Содержание

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.

Нам предстоит изучить данные, сделать предобработку, выполнить некоторые рассчеты и произвести анализ данных полученных результатов. Также наверняка в данных будут встречаться выбросы, которые необходимо будет обработать. Главной задачей стоит ответ на постпавленные заказчиком вопросы: Какие факторы влияют на стоимость квартиры? Как влияет площадь квартиры, удаление от центра, кол-во комнат и тд на стоимость?

В конце подведем итог и сделаем вывод.

Изучение данных из файла

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

In [2]:
```
url = 'https://code.s3.yandex.net/datasets/real_estate_data.csv'

try:
    df = pd.read_csv('real_estate_data.csv', sep='\t')
except:
    df = pd.read_csv(url, sep='\t')
```

In [3]:
```
pd.options.display.max_columns = 50
```

In [4]:
```
df.head()
```

Out[4]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70	16.0	
1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN	11.0	
2	10	5196000.0	56.0	2015-08-20T00:00:00	2	NaN	5.0	
3	0	64900000.0	159.0	2015-07-24T00:00:00	3	NaN	14.0	
4	2	10000000.0	100.0	2018-06-19T00:00:00	2	3.03	14.0	

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

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

Вот и наши данные:

- `airports_nearest` — расстояние до ближайшего аэропорта в метрах (м)
- `balcony` — число балконов
- `ceiling_height` — высота потолков (м)
- `cityCenters_nearest` — расстояние до центра города (м)
- `days_exposition` — сколько дней было размещено объявление (от публикации до снятия)
- `first_day_exposition` — дата публикации
- `floor` — этаж
- `floors_total` — всего этажей в доме
- `is_apartment` — апартаменты (булев тип)
- `kitchen_area` — площадь кухни в квадратных метрах (м²)
- `last_price` — цена на момент снятия с публикации
- `living_area` — жилая площадь в квадратных метрах (м²)
- `locality_name` — название населённого пункта
- `open_plan` — свободная планировка (булев тип)
- `parks_around3000` — число парков в радиусе 3 км
- `parks_nearest` — расстояние до ближайшего парка (м)
- `ponds_around3000` — число водоёмов в радиусе 3 км
- `ponds_nearest` — расстояние до ближайшего водоёма (м)
- `rooms` — число комнат
- `studio` — квартира-студия (булев тип)
- `total_area` — площадь квартиры в квадратных метрах (м²)
- `total_images` — число фотографий квартиры в объявлении

In [6]:
```
df.describe()
```

Out[6]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	livi
count	23699.000000	2.369900e+04	23699.000000	23699.000000	14504.000000	23613.000000	21796
mean	9.858475	6.541549e+06	60.348651	2.070636	2.771499	10.673824	34
std	5.682529	1.088701e+07	35.654083	1.078405	1.261056	6.597173	22
min	0.000000	1.219000e+04	12.000000	0.000000	1.000000	1.000000	2
25%	6.000000	3.400000e+06	40.000000	1.000000	2.520000	5.000000	18
50%	9.000000	4.650000e+06	52.000000	2.000000	2.650000	9.000000	30
75%	14.000000	6.800000e+06	69.900000	3.000000	2.800000	16.000000	42
max	50.000000	7.630000e+08	900.000000	19.000000	100.000000	60.000000	409

Вывод

В наших данных 23699 строк. В некоторых есть пропуски, с которыми предстоит разобраться. Судя по минимальной и максимальной стоимости квартир, в данных точно будут присутсвовать выбросы, с которыми мы разберемся позднее.

Предобработка данных

В некоторых колонках отсутствуют значения. Посмотрим на кол-во пропусков, их долю и попробуем разобраться с ними

In [7]:
```
df.isnull().mean().sort_values(ascending=False)
```

Out[7]:
```
is_apartment            0.882906
parks_nearest           0.659100
ponds_nearest           0.615596
balcony                 0.486054
ceiling_height          0.387991
airports_nearest        0.233850
cityCenters_nearest     0.232879
ponds_around3000        0.232837
parks_around3000        0.232837
days_exposition         0.134225
kitchen_area            0.096122
living_area             0.080299
floors_total            0.003629
locality_name           0.002068
total_images            0.000000
last_price              0.000000
studio                  0.000000
floor                   0.000000
rooms                   0.000000
first_day_exposition    0.000000
total_area              0.000000
open_plan               0.000000
dtype: float64
```

Начнем с данных, доля пропусков в которых больше всего

- `is_apartment` - Отсутствие значения скорее всего говорит о том, что данный объект не является апартаментами. Логично заменить пропуски на False
- `ponds_around3000` - Логично предположить, что водоёмов в радиусе 3км нет. Заменяем но ноль

- `parks_around3000` - Логично предположить, что парков в радиусе 3км нет. Заменяем но ноль
- `balcony` - Если человек не указал число балконов — скорее всего, их нет. Такие пропуски правильно заменить на 0.
- `days_exposition` - Поскольку дата публикации известна для всех данных, вероятнее всего пропуски в данной колонке обусловлены тем, что на момент выгрузки данных, объявление снято не было. Тк нам предстоит изучить время продажи квартиры и построить гистограмму, предлагаю заменить их на максимальное значение + 100 дней. На гистограмме данные объявления не будут мешать остальным данным, а расположаться отдельно от всех
- `ceiling_height` - Скорее всего, в объявлении не указана высота потолков. Предлагаю заменить ее на медианное значение.
- `floors_total` - Таких дынных ничтожно мало (менее 1%). Предлагаю избавиться от них.
- `locality_name` - Таких дынных ничтожно мало (менее 1%). Предлагаю избавиться от них.

Взглянем на данные, полученные автоматически на основе картографических данных. Это колонки: `parks_nearest`, `ponds_nearest`, `airports_nearest` и `cityCenters_nearest`. Могу предположить, что данные отсутствуют из-за того, что у объекта отсутсвтует точный адрес. Например, не указан номер дома. Из-за этого получить расстояние не удалось. Оставим пока пропуски в этих данных.
Была мысль заменить пропуски в `airports_nearest` на среднее значение данного показателя для каждого населенного пункта, но нам пока не нужны эти данные, поэтому время на это тратить не будем.
В работе нам точно понадобится информация о расстоянии до центра для Санкт-Петербурга. Тут тоже можно подумать и заменить пропуски на конкретные значения, но для начала посмотрим сколько всего таких квартир в Санкт-петербурге, для которых нет расстояния до центра.

In [8]:
```
#Сразу посчитаем долю таких квартир
round(len(df.query('(cityCenters_nearest.isnull()) and (locality_name == "Санкт-Петер
```

Out[8]:
```
0.39
```

Таких квартир менее 0.5%. Оставим эти пропуски, а когда дойдем до пункта с анализом расстояния квартир и цены, скорее всего, просто избавимся от них.
А теперь вернемся к остальным данным

In [9]:
```
#Заменяем пропуски
df['is_apartment'] = df['is_apartment'].fillna(False)
df['ponds_around3000'] = df['ponds_around3000'].fillna(0).astype('int')
df['parks_around3000'] = df['parks_around3000'].fillna(0).astype('int')
df['balcony'] = df['balcony'].fillna(0).astype('int')
df['days_exposition'] = df['days_exposition'].fillna(df['days_exposition'].max() + 10
df['ceiling_height'] = df['ceiling_height'].fillna(df['ceiling_height'].median())

# Удаляем отсутсвующие данные по локации и кол-ву этажей
df.dropna(subset=['locality_name'], inplace=True)
df.dropna(subset=['floors_total'], inplace=True)
```

- `kitchen_area` - Можно предположить, что в объявлении указана только общая площадь, но не указана площадь кухни. Предлагаю найти отношение средней площади

кухни к средней жилой и умножить данный коэффициент на общую площадь в тех местах, где значение пропущено. Тоже самое касается и `living_area`.\

Но для большей точности, предлагаю высчитывать средние площади в разрезе по комнатам, тк площадь кухни в однокомнатной и пятикомнатной квартире, наверняка сильно различаются.
Создадим сводную таблицу и посчитаем средние площади кухни и жилой, а также их отношение к общей.

In [10]:
```
# Создадим сводную таблицу
areas  = df.pivot_table(index='rooms', values=['kitchen_area', 'living_area', 'total_
areas.columns=['kitchen_avg', 'living_avg', 'total_avg']
areas['kitchen_ratio'] = areas['kitchen_avg'] / areas['total_avg']
areas['living_ratio'] = areas['living_avg'] / areas['total_avg']
display(areas)
#Присоединим колонки kitchen_ratio и living_ratio к общему датафрейму
df = df.join(areas[['kitchen_ratio', 'living_ratio']], on='rooms')
#Заменим пропуски в строках с количеством комант 0 на среднее по всему датафрейму
```

rooms	kitchen_avg	living_avg	total_avg	kitchen_ratio	living_ratio
0	NaN	18.865246	29.321701	NaN	0.643389
1	9.544076	17.899507	37.636569	0.253585	0.475588
2	9.970915	31.731865	55.821274	0.178622	0.568455
3	11.292920	47.374108	77.706740	0.145327	0.609652
4	13.672140	66.994806	107.521459	0.127157	0.623083
5	18.367649	100.009247	161.517200	0.113719	0.619186
6	21.098229	131.799796	202.571635	0.104152	0.650633
7	21.998077	163.564151	265.325424	0.082910	0.616466
8	24.866667	168.683333	259.233333	0.095924	0.650701
9	25.071429	190.257143	305.975000	0.081939	0.621806
10	22.866667	165.566667	259.566667	0.088096	0.637858
11	12.600000	133.900000	188.900000	0.066702	0.708841
12	112.000000	409.700000	900.000000	0.124444	0.455222
14	21.250000	195.150000	304.200000	0.069855	0.641519
15	100.000000	409.000000	590.000000	0.169492	0.693220
16	13.000000	180.000000	270.000000	0.048148	0.666667
19	27.600000	264.500000	374.600000	0.073679	0.706086

Как я и предполагал, для разных по количеству комнат квартир, отношения получились разными.
Так как для квартир с 0 комнатами нет информации о площади кухни, пропуски в `kitchen_ratio` заменим на отношение средней площади кухни к средней общей площади по всему датафреймк

In [11]:
```
#Заменим пропуски в строках с количеством комант 0 на среднее по всему датафрейму
kitchen_ratio = df['kitchen_area'].mean() / df['total_area'].mean()
living_ratio = df['living_area'].mean() / df['total_area'].mean()
df['kitchen_ratio'] = df['kitchen_ratio'].fillna(kitchen_ratio)
df['living_ratio'] = df['living_ratio'].fillna(living_ratio)
```

Теперь мы можем заменить пропуски в площадях кухни и в жилой площади умножением отношения на общую площадь

In [12]:
```
df['kitchen_area'] = df['kitchen_area'].fillna(df['kitchen_ratio'] * df['total_area'])
df['living_area'] = df['living_area'].fillna(df['living_ratio'] * df['total_area'])
```

In [13]:
```
df[df['kitchen_area'] + df['living_area'] > df['total_area']]
```

Out[13]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total
184	7	2600000.0	30.20	2018-02-14T00:00:00	1	2.65	9.0
424	14	11500000.0	98.00	2016-05-10T00:00:00	3	2.65	3.0
440	8	2480000.0	27.11	2018-03-12T00:00:00	0	2.65	17.0
545	9	4700000.0	23.80	2018-12-28T00:00:00	1	2.65	18.0
551	8	3100000.0	31.59	2018-03-08T00:00:00	1	2.70	19.0
...	...	...	...	...	...	...	...
22246	6	3100000.0	27.30	2018-05-29T00:00:00	0	2.70	16.0
22680	4	2100000.0	23.60	2016-02-25T00:00:00	1	2.75	25.0
22907	9	65000000.0	228.00	2016-06-02T00:00:00	4	2.65	5.0
23191	3	1900000.0	18.90	2016-04-04T00:00:00	1	2.65	16.0
23202	13	4919880.0	67.92	2018-10-23T00:00:00	2	2.65	3.0

138 rows × 24 columns

In [14]:
```
df.isnull().mean()
```

Out[14]:
```
total_images            0.000000
last_price              0.000000
total_area              0.000000
first_day_exposition    0.000000
rooms                   0.000000
ceiling_height          0.000000
floors_total            0.000000
living_area             0.000000
floor                   0.000000
is_apartment            0.000000
studio                  0.000000
open_plan               0.000000
kitchen_area            0.000000
balcony                 0.000000
locality_name           0.000000
airports_nearest        0.234415
cityCenters_nearest     0.233439
parks_around3000        0.000000
parks_nearest           0.659240
ponds_around3000        0.000000
ponds_nearest           0.616550
days_exposition         0.000000
kitchen_ratio           0.000000
living_ratio            0.000000
dtype: float64
```

С пропусками разобрались, теперь посмотрим на дубли.

In [15]:
```
df.duplicated().sum()
```

Out[15]:
```
0
```

Явных дубликатов нет. Это не может не радовать! Но попробуем поискать неявные дубликаты. Взглянем на обновленный датафрейм.

In [16]:
```
df.head()
```

Out[16]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.70	16.0	51.0
1	7	3350000.0	40.4	2018-12-04T00:00:00	1	2.65	11.0	18.6
2	10	5196000.0	56.0	2015-08-20T00:00:00	2	2.65	5.0	34.3
3	0	64900000.0	159.0	2015-07-24T00:00:00	3	2.65	14.0	96.9
4	2	10000000.0	100.0	2018-06-19T00:00:00	2	3.03	14.0	32.0

В названии населенного пункта могут встретиться неявные дубликаты. Ознакомимся с данными этой колонки поближе

In [17]:
```
df['locality_name'].value_counts()
```

Out[17]:
```
Санкт-Петербург                  15651
посёлок Мурино                     520
посёлок Шушары                     439
Всеволожск                         398
Пушкин                             369
                                   ...
поселок Каложицы                     1
посёлок Платформа 69-й километр      1
поселок Почап                        1
поселок Гончарово                    1
посёлок Дзержинского                 1
Name: locality_name, Length: 364, dtype: int64
```

365 населенных пунктов. Но больше половины объявлений, что логично, в Санкт-Петербурге.

Пробежимся глазами по списку населенных пунктов.

In [18]:
```
sorted(df['locality_name'].astype('str').unique())
```

Out[18]: ['Бокситогорск',
'Волосово',
'Волхов',
'Всеволожск',
'Выборг',
'Высоцк',
'Гатчина',
'Зеленогорск',
'Ивангород',
'Каменногорск',
'Кингисепп',
'Кириши',
'Кировск',
'Колпино',
'Коммунар',
'Красное Село',
'Кронштадт',
'Кудрово',
'Лодейное Поле',
'Ломоносов',
'Луга',
'Любань',
'Мурино',
'Никольское',
'Новая Ладога',
'Отрадное',
'Павловск',
'Петергоф',
'Пикалёво',
'Подпорожье',
'Приморск',
'Приозерск',
'Пушкин',
'Санкт-Петербург',
'Светогорск',
'Сертолово',
'Сестрорецк',
'Сланцы',
'Сосновый Бор',
'Сясьстрой',
'Тихвин',
'Тосно',
'Шлиссельбург',
'городской поселок Большая Ижора',
'городской поселок Янино-1',
'городской посёлок Будогощь',
'городской посёлок Виллози',
'городской посёлок Лесогорский',
'городской посёлок Мга',
'городской посёлок Назия',
'городской посёлок Новоселье',
'городской посёлок Павлово',
'городской посёлок Рощино',
'городской посёлок Свирьстрой',
'городской посёлок Советский',
'городской посёлок Фёдоровское',
'городской посёлок Янино-1',
'деревня Агалатово',
'деревня Аро',
'деревня Батово',
'деревня Бегуницы',
'деревня Белогорка',
'деревня Большая Вруда',
'деревня Большая Пустомержа',
'деревня Большие Колпаны',

'деревня Большое Рейзино',
'деревня Большой Сабск',
'деревня Бор',
'деревня Борисова Грива',
'деревня Ваганово',
'деревня Вартемяги',
'деревня Вахнова Кара',
'деревня Выскатка',
'деревня Гарболово',
'деревня Глинка',
'деревня Горбунки',
'деревня Гостилицы',
'деревня Заклинье',
'деревня Заневка',
'деревня Зимитицы',
'деревня Извара',
'деревня Иссад',
'деревня Калитино',
'деревня Кальтино',
'деревня Камышовка',
'деревня Каськово',
'деревня Келози',
'деревня Кипень',
'деревня Кисельня',
'деревня Колтуши',
'деревня Коркино',
'деревня Котлы',
'деревня Кривко',
'деревня Кудрово',
'деревня Кузьмолово',
'деревня Курковицы',
'деревня Куровицы',
'деревня Куттузи',
'деревня Лаврики',
'деревня Лаголово',
'деревня Лампово',
'деревня Лесколово',
'деревня Лопухинка',
'деревня Лупполово',
'деревня Малая Романовка',
'деревня Малое Верево',
'деревня Малое Карлино',
'деревня Малые Колпаны',
'деревня Мануйлово',
'деревня Меньково',
'деревня Мины',
'деревня Мистолово',
'деревня Ненимяки',
'деревня Нижние Осельки',
'деревня Нижняя',
'деревня Низино',
'деревня Новое Девяткино',
'деревня Новолисино',
'деревня Нурма',
'деревня Оржицы',
'деревня Парицы',
'деревня Пельгора',
'деревня Пеники',
'деревня Пижма',
'деревня Пикколово',
'деревня Пудомяги',
'деревня Пустынка',
'деревня Пчева',
'деревня Рабитицы',
'деревня Разбегаево',

```
'деревня Раздолье',
'деревня Разметелево',
'деревня Рапполово',
'деревня Реброво',
'деревня Русско',
'деревня Сижно',
'деревня Снегирёвка',
'деревня Старая',
'деревня Старая Пустошь',
'деревня Старое Хинколово',
'деревня Старополье',
'деревня Старосиверская',
'деревня Старые Бегуницы',
'деревня Суоранда',
'деревня Сяськелево',
'деревня Тарасово',
'деревня Терпилицы',
'деревня Тихковицы',
'деревня Тойворово',
'деревня Торосово',
'деревня Торошковичи',
'деревня Трубников Бор',
'деревня Фалилеево',
'деревня Фёдоровское',
'деревня Хапо-Ое',
'деревня Хязельки',
'деревня Чудской Бор',
'деревня Шпаньково',
'деревня Щеглово',
'деревня Юкки',
'деревня Ялгино',
'деревня Яльгелево',
'деревня Ям-Тесово',
'коттеджный поселок Кивеннапа Север',
'коттеджный поселок Счастье',
'коттеджный посёлок Лесное',
'поселок Аннино',
'поселок Барышево',
'поселок Бугры',
'поселок Возрождение',
'поселок Войсковицы',
'поселок Володарское',
'поселок Гаврилово',
'поселок Гарболово',
'поселок Гладкое',
'поселок Глажево',
'поселок Глебычево',
'поселок Гончарово',
'поселок Громово',
'поселок Дружноселье',
'поселок Елизаветино',
'поселок Жилгородок',
'поселок Жилпосёлок',
'поселок Житково',
'поселок Заводской',
'поселок Запорожское',
'поселок Зимитицы',
'поселок Ильичёво',
'поселок Калитино',
'поселок Каложицы',
'поселок Кингисеппский',
'поселок Кирпичное',
'поселок Кобралово',
'поселок Кобринское',
'поселок Коммунары',
```

```
'поселок Коробицыно',
'поселок Котельский',
'поселок Красная Долина',
'поселок Красносельское',
'поселок Лесное',
'поселок Лисий Нос',
'поселок Лукаши',
'поселок Любань',
'поселок Мельниково',
'поселок Мичуринское',
'поселок Молодцово',
'поселок Мурино',
'поселок Новый Свет',
'поселок Новый Учхоз',
'поселок Оредеж',
'поселок Пансионат Зелёный Бор',
'поселок Первомайское',
'поселок Перово',
'поселок Петровское',
'поселок Победа',
'поселок Поляны',
'поселок Почап',
'поселок Починок',
'поселок Пушное',
'поселок Пчевжа',
'поселок Рабитицы',
'поселок Романовка',
'поселок Ромашки',
'поселок Рябово',
'поселок Севастьяново',
'поселок Селезнёво',
'поселок Сельцо',
'поселок Семиозерье',
'поселок Семрино',
'поселок Серебрянский',
'поселок Совхозный',
'поселок Старая Малукса',
'поселок Стеклянный',
'поселок Сумино',
'поселок Суходолье',
'поселок Тельмана',
'поселок Терволово',
'поселок Торковичи',
'поселок Тёсово-4',
'поселок Углово',
'поселок Усть-Луга',
'поселок Ушаки',
'поселок Цвелодубово',
'поселок Цвылёво',
'поселок городского типа Большая Ижора',
'поселок городского типа Вырица',
'поселок городского типа Дружная Горка',
'поселок городского типа Дубровка',
'поселок городского типа Ефимовский',
'поселок городского типа Кондратьево',
'поселок городского типа Красный Бор',
'поселок городского типа Кузьмоловский',
'поселок городского типа Лебяжье',
'поселок городского типа Лесогорский',
'поселок городского типа Назия',
'поселок городского типа Никольский',
'поселок городского типа Приладожский',
'поселок городского типа Рахья',
'поселок городского типа Рощино',
'поселок городского типа Рябово',
```

```
'поселок городского типа Синявино',
'поселок городского типа Советский',
'поселок городского типа Токсово',
'поселок городского типа Форносово',
'поселок городского типа имени Свердлова',
'поселок станции Вещево',
'поселок станции Корнево',
'поселок станции Лужайка',
'поселок станции Приветнинское',
'посёлок Александровская',
'посёлок Алексеевка',
'посёлок Аннино',
'посёлок Белоостров',
'посёлок Бугры',
'посёлок Возрождение',
'посёлок Войскорово',
'посёлок Высокоключевой',
'посёлок Гаврилово',
'посёлок Дзержинского',
'посёлок Жилгородок',
'посёлок Ильичёво',
'посёлок Кикерино',
'посёлок Кобралово',
'посёлок Коробицыно',
'посёлок Левашово',
'посёлок Ленинское',
'посёлок Лисий Нос',
'посёлок Мельниково',
'посёлок Металлострой',
'посёлок Мичуринское',
'посёлок Молодёжное',
'посёлок Мурино',
'посёлок Мыза-Ивановка',
'посёлок Новогорелово',
'посёлок Новый Свет',
'посёлок Пансионат Зелёный Бор',
'посёлок Парголово',
'посёлок Перово',
'посёлок Песочный',
'посёлок Петро-Славянка',
'посёлок Петровское',
'посёлок Платформа 69-й километр',
'посёлок Плодовое',
'посёлок Плоское',
'посёлок Победа',
'посёлок Поляны',
'посёлок Понтонный',
'посёлок Пригородный',
'посёлок Пудость',
'посёлок Репино',
'посёлок Ропша',
'посёлок Сапёрное',
'посёлок Сапёрный',
'посёлок Сосново',
'посёлок Старая Малукса',
'посёлок Стеклянный',
'посёлок Стрельна',
'посёлок Суйда',
'посёлок Сумино',
'посёлок Тельмана',
'посёлок Терволово',
'посёлок Торфяное',
'посёлок Усть-Ижора',
'посёлок Усть-Луга',
'посёлок Форт Красная Горка',
```

```
 'посёлок Шугозеро',
 'посёлок Шушары',
 'посёлок Щеглово',
 'посёлок городского типа Важины',
 'посёлок городского типа Вознесенье',
 'посёлок городского типа Вырица',
 'посёлок городского типа Красный Бор',
 'посёлок городского типа Кузнечное',
 'посёлок городского типа Кузьмоловский',
 'посёлок городского типа Лебяжье',
 'посёлок городского типа Мга',
 'посёлок городского типа Павлово',
 'посёлок городского типа Рощино',
 'посёлок городского типа Рябово',
 'посёлок городского типа Сиверский',
 'посёлок городского типа Тайцы',
 'посёлок городского типа Токсово',
 'посёлок городского типа Ульяновка',
 'посёлок городского типа Форносово',
 'посёлок городского типа имени Морозова',
 'посёлок городского типа имени Свердлова',
 'посёлок при железнодорожной станции Вещево',
 'посёлок при железнодорожной станции Приветнинское',
 'посёлок станции Громово',
 'посёлок станции Свирь',
 'садоводческое некоммерческое товарищество Лесная Поляна',
 'садовое товарищество Новая Ропша',
 'садовое товарищество Приладожский',
 'садовое товарищество Рахья',
 'садовое товарищество Садко',
 'село Копорье',
 'село Никольское',
 'село Павлово',
 'село Паша',
 'село Путилово',
 'село Рождествено',
 'село Русско-Высоцкое',
 'село Старая Ладога',
 'село Шум']
```

Есть некоторые неявные дубликаты. В основном между посёлок и поселок. Но их количество, как правило по одному, а это ничтожно мало в объеме общих данных. Не будет тратить на наих время.

Теперь разберемся с типами данных.
Во-первых предлагаю преобразовать дату и время в колонке `first_day_exposition`

In [19]:
```
df['first_day_exposition'] = pd.to_datetime(df['first_day_exposition'], format='%Y-%m
```

Также переведем формат колонки `floors_total` в целочисленный

In [20]:
```
df['floors_total'] = df['floors_total'].astype('int')
```

Дату привели в нужный формат. Теперь для экономии памяти предлагаю заменить int64 и float64 на int32 и float32 соответсвенно

In [21]:
```
for column in df.columns:
    if df.dtypes[column] == 'int64':
        df[column] = df[column].astype('int32')
    elif df.dtypes[column] == 'float64':
        df[column] = df[column].astype('float32')
# цикл пробегается по колонкам датафрейма и меняет тип данных в соответствии с услови
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23565 entries, 0 to 23698
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   total_images          23565 non-null  int32         
 1   last_price            23565 non-null  float32       
 2   total_area            23565 non-null  float32       
 3   first_day_exposition  23565 non-null  datetime64[ns]
 4   rooms                 23565 non-null  int32         
 5   ceiling_height        23565 non-null  float32       
 6   floors_total          23565 non-null  int32         
 7   living_area           23565 non-null  float32       
 8   floor                 23565 non-null  int32         
 9   is_apartment          23565 non-null  bool          
 10  studio                23565 non-null  bool          
 11  open_plan             23565 non-null  bool          
 12  kitchen_area          23565 non-null  float32       
 13  balcony               23565 non-null  int32         
 14  locality_name         23565 non-null  object        
 15  airports_nearest      18041 non-null  float32       
 16  cityCenters_nearest   18064 non-null  float32       
 17  parks_around3000      23565 non-null  int32         
 18  parks_nearest         8030 non-null   float32       
 19  ponds_around3000      23565 non-null  int32         
 20  ponds_nearest         9036 non-null   float32       
 21  days_exposition       23565 non-null  int32         
 22  kitchen_ratio         23565 non-null  float32       
 23  living_ratio          23565 non-null  float32       
dtypes: bool(3), datetime64[ns](1), float32(11), int32(8), object(1)
memory usage: 2.3+ MB
```

Отлично! Память сэкономили

Вывод по предобработке

После выполнения предобрабокти данных, нам удалось избавиться от большинства пропусков в данных. С некоторыми данными пришлосьпопращаться вовсе.
Данные, полученные картографическим путем заменить не удалось.

Расчёты и добавление результатов в таблицу

Добавим в датафрейм колонку с ценой за метр квадратный. Для удобства добавим новую колонку полсе колонки с общей площадью. Также предлагаю сразу округлить данное значение

In [23]:
```
df.insert(3, 'price_per_meter', round(df['last_price']/df['total_area']))
```

Добавим день недели, месяц и год в датафрейм. Добалю после даты для удобства

In [24]:
```
#Функция для преобразования числовго дня в привычный
def correct_days(day):
    if day == 0:
        return 'monday'
    elif day == 1:
        return 'Tuesday'
    elif day == 2:
        return 'Wednesday'
```

```
    elif day == 3:
        return 'Thursday'
    elif day == 4:
        return 'Friday'
    elif day == 5:
        return 'Saturday'
    elif day == 6:
        return 'Sunday'
# Функция для преобразования числовго месяца в привычный
def correct_month(month):
    if month == 1:
        return 'January'
    elif month == 2:
        return 'February'
    elif month == 3:
        return 'March'
    elif month == 4:
        return 'April'
    elif month == 5:
        return 'May'
    elif month == 6:
        return 'June'
    elif month == 7:
        return 'July'
    elif month == 8:
        return 'August'
    elif month == 9:
        return 'September'
    elif month == 10:
        return 'October'
    elif month == 11:
        return 'November'
    elif month == 12:
        return 'December'
```

In [25]:
```
df.insert(5, 'day', df['first_day_exposition'].dt.weekday.apply(correct_days))
df.insert(6, 'month', df['first_day_exposition'].dt.month.apply(correct_month))
df.insert(7, 'year', df['first_day_exposition'].dt.year)
```

Добавим категоризацию этажа квартиры: первый, послдений, другое.

In [26]:
```
def category_floor(row):
    if row['floor'] == 1:
        return 'first'
    elif row['floor'] == row['floors_total']:
        return 'last'
    else:
        return 'other'
```

In [27]:
```
df.insert(13, 'floor_category', df.apply(category_floor, axis=1))
```

Соотношение жилой и общей площади, а также отношение площади кухни к общей мы добавили на предыдущем этапе.

In [28]:
```
df.head()
```

Out[28]:

	total_images	last_price	total_area	price_per_meter	first_day_exposition	day	month	y
0	20	13000000.0	108.000000	120370.0	2019-03-07	Thursday	March	2
1	7	3350000.0	40.400002	82921.0	2018-12-04	Tuesday	December	2
2	10	5196000.0	56.000000	92786.0	2015-08-20	Thursday	August	2
3	0	64900000.0	159.000000	408176.0	2015-07-24	Friday	July	2
4	2	10000000.0	100.000000	100000.0	2018-06-19	Tuesday	June	2

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23565 entries, 0 to 23698
Data columns (total 29 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   total_images          23565 non-null  int32         
 1   last_price            23565 non-null  float32       
 2   total_area            23565 non-null  float32       
 3   price_per_meter       23565 non-null  float32       
 4   first_day_exposition  23565 non-null  datetime64[ns]
 5   day                   23565 non-null  object        
 6   month                 23565 non-null  object        
 7   year                  23565 non-null  int64         
 8   rooms                 23565 non-null  int32         
 9   ceiling_height        23565 non-null  float32       
 10  floors_total          23565 non-null  int32         
 11  living_area           23565 non-null  float32       
 12  floor                 23565 non-null  int32         
 13  floor_category        23565 non-null  object        
 14  is_apartment          23565 non-null  bool          
 15  studio                23565 non-null  bool          
 16  open_plan             23565 non-null  bool          
 17  kitchen_area          23565 non-null  float32       
 18  balcony               23565 non-null  int32         
 19  locality_name         23565 non-null  object        
 20  airports_nearest      18041 non-null  float32       
 21  cityCenters_nearest   18064 non-null  float32       
 22  parks_around3000      23565 non-null  int32         
 23  parks_nearest         8030 non-null   float32       
 24  ponds_around3000      23565 non-null  int32         
 25  ponds_nearest         9036 non-null   float32       
 26  days_exposition       23565 non-null  int32         
 27  kitchen_ratio         23565 non-null  float32       
 28  living_ratio          23565 non-null  float32       
dtypes: bool(3), datetime64[ns](1), float32(12), int32(8), int64(1), object(4)
memory usage: 3.1+ MB
```

Выводы по расчётам

На данном этапе мы добавили в датафрейм новые данные, которые понадобятся нам в дальнейшем для анализа данных. Добавили день, месяц и год размещения объявления, цену за квадратный метр. Уже ранее были добавлены отношения жилой и площади к кухни к общей

Исследовательский анализ данных

Площадь, цена, количество комнат, высота потолков

Теперь можно приступать к анализу данных. Для начала предлагаю избавиться от явно ненужных для нашего проекта столбцов.

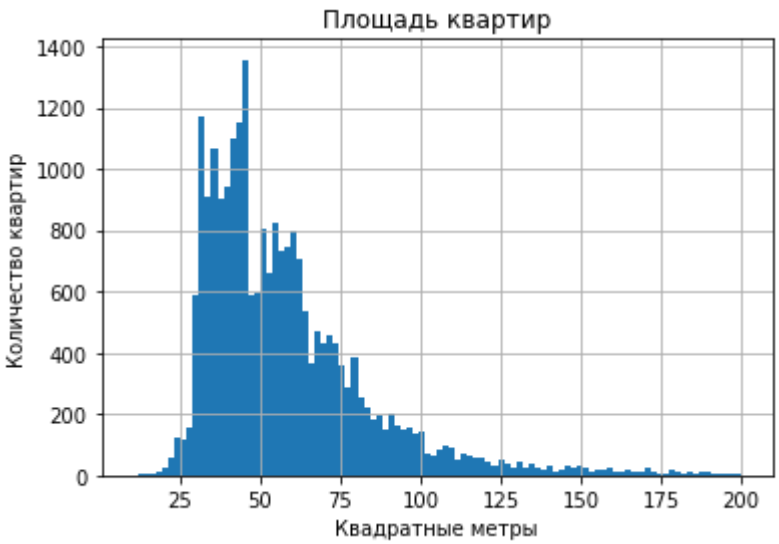
In [30]:
```
df = df.drop(columns=['total_images', 'floors_total', 'floor', 'parks_around3000',
         'is_apartment', 'studio', 'open_plan', 'balcony',
         'parks_nearest', 'ponds_around3000', 'ponds_nearest',])
```

Изучим следующие параметры: площадь, цена, число комнат, высота потолков. Постройте гистограммы для каждого параметра.

Гистограмма площади квартир.

In [31]:
```
# Построим гистограмму площади квартир
plt.hist(df['total_area'], bins=100, range=(10,200))
plt.title('Площадь квартир')
plt.xlabel('Квадратные метры')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

# Выведем количество квартира с максимальной площадью
print('Наибольшие по площади варианты:')
display((df
    .pivot_table(index='total_area', values='last_price', aggfunc='count')
    .sort_values('total_area', ascending=False)
    .rename(columns={'last_price': 'count'})
    .head(10)
))
# Статистический анализ
df['total_area'].describe()
```




```
Наибольшие по площади варианты:
```

total_area	count
900.000000	1
631.200012	1
631.000000	1
618.000000	1
590.000000	1
517.000000	1
507.000000	1
500.000000	2
495.000000	1
494.100006	1

Out[31]:
```
count    23565.000000
mean        60.322979
std         35.657131
min         12.000000
25%         40.000000
50%         52.000000
75%         69.699997
max        900.000000
Name: total_area, dtype: float64
```

Пик гистограммы приходится на квартиры с площадбю около 45м2. Можно заметить, что большинство квартир с площадью 30-60м2. Квартир с площадью более 100м2 намного меньше. Также вы видим, что есть квартиры с очень большой площадью. Возможно, это выбросы. Разберемся с этим позже.

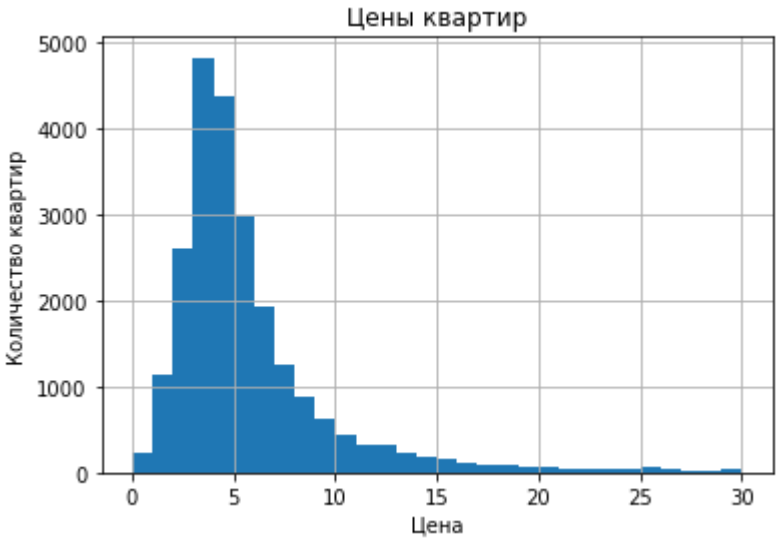
Гистограмма цен на квартиры

Поскольку цена указана точная, а суммы достаточно велики, построим гистограмму по стоимости в млн, округленной до 3х знаков. Так будет удобнее работать с гистограммой

In [32]:
```
#Добавим колонку с округленной ценой для удобства работы.
df['price_round'] = round(df['last_price'] / 1000000 ,3)
# Построим гистограмму стоимости квартир
plt.hist(df['price_round'], bins=30, range=(0, 30))
plt.title('Цены квартир')
plt.xlabel('Цена')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

#квартиры с очень высокими ценами
print('Наибольшие по цене варианты:')
display((df
    .pivot_table(index='last_price', values='total_area', aggfunc='count')
    .sort_values('last_price', ascending=False)
    .rename(columns={'total_area': 'count'})
    .head(10)
))

# Статистический анализ
df['last_price'].describe()
```



Наибольшие по цене варианты:

last_price	count
763000000.0	1
420000000.0	1
401300000.0	1
330000000.0	1
300000000.0	1
289238400.0	1
245000000.0	1
240000000.0	1
230000000.0	1
190870000.0	1

Out[32]:
```
count        23565.0
mean       6540059.0
std       10910923.0
min          12190.0
25%        3400000.0
50%        4646000.0
75%        6790000.0
max      763000000.0
Name: last_price, dtype: float64
```

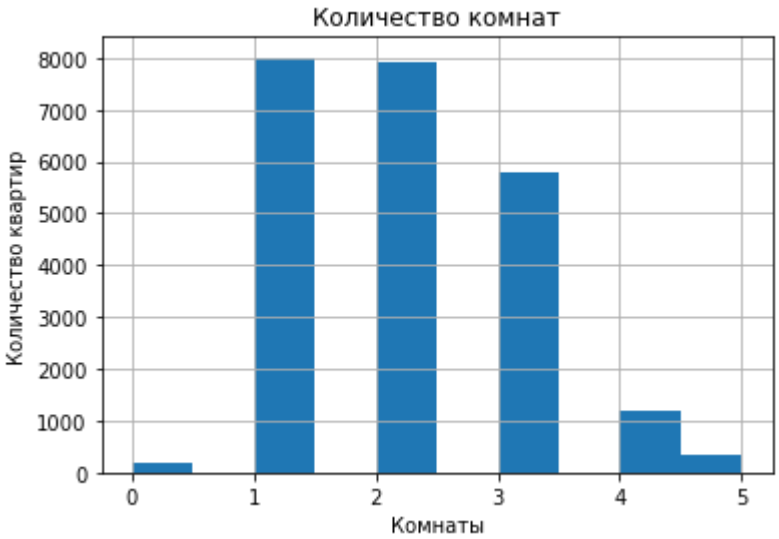
Пик приходится на отметку около 4,5 млн. Выше 10 млн квартир заметно меньше, а выше 20млн еще меньше. Также есть квартиры с очень большой стоимостью. Это очень похоже на выбросы. Будем разбираться и с этим.

Гистограмма по количеству комнат

Построим гистограмму по количеству комнат.

In [33]:
```
plt.hist(df['rooms'], bins=10, range=(0, 5))
plt.title('Количество комнат')
plt.xlabel('Комнаты')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
#квартиры с очень большим количество комнат
print('Наибольшее количество комнат:')
```

```
display((df
    .pivot_table(index='rooms', values='total_area', aggfunc='count')
    .sort_values('rooms', ascending=False)
    .rename(columns={'total_area': 'count'})
    .head(10)
))
# Статистический анализ
df['rooms'].describe()
```



Наибольшее количество комнат:

rooms	count
19	1
16	1
15	1
14	2
12	1
11	2
10	3
9	8
8	12
7	59

Out[33]:
```
count    23565.000000
mean         2.070656
std          1.078591
min          0.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         19.000000
Name: rooms, dtype: float64
```

Большинство квартир с 2 - 3 комнатами. Квартир с более чем 6 комнатами практически нет. Но есть и объявления с более чем 10 комнатами. С выбросами разберемся позже.

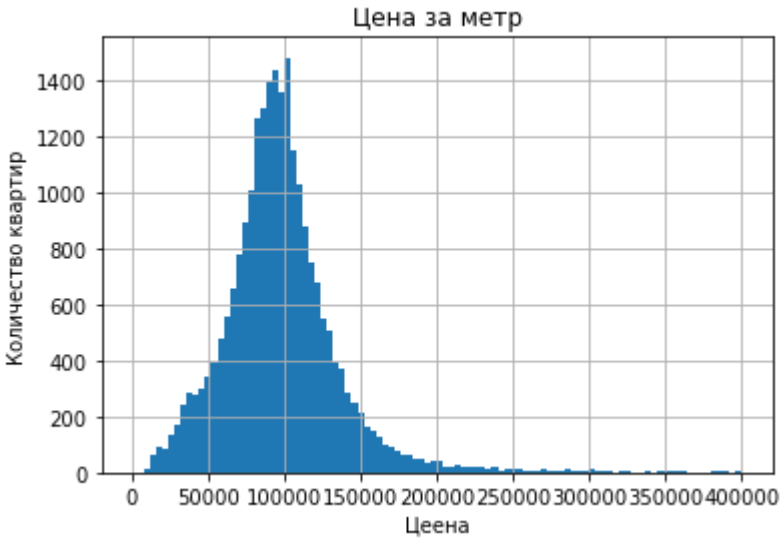
Гистрограмма цены на м2

Так же, я бы изучил цены за квадратный метр жилья

In [34]:
```
plt.hist(df['price_per_meter'], bins=100, range=(0, 400000))
```

```
plt.title('Цена за метр')
plt.xlabel('Цеена')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

#Статистический анализ
df['price_per_meter'].describe()
```



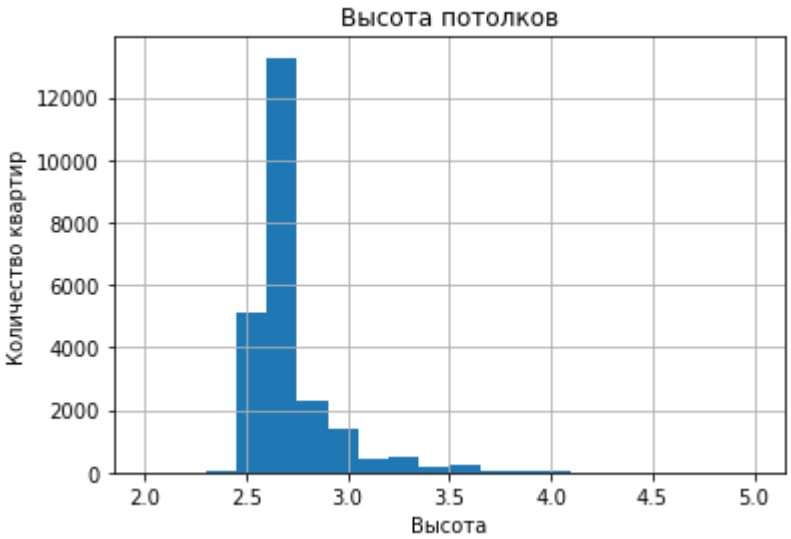
Out[34]:
```
count    2.356500e+04
mean     9.940578e+04
std      5.038940e+04
min      1.120000e+02
25%      7.656600e+04
50%      9.500000e+04
75%      1.142130e+05
max      1.907500e+06
Name: price_per_meter, dtype: float64
```

Первое нормальное распредление за сегодня! Пик пришелся на отметку в 100 тыс. рублей. Квартиры со стоимость за метр более 250 тыс. рублей встречаются редко. Медиана на отметке 95 тыс рублей

Гистограмма по высоте потолков

Построим гистограмму по высоте потолков

In [35]:
```
plt.hist(df['ceiling_height'], bins=20, range=(2, 5))
plt.title('Высота потолков')
plt.xlabel('Высота')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
#квартиры с очень большой высотой потолков
print('Квартиры с наибольшей высотой потолков:')
display((df
    .pivot_table(index='ceiling_height', values='total_area', aggfunc='count')
    .sort_values('ceiling_height', ascending=False)
    .rename(columns={'total_area': 'count'})
    .head(10)
))
# Статистический анализ
df['ceiling_height'].describe()
```



Квартиры с наибольшей высотой потолков:

ceiling_height	count
100.0	1
32.0	2
27.5	1
27.0	8
26.0	1
25.0	7
24.0	1
22.6	1
20.0	1
14.0	1

Out[35]:
```
count    23565.000000
mean         2.724316
std          0.991057
min          1.000000
25%          2.600000
50%          2.650000
75%          2.700000
max        100.000000
Name: ceiling_height, dtype: float64
```

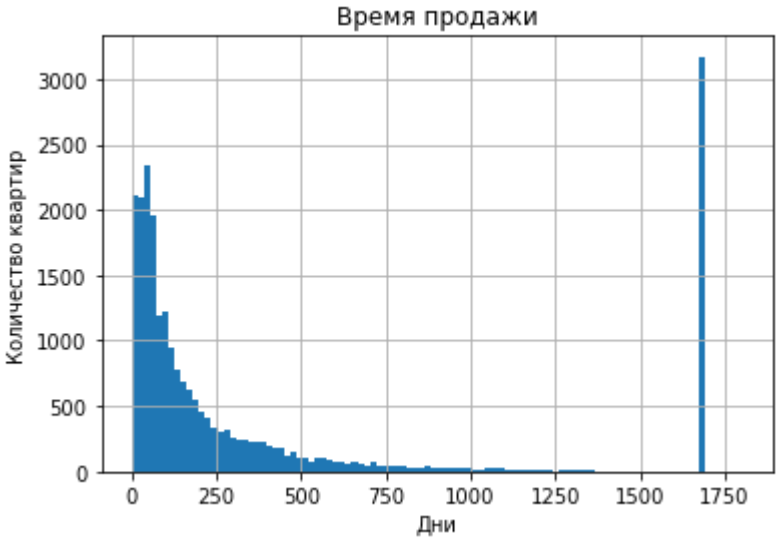
Пик приходится на квартиры с высотой около 2,6 метров. Но как мы прекрасно видим, есть квартиры с нереальной высотой потолков. 100 метров. Или 30 метров. А как лампочки в люстре поменять? :) Также есть потолки высотой менее 2 метров. Но как и в предыдущих случаях, выбросы оставим на потом.

Изучение времени продажи квартир

Большинство квартир имеют высоту потолков в диапозоне от 2,6 до 2,7 метров. Пик приходится на квартиры с высотой потолков 2,65 метров. Медиана получилась 2,65 метров

Изучим время продажи квартир. построим гистограмму и проанализируем ее. Не стоит забывать, что мы заменили пропуски в значениях максимальной продолжительностью + 100 дней. Соответсвенно на гррафике должна быть видна отдельная группа.

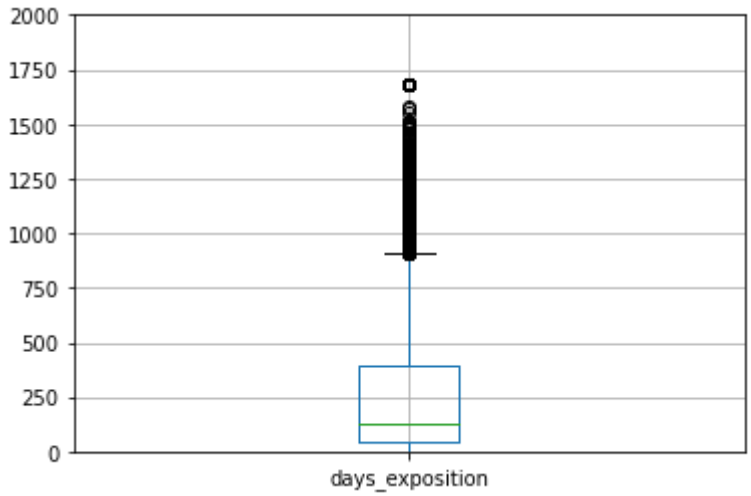
In [36]:
```
plt.hist(df['days_exposition'], bins=100, range=(0, 1800))
plt.title('Время продажи')
plt.xlabel('Дни')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
```



Как и предполагалось, большое количество квартир выделилось в отдельную группу справа. Мы приняли за истину то, что данные квартиры скорее всего не были проданы на момент сбора данных для нашего проекта. Также, в данных наверняка есть выбросы. Построим диаграмму размаха и избавимся от выбросов.

In [37]:
```
print('Диаграмма размаха:')
df.boxplot(column='days_exposition')
plt.ylim(0, 2000)
plt.show()
print('Статистический анализ:')
print(df['ceiling_height'].describe())
```

```
Диаграмма размаха:
```

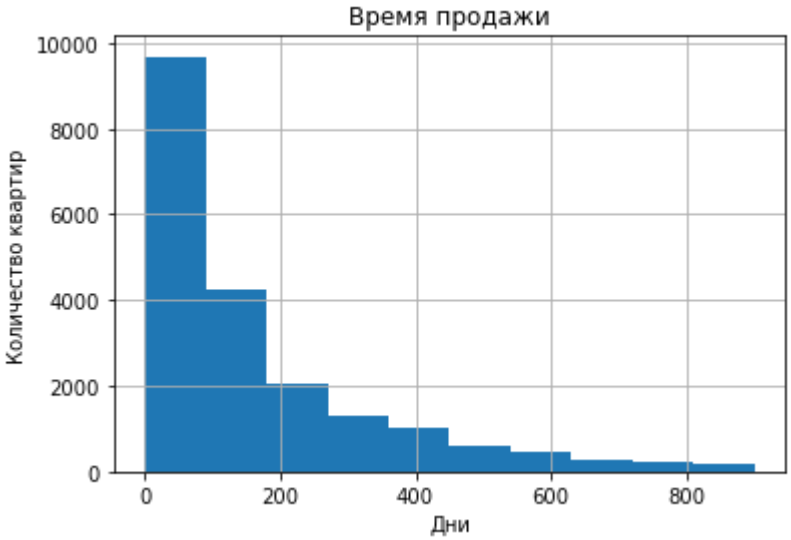



```
Статистический анализ:
count    23565.000000
mean         2.724316
std          0.991057
min          1.000000
25%          2.600000
50%          2.650000
75%          2.700000
max        100.000000
Name: ceiling_height, dtype: float64
```

Наш график указывает на то, что значения выше чем 900 дней являются выбросами.

Избавимся от них и построим гистограмму по "чистым" данным

In [38]:
```
# Избавимся о выбросов
good_days = df.query('days_exposition <= 900')
# Строим гистограмму
plt.hist(good_days['days_exposition'], bins=10, range=(0, 900))
plt.title('Время продажи')
plt.xlabel('Дни')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
# Сатистический анализ
print('Сатистический анализ:')
good_days['days_exposition'].describe()
```

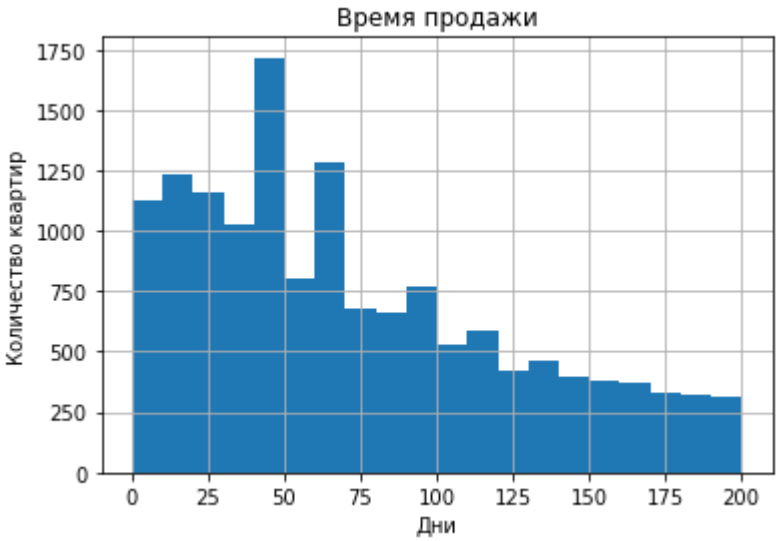



```
Сатистический анализ:
```

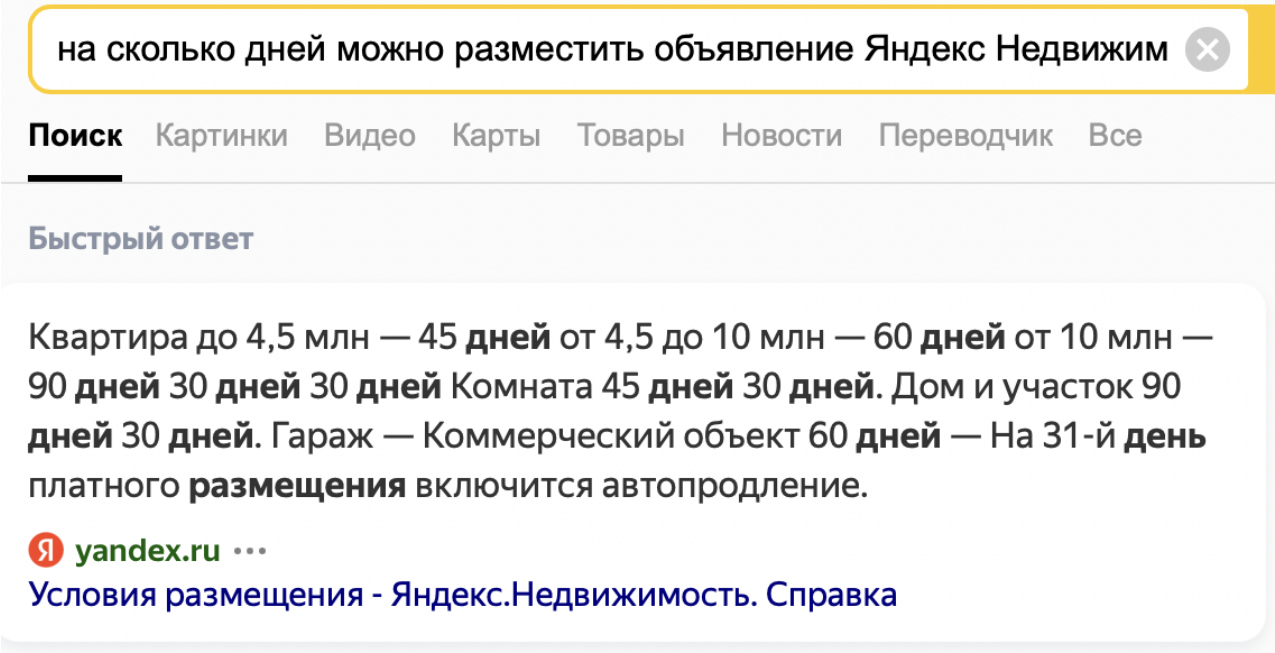
Out[38]:
```
count    19989.000000
mean       162.155185
std        177.203832
min          1.000000
25%         44.000000
50%         92.000000
75%        217.000000
max        900.000000
Name: days_exposition, dtype: float64
```

На гистограмме мы видим, что большинство квартир продаются в период от 0 до 100 дней. Но это слишком большой промежуток. Изменим диапозон построения гистограммы и количество корзин для более наглядного результата.

In [39]:
```
plt.hist(good_days['days_exposition'], bins=20, range=(0, 200))
plt.title('Время продажи')
plt.xlabel('Дни')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
```



Мы видим несколько явных пиков. Первый на отметке 45 дней, второй 65 дней. Медиана расположилась на отметке в 92 дня. Также есть несколько квартир, которые продали за 1 день.
Еще необходимо обратить внимание на то, как сильно отличается среднее от медианного значения. 162 против 92 дней. Медиана лучше описывает ситуацию, тк не подвержена выбросам.
В основном, квартиры продаются в срок до 100 дней.
Мне стало интересно разобраться с природой появления пиков на отметках 45 и 60 дней. Я предположил, что данные всплески связаны с условием размещения объявлений на сервисе Яндекс.Нездвижимость. Возможно, бесплатно объявления можно разместить как раз на 45 или 60 дней. Я вбил интересующий запрос в поисковую строку и уже готовился потирать ручки. Первый же результат подтвердил мою теорию:



Но стоило мне пройти по ссылке, как радость сменилась отчаянием... Данные условия распространяются на регионы, а для Москвы и Санкт-Петербурга условия иные:

Сроки публикации

▼ В Москве, Санкт-Петербурге и их областях

Тип сделки	Бесплатное объявление	Платное объявление
Продажа	120 дней	30 дней
Аренда длительная	90 дней	
Аренда посуточная	120 дней	

Можно предположить, что на момент сбора данных, для Санкт-Петербурга были иные условия по публикации бесплатных объявлений. Нужно это проверить. Посмотрим на цены объявлений, которые сняли на 45й день

In [40]:
```
df[df['days_exposition'] == 45]['last_price'].describe()
```

Out[40]:
```
count    8.790000e+02
mean     3.318170e+06
std      4.291974e+06
min      4.400000e+05
25%      2.400000e+06
50%      3.250000e+06
75%      3.900000e+06
max      1.240000e+08
Name: last_price, dtype: float64
```

Всего 879 объявлений. Есть объявления и выше 4,5 млн, которые не подходят к условиям размещения. Но тут уже можно предположить, что эти квартиры как раз и были сняты на 45й день автором, а не окончанием бесплатного срока размещения. Посмотрим сколько таких объявлений

In [41]:
```
len(df[(df['days_exposition'] == 45) & (df['last_price'] >= 4500000)])
```

Out[41]:
```
50
```

Всего 50 объявлений. Но какое количество объявлений, стоимость квартир в которых менее 4,5 млн были реально закрыты на 45й день, а не просто сняты по истечению бесплатного срока? Сказать практически невозможно.

Предлагаю также проверить гипотезу "бесплатный срок объявлений от 4,5 до 10 млн рублей - 60 дней" Проведем аналогичные манипуляции

In [42]:
```
df[df['days_exposition'] == 60]['last_price'].describe()
```

Out[42]:
```
count    5.380000e+02
mean     6.025422e+06
std      1.679605e+06
min      1.200000e+06
25%      4.900000e+06
50%      5.776000e+06
75%      6.950000e+06
max      1.390000e+07
Name: last_price, dtype: float64
```

In [43]:
```
df[(df['days_exposition'] == 60) & (df['last_price'] >= 10000000)]
```

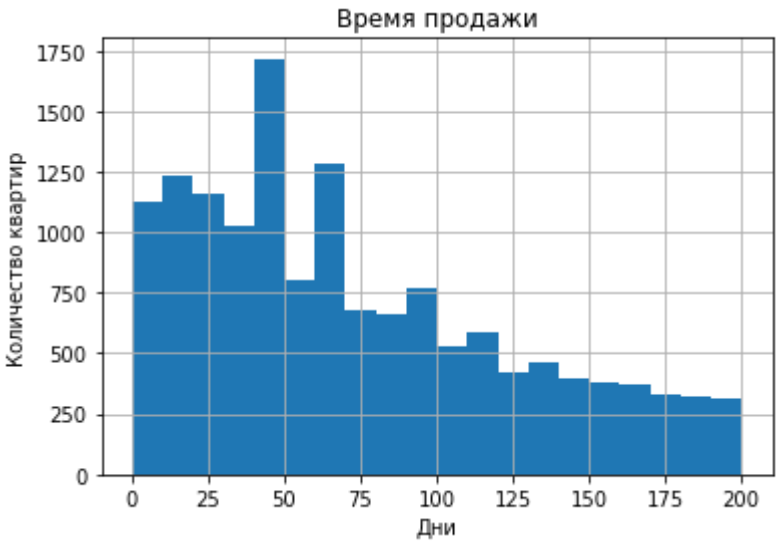
Out[43]:

	last_price	total_area	price_per_meter	first_day_exposition	day	month	year	roc
5556	11300000.0	89.000000	126966.0	2019-01-25	Friday	January	2019	
6069	12000000.0	87.000000	137931.0	2018-06-26	Tuesday	June	2018	
10874	10800000.0	65.000000	166154.0	2018-11-13	Tuesday	November	2018	
11377	13850000.0	107.000000	129439.0	2017-06-22	Thursday	June	2017	
21745	13900000.0	126.099998	110230.0	2018-07-30	monday	July	2018	

538 объявлений и из них лишь 5 выбиваются из условий. Но тут также нельзя однозначно судить. Мой вывод следующий: пики со значениями 45 и 60 дней практически наверняка связаны с условием размещения бесплатных объявлений в сервисе. Иначе я не могу объянить их природу.

Поэтому,предлагаю не заострять на них внимание. Выведем еще раз график перед глазами и сформируем общий вывод

In [44]:
```
plt.hist(good_days['days_exposition'], bins=20, range=(0, 200))
plt.title('Время продажи')
plt.xlabel('Дни')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
```



Чем дольше объявление опубликовано, тем меньше вероятность, что его купят. Много квартир покупают в первый месяц после публикации. Если не обращать внимание на пики в 45 и 60 дней, то график практически равномерно снижается с течением времени и стремится к нулю.

Можно предложить следующую классификацию:

- Продажа до 30 дней - быстрая
- Продажа от 30 до 90 дней - средняя
- Продажа больше 90 дней - медленная

Продажи за 900 и более дней признаем выбросами и убираем их. Это крайне долгий срок продажи

Факторы, влияющие на стоимость квартиры

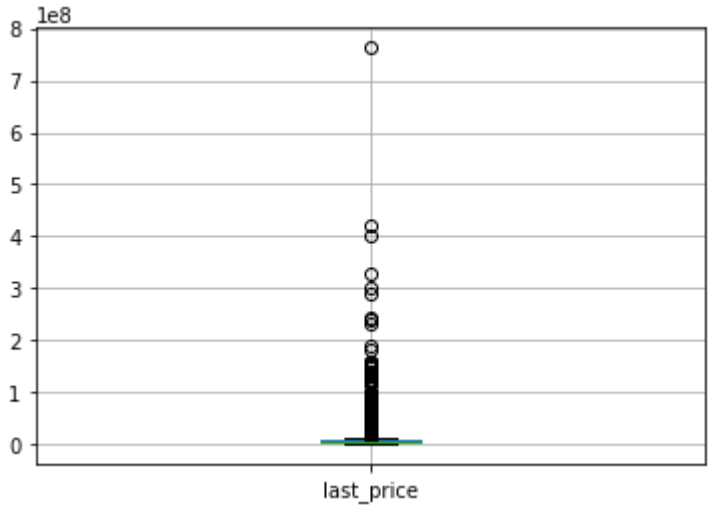
Изучим как влияют следующие факторы на стоимость квартиры:

- Площадь квартиры
- Количество комнат
- Расстояние до центра города
- Этаж (первый, последний или другое)

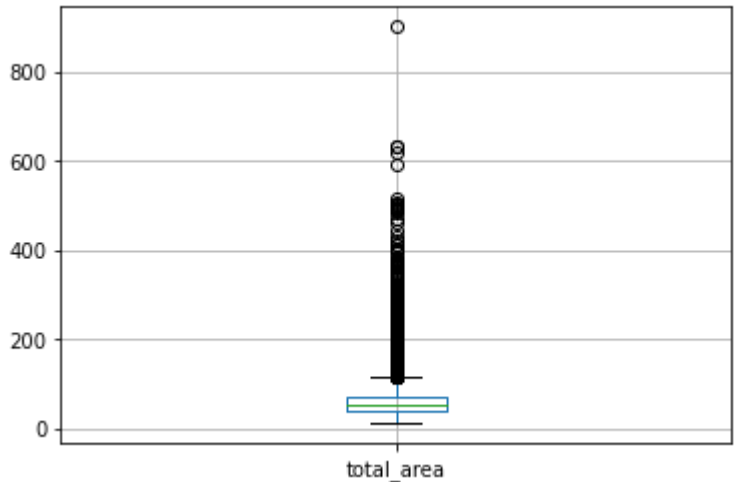
Но мы помним из пунктов выше, что в этих данных, вероятно, могут быть выбросы.
Построим диаграммы размаха и ознакомимся с результатами.

In [45]:
```
for i in ['last_price', 'total_area', 'days_exposition', 'ceiling_height', 'rooms']:
    print('Диаграмма размаха:')
    df.boxplot(column=i)
    plt.show()
    print('Статистический анализ:')
    print(df[i].describe())
```

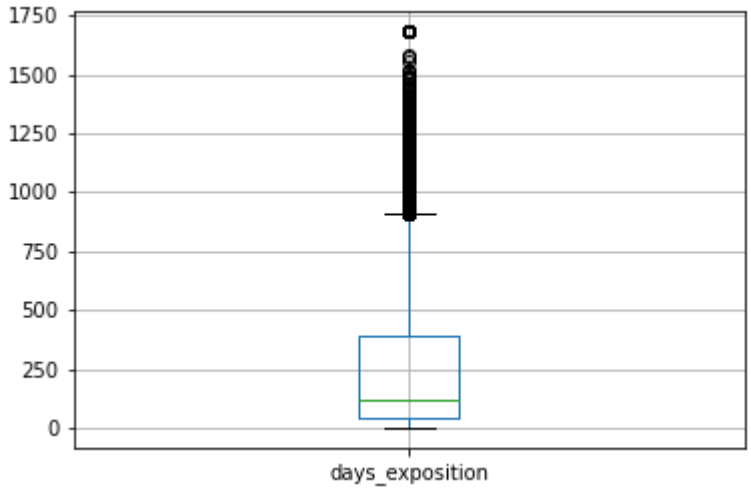
```
Диаграмма размаха:
```



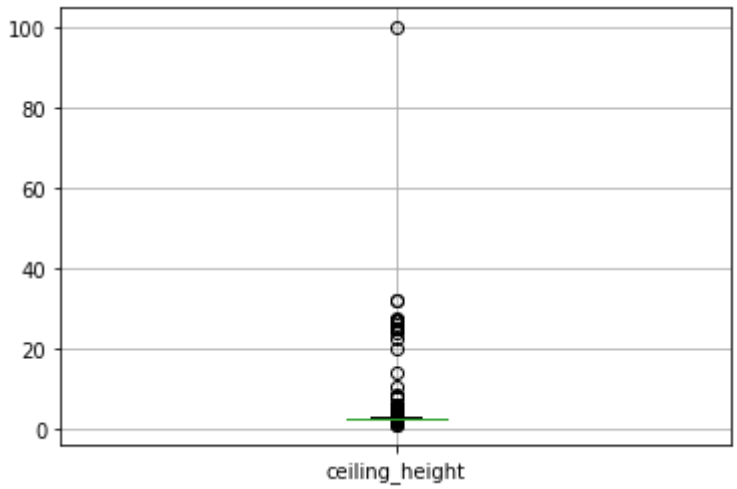

```
Статистический анализ:
count        23565.0
mean       6540059.0
std       10910923.0
min          12190.0
25%        3400000.0
50%        4646000.0
75%        6790000.0
max      763000000.0
Name: last_price, dtype: float64
Диаграмма размаха:
```



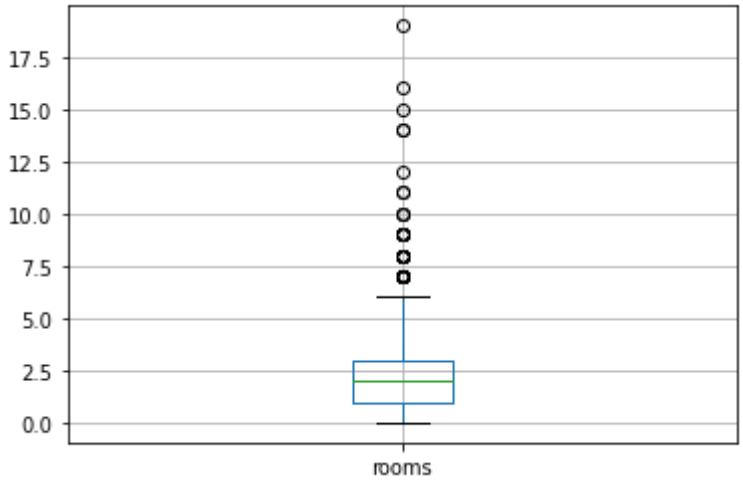
```
Статистический анализ:
count    23565.000000
mean        60.322979
std         35.657131
min         12.000000
25%         40.000000
50%         52.000000
75%         69.699997
max        900.000000
Name: total_area, dtype: float64
Диаграмма размаха:
```




```
Статистический анализ:
count    23565.000000
mean       382.487588
std        550.966817
min          1.000000
25%         45.000000
50%        124.000000
75%        390.000000
max       1680.000000
Name: days_exposition, dtype: float64
Диаграмма размаха:
```




```
Статистический анализ:
count    23565.000000
mean         2.724316
std          0.991057
min          1.000000
25%          2.600000
50%          2.650000
75%          2.700000
max        100.000000
Name: ceiling_height, dtype: float64
Диаграмма размаха:
```

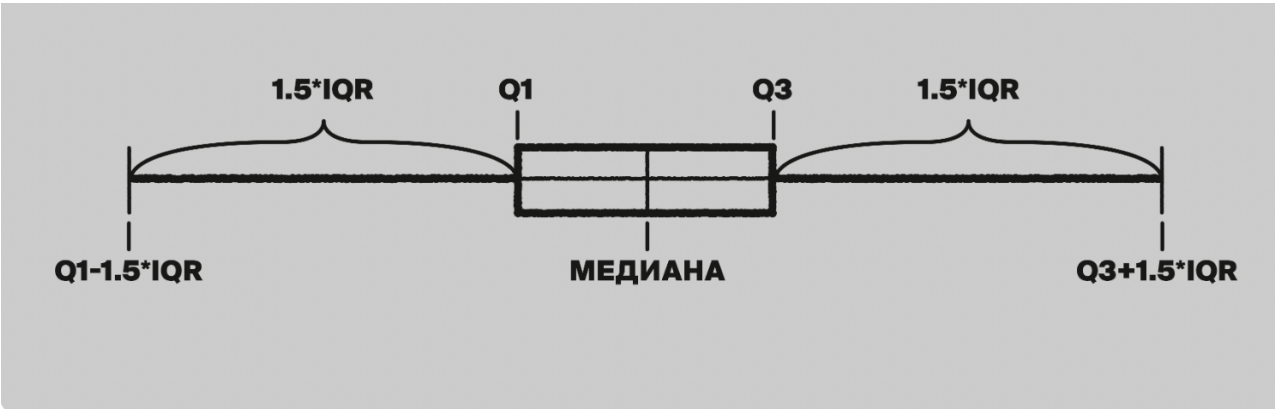



```
Статистический анализ:
count    23565.000000
mean         2.070656
std          1.078591
min          0.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         19.000000
Name: rooms, dtype: float64
```

Теперь предлагаю избавиться от выбросов в данных. Чтобы не потерять наш датафрейм, данные без выбросов сохраним в новой переменной `good_data`. Воспользуемся методом `.copy()`, чтобы не потерять наш исходный датафрейм. В пункте с анализом квартир в центре города я подробно объясню для чего это необходимо.

In [46]:
```
good_data = df.copy()
```

Напомним как выглядит схема диаграммы размаха. Это нам поможет при составлении функции отброса выбросов.



In [47]:
```
# Создадим функцию,которая возвращает датафрейм со значениями без выбросов.
def clean_data(data, column):
    q1 = data[column].quantile(0.25)
    q3 = data[column].quantile(0.75)
    iqr = q3 - q1
    max = q3+(1.5*iqr)
    min = q1-(1.5*iqr)
    pure = data.loc[(data[column] > min) & (data[column] < max), column]
    return pure
li1 = ['last_price', 'total_area', 'days_exposition', 'ceiling_height', 'rooms']
# Создаздим список столбцов, которые нужно обработать и применим к ним функцию clean
for col in li1:
    good_data[col] = clean_data(good_data, col)
good_data.describe()
```

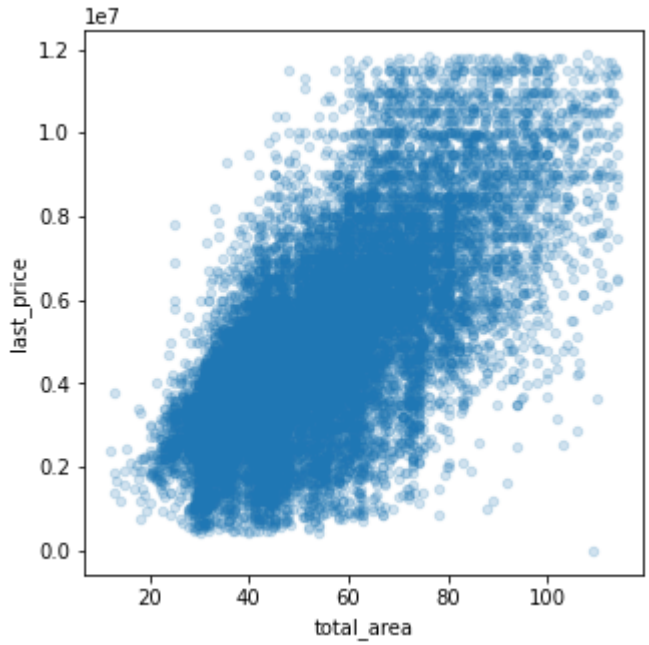
Out[47]:

	last_price	total_area	price_per_meter	year	rooms	ceiling_height	li
count	21519.0	22326.000000	2.356500e+04	23565.000000	23371.000000	20616.000000	2356
mean	4837086.5	54.319794	9.940578e+04	2017.371016	2.030208	2.630790	3
std	2215761.0	19.331085	5.038940e+04	1.037393	0.973563	0.081586	
min	12190.0	12.000000	1.120000e+02	2014.000000	0.000000	2.450000	
25%	3300000.0	39.400002	7.656600e+04	2017.000000	1.000000	2.600000	
50%	4400000.0	50.000000	9.500000e+04	2017.000000	2.000000	2.650000	3
75%	5999999.5	65.500000	1.142130e+05	2018.000000	3.000000	2.650000	4
max	11866860.0	114.199997	1.907500e+06	2019.000000	5.000000	2.850000	4

Выбросы откинули. Теперь можно разобраться с зависимости цены от различных факторов

Зависимость стоимости квартиры от площади

In [48]:
```
# Зависимость цены от площади
good_data.plot(x='total_area', y='last_price', kind='scatter', figsize=(5, 5), alpha
plt.show()
print('Корреляция цены и площади:','\n')
good_data[['last_price', 'total_area']].corr()
```




```
Корреляция цены и площади:
```

Out[48]:

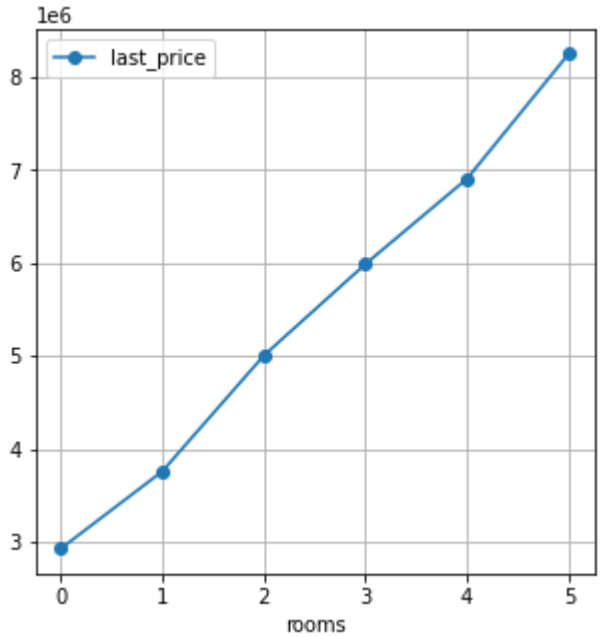
	last_price	total_area
last_price	1.000000	0.694052
total_area	0.694052	1.000000

Корреляция составила практически 0.7, что является достаточно высоким результатом. Это говорит о том, что зависимость высокая. График также показывает положительную зависимость. При увеличении площади, стоимость квартиры также растет

Зависимость стоимости от количества комнат

In [49]:
```
# Зависимость цены от количества комнат
(
good_data
```

```
    .pivot_table(index='rooms', values='last_price')
    .plot(grid=True, style='o-', figsize=(5, 5))
)
plt.show()
print('Корреляция цены и количества комнат:','\n')
good_data[['last_price', 'rooms']].corr()
```




```
Корреляция цены и количества комнат:
```

Out[49]:

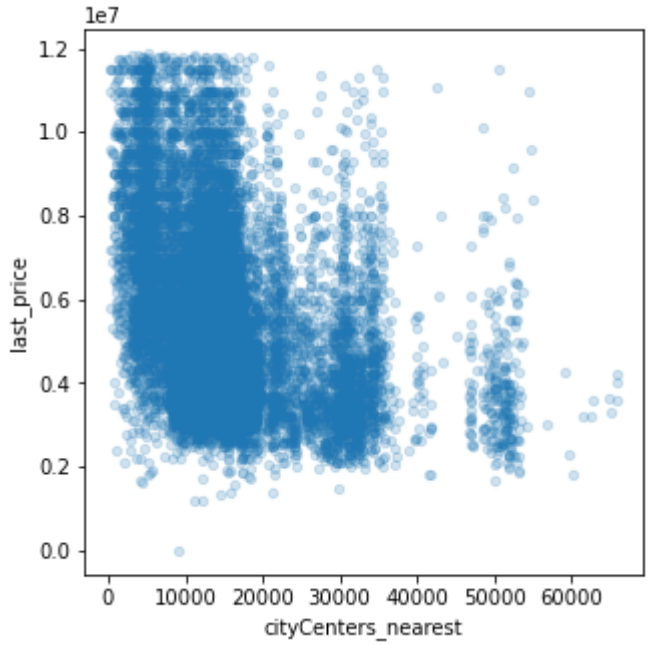
	last_price	rooms
last_price	1.000000	0.448907
rooms	0.448907	1.000000

Из график прекрасно видно, что стоимость квартиры и кол-во комнат тесно связаны.
Коэффициент Пирсона равен 0.44 что является умеренным значением.

Зависимость стоимости от удаления от центра

In [50]:

```
# Зависимость цены от расттояния до центра города
good_data.plot(x='cityCenters_nearest', y='last_price', kind='scatter', figsize=(5,
plt.show()
print('Корреляция цены и расстояния до центра города:','\n')
good_data[['last_price', 'cityCenters_nearest']].corr()
```



Корреляция цены и расстояния до центра города:

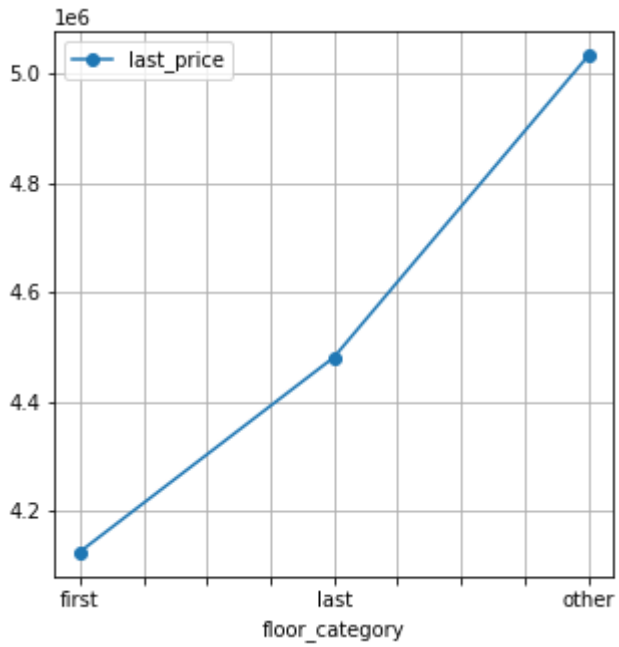
Out[50]:

	last_price	cityCenters_nearest
last_price	1.000000	-0.344927
cityCenters_nearest	-0.344927	1.000000

На графике видно, что чем ближе квартира к центру, тем цена ее выше. Коэффициент Пирсона в данном случае отрицателен и равен -0.34, что говорит об умеренной обратной связи. То есть при увеличении расстояния до центра, стоимость снижается.

Зависимость стоиомсти от этажа.

In [51]:
```
# Зависимость цены от этажа
(
good_data
    .pivot_table(index='floor_category', values='last_price')
    .plot(grid=True, style='o-', figsize=(5, 5))
)
plt.show()
```

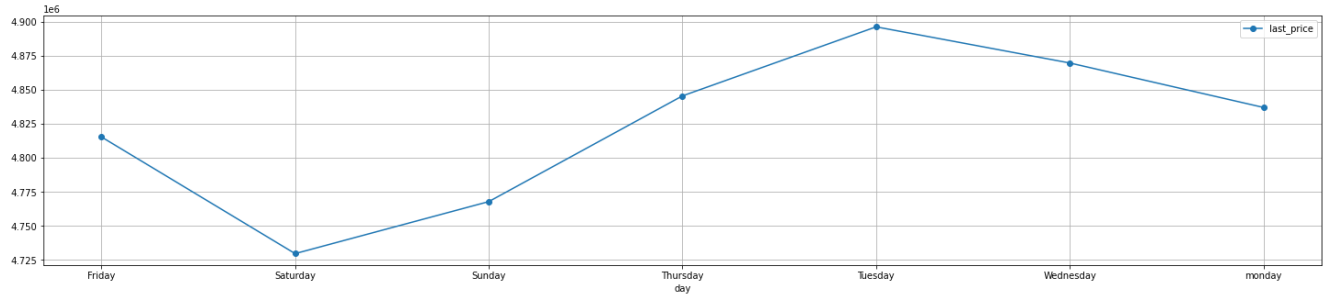


Отчетливо видно, что квартиры на первых этажах стоят дешевле всего. Следом идут квартиры на послежних этажах, а самые дорогие квартиры из категории "другие".

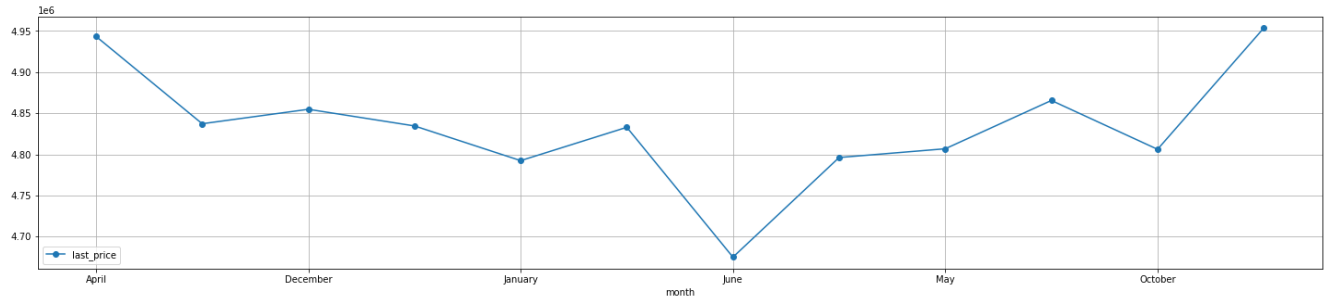
Коэффициент корреляции Пирсона посчитать не получится, тк в данном случае мы искали зависимость стоимости от категориального значения

Зависимость стоимости от даты.

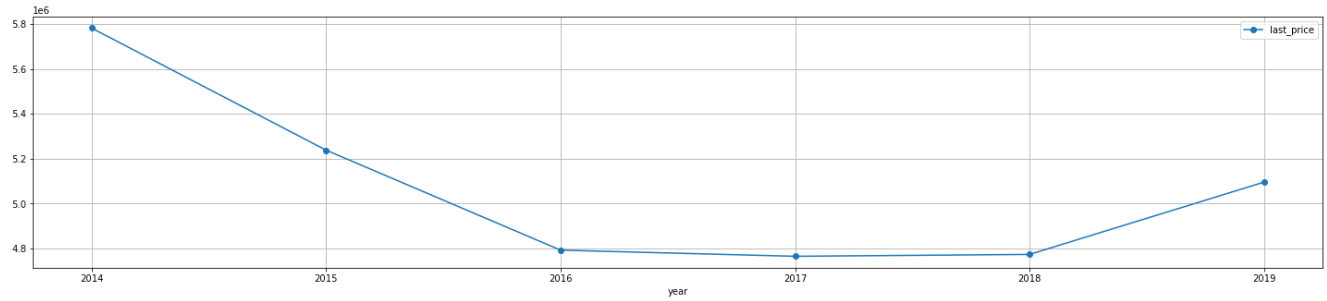
In [52]:
```
# Автоматизируем построение графиков.
for i in ['day', 'month', 'year']:
    (
    good_data
        .pivot_table(index=i, values='last_price')
        .plot(grid=True, style='o-', figsize=(25, 5))
    )
    plt.show()
    display(good_data.pivot_table(index=i, values='last_price').sort_values('last_pr
```



day	last_price
Saturday	4729754.0
Sunday	4767974.5
Friday	4815482.0
monday	4837175.5
Thursday	4845654.5
Wednesday	4869843.5
Tuesday	4896393.0



month	last_price
June	4674983.0
January	4792335.5
March	4796123.0
October	4806094.5
May	4806753.0
July	4832851.5
February	4834506.5
August	4837171.0
December	4854751.5
November	4865480.0
April	4943327.5
September	4953424.5



year	last_price
2017	4764757.0
2018	4773183.5
2016	4792287.0
2019	5095495.0
2015	5238064.5
2014	5782435.5

Касательно дня недели и месяца можно сказать следующее: Конечно, разница в цене и дате размещения есть. Например, квартиры размещенные во вторник стоят дороже остальных. Также можно сказать и про месяц размещения: Сентябрьские квартиры оказались самыми дорогими. Но в целом, разброс достаточно невелик. Разброс от "самого дорогого" до "самого дешевого" дня недели составил чуть менее 300 тыс рублей, а разброс в месяцах оказался еще меньше. Да, некоторая зависимость безусловно есть, но она не критическая.

Что же касается года размещения, то тут картина более интересная. В 2014 квартиры стоили явно дороже. После этого стоимость квартир пошла на спад к 2016 году. Далее, пару лет стоимость практически не изменялась, а вот в 2018 году произошел рост цен.

Общий вывод по поданному пунтку

Мы проанализировали некоторые факторы, которые тем или иным способом влияют на стоимость квартиры. Ниже результаты, которые мы получили. Представлю их по порядку

влияния фактора на стоимость:

- **Площадь квартиры** - Данный фактор влияет на стоимость больше остальных. Коэффициент корреляции равен почти 0.7. Чем больше площадь, тем выше стоимость квартиры.
- **Количество комнат** - Далее по значимости идет кол-во комнат с коэффициентов корреляции 0.44. Достаточно значительная корреляция.
- **Расстояние до центра** - На почетном третьем месте по коэффициенту корреляции оказывается расстояние до центра. Коэффициент -0.34.
- **Этаж** - Хоть у этажа и нет коэффициента корреляции, зависимость этого фактора достаточно высокая. Квартиры на первом этаже в среднем практически на миллион дешевле квартир квартир на других этажах
- **Дата размещения** - Дата размещения, в целом, несильно влияют на стоимость, но все-таки определенная зависимость есть. А вот год размещения достаточно сильно повляиюл на стоимость квартир.

Населенные пункты с наибольшим числом объявлений

In [53]:
```
# Создадим сводную таблицу с количеством объявлений и средней ценой за метр
top_locality = (
good_data
    .pivot_table(index='locality_name', values='price_per_meter', aggfunc=['count',
)
# Переименуем колонки и обновим наш топ, оставив в нем лишь первые 10 городов
top_locality.columns = ['count', 'mean']
top_locality = top_locality.sort_values('count', ascending=False).head(10)
top_locality
```

Out[53]:

locality_name	count	mean
Санкт-Петербург	15651	114868.875000
посёлок Мурино	520	85673.257812
посёлок Шушары	439	78551.359375
Всеволожск	398	68654.476562
Пушкин	369	103125.820312
Колпино	338	75424.570312
посёлок Парголово	327	90175.890625
Гатчина	307	68746.109375
деревня Кудрово	299	92473.585938
Выборг	237	58141.917969

Больше всего объявлений и самые дорогие квартиры в городе Санкт-Петербург. Город Пушкин занимает второе место по стоимости квартир и входит в среднюю группу по количеству объявлений. Меньше всего объявлений и самые дешевые квартиры в Выборге

Анализ квартир из центральной зоны

Определение центральной зоны

Нам предстоить выделить квартиры из центра в отдельную категорию и провести исследовательский анализ данных.

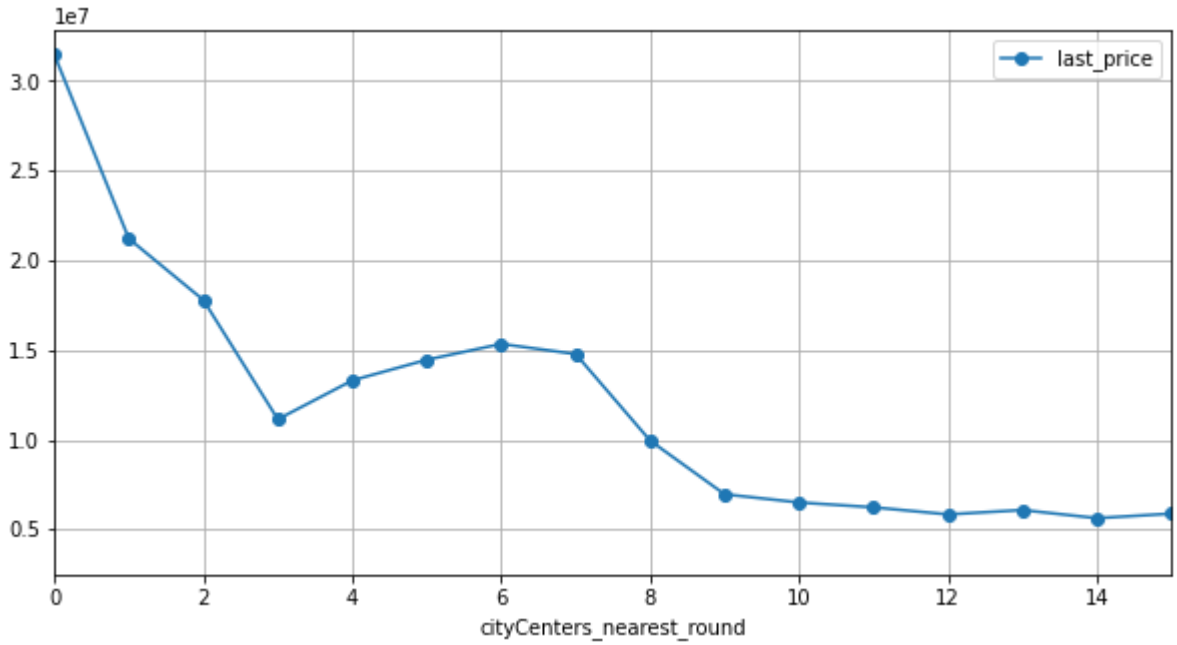
Создадим колонку с округленным до километра расстоянием до центра. Посчитаем среднюю цену для каждого километра, построем график зависимости расстояния от средней цены и определим зону центра

In [54]:
```
#Создазим колонку с округленным расстояниейм до центра
df['cityCenters_nearest_round'] = round(df['cityCenters_nearest']/1000, 0)

# Создадим переменную с объявлениями из Санкт_Петербурга
data_spb = df.query('locality_name == "Санкт-Петербург"')

#Построим график для определния центральной зоны
(
data_spb
    .pivot_table(index='cityCenters_nearest_round', values='last_price', aggfunc='me
    .plot(grid=True, style='o-', xlim=(0,15), figsize=(10, 5))
)
```

Out[54]:
```
<AxesSubplot:xlabel='cityCenters_nearest_round'>
```



На графике есть некоторый спад в районе 3км, но потом цена поднимается. Вероятно, это обусловлено географической особенностью зоны в радиусе 3-5км от центра города. А вот после 7км средняя стоимость начинает заметно уменьшаться и далее практически не увеличивается. Определяем, что центральная зона - 7км.

Выделим сегмент квартир в центре и проанализируем данные. Данные сохраним в перемнную `spb_center`
По сути, нам нужно повторить пунткы 5.1 - 5.3 только по определенному сегменту квартир. Поэтому описаний практически не будет, только выводы.

In [55]:
```
spb_center = data_spb.query('cityCenters_nearest_round < 7')
```

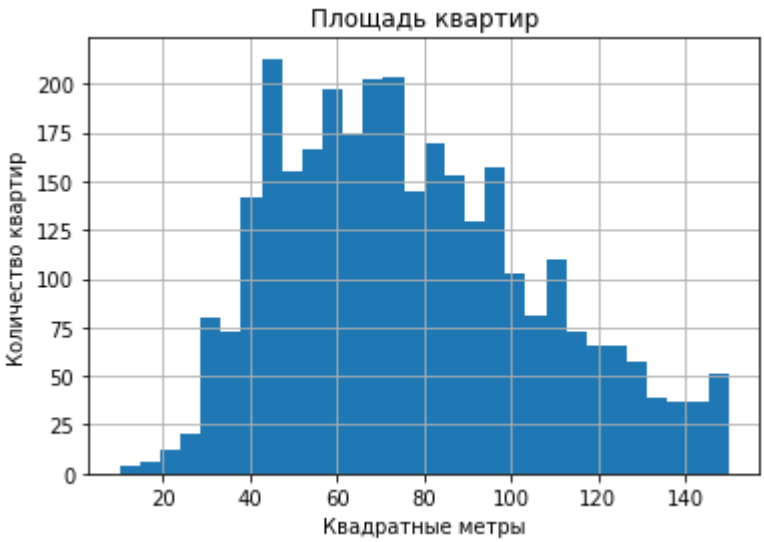
In [56]:
```
# Выделим сегмент квартир в центре
spb_center = data_spb[data_spb['cityCenters_nearest_round'] <= 7]
```

Гистограмма площади квартир

In [57]:
```
# Построим гистограмму площади квартир
```

```
plt.hist(spb_center['total_area'], bins=30, range=(10,150))
plt.title('Площадь квартир')
plt.xlabel('Квадратные метры')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

# Выведем количество квартира с максимальной площадью
print('Наибольшие по площади варианты:')
display((spb_center
    .pivot_table(index='total_area', values='last_price', aggfunc='count')
    .sort_values('total_area', ascending=False)
    .rename(columns={'last_price': 'count'})
    .head(10)
))
# Статистический анализ
spb_center['total_area'].describe()
```



Наибольшие по площади варианты:

total_area	count
631.200012	1
631.000000	1
618.000000	1
590.000000	1
517.000000	1
507.000000	1
500.000000	2
495.000000	1
494.100006	1
491.000000	1

Out[57]:
```
count    3518.000000
mean       93.083366
std        59.429222
min        12.000000
25%        56.925001
50%        78.599998
75%       110.000000
max       631.200012
Name: total_area, dtype: float64
```

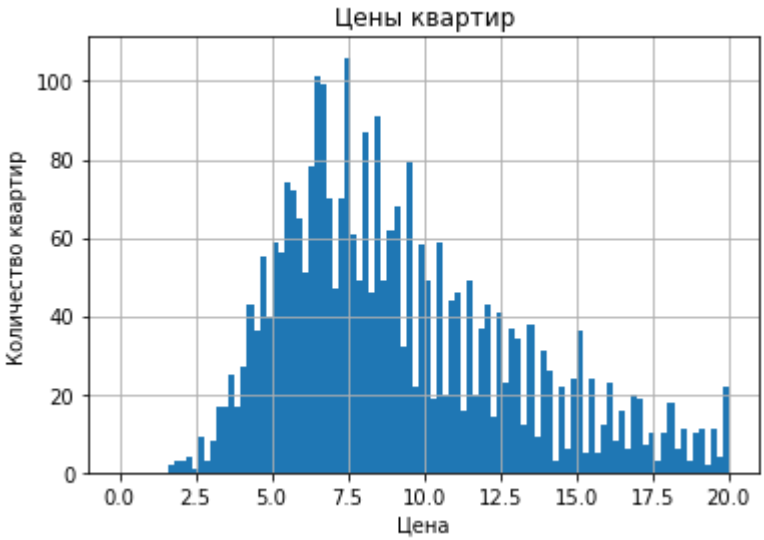
Пик гистограммы приходится на квартиры с площадбю около 45м2. Для данных по общей базы пик тоже приходился на 45м2. Однако можно заметить, что в центре квартиры больше. Медиана 78м2 против 52м2 по всей базе. Среднее 93м2 против 60м2. Можно однозначно сказать, что квартиры в центре города больше

Гистограмма стоимости квартир

In [58]:
```
# Построим гистограмму стоимости квартир
plt.hist(spb_center['price_round'], bins=100, range=(0, 20))
plt.title('Цены квартир')
plt.xlabel('Цена')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

# Количсетво квартир
display(spb_center['price_round'].value_counts().head(10))

# Статистический анализ
spb_center['last_price'].describe()
```




```
8.5     50
7.5     49
8.0     42
6.5     42
9.5     35
9.0     35
10.5    33
12.0    33
6.7     32
10.0    30
Name: price_round, dtype: int64
```

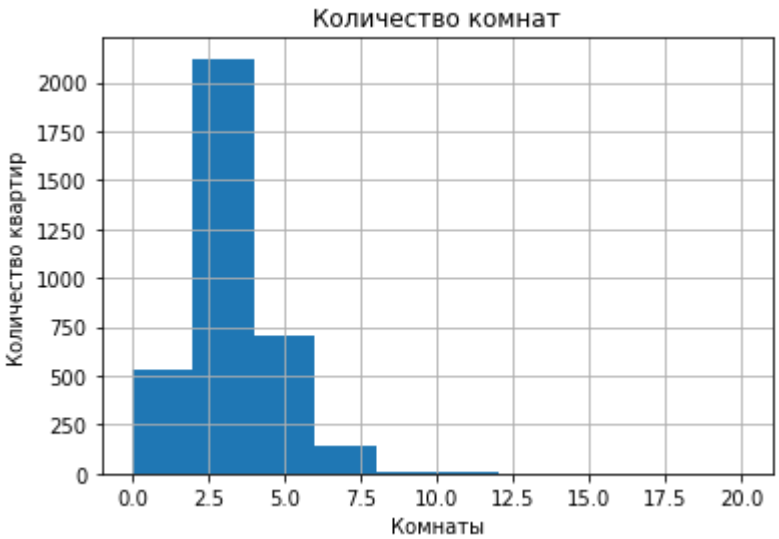
Out[58]:
```
count         3518.0
mean      14888101.0
std       24340466.0
min        1600000.0
25%        6667500.0
50%        9200000.0
75%       14497500.0
max      763000000.0
Name: last_price, dtype: float64
```

Квартир менее 4 млн очень мало. Пик приходится на квартиры стоимость около 8,5 млн рублей. Медиана составила 7,6 млн рублей, что больше медианы по всей базе на почти 3млн рублей. Среднее 14,8млн больше среднего по всей базе более чем в 2 раза.
Что, в принципе, и логично - квартиры в центре значительно дороже остальных.

Гистограмма по количеству комнат

In [59]:
```
# Гистограмма по количеству комнат
plt.hist(spb_center['rooms'], bins=10, range=(0, 20))
plt.title('Количество комнат')
plt.xlabel('Комнаты')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()
#квартиры с очень большим количество комнат
print('Наибольшее количество комнат:')
display((spb_center
    .pivot_table(index='rooms', values='total_area', aggfunc='count')
    .rename(columns={'total_area': 'count'})
    .sort_values('count', ascending=False)
    .head(10)
))

# Статистический анализ
spb_center['rooms'].describe()
```



Наибольшее количество комнат:

rooms	count
3	1088
2	1033
1	518
4	488
5	216
6	83
7	54
0	17
8	10
9	3

Out[59]:
```
count    3518.000000
mean        2.833712
std         1.450317
min         0.000000
25%         2.000000
50%         3.000000
75%         3.000000
max        19.000000
Name: rooms, dtype: float64
```

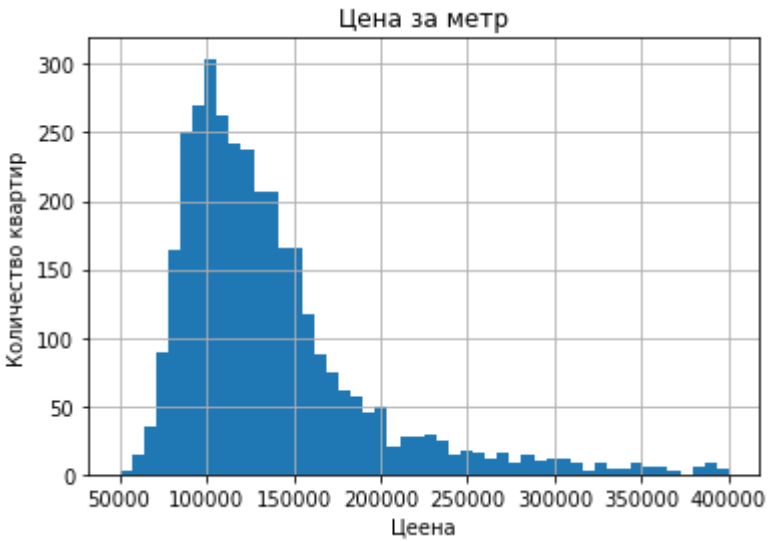
Большинство квартир с 2-3 комнатами. Медиана составила 3 комнаты, в отличии от всех квартир, где медиана ровнялась 2. Соответсвенно, в центре больше трехкомнатынх квартир, в то время как в остальных частях города преимущественно двухкомнатные квартиры

Гистограмма по цене за метр

In [60]:
```
# Гистограмма по цене за метр
plt.hist(spb_center['price_per_meter'], bins=50, range=(50000, 400000))
plt.title('Цена за метр')
plt.xlabel('Цеена')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

#квартиры с очень большим количество комнат
display(round(spb_center['price_per_meter'] / 1000 ,2).value_counts().head(10))

# Статистический анализ
spb_center['price_per_meter'].describe()
```




```
100.000000    32
125.000000    18
200.000000    11
93.750000      8
95.000000      8
133.330002     7
160.000000     7
150.000000     7
140.000000     7
178.570007     6
Name: price_per_meter, dtype: int64
```

Out[60]:
```
count    3.518000e+03
mean     1.452457e+05
std      9.370804e+04
min      1.534500e+04
25%      9.995575e+04
50%      1.235980e+05
75%      1.558140e+05
max      1.907500e+06
Name: price_per_meter, dtype: float64
```

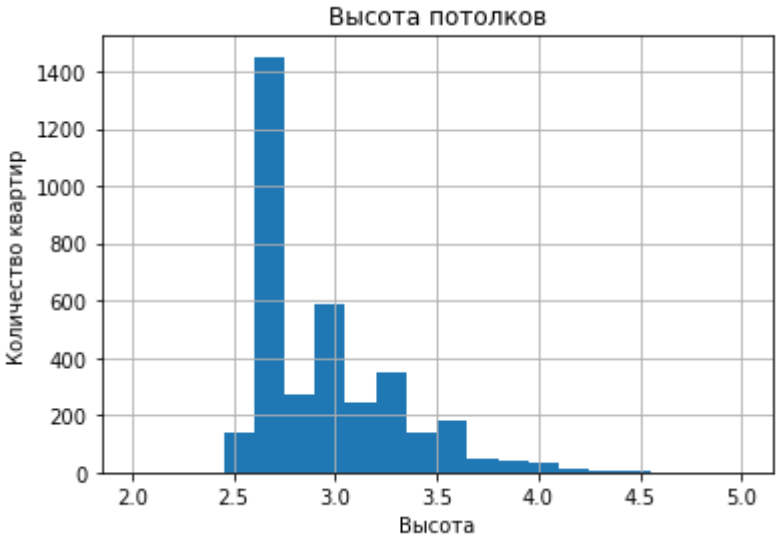
Похоже на нормальное распределение. Медиана равна примерно 123 тыс рублей, что почти на 30 тыс рублей больше остальных объявлений. У большинства квартир стоимость квадратного метра в диапазоне от 90 до 120 тыс рублей.

Гистограмма по высоте потолков

In [61]:
```
# Гистограмма по высоте потолков
plt.hist(spb_center['ceiling_height'], bins=20, range=(2, 5))
plt.title('Высота потолков')
plt.xlabel('Высота')
plt.ylabel('Количество квартир')
plt.grid(visible=True)
plt.show()

#квартиры по высоте потолков
display(spb_center['ceiling_height'].value_counts().head(10))


# Статистический анализ
spb_center['ceiling_height'].describe()
```




```
2.65    1259
3.00     453
3.20     180
2.70     123
2.80     112
3.10     111
3.50     107
2.50     103
3.30      91
3.40      82
Name: ceiling_height, dtype: int64
```

Out[61]:
```
count    3518.000000
mean        2.954473
std         0.737483
min         2.400000
25%         2.650000
50%         2.800000
75%         3.150000
max        32.000000
Name: ceiling_height, dtype: float64
```

Пока пик приходится на квартиры с высотой около 2,7 метров. Медина составила 2,8 метра, а срднее около 2,95 метра. Получается, что в центре квартиры в среднем выше остальных на 20-25 см

В этом пункте я также хочу объяснить решение создать датафрейм `good_data` в пункте 5.3 именно с использованием метода `.copy()`. Дело в том, что если бы я не применил метод `.copy()`, а просто объявил переменную `good_data = df`, а потом применил к ней функцию по очистке от выбросов, то и в исходном датафрейме `df` также бы убрались те самые выбросы. Это особенность языка python. Очевидно, что выбросы для всего датафрейма и выбросы для квартир только из центра будут разными, продемонстрирую это наглядно ниже:

In [62]:
```
# Добавим колонку с округленным растояние до центра в датафрейм с обработанными выбр
good_data['cityCenters_nearest_round'] = round(good_data['cityCenters_nearest']/1000
# Стат анализ квартир в центре из датафрейма с обработанными выбросами
display(good_data[good_data['cityCenters_nearest_round'] <= 7].describe())
# Стат анализ квартир в центре из полного датафрейма
display(df[df['cityCenters_nearest_round'] <= 7].describe())
```

	last_price	total_area	price_per_meter	year	rooms	ceiling_height	living_
count	2292.0	2726.000000	3.518000e+03	3518.000000	3360.000000	1849.000000	3518.00
mean	7436827.5	69.433678	1.452457e+05	2017.171120	2.642857	2.663092	55.21
std	2154672.0	22.494978	9.370804e+04	1.125478	1.115725	0.068216	38.79
min	1600000.0	12.000000	1.534500e+04	2014.000000	0.000000	2.450000	2.00
25%	5820000.0	51.000000	9.995575e+04	2017.000000	2.000000	2.650000	31.20
50%	7350000.0	69.000000	1.235980e+05	2017.000000	3.000000	2.650000	46.31
75%	9000000.0	86.674995	1.558140e+05	2018.000000	3.000000	2.650000	67.00
max	11866860.0	114.199997	1.907500e+06	2019.000000	5.000000	2.850000	409.00

	last_price	total_area	price_per_meter	year	rooms	ceiling_height	living
count	3518.0	3518.000000	3.518000e+03	3518.000000	3518.000000	3518.000000	3518.0
mean	14888101.0	93.083366	1.452457e+05	2017.171120	2.833712	2.954473	55.2
std	24340466.0	59.429222	9.370804e+04	1.125478	1.450317	0.737483	38.7
min	1600000.0	12.000000	1.534500e+04	2014.000000	0.000000	2.400000	2.0
25%	6667500.0	56.925001	9.995575e+04	2017.000000	2.000000	2.650000	31.2
50%	9200000.0	78.599998	1.235980e+05	2017.000000	3.000000	2.800000	46.3
75%	14497500.0	110.000000	1.558140e+05	2018.000000	3.000000	3.150000	67.0
max	763000000.0	631.200012	1.907500e+06	2019.000000	19.000000	32.000000	409.0

Как мы видим, статистические показатели сильно различаются. Для квартир из центра, 3й квартиль стоимости равено примерно 14,5млн рублей, а для всех квартир данное значение

уже являлось выбросом. Именно поэтому я считаю, что для квартир в центре города, необходимо отдельно искать и убирать выбросы.

Избавимся от выбросов для квартир в центра.

In [63]:
```
spb_center_good = spb_center.copy()
```

In [64]:
```
def clean_data(data, column):
    q1 = data[column].quantile(0.25)
    q3 = data[column].quantile(0.75)
    iqr = q3 - q1
    max = q3+(1.5*iqr)
    min = q1-(1.5*iqr)
    pure = data.loc[(data[column] > min) & (data[column] < max), column]
    return pure
li1 = ['last_price', 'total_area', 'days_exposition', 'ceiling_height', 'rooms']
# Создаздим список столбцов, которые нужно обработать и применим к ним функцию clean
for col in li1:
    spb_center_good[col] = clean_data(spb_center_good, col)
spb_center_good.describe()
```

Out[64]:

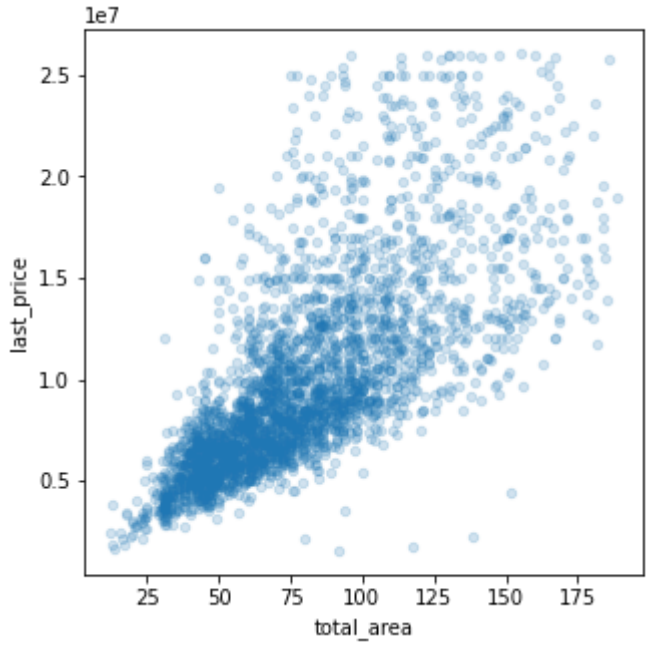
	last_price	total_area	price_per_meter	year	rooms	ceiling_height	living_
count	3160.0	3321.000000	3.518000e+03	3518.000000	3127.000000	3431.000000	3518.000
mean	9942668.0	82.528381	1.452457e+05	2017.171120	2.494404	2.907578	55.213
std	4933581.5	35.703190	9.370804e+04	1.125478	0.945348	0.309816	38.790
min	1600000.0	12.000000	1.534500e+04	2014.000000	1.000000	2.400000	2.000
25%	6482500.0	55.500000	9.995575e+04	2017.000000	2.000000	2.650000	31.203
50%	8500000.0	75.900002	1.235980e+05	2017.000000	3.000000	2.800000	46.316
75%	12300000.0	102.599998	1.558140e+05	2018.000000	3.000000	3.100000	67.000
max	26037742.0	189.000000	1.907500e+06	2019.000000	4.000000	3.880000	409.000

Теперь мы избавились от выбросов в центре и можно изучить зависимсоти стоимости квартир от различных факторов

Подводя небольшой итог можно сказать, что квартиры в центре сильно дорже остальных квартир. Потолки в центре тоже серьезно выше. И количество комнат в среднем больше.

Зависимость цены от площади

In [65]:
```
# Зависимость цены от площади
spb_center_good.plot(x='total_area', y='last_price', kind='scatter', figsize=(5, 5),
plt.show()
print('Корреляция цены и площади:','\n')
spb_center_good[['last_price', 'total_area']].corr()
```



Корреляция цены и площади:

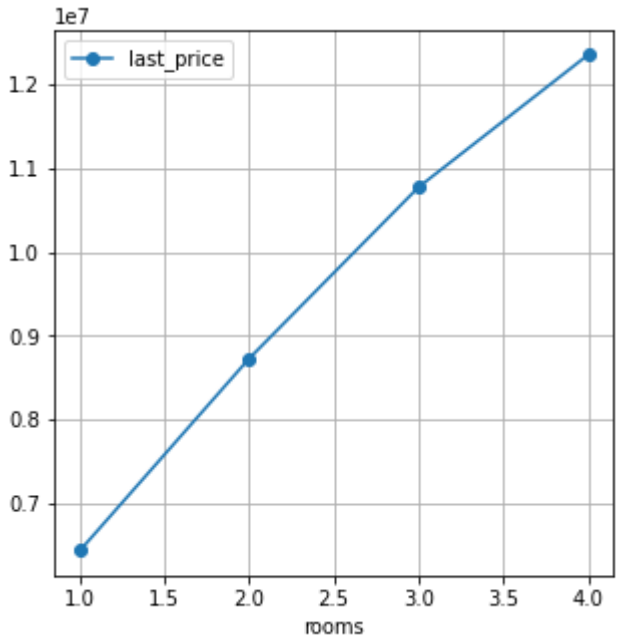
Out[65]:

	last_price	total_area
last_price	1.000000	0.717992
total_area	0.717992	1.000000

Корреляция составила уже чуть более 0.7, против чуть менее 0.7 по базе в целом. Это говорит о том, что зависимость высокая. График также показывает положительную зависимость. При увеличении площади, стоимость квартиры также растет

Зависимость стоимости от количества комнат

In [66]:
```
# Зависимость стоимости от количества комнат
(
spb_center_good
    .pivot_table(index='rooms', values='last_price')
    .plot(grid=True, style='o-', figsize=(5, 5))
)
plt.show()
print('Корреляция цены и количества комнат:','\n')
spb_center_good[['last_price', 'rooms']].corr()
```



Корреляция цены и количества комнат:

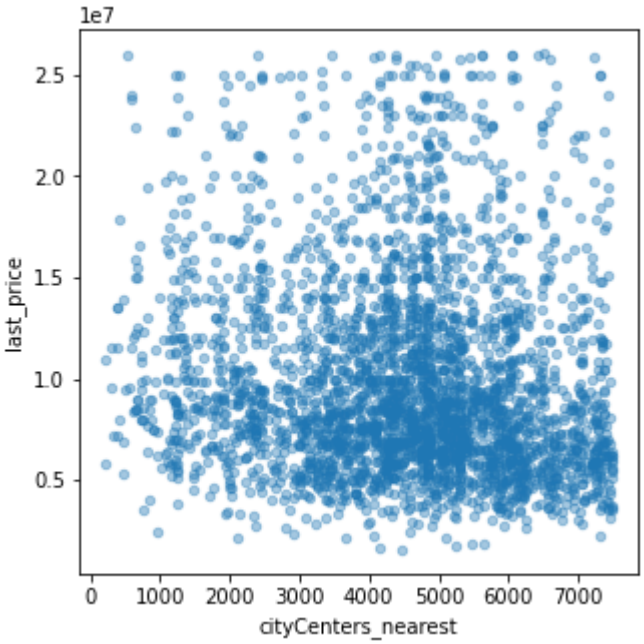
Out[66]:

	last_price	rooms
last_price	1.000000	0.400969
rooms	0.400969	1.000000

Из график прекрасно видно, что стоимость квартиры и кол-во комнат тесно связаны. Коэффициент Пирсона равен 0.4 что является умеренным значением. Чем больше комнат, тем выше цена квартиры в центре города.

Зависимость стоимости от удаления от центра

In [67]:
```
# Зависимость цены от расттояния до центра города
spb_center_good.plot(x='cityCenters_nearest', y='last_price', kind='scatter', figsiz
plt.show()
print('Корреляция цены и расстояния до центра города:','\n')
spb_center_good[['last_price', 'cityCenters_nearest']].corr()
```



Корреляция цены и расстояния до центра города:

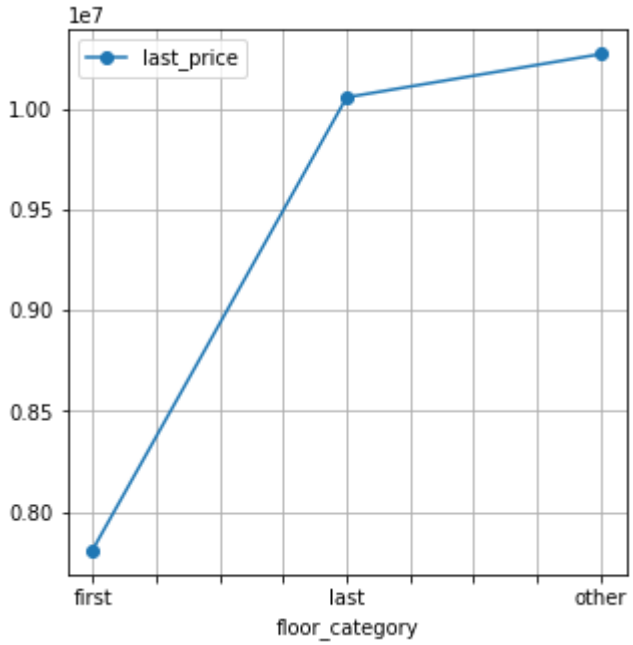
Out[67]:

	last_price	cityCenters_nearest
last_price	1.000000	-0.110034
cityCenters_nearest	-0.110034	1.000000

Коэффициент Пирсона практически равен нулю, посколько мы и так анализируем квартиры, которые располагаются в центре города.

Зависимость стоиомсти от этажа.

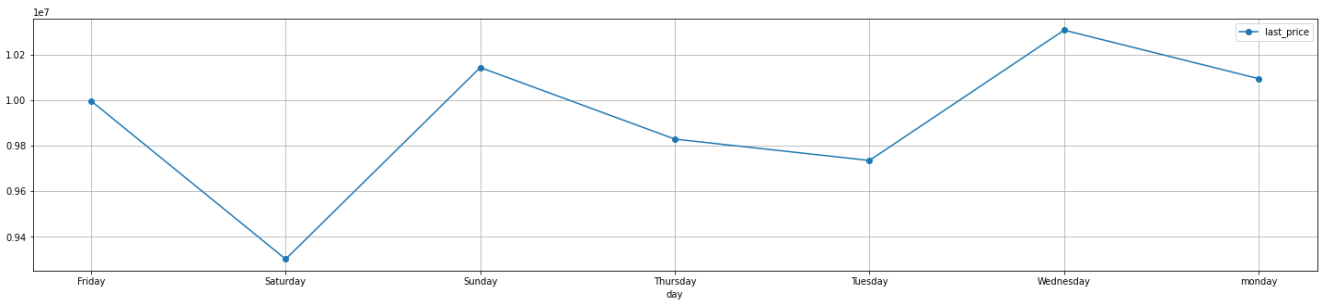
In [68]:
```
# Зависимость цены от этажа
(
spb_center_good
    .pivot_table(index='floor_category', values='last_price')
    .plot(grid=True, style='o-', figsize=(5, 5))
)
plt.show()
```



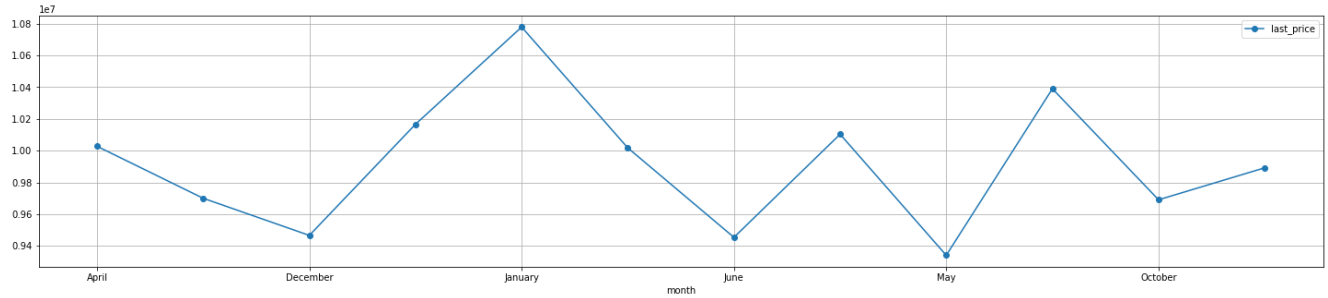
Отчетливо видно, что квартиры на первых этажах стоят дешевле всего. Следом идут квартиры на послежних этажах, а самые дорогие квартиры из категории "другие".
Но в отличии от квартир по базе в целом, разница между квартирами на послежних этажах и на "других" не такая заметная. Это говорит о том, что квартиры на последних этажах в центре города ценятся больше, чем квартиры на последних этажах в остальной части города. Центр Питер, доступ к крышам... Кажется я отчетливо понимаю почему так происходит)

Зависимость стоимости от даты.

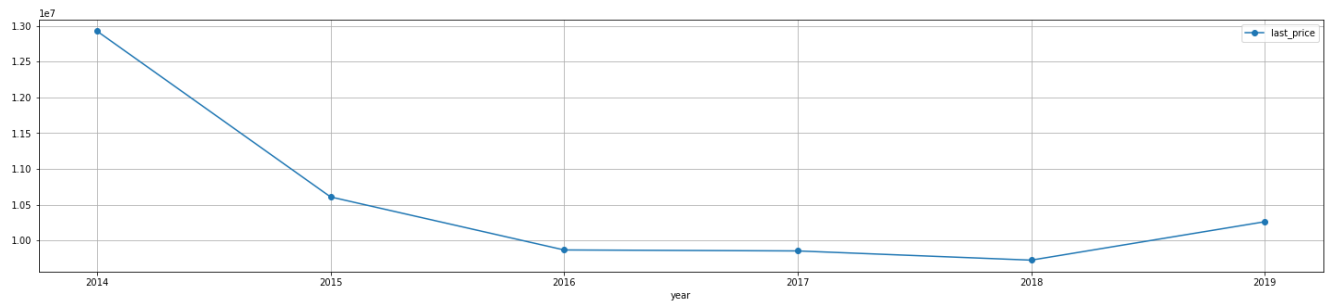
In [69]:
```
# Автоматизируем построение графиков.
for i in ['day', 'month', 'year']:
    (
    spb_center_good
        .pivot_table(index=i, values='last_price')
        .plot(grid=True, style='o-', figsize=(25, 5))
    )
    plt.show()
    display(spb_center_good.pivot_table(index=i, values='last_price').sort_values('l
```



day	last_price
Saturday	9302527.0
Tuesday	9735297.0
Thursday	9828800.0
Friday	9996427.0
monday	10093918.0
Sunday	10142941.0
Wednesday	10306733.0



month	last_price
May	9340917.0
June	9454005.0
December	9466281.0
October	9690723.0
August	9700188.0
September	9891882.0
July	10017503.0
April	10027629.0
March	10103493.0
February	10166505.0
November	10389962.0
January	10778812.0



year	last_price
2018	9725933.0
2017	9854745.0
2016	9868334.0
2019	10262525.0
2015	10609193.0
2014	12926177.0

Самые дорогие квартиры размещали в среду, а не во вторник как по общей базе. Также в общей базе самые дорогие объявления были размещены в сентябре, а для квартир в центре самым дорогим месяцем стал январь. По годам для квартир в центре график более ровный. С 2014 года квартира все время дешевели, и лишь в 2018 году средняя стоимость начала увеличиваться.

Вывод по квартирам в центре

Проанализировав зависимости стоиомсти квартир от некоторых факторов, можно сказать, что стоиомсть квартиры в центре все также сильно зависит от площади самой квартиры. Чуть меньше остальных, квартиры в центре зависят от этажа, на котором расположены. А разница между стоимостью квартиры на последнем и "другом" этаже минимальная. И в целом, квартиры в центре дороже и больше остальных.

Общий вывод

Теперь можно подвести полный итог по проделанной работе.
Был выполнен большой объем работы: изучение файла, борьба с пропусками и аномалиями, расчеты некоторых результатов, выявление выбросов в данных, которые помешали бы анализу. Проанализированы основные факторы, которые влияют на стоимость квартиры.
Из самым популярных населенных пунктов, дороже всего квартиры стоят в Санкт-Петербурге. Дешевле всего - Выборг

Быстрыми, можно назвать продажи до 30 дней, а медленными 90+ дней. В основном, квартиры продаются в срок до 200 дней.

По нашим данным можно сделать следующие выводы: Стоимость квартиры напрямую зависит от многих факторов: площадь квартиры, кол-во комнат, этаж и даже дата размещения объявления.

Больше всего на стоимость влияет площадь квартиры и только потом уже кол-во комнат.

Также можно смело сказать, что средня цена квартир, расположенных не на первом и не на посленем этаже выше остальных.
Расстояние до центра оказалось очень важным фактором для нашего анализа.

Мы наглядно показали, что квартиры в центре нужно выделять в отдельную категорию и работать с ними отдельно от основных данных. Квартиры в центре сильно дороже остальных, имеют бОльшую площадь, бОльшее кол-во комнат и более выоские потолки.
