Содержание

Исследование объявлений о продаже квартир

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

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

Изучение данных из файла

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter('ignore')

data = pd.read_csv('/datasets/real_estate_data.csv', sep='\t')
display(data.head(10))
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartmen
0	20	13000000.0	108.00	2019-03-07T00:00:00	3	2.70	16.0	51.00	8	NaN
1	7	3350000.0	40.40	2018-12-04T00:00:00	1	NaN	11.0	18.60	1	NaN

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

10 rows × 22 columns

In [2]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 22 columns):
total_images            23699 non-null int64
last_price              23699 non-null float64
total_area              23699 non-null float64
first_day_exposition    23699 non-null object
rooms                   23699 non-null int64
ceiling_height          14504 non-null float64
floors_total            23613 non-null float64
living_area             21796 non-null float64
floor                   23699 non-null int64
is_apartment            2775 non-null object
studio                  23699 non-null bool
open_plan               23699 non-null bool
kitchen_area            21421 non-null float64
balcony                 12180 non-null float64
locality_name           23650 non-null object
airports_nearest        18157 non-null float64
cityCenters_nearest     18180 non-null float64
parks_around3000        18181 non-null float64
parks_nearest           8079 non-null float64
```

```
ponds_around3000        18181 non-null float64
ponds_nearest           9110 non-null float64
days_exposition         20518 non-null float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 3.7+ MB
```

In [3]:

```
data.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
```

In [4]:

```
data.describe()
```

Out[4]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floor	kit
count	23699.000000	2.369900e+04	23699.000000	23699.000000	14504.000000	23613.000000	21796.000000	23699.000000	21
mean	9.858475	6.541549e+06	60.348651	2.070636	2.771499	10.673824	34.457852	5.892358	
std	5.682529	1.088701e+07	35.654083	1.078405	1.261056	6.597173	22.030445	4.885249	
min	0.000000	1.219000e+04	12.000000	0.000000	1.000000	1.000000	2.000000	1.000000	
25%	6.000000	3.400000e+06	40.000000	1.000000	2.520000	5.000000	18.600000	2.000000	
50%	9.000000	4.650000e+06	52.000000	2.000000	2.650000	9.000000	30.000000	4.000000	
75%	14.000000	6.800000e+06	69.900000	3.000000	2.800000	16.000000	42.300000	8.000000	
max	50.000000	7.630000e+08	900.000000	19.000000	100.000000	60.000000	409.700000	33.000000	

In [5]:

```
data.corr()
```

Out[5]:

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floor	studio

	total_images	last_price	total_area	rooms	ceiling_height	floors_total	living_area	floor	studio
total_images	1.000000	0.104473	0.115352	0.099288	-0.001987	0.010427	0.104780	0.031340	-0.029303
last_price	0.104473	1.000000	0.653675	0.363343	0.085430	-0.006984	0.566492	0.026576	-0.025362
total_area	0.115352	0.653675	1.000000	0.758344	0.095490	-0.075774	0.939537	-0.024754	-0.072653
rooms	0.099288	0.363343	0.758344	1.000000	0.054457	-0.228215	0.845977	-0.150862	-0.147286
ceiling_height	-0.001987	0.085430	0.095490	0.054457	1.000000	-0.028732	0.090650	-0.011798	0.001674
floors_total	0.010427	-0.006984	-0.075774	-0.228215	-0.028732	1.000000	-0.169311	0.678059	0.070151
living_area	0.104780	0.566492	0.939537	0.845977	0.090650	-0.169311	1.000000	-0.097210	-0.056231
floor	0.031340	0.026576	-0.024754	-0.150862	-0.011798	0.678059	-0.097210	1.000000	0.036940
studio	-0.029303	-0.025362	-0.072653	-0.147286	0.001674	0.070151	-0.056231	0.036940	1.000000
open_plan	-0.024407	-0.008802	-0.034885	-0.087500	0.042777	0.050791	-0.033711	0.035824	-0.004235
kitchen_area	0.104756	0.519869	0.609121	0.269945	0.087641	0.163944	0.428674	0.135531	NaN
balcony	0.121693	0.029646	0.047937	0.017991	0.040523	0.194065	0.018849	0.168773	0.031455
airports_nearest	-0.002298	-0.026239	-0.030753	-0.061199	-0.023947	0.108288	-0.057912	0.071597	-0.021876
cityCenters_nearest	-0.047666	-0.206747	-0.231446	-0.184864	-0.091689	0.019774	-0.231368	0.009084	-0.007029
parks_around3000	0.021120	0.151058	0.164689	0.137257	0.065915	-0.252833	0.184453	-0.163784	-0.030202
parks_nearest	-0.008347	-0.016414	-0.021497	-0.054549	-0.019167	0.097527	-0.050167	0.073045	0.009134
ponds_around3000	-0.011553	0.159996	0.162346	0.092693	0.078209	-0.122735	0.148933	-0.076312	0.000939
ponds_nearest	-0.003034	-0.084809	-0.097969	-0.057689	-0.059090	0.038864	-0.081674	0.024850	0.002606
days_exposition	-0.026657	0.081146	0.149675	0.126961	0.019091	-0.052234	0.142454	-0.039463	-0.022476

In [6]:

```
data.duplicated().sum()
```

Out[6]:

```
0
```

Вывод

Видим, что можно сделать:

- Перевести в тип `int` следующие столбцы:
  - `last_price`
  - `floors_total`
  - `balcony`
  - `airports_nearest`
  - `cityCenter_nearest`
  - `parks_around3000`
  - `parks_nearest`

    - `ponds_around3000`
    - `ponds_nearest`
    - `days_exposition` **(все эти столбцы содержат значения для натуральных чисел, поэтому нет смысла оставлять их в дробном виде)**
- Перевести в тип `date` столбец `first_day_exposition`
- Перевести в тип `boolean` столбец `is_apartment`
- Пропуски:
    - `parks_around3000` и `ponds_around3000` пропуски из-за отсутствия парков/водоемов в радиусе **3** км. Нужно подставить `0`
    - `parks_nearest` и `ponds_nearest`: тут неизвестно, как заменять значения, ведь пруд может находится как в **5** км, так и в **35** км. Трогать пропуски в этих колонках не будем
    - `balcony`: подставить `0` в пропуски. Пропуски скорее всего из-за отсутствия балконов и тем, что клиент не стал указывать кол-во
    - `locality_name` с пустыми значениями можно удалить, там **49** пропусков и это мало повлияет на данные
    - `ceiling_height` для заполнения пропусков узнаем среднее значение высоты, сгруппированное по `locality_name`. Можно было бы еще добавить для группировки `floors_total`, но они слабо друг с другом кореллируют, значит не имеют друг на друга влияние от изменения значений.
    - `living_area` для заполнения пропусков узнаем медианное значение жилой площади, сгрупированное по `locality_name` и `rooms`, исходя из коэфициента корреляции
    - `is_apartment` в этом столбце пропуски вероятнее всего из-за того, что объект не является апартаментом и клиент не стал это указывать. Заменим на `False`
    - `kitchen_area` для заполнения пропусков узнаем медианное значение **(так как был замечены значения площади кухни в 1 кв. м., то во избежание влияния на значение выбросов берем медиану)** площади кухни, сгрупированное по `locality_name` и `total_area`, опираясь на коэфициент корреляции
    - `cityCenter_nearest` и `airport_nearest` для заполнения пропусков узнаем медианное значение расстояния до центра и расстояние до аэропорта соответсвенно, сгрупированное по `locality_name`
    - `days_exposition` для звполнения пропусков узнаем медианное значение дней, сгрупированное по `first_day_exposition`

Слава богу обошлось без дубликатов.

Предобработка данных

Обработка пропусков

In [7]:

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

Out[7]:

```
total_images                0
last_price                  0
total_area                  0
first_day_exposition        0
rooms                       0
ceiling_height           9195
floors_total               86
living_area              1903
floor                       0
is_apartment            20924
studio                      0
open_plan                   0
kitchen_area             2278
balcony                 11519
locality_name              49
airports_nearest         5542
cityCenters_nearest      5519
parks_around3000         5519
```

```
parks_around3000         5518
parks_nearest           15620
ponds_around3000         5518
ponds_nearest           14589
days_exposition          3181
dtype: int64
```

In [8]:

```
# Заполним нулями очевидные пропуски:
data['parks_around3000'] = data['parks_around3000'].fillna(0)
data['ponds_around3000'] = data['ponds_around3000'].fillna(0)
data['balcony'] = data['balcony'].fillna(0)

# Удалим те самые 49 пропусков в locality_name:
data = data.dropna(subset=['locality_name'])

# Далее будем подставлять медианы и средние по сгруппированным значениям:
data['ceiling_height'] = data['ceiling_height'].fillna(data.groupby('locality_name')['ce
iling_height'].transform('mean'))
data['living_area'] = (
    data['living_area']
    .fillna(data.groupby(['locality_name', 'rooms'])['living_area'].transform('median'))
)
data['is_apartment'] = data['is_apartment'].fillna(False)
data['kitchen_area'] = (
    data['kitchen_area']
    .fillna(data.groupby(['locality_name', 'total_area'])['kitchen_area'].transform('med
ian'))
)
data['city_nearest'] = (
    data['cityCenters_nearest']
    .fillna(data.groupby('locality_name')['cityCenters_nearest'].transform('median'))
)
data['airports_nearest'] = (
    data['airports_nearest']
    .fillna(data.groupby('locality_name')['airports_nearest'].transform('median'))
)
data['days_exposition'] = data['days_exposition'].fillna(data.groupby('first_day_exposit
ion')['days_exposition'].transform('median'))

# Обработаем оставшиеся пропуски
data['ceiling_height'] = data['ceiling_height'].fillna(data['ceiling_height'].median())
data['floors_total'] = data['floors_total'].fillna(data['floors_total'].median())
data['living_area'] = data['living_area'].fillna(data['living_area'].median())
data['kitchen_area'] = data['kitchen_area'].fillna(data.groupby('locality_name')['kitchen
_area'].transform('median'))
data['kitchen_area'] = data['kitchen_area'].fillna(data['kitchen_area'].median())
data.isna().sum()
```

Out[8]:

```
total_images                0
last_price                  0
total_area                  0
first_day_exposition        0
rooms                       0
ceiling_height              0
floors_total                0
living_area                 0
floor                       0
is_apartment                0
studio                      0
open_plan                   0
kitchen_area                0
balcony                     0
locality_name               0
airports_nearest         5386
cityCenters_nearest      5511
parks_around3000            0
parks_nearest           15586
ponds_around3000            0
ponds_nearest           14565
```

```
days_exposition            111
city_nearest              5386
dtype: int64
```

Я решила не трогать оставшиеся пропущенные значения `airports_nearest` и `cityCenters_nearest`. Заменять медианой по всему датафрейму будет необъективно. Скорее всего, пропущенные значения остались, т.к. в некоторых группах в принципе нет значений для подсчета среднего/медианы.

Замена типа данных

In [9]:

```
import numpy as np

# Переведем в тип Int64:
data['last_price'] = data['last_price'].astype('Int64')
data['floors_total'] = data['floors_total'].astype('Int64')
data['balcony'] = data['balcony'].astype('Int64')
data['airports_nearest'] = np.floor(pd.to_numeric(data['airports_nearest'], errors='coer
ce')).astype('Int64')
data['cityCenters_nearest'] = data['cityCenters_nearest'].astype('Int64')
data['parks_around3000'] = data['parks_around3000'].astype('Int64')
data['parks_nearest'] = data['parks_nearest'].astype('Int64')
data['ponds_around3000'] = data['ponds_around3000'].astype('Int64')
data['ponds_nearest'] = data['ponds_nearest'].astype('Int64')
data['days_exposition'] = np.floor(pd.to_numeric(data['days_exposition'], errors='coerce
')).astype('Int64')

# Переведем в тип datetime колонку first_day_exposition:
data['first_day_exposition'] = pd.to_datetime(data['first_day_exposition'], format='%Y-%
m-%dT%H:%M:%S')

# Переведем колонку is_apartment в булев тип:
data['is_apartment'] = data['is_apartment'].astype('bool')

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>

Int64Index: 23650 entries, 0 to 23698

Data columns (total 23 columns):

total_images            23650 non-null int64

last_price              23650 non-null Int64

total_area              23650 non-null float64

first_day_exposition    23650 non-null datetime64[ns]

rooms                   23650 non-null int64

ceiling_height          23650 non-null float64

floors_total            23650 non-null Int64

living_area             23650 non-null float64

floor                   23650 non-null int64

is_apartment            23650 non-null bool

studio                  23650 non-null bool

open_plan               23650 non-null bool

kitchen_area            23650 non-null float64
```

```
balcony                 23650 non-null Int64

locality_name           23650 non-null object

airports_nearest        18264 non-null Int64

cityCenters_nearest     18139 non-null Int64

parks_around3000        23650 non-null Int64

parks_nearest           8064 non-null Int64

ponds_around3000        23650 non-null Int64

ponds_nearest           9085 non-null Int64

days_exposition         23539 non-null Int64

city_nearest            18264 non-null float64

dtypes: Int64(10), bool(3), datetime64[ns](1), float64(5), int64(3), object(1)

memory usage: 4.1+ MB
```

Дубликаты

In [10]:

```
# Посчитаем количество явных дубликатов.

data.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

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

Out[11]:

```
Санкт-Петербург               15721
посёлок Мурино                  522
посёлок Шушары                  440
Всеволожск                      398
Пушкин                          369
                              ...
поселок Севастьяново              1
деревня Зимитицы                  1
посёлок Белоостров                1
деревня Мануйлово                 1
садовое товарищество Садко        1
Name: locality_name, Length: 364, dtype: int64
```

Увидели, что есть значения `поселок` и `посёлок`, приведем их к единому значению:

In [12]:

```
data['locality_name'] = data['locality_name'].str.replace('ё', 'е')
data['locality_name'].value_counts()
```

Out[12]:

```
Санкт-Петербург               15721
поселок Мурино                  556
поселок Шушары                  440
Всеволожск                      398
Пушкин                          369
                              ...
деревня Кривко                    1
```

```
деревня Пикколово                   1
коттеджный поселок Лесное           1
поселок Красносельское              1
садовое товарищество Садко          1
Name: locality_name, Length: 330, dtype: int64
```

Расчёты и добавление результатов в таблицу

In [13]:

```
# Добавим в таблицу цену за квадратный метр

data['price_per_meter'] = data['last_price'] / data['total_area']
data['price_per_meter'] = data['price_per_meter'].round().astype('int') # Переведем в це
лочисленный тип
data['price_per_meter']
```

Out[13]:

```
0        120370
1         82921
2         92786
3        408176
4        100000
          ...
23694     72491
23695     52542
23696     44092
23697    149511
23698     41796
Name: price_per_meter, Length: 23650, dtype: int64
```

In [14]:

```
# Добавим в таблицу день недели
data['weekday'] = data['first_day_exposition'].dt.weekday

# Добавим в таблицу месяц
data['month'] = data['first_day_exposition'].dt.month

# Добавим в таблицу год
data['year'] = data['first_day_exposition'].dt.year

# Создадим функцию для категоризации этажей:
def create_category_floor(data):
    if data['floor'] == 1:
        return 1 # Первый этаж

    if data['floor'] == data['floors_total'] and data['floor'] != 1:
        return 3 # Последний этаж

    else:
        return 2 # Другой этаж

data['category_floor'] = data.apply(create_category_floor, axis=1)

# Добавим столбцы с отношениями жилой и кухонной площадей к общей
data['ratio_living_to_total'] = data['living_area'] / data['total_area']
data['ratio_kitchen_to_total'] = data['kitchen_area'] / data['total_area']

data.head(10)
```

Out[14]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment
0	20	13000000	108.00	2019-03-07	3	2.700000	16	51.00	8	False
1	7	3350000	40.40	2018-12-04	1	2.644470	11	18.60	1	False
2	10	5196000	56.00	2015-08-20	2	2.803709	5	34.30	4	False

3	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	False
4	2	10000000	100.00	2018-06-19	2	3.030000	14	32.00	13	False	
5	10	2890000	30.40	2018-09-10	1	2.617600	12	14.40	5	False	
6	6	3700000	37.30	2017-11-02	1	2.630292	26	10.60	6	False	
7	5	7915000	71.60	2019-04-18	2	2.803709	24	31.00	22	False	
8	20	2900000	33.16	2018-05-23	1	2.938900	27	15.43	26	False	
9	18	5400000	61.00	2017-02-26	3	2.500000	9	43.60	7	False	

10 rows × 30 columns

Исследовательский анализ данных

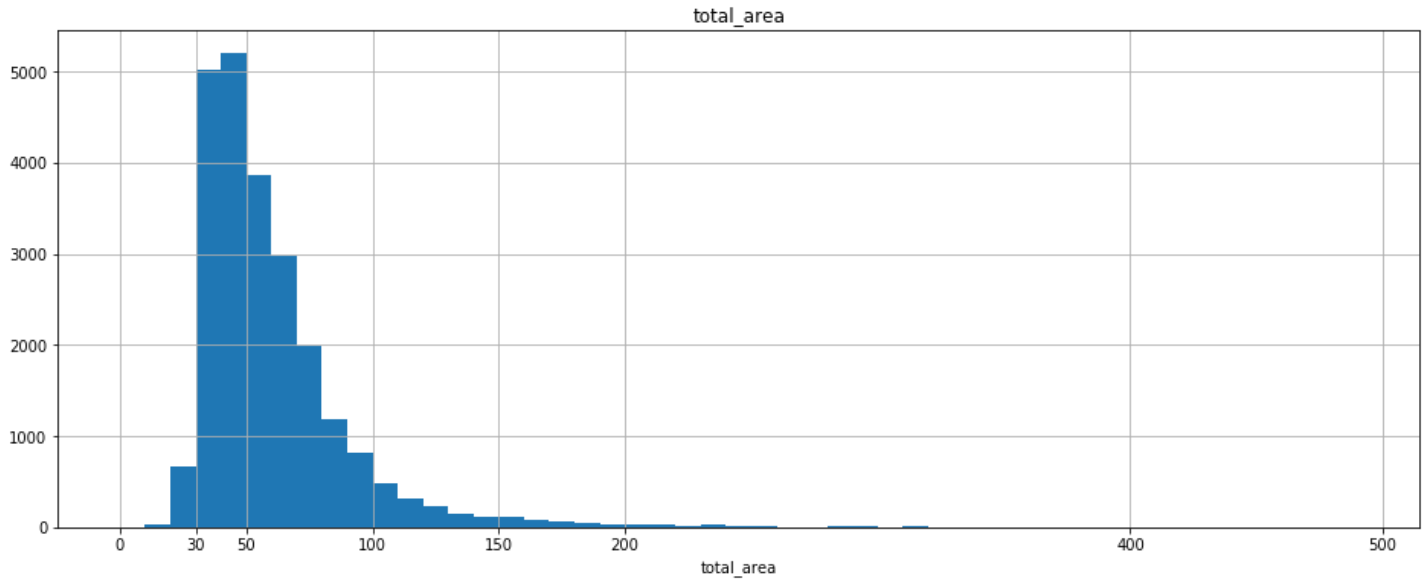
Исследование площади, цены, числа комнат, высоты потолков

In [15]:

```
data['total_area'].hist(bins=range(0,500,10), figsize=(16, 6))
plt.xticks(ticks=(0,30,50,100,150,200,400,500))
plt.xlabel('total_area')
plt.title('total_area')
```

Out[15]:

```
Text(0.5, 1.0, 'total_area')
```



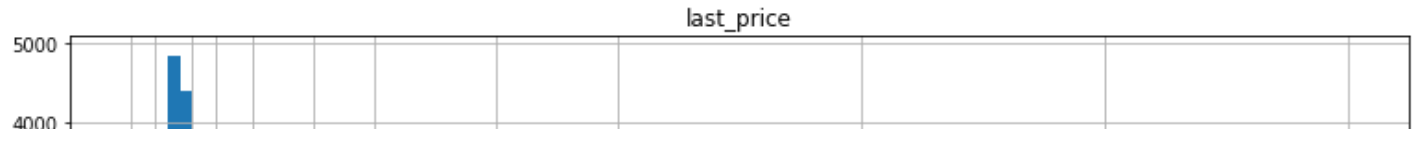
Большинство значений приходится на объекты площадью от **30** до **70** кв. м.

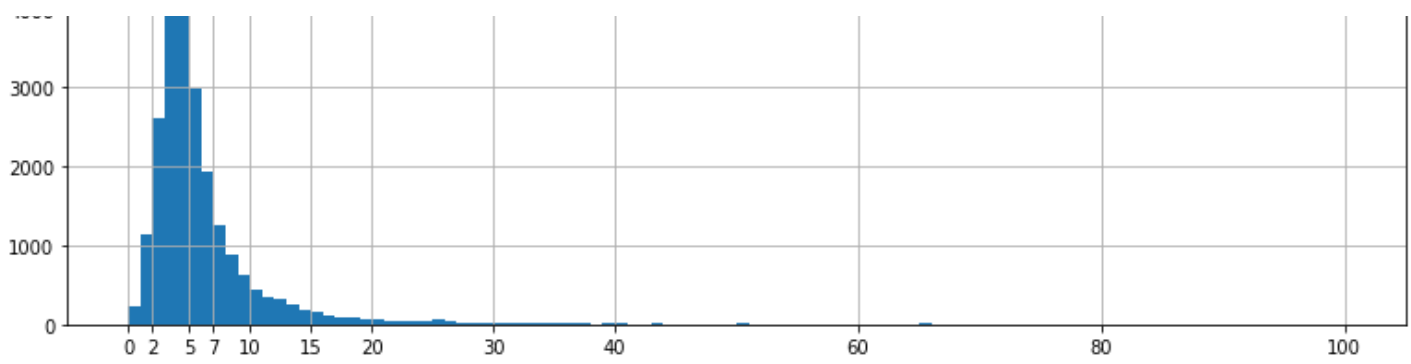
In [16]:

```
data['last_price'] = data['last_price'] / 1000000


data['last_price'].hist(bins=100, range=(0, 100), figsize=(13,4))
plt.xticks(ticks=(0,2,5,7,10,15,20,30,40,60,80,100))
plt.title('last_price')
```

Out[16]:

```
Text(0.5, 1.0, 'last_price')
```



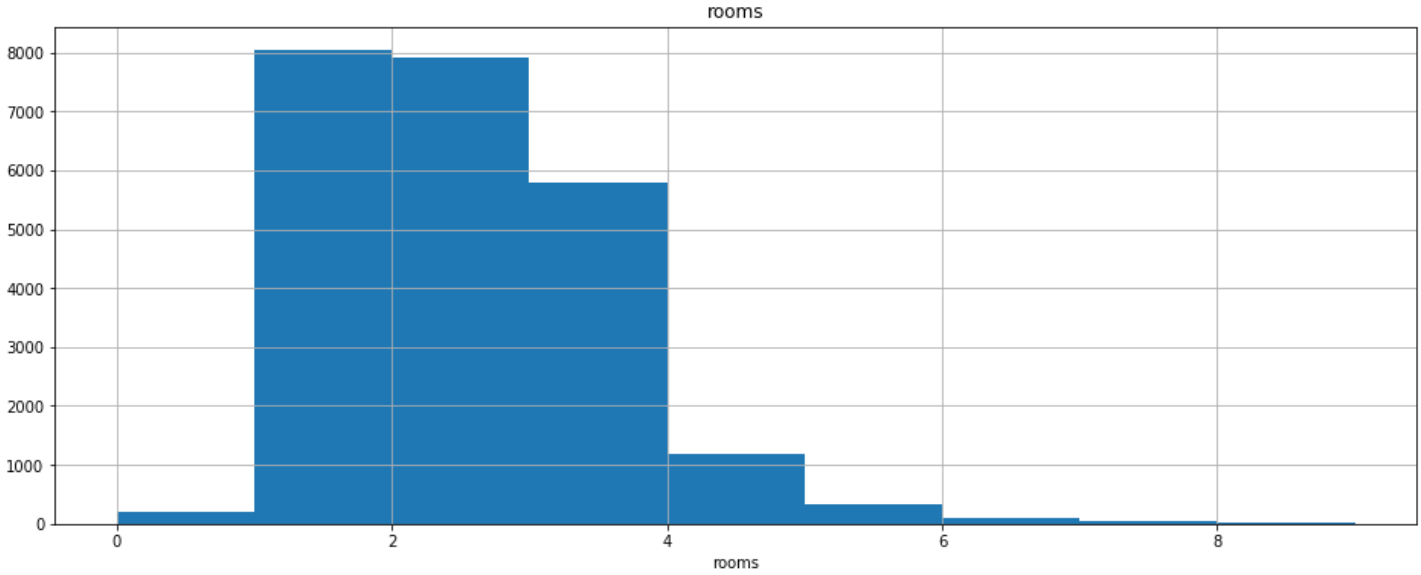


In [17]:

```
data['rooms'].hist(bins=range(0,10,1), figsize=(16, 6))
plt.xlabel('rooms')
plt.title('rooms')
```

Out[17]:

```
Text(0.5, 1.0, 'rooms')
```

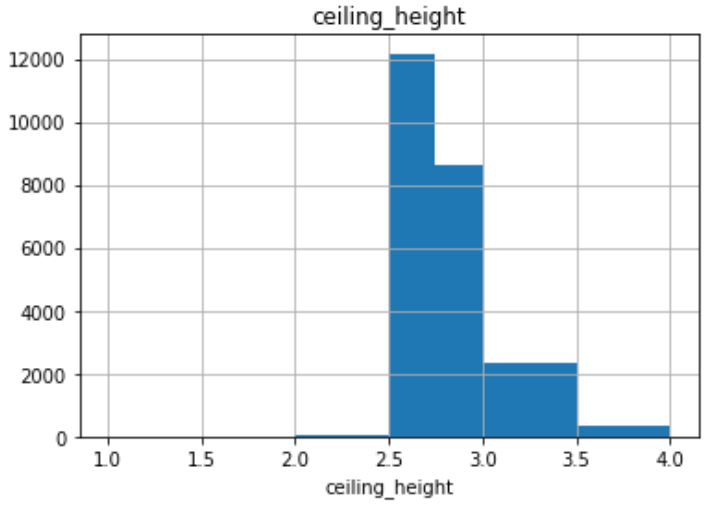


In [18]:

```
data['ceiling_height'].hist(bins=[1,2,2.5,2.75,3,3.5,4])
#plt.xticks(ticks=(2, 2.5, 3, 3.5, 4, 5))
plt.xlabel('ceiling_height')
plt.title('ceiling_height')
```

Out[18]:

```
Text(0.5, 1.0, 'ceiling_height')
```



Вывод

Гистограммы выглядят ожидаемо, можно избавиться от хвостов с слишком высокими значениями.

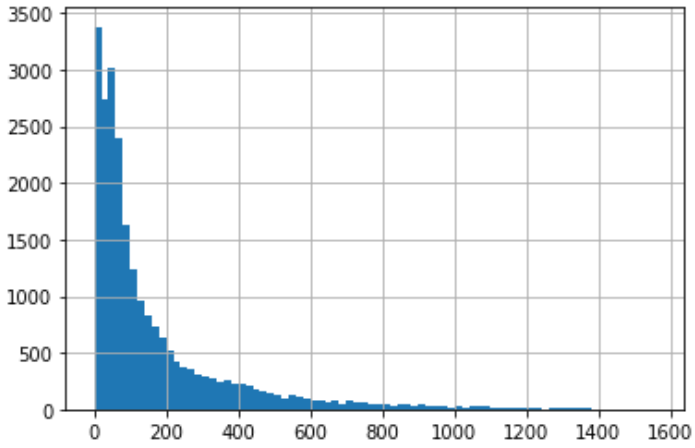
Исследование количества времени для продажи квартиры

In [19]:

```
data['days_exposition'].hist(bins=range(0,1580,20))
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd26156c710>
```



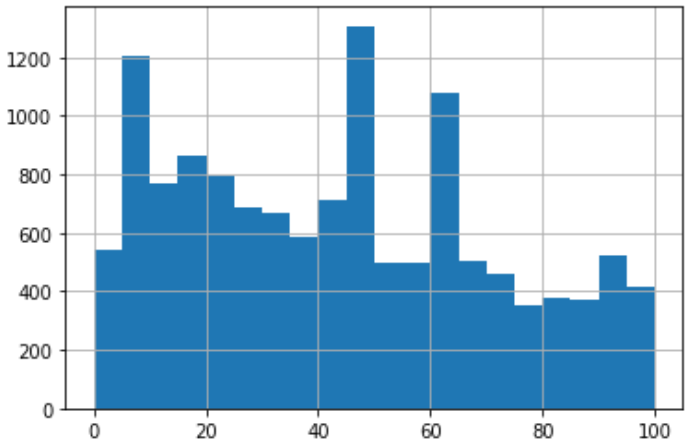
Странно, что есть пик около **0 (20** дней), выбивается также значение в **60** дней, таких чуть больше **3000** случаев. Надо "укрупнить эту зону".

In [20]:

```
data['days_exposition'].hist(range=(0,100), bins=20)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2613b9510>
```



Видим, что в этой гистограмме значения с **10, 50** и **65** дней выбиваются из общего ряда данных, таких случаев больше **1000** для каждого.
Можно выдвинуть **гипотезу,** что сняли с объявления в течение **10** дней из-за того, что передумали. В оставшихся двух диапазонах **(50** и **60** дней), могут делать срочные продажи однокомнатных квартир, что тоже возможно.

In [21]:

```
print('Медианное значение количества дней:', data['days_exposition'].median())
print('Среднее значение количества дней:', data['days_exposition'].mean())
```

```
Медианное значение количества дней: 83.0

Среднее значение количества дней: 166.3924125918688
```

In [22]:

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

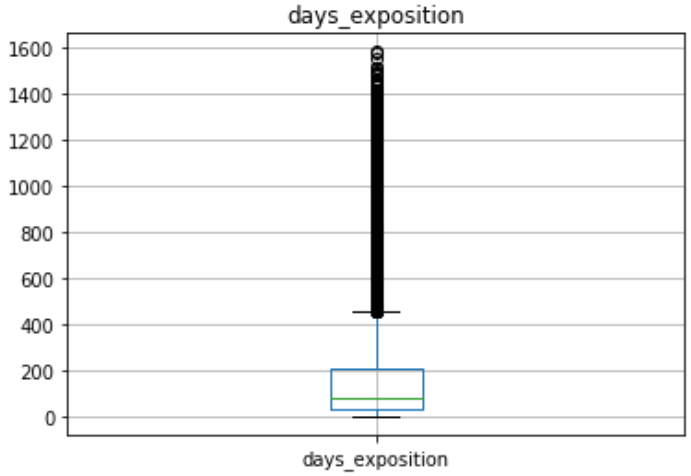
Out[22]:

```
count    23539.000000
mean       166.392413
std        214.029705
min          1.000000
25%         37.000000
50%         83.000000
75%        205.000000
max       1580.000000
Name: days_exposition, dtype: float64
```

Можно увидеть, что среднее в два раза больше медианы, из чего следует, что в распределении есть длинный хвост с высокими значениями или несколько очень высоких значений. Это влияет на среднее.

In [23]:

```
data.boxplot('days_exposition')
plt.title('days_exposition')
plt.show()
```



На диаграмме размаха видно, что большая часть значений лежит в пределе до **200** дней. Что выглядит нормально. Но при этом **1**-й квартиль лежит на уровне менее **40** дней, это уже подозрительно

In [24]:

```
data.sort_values('days_exposition', ascending=False).head(10)
```

Out[24]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartr
18843	11	3.500000	48.20	2014-12-15	2	2.500000	5	27.40	2	F
1109	1	34.878556	95.80	2014-12-09	2	2.803709	6	58.30	5	
9553	11	10.600000	80.00	2014-12-09	3	2.850000	17	44.00	9	F
1885	2	13.300000	79.60	2014-12-09	2	3.000000	8	42.70	7	F
20969	18	13.300000	133.00	2015-02-11	4	3.000000	5	58.00	2	F
6691	3	4.400000	42.98	2014-12-12	1	2.706667	5	18.72	1	F
14093	7	9.500000	100.00	2015-02-19	4	3.000000	5	76.00	5	F
15313	1	9.278000	139.80	2014-12-28	4	2.800000	5	67.45	4	F
19123	9	5.990000	82.00	2015-02-19	3	3.000000	4	60.00	2	F
11955	4	11.400000	76.00	2014-12-09	2	2.803709	9	36.00	3	F

10 rows × 30 columns

Пробежавшись по строкам с самыми высокими значениями дней продаж, можно увидеть, что **7** из **10** самых долгосрочных объявлений приходится на декабрь **2014** года. Посмотрим, сколько объявлений приходится на декабрь **2014** года:

In [25]:

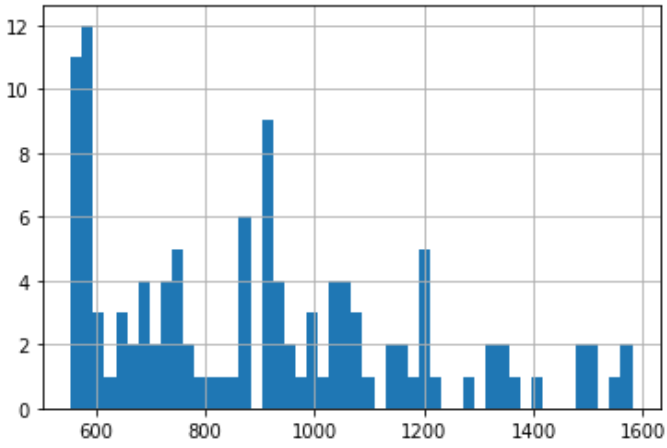
```
sample = data.query('year == 2014 and month == 12')

print(sample['year'].count())
print(data.query('year == 2014')['year'].count())
```

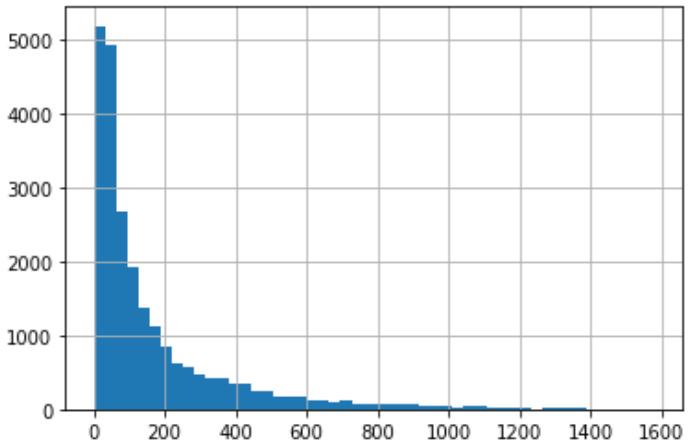
```
116

136
```

Сперва сравним распределение времени в декабре **2014** года со всеми годами.

In [26]:

```
sample['days_exposition'].hist(bins=50)
plt.show()
data['days_exposition'].hist(range=(0, 1580), bins=50)
plt.show()
```





В графике за декабрь **2014** года минимальное значение начинается с **300** дней **(**чуть меньше года**).** Возможно это тестовые объявления**,** про которые все забыли**.**

Вывод

Обычно продажа занимает около **200** дней**.** Необычно долгие продажи длятся более **400** дней **(**год **3** месяца**).** Необычно быструю продажу можно взять за **40** дней и меньше**..**

Анализ редких и выбивающихся значений

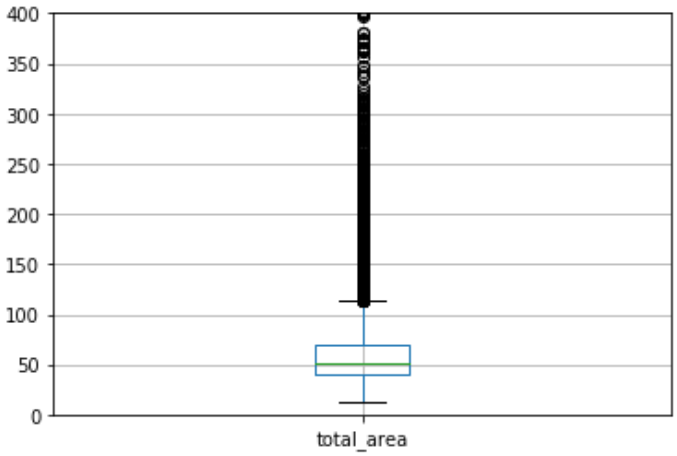
Обязательно стоит изучить площадь, цену, число комнат, высота потолков, время продажи

In [27]:

```
plt.ylim(0, 400) # Увеличим масштабы диаграммы размаха
data.boxplot('total_area')
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd261d60090>
```



Начнем с **общей площади.** Если посмотреть на диаграммы в пункте **4.1** и на диаграмму размаха, то объектов с общей площадью более **110** метров **(**примерно**)** уже считаются выбросами. Объектов в **150-200** кв.м. еще достаточно, поэтому из данных уберем объекты площадью более **220** кв.м., их гораздо меньше и они мало повлияют на данные.

In [28]:

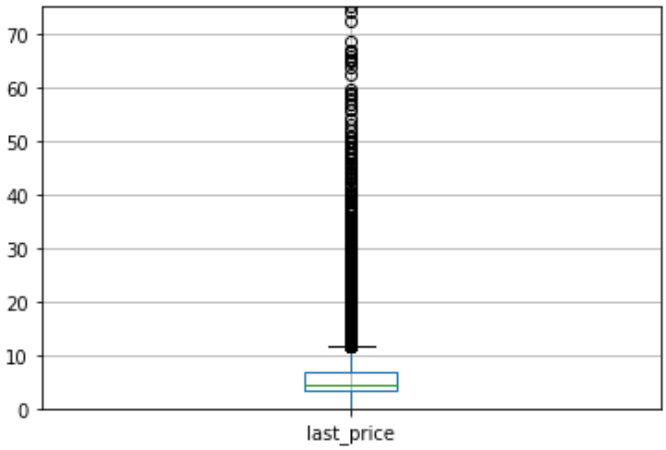
```
normal_data = data.query('total_area < 200')
```

In [29]:

```
plt.ylim(0,75)
normal_data.boxplot('last_price')
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd261313290>
```



In [30]:

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

Out[30]:

```
count    23419.000000
mean         6.021068
std          6.229997
```

```
min           0.012190
25%           3.400000
50%           4.600000
75%           6.700000
max         330.000000
Name: last_price, dtype: float64
```

Выбросим хвост с совсем малым количество значений, где цена больше **37** млн.

In [31]:

```
normal_data = normal_data.query('last_price < 37')
```

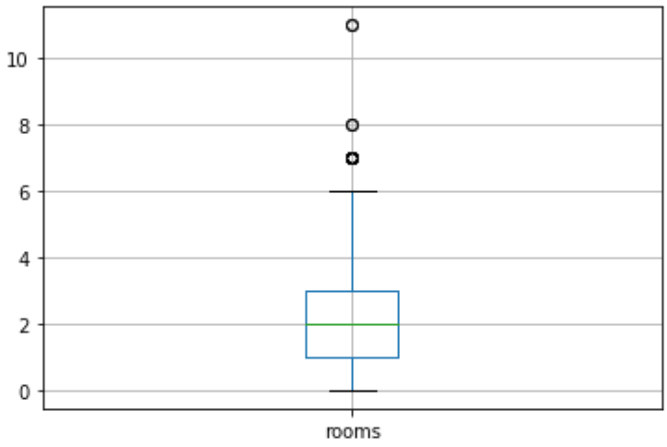
Комнаты: гистограмма в пункте **4.1** ожидаема, поэтому можно выбросить хвост из совсем больших значений. Перед этим снова построим диаграмму размаха для комнаты.

In [32]:

```
normal_data.boxplot('rooms')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd262867c10>
```



Большая часть значений лежит в пределах **3**-х комнат, ожидаемо. Тогда избавимся от объектов более **7** комнат, т.к. их, если судить по диаграмме в пункте **4.1** мало.

In [33]:

```
normal_data = normal_data.query('rooms <= 7')
```

Изучим **высоту потолков:**

In [34]:

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

Out[34]:

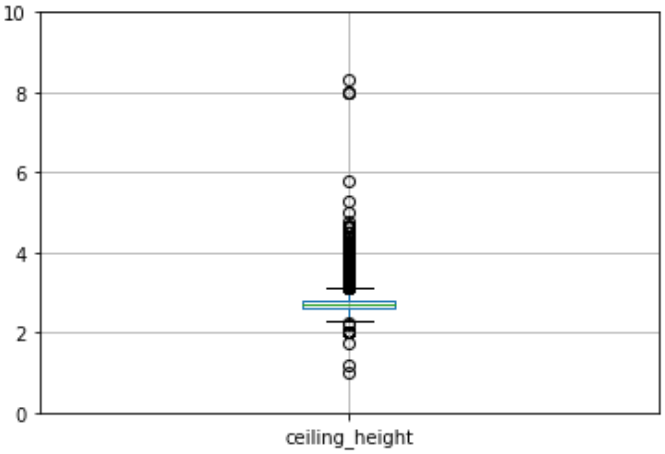
```
count    23295.000000
mean         2.765086
std          1.014955
min          1.000000
25%          2.600000
50%          2.724262
75%          2.803709
max        100.000000
Name: ceiling_height, dtype: float64
```

In [35]:

```
plt.ylim(0, 10)
normal_data.boxplot('ceiling_height')
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd26105aa50>
```



Если судить по диаграмме размаха, то выбросы есть. Но их настолько мало, что они практически не влияют на среднее, а среднее почти равны с медианой, если судить по описанию. Но так или иначе, уберем эти выбросы: меньше **2**-х метров и больше **4**-х метров.

In [36]:

```
normal_data = normal_data.query('2 <= ceiling_height <= 4')
```

Время продажи: часть исследований мы уже произвели в пункте **4.2.**

Посмотрим, как соотносятся быстрые и медленные объявления к нормальным в каждом году

In [37]:

```
normal_data['too_fast'] = (normal_data['days_exposition'] < 20) | (normal_data['days_exposition'].isna())
normal_data['too_slow'] = (normal_data['days_exposition'] > 730) | (normal_data['days_exposition'].isna())
too_fast_stat = normal_data.pivot_table(index='year', values='too_fast')
too_fast_stat

#normal_data.pivot_table(index='rooms', values=['last_price', 'days_exposition'], aggfunc=['median', 'count'])
```

Out[37]:

	too_fast
year	
2014	**0.000000**
2015	**0.003552**
2016	**0.016660**
2017	**0.096111**
2018	**0.127288**
2019	**0.546610**

In [38]:

```
too_slow_stat = normal_data.pivot_table(index='year', values='too_slow')
too_slow_stat
```

Out[38]:

	too_slow

year	too_slow
2014	0.651163
year	
2015	0.372114
2016	0.094409
2017	0.002368
2018	0.000832
2019	0.033192

Делаем вывод, что в **2014** году **65%** являются очень долгими. В **2019** году наоборот: очень много краткосрочных объявлений, целых **54%**.

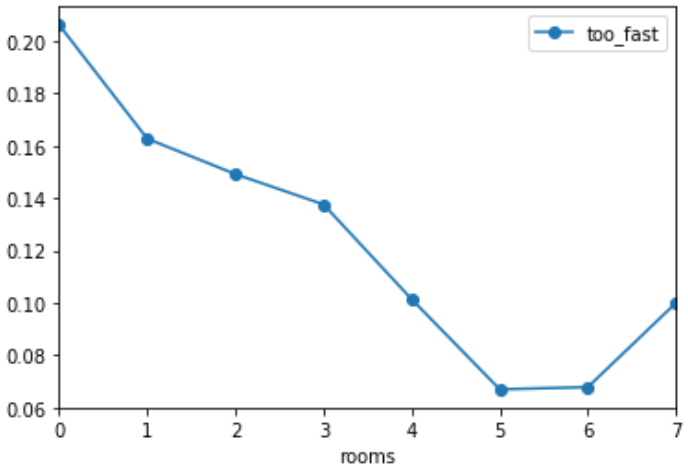
Зависит ли длительность объявления от характеристик квартиры:

In [39]:

```
# Посмотрим, как зависит длительность объявления от количества комнат
normal_data.pivot_table(index='rooms', values='too_fast').plot(style='o-')
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd260fd9c10>
```



Очевидно, что студии **(0** комнат**)** и однокомнатные имеют бОльшую долю очень быстрых объявлений.

Т.к. во всех годах, так или иначе, присутствуют очень быстрые и очень медленные объявления, то я сделаю фильтрацию по всему датафрейму:

In [40]:

```
filtered_data = normal_data.query('20 <= days_exposition <= 720') # Отфильтруем таблицу о
т аномально коротких и долгих объявлений
```

Исследование факторов, влияющих на стоимость квартиры

С помощью коэффициента Пирсона проследим взаимосвязь цены квартиры и цены за кв. м. от следующих показателей:

- общая площадь
- кол-ва комнат
- приближенность к центру
- день недели
- месяц
- год
- на каком этаже: первый, последний, другой

In [41]:

```
# Создадим отдельную таблицу с нужными для нас колонками:
filtered_data.columns
```

Out[41]:

```
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', 'city_nearest', 'price_per_meter',
       'weekday', 'month', 'year', 'category_floor', 'ratio_living_to_total',
       'ratio_kitchen_to_total', 'too_fast', 'too_slow'],
      dtype='object')
```

In [42]:

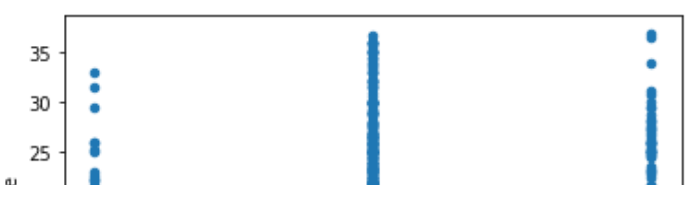
```
table = filtered_data.drop(columns=['total_images', 'first_day_exposition', 'ceiling_heig
ht', 'floors_total',
                                    'living_area', 'is_apartment', 'studio', 'open_plan',
'kitchen_area',
                                    'balcony', 'locality_name', 'airports_nearest', 'park
s_around3000',
                                    'parks_nearest', 'ponds_around3000', 'ponds_nearest',
'days_exposition',
                                    'city_nearest', 'ratio_living_to_total',
                                    'ratio_kitchen_to_total', 'too_fast', 'too_slow', 'fl
oor'])

table.corr()
```

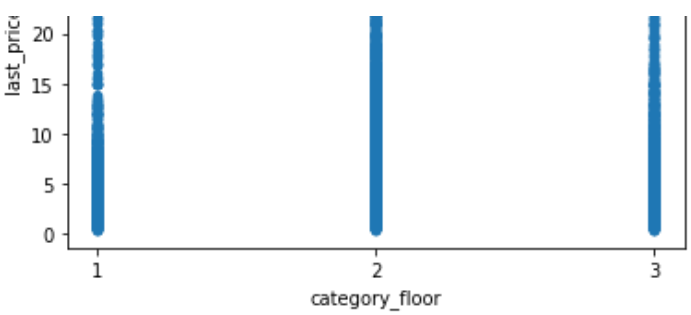
Out[42]:

	last_price	total_area	rooms	cityCenters_nearest	price_per_meter	weekday	month	year	categ
last_price	1.000000	0.785208	0.492713	-0.344972	0.700637	-0.014862	0.000091	-0.002886	
total_area	0.785208	1.000000	0.795608	-0.232152	0.183048	-0.013873	0.002465	-0.043615	
rooms	0.492713	0.795608	1.000000	-0.165027	-0.050160	-0.004170	0.000041	-0.033539	
cityCenters_nearest	-0.344972	-0.232152	-0.165027	1.000000	-0.386919	0.007935	-0.007117	0.016250	
price_per_meter	0.700637	0.183048	-0.050160	-0.386919	1.000000	-0.011127	-0.003838	0.049179	
weekday	-0.014862	-0.013873	-0.004170	0.007935	-0.011127	1.000000	0.001691	-0.001618	
month	0.000091	0.002465	0.000041	-0.007117	-0.003838	0.001691	1.000000	-0.233106	
year	-0.002886	-0.043615	-0.033539	0.016250	0.049179	-0.001618	-0.233106	1.000000	
category_floor	0.059833	0.049949	0.012954	-0.028319	0.043353	0.003891	0.008192	0.002786	

In [43]:

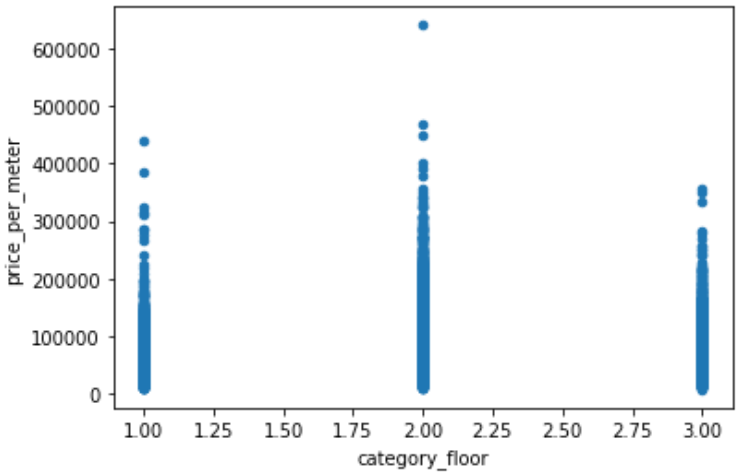
```
table.plot(x='category_floor', y='last_price', kind='scatter')
plt.xticks(ticks=(1, 2, 3))
plt.show()
table.plot(x='category_floor', y='price_per_meter', kind='scatter')
```





Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd258cb7bd0>
```



Вывод

1. Другие этажи чаще дороже, чем первые и последние
2. День недели, месяц, год никак не влияют на цену
3. Общая площадь сильно влияет на цену
4. Число комнат также влият на общую стоимость квартиры
5. Чем дальше от центра, тем дешевле стоимость квартиры и цена за кв. м.

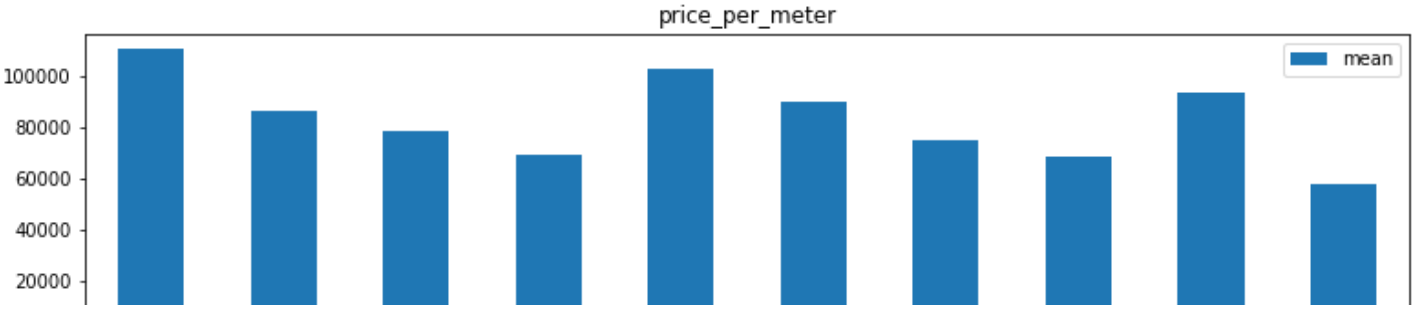
Анализ десяти населённых пунктов с наибольшим числом объявлений

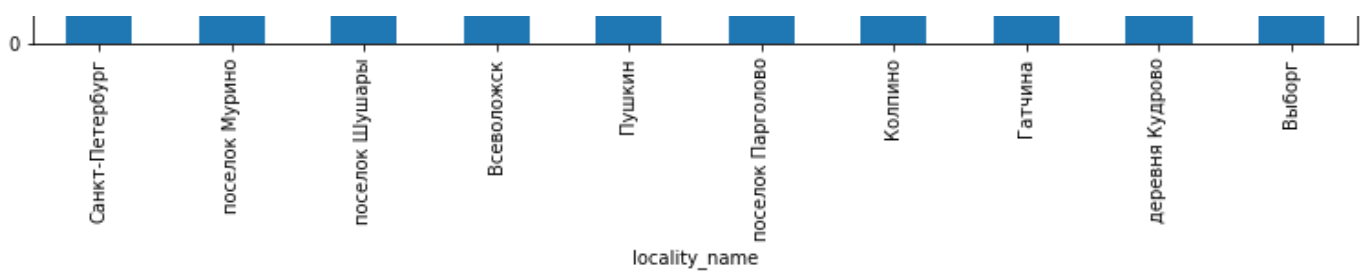
In [44]:

```
# Выведем на экран топ-10 по кол-ву обявлений городов со средней ценой за кв. м. в каждом городе

table1 = (
    filtered_data.pivot_table(index='locality_name', values='price_per_meter', aggfunc=['mean', 'count'])
)
table1.columns = ['mean', 'count']
table1.sort_values('count', ascending=False).head(10).plot(kind='bar', y='mean', figsize=(13,3))
plt.title('price_per_meter')
```

Out[44]:

```
Text(0.5, 1.0, 'price_per_meter')
```





Ожидаемо увидеть на первом месте Санкт-Петербург. На втором также довольно популярный поселок Мурино.

In [45]:

```
# Выведем на экран топ-5 мест с самыми дорогими объектами
filtered_data.pivot_table(index='locality_name', values='last_price').sort_values('last_price', ascending=False).head()
```

Out[45]:

locality_name	last_price
поселок Репино	12.166854
деревня Бор	10.397600
поселок Александровская	8.725000
Санкт-Петербург	6.930719
Сестрорецк	6.425404

In [46]:

```
# Выведем на экран топ-5 мест с самыми дешевыми объектами
filtered_data.pivot_table(index='locality_name', values='last_price').sort_values('last_price').head()
```

Out[46]:

locality_name	last_price
деревня Вахнова Кара	0.4500
деревня Старополье	0.4600
поселок Совхозный	0.5175
поселок станции Свирь	0.5750
деревня Выскатка	0.5850

Вывод

В топе самых дорогих объектов можно понять пункт с поселком Репино, все-таки курортный поселок, возможны дорогие объекты. Поселок Александровская также граничит с городом. С Санкт-Петербургом тоже неудивительно. Сестрорецк тоже является курортным городом. А вот деревня Бор кажется импостером.

Анализ изменения цены по степени удалённости от центра

In [47]:

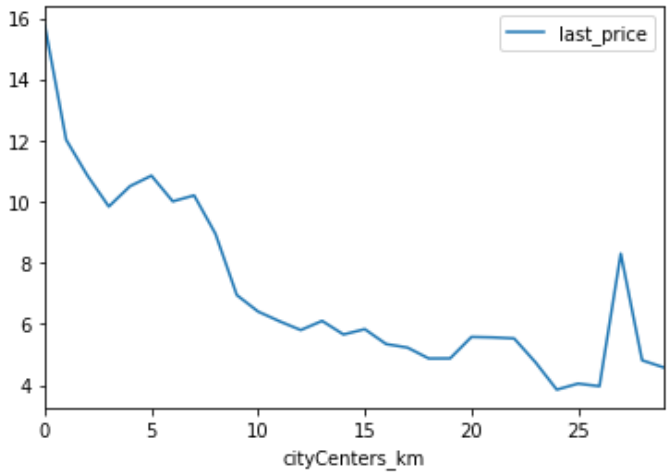
```
spb = filtered_data.query('locality_name in "Санкт-Петербург"')

spb['cityCenters_km'] = spb['cityCenters_nearest'] / 1000
spb['cityCenters_km'] = spb['cityCenters_km'].round().astype('Int64')
```

In [48]:

```
# Сделаем сводную таблицу, сгруппировав по километрам, за значения возьму цену и сразу по строю график. Бритва Окамы, юху!
spb.pivot_table(index='cityCenters_km', values='last_price').plot()
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd258dc8210>
```



Упс, видно выброс в промежутке от **26** км до **29** км. Это аномальные значения. Делать мы с этим, конечно же, ничего не будем.

Вывод

Выяснили, что большинство квартир в центре находятся в диапазоне от **3**-х до **7** километров. Средняя стоимость этих квартир, опираясь на график, составляет около **11** млн. руб.. Реалистично.

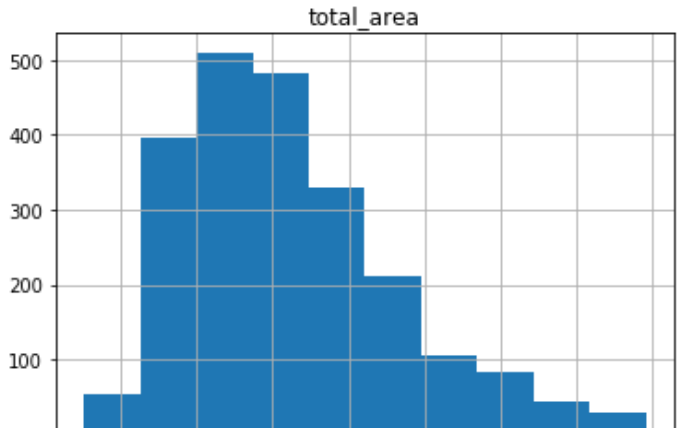
Сравнение выводов по квартирам в центре и общих выводов по всему городу

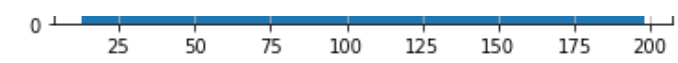
In [49]:

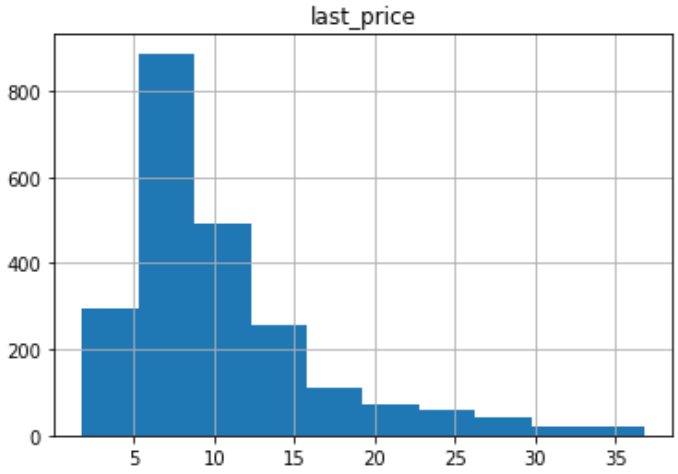
```
spb_center = spb.query('3 <= cityCenters_km <= 7')
```

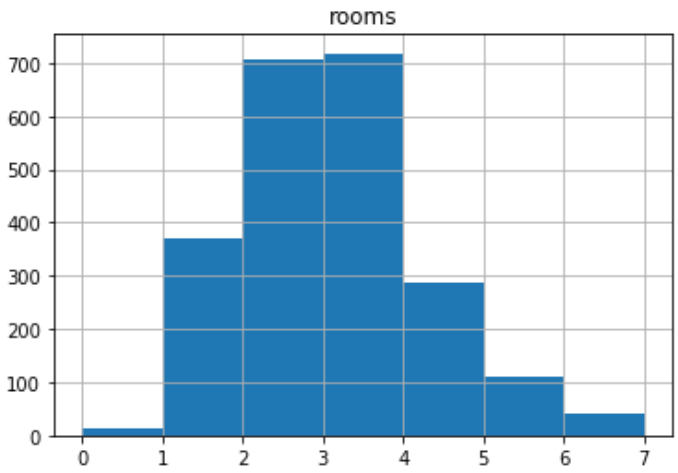
In [50]:

```
spb_center['total_area'].hist()
plt.title('total_area')
plt.show()
spb_center['last_price'].hist()
plt.title('last_price')
plt.show()
spb_center['rooms'].hist(bins=range(0, 8, 1))
plt.title('rooms')
plt.show()
spb_center['ceiling_height'].hist(bins=[1,2,2.5,2.75,3,3.25,3.5,4])
plt.title('ceiling_height')
```



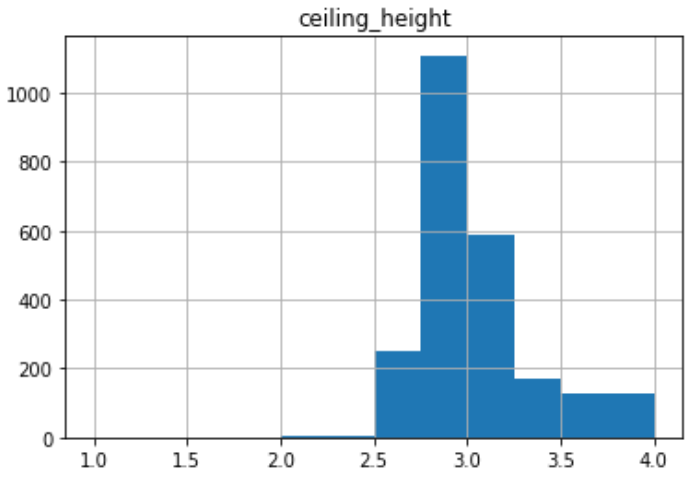






Out[50]:

```
Text(0.5, 1.0, 'ceiling_height')
```



Вывод

1. В центре СПб средняя площадь квартиры от **50** до **80** метров. В остальных районах от **30** до **50** кв. м.
2. В других районах средння стоимость составляет от **3** до **5** млн. В центре СПб от **5** до **9** млн. руб.
3. Среднее кол-во комнат в квартирах в центре СПб от **2** до **4** комнат. В других районах от **1** до **3** комнат.
4. Большинство квартир в центре СПб с потолками высотой от **2.75** м до **3** м. В остальных от **2.5** м до **2.75**.

Можно сказать, что характеристики для квартир в центре СПб выше (как потолки квартир на Невском), чем в остальных районах и поселках.

Общий вывод

Прежде, чем приступить к проекту, я внимательно изучила данные и составила мини-план по предобработке

данных. Опираясь на этот план я сделала:

- обработку пропусков
- замену типов данных
- избавилась от неявлных дубликатов.

Для более объективной оценки стоимости недвижимости был расчитан показатель цены за квадратный метр, по которому в будущем тоже вёлся анализ.

Далее я произвела краткое исследование площади, цены, числа комнат и высоты потолка. Нашла аномально длинные и короткие объявления и сделала вывод: **2014** год с самыми долгими объявлениями, **2019** год с большим числом краткосрочных объявлений.

Выяснила, какие факторы влияют на стоимость квартиры, а какие нет. И укрепила мнение о том, что расстояние от центра и стоимость квартиры отрицательно корреллируют.

Увидела, в каком диапазоне находится центр СПб. И узнала, какие характеристики и стомость у квартир в центре города.

В проекте старалась указывать в комментариях некоторые моменты, чтобы было удобнее понимать. Также в пункте "Исследовательский анализ данных" оставляла промежуточный краткие промежуточные выводы.

Чек-лист готовности проекта

Поставьте '**x**' в выполненных пунктах. Далее нажмите **Shift+Enter**.

- **[x]** открыт файл
- **[x]** файлы изучены (выведены первые строки, метод **info()**)
- **[x]** определены пропущенные значения
- **[x]** заполнены пропущенные значения
- **[x]** есть пояснение, какие пропущенные значения обнаружены
- **[x]** изменены типы данных
- **[x]** есть пояснение, в каких столбцах изменены типы и почему
- **[x]** посчитано и добавлено в таблицу: цена квадратного метра
- **[x]** посчитано и добавлено в таблицу: день недели, месяц и год публикации объявления
- **[x]** посчитано и добавлено в таблицу: этаж квартиры; варианты — первый, последний, другой
- **[x]** посчитано и добавлено в таблицу: соотношение жилой и общей площади, а также отношение площади кухни к общей
- **[x]** изучены следующие параметры: площадь, цена, число комнат, высота потолков
- **[x]** построены гистограммы для каждого параметра
- **[x]** выполнено задание: "Изучите время продажи квартиры. Постройте гистограмму. Посчитайте среднее и медиану. Опишите, сколько обычно занимает продажа. Когда можно считать, что продажи прошли очень быстро, а когда необычно долго?"
- **[x]** выполнено задание: "Уберите редкие и выбивающиеся значения. Опишите, какие особенности обнаружили."
- **[x]** выполнено задание: "Какие факторы больше всего влияют на стоимость квартиры? Изучите, зависит ли цена от квадратного метра, числа комнат, этажа (первого или последнего), удалённости от центра. Также изучите зависимость от даты размещения: дня недели, месяца и года. "Выберите **10** населённых пунктов с наибольшим числом объявлений. Посчитайте среднюю цену квадратного метра в этих населённых пунктах. Выделите населённые пункты с самой высокой и низкой стоимостью жилья. Эти данные можно найти по имени в столбце '***locality_name***'. "
- **[x]** выполнено задание: "Изучите предложения квартир: для каждой квартиры есть информация о расстоянии до центра. Выделите квартиры в Санкт-Петербурге (***'locality_name'***). Ваша задача — выяснить, какая область входит в центр. Создайте столбец с расстоянием до центра в километрах: округлите до целых значений. После этого посчитайте среднюю цену для каждого километра. Постройте график: он должен показывать, как цена зависит от удалённости от центра. Определите границу, где график сильно меняется — это и будет центральная зона. "
- **[x]** выполнено задание: "Выделите сегмент квартир в центре. Проанализируйте эту территорию и изучите следующие параметры: площадь, цена, число комнат, высота потолков. Также выделите факторы, которые влияют на стоимость квартиры (число комнат, этаж, удалённость от центра, дата размещения объявления). Сделайте выводы. Отличаются ли они от общих выводов по всему городу?"

- **[x]** в каждом этапе есть выводы
- **[x]** есть общий вывод