Анализ рынка недвижимости Санкт-Петербурга

Содержание

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

По каждой квартире на продажу доступны два вида данных:

1. Первые вписаны пользователем.
2. Вторые получены автоматически на основе картографических данных. Например, расстояние до центра, аэропорта, ближайшего парка и водоёма.

Конкретизируем задачу: необходимо установить зависимость конечной цены проданной квартиры от её различных параметров, а также сформулировать предложения по составу ключевых параметров будущей системы отслеживания объявлений

Данная конкретизация справедлива, поскольку текущая цена непроданных квартир часто зависит от субъективных факторов: жадности продавцов, ажиотажа на рынке и т.п.

Изучение данных

Импортируем необходимые библиотеки, загрузим данные из файла и выведем 5 случайных строк:

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

try:
    realty = pd.read_csv('/datasets/real_estate_data.csv', sep='\t')
except:
    realty = pd.read_csv('real_estate_data.csv.csv', sep='\t')

realty.sample(5)
```

Out[1]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcony	locality_name	airports_
2931	14	2550000.0	44.0	2018-08-08T00:00:00	2	2.6	2.0	26.0	1	NaN	...	8.0	2.0	деревня Вартемяги	
580	16	5270000.0	56.7	2019-02-07T00:00:00	3	2.5	5.0	42.0	3	NaN	...	5.1	NaN	Санкт-Петербург	
22849	6	2380000.0	25.0	2016-03-22T00:00:00	1	NaN	25.0	19.0	15	NaN	...	4.0	1.0	посёлок Шушары	
14013	7	2700000.0	36.6	2017-11-26T00:00:00	1	NaN	18.0	23.0	2	NaN	...	8.3	NaN	Всеволожск	
6827	0	4070000.0	60.0	2017-08-15T00:00:00	3	NaN	9.0	42.9	8	NaN	...	6.2	2.0	Санкт-Петербург	

5 rows × 22 columns

Обзор данных

Выведем общую информацию о таблице:

In [2]:
```
realty.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
```

Всего таблица состоит из 23699 строк, 22 столбцов и содержит данные следующих типов: bool(2 столбца), float64(14 столбцов), int64(3 столбца), object(3 столбца).

Каждая строка соответствует описанию одного объекта недвижимости.

Столбцы `cityCenters_nearest`, `parks_around3000`, `ponds_around3000` озаглавлены в смешаном стиле. На этапе предобработки следует их переименовать.

Сопоставим предоставленную документацию к таблице с выявленными фактическими типами данных:

- airports_nearest — расстояние до ближайшего аэропорта в метрах (м) - float64
- balcony — число балконов - float64
- ceiling_height — высота потолков (м) - float64
- cityCenters_nearest — расстояние до центра города (м) - float64
- days_exposition — сколько дней было размещено объявление (от публикации до снятия) - float64
- first_day_exposition — дата публикации - object
- floor — этаж - int64
- floors_total — всего этажей в доме - float64
- is_apartment — апартаменты (булев тип) - object
- kitchen_area — площадь кухни в квадратных метрах ($м^2$) - float64
- last_price — цена на момент снятия с публикации - float64
- living_area — жилая площадь в квадратных метрах ($м^2$) - float64
- locality_name — название населённого пункта - object
- open_plan — свободная планировка (булев тип) - bool
- parks_around3000 — число парков в радиусе 3 км - float64
- parks_nearest — расстояние до ближайшего парка (м) - float64
- ponds_around3000 — число водоёмов в радиусе 3 км - float64
- ponds_nearest — расстояние до ближайшего водоёма (м) - float64
- rooms — число комнат - int64
- studio — квартира-студия (булев тип) - bool
- total_area — площадь квартиры в квадратных метрах ($м^2$) - float64
- total_images — число фотографий квартиры в объявлении - int64

Из сопоставления видно, что *не все фактические типы данных соответствуют подразумеваемым*. На этапе предобработки целесообразно осуществить следующее преобразование типов.

1. К типу **bool** - столбец `is_apartment`, представленный в строковом типе.
2. К типу **int64** - столбцы:
   - `ponds_around3000`, т.к. количество водоёмов не может выражаться вещественным числом;
   - `parks_around3000`, т.к. количество парков не может выражаться вещественным числом;
   - `floors_total`, т.к. количество этажей не может выражаться вещественным числом;
   - `days_exposition` - несмотря на то, что количество дней может выражаться вещественным числом, в контексте исследования нас скорее будет интересовать целочисленная длинельность размещения, что позволит при необходимости категоризовать данное поле для удобства исследования;
   - `balcony`, т.к. количество балконов не может выражаться вещественным числом.

1. К типу **datetime64** - столбец `first_day_exposition`, содержащий даты, выраженные строками.

Разделим столбцы на заполняемые пользователем и рассчитываемые автоматически (это позволит преположить характер возникновения аномалий в данных):

- столбцы, заполняемые пользователем:
  - balcony
  - ceiling_height
  - floor
  - floors_total
  - is_apartment
  - kitchen_area
  - living_area
  - locality_name
  - open_plan
  - rooms
  - studio
  - total_area
- столбцы, заполняемые автоматически:
  - airports_nearest
  - cityCenters_nearest
  - days_exposition
  - first_day_exposition
  - last_price
  - parks_around3000
  - parks_nearest
  - ponds_around3000
  - ponds_nearest
  - total_images

Также в таблице наблюдаются пропуски в ряде колонок данных. Оценим количество этих пропусков в разных колонках.

In [3]:
```
realty.isna().sum()
```

Out[3]:
```
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
```

На первый взгляд пропусков много, в некоторых столбцах - более половины. Отсортируем их в процентном выражении:

In [4]:
```
(pd.DataFrame(
    round((realty.isna().mean()*100),2), columns=['NaNs, %'])
    .sort_values(by='NaNs, %', ascending=False)
    .style.format('{:.2f}')
    .background_gradient('coolwarm')
)
```

Out[4]:

	NaNs, %
is_apartment	88.29
parks_nearest	65.91
ponds_nearest	61.56
balcony	48.61
ceiling_height	38.80
airports_nearest	23.38
cityCenters_nearest	23.29
ponds_around3000	23.28
parks_around3000	23.28
days_exposition	13.42
kitchen_area	9.61
living_area	8.03
floors_total	0.36
locality_name	0.21
total_images	0.00
last_price	0.00
studio	0.00
floor	0.00
rooms	0.00
first_day_exposition	0.00
total_area	0.00
open_plan	0.00

Мы получили отсортированный список столбцов, в которых есть пропуски.

Замечание: Сразу стоит отметить 13.42% пропусков в столбце `'days_exposition'`. Это скорее всего свидетельствует, что данные квартиры не проданы. Следовательно их цена - не окончательная и не может использоваться для целей настоящего исследования.

Ознакомимся со значениями в порядке убывания процента пропусков и определим, каким способом и насколько целесообразно данные пропуски устранять. Начнём с булевского столбца:

In [5]:
```
realty['is_apartment'].unique()
```

Out[5]:
```
array([nan, False, True], dtype=object)
```

In [6]:
```
realty['is_apartment'].value_counts()
```

Out[6]:
```
False    2725
True       50
Name: is_apartment, dtype: int64
```

Столбец `'is_apartment'` имеет тип `object` и содержит 50 значений `True` и 2725 значений `False`. Тип `object` определяется наличием в нём пропусков, которые, очевидно, не являются булевскими значениями.

Для количественных столбцов, содержащих пропуски, охарактеризуем их значения с целью предварительного выявления возможных аномальных значений:

In [7]:
```
(
realty[['parks_nearest', 'ponds_nearest', 'balcony',
        'ceiling_height', 'airports_nearest', 'cityCenters_nearest',
        'ponds_around3000', 'parks_around3000', 'days_exposition',
        'kitchen_area', 'living_area', 'floors_total']]
    .describe()
)
```

Out[7]:

	parks_nearest	ponds_nearest	balcony	ceiling_height	airports_nearest	cityCenters_nearest	ponds_around3000	parks_around3000	days_exposition	kitchen_area	living_a
count	8079.000000	9110.000000	12180.000000	14504.000000	18157.000000	18180.000000	18181.000000	18181.000000	20518.000000	21421.000000	21796.000
mean	490.804555	517.980900	1.150082	2.771499	28793.672193	14191.277833	0.770255	0.611408	180.888634	10.569807	34.457
std	342.317995	277.720643	1.071300	1.261056	12630.880622	8608.386210	0.938346	0.802074	219.727988	5.905438	22.030
min	1.000000	13.000000	0.000000	1.000000	0.000000	181.000000	0.000000	0.000000	1.000000	1.300000	2.000
25%	288.000000	294.000000	0.000000	2.520000	18585.000000	9238.000000	0.000000	0.000000	45.000000	7.000000	18.600
50%	455.000000	502.000000	1.000000	2.650000	26726.000000	13098.500000	1.000000	0.000000	95.000000	9.100000	30.000
75%	612.000000	729.000000	2.000000	2.800000	37273.000000	16293.000000	1.000000	1.000000	232.000000	12.000000	42.300
max	3190.000000	1344.000000	5.000000	100.000000	84869.000000	65968.000000	3.000000	3.000000	1580.000000	112.000000	409.700

Предварительно можно констатировать, что указанные объекты недвижимости:

- располагаются на расстоянии от 1 метра до более чем 3 километров от парков (возможны аномально большие выбросы, поскольку парк представляет собой локальный объект городской инфраструктуры; выбросы скорее всего связаны с автоматическим заполнением данного поля для небольших населённых пунктов Ленинградской области расстоянием до ближайшего парка Санкт-Петербурга);
- располагаются на расстоянии от 13 метров до более чем 1.3 километра от водоёмов;
- располагаются на расстоянии от 0 метров до почти 84 километров от аэропорта (очевидно наличие аномальных значений);
- располагаются на расстоянии от 181 метра до почти 66 километров от центра города (очевидно наличие аномальных значений, связанных, вероятно, с некорректным рассчётом расстояния);
- имеют до 3 водоёмов и до 3 парков в радиусе 3 километров;
- экспонировались от 1 до 1580 дней (до 4.3 года).

Владельцы объектов недвижимости указали, что они обладают следующими характеристиками:

- имеют от 0 до 5 балконов (возможно наличие аномальных значений);
- высота потолков составляет от 1 до 100 метров (очевидно наличие аномальных значений);
- имеют жилую площадь от 2 до почти 410 кв.м., площадь кухни - от 1.3 до 112 кв.м. (в обоих случаях очевидно наличие аномальных значений);
- расположены в домах высотой от 1 до 60 этажей (60 этажей представляются неправдоподобными для жилой недвижимости Санкт-Петербурга, самый высокий жилой комплекс «Князь Александр Невский» возвышается всего на 37 этажей).

Дополнительным аргументом в пользу возможного наличия аномальных значений являются большие значения стандартного отклонения.

Последним столбцом, содержащим пропуски, является `'locality_name'` - название населённого пункта. Выведем список его уникальных значений:

In [8]:
```
realty['locality_name'].sort_values().unique()
```

Out[8]:
```
array(['Бокситогорск', 'Волосово', 'Волхов', 'Всеволожск', 'Выборг',
       'Высоцк', 'Гатчина', 'Зеленогорск', 'Ивангород', 'Каменногорск',
       'Кингисепп', 'Кириши', 'Кировск', 'Колпино', 'Коммунар',
       'Красное Село', 'Кронштадт', 'Кудрово', 'Лодейное Поле',
       'Ломоносов', 'Луга', 'Любань', 'Мурино', 'Никольское',
       'Новая Ладога', 'Отрадное', 'Павловск', 'Петергоф', 'Пикалёво',
       'Подпорожье', 'Приморск', 'Приозерск', 'Пушкин', 'Санкт-Петербург',
       'Светогорск', 'Сертолово', 'Сестрорецк', 'Сланцы', 'Сосновый Бор',
       'Сясьстрой', 'Тихвин', 'Тосно', 'Шлиссельбург',
       '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', nan],
      dtype=object)
```

In [9]:
```
# Количество различных населённых пунктов
len(realty['locality_name'].unique())
```

Out[9]:
```
365
```

In [10]:
```
# Количество пропусков населённых пунктов
realty['locality_name'].isna().sum()
```

Out[10]:
```
49
```

Предварительно можно констатировать, что названия 364 населённых пунктов (без учёта 49 пропусков - `NaN`) за редким исключением содержат их тип. Возможно, целесообразно выделить тип населенного пункта в виде отдельной категории.

Осталось предварительно рассмотреть значения столбцов, *не содержащих* пропусков.

Для количественных столбцов, не содержащих пропуски, охарактеризуем их значения с целью предварительного выявления возможных аномальных значений:

In [11]:
```
(
realty[['total_images', 'last_price',
        'floor', 'rooms', 'total_area']]
    .describe()
)
```

Out[11]:

	total_images	last_price	floor	rooms	total_area
count	23699.000000	2.369900e+04	23699.000000	23699.000000	23699.000000
mean	9.858475	6.541549e+06	5.892358	2.070636	60.348651
std	5.682529	1.088701e+07	4.885249	1.078405	35.654083
min	0.000000	1.219000e+04	1.000000	0.000000	12.000000
25%	6.000000	3.400000e+06	2.000000	1.000000	40.000000
50%	9.000000	4.650000e+06	4.000000	2.000000	52.000000
75%	14.000000	6.800000e+06	8.000000	3.000000	69.900000
max	50.000000	7.630000e+08	33.000000	19.000000	900.000000

Предварительно можно констатировать, что указанные объекты недвижимости:

- представлены от 0 до 50 фотографиями;
- имеют последнюю зафиксированную цену в диапазоне от 12 с небольшим тысяч до 763 миллионов денежных единиц, возможно, рублей (в данном столбце в зависимости от вида валюты могут быть аномалии как внизу, так и вверху диапазона).

Владельцы объектов недвижимости указали, что они обладают следующими характеристиками:

- расположены на 1-33 этаже здания;
- содержат от 0 до 19 комнат (очевидно наличие аномальных значений);
- имеют общую площадь от 12 до 900 кв.м. (очевидно наличие аномалий на верхней границе значений).

Дополнительным аргументом в пользу возможного наличия аномальных значений являются большие значения стандартного отклонения.

Булевы столбцы `'studio'` и `'open_plan'` не содержат аномалий, поскольку имеют правильный тип данных и в них отсутствуют пропуски. Столбец `'first_day_exposition'` перед анализом необходимо привести к корректному типу данных.

Выводы

1. В оформлении столбцов представленной таблицы допущено нарушение "хорошего стиля", мешающее наглядности восприятия их названий.

1. 13.42% пропусков в столбце `'days_exposition'` скорее всего свидетельствуют о том, что данные квартиры не проданы. Следовательно их цена - не окончательная и не может использоваться для целей настоящего исследования.

1. Строки таблицы содержат данные по 22 характеристикам некоторого объекта недвижимости:
   - Часть столбцов ('balcony', 'ceiling_height', 'floor', 'floors_total', 'is_apartment', 'kitchen_area', 'living_area', 'locality_name', 'open_plan', 'rooms', 'studio', 'total_area') заполняются пользователем при публикации объявления.
   - Другая часть ('airports_nearest', 'cityCenters_nearest', 'days_exposition', 'first_day_exposition', 'last_price', 'parks_around3000', 'parks_nearest', 'ponds_around3000', 'ponds_nearest', 'total_images') заполняется автоматически на основе картографических и введённых пользователем данных.

1. В 14 столбцах наблюдаются пропуски данных, составляющие от 0.21% до 88.29% от общего количества строк в зависимости от столбца.

1. В 2 столбцах таблицы требуется провести преобразование типов:
   - 'is_apartment' - к булевскому типу;
   - 'first_day_exposition' - к типу даты.

1. Кроме того, 5 вещественных столбцов ('ponds_around3000', 'parks_around3000', 'floors_total', 'days_exposition', 'balcony') целесообразно привести к целому типу данных.

1. Предварительно, с разной степенью уверенности, можно подозревать аномалии данных в следующих 11 столбцах: 'parks_nearest', 'airports_nearest', 'cityCenters_nearest', 'ceiling_height', 'balcony', 'living_area', 'kitchen_area', 'floors_total', 'last_price', 'rooms', 'total_area'. *Дополнительным аргументом в пользу возможного наличия аномальных значений являются большие значения стандартного отклонения.*

1. Также предварительно можно констатировать, что названия 'locality_name' 365 населённых пунктов за редким исключением содержат их тип. Возможно, целесообразно выделить тип населенного пункта в виде отдельной категории.

Замечание. Вместе с тем, инженерам по данным следует указать на неполноту описания данных (не указана валюта, в которой выражена стоимость объектов). Данный фактор может влиять на интерпретацию значений стоимости 'last_price'. В дальнейшем в исследовании будем предполагать, что речь идёт о российских рублях.

Предобработка данных

Переименование столбцов

Предобработку данных начнём с исправления наименований столбцов, озаглавленных в смешаном стиле.

In [12]:
```
realty.columns
```

Out[12]:
```
Index(['total_images', 'last_price', 'total_area', 'first_day_exposition',
       'rooms', 'ceiling_height', 'floors_total', 'living_area', 'floor',
       'is_apartment', 'studio', 'open_plan', 'kitchen_area', 'balcony',
       'locality_name', 'airports_nearest', 'cityCenters_nearest',
       'parks_around3000', 'parks_nearest', 'ponds_around3000',
       'ponds_nearest', 'days_exposition'],
      dtype='object')
```

Смешаный стиль наблюдается в оглавлении столбцов `'cityCenters_nearest'`, `'parks_around3000'`, `'ponds_around3000'`. Переименуем их в хорошем стиле и выведем повторно список для проверки:

In [13]:
```
realty = realty.rename(
    columns={
        'cityCenters_nearest' : 'city_centers_nearest',
        'parks_around3000' : 'parks_around_3000',
        'ponds_around3000' : 'ponds_around_3000'
    }
)
realty.columns
```

Out[13]:
```
Index(['total_images', 'last_price', 'total_area', 'first_day_exposition',
       'rooms', 'ceiling_height', 'floors_total', 'living_area', 'floor',
       'is_apartment', 'studio', 'open_plan', 'kitchen_area', 'balcony',
       'locality_name', 'airports_nearest', 'city_centers_nearest',
       'parks_around_3000', 'parks_nearest', 'ponds_around_3000',
       'ponds_nearest', 'days_exposition'],
      dtype='object')
```

Теперь все столбцы поименованы в хорошем стиле. Перейдём к устранению аномалий и пропусков в данных.

Устранение пропусков в данных

Как было отмечено в обзоре данных, 13.42% строк с пропусками в столбце `'days_exposition'` не могут использоваться для целей настоящего исследования. Удалим их:

In [14]:
```
realty.dropna(subset=['days_exposition'], inplace=True)
```

Обрабатывать столбцы будем в порядке убывания количества пропусков. Больше всего пропусков (более 88%) на этапе обзора выявлено в столбце `'is_apartment'`.

В описании данных столбец `'is_apartment'` имеет тип object. Попытки преобразовать его вручную к типу `bool` с сохранением пропусков показали, что именно пропуски меняют тип столбца.

По современным данным портала Яндекс.Недвижимость подавляющее большинство объектов недвижимости на рынке Санкт-Петербурга и ленинградской области не является апартаментами.

Поэтому логично предположить, что если это поле не заполнено, то объект недвижимости не является апартаментами. Заполняем пропуски значением `False`:

In [15]:
```
realty['is_apartment'] = realty['is_apartment'].fillna(False)
print(realty['is_apartment'].unique())
# Проверка
realty['is_apartment'].isna().sum()
```

```
[False  True]
```

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

Пропуски в столбце `'is_apartment'` заполнены. Рассмотрим теперь столбцы с наименьшим количеством пропусков, согласно результатов обзора данных:

- `'locality_name'` - 0.21% пропусков;
- `'floors_total'` - 0.36% пропусков.

Данные пропуски носят единичный характер, а кроме того, сложно анализировать недвижимость, которая находится неизвестно где. Целесообразно заполнить `'locality_name'` некоторым стандартным значением и полностью удалить строки с пропусками в `'floors_total'`.

Замечание. *Поскольку данные поля заполняются пользователем сервиса недвижимости, целесообразно рекомендовать команде сервиса сделать их обязательными для заполнения (поле `'locality_name'`), либо привязать к значениям других полей и задать значение по умолчанию (поле `'floors_total'`).*

Заполним пропуски названий некоторым стандартным значением:

In [16]:
```
realty['locality_name'].fillna('Unknown', inplace=True)
realty['locality_name'].isna().sum()
```

Out[16]:
```
0
```

Удалим пропуски в `'floors_total'`:

In [17]:
```
realty.dropna(subset=['floors_total'], inplace=True)
realty['floors_total'].isna().sum()
```

Out[17]:
```
0
```

Замечание. *Поскольку нет привязки к адресу домов, исследовать `'floors_total'` на аномалии не представляется возможным.*

Среди столбцов с пропусками можно выделить `'balcony'`, в котором достаточно легко предположить, чем данные пропуски следует заполнять: если количество балконов не указано, то скорее всего их нет - проставляем нули:

In [18]:
```
realty['balcony'] = realty['balcony'].fillna(0)
realty['balcony'].sort_values().unique()
```

Out[18]:
```
array([0., 1., 2., 3., 4., 5.])
```

Ситуация с 3, 4 и 5 балконами выглядит аномальной.

Очевидно, количество балконов зависит от количества комнат в квартире, которое, в свою очередь, зависит от общей площади.

Таким образом, прежде чем изучать аномалии в количестве балконов необходимо изучить данные в столбцах `'total_area'`, `'rooms'`. Это мы будем делать позже, в при исследовании данных. Поэтому отложим возможное устранение аномалий с балконами.

Особое внимание привлекают пропуски в кластере полей `'ponds_around_3000'`, `'ponds_nearest'`, `'parks_around_3000'`, `'parks_nearest'`, а также `'airports_nearest'` и `'city_centers_nearest'`, содержащих данные картографического сервиса. Выясним, связаны ли они.

Меньше всего пропусков в столбцах `'ponds_around_3000'` и `'parks_around_3000'` (по 23.28%, согласно обзору данных). Выделим подмножество строк, где значения `'ponds_around_3000'` пропущены:

In [19]:
```
no_ponds_3000 = realty[realty['ponds_around_3000'].isna()]
len(no_ponds_3000)
```

Out[19]:
```
4644
```

В 4644 строках пропущены значения в `'ponds_around_3000'`. Проверим, сколько среди них пропущено значений в столбцах `'ponds_nearest'`, `'parks_nearest'`, `'parks_around_3000'`, `'airports_nearest'` и `'city_centers_nearest'`:

In [20]:
```
(
no_ponds_3000[['ponds_nearest', 'parks_nearest', 'parks_around_3000',
               'airports_nearest', 'city_centers_nearest']]
    .isna().sum()
)
```

Out[20]:
```
ponds_nearest           4644
parks_nearest           4644
parks_around_3000       4644
airports_nearest        4644
city_centers_nearest    4644
dtype: int64
```

Таким образом, *в выделенных строках (`no_ponds_3000`) одновременно пропущены значения в столбцах `'ponds_around_3000'`, `'ponds_nearest'`, `'parks_around_3000'`, `'parks_nearest'`, а также `'airports_nearest'` и `'city_centers_nearest'`.*

Данная ошибка, безусловно, носит технологический характер. Вероятные причины - сбой в работе картографического сервиса или ошибка извлечения данных из него.

Следует уведомить команду разработки картографического сервиса и инженеров по данным.

Для исправления указанных 4644 пропусков у нас нет никаких данных, однако они составляют около 23% от объёма исходной таблицы:

In [21]:
```
round(4644 * 100 / len(realty), 2)
```

Out[21]:
```
22.72
```

Поэтому пока данные пропуски придётся оставить. Но можно попытаться заполнить данные столбцы хотя бы частично.

Однако, прежде проверим, какие данные соответствуют пропускам в `'ponds_nearest'` и `'parks_nearest'`:

In [22]:
```
realty[realty['ponds_nearest'].isna()]['ponds_around_3000'].unique()
```

Out[22]:
```
array([ 0., nan])
```

In [23]:
```
realty[realty['parks_nearest'].isna()]['parks_around_3000'].unique()
```

Out[23]:
```
array([ 0., nan])
```

Очевидно, пропуски соответствуют ситуации, когда в 3-километровом радиусе отсутствуют пруды или парки, либо когда данные о таких объектах отсутствуют. Проверим, можем ли мы заполнить эти пропуски - сгруппируем данные по населённому пункту и количеству парков или водоёмов, рассчитаем медианы расстояний и выведем и выведем медианы, соответствующие пропускам в `'ponds_nearest'` и `'parks_nearest'`:

In [24]:
```
ponds_nearest_madians = realty.groupby(by=['locality_name', 'ponds_around_3000']) \
                              ['ponds_nearest'].median()
ponds_nearest_madians[ponds_nearest_madians.isna()]
```

Out[24]:
```
locality_name         ponds_around_3000
Unknown               0.0                 NaN
Зеленогорск           0.0                 NaN
Колпино               0.0                 NaN
Красное Село          0.0                 NaN
Кронштадт             0.0                 NaN
Ломоносов             0.0                 NaN
Павловск              0.0                 NaN
Петергоф              0.0                 NaN
Пушкин                0.0                 NaN
Санкт-Петербург       0.0                 NaN
Сестрорецк            0.0                 NaN
поселок Лисий Нос     0.0                 NaN
посёлок Левашово      0.0                 NaN
посёлок Лисий Нос     0.0                 NaN
посёлок Металлострой  0.0                 NaN
посёлок Молодёжное    0.0                 NaN
посёлок Парголово     0.0                 NaN
посёлок Песочный      0.0                 NaN
посёлок Понтонный     0.0                 NaN
посёлок Репино        0.0                 NaN
посёлок Стрельна      0.0                 NaN
посёлок Усть-Ижора    0.0                 NaN
посёлок Шушары        0.0                 NaN
Name: ponds_nearest, dtype: float64
```

In [25]:
```
parks_nearest_madians = realty.groupby(by=['locality_name', 'parks_around_3000']) \
                              ['parks_nearest'].median()
parks_nearest_madians[parks_nearest_madians.isna()]
```

Out[25]:
```
locality_name            parks_around_3000
Unknown                  0.0                 NaN
Зеленогорск              0.0                 NaN
Колпино                  0.0                 NaN
Красное Село             0.0                 NaN
Кронштадт                0.0                 NaN
Ломоносов                0.0                 NaN
Павловск                 0.0                 NaN
Петергоф                 0.0                 NaN
Пушкин                   0.0                 NaN
Сестрорецк               0.0                 NaN
поселок Лисий Нос        0.0                 NaN
посёлок Александровская  0.0                 NaN
посёлок Белоостров       0.0                 NaN
посёлок Левашово         0.0                 NaN
посёлок Лисий Нос        0.0                 NaN
посёлок Металлострой     0.0                 NaN
посёлок Молодёжное       0.0                 NaN
посёлок Парголово        0.0                 NaN
посёлок Песочный         0.0                 NaN
посёлок Петро-Славянка   0.0                 NaN
посёлок Понтонный        0.0                 NaN
посёлок Репино           0.0                 NaN
посёлок Сапёрный         0.0                 NaN
посёлок Стрельна         0.0                 NaN
посёлок Усть-Ижора       0.0                 NaN
посёлок Шушары           0.0                 NaN
Name: parks_nearest, dtype: float64
```

Нулевым позициям в `'ponds_around_3000'` и `'parks_around_3000'` невозможно поставить в соответствие медианное значение до ближайшего пруда или парка.

Поэтому пропуски в этих столбцах заполнять не будем.

Однако, медианными значениями можно попробовать заполнить расстояния до аэропорта и до центра, основываясь на группировке по населённому пункту (предварительно исключив из рассчёта медианы описанные выше массовые пропуски во множестве связанных столбцов):

In [26]:
```
airports_nearest_medians = realty\
                       .dropna(subset=['ponds_around_3000'])\
                       .groupby('locality_name')['airports_nearest']\
                       .median()
print('Всего медиан по городам', len(airports_nearest_medians))
print('Пустых медиан по городам', airports_nearest_medians.isna().sum())
airports_nearest_medians
```

```
Всего медиан по городам 27
Пустых медиан по городам 0
```

Out[26]:
```
locality_name
Unknown                  22669.0
Зеленогорск              72284.0
Колпино                  26232.0
Красное Село             25717.0
Кронштадт                67847.0
Ломоносов                48466.0
Павловск                 20502.0
Петергоф                 39187.5
Пушкин                   15745.0
Санкт-Петербург          26806.5
Сестрорецк               56914.0
поселок Лисий Нос        55930.0
посёлок Александровская  12896.5
посёлок Белоостров       57769.0
посёлок Левашово         52693.0
посёлок Лисий Нос        54988.5
посёлок Металлострой     25735.0
посёлок Молодёжное       84006.0
посёлок Парголово        53426.0
посёлок Песочный         54831.5
посёлок Петро-Славянка   20605.0
посёлок Понтонный        30642.0
посёлок Репино           62111.0
посёлок Сапёрный         34134.0
посёлок Стрельна         28317.0
посёлок Усть-Ижора       26510.5
посёлок Шушары           17347.0
Name: airports_nearest, dtype: float64
```

In [27]:
```
city_centers_nearest_medians = realty\
                               .dropna(subset=['ponds_around_3000'])\
                               .groupby('locality_name')['city_centers_nearest']\
                               .median()
print('Всего медиан по городам', len(city_centers_nearest_medians))
print('Пустых медиан по городам', city_centers_nearest_medians.isna().sum())
city_centers_nearest_medians
```

```
Всего медиан по городам 27
Пустых медиан по городам 0
```

Out[27]:
```
locality_name
Unknown                    8781.0
Зеленогорск               53383.0
Колпино                   32018.0
Красное Село              29142.0
Кронштадт                 49572.5
Ломоносов                 51730.0
Павловск                  32655.0
Петергоф                  32999.0
Пушкин                    27898.0
Санкт-Петербург           12309.0
Сестрорецк                34821.0
поселок Лисий Нос         28246.0
посёлок Александровская   27468.0
посёлок Белоостров        38868.0
посёлок Левашово          25727.0
посёлок Лисий Нос         27297.0
посёлок Металлострой      27482.0
посёлок Молодёжное        65105.0
посёлок Парголово         19301.5
посёлок Песочный          26099.5
посёлок Петро-Славянка    27165.0
посёлок Понтонный         32354.0
посёлок Репино            43210.0
посёлок Сапёрный          35846.0
посёлок Стрельна          23506.0
посёлок Усть-Ижора        28222.5
посёлок Шушары            24168.5
Name: city_centers_nearest, dtype: float64
```

Здесь ***следует отметить***, что лишь в 26 (не считая Unknown) из 364 городов, не искажённых массовым сбоем работы с картографическим сервисом, мы можем рассчитать медиану расстояния до аэропорта и центра города.

Составим функцию для заполнения пропусков в `'airports_nearest'`:

In [28]:
```
# Определение функции airports_nearest_fill_nans для замены пропусков
# в столбце 'airports_nearest'
#=====================================================================
# На вход функции подаётся строка таблицы, содержащая:
#   - текущее значение расстояния 'airports_nearest'
#   - название населённого пункта 'locality_name'
# Функция возвращает откорректированное значение расстояния
def airports_nearest_fill_nans(row):
    airports_nearest = row['airports_nearest']
    locality_name = row['locality_name']

    try:
        if math.isnan(airports_nearest):
            airports_nearest = airports_nearest_medians[locality_name]
    except:
        return airports_nearest
    return airports_nearest

# Протестируем функцию
# 1. Зададим тестовый набор
row_values = [math.nan, 'Колпино'] # значения
row_columns = ['airports_nearest', 'locality_name'] #названия столбцов
# 2. Сформируем строку
r = pd.Series(data=row_values, index=row_columns)
# Вызовем функцию
airports_nearest_fill_nans(r)
```

Out[28]:
```
26232.0
```

Функция работает корректно. Применим её для заполнения пропусков в столбце `'airports_nearest'`:

In [29]:
```
print("Количество пропусков в 'airports_nearest' до правки:", realty['airports_nearest'].isna().sum())

# Заполнение пропусков
realty['airports_nearest'] = realty.apply(airports_nearest_fill_nans, axis=1)

# Проверка результатов заполнения
print("Количество пропусков в 'airports_nearest' после правки:", realty['airports_nearest'].isna().sum())
```

```
Количество пропусков в 'airports_nearest' до правки: 4666
Количество пропусков в 'airports_nearest' после правки: 4605
```

Данным действием мы исправили порядка 1.3% пропусков. Вообще говоря, этого можно было бы и не делать. Но для концептуальной целостности проделаем то же самое с пропусками в `'city_centers_nearest'`.

Составим функцию для заполнения пропусков:

In [30]:
```
# Определение функции city_centers_nearest_fill_nans для замены пропусков
# в столбце 'city_centers_nearest'
#=====================================================================
# На вход функции подаётся строка таблицы, содержащая:
#   - текущее значение расстояния 'city_centers_nearest'
#   - название населённого пункта 'locality_name'
# Функция возвращает откорректированное значение расстояния
def city_centers_nearest_fill_nans(row):
    city_centers_nearest = row['city_centers_nearest']
    locality_name = row['locality_name']

    try:
        if math.isnan(city_centers_nearest):
            city_centers_nearest = city_centers_nearest_medians[locality_name]
    except:
        return city_centers_nearest
```

```
    return city_centers_nearest

# Протестируем функцию
# 1. Зададим тестовый набор
row_values = [math.nan, 'Колпино'] # значения
row_columns = ['city_centers_nearest', 'locality_name'] #названия столбцов

# 2. Сформируем строку
r = pd.Series(data=row_values, index=row_columns)

# Вызовем функцию
city_centers_nearest_fill_nans(r)
```

Out[30]:
```
32018.0
```

Функция работает корректно. Применим её для заполнения пропусков в столбце `'city_centers_nearest'`:

In [31]:
```
print("Количество пропусков в 'city_centers_nearest' до правки:",
      realty['city_centers_nearest'].isna().sum())

# Заполнение пропусков
realty['city_centers_nearest'] = realty.apply(city_centers_nearest_fill_nans,
                                              axis=1)
# Проверка результатов заполнения
print("Количество пропусков в 'city_centers_nearest' после правки:",
      realty['city_centers_nearest'].isna().sum())
```

```
Количество пропусков в 'city_centers_nearest' до правки: 4644
Количество пропусков в 'city_centers_nearest' после правки: 4605
```

Данным действием мы исправили порядка 0.8% пропусков.

Осталось решить, что делать с пропусками в столбцах `'ceiling_height'`, `'kitchen_area'`, `'living_area'`. Данные столбцы можно было бы попытаться заполнить, если бы была более точная привязка к адресу, а не только к названию населённого пункта. Кроме того, на этапе обзора, как мы помним, в данных былывыявлены аномально высокие потолки и большие или, наоборот, слишком маленькие площади. Поэтому заполнять пропуски в данных столбцах также не целесообразно.

Анализировать аномалии в таблице мы будем позже, на этапе исследовательсткого анализа. Сейчас же проведём преобразование типов данных в таблице в соответствии с рекомендациями, сформулированными на этапе обзора.

Преобразование типов данных

Предположение о неверном типе данных в столбце `'is_apartment'` не подтвердилось. Тип object столбец принимал по причине наличия в нём пропусков данных (т.к. тип `NaN` не является булевским). Таким образом, осталось привести типы данных в следующих столбцах:

1. К типу **int64** - столбцы:
   - `'ponds_around_3000'`, т.к. количество водоёмов не может выражаться вещественным числом;
   - `'parks_around_3000'`, т.к. количество парков не может выражаться вещественным числом;
   - `'floors_total'`, т.к. количество этажей не может выражаться вещественным числом;
   - `'days_exposition'` - несмотря на то, что количество дней может выражаться вещественным числом, в контексте исследования нас скорее будет интересовать целочисленная длинельность размещения, что позволит при необходимости категоризовать данное поле для удобства исследования;
   - `'balcony'`, т.к. количество балконов не может выражаться вещественным числом.

1. К типу **datetime64** - столбец `'first_day_exposition'`, содержащий даты, выраженные строками.

Начнём с числовых столбцов. Методом .astype('int') можно привести тип только для столбцов, не содержащих пропуски:

In [32]:
```
# Приведение столбца 'floors_total' к целочисленным значениям
realty['floors_total'] = realty['floors_total'].astype('int')

# Приведение столбца 'balcony' к целочисленным значениям
realty['balcony'] = realty['balcony'].astype('int')

# Проверим результат
print(realty['floors_total'].dtype)
print(realty['balcony'].dtype)
```

```
int64
int64
```

Для приведения остальных столбцов (`'ponds_around_3000'`, `'parks_around_3000'`, `'days_exposition'`) напишем функцию:

In [33]:
```
# Определение функции to_int_skip_nans для преобразования типа к int
#=====================================================================
# На вход функции подаётся значение для преобразования типов
# Функция возвращает значение в преобразованном типе или NaN
def to_int_skip_nans(value):
    try:
        ret = int(value)
    except:
        ret = value
    return ret

# Протестируем функцию
# 1. Зададим тестовый набор
test_values = [math.nan, 34.5, 0.23] # значения
# 2. Вызовем функцию
for val in test_values:
    print(f'Значению {val} соответствует выход: {to_int_skip_nans(val)}')
```

```
Значению nan соответствует выход: nan
Значению 34.5 соответствует выход: 34
Значению 0.23 соответствует выход: 0
```

Функция работает корректно, оставляя NaN на месте. Применим её к столбцам `'ponds_around_3000'`, `'parks_around_3000'`, `'days_exposition'`:

In [34]:
```
columns = ['ponds_around_3000', 'parks_around_3000', 'days_exposition']

for col in columns:
    realty[col] = realty[col].apply(to_int_skip_nans)
    print(realty[col].dtype)
```

```
float64
float64
int64
```

Очевидно, наличие NaN определяет вещественный тип столбца. Поэтому приведение типов сработало только на столбце `'days_exposition'`. Оставим численные столбцы и перейдём к корректировке дат в `'first_day_exposition'`:

In [35]:
```
realty['first_day_exposition'] = pd.to_datetime(realty['first_day_exposition'],
                                                format='%Y-%m-%dT%H:%M:%S')
realty.head(2)
```

Out[35]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcony	locality_name	airports_neare
1	7	3350000.0	40.4	2018-12-04	1	NaN	11	18.6	1	False	...	11.0	2	посёлок Шушары	12817
2	10	5196000.0	56.0	2015-08-20	2	NaN	5	34.3	4	False	...	8.3	0	Санкт-Петербург	21741

2 rows × 22 columns

Даты успешно преобразованы.

Выводы

В ходе предварительной обработки данных нами были проведены:

- Переименование столбцов к хорошему стилю.
- Устранение пропусков данных.
- Преобразование типов данных.

1. В ходе устранения пропусков часть строк с пропусками в `'days_exposition'`, `'locality_name'` и `'floors_total'` удалена полностью.

Замечание. *Поскольку данные поля заполняются пользователем сервиса недвижимости, целесообразно рекомендовать команде сервиса сделать их обязательными для заполнения (поле `'locality_name'`), либо привязать к значениям других полей и задать значение по умолчанию (поле `'floors_total'`).*

1. Пропуски в столбцах `'is_apartment'`, `'balcony'` устранены полностью, исходя из здравого смысла.

1. В 4644 строках датасета обнаружено одновременное отсутствие значений в столбцах `'ponds_around_3000'`, `'ponds_nearest'`, `'parks_around_3000'`, `'parks_nearest'`, а также `'airports_nearest'` и `'city_centers_nearest'`. Данная ошибка носит технологический характер. Вероятные причины - сбой в работе картографического сервиса или ошибка извлечения данных из него.

Следует уведомить команду разработки картографического сервиса и инженеров по данным.

1. Для исправления указанных пропусков в 4644 строках у нас нет данных, хотя они составляют около 23% от объёма исходной таблицы.

1. В результате удалось заполнить медианами часть значений столбцов `'airports_nearest'` и `'city_centers_nearest'` (не более 1.3% от общего количества пропусков). Пропуски в `'ponds_around_3000'`, `'ponds_nearest'`, `'parks_around_3000'`, `'parks_nearest'` решено не заполнять.

1. Столбцы `'ceiling_height'`, `'kitchen_area'`, `'living_area'` можно было бы попытаться заполнить, если бы была более точная привязка к адресу (вплоть до дома), а не только к названию населённого пункта. Кроме того, на этапе обзора, как мы помним, в данных быливыявлены аномально высокие потолки и большие или, наоборот, слишком маленькие площади. Поэтому заполнять пропуски в данных столбцах также не целесообразно.

Замечание: в дальнейшем, в ходе исследований, возможно, дополнительные пропуски будут удалены.

1. В ходе преобразования типов приведены к формату дат значения в столбце `'first_day_exposition'`? а также к целому типу - значения столбцов `'floors_total'` и `'balcony'`.

1. Очевидно, наличие NaN определяет вещественный тип столбца. Поэтому значения столбцов `'ponds_around_3000'`, `'parks_around_3000'` к целому типу привести не удалось.

Расчёты и добавление результатов в таблицу

Для дальнейших исследований нам понадобится обогатить данные:

- ввести цену квадратного метра общей площади
- ввести день недели, месяц и год публикации объявления
- ввести категории этажа квартиры - первый, последний, другой
- узнать соотношение жилой и общей площади, а также кухни и общей площади

Расчёт цены квадратного метра

Рассчитаем цену квадратного мерта общей площади:

In [36]:
```
realty['price_per_m2'] = round(realty['last_price'] / realty['total_area'], 2)
realty[['last_price', 'price_per_m2']].head(2)
```

Out[36]:

	last_price	price_per_m2
1	3350000.0	82920.79
2	5196000.0	92785.71

Столбец цен добавлен.

Категоризация даты публикации

Введём день недели, месяц и год публикации объявления:

In [37]:
```
# Введём день недели
realty['first_exposition_weekday'] = realty['first_day_exposition'].dt.weekday
```

```
# Введём месяц
realty['first_exposition_month'] = realty['first_day_exposition'].dt.month

# Введём год
realty['first_exposition_year'] = realty['first_day_exposition'].dt.year

# Проверка
realty[['first_day_exposition', 'first_exposition_weekday',
       'first_exposition_month', 'first_exposition_year']].head(2)
```

Out[37]:

	first_day_exposition	first_exposition_weekday	first_exposition_month	first_exposition_year
1	2018-12-04	1	12	2018
2	2015-08-20	3	8	2015

Столбцы, категоризирующие дату публикации объявления добавлены.

Категоризация этажа квартиры

Введём категории этажа квартиры - первый, последний, другой. Для этого напишем функцию:

In [38]:

```
# Определение функции get_floor_cat для категоризации этажа
#=====================================================================
# На вход функции подаётся строка, содержащая
#    'floor' - номер этажа
#    'floors_total' - количество этажей
# Функция возвращает значение категории
def get_floor_cat(row):
    floors_total = row['floors_total']
    floor = row['floor']

    try:
        if floor == 1:
            ret = 'первый'
        elif floor == floors_total:
            ret = 'последний'
        else:
            ret = 'другой'
    except:
        ret = 'другой'
    return ret

# Протестируем функцию
# 1. Зададим тестовый набор
df = pd.DataFrame(
    {
        'floors_total' : pd.Series([math.nan, math.nan, 33]),
        'floor' : pd.Series([1, 2, 33])
    }
)
print(df)
# 3. Вызовем функцию
for index, r in df.iterrows():
    print(get_floor_cat(r))
```

```
   floors_total  floor
0           NaN      1
1           NaN      2
2          33.0     33
первый
другой
последний
```

Функция работает корректно. Применим её для категоризации этажей:

In [39]:

```
realty['floor_cat'] = realty.apply(get_floor_cat, axis=1)
realty[['floors_total', 'floor', 'floor_cat']].sample(5)
```

Out[39]:

	floors_total	floor	floor_cat
5445	15	14	другой
3697	4	4	последний
1460	9	7	другой
22825	19	13	другой
1479	12	5	другой

Этажи категоризированы.

Соотношения жилой площади и кухни к общей площади квартиры

Осталось рассчитать соотношения жилой и общей площади, а также кухни и общей площади. Для этого также зададим функции:

In [40]:

```
# Определение функции get_living_total_ratio отношения жилой к общей площади
#=========================================================================
# На вход функции подаётся строка, содержащая
#    'total_area' - размер общей площади
#    'living_area' - размер жилой площади
# Функция возвращает значение отношения площадей
def get_living_total_ratio(row):
    total_area = row['total_area']
    living_area = row['living_area']

    try:
        ret = round(living_area / total_area, 2)
    except:
        ret = math.nan
    return ret

# Определение функции get_kitchen_total_ratio отношения кухни к общей площади
#=========================================================================
# На вход функции подаётся строка, содержащая
```

```
#    'total_area' - размер общей площади
#    'kitchen_area' - размер кухни
# Функция возвращает значение отношения площадей
def get_kitchen_total_ratio(row):
    total_area = row['total_area']
    kitchen_area = row['kitchen_area']

    try:
        ret = round(kitchen_area / total_area, 2)
    except:
        ret = math.nan
    return ret

# Протестируем функции

# 1. Зададим тестовый набор
df = pd.DataFrame(
    {
        'kitchen_area' : pd.Series([math.nan, 12, 15]),
        'living_area' : pd.Series([20, 15, 18]),
        'total_area' : pd.Series([30, math.nan, 40])

    }
)
display(df)

# 2. Вызовем функции
df['living_ratio'] = df.apply(get_living_total_ratio, axis=1)
df['kitchen_ratio'] = df.apply(get_kitchen_total_ratio, axis=1)
df
```

	kitchen_area	living_area	total_area
0	NaN	20	30.0
1	12.0	15	NaN
2	15.0	18	40.0

Out[40]:

	kitchen_area	living_area	total_area	living_ratio	kitchen_ratio
0	NaN	20	30.0	0.67	NaN
1	12.0	15	NaN	NaN	NaN
2	15.0	18	40.0	0.45	0.38

Функции работают корректно. Применим их для добавления столбцов к таблице `realty`:

In [41]:
```
# Добавление отношения жилой к общей площади
realty['living_ratio'] = realty.apply(get_living_total_ratio, axis=1)

# Добавление отношения кухни к общей площади
realty['kitchen_ratio'] = realty.apply(get_kitchen_total_ratio, axis=1)

# Проверка
realty[['total_area', 'living_area', 'kitchen_area', 'living_ratio', 'kitchen_ratio']].sample(5)
```

Out[41]:

	total_area	living_area	kitchen_area	living_ratio	kitchen_ratio
4071	36.0	16.0	10.0	0.44	0.28
10573	39.0	29.0	5.5	0.74	0.14
13186	70.0	43.4	8.5	0.62	0.12
21196	55.9	25.0	13.0	0.45	0.23
4529	29.0	17.0	5.0	0.59	0.17

Нужные отношения добавлены в таблицу.

Выводы

1. В данном разделе мы добавили все требуемые данные в таблицу. Добавление прошло без ошибок, поскольку все функции были реализованы безопасно.

1. В дальнейшем, в ходе анализа некоторые значения, на основе которых проведено обогащение, возможно будут исправлены. **Это сдедует учесть и при необходимости повторить расчёт.**

Исследовательский анализ данных

Приступим к исследованию данных.

На данном этапе для целей исследования нам необходимо максимально очистить таблицу от аномальных значений и выбросов для получения сколько-нибудь доверенных результатов.

Чтобы иметь возможность сравнения с исходным множеством данных для дальнейших рассуждений создадим копию таблицы `realty` и созраним её в переменную `good_realty`:

In [42]:
```
good_realty = realty.copy()
good_realty.head(2)
```

Out[42]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	ponds_around_3000	ponds_nearest	days_exposition
1	7	3350000.0	40.4	2018-12-04	1	NaN	11	18.6	1	False	...	0.0	NaN	81
2	10	5196000.0	56.0	2015-08-20	2	NaN	5	34.3	4	False	...	2.0	574.0	558

2 rows × 29 columns

Исследование характеристик объектов недвижимости

Начнём с характеристик выставленных на продажу объектов недвижимости.

На первом этапе изучим следующие параметры:

- площадь - `'total_area'`,
- цена - `'last_price'`,
- число комнат - `'rooms'`,
- высота потолков - `'ceiling_height'`.

Для удобства работы создадим несколько функций:

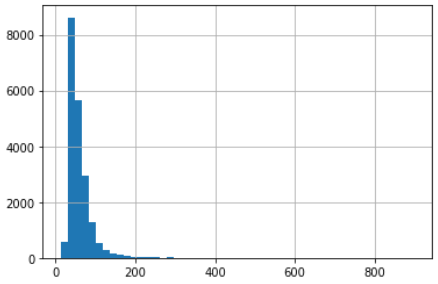
In [43]:
```
# Задание функции, характеризующей данные столбца датафрейма
# и выводящей для него гистограмму и диаграмму размаха
# Входные данные:
#     df - датафрейм
#     col - столбец
#     bin_s - количество корзин гистограммы (по умолчанию 50)
def characterize_hist_box(df, col, bin_s=50):
    # Характеристика столбца
    print(df[col].describe())
    # Построение диаграмм
    df[col].hist(bins=bin_s)
    plt.show()
    df.boxplot(column=col)
    plt.show()

# Задание функции, характеризующей данные столбца датафрейма
# и выводящей для него диаграмму размаха
# Входные данные:
#     df - датафрейм
#     col - столбец
#     bin_s - количество корзин гистограммы (по умолчанию 50)
def characterize_box(df, col):
    # Характеристика столбца
    print(df[col].describe())
    # Построение диаграмм
    df.boxplot(column=col)
    plt.show()
```

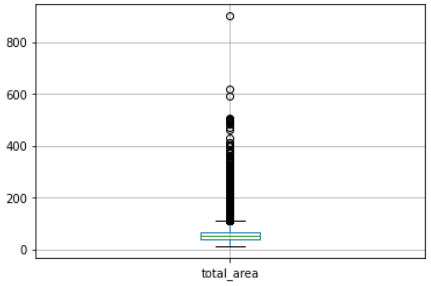
Исследование распределения общей площади квартиры

Охарактеризуем столбец `'total_area'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [44]:
```
characterize_hist_box(df=good_realty, col='total_area')
```

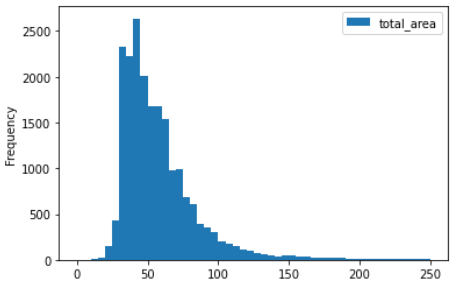
```
count    20441.000000
mean        58.857697
std         34.061930
min         12.000000
25%         39.580000
50%         50.800000
75%         67.800000
max        900.000000
Name: total_area, dtype: float64
```

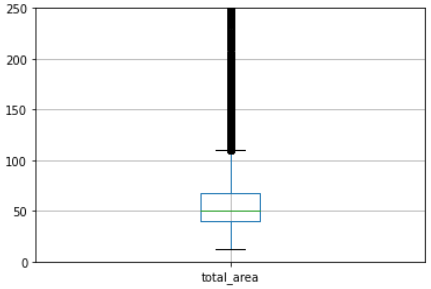




Очевидно, медиана площади квартиры находится в районе 50 кв.м., значения свыше 250 кв.м. являются единичными выбросами (их можно удалить), а достоверные значения предварительно, находятся в диапазоне от 10 до не более чем 250 кв.м.учтём это при построении диаграмм:

In [45]:
```
# Построение диаграмм
good_realty.plot(y='total_area', kind='hist', bins=50, range=(0, 250))
plt.show()
plt.ylim(0, 250)
good_realty.boxplot(column='total_area')
plt.show()
```





Распределение общей площади квартир похоже на искажённое Пуассоновское, медиана находится в районе 50 кв.м., нижний ус диаграммы размаха упирается в минимальное значение общей площади, верхний - отсекает значения свыше 110.13 кв.м., считая их выбросами:

In [46]:
```
characteristics = good_realty['total_area'].describe()
characteristics['75%'] + 1.5 *(characteristics['75%'] - characteristics['25%'])
```

Out[46]:
```
110.13
```

Статистические характеристики столбца следующие:

- средняя площадь составляет около 58 кв.м.
- медиана - 50.8 кв.м.
- 25% квартир имеют площадь до 39.58 кв.м. (Q1)
- 75% квартир имеют площадь до 67.8 кв.м. (Q3)
- стандартное отклонение достаточно велико - 34.06, следовательно, велико влияние выбросов

Верхний ус диаграмы размаха отсекает всего около 5% значений:

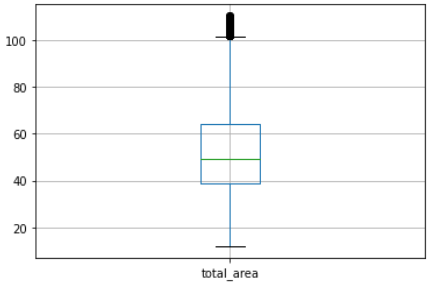
In [47]:
```
round(len(good_realty[good_realty['total_area'] > 110.13]) * 100 / len(good_realty), 2)
```

Out[47]:
```
5.02
```

Удалим их и посмотрим, как изменится распределение:

In [48]:
```
# Удаление выбросов
good_realty = good_realty.query('total_area <= 110.25')

# Характеристика столбца
characterize_box(good_realty, 'total_area')
```

```
count    19417.000000
mean        53.327463
std         18.616255
min         12.000000
25%         39.000000
50%         49.120000
75%         64.000000
max        110.200000
Name: total_area, dtype: float64
```



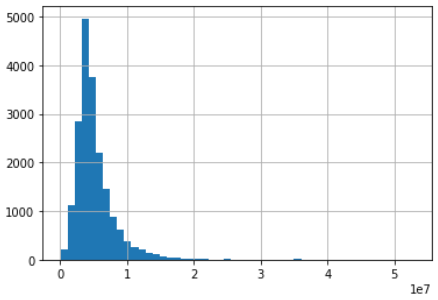
Характеристики распределения существенно улучшились.

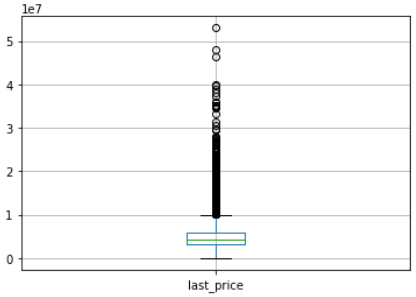
Исследование распределения цены квартиры

Охарактеризуем столбец `'last_price'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [49]:
```
characterize_hist_box(good_realty, 'last_price')
```

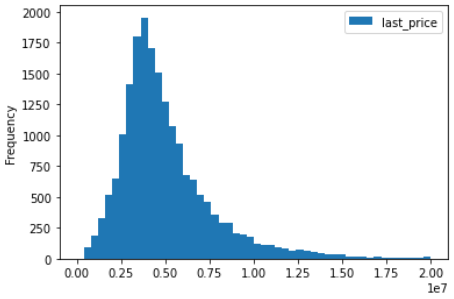
```
count    1.941700e+04
mean     5.121432e+06
std      3.084219e+06
min      1.219000e+04
25%      3.330000e+06
50%      4.400000e+06
75%      6.025000e+06
max      5.300000e+07
Name: last_price, dtype: float64
```

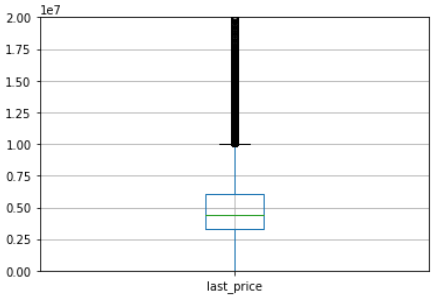




Очевидно, большинство адекватных значений стоимости сосредоточены в диапазоне до 10 млн. Кроме того, гистограмма показывает, что выбросы свыше 20 млн. - единичные (вероятно, их можно выбросить). Учтём это при построении диаграмм:

In [50]:
```
# Построение диаграмм
good_realty.plot(y='last_price', kind='hist', bins=50, range=(0, 20000000))
plt.show()
plt.ylim(0, 20000000)
good_realty.boxplot(column='last_price')
plt.show()
```





Распределение стоимости квартир похоже на Пуассоновское, медиана находится в районе 5 млн., нижний ус диаграммы размаха невозможно различить (минимальное значение, как мы помним, 12190), верхний - отсекает значения свыше 10 млн., считая их выбросами:

In [51]:
```
characteristics = good_realty['last_price'].describe()
characteristics['75%'] + 1.5 *(characteristics['75%'] - characteristics['25%'])
```

Out[51]:
```
10067500.0
```

Статистические характеристики столбца следующие:

- средняя стоимость квартиры составляет около 5.12 млн.
- медиана - 4.4 млн.
- 25% квартир проданы по цене не выше 3.33 млн. (Q1)
- 75% квартир проданы по цене не выше 6 млн. (Q3)
- min цена составляет 12 190 руб.

Минимальная цена за квартиру настораживает. Рассмотрим квартиры с наименьшей стоимостью стоимостью в пределах до 1 млн.:

In [52]:
```
good_realty.query('last_price < 1000000').sort_values(by='last_price').head()
```

Out[52]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	ponds_around_3000	ponds_nearest	days_expos
8793	7	12190.0	109.0	2019-03-20	2	2.75	25	32.0	25	False	...	0.0	NaN	
14911	5	430000.0	54.0	2018-06-26	2	NaN	3	NaN	3	False	...	NaN	NaN	
16274	18	440000.0	40.0	2018-07-10	1	NaN	5	NaN	1	False	...	NaN	NaN	
17676	0	450000.0	36.5	2018-02-01	1	NaN	5	17.3	4	False	...	NaN	NaN	
16219	14	450000.0	38.5	2018-07-11	2	2.65	2	NaN	1	False	...	NaN	NaN	

5 rows × 29 columns

Налицо аномальное несоответствие стоимости и площади самой дешёвой квартиры. Целесообразно удалить этот выброс (подходят все квартиры стоимостью выше 400 000). Верхний ус диаграммы размаха отсекает 5.81% объявлений:

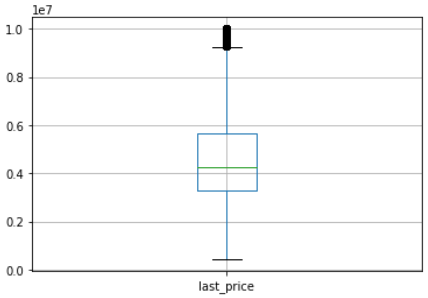
In [53]:
```
round(len(good_realty[good_realty['last_price'] > 10000000]) * 100 / len(good_realty), 2)
```

Out[53]:
```
5.81
```

Очистим таблицу от аномальных значений. Оставим только квартиры стоимостью от 400 тысяч до 10 миллионов и посмотрим, как изменится распределение:

In [54]:
```
# Удаление выбросов
good_realty = good_realty.query('last_price > 400000 and last_price <= 10000000')

# Характеристика столбца
characterize_box(good_realty, 'last_price')
```

```
count    1.828800e+04
mean     4.579979e+06
std      1.905149e+06
min      4.300000e+05
25%      3.288450e+06
50%      4.265000e+06
75%      5.675512e+06
max      1.000000e+07
Name: last_price, dtype: float64
```



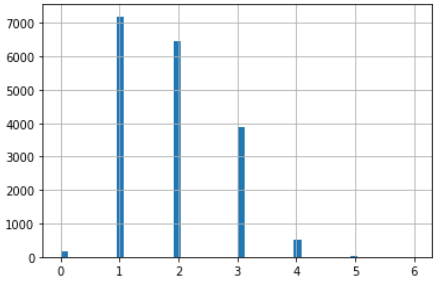
На диаграмме размаха появился нижний ус! Характеристики распределения существенно улучшились.

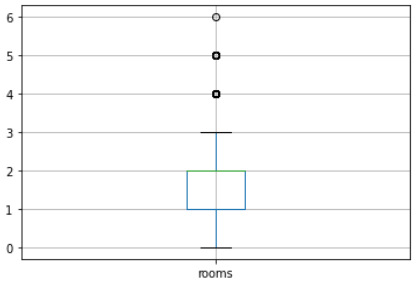
Исследование распределения количества комнат

Охарактеризуем столбец `'rooms'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [55]:
```
characterize_hist_box(df=good_realty, col='rooms')
```

```
count    18288.000000
mean         1.861330
std          0.869274
min          0.000000
25%          1.000000
50%          2.000000
75%          2.000000
max          6.000000
Name: rooms, dtype: float64
```





Распределение количества комнат похоже на Пуассоновское, медиана находится в районе 2, нижний ус диаграммы размаха упирается в 0, верхний - отсекает значения свыше 3, считая их выбросами.

Рассмотрим какие площади занимают квартиры от 4 комнат:

In [56]:
```
good_realty.query('rooms >= 4')['total_area'].describe()
```

Out[56]:
```
count    542.000000
mean      79.632841
std       17.267735
min       40.000000
25%       72.000000
50%       79.550000
75%       93.800000
max      110.200000
Name: total_area, dtype: float64
```

Данные характеристики хорошо соотносятся с живыми данными Яндекс.Недвижимости. В Санкт-Петербурге и окрестностях действительно существуют очень маленькие 4- и более комнатные квартиры.

Рассмотрим, однако, квартиры с 0 комнат. **Гипотеза** заключается в том, что ноль проставляют для квартир свободной планировки. Проверим её:

In [57]:
```
round(
    good_realty.query('rooms == 0')['open_plan'].sum() * 100 / len(good_realty.query('rooms == 0')), 2
)
```

Out[57]:
```
32.58
```

Неожиданно, но среди 0-комнатных квартир только 32.58% помечены, как свободная планировка. Очевидно, это ошибка ввода данных пользователем (забыли проставить "галочку"). 0-комнатных квартир фиксированной планировки не бывает.

Оценим характеристики значений общей полощади квартир, которые забыли пометить, как свободной планировки:

In [58]:
```
good_realty.query('rooms == 0 and not open_plan')['total_area'].describe()
```

Out[58]:
```
count    120.000000
mean      27.062333
std        7.406973
min       15.500000
25%       24.000000
50%       26.000000
75%       28.220000
max       73.600000
Name: total_area, dtype: float64
```

Площадь таких квартир явно не способна вместить 9 или 10 комнат, чтобы можно было говорить об ошибках ввода другой природы.

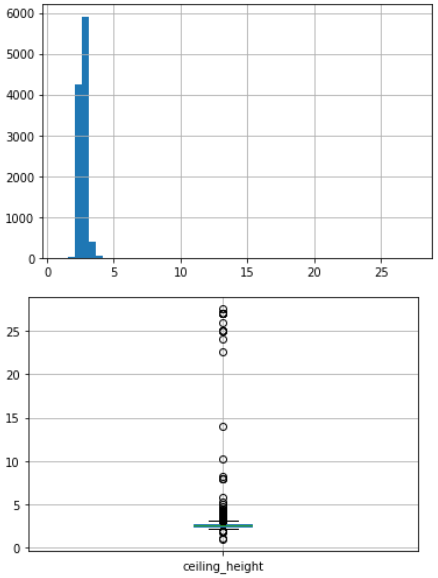
В идеале, надо было бы проставить значение `True` в поле `'open_plan'` для данных 120 квартир, однако эта информация далее использоваться не будет, поэтому оставим столбцы `'rooms'` и `'open_plan'` без изменений.

Исследование распределения высоты потолков

Охарактеризуем столбец `'ceiling_height'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

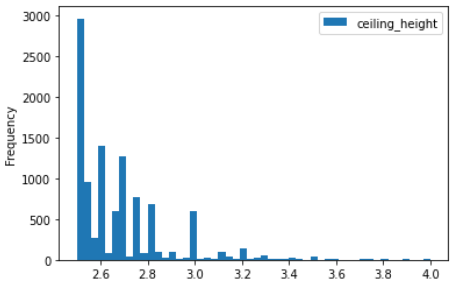
In [59]:
```
characterize_hist_box(df=good_realty, col='ceiling_height')
```

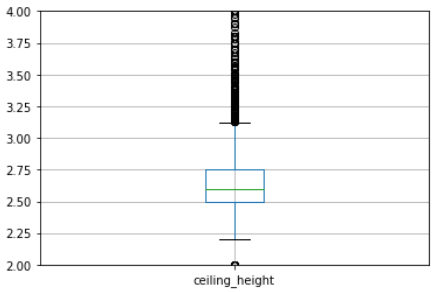
```
count    10641.000000
mean         2.714174
std          0.942705
min          1.000000
25%          2.500000
50%          2.600000
75%          2.750000
max         27.500000
Name: ceiling_height, dtype: float64
```



Очевидно, большинство адекватных значений высоты сосредоточены в диапазоне до 3 метров. Кроме того, гистограмма показывает как выбросы свыше 4 метров - единичные (вероятно, их можно выбросить), так и менее 2.5 метров (вероятно, их также можно выбросить). Учтём это при построении диаграмм:

In [60]:
```
# Построение диаграмм
good_realty.plot(y='ceiling_height', kind='hist', bins=50, range=(2.5, 4))
plt.show()
plt.ylim(2, 4)
good_realty.boxplot(column='ceiling_height')
plt.show()
```





Отметим, что высота потолков по аналогии с количеством комнат - дискретная величина. Этот факт нашел отражение на гистограмме, которая выглядит похожей на пуассоновское распределение дискретной случайной величины.

Медиана находится в районе 2.6 м., нижний ус диаграммы размаха отсекает значения менее 2.13 м., считая их выбросами:

In [61]:
```
characteristics = good_realty['ceiling_height'].describe()
characteristics['25%'] - 1.5 * (characteristics['75%'] - characteristics['25%'])
```

Out[61]:
```
2.125
```

Верхний ус отсекает значения свыше 3.13 м., считая их выбросами:

In [62]:
```
characteristics['75%'] + 1.5 * (characteristics['75%'] - characteristics['25%'])
```

Out[62]:
```
3.125
```

Статистические характеристики столбца следующие:

- средняя высота потолков составляет около 2.71 м.
- медиана - 2.6 м.
- 25% квартир имеют высоту потолков до 2.5 м. (Q1)
- 75% квартир имеют высоту потолков до 2.75 м. (Q3)
- стандартное отклонение достаточно мало - 0.94, следовательно, влияние выбросов невелико
- максимальное значение составляет 27.5 м.
- минимальное - 1 м.

Для дальнейших рассуждений аномально большие и малые значения высоты лучше убрать. Усы диаграмы размаха отсекают всего 2.65% значений:

In [63]:
```
round(len(good_realty.query('ceiling_height < 2.125 or ceiling_height > 3.125')) * 100 /
      len(good_realty), 2)
```

Out[63]:
```
2.65
```

Удалить их, сохранив имеющиеся пропуски в данном столбце, с использованием срезов нельзя, поскольку они удалят и пропуски, которых в данном столбце гораздо больше, чем аномальных значений:

In [64]:
```
round(good_realty['ceiling_height'].isna().sum() * 100 / len(good_realty), 2)
```

Out[64]:
```
41.81
```

Терять почти 42% записей в таблице не целесообразно. Тем более, что в целом характеристики распределения высоты потолков удовлетворительны, а выбросы - очень редки.

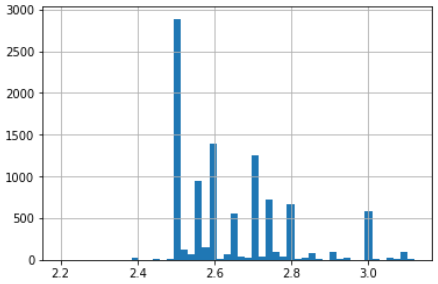
Избавимся от выбросов, заменив их на заведомо большое значение (например, 400) с использованием метода where(), а затем удалив соответствующие строки методом drop():

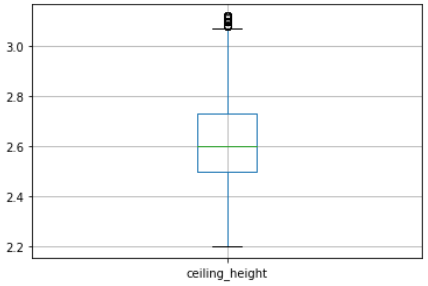
In [65]:
```
# Замена выбросов в 'ceiling_height' на 400
good_realty['ceiling_height'].where(
    (
        (good_realty['ceiling_height'].isna()) |
        ((good_realty['ceiling_height'] >= 2.125) & (good_realty['ceiling_height'] <= 3.125))
    ),
    400,
    inplace=True
)

# Удаление строк, в которых 'ceiling_height' == 400
good_realty.drop(
    labels=good_realty.query('ceiling_height == 400').index,
    axis=0,
    inplace=True
)
```

Рассмотрим обновлённое распределение:

In [66]:
```
characterize_hist_box(df=good_realty, col='ceiling_height')
```

```
count    10156.000000
mean         2.644355
std          0.149589
min          2.200000
25%          2.500000
50%          2.600000
75%          2.730000
max          3.120000
Name: ceiling_height, dtype: float64
```





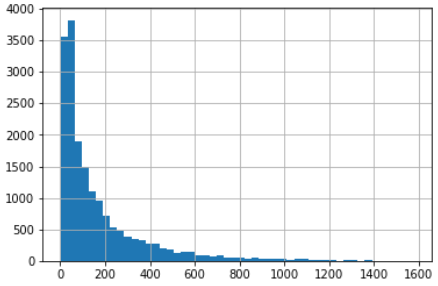
Характеристики распределения существенно улучшились.

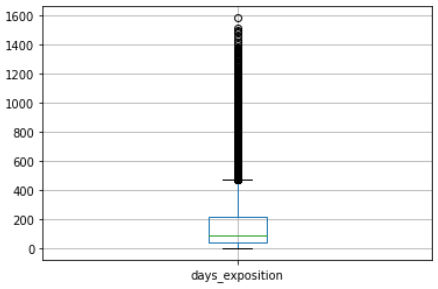
Анализ времени продажи квартиры

Построим гистограмму на 50 корзин, а также охарактеризуем значения столбца `'days_exposition'`.

In [67]:
```
characterize_hist_box(df=good_realty, col='days_exposition')
```

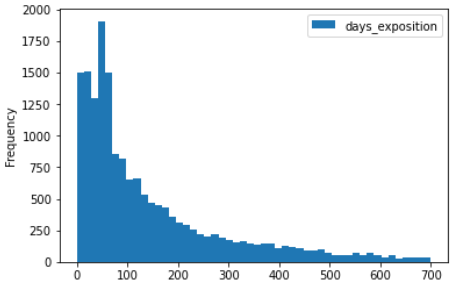
```
count    17803.000000
mean       168.657361
std        206.099460
min          1.000000
25%         43.000000
50%         89.000000
75%        214.000000
max       1580.000000
Name: days_exposition, dtype: float64
```

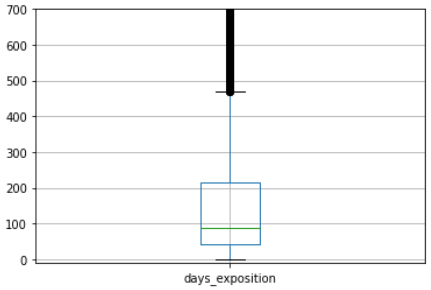




Медиана времени продажи находится в районе 90 дней, значения свыше 700 дней, согласно гистограмме, являются единичными выбросами (их можно удалить), а достоверные значения, предварительно, находятся в диапазоне до 214 дней. Учтём это при построении диаграмм:

In [68]:
```
# Построение диаграмм
good_realty.plot(y='days_exposition', kind='hist', bins=50, range=(0, 700))
plt.show()
plt.ylim(-10, 700)
good_realty.boxplot(column='days_exposition')
plt.show()
```





Распределение времени продажи квартир похоже на искажённое Пуассоновское, нижний ус диаграммы размаха упирается в минимальное значение - 1 день, верхний - отсекает значения свыше 470 дней, считая их выбросами:

In [69]:
```
characteristics = good_realty['days_exposition'].describe()
characteristics['75%'] + 1.5 * (characteristics['75%'] - characteristics['25%'])
```

Out[69]:
```
470.5
```

Статистические характеристики столбца следующие:

- среднее время продажи составляет около 170 дней
- медиана - 90 дней
- 25% квартир были проданы за время до 43 дней (Q1), что можно считать быстрыми продажами
- 75% квартир были проданы за время до 214 дней (Q3), что соответствует чуть более чем 7 месяцам, и всё, что продавалось дольше можно считать долгими продажами
- минимальное время продажи - 1 день
- максимальное - 1580 дней (более 4.3 года)
- стандартное отклонение достаточно велико - 206, следовательно - велико влияние выбросов

Верхний ус диаграмы размаха отсекает всего 8.37% значений:

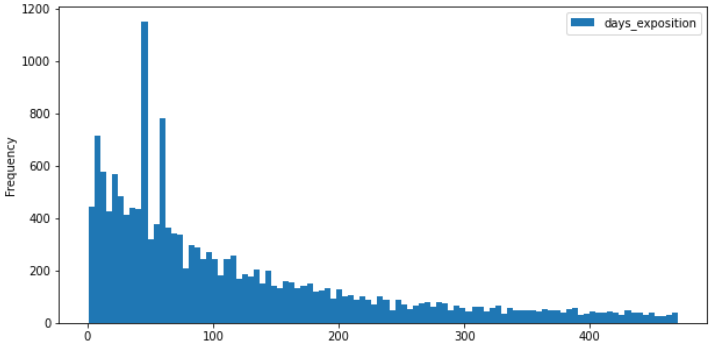
In [70]:
```
round(len(good_realty[good_realty['days_exposition'] > 470]) * 100 / len(good_realty), 2)
```

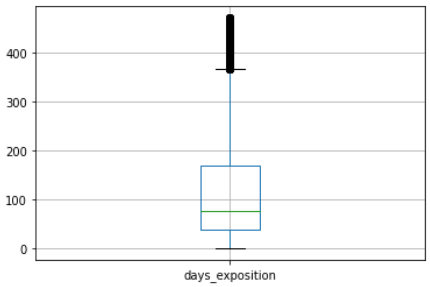
Out[70]:
```
8.37
```

Удалим их и посмотрим, как изменится распределение:

In [71]:
```
# Удаление выбросов
good_realty = good_realty.query('days_exposition <= 470')

# Характеристика столбца
print(good_realty['days_exposition'].describe())

# Построение диаграмм
good_realty.plot(y='days_exposition', kind='hist', bins=100, figsize=(10, 5))
plt.show()
good_realty.boxplot(column='days_exposition')
plt.show()
```

```
count    16313.000000
mean       118.766812
std        111.477790
min          1.000000
25%         39.000000
50%         76.000000
75%        170.000000
max        470.000000
Name: days_exposition, dtype: float64
```





Статистические характеристики столбца изменились:

- среднее время продажи снизилось до 119 дней
- медиана - до 76 дней
- 25% квартир были проданы за время до 39 дней (Q1), что теперь можно считать быстрыми продажами
- 75% квартир были проданы за время до 170 дней (Q3), что соответствует снижению до 5.7 месяцев, и всё, что продавалось дольше можно теперь считать долгими продажами
- стандартное отклонение также снизилось - с 206.09 до 112.21, следовательно - влияние выбросов снизилось почти в 2 раза

В целом, характеристики распределения улучшились.

Пуассоновский характер распределения портят выбросы в районах 10-15, 25-35, 45-50, 65-70 дней. В среднем, каждые две недели до медианного времени продажи резко возрастает количество закрываемых объявлений.

Это не может быть связано со сроками публикации объявлений по Санкт-Петербургу и Ленинградской области.

Вероятную причину такого поведения пользователей по имеющимся данным предположить сложно.

Исследование факторов, влияющих на стоимость квартиры

Зависимость стоимости квартиры от общей площади, количества комнат, удалённости от центра

Рассмотрим зависимость стоимости квартиры `'last_price'` от общей площади `'total_area'`, количества комнат `'rooms'`, удалённости от центра `'city_centers_nearest'`.

Построим матрицу попарной корреляции Пирсона для этих параметров:

In [72]:
```
columns = ['last_price', 'total_area', 'rooms', 'city_centers_nearest']
good_realty[columns].corr().style.background_gradient('coolwarm')
```

Out[72]:

	last_price	total_area	rooms	city_centers_nearest
last_price	1.000000	0.663150	0.415794	-0.310143
total_area	0.663150	1.000000	0.806067	-0.067345
rooms	0.415794	0.806067	1.000000	-0.037550
city_centers_nearest	-0.310143	-0.067345	-0.037550	1.000000

Заметно, что цена больше всего связана с общей площадью квартиры. В меньшей степени она зависит от количества комнат (возможно из-за того, что мы оставили в данных 0-комнатные квартиры).

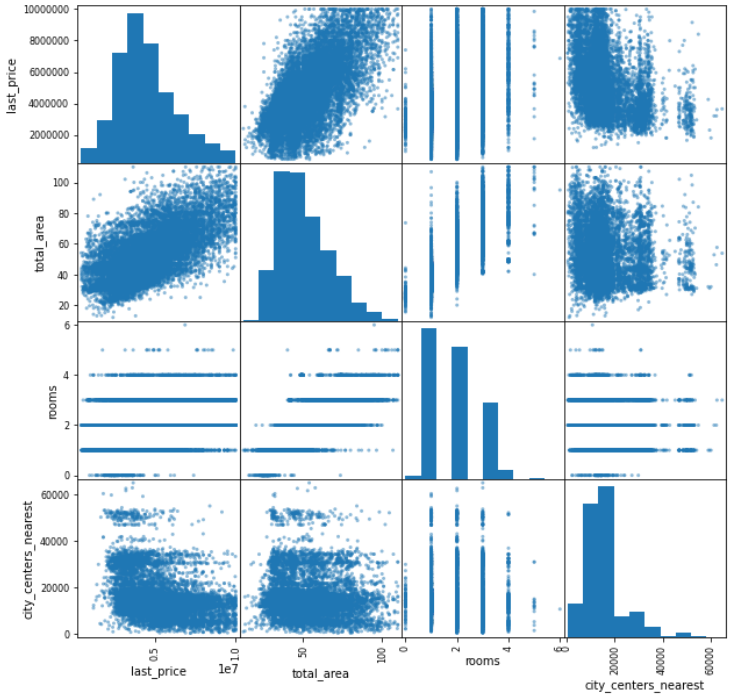
Слабее всего цена зависит от расстояния до центра. При этом логично наблюдается обратная зависимость: чем меньше расстояние до центра, тем выше цена.

Количество комнат совершенно справедливо существенно зависит от площади.

А вот от расстояния до центра ни количество комнат, ни площадь квартир существенно не зависят. При этом коэффициенты корреляции намекают, что с увеличением расстояния до центра площадь и количество комнат слегка снижаются.

Построим попарные диаграммы рассеивания для данных столбцов для иллюстрации указанных закономерностей:

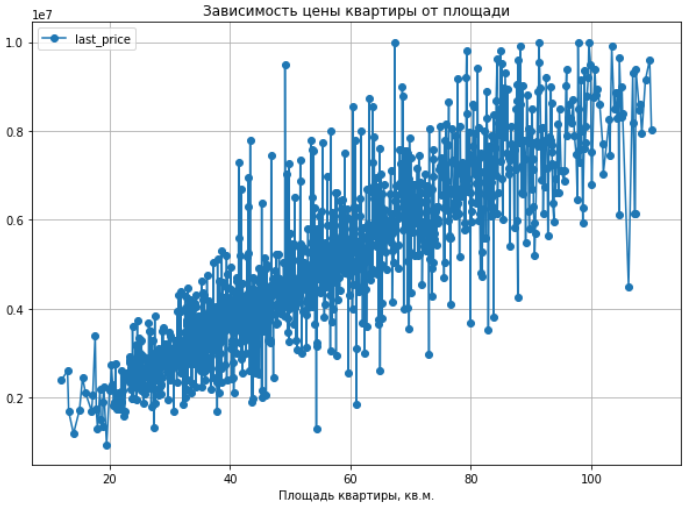
In [73]:
```
pd.plotting.scatter_matrix(good_realty[columns],
                           grid=True, figsize=(10, 10))
plt.show()
```

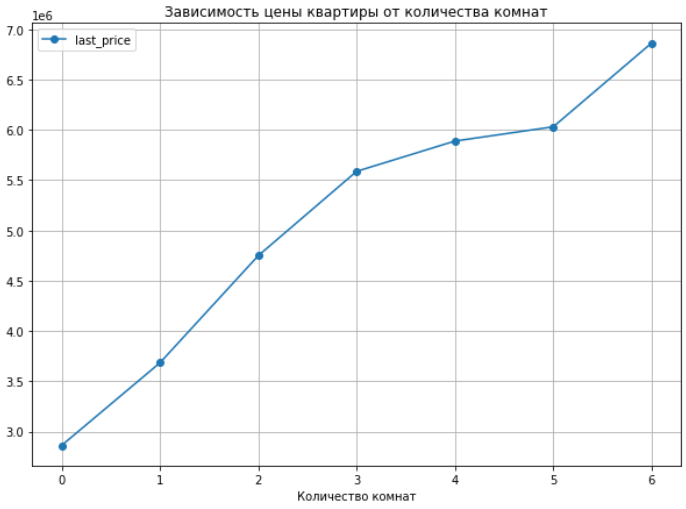


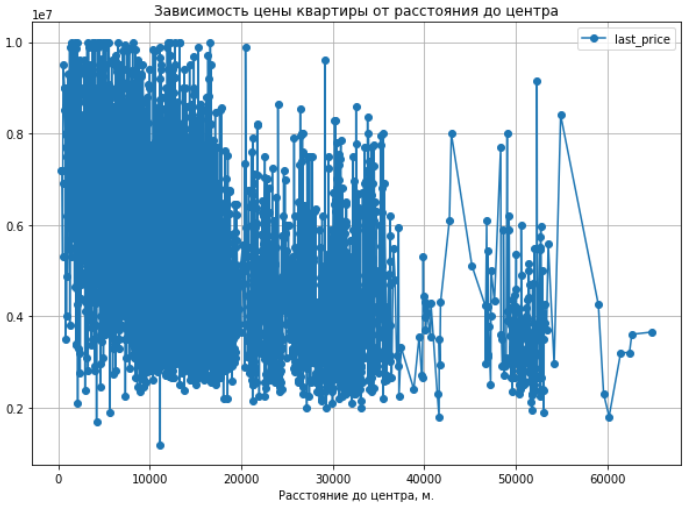
Зависимость цены от площади прослеживается чётко. Также заметно, что с увеличением количества комнат (не рассматривая 0) растёт и цена. Зависимость цены от удалённости (правый верхний угол) - слабая и обратная.

Построим графики зависимости цены от указанных параметров:

In [74]:
```
# Зададим функцию, строящую график зависимости двух величин
# На вход подаются:
#    df - датафрейм
#    x - значения по оси X
#    y - значения по оси Y
#    titl - заголовок графика
#    xlbl - подпись оси X
#    aggf - функция усреднения (по умолчанию - среднее)
#    figsz - размер графика (по умолчанию - (10, 7))
def build_plot(df, x, y, titl, xlbl, aggf='mean', figsz=(10, 7)):
    df.pivot_table(
        index=x,
        values=y,
        aggfunc=aggf
    ).plot(
        title=titl,
        figsize=figsz,
        grid=True,
        style='o-',
        xlabel=xlbl
    )
    plt.show()

# # Построим графики зависимости стоимости от параметров
build_plot(good_realty, 'total_area', 'last_price',
           'Зависимость цены квартиры от площади', 'Площадь квартиры, кв.м.')
build_plot(good_realty, 'rooms', 'last_price',
           'Зависимость цены квартиры от количества комнат', 'Количество комнат')
build_plot(good_realty, 'city_centers_nearest', 'last_price',
           'Зависимость цены квартиры от расстояния до центра', 'Расстояние до центра, м.')
```







Очевидно, график лучше гистограммы отражает зависимость усреднённой цены от дискретной величины. Мы видим, что, начиная с 3 комнат рост стоимости существенно замедляется.

Зависимость стоимости квартиры от этажа

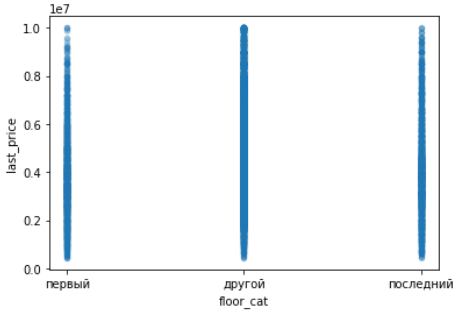
Рассмотрим теперь зависимость стоимости квартиры `'last_price'` от этажа `'floor_cat'`. Вычислить коэффициент корреляции Пирсона для этих параметров невозможно, т.к. один из них строковый. Однако, корреляцию можно оценить со значением этажа `'floor'`:

In [75]:
```
good_realty['last_price'].corr(good_realty['floor'])
```

Out[75]:
```
0.1642333042115989
```

Отметим, что корреляция цены с этажом самая слабая, по сравнению с рассмотренными выше параметрами. Выведем для данных параметров диаграмму рассеяния, снизив прозрачность:

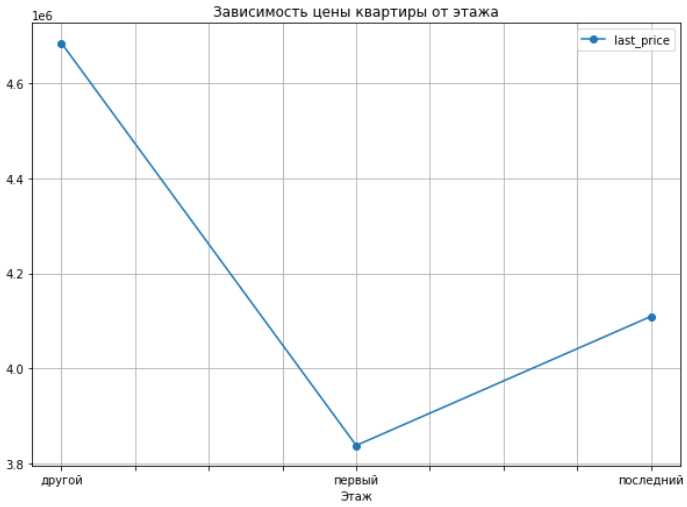
In [76]:
```
good_realty.plot(x='floor_cat', y='last_price', kind='scatter', alpha=0.2)
plt.show()
```



На диаграмме отчётливо видно, что дорогие квартиры на первом и последнем этаже ценятся меньше. В нижнем и среднем ценовом диапазоне спрос на первый и последний этажи почти не отличается от спроса на любой другой этаж.

Построим график зависимости средней цены квартиры от этажа:

In [77]:
```
build_plot(good_realty, 'floor_cat', 'last_price',
           'Зависимость цены квартиры от этажа', 'Этаж')
```



Очевидно, меньше всего ценится первый этаж по причине близости к подвалу и повышенных рисков.

Последние этажи - чуть дороже, но не так ценятся как средние этажи дома. Это объясняется близостью к крыше, техническому этажу и коммуникациям, что опять-таки повышает риски владения.

Зависимость стоимости квартиры от от дня недели, месяца и года размещения объявления

Рассмотрим зависимость стоимости квартиры `'last_price'` от дня недели `'first_exposition_weekday'`, месяца `'first_exposition_month'` и года размещения объявления `'first_exposition_year'`.

Построим матрицу попарной корреляции Пирсона для этих параметров, поскольку все они числовые:

In [78]:
```
columns = ['last_price', 'first_exposition_weekday',
           'first_exposition_month', 'first_exposition_year']
good_realty[columns].corr().style.background_gradient('coolwarm')
```

Out[78]:

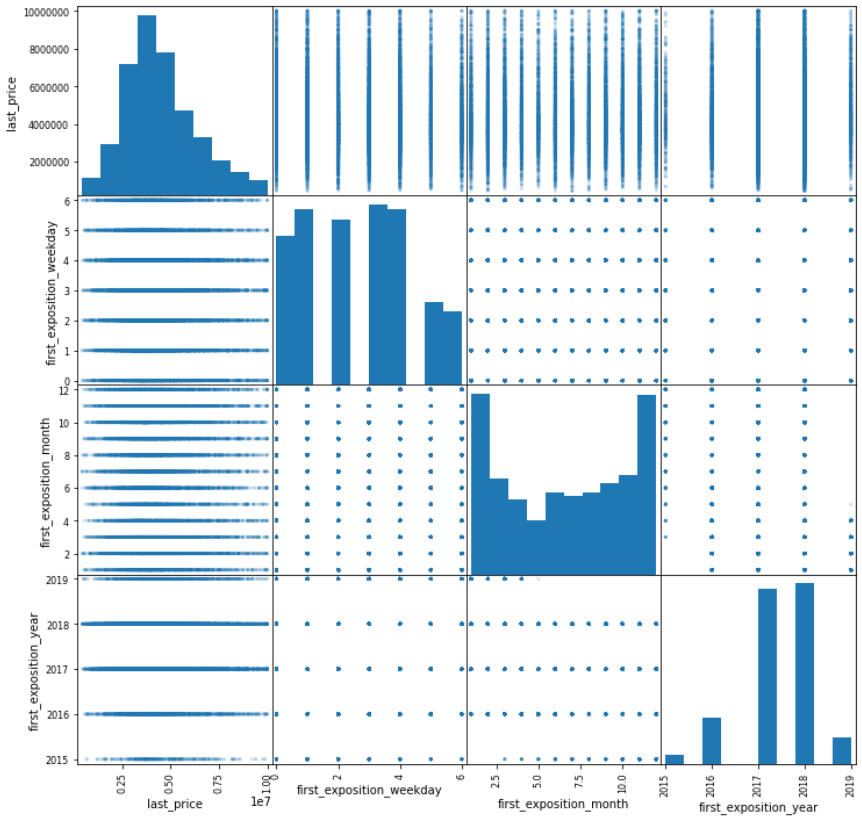
	last_price	first_exposition_weekday	first_exposition_month	first_exposition_year
last_price	1.000000	-0.009458	0.003674	0.012671
first_exposition_weekday	-0.009458	1.000000	0.011235	-0.002818
first_exposition_month	0.003674	0.011235	1.000000	-0.231484
first_exposition_year	0.012671	-0.002818	-0.231484	1.000000

Очевидно, ежедневная и ежемесячная корреляция - очень слабые. При этом в рабочие дни квартиры продаются дороже, чем в конце.

Наиболее сильно цена квартир связана с годом. Что может объясняться годовыми изменениями экономической ситуации и движениями рынка.

Построим попарные диаграммы рассеивания для данных столбцов для иллюстрации указанных закономерностей:

In [79]:
```
pd.plotting.scatter_matrix(good_realty[['last_price', 'first_exposition_weekday',
                                        'first_exposition_month', 'first_exposition_year']],
                           grid=True, figsize=(12, 12), alpha=0.2)
plt.show()
```

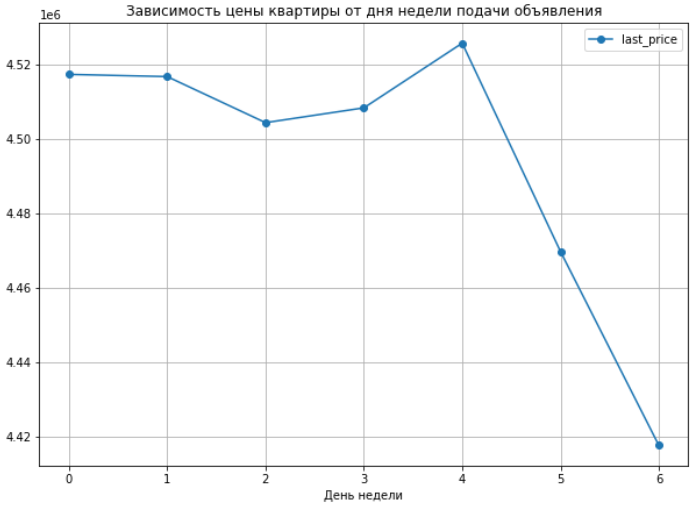


На основании диаграмм можно сделать следующие наблюдения и выводы:

- в субботу и воскресенье одновременно уменьшается как количество размещаемых объявлений, так и средняя цена проданных квартир
- пики размещений приходятся на январь и декабрь, при этом можно заметить некоторое снижение средней цены, что объясняется пред- и постновогодними скидками
- объёмы размещений постепенно падают с февраля по апрель, в мае возобновляют рост и держатся приблизительно на одном уровне до декабря
- отчётливо видно, что в 2015, 2016 гг. наблюдалось проседание рынка, что повлияло на среднюю стоимость проданных квартир (данных за 2019 год недостаточно, чтобы следать аналогичный вывод)
- на диаграмме зависимости месяца от года можно оценить охват анализируемой выборки объяалений - с марта 2015 по май 2019 годов

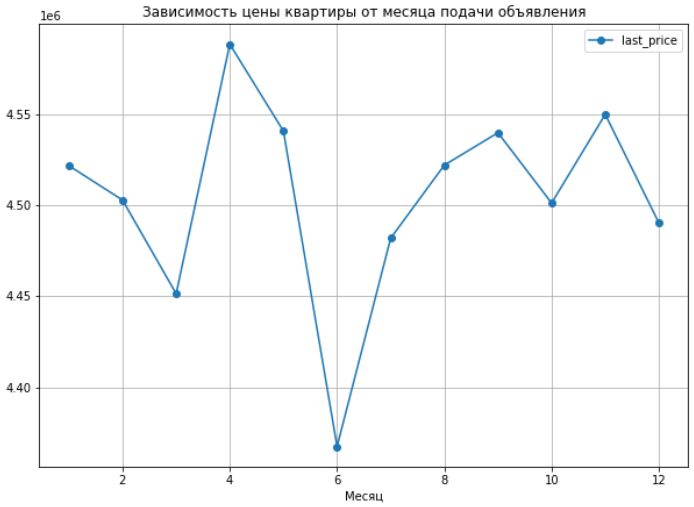
На диаграммах рассеяния плохо исследовать зависимость цены от дискретных величин. Зато дискретные значения дней недели, месяцев и лет, а также зависимость от них средней цены квартир удобно и наглядно изобразить на графиках:

In [80]:
```
build_plot(good_realty, 'first_exposition_weekday', 'last_price',
           'Зависимость цены квартиры от дня недели подачи объявления', 'День недели')
```



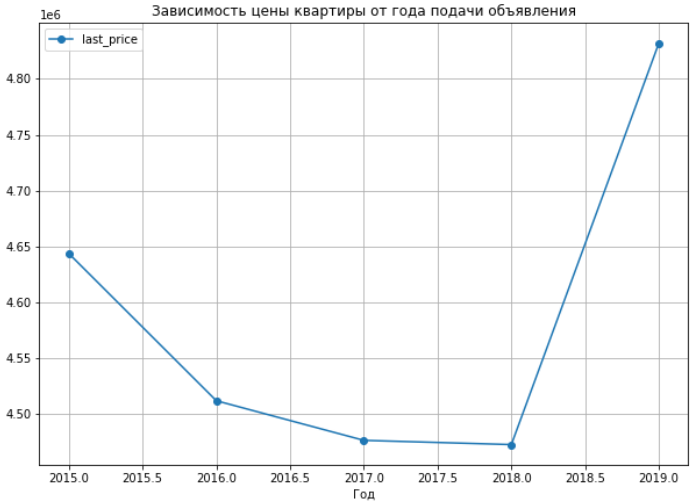
Стоимость квартир слабо колеблется в течение недели, а в выходные продавцы охотнее идут на уступки.

In [81]:
```
build_plot(good_realty, 'first_exposition_month', 'last_price',
           'Зависимость цены квартиры от месяца подачи объявления', 'Месяц')
```



Резкий спад цены в июне обусловлен снижением спроса на пике сезона отпусков. Снижение цен в январе и марте можно объяснить предпраздничными распродажами и скидками от застройщиков. Подъём цен в апреле - весенним ростом спроса на недвижимость, обусловленным психологическими факторами: люди хотят обновления, а также торопятся купить квартиру, чтобы успеть сделать ремонт в тёплое время года.

In [82]:
```
build_plot(good_realty, 'first_exposition_year', 'last_price',
           'Зависимость цены квартиры от года подачи объявления', 'Год')
```



На графике наглядно представлено, как увеличение предложения в 2017 и 2018 годах привело к снижению средней стоимости квартиры. Рынок в действии!

К сожалению, данные представлены не за целый 2019 год. Поэтому сложно объяснить столь резкий рост средней цены за квартиру в этом году.

Анализ 10 населённых пунктов с наибольшим количеством объявлений

Отберём 10 населённых пунктов с наибольшим количеством объявлений и отсортируем их по убыванию средней цены квадратного метра:

In [83]:
```
# Составим сводную таблицу по городам с:
#   - количеством объявлений
#   - средней ценой квадратного метра
#   - медианной ценой квадратного метра
top_10_city_exposition = good_realty.pivot_table(
    index='locality_name',
    values='price_per_m2',
    aggfunc=['count', 'mean', 'median']
)

# Переименуем колонки
top_10_city_exposition.columns = ['count', 'price_m2_mean', 'price_m2_median']

# выделим 10 городов с наибольшим количеством объявлений
top_10_city_exposition = top_10_city_exposition.sort_values(by='count', ascending=False).head(10)

# Отсортируем по среднему
top_10_city_exposition.sort_values(by='price_m2_mean', ascending=False)
```

Out[83]:

locality_name	count	price_m2_mean	price_m2_median
Санкт-Петербург	10215	103406.838193	100214.750
Пушкин	251	99141.844622	97451.270
деревня Кудрово	262	92584.513626	91901.640
посёлок Парголово	282	89163.305780	91138.910
посёлок Мурино	475	85969.843326	86268.870
посёлок Шушары	377	77787.199682	75917.740
Колпино	279	74668.846810	73728.810
Гатчина	238	68270.643067	67180.705
Всеволожск	297	66528.626835	65540.540
Выборг	179	57059.266816	56756.760

Среди 10 городов с наибольшим количеством объявлений дороже всего квадратные метры в Санкт-Петербурге, Пушкине, Кудрово. Дешевле всего - в Выборге, Всеволжске и Гатчине.

Изучение предложения квартир в Санкт-Петербурге

Выделим предложение квартир в Санкт-Петербурге:

In [84]:
```
piter_realty = good_realty.query('locality_name == "Санкт-Петербург"').copy()
piter_realty.head(2)
```

Out[84]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	ponds_around_3000	ponds_nearest	days_expositio
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.0	13	False	...	1.0	48.0	12
9	18	5400000.0	61.0	2017-02-26	3	2.50	9	43.6	7	False	...	0.0	NaN	28

2 rows × 29 columns

Создадим столбец с расстоянием до центра в колометрах и рассмотрим его характеристики:

In [85]:
```
# Переведём расстояния в километры
piter_realty['city_centers_nearest_km'] = piter_realty['city_centers_nearest']\
                                          .apply(lambda x: x / 1000)
# Проверим результат
piter_realty['city_centers_nearest_km'].describe()
```

Out[85]:
```
count    10215.000000
mean        12.738503
std          4.209802
min          0.329000
25%         10.667000
50%         13.053000
75%         15.365000
max         29.493000
Name: city_centers_nearest_km, dtype: float64
```

Среднее довольно близко к медиане. Наблюдается довольно большое отклонение - возможно люди по-разному измеряют расстояние: одни по дорогам, другие - напрямую.

Удивление вызывают 29 километров от центра - неужели Питер такой большой?

Приведём столбец километров к целому типу:

In [86]:
```
piter_realty['city_centers_nearest_km'] = piter_realty['city_centers_nearest_km'].astype('int')
```

Вычислим среднюю цену для каждого километра и построим график зависимости:

In [87]:
```
build_plot(piter_realty, 'city_centers_nearest_km', 'price_per_m2',
           'Зависимость цены кв.м. от удалённости от центра', 'Расстояние до центра в км.')
```

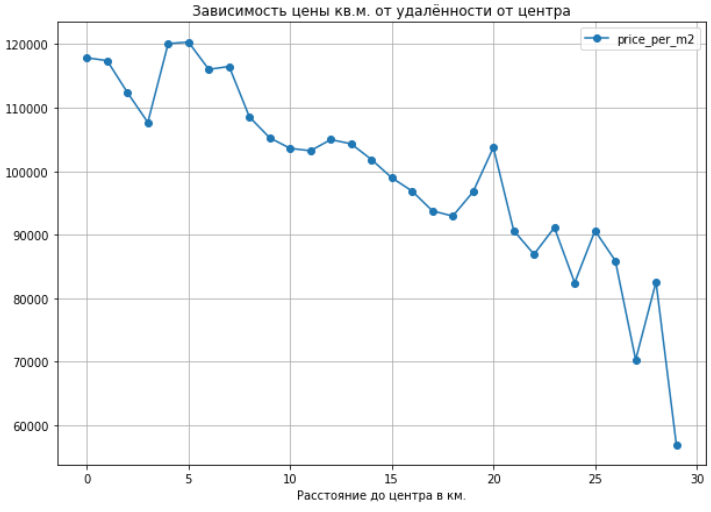


График показывает, что, начиная с 7 километра, стоимость квадратного метра снижается. Значит **зону центра можно ограничить 7 километрами**.

Удивление вызывают артефакты в центральной зоне (на расстоянии от 0 до 4 км.), а также на периферии (на расстоянии свыше 18 км.).

Артефакты на большом расстоянии могут говорить о, мягко говоря, странной ценовой политике застройщиков новых высотных жилых кварталов.

Причину артефактов в центре предположить сложно. Скорее всего в ходе исследований мы удалили слишком мого записей, посчитав их выбросами по отдельным параметрам.

Для сравнения построим аналогичный график по неочищенным данным, только в качестве функции аггрегации применим медиану, как более устойчивую к выбросам:

In [88]:
```
piter_realty_original = realty.query('locality_name == "Санкт-Петербург"').copy()

# Переведём расстояния в километры
piter_realty_original['city_centers_nearest_km'] = piter_realty_original['city_centers_nearest']\
                                                   .apply(lambda x: x / 1000)
piter_realty_original['city_centers_nearest_km'] = piter_realty_original['city_centers_nearest_km']\
                                                   .astype('int')
# Построим график
build_plot(piter_realty_original, 'city_centers_nearest_km', 'price_per_m2',
           'Зависимость цены кв.м. от удалённости от центра, оригинальные данные',
           'Расстояние до центра в км.', aggf='median')
```

Очевидна аналогичная тенденция. Зона центра расширилась до 8 км., однако поведение графика в ней указывает что в среднем на протяжении 2015-2019 гг. в 4-километровой зоне квартиры продавались по заниженным ценам, а на периферии - наоборот.

Анализ предложений в центре Санкт-Петербурга

Выделим центральный сегмент предлоений квартир в Питере на основе очищенных данных:

In [89]:
```
piter_realty_center = piter_realty.query('city_centers_nearest_km <= 7').copy()
piter_realty_center.sample(5)
```

Out[89]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	ponds_nearest	days_exposition	price_per_m2
23602	0	8700000.0	73.0	2019-01-23	3	2.65	4	NaN	4	False	...	151.0	34	119178.08
5292	11	5450000.0	56.6	2018-10-24	3	2.50	5	40.0	5	False	...	451.0	92	96289.75
3816	4	4779000.0	53.1	2016-05-16	1	NaN	13	19.8	6	True	...	826.0	60	90000.00
15426	19	5000000.0	60.2	2017-08-17	2	NaN	7	27.8	1	False	...	NaN	35	83056.48
11705	5	3380000.0	26.0	2018-02-20	0	2.80	17	17.0	8	False	...	802.0	45	130000.00

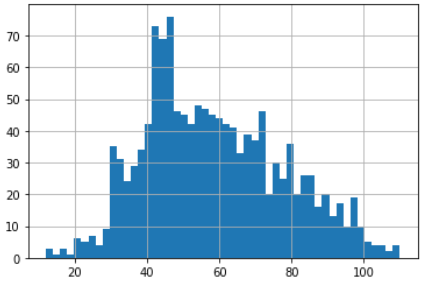
5 rows × 30 columns

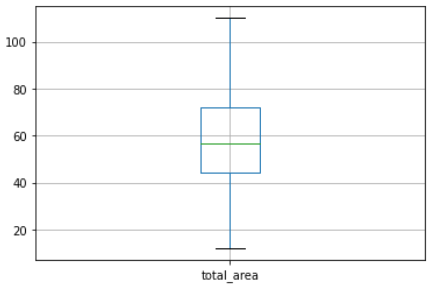
Анализ площади, цены, количества комнат и высоты потолков квартир в центре Санкт-Петербурга

Охарактеризуем столбец `'total_area'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [90]:
```
characterize_hist_box(df=piter_realty_center, col='total_area')
```

```
count    1314.000000
mean       58.571971
std        19.076970
min        12.000000
25%        44.000000
50%        56.400000
75%        72.000000
max       110.000000
Name: total_area, dtype: float64
```





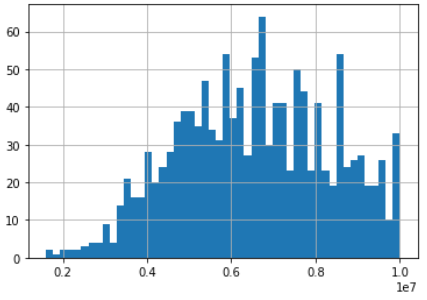
Сравнивая характеристики площади квартир в центре с данными по всей базе (см. п. 4.1.1) можно сделать следующие **выводы**:

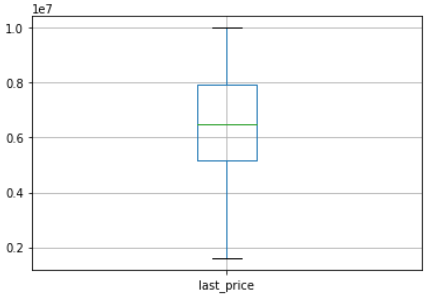
- самые большие и самые маленькие квартиры в базе расположены в центре Питера
- средние показатели площади в центре выше соответствующих показателей по всей базе примерно на 15%
- в целом, судя по количеству объявлений в центре, распределение как площадей, так и других параметров в базе определяется распределением в центре Санкт-Петербурга

Охарактеризуем столбец `'last_price'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [91]:
```
characterize_hist_box(df=piter_realty_center, col='last_price')
```

```
count    1.314000e+03
mean     6.545203e+06
std      1.793391e+06
min      1.600000e+06
25%      5.163750e+06
50%      6.500000e+06
75%      7.950000e+06
max      1.000000e+07
Name: last_price, dtype: float64
```





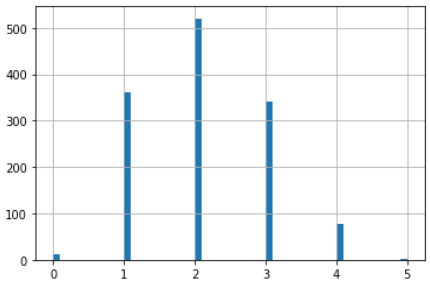
Сравнивая характеристики цены квартир в центре с данными по всей базе (см. п. 4.1.2) можно сделать следующие **выводы**:

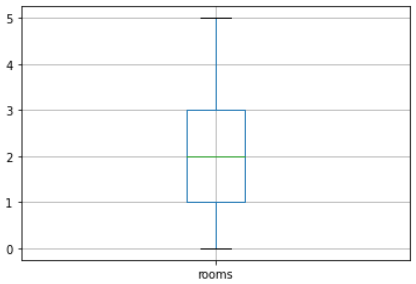
- в центре нет дешёвых квартир, однако есть самые дорогие
- средние показатели стоимости квартир в центре выше соответствующих показателей по всей базе примерно в 1.5 раза
- в целом, можно утверждать, что в центре сосредоточены самые дорогие квартиры в базе

Охарактеризуем столбец `'rooms'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [92]:
```
characterize_hist_box(df=piter_realty_center, col='rooms')
```

```
count    1314.000000
mean        2.088280
std         0.906001
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max         5.000000
Name: rooms, dtype: float64
```





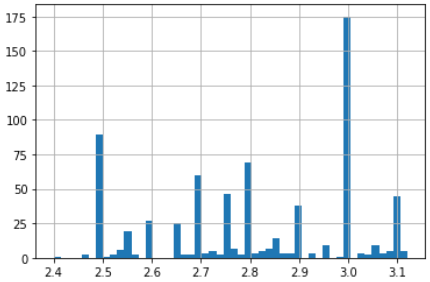
Сравнивая характеристики количества комнат в квартирах в центре с данными по всей базе (см. п. 4.1.3) можно сделать следующие **выводы**:

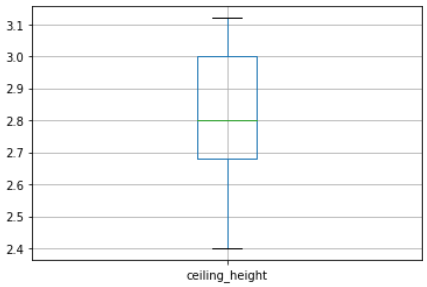
- в центре почти нет квартир свободной планировки
- количество 1-комнатных значительно меньше по сравнению с 2-комнатными (в целом по базе - наоборот, лидируют однушки)
- это сказалось на медиане - она вплотную приблизилась к среднему количеству комнат

Охарактеризуем столбец `'ceiling_height'`, а также построим для него гистограмму на 50 корзин и диаграмму размаха:

In [93]:
```
characterize_hist_box(df=piter_realty_center, col='ceiling_height')
```

```
count    705.000000
mean       2.812057
std        0.194071
min        2.400000
25%        2.680000
50%        2.800000
75%        3.000000
max        3.120000
Name: ceiling_height, dtype: float64
```





Сравнивая характеристики высоты потолков в квартирах в центре с данными по всей базе (см. п. 4.1.3) можно сделать следующие **выводы**:

- квартиры в центре обладают преимущественно высокими потолками - подавляющее большинство потолков выше 2.7 м. (по всей базе - не выше 2.6 м.)
- лидерами по количеству являются 3-метровые потолки
- средняя высота потолков выросла с 2.6 до 2.8 м. по сравнению с данными по всей базе

Анализ влияния на стоимость квартиры в центре Санкт-Петербурга количества комнат, этажа, удалённости от центра, даты размещения объявления

Рассмотрим для центра Питера зависимость стоимости квартиры `'last_price'` от количества комнат `'rooms'` и удалённости от центра `'city_centers_nearest'`.

Построим матрицу попарной корреляции Пирсона для этих параметров:

In [94]:
```
columns = ['last_price', 'rooms', 'city_centers_nearest']
piter_realty_center[columns].corr().style.background_gradient('coolwarm')
```

Out[94]:

	last_price	rooms	city_centers_nearest
last_price	1.000000	0.445522	-0.155465
rooms	0.445522	1.000000	-0.170581
city_centers_nearest	-0.155465	-0.170581	1.000000

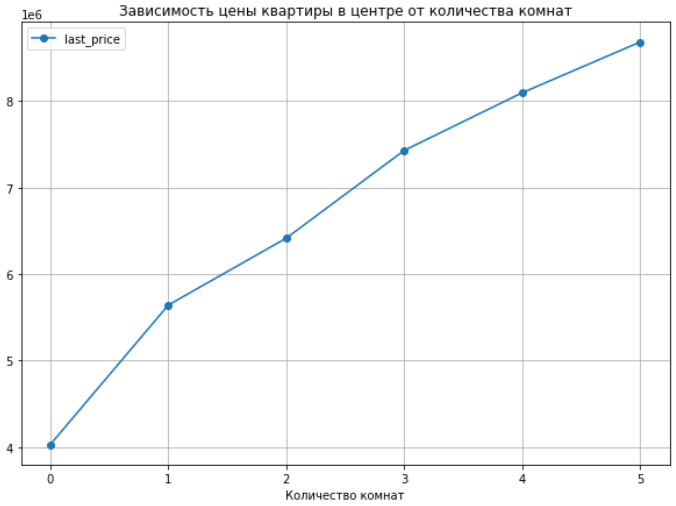
Сравнивая полученные корреляции с рассчитанными ранее для всей базы объявлений (см. п. 4.3.1), можно сделать следующие **выводы**:

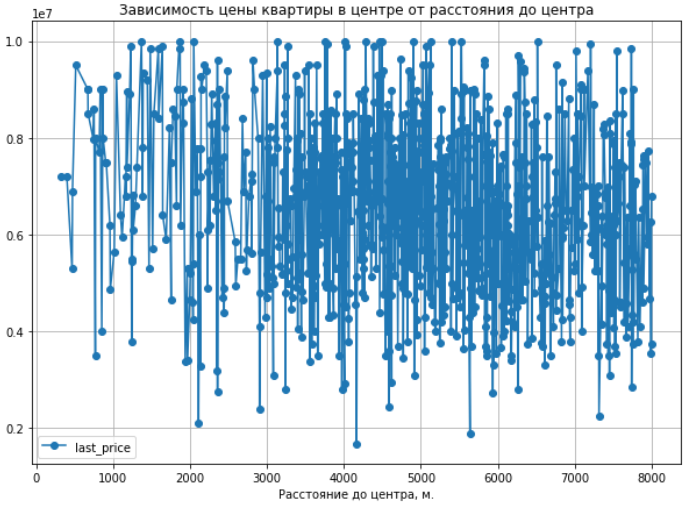
1. В центре Питера цена квартиры несколько сильнее зависит от количества комнат.
2. Вместе с тем, зависимость от близости к центру закономерно ослабла.
3. Количество комнат в квартирах в центре несколько сильнее зависит от близости к нему - в ценре большинство квартир многокомнатные. Коммунальные?

Построим графики зависимости стоимости от прочих параметров для иллюстрации указанных закономерностей:

In [95]:
```
# Построим графики зависимости стоимости от параметров
build_plot(piter_realty_center, 'rooms', 'last_price',
           'Зависимость цены квартиры в центре от количества комнат', 'Количество комнат')
```

```
build_plot(piter_realty_center, 'city_centers_nearest', 'last_price',
           'Зависимость цены квартиры в центре от расстояния до центра', 'Расстояние до центра, м.')
```

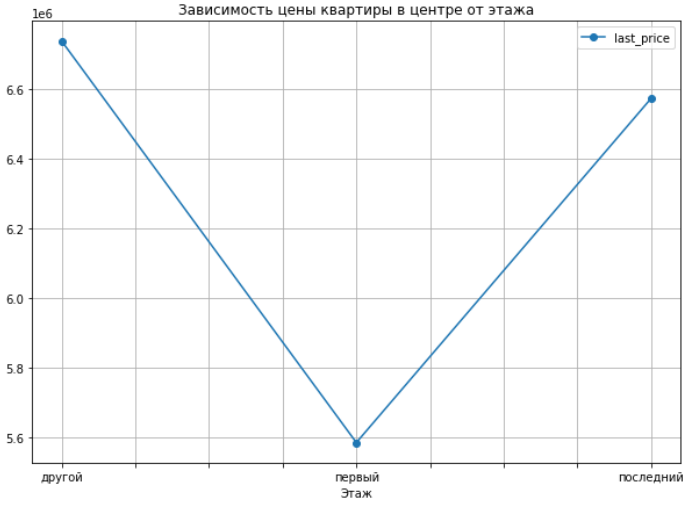




Графики полностью подтверждают сделанные выше выводы.

Рассмотрим теперь зависимость стоимости квартиры `'last_price'` от этажа `'floor_cat'`. Вычислить коэффициент корреляции Пирсона для этих параметров невозможно, т.к. один из них строковый. Поэтому сразу построим график:

In [96]:
```
build_plot(piter_realty_center, 'floor_cat', 'last_price',
           'Зависимость цены квартиры в центре от этажа', 'Этаж')
```



Сравнивая полученный график с аналогичным графиком для всей базы объявлений (см. п. 4.3.2), можно сделать следующие **выводы**:

1. В домах в центре Питера последние этажи престижнее, чем на периферии. Возможно, на это влияют виды (знаменитые Петербургские крыши) и отсутствие коммуникаций, проходящих по чердакам.
2. Квартиры на 1 этаже по-прежнему значительно дешевле прочих.
3. Квартиры в центре на 1 и средних этажах дома в среднем дороже аналогичных квартир по всей базе в 1.45 раза, на последнем этаже - в 1.6 раза.

Рассмотрим зависимость стоимости квартиры `'last_price'` от дня недели `'first_exposition_weekday'`, месяца `'first_exposition_month'` и года размещения объявления `'first_exposition_year'`.

Построим матрицу попарной корреляции Пирсона для этих параметров, поскольку все они числовые:

In [97]:
```
columns = ['last_price', 'first_exposition_weekday',
           'first_exposition_month', 'first_exposition_year']
piter_realty_center[columns].corr().style.background_gradient('coolwarm')
```

Out[97]:

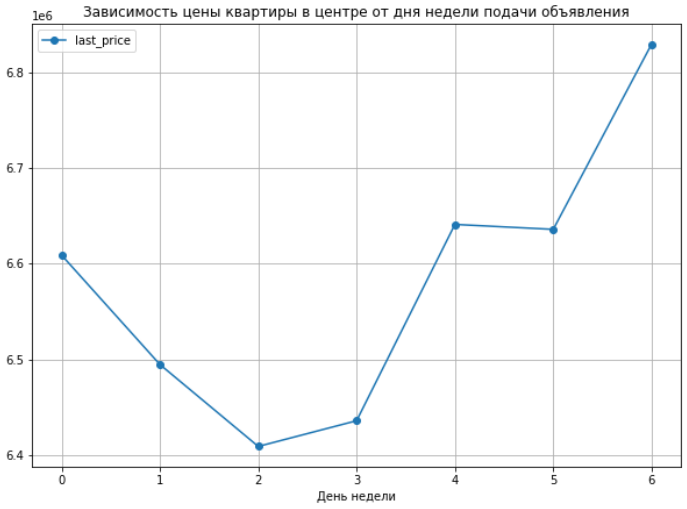
	last_price	first_exposition_weekday	first_exposition_month	first_exposition_year
last_price	1.000000	0.030846	-0.039863	-0.002098
first_exposition_weekday	0.030846	1.000000	0.015774	0.047027
first_exposition_month	-0.039863	0.015774	1.000000	-0.218136
first_exposition_year	-0.002098	0.047027	-0.218136	1.000000

Сравнивая полученные корреляции с рассчитанными ранее для всей базы объявлений (см. п. 4.3.3), можно сделать следующие **выводы**:

1. В центре Питера цена квартиры существенно сильнее зависит от дня недели, в который было размещено объявление, причём в выходные владельцы квартир склонны завышать цены.
2. Зависимость цены в центре от месяца размещения также окрепла и приобрела отрицательный тренд - к концу года квартиры продают дешевле, возможно желая поскорее получить деньги и рассматривать новую покупку в начале следующего года.
3. Зависимость от года, наоборот, ослабла и также приобрела отрицательный тренд, т.е. от 2015 к 2019 году средняя стоимость квартир в центре снижалась.

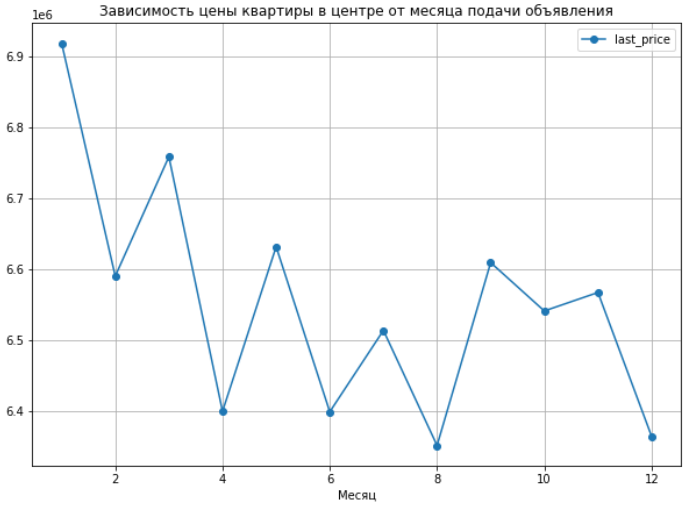
Построим графики зависимости стоимости от прочих параметров для иллюстрации указанных закономерностей:

In [98]:
```
build_plot(piter_realty_center, 'first_exposition_weekday', 'last_price',
           'Зависимость цены квартиры в центре от дня недели подачи объявления', 'День недели')
```



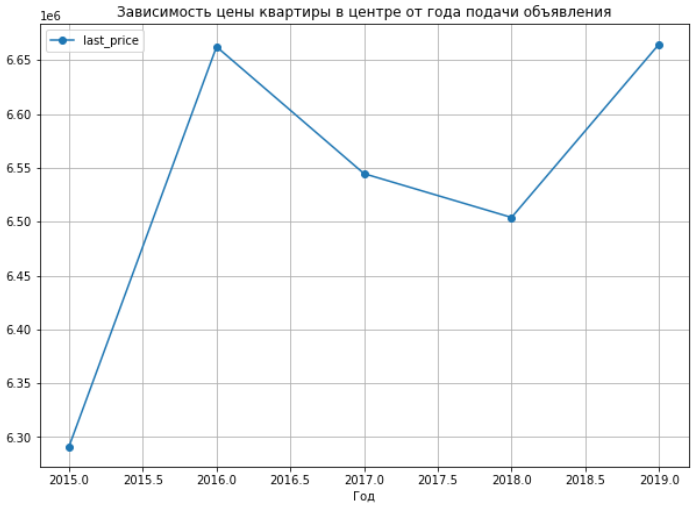
Стоимость квартир проседает к середине недели и резко возрастает к выходным. В целом по базе, как мы помним, в течение недели цены оставались более-менее в покое, а к выходным снижались.

In [99]:
```
build_plot(piter_realty_center, 'first_exposition_month', 'last_price',
           'Зависимость цены квартиры в центре от месяца подачи объявления', 'Месяц')
```



Зависимость цены квартиры в центре от месяца ведёт себя совсем не так, как в целом по базе. Год начинается с высоких цен, затем по август наблюдаются "качели" с общим трендом на снижение до 9%, затем небольшой рост и снова снижение до августовского уровня.

In [100…
```
build_plot(piter_realty_center, 'first_exposition_year', 'last_price',
           'Зависимость цены квартиры в центре от года подачи объявления', 'Год')
```



Зависимость цены квартиры в центре Питера от года публикации объявления в целом ведёт себя в соответствии с общими тенденциями по базе, за исключением необъяснимого изменения тренда в 2015-2016 годах. Если в целом по базе квартиры в 2016 году стоили на 3% дешевле, чем в 2015, то квартиры в центре - наоборот, на 5.5% дороже.

Выводы

Исследовательский анализ данных состоял из следующих этапов:

1. Исследование характеристик объектов недвижимости
2. Анализ времени продажи квартиры
3. Исследование факторов, влияющих на стоимость квартиры
4. Анализ 10 населённых пунктов с наибольшим количеством объявлений
5. Изучение предложения квартир в Санкт-Петербурге
6. Анализ предложений в центре Санкт-Петербурга

В ходе исследования характеристик объектов в целом по базе недвижимости были получены следующие результаты:

- средняя площадь квартир составляет около 58 кв.м., 25% квартир имеют площадь до 39.5 кв.м., 75% квартир - до 67.8 кв.м., стандартное отклонение достаточно велико (велико влияние выбросов)
- средняя стоимость квартиры составляет около 5.12 млн., 25% квартир проданы по цене не выше 3.33 млн., 75% квартир - по цене не выше 6 млн., минимальная по базе цена составляет 12 190 руб. за 100-метровую квартиру - налицо аномальное несоответствие
- в Санкт-Петербурге и окрестностях существуют очень маленькие 4- и более комнатные квартиры
- среди 0-комнатных квартир только 32.58% помечены, как свободная планировка (очевидно, это ошибка ввода данных пользователем - забыли проставить "галочку", 0-комнатных квартир фиксированной планировки не бывает
- средняя высота потолков в квартирах составляет около 2.71 м., 25% квартир имеют высоту потолков до 2.5 м., 75% квартир - до 2.75 м., стандартное отклонение достаточно мало (влияние выбросов невелико)
- максимальное (27.5 м.) и минимальное (1 м.) значения высоты потолков признаны аномалиями

- среднее время продажи по итогам исследования составило 119 дней, медиана - 76 дней, 25% квартир были проданы за время до 39 дней (***быстрые продажи***), 75% квартир - за время до 170 дней (5.7 месяцев), всё, что продавалось дольше можно считать ***долгими продажами***, стандартное отклонение в процессе борьбы с выбросами снизилось - с 206.09 до 112.2, следовательно - влияние выбросов снизилось почти в 2 раза

- цена квартиры больше всего связана с её общей площадью, в меньшей - от количества комнат, слабее всего - от расстояния до центра (обратная зависимость)
- количество комнат существенно зависит от площади, а от расстояния до центра ни количество комнат, ни площадь квартир существенно не зависят
- меньше всего ценится первый этаж (вероятно, по причине близости к подвалу и повышенных рисков), последние этажи - чуть дороже, но не так ценятся как средние этажи дома
- стоимость очень слабо коррелирует с днём недели и месяцем размещения объявления, при этом выставленные в рабочие дни квартиры продаются дороже
- более сильно цена квартир связана с годом размещения (что может объясняться годовыми изменениями экономической ситуации и движениями рынка)
- стоимость квартир слабо колеблется в течение недели, а в выходные продавцы охотнее идут на уступки
- резкий спад цены в июне обусловлен снижением спроса на пике сезона отпусков, снижение цен в январе и марте можно объяснить предпраздничными распродажами и скидками от продавцов, подъём цен в апреле - весенним ростом спроса на недвижимость
- увеличение предложения в 2017 и 2018 годах привело к снижению средней стоимости квартиры (рынок в действии)

- среди 10 городов с наибольшим количеством объявлений дороже всего квадратные метры в Санкт-Петербурге, Пушкине, Кудрово, дешевле всего - в Выборге, Всеволжске и Гатчине

- среднее расстояние до центра в километрах довольно близко к медиане, наблюдается довольно большое стандартное отклонение - возможно люди по-разному измеряют расстояние: одни по дорогам, другие - напрямую
- начиная с 7 километра, стоимость квадратного метра снижается, значит ***зону центра можно ограничить 7 километрами***.
- отмечены артефакты стоимости жилья в центральной зоне (на расстоянии от 0 до 4 км., причину артефактов в центре предположить сложно), а также на периферии (на расстоянии свыше 18 км., могут говорить о, мягко говоря, странной ценовой политике застройщиков новых высотных жилых кварталов)
- при сравнении графиков цены отрасстояния до центра по очищенным и неочищенным данным (во втором случае в качестве функции аггрегации использована медиана, как более устойчивая к выбросам) прослеживаются сходные тенденции - зона центра в неочищенных данных расширилась до 8 км., однако поведение графика в ней указывает что в среднем на протяжении 2015-2019 гг. в 4-километровой зоне квартиры продавались по заниженным ценам, а на периферии - наоборот

- самые большие и самые маленькие квартиры в базе расположены в центре Питера
- средние показатели площади в центре выше соответствующих показателей по всей базе примерно на 15%
- в целом, судя по количеству объявлений в центре, распределение как площадей, так и других параметров в базе определяется распределением в центре Санкт-Петербурга
- в центре нет дешёвых квартир, однако сосредоточены самые дорогие
- средние показатели стоимости квартир в центре выше соответствующих показателей по всей базе примерно в 1.5 раза
- в центре почти нет квартир свободной планировки

- количество 1-комнатных в центре значительно меньше по сравнению с 2-комнатными (в целом по базе - наоборот, лидируют однушки)
- квартиры в центре обладают преимущественно высокими потолками - подавляющее большинство потолков выше 2.7 м. (по всей базе - не выше 2.6 м.), лидерами по количеству являются 3-метровые потолки, средняя высота потолков выросла с 2.6 до 2.8 м. по сравнению с данными по всей базе
- в центре Питера цена квартиры сильнее зависит от количества комнат, зависимость от близости к центру закономерно ослабла
- количество комнат в квартирах в центре сильнее зависит от близости к нему - в ценре большинство квартир многокомнатные (коммунальные?)
- в домах в центре Питера последние этажи престижнее, чем на периферии. Возможно, на это влияют виды (знаменитые Петербургские крыши) и отсутствие коммуникаций, проходящих по чердакам, квартиры на 1 этаже по-прежнему значительно дешевле прочих
- квартиры в центре на 1 и средних этажах дома в среднем дороже аналогичных квартир по всей базе в 1.45 раза, на последнем этаже - в 1.6 раза
- в центре Питера цена квартиры существенно сильнее зависит от дня недели, в который было размещено объявление, причём в выходные владельцы квартир склонны завышать цены (в целом по базе в течение недели цены оставались более-менее в покое, а к выходным снижались)
- зависимость цены в центре от месяца размещения также окрепла и приобрела отрицательный тренд (ведёт себя совсем не так, как в целом по базе) - год начинается с высоких цен, затем по август наблюдаются "качели" с общим трендом на снижение до 9%, затем небольшой рост и снова снижение до августовского уровня
- зависимость от года, наоборот, ослабла и также приобрела отрицательный тренд, т.е. от 2015 к 2019 году средняя стоимость квартир в центре снижалась (в целом ведёт себя в соответствии с общими тенденциями по базе, за исключением необъяснимого изменения тренда в 2015-2016 годах: если в целом по базе квартиры в 2016 году стоили на 3% дешевле, чем в 2015, то квартиры в центре - наоборот, на 5.5% дороже)

Общий вывод и предложения

В ходе проекта исследования проводились в несколько этапов. Подробные выводы по каждому этапу представлены в соответствующих разделах отчёта.

Наиболее сильно на стоимость квартиры влияют следующие параметры (в порядке убывания влияния):

- общая площадь
- количество комнат
- расстояние до центра
- этаж

В этих условиях представляется целесообразным использовать следующую логику построения автоматизированной системы оценки объявлений:

1. Построить на основе статистических данных систему распределений зависимости цены квартиры от её общей площади, ранжированную на несколько уровней:
   - на верхнем уровне - для каждой категории этажа (первый, последний, другой)
   - на следующем уровне - для каждого количества комнат
   - на низшем уровне - для каждого расстояния до центра (в километрах)

1. Для каждого распределения описать его характеристики.

1. При поступлении очередного объявления находить по его параметрам нужное распределение и если заявленная цена отличается от средней более чем на стандартное отклонение - помечать его, как требующее дополнительной проверки.

Замечание 1. *Инженерам по данным следует указать на неполноту описания данных (не указана валюта, в которой выражена стоимость объектов). Данный фактор может влиять на интерпретацию значений стоимости 'last_price'.*

Замечание 2. *Поскольку данные поля заполняются пользователем сервиса недвижимости, целесообразно рекомендовать команде сервиса сделать их обязательными для заполнения (поле `'locality_name'`), либо привязать к значениям других полей и задать значение по умолчанию (поле `'floors_total'`).*

Замечание 3. *В 4644 строках датасета обнаружено одновременное отсутствие значений в столбцах `'ponds_around_3000'`, `'ponds_nearest'`, `'parks_around_3000'`, `'parks_nearest'`, а также `'airports_nearest'` и `'city_centers_nearest'`. Данная ошибка носит технологический характер. Вероятные причины - сбой в работе картографического сервиса или ошибка извлечения данных из него. Следует уведомить команду разработки картографического сервиса и инженеров по данным.*