Количество фото в объявлениях выглядит правдоподобно

✓ Комментарий ревьюера ✓

В целом, работа по предобработке проведена очень качественно. Можно еще сэкономить память, изменяя типы данных. Например, дополнительного изменить их у площадей квартир с float64 на float32.

✗ Комментарий ревьюера ✗

Также важно посмотреть, какая доля от изначального количества данных осталась после предобработки. Для этого рекомендую зафиксировать начальное количество данных в переменную с помощью метода shape, а затем использовать ее в расчетах.

Прикрепляю полезную ссылку: <https://tonais.ru/library/poluchenie-formy-razmera-dataframe>

Комментарий И1

После удаления дубликатов с числом дней публикаций 45,60,90

- удалено дубликатов: 1622
- осталось записей 22077
- процент удаленной информации 6.8%

После удаления всех аномально больших площадей

- жилых помещений площадью более 100кв.м
- кухонь площадью более 35кв.м
- общих площадей более 200кв.м.
- Осталось записей : 21507
- Удалено: 570 записей
- Доля удаленной информации по площадям около 2,4%
- Доля всей удаленной информации 9,2%

✓ Комментарий ревьюера v2 ✓

Удалось сохранить более 90% от изначальных данных - это хороший результат! 👍

```
# Код ревьюера
data_new.shape[0] / data.shape[0]
```

```
0.9074644499767923
```

✗ Посчитайте и добавьте в таблицу новые столбцы

- цена одного квадратного метра - price_metr
- день недели публикации объявления (0 – понедельник, 1 – вторник и так далее)-day_week_explotion
- месяц публикации объявления - month_explotion
- год публикации объявления - year_explotion
- тип этажа квартиры (значения – «первый», «последний», «другой») - floor_type
- расстояние до центра города в километрах (переведите из м в км и округлите до целых значений).

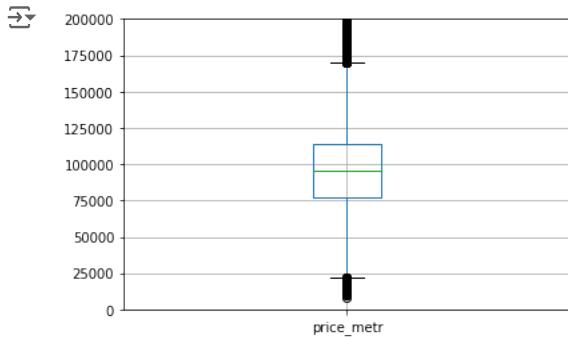
✗ Цена одного квадратного метра price_metr

```
data_new['price_metr'] = data_new['last_price'] / data_new['total_area']
data_new['price_metr'] = data_new['price_metr'].round()
data_new['price_metr']
```

```
0      120370.0
1      83750.0
2      92786.0
3      408176.0
```

```
5      96333.0
...
23693   74194.0
23694   72932.0
23696   44643.0
23697   150987.0
23698   42188.0
Name: price_metr, Length: 21506, dtype: float64
```

```
data_new[['price_metr']].boxplot()
plt.ylim(0, 200000)
plt.show()
```

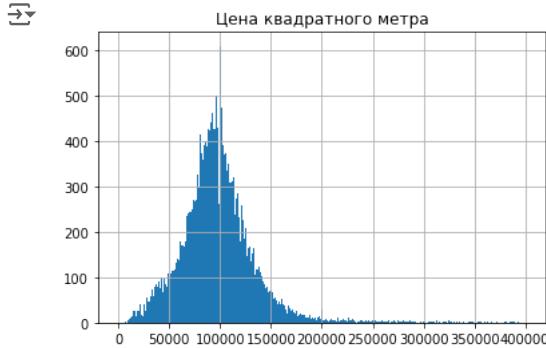


```
data_new['price_metr'].describe()
```

```
count    2.150600e+04
mean     9.833750e+04
std      4.187189e+04
min      7.963000e+03
25%     7.727300e+04
50%     9.537000e+04
75%     1.142860e+05
max     1.546729e+06
Name: price_metr, dtype: float64
```

Основная масса данных в диапазоне 77'000 до 115'000

```
data_new[['price_metr']].hist(bins = 300, range = (0,400000))
plt.title('Цена квадратного метра')
plt.show()
```



Вывод

Распределение цены одного квадратного метра имеет нормальных характер, что свидетельствует о достоверности данных.

У распределения есть длинный хвост, говорящий о том, что есть квартиры с аномально высокой ценой за кв.м. и их можно не учитывать в исследованиях

▼ День недели публикации объявления day_week_explotion

(0 – понедельник, 1 – вторник и так далее)

```
data_new['first_day_exposition'].dtype
```

```
dtype('<M8[ns]')
```

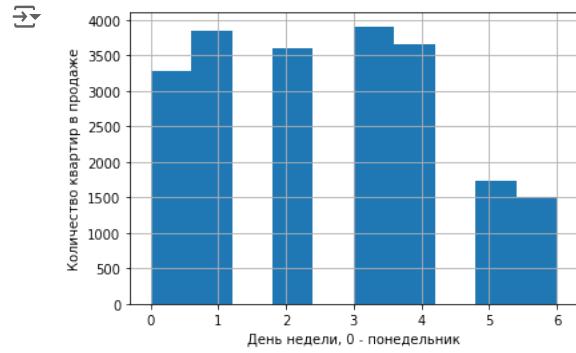
```
data_new['day_week_explotion'] = data_new['first_day_exposition'].dt.weekday
```

```
data_new['day_week_explotion'].head()

→ 0    3
  1    1
  2    3
  3    4
  5    0
Name: day_week_explotion, dtype: int64
```

Посмотрим в какие дни недели чаще всего публикуют объявления и влияет ли день недели на публикацию объявлений квартиры.

```
data_new['day_week_explotion'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('День недели, 0 - понедельник')
plt.show()
```



Вывод

Ожидаемо, в выходные меньше всего публикаций о продаже квартир. Объявления чаще всего публикуют риелторы, а они привыкли работать в будние дни, а выходные отдыхать.

▼ Месяц публикации объявления month_explotion

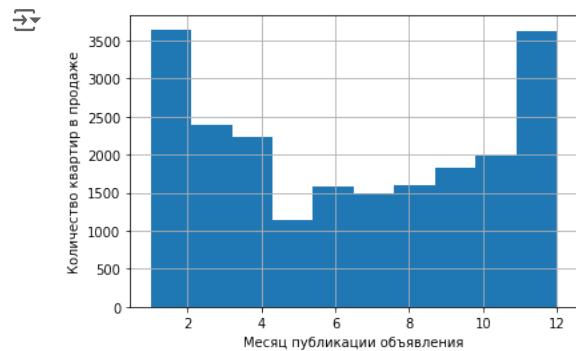
```
data_new['month_explotion'] = data_new['first_day_exposition'].dt.month
```

```
data_new['month_explotion'].head()
```

```
→ 0    3
  1    12
  2    8
  3    7
  5    9
Name: month_explotion, dtype: int64
```

Посмотрим, как месяц зависит на публикацию объявлений.

```
data_new['month_explotion'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Месяц публикации объявления')
plt.show()
```



Вывод

Ожидаемо, снижается количество объявлений в мае, когда начинается дачный сезон.

В летние месяцы количество объявлений несколько ниже, чем в осенние месяцы.

Пик приходится на зимние месяцы.

Странно выглядят сильные пики в январе и декабре, под Новый год. Либо происходит задвоение данных связанное с переходом через год, либо действительно под новый год люди хотят доделать недоделанные дела и в новый год начать с серьезных дел, как продажа квартиры.

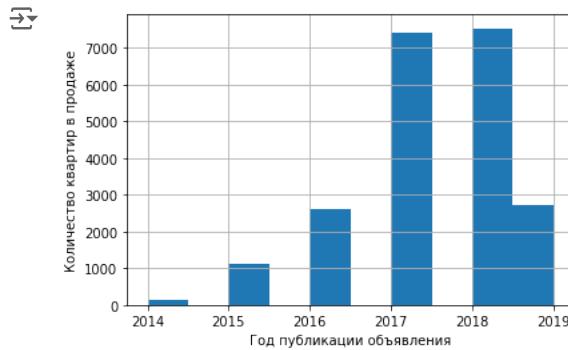
▼ Год публикации объявления - year_explotion

```
data_new['year_explotion'] = data_new['first_day_exposition'].dt.year
```

```
data_new['year_explotion'].head()
```

```
0    2019  
1    2018  
2    2015  
3    2015  
5    2018  
Name: year_explotion, dtype: int64
```

```
data_new['year_explotion'].hist()  
plt.ylabel('Количество квартир в продаже')  
plt.xlabel('Год публикации объявления')  
plt.show()
```



Вывод

Данные для 2019 года даны не за весь год, а только до мая 2019-05-03, поэтому про этот год сказать ничего нельзя.

От года к году число объявлений о продаже квартир растет.

В 2017 и 2018 рост сильно замедлился, но количество объявлений осталось на существенно высоком уровне.

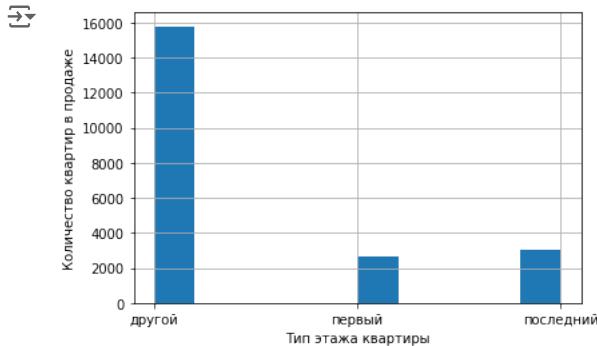
Возможно это тот предел, который вообще может быть достигнут для данного региона.

▼ Тип этажа квартиры (значения – «первый», «последний», «другой») - floor_type

```
def typefloor(data_floors):  
    floor = data_floors['floor']  
    floors_total = data_floors['floors_total']  
    if (floor == 1):  
        return 'первый'  
    if (floor == floors_total):  
        return 'последний'  
    return 'другой'  
data_new['floor_type'] = data_new.apply(typefloor, axis=1)  
display(data_new[['floor_type', 'floor', 'floors_total']].head())
```

	floor_type	floor	floors_total
0	другой	8	16
1	первый	1	11
2	другой	4	5
3	другой	9	14
5	другой	5	12

```
data_new['floor_type'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Тип этажа квартиры')
plt.show()
```



Вывод

Примерно одинаковое количество квартир в продаже на первом и на последнем этаже. На других этажах предсказуемо квартир в продаже больше.

▼ Расстояние до центра города в километрах cityCenters_km

Переведем данные из м в км и округлим сделаем целочисленный тип

```
data_new['cityCenters_km'] = data_new['cityCenters_nearest']/1000
data_new['cityCenters_km'] = data_new['cityCenters_km'].round()
data_new['cityCenters_km'] = data_new['cityCenters_km'].astype(int)
display(data_new[['cityCenters_nearest','cityCenters_km']].head())
```

	cityCenters_nearest	cityCenters_km
0	16028	16
1	18603	19
2	13933	14
3	6800	7
5	21291	21

Вывод

Данные переведены из м в км, округлены и переведены в целочисленный тип

✖ Комментарий рецензента ✖

Стоит вывести несколько строк измененного датасета

Комментарий И1

Посмотрим, какой датасет у нас получился

```
display(data_new.head())
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment
0	20	13000000.0	108	2019-03-07	3	2.7	16	51	8	False
1	7	3350000.0	40	2018-12-04	1	2.6	11	18	1	False
2	10	5196000.0	56	2015-08-20	2	2.6	5	34	4	False
3	0	6490000.0	159	2015-07-24	3	3.0	14	94	9	False
5	10	2890000.0	30	2018-09-10	1	2.6	12	14	5	False

5 rows × 29 columns

✓ Комментарий рецензента v2 ✓

Теперь мы уверены, что изменения внесены верно! 🙌

✓ Проведите исследовательский анализ данных

✗ Комментарий ревьюера ✗

Здесь нас просят рассмотреть следующие параметры: Изучите следующие параметры объектов: общая площадь; жилая площадь; площадь кухни; цена объекта; количество комнат; высота потолков; этаж квартиры; тип этажа квартиры («первый», «последний», «другой»); общее количество этажей в доме; расстояние до центра города в метрах; расстояние до ближайшего аэропорта; расстояние до ближайшего парка; день и месяц публикации объявления. Постройте отдельные гистограммы для каждого из этих параметров. Опишите все ваши наблюдения по параметрам в ячейке с типом markdown.

Нам нужно это сделать, чтобы оценить распределение данных до и после предобработки

Комментарий И1

Ничего вышеперечисленного не выводила в этом пункте, т.к. сполна все пункты рассмотрены в предобработке. И дублировать одну и туже информацию не видела смысла. Приведет это к тому, что я буду создавать дубли пунктов и копировать туда выводы и конечные графики.

✓ Комментарий ревьюера v2 ✓

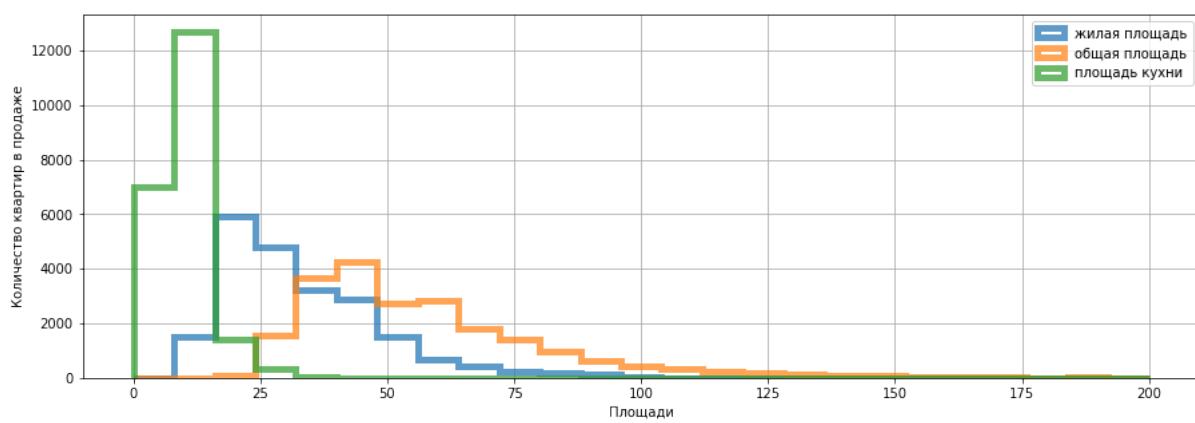
На самом деле, нас неспроста просили сначала вывести гистограммы числовых столбцов таблицы до предобработки, а затем проанализировать их после нее - это необходимо, чтобы мы могли оценить распределение данных до и после предобработки

Комментарий И2 Можно видеть, что на этапе предобработки я старалась выводить в конце пунктов результаты до и после, где это требовалось.

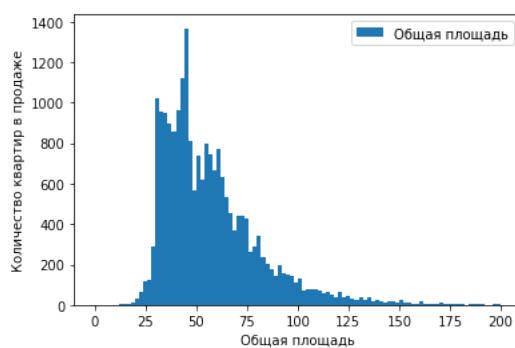
Я согласна, что для визуализации хорошо бы сделать одельные пункты только с графиками и выводами, чтобы не листать всю работу. Но чисто механическо

Комментарий И1 **Общая площадь**

```
ax = data_new.plot(
    kind='hist',
    y='living_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='жилая площадь'
)
data_new.plot(
    kind='hist',
    y='total_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='общая площадь',
    ax=ax,
    grid=True,
    legend=True
)
data_new.plot(
    kind='hist',
    y='kitchen_area',
    histtype='step',
    range=(0, 200),
    bins=25,
    linewidth=5,
    alpha=0.7,
    label='площадь кухни',
    ax=ax,
    grid=True,
    legend=True,
    figsize=(15,5)
)
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Площади')
plt.show()
```



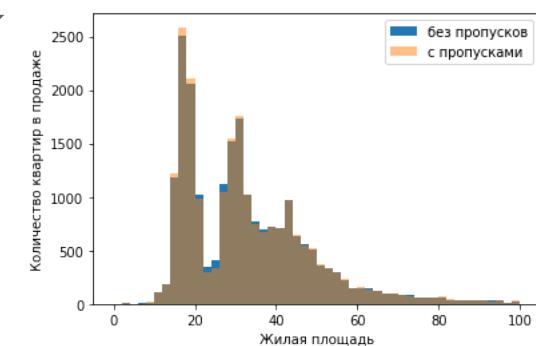
```
data_new.plot(
    kind='hist',
    y='total_area',
    bins=100,
    range=(0, 200),
    label='Общая площадь'
)
plt.xlabel('Общая площадь')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



На гистограмме видны три пика, как раз характеризующие однокомнатные, двухкомнатные и трехкомнатные квартиры.

Комментарий И1 Жилая площадь

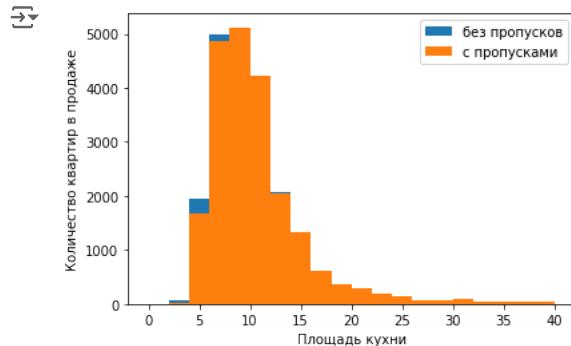
```
ax = data_new.plot(
    kind='hist',
    y='living_area',
    bins=50,
    range=(0, 100),
    label='без пропусков'
)
data.plot(
    kind='hist',
    y='living_area',
    ax=ax,
    alpha=0.5,
    bins=50,
    range=(0, 100),
    label='с пропусками'
)
plt.xlabel('Жилая площадь')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



На жилую площадь больше всего влияет зависимость от числа комнат, в пользу этого говорят соответствующие пики на гистограмме

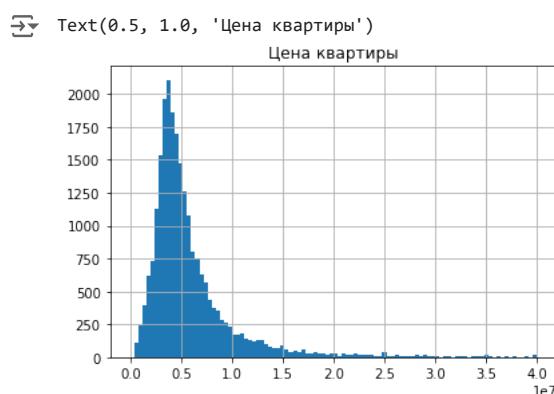
Комментарий И1 Площадь кухни

```
ax=data_new.plot(
    kind='hist',
    y='kitchen_area',
    bins=20,
    range=(0, 40),
    label='без пропусков'
)
data.plot(
    kind='hist',
    y='kitchen_area',
    bins=20,
    ax=ax,
    alpha=1,
    range=(0, 40),
    label='с пропусками'
)
plt.xlabel('Площадь кухни')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



Комментарий И1 Цена объекта

```
data[['last_price']].hist(bins = 100, range = (0,40000000))
plt.title('Цена квартиры')
```



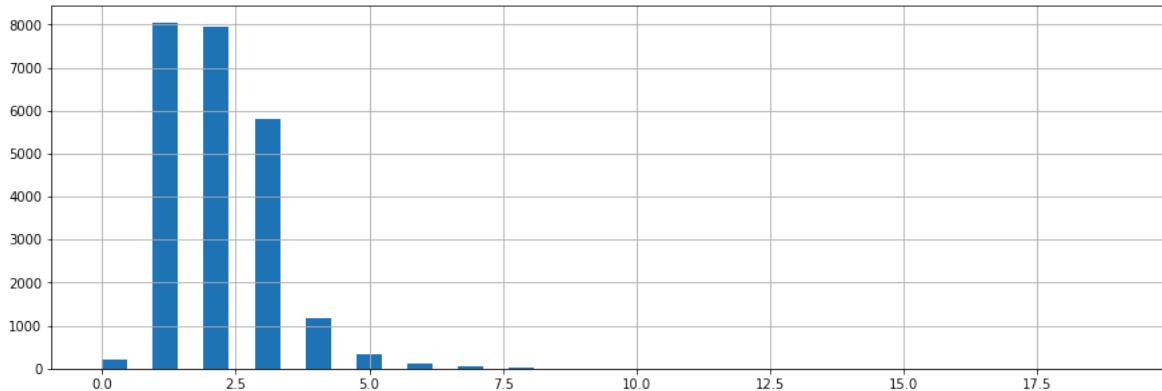
Распределение цены квартир выглядит вполне правдоподобно. Основной пик на 6.5млн. Далее большой "хвост" для экстрадорогих квартир.

75% данных умещается в цену до 6.8 млн

Комментарий И1 Количество комнат

```
data['rooms'].hist(bins=40, range=(0,19), figsize=(15,5))
```

⇨ <AxesSubplot:>

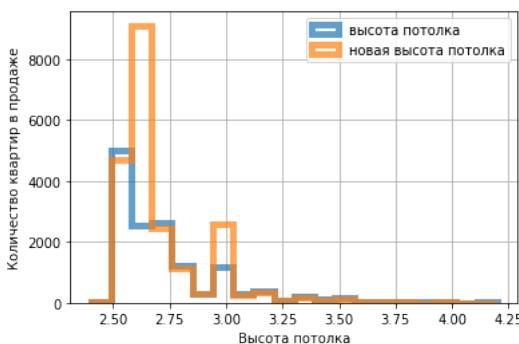


Однокомнатных и двухкомнатных квартир больше всего. Чуть меньше трехкомнатных. Квартиры с 4 и более комнатами продаются совсем мало.

Комментарий И1 Высота потолков

```
ax = data.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(2.4, 4.2),
    bins=20,
    linewidth=5,
    alpha=0.7,
    label='высота потолка'
)
data_new.plot(
    kind='hist',
    y='ceiling_height',
    histtype='step',
    range=(2.4, 4.2),
    bins=20,
    linewidth=5,
    alpha=0.7,
    label='новая высота потолка',
    ax=ax,
    grid=True,
    legend=True,
)
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Высота потолка')
plt.show()
```

⇨ <Figure>

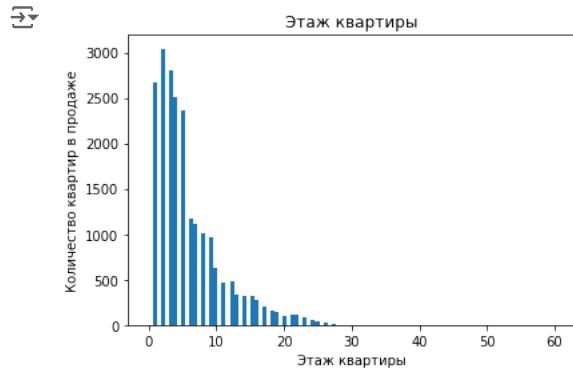


"Хвосты" аномальных значений убраны и заменены медианными значениями и они дали новый пик в распределении высоты потолков в районе 2.6м Отчетливее стал виден пик в 3м для квартир в центре города. Центр города всегда характеризуется более

уникальным жильем. На периферии сторится больше однотипного жилья.

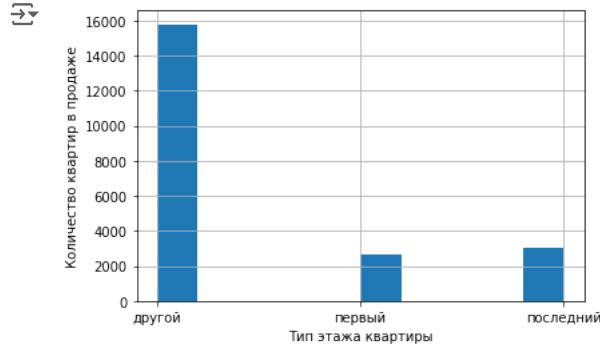
Комментарий И1 Этаж квартиры

```
plt.hist(data_new['floor'], bins=100, range=(0,60),)
plt.title('Этаж квартиры')
plt.xlabel('Этаж квартиры')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



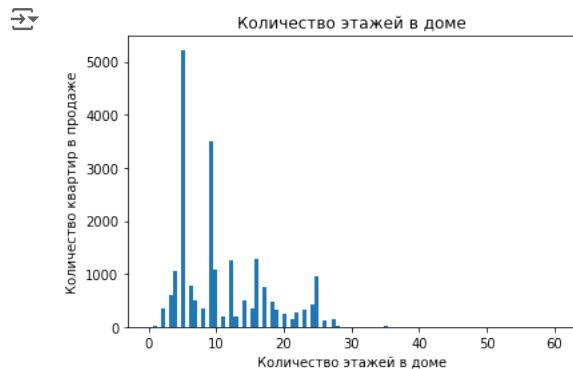
Комментарий И1 Тип этажа квартиры («первый», «последний», «другой»)

```
data_new['floor_type'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Тип этажа квартиры')
plt.show()
```



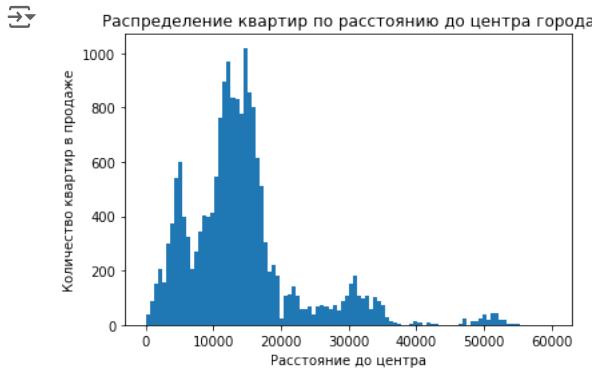
Комментарий И1 Общее количество этажей в доме

```
plt.hist(data_new['floors_total'], bins=100, range=(0,60),)
plt.title('Количество этажей в доме')
plt.xlabel('Количество этажей в доме')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



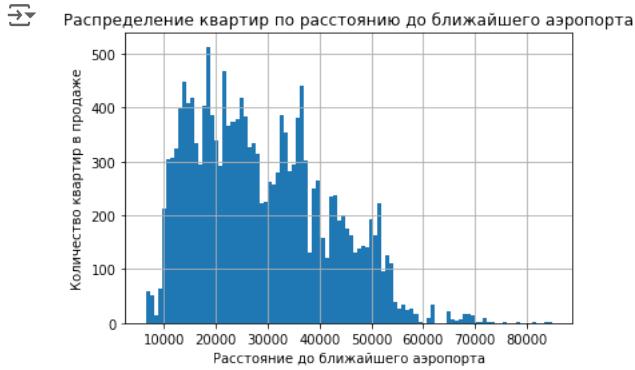
Комментарий И1 Расстояние до центра города в метрах

```
plt.hist(data['cityCenters_nearest'], range=(0,60000), bins=100)
plt.title('Распределение квартир по расстоянию до центра города')
plt.xlabel('Расстояние до центра')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



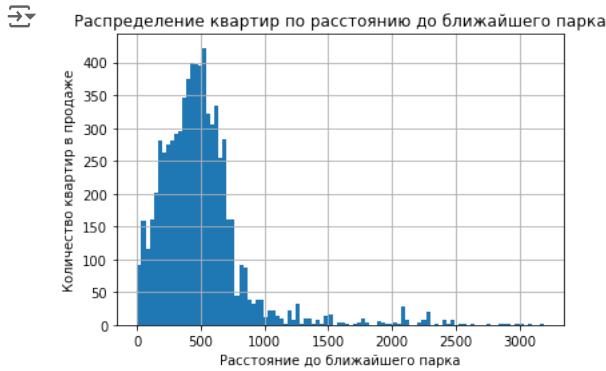
Комментарий И1 Расстояние до ближайшего аэропорта

```
data_new['airports_nearest'].hist(bins=100)
plt.title('Распределение квартир по расстоянию до ближайшего аэропорта')
plt.xlabel('Расстояние до ближайшего аэропорта')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



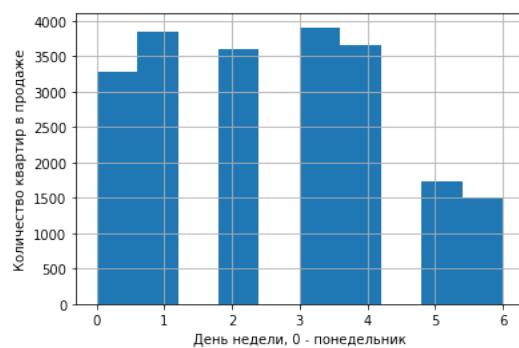
Комментарий И1 Расстояние до ближайшего парка

```
data_new['parks_nearest'].hist(bins=100)
plt.title('Распределение квартир по расстоянию до ближайшего парка')
plt.xlabel('Расстояние до ближайшего парка')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



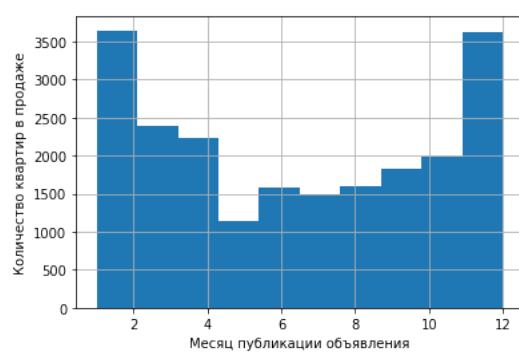
Комментарий И1 День и месяц публикации объявления

```
data_new['day_week_explotion'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('День недели, 0 - понедельник')
plt.show()
```



Ожидаемо, в выходные меньше всего публикуют о продаже квартир. Объявления чаще всего публикуют риелторы, а они привыкли работать в будние дни, а выходные отдыхать.

```
data_new['month_explotion'].hist()
plt.ylabel('Количество квартир в продаже')
plt.xlabel('Месяц публикации объявления')
plt.show()
```



Ожидаемо, снижается количество объявлений в мае, когда начинается дачный сезон.

В летние месяцы количество объявлений несколько ниже, чем в осенние месяцы.

Пик приходится на зимние месяцы.

Странно выглядят сильные пики в январе и декабре, под Новый год. Либо происходит задвоение данных связанное с переходом через год, либо действительно под новый год люди хотят доделать недоделанные дела и в новый год начать с серьезных дел, как продажа квартиры.

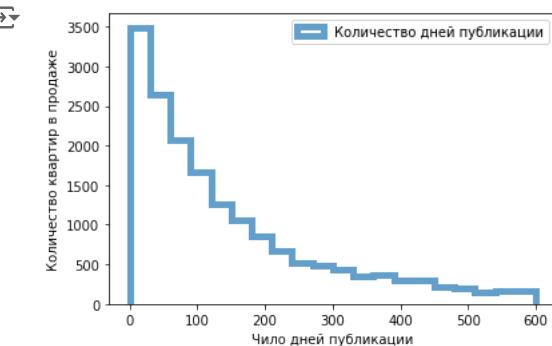
✓ Комментарий ревьюера v2 ✓

Хороший результат. Хочу добавить, что обычно размеры двухкомнатных квартир составляют от 50 до 100 квадратных метров, что и соответствует проведенному анализу.

Могу посоветовать на будущее хорошую статью по оформлению графиков: <https://devpractice.ru/matplotlib-lesson-3-3-text-elements/>

▼ Скорость продажи квартиры

```
data_new.plot(
    kind='hist',
    y='days_exposition',
    histtype='step',
    bins=20,
    range = (0,600),
    linewidth=5,
    alpha=0.7,
    label='Количество дней публикации'
)
plt.xlabel('Число дней публикации')
plt.ylabel('Количество квартир в продаже')
plt.show()
```



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

	count	mean	std	min	25%	50%	75%	max
	18482.000000	188.699383	222.727773	1.000000	41.000000	108.000000	247.000000	1580.000000
Name:	days_exposition	dtype:	float64					

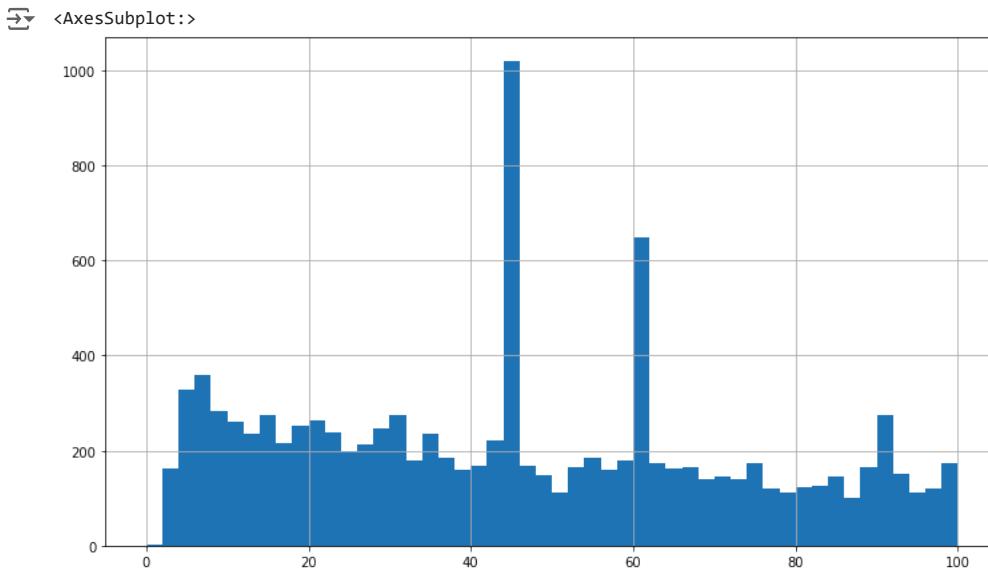
Вывод

Среднее число дней, сколько продаются квартиры 186 дней. Среднее сильно смещено, т.к. много квартир, которые годами висят в продаже.

Медианное значение числа дней в продаже равно 94, что ближе к истине - это три месяца. Действительно адекватный срок продажи. До 75% квартир продается за 237 дней (8 месяцев). Медленной продажей можно считать все что больше 8 месяцев.

Очень быстрой продажей можно считать быстрее 43 дней. За это время продается 25% всех квартир.

```
# Код рецензента
data['days_exposition'].hist(bins=50, range=(0,100), figsize=(12, 7))
```



✓ Комментарий рецензента ✓

На гистограмме заметны пики примерно через 45, 60 и 90 дней после начала продажи. Можно посмотреть на условия размещения объявлений в Яндекс.Недвижимости - <https://yandex.ru/support/realty/owner/home/add-ads-housing.html> С учетом того, что после формирования датасета правила размещения поменялись, эти пики являются следами автоматического снятия объявлений - поэтому оценивать скорость продажи стоит действительно по диаграмме размаха

▼ Факторы больше всего влияющие на общую (полную) стоимость объекта

Рассмотрим следующие факторы, от которых зависит стоимость квартиры:

- общей площади - total_area
- жилой площади - living_area

- площади кухни - kitchen_area
- количества комнат -rooms
- этажа, на котором расположена квартира (первый, последний, другой) - floor_type
- даты размещения (день недели, месяц, год) day_week_explotion, year_explotion, month_explotion

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

Выведем коэффициент корреляции факторов указанных выше с общей стоимостью квартиры last_price и с ценой за метр price_metr

```
data_analysis = data_new[['last_price','total_area','living_area','kitchen_area','rooms','floor_type','day_week_explotion', 'month_explotion','year_explotion']]  
print( data_analysis.corr().loc[:,['last_price']])
```

	last_price
last_price	1.000000
total_area	0.680849
living_area	0.579846
kitchen_area	0.514898
rooms	0.389734
day_week_explotion	-0.013029
month_explotion	0.007390
year_explotion	-0.036593

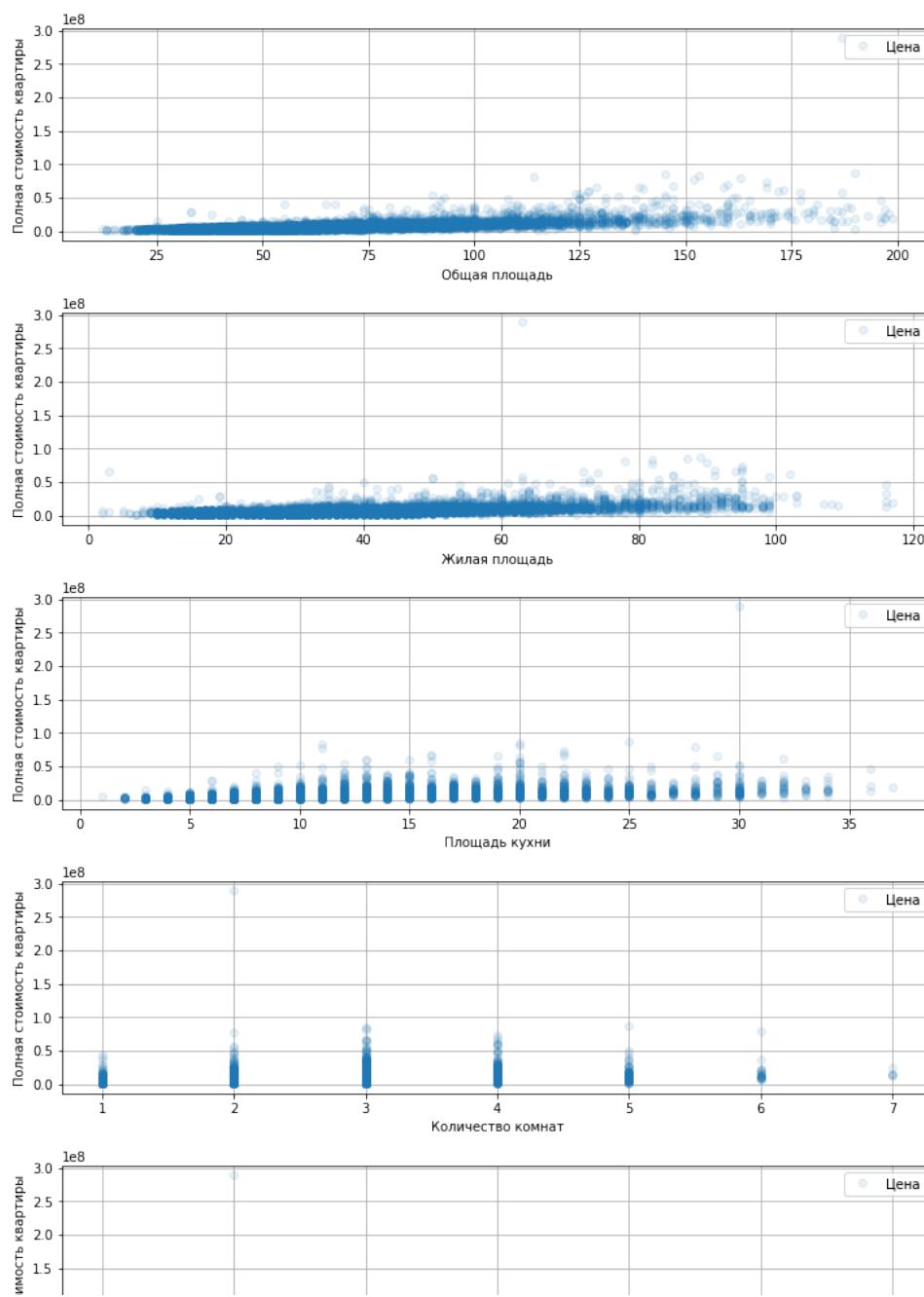
Самый большой коэффициент линейной корреляции полной стоимости квартиры с общей площадью. Меньше корреляция цены с жилой площадью, площадью кухни. Совсем незначительный коэффициент корреляции с количеством комнат.

Общая стоимость квартиры совсем не коррелирует с датой: днем, месяцем, годом публикации квартиры.

Посмотрим графики зависимости общей стоимости от всех факторов

```
data_analysis.plot(  
    x='total_area',  
    y='last_price',  
    xlabel='Общая площадь',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='living_area',  
    y='last_price',  
    xlabel='Жилая площадь',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='kitchen_area',  
    y='last_price',  
    xlabel='Площадь кухни',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='rooms',  
    y='last_price',  
    xlabel='Количество комнат',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='day_week_explotion',  
    y='last_price',  
    xlabel='День недели размещения объявления',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='month_explotion',  
    y='last_price',  
    xlabel='Месяц размещения объявления',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)  
plt.show()  
data_analysis.plot(  
    x='year_explotion',  
    y='last_price',  
    xlabel='Год размещения объявления',  
    ylabel='Полная стоимость квартиры',  
    style='o',  
    grid=True,  
    figsize=(12,3),  
    label = 'Цена',  
    alpha=0.1  
)
```

```
    alpha=0.1  
)  
plt.show()
```



Уберем из данных выбросы:

- цены больше 15'000'000
- общую площадь больше 200кв.м.
- жилую площадь больше 130кв.м
- площадь кухни больше 40кв.м
- количество комнат больше 7

Посчитаем новые коэффициенты корреляции.

```
print( data_analysis.query('last_price<15000000 and total_area<200 and living_area<130 and kitchen_area<40 and rooms<7' ).\n    corr().loc[:,['last_price']])
```

	last_price
last_price	1.000000
total_area	0.740446
living_area	0.630120
kitchen_area	0.552216
rooms	0.479015
day_week_explotion	-0.016730
month_explotion	0.002562
year_explotion	-0.010384

Коэффициенты корреляции несколько подросли.

Посчитаем средние значения цены для с помощью сводной таблицы и еще раз пересчитаем коэффициент корреляции

```
#средняя цена от общей площади
stat_total_area = data_analysis.query('last_price<15000000 and total_area<200').\
pivot_table(index='total_area', values='last_price')
stat_total_area['total_area'] = stat_total_area.index
print('Коэффициент корреляции с общей площадью',\
      stat_total_area['last_price'].corr(stat_total_area['total_area']))

#медианная цена от общей площади
median_total_area = data_analysis.query('last_price<15000000 and total_area<200').\
pivot_table(index='total_area', values='last_price', aggfunc='median')
median_total_area['total_area'] = median_total_area.index
print('Коэффициент корреляции медианы цены с общей площадью',\
      median_total_area['last_price'].corr(median_total_area['total_area']))

#средняя цена от жилой площади
stat_living_area = data_analysis.query('last_price<15000000 and living_area<130').\
pivot_table(index='living_area', values='last_price')
stat_living_area['living_area'] = stat_living_area.index
print('Коэффициент корреляции с жилой площадью',\
      stat_living_area['last_price'].corr(stat_living_area['living_area']))

#медианная цена от жилой площади
median_living_area = data_analysis.query('last_price<15000000 and living_area<130').\
pivot_table(index='living_area', values='last_price', aggfunc='median')
median_living_area['living_area'] = median_living_area.index
print('Коэффициент корреляции медианы цены с жилой площадью',\
      median_living_area['last_price'].corr(median_living_area['living_area']))

stat_kitchen_area = data_analysis.query('last_price<15000000 and kitchen_area<40').\
pivot_table(index='kitchen_area', values='last_price')
stat_kitchen_area['kitchen_area'] = stat_kitchen_area.index
print('Коэффициент корреляции с площадью кухни',\
      stat_kitchen_area['last_price'].corr(stat_kitchen_area['kitchen_area']))

stat_rooms = data_analysis.query('last_price<15000000 and rooms<7').\
pivot_table(index='rooms', values='last_price')
stat_rooms['rooms'] = stat_rooms.index
print('Коэффициент корреляции с числом комнат',\
      stat_rooms['last_price'].corr(stat_rooms['rooms']))

stat_floor = data_analysis.query('last_price<15000000').\
pivot_table(index='floor_type', values='last_price')
stat_floor['floor_type'] = stat_floor.index

stat_day_week = data_analysis.query('last_price<15000000').\
pivot_table(index='day_week_explotion', values='last_price')
stat_day_week['day_week_explotion'] = stat_day_week.index

stat_month = data_analysis.query('last_price<15000000').\
pivot_table(index='month_explotion', values='last_price')
stat_month['month_explotion'] = stat_month.index
print('Коэффициент корреляции с месяцем размещения объявления',\
      stat_month['last_price'].corr(stat_month['month_explotion']))

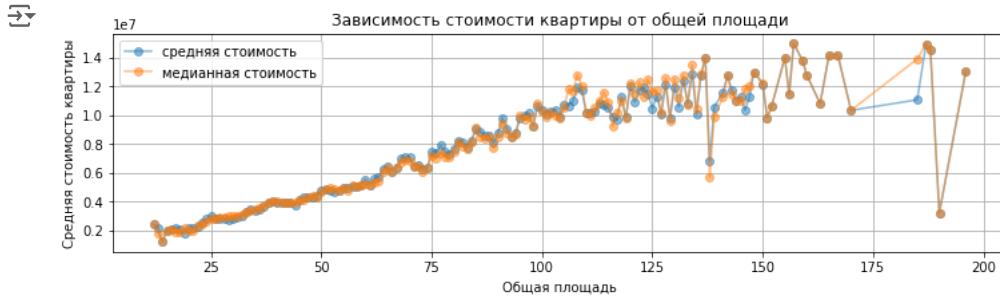
stat_year = data_analysis.query('last_price<15000000').\
pivot_table(index='year_explotion', values='last_price')
stat_year['year_explotion'] = stat_year.index
print('Коэффициент корреляции с годом размещения объявления',\
      stat_year['last_price'].corr(stat_year['year_explotion']))
```

- ⤵ Коэффициент корреляции с общей площадью 0.9117274178092887
- Коэффициент корреляции медианы цены с общей площадью 0.911202003069811
- Коэффициент корреляции с жилой площадью 0.9690312346795481
- Коэффициент корреляции медианы цены с жилой площадью 0.9644300527029034
- Коэффициент корреляции с площадью кухни 0.9566984632517839
- Коэффициент корреляции с числом комнат 0.9984635668590505
- Коэффициент корреляции с месяцем размещения объявления 0.07717819706118083
- Коэффициент корреляции с годом размещения объявления -0.681389964132372

```

ax = stat_total_area.plot(
    y='last_price',
    x='total_area',
    xlabel='Общая площадь',
    ylabel='Средняя стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    label='средняя стоимость',
    title = 'Зависимость стоимости квартиры от общей площади'
    #ylim=(0, 1000)
)
median_total_area.plot(
    y='last_price',
    x='total_area',
    xlabel='Общая площадь',
    ylabel='Средняя стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    ax=ax,
    label='медианная стоимость',
    title = 'Зависимость стоимости квартиры от общей площади'
    #ylim=(0, 1000)
)
plt.show()

```



```

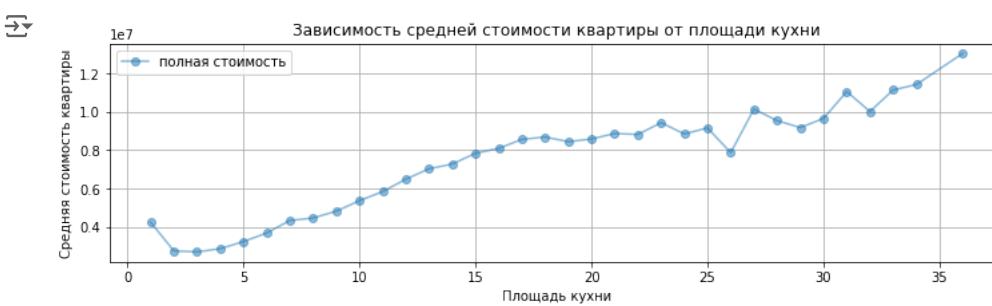
ax=stat_living_area.plot(
    y='last_price',
    x='living_area',
    xlabel='Жилая площадь',
    ylabel='Стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    label='средняя стоимость',
    title = 'Зависимость стоимости квартиры от жилой площади'
    #ylim=(0, 1000)
)
median_living_area.plot(
    y='last_price',
    x='living_area',
    xlabel='Жилая площадь',
    ylabel='Стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    ax=ax,
    label='медиана стоимости',
    title = 'Зависимость стоимости квартиры от жилой площади'
    #ylim=(0, 1000)
)
plt.show()

```



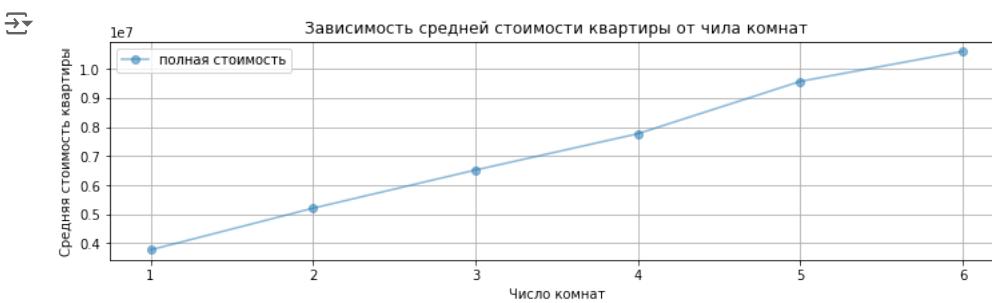
```
stat_kitchen_area.plot(
    y='last_price',
    x='kitchen_area',
    xlabel='Площадь кухни',
    ylabel='Средняя стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    label='полная стоимость',
    title = 'Зависимость средней стоимости квартиры от площади кухни'
    #ylim=(0, 1000)
)

plt.show()
```



```
stat_rooms.plot(
    y='last_price',
    x='rooms',
    xlabel='Число комнат',
    ylabel='Средняя стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    label='полная стоимость',
    title = 'Зависимость средней стоимости квартиры от числа комнат'
    #ylim=(0, 1000)
)

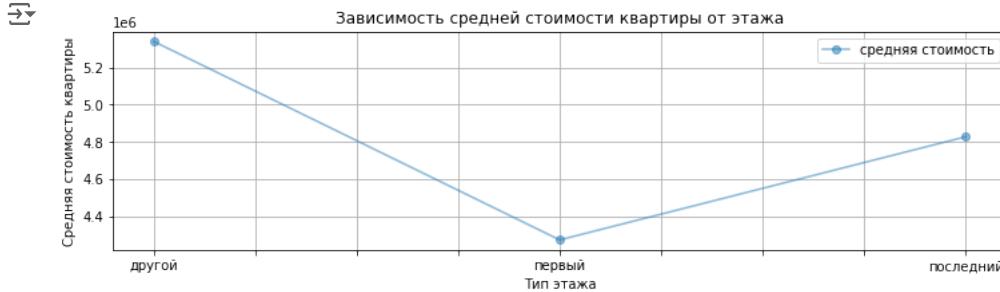
plt.show()
```



```

stat_floor.plot(
    y='last_price',
    x='floor_type',
    xlabel='Тип этажа',
    ylabel='Средняя стоимость квартиры',
    style='o-',
    grid=True,
    figsize=(12,3),
    alpha = 0.5,
    label='средняя стоимость',
    title = 'Зависимость средней стоимости квартиры от этажа'
    #ylim=(0, 1000)
)
plt.show()

```



Видим, что квартиры на первом этаже в целом дешевле всех остальных.

ВЫВОД:

Получаем, что для квартир общей площадью до 200кв.м и ценой до 15млн коэффициент корреляции стремится к 1 между средней и медианной ценой и площадью: обжей, жилой, кухни и числом комнат.

Год размещения объявления влияет, но в меньшей степени на полную стоимость квартиры. День и месяц не оказывают влияния на полную цену.

✓ Комментарий ревьюера ✓

Хорошие результаты. Отмету, что скорее всего, низкую цену на квартиры на первом и последнем этажах можно аргументировать плохими условиями - шум от улиц и моторов лифта, охлаждаемость помещения зимой.

▼ Средняя цена квадратного метра в разных населенных пунктах

Посчитаем среднюю цену одного квадратного метра в 10 населённых пунктах с наибольшим числом объявлений. Выделим населённые пункты с самой высокой и низкой стоимостью квадратного метра.

Выведем 10 населенных пунктов с наибольшим числом объявлений

```

locality_counts = data_new.pivot_table(index='locality_name', values='price_metr', \
                                         aggfunc=['count','mean'])
locality_counts.columns=[ 'Число объявлений', 'Средняя цена метра']

print('10 населенных пунктов с наибольшим числом объявлений')
locality_counts = locality_counts.sort_values(by='Число объявлений',ascending=False)

locality_counts.head(10)

```

→ 10 населенных пунктов с наибольшим числом объявлений
 Чило объявлений Средняя цена метра

locality_name	count	mean
санкт-петербург	14295	112785.890031
поселок мурино	519	86150.699422
кудрово	427	95765.786885
поселок шушары	402	79362.604478
всеволожск	367	68904.961853
пушкин	342	103571.979532
колпино	305	75935.403279
поселок парголово	290	91255.165517
гатчина	282	68699.425532
выборг	207	58125.685990

```
locality_counts.columns=['count','mean']
# сохраним только 10 значений
locality_counts_10 = locality_counts.query('count>=237')
locality_counts_10
```

locality_name	count	mean
санкт-петербург	14295	112785.890031
поселок мурино	519	86150.699422
кудрово	427	95765.786885
поселок шушары	402	79362.604478
всеволожск	367	68904.961853
пушкин	342	103571.979532
колпино	305	75935.403279
поселок парголово	290	91255.165517
гатчина	282	68699.425532

```
print('Населенный пункт с самой высокой средней стоимостью квадратного метра')
display(locality_counts.loc[ locality_counts['mean']==locality_counts['mean'].max() ])
print('Населенный пункт с самой низкой средней стоимостью квадратного метра')
display(locality_counts.loc[ locality_counts['mean']==locality_counts['mean'].min() ])
#locality
```

→ Населенный пункт с самой высокой средней стоимостью квадратного метра

locality_name	count	mean
санкт-петербург	14295	112785.890031

Населенный пункт с самой низкой средней стоимостью квадратного метра

locality_name	count	mean
деревня старополье	2	11357.5

```
locality_counts_10.plot(
    y='count',
    kind='pie',
    figsize=(8,8),
    title='Населенные пункты с наибольшим числом квартир в продаже',
    ylabel='Число квартир в продаже'
)
```