Исследование объявлений о продаже квартир

Данные сервиса **Яндекс.Недвижимость** — архив объявлений о продаже квартир в Санкт-Петербурге и соседних населённых пунктов 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.

План

1. Изучение входной информации

- Описание данных
- Изучение общей информации о предоставляемых данных

2. Предобработка данных

- Определение и заполнение пропущенных значений
- Изменение типов данных и наименований столбцов
- Изучение первичных параметров, удаление редких и выбивающихся значений.

3. Добавление данных в таблицу

- Цена квадратного метра
- День недели, месяц и год публикации объявления
- Этаж квартиры
- Соотношение жилой и общей площади, а также отношение площади кухни к общей
- Описание данных после предобработки и добавления данных

4. Исследовательский анализ данных

- Площадь
- Изучение времени продажи квартиры.
- Какие факторы больше всего влияют на стоимость квартиры?
- Изучение предложений квартир
- Выделиние сегмента квартир в центре и его анализ.

5. Общий вывод

1. Изучение входной информации

Описание данных

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

Описание данных после предобработки и добавления данных

Изучение общей информации о предоставляемых данных

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from random import randint
import os
from pathlib import Path
import urllib
import numpy as np
```

In [2]:
```
path = 'datasets/real_estate_data.csv'
url = 'https://code.s3.yandex.net/datasets/real_estate_data.csv'
Path('datasets').mkdir(parents=True, exist_ok=True)
if not os.path.exists(path):
    print(f'real_estate_data.csv не найден. Будет загружен из сети.')
    _ = urllib.request.urlretrieve(url, path)
base = pd.read_csv('datasets/real_estate_data.csv', sep='\t')
```

In [3]:
```
pd.options.mode.chained_assignment = None
```

In [4]:
```
data = pd.DataFrame(base)
data.info()
```

```
print(data.shape)
```

```
<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
(23699, 22)
```

In [5]:
```
display(data.head(3))
display(data.describe().T)
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_a
0	20	13000000.0	108.0	2019-03-07T00:00:00	3	2.7	16.0	5
1	7	3350000.0	40.4	2018-12-04T00:00:00	1	NaN	11.0	1
2	10	5196000.0	56.0	2015-08-20T00:00:00	2	NaN	5.0	3

3 rows × 22 columns

	count	mean	std	min	25%	50%	75%	
total_images	23699.0	9.858475e+00	5.682529e+00	0.0	6.00	9.00	14.0	
last_price	23699.0	6.541549e+06	1.088701e+07	12190.0	3400000.00	4650000.00	6800000.0	763
total_area	23699.0	6.034865e+01	3.565408e+01	12.0	40.00	52.00	69.9	
rooms	23699.0	2.070636e+00	1.078405e+00	0.0	1.00	2.00	3.0	
ceiling_height	14504.0	2.771499e+00	1.261056e+00	1.0	2.52	2.65	2.8	
floors_total	23613.0	1.067382e+01	6.597173e+00	1.0	5.00	9.00	16.0	
living_area	21796.0	3.445785e+01	2.203045e+01	2.0	18.60	30.00	42.3	

	count	mean	std	min	25%	50%	75%
floor	23699.0	5.892358e+00	4.885249e+00	1.0	2.00	4.00	8.0
kitchen_area	21421.0	1.056981e+01	5.905438e+00	1.3	7.00	9.10	12.0
balcony	12180.0	1.150082e+00	1.071300e+00	0.0	0.00	1.00	2.0
airports_nearest	18157.0	2.879367e+04	1.263088e+04	0.0	18585.00	26726.00	37273.0
cityCenters_nearest	18180.0	1.419128e+04	8.608386e+03	181.0	9238.00	13098.50	16293.0
parks_around3000	18181.0	6.114075e-01	8.020736e-01	0.0	0.00	0.00	1.0
parks_nearest	8079.0	4.908046e+02	3.423180e+02	1.0	288.00	455.00	612.0
ponds_around3000	18181.0	7.702547e-01	9.383456e-01	0.0	0.00	1.00	1.0
ponds_nearest	9110.0	5.179809e+02	2.777206e+02	13.0	294.00	502.00	729.0
days_exposition	20518.0	1.808886e+02	2.197280e+02	1.0	45.00	95.00	232.0

Вывод

Из общей информации о входных данных становится ясно, что имеют место пропуски в данных
Отрицательных значений нет. В датасете присутствиют значения различных типов данных.
Имеют место выбросы в данных

2. Предобработка данных

In [6]:
```
def get_empty_cols(data):
    '''Создание кортежа с столбцами, в которых присутствуют пропущенные значения'''
    cols_had_null = tuple([col for col in data.columns if data[col].isna().sum()])
    return cols_had_null
```

Определение и заполнение пропущенных значений

In [7]:
```
empty_cols = get_empty_cols(data)
print(*empty_cols, sep='\n')
```

```
ceiling_height
floors_total
living_area
is_apartment
kitchen_area
balcony
locality_name
airports_nearest
cityCenters_nearest
parks_around3000
parks_nearest
ponds_around3000
ponds_nearest
days_exposition
```

In [8]:
```
print('locality_name\'s empty values {:.2%}'.format(data.locality_name.isna().sum() / d
data = data[~data['locality_name'].isna()]
```

```
locality_name's empty values 0.21%
```

In [9]:
```
def parse_locality(row):
    i = 0
    while i < len(row):
        if row[i].isupper():
            break
        i += 1
    row = row[i:]
    row.lower()
    row.replace('ë', 'e')
    row.capitalize()
    return row

data.loc[:, 'locality_name'] = data.locality_name.apply(parse_locality)
```

In [10]:
```
data.is_apartment = data.apply(lambda x: True if x['rooms'] > 1 and x['studio'] is Fals
```

In [11]:
```
def insert_by_locality(data, col, fill_by_med=True):
    medians = data.groupby('locality_name')[col].median().dropna()
    for index, value in zip(medians.index, medians):
        data.loc[(data[col].isna()) & (data['locality_name'] == index), col] = value
    print('Has been filled by median value of groupped data', end=' ')
    if data[col].isna().sum() and fill_by_med:
        data.loc[:, col] = data.fillna(data[col].median())
        print('& been filled by median of col')
    else:
        print()
    return data

def apply_insert_by_locality_for_cols(data, cols, fill_all=True):
    for col in cols:
#         print('{: <35} {:.3%}'.format(col + '\'s empty values', data[col].isna().sum(
        data = insert_by_locality(data, col, fill_by_med=fill_all)
#         print('#' * 50)
    return data
```

In [12]:
```
first_part = 'ceiling_height', 'floors_total', 'living_area', 'kitchen_area', 'balcony'
data = apply_insert_by_locality_for_cols(data, first_part)

second_part = 'airports_nearest', 'cityCenters_nearest', 'parks_around3000', 'parks_nea
                    'ponds_around3000', 'ponds_nearest'
data = apply_insert_by_locality_for_cols(data, second_part, fill_all=False)

print(('#' * 50 + '\n') * 2)
if second_part == get_empty_cols(data):
    print('Succesfully filled first part empty cols, the other part need to be parse fo
```

```
Has been filled by median value of groupped data & been filled by median of col
Has been filled by median value of groupped data
Has been filled by median value of groupped data & been filled by median of col
Has been filled by median value of groupped data & been filled by median of col
Has been filled by median value of groupped data & been filled by median of col
Has been filled by median value of groupped data & been filled by median of col
Has been filled by median value of groupped data
Has been filled by median value of groupped data
Has been filled by median value of groupped data
Has been filled by median value of groupped data
Has been filled by median value of groupped data
Has been filled by median value of groupped data
##################################################
```

```
##################################################

Succesfully filled first part empty cols, the other part need to be parse for filling
```

In [13]:
```
def parse_two_cols(data, f, s):
    '''Нахождение данных в столбце s для столбца f DataFrame data
    Заполнение спец значением -1 и перевод к т д int
    '''
    def appl(row):
        if row[s] > 0 and f != 'cityCenters_nearest':
            return 3000
        else:
            return row[f]

    is_check = data[f].isna().sum() == ((data[f].isna()) & (data[s].isna())).sum()
    data.loc[:, s] = data[s].fillna(-1.0).astype(int)
    if is_check:
        print(f'There\'s no reason to search info in {s} for {f}')
    else:
        print(f'Search info in {s} for {f}')
        data.loc[:, f] = data.apply(appl, axis=1)
    data.loc[:, f] = data[f].fillna(-1.0)
    data.loc[:, f] = data[f].astype(int)
    if s != 'airports_nearest':
        data = data.drop(s, axis=1)
    return data

cols_parse = ('parks_around3000', 'parks_nearest'), ('ponds_around3000', 'ponds_nearest
              ('airports_nearest', 'cityCenters_nearest')
cols_parse = tuple([item[::-1] for item in cols_parse])

for cols in cols_parse:
    data = parse_two_cols(data, *cols)
```

```
Search info in parks_around3000 for parks_nearest
Search info in ponds_around3000 for ponds_nearest
There's no reason to search info in airports_nearest for cityCenters_nearest
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23650 entries, 0 to 23698
Data columns (total 20 columns):
total_images            23650 non-null int64
last_price              23650 non-null float64
total_area              23650 non-null float64
first_day_exposition    23650 non-null object
rooms                   23650 non-null int64
ceiling_height          23650 non-null float64
floors_total            23650 non-null float64
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 float64
locality_name           23650 non-null object
airports_nearest        23650 non-null int64
cityCenters_nearest     23650 non-null int64
parks_nearest           23650 non-null int64
ponds_nearest           23650 non-null int64
days_exposition         23650 non-null float64
```

```
dtypes: bool(3), float64(8), int64(7), object(2)
memory usage: 3.3+ MB
```

In [15]:
```
mem_before = 3.0
```

In [16]:
```
# help(np.uint16)
# help(np.uint8)
```

uint16 -> (0 to 65535) uint8 -> (0 to 255)

In [17]:
```
cols_parse = 'airports_nearest', 'cityCenters_nearest',\
                    'parks_nearest', 'ponds_nearest'
data.loc[:, cols_parse].describe().T
```

Out[17]:

	count	mean	std	min	25%	50%	75%	max
airports_nearest	23650.0	23512.449852	16667.539974	-1.0	11942.00	23140.0	35841.0	84869.0
cityCenters_nearest	23650.0	11511.521268	9633.272673	-1.0	3870.25	11753.0	15743.0	65968.0
parks_nearest	23650.0	1210.685666	1300.745383	-1.0	-1.00	460.0	3000.0	3190.0
ponds_nearest	23650.0	1425.453362	1330.468317	-1.0	503.00	503.0	3000.0	3000.0

In [18]:
```
for col in ['parks_nearest', 'ponds_nearest']:
    data.loc[:, col] = data.loc[:, col].astype(np.uint16)
```

In [19]:
```
print(len(get_empty_cols(data)))
```

```
0
```

Вывод

Были обработаны отсутствия значений в столбцах `locality_name` , `apartmnent` , `ceiling_height` , `floors_total` , `living_area` , `kitchen_area` , `balcony` , `days_exposition` , `airports_nearest` , `cityCenters_nearest` , `parks_around3000` , `parks_nearest` , `ponds_around3000` , `ponds_nearest` . (с помощью группировки по локации удалось заполнить пропуски в данных,
но не полностью- оставшиеся пропуски были заполнены медианами по соответствующим столбцам)
Также были обработаны значения столбца `locality_name` (выделены наименования)

Изменение типов данных и наименований столбцов

In [20]:
```
data.loc[:, 'floors_total'] = data.floors_total.astype(np.uint8)
data.loc[:, 'balcony'] = data.balcony.astype(np.uint8)
data.loc[:, 'days_exposition'] = data.days_exposition.astype(np.uint32)
data.loc[:, 'last_price'] = data.last_price.astype(np.uint32)
```

In [21]:
```
data.columns = ['total_images', 'price', 'total_area', 'first_day_exp',
       'rooms', 'ceiling_height', 'floors_total', 'living_area', 'floor',
       'apartment', 'studio', 'open_plan', 'kitchen_area', 'balcony',
       'location', 'air_nearest', 'city_nearest',
       'parks_nearest', 'ponds_nearest', 'days_exp']
```

Вывод

Были изменены типы данных столбцов с тд float на int (для удобства анализа и дальнейшей обработки)
`floors_total` , `balcony` , `days_exposition` , `last_price` .

Изучение первичных параметров, удаление редких и выбивающихся значений.

In [22]:
```
def describe_enhanced(data, list_cols):
    '''Поиск значений low_iqr и up_iqr для солбцов в list_cols DataFrame data
    и занесений значений в DataFrame метода .describe()'''
    def get_lowest_uppest(col):
        '''Получение нижнего и верхнего "усов" данных - то,
        что будет добавлено в DataFrame метода .describe()'''
        col_info = dict(col.describe())
        iqr = col_info['75%'] - col_info['25%']
        lowest = col_info['25%'] - 1.5 * iqr
        lowest = lowest if lowest >= 0 else 0
        uppest = col_info['75%'] + 1.5 * iqr
        return lowest, uppest

    descr = pd.DataFrame(data[list_cols].describe())
    lowers, uppers = [], []
    for item in list_cols:
        temp = get_lowest_uppest(data[item])
        lowers.append(temp[0])
        uppers.append(temp[1])
    to_add = pd.DataFrame([lowers, uppers], index=(('low_iqr', 'up_iqr')))
    to_add.columns = list_cols
    descr = descr.append(to_add)
    return descr

def del_anomal_values(data, info_descr, list_cols):
    for col in list_cols:
        low, up = info_descr[col]['low_iqr'], info_descr[col]['up_iqr']
        data = data[(low < data[col]) & (data[col] < up)]
    return data
```

Площадь недвижимости

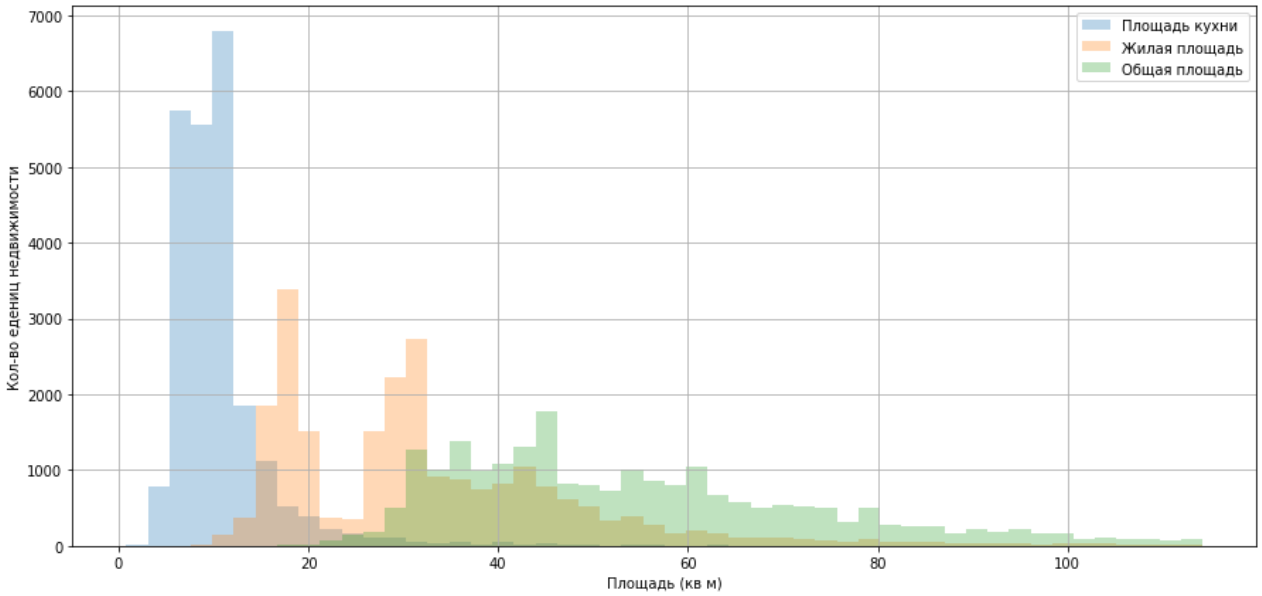
В Санкт-Петербурге УН жилой площади также 9 кв. метров на человека в отдельных домах и квартирах, 15 кв. метров — для коммуналок.

In [23]:
```
areas_list = ['kitchen_area', 'living_area', 'total_area']
areas_info = describe_enhanced(data, areas_list)
display(areas_info)
```

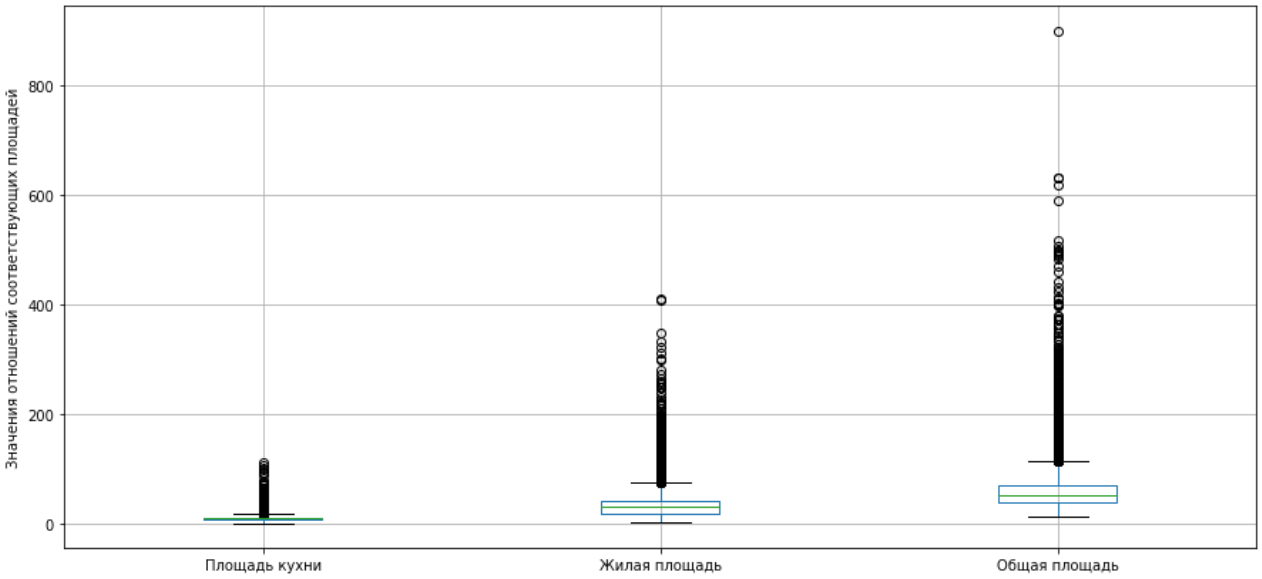
	kitchen_area	living_area	total_area
count	23650.000000	23650.000000	23650.000000
mean	10.465152	34.054007	60.329069
std	5.631919	21.226308	35.661808
min	1.300000	2.000000	12.000000
25%	7.200000	19.000000	40.000000

	kitchen_area	living_area	total_area
50%	9.600000	30.400000	52.000000
75%	11.457500	41.100000	69.700000
max	112.000000	409.700000	900.000000
low_iqr	0.813750	0.000000	0.000000
up_iqr	17.843750	74.250000	114.250000

In [24]:
```
to_plot = ['kitchen_area', 'living_area', 'total_area'], ['Площадь кухни', 'Жилая площа
data.plot(kind='hist', y=to_plot[0], label=to_plot[1], bins=50, alpha=0.3, grid=True, l
          range=(areas_info['kitchen_area']['low_iqr'], areas_info['total_area']['up_iq
plt.xlabel('Площадь (кв м)')
plt.ylabel('Кол-во едениц недвижимости')
plt.show()
```



In [25]:
```
data.plot(kind='box', y=to_plot[0], label=to_plot[1], grid=True, legend=True, figsize=(
plt.ylabel('Значения отношений соответствующих площадей')
plt.show()
```



In [26]:
```
data = del_anomal_values(data, areas_info, to_plot[0])
```

Вывод

Были проанализированы площадь кухни, жилая и общая площади. Было выявлено, что имеют место аномалии (выбросы)
в данных. С помощью границы `(Q1 - 1.5 * IQR < values < Q3 + 1.5 * IQR)` были отброшены выбивающиеся значения из данных

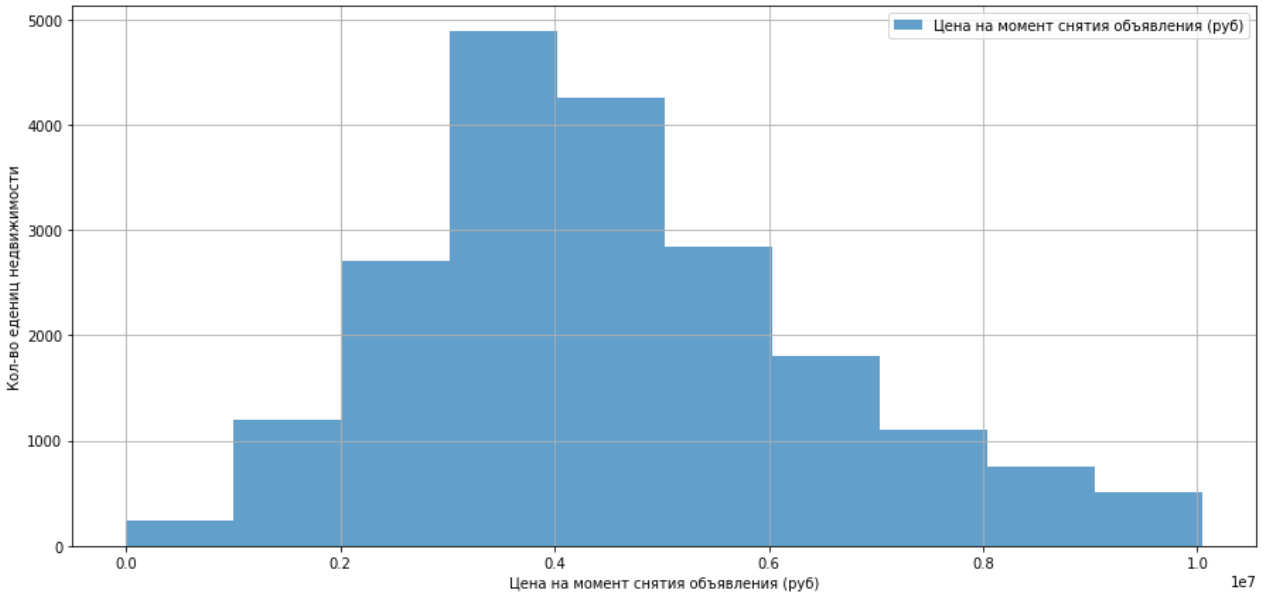
Цена недвижимости

In [27]:
```
price_info = describe_enhanced(data, ['price'])
display(price_info)
```

	price
count	2.140900e+04
mean	5.048886e+06
std	2.933212e+06
min	4.300000e+05
25%	3.300000e+06
50%	4.400000e+06
75%	6.000000e+06
max	5.300000e+07
low_iqr	0.000000e+00
up_iqr	1.005000e+07

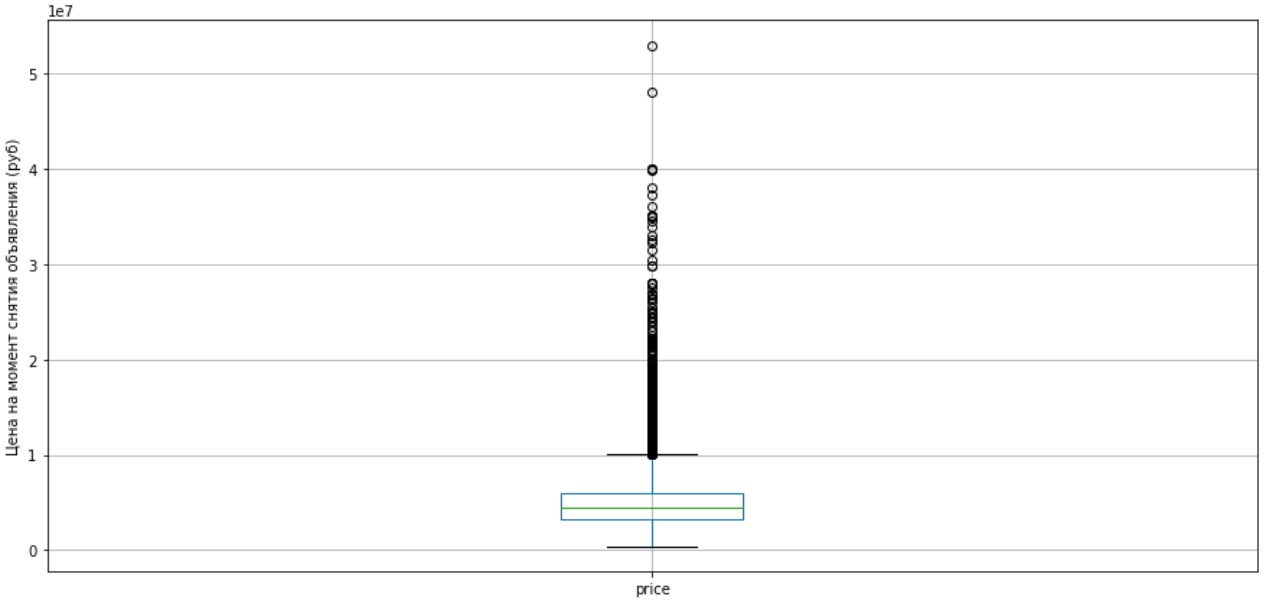
In [28]:
```
to_plot = 'price', 'Цена на момент снятия объявления (руб)'
data.plot(kind='hist', y=to_plot[0], bins=10, label=to_plot[1], alpha=0.7, grid=True, l
              range=(price_info[to_plot[0]]['low_iqr'], price_info[to_plot[0]]['up_iqr'
plt.xlabel(to_plot[1])
```

```
plt.ylabel('Кол-во едениц недвижимости')
plt.show()
```



In [29]:
```
data.plot(kind='box', y=to_plot[0], grid=True, legend=True, figsize=(15, 7))
plt.ylabel(to_plot[1])
plt.plot()
```

Out[29]: []



In [30]:
```
data = del_anomal_values(data, price_info, ['price'])
```

In [31]:
```
data.loc[:, 'price_m'] = data.price.apply(lambda x: x / 1_000_000).astype(np.uint8)
```

Вывод

Была проанализирована цена на момент снятия объявления (руб), получилось выявить, что имеют место аномалии (выбросы)
в данных. С помощью границы `(Q1 - 1.5 * IQR < values < Q3 + 1.5 * IQR)` были

отброшены выбивающиеся значения из данных

Также добавили столбец со значениями цены недвижимости в млн. руб.

Количество комнат

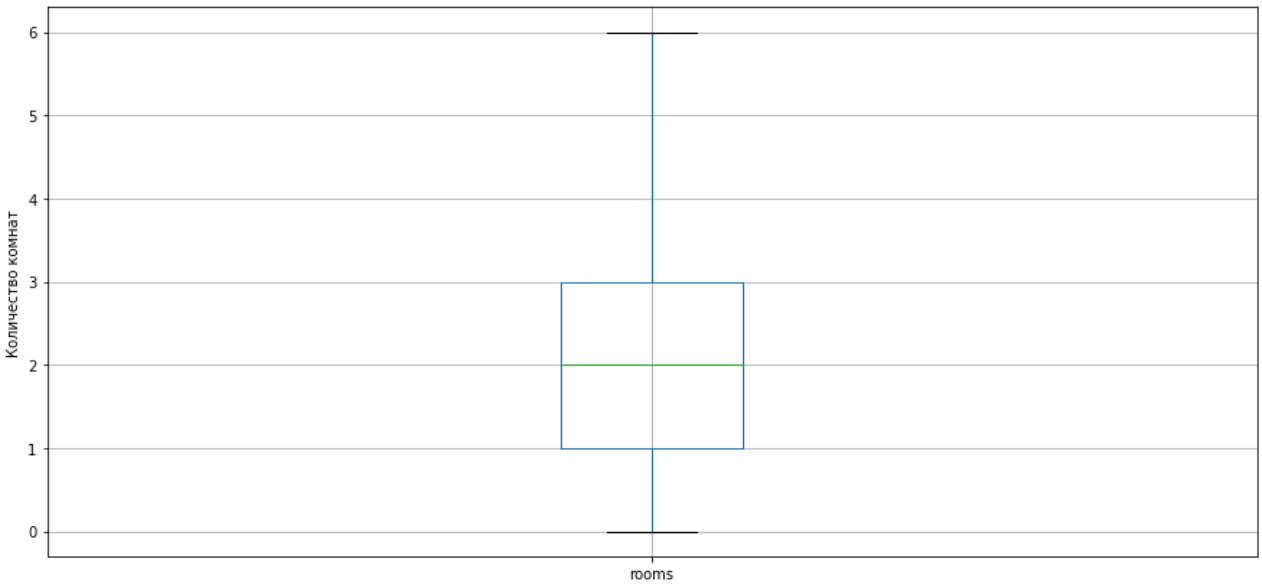
In [32]:
```
rooms_info = describe_enhanced(data, ['rooms'])
rooms_info
```

Out[32]:

	rooms
count	20286.000000
mean	1.882037
std	0.877532
min	0.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	6.000000
low_iqr	0.000000
up_iqr	6.000000

In [33]:
```
to_plot = 'rooms', 'Количество комнат'
data.plot(kind='box', y=to_plot[0], grid=True, legend=True, figsize=(15, 7))
plt.ylabel(to_plot[1])
plt.plot()
```

Out[33]: []



Вывод

Был проанализирован параметр - количество комнат, получилось выявить, что аномалии (выбросы) отсутствуют

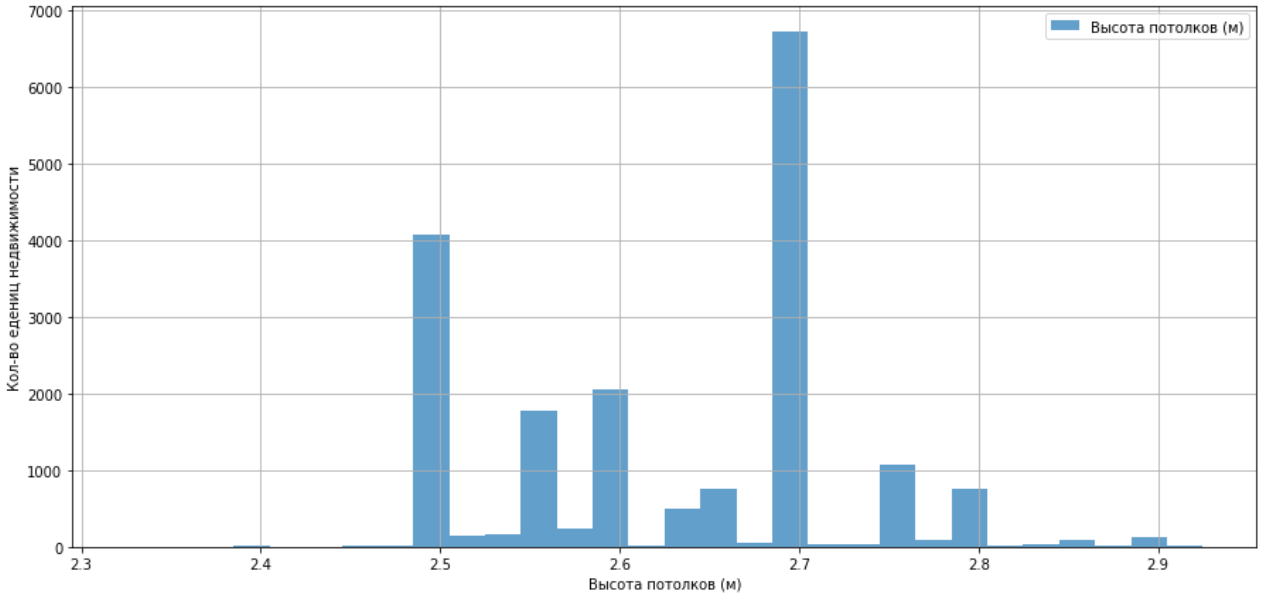
Высота потолков

In [34]:
```
ceil_h_info = describe_enhanced(data, ['ceiling_height'])
ceil_h_info
```

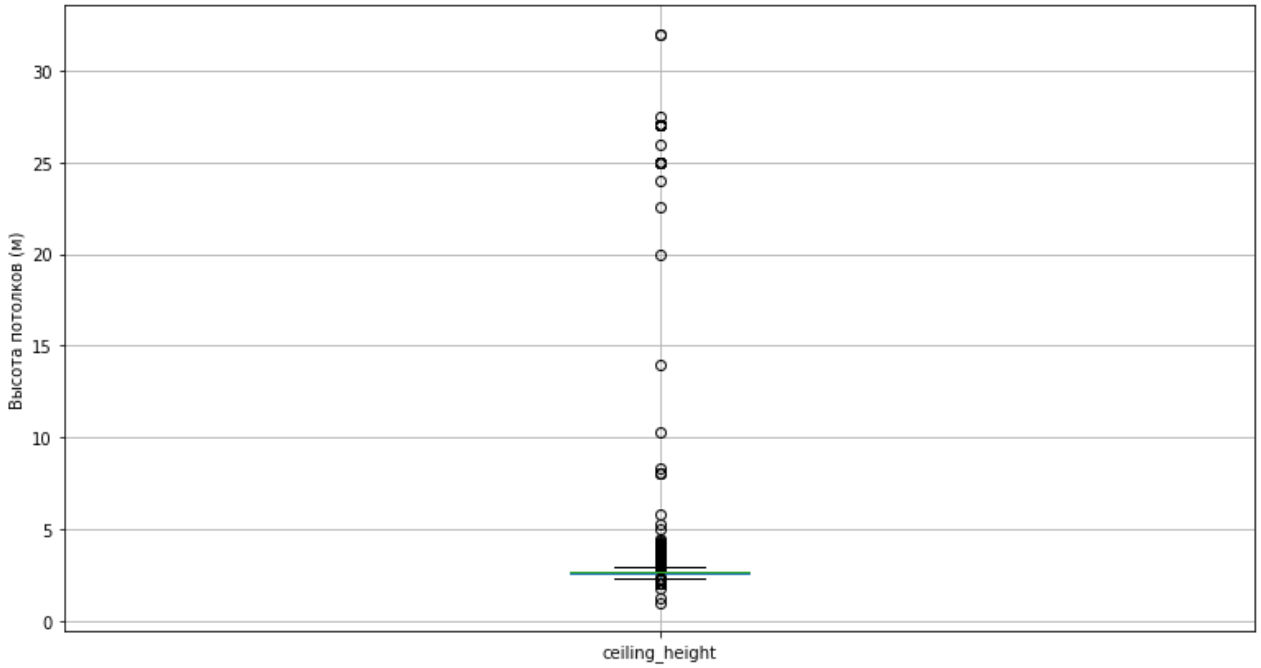
Out[34]:

	ceiling_height
count	20286.000000
mean	2.697232
std	0.824417
min	1.000000
25%	2.550000
50%	2.700000
75%	2.700000
max	32.000000
low_iqr	2.325000
up_iqr	2.925000

In [35]:
```
to_plot = 'ceiling_height', 'Высота потолков (м)'
data.plot(kind='hist', y=to_plot[0], label=to_plot[1],\
          range=(ceil_h_info[to_plot[0]]['low_iqr'], ceil_h_info[to_plot[0]]['up_iqr']),
          bins=30, alpha=0.7, grid=True, legend=True, figsize=(15, 7))
plt.xlabel(to_plot[1])
plt.ylabel('Кол-во едениц недвижимости')
plt.show()
```



In [36]:
```
data.plot(kind='box', y=to_plot[0], grid=True, legend=True, figsize=(13, 7))
plt.ylabel(to_plot[1])
plt.show()
```



In [37]:
```
data = del_anomal_values(data, ceil_h_info, [to_plot[0]])
```

Вывод

Была проанализирована высота потолков (м), получилось выявить, что имеют место аномалии (выбросы)
в данных. С помощью границы `(Q1 - 1.5 * IQR < values < Q3 + 1.5 * IQR)` были отброшены выбивающиеся значения из данных

3. Добавление данных в таблицу

Цена квадратного метра

In [38]:
```
data.loc[:, 'price_sq_m'] = (data.loc[:, 'price'] / data.loc[:, 'total_area'])
```

День недели, месяц и год публикации объявления

In [39]:
```
data.loc[:, 'date_exp'] = pd.to_datetime(data['first_day_exp'])
data.loc[:, 'day_exp'] = data['date_exp'].dt.day
data.loc[:, 'month_exp'] = data['date_exp'].dt.month
data.loc[:, 'year_exp'] = data['date_exp'].dt.year
data = data.drop('first_day_exp', axis=1)
```

Этаж квартиры

In [40]:
```
def apl_floor(row):
    if row.floor == row.floors_total:
        return 'Последний'
    if row.floor == 1:
        return 'Первый'
    return 'Другой'

data.loc[:, 'floor'] = data.apply(apl_floor, axis=1)
data = data.drop('floors_total', axis=1)
```

Соотношение жилой и общей площади и отношение площади кухни к общей

In [41]:
```
data.loc[:, 'living_total'] = round(data.living_area / data.total_area, 2)
data.loc[:, 'kitchen_total'] = round(data.kitchen_area / data.total_area, 2)
```

In [42]:
```
data.loc[:, 'city_nearest_km'] = (data.city_nearest // 1000).astype(np.uint8)
```

Вывод

Были добавлены/изменены след данные: `price_sq_m` , `date_exp` , `day_exp` , `month_exp` , `year_exp` ,
`floor` , `living_total` , `kitchen_total` , `city_nearest_km`

Описание данных после предобработки и добавления данных

- **balcony** — число балконов
- **ceiling_height** — высота потолков (м)
- **city_nearest** — расстояние до центра города (м)
- **days_exp** — сколько дней было размещено объявление (от публикации до снятия)
- **apartment** — апартаменты (булев тип)
- **kitchen_area** — площадь кухни в квадратных метрах ($м^2$)
- **price** — цена на момент снятия с публикации (руб)
- **living_area** — жилая площадь в квадратных метрах($м^2$)
- **localation** — название населённого пункта
- **open_plan** — свободная планировка (булев тип)
- **parks_nearest** — расстояние до ближайшего парка (м)
- **ponds_nearest** — расстояние до ближайшего водоёма (м)
- **rooms** — число комнат
- **studio** — квартира-студия (булев тип)
- **total_area** — площадь квартиры в квадратных метрах ($м^2$)
- **total_images** — число фотографий квартиры в объявлении
- **price_sq_m** цена квадратного метра
- **date_exp** дата создания объявления
- **day_exp** день создания объявления
- **month_exp** месяц создания объявления
- **year_exp** год создания объявления
- **floor** этаж (теперь признак категориальный)
- **living_total** отношение жилой к общей площади
- **kitchen_total** отношение площади кухни к общей
- **price_m** — цена на момент снятия с публикации (млн. руб)
- **city_nearest_km** — расстояние до центра города (м)

In [43]:
```
for col in ['total_images', 'rooms', 'ceiling_height', 'balcony', 'days_exp',\
            'day_exp', 'month_exp', 'year_exp', 'city_nearest_km']:
    data.loc[:, col] =  data.loc[:, col].astype(np.uint16)
```

In [44]:
```
mem_after = 1.8
```

```
print('За счёт грамотного хранеиия информации (выбор uint16 & uint8) удалось достичь{:
      format(mem_after / mem_before))
```

```
За счёт грамотного хранеиия информации (выбор uint16 & uint8) удалось достичь 60% эконо
мии памяти
```

4. Исследовательский анализ данных

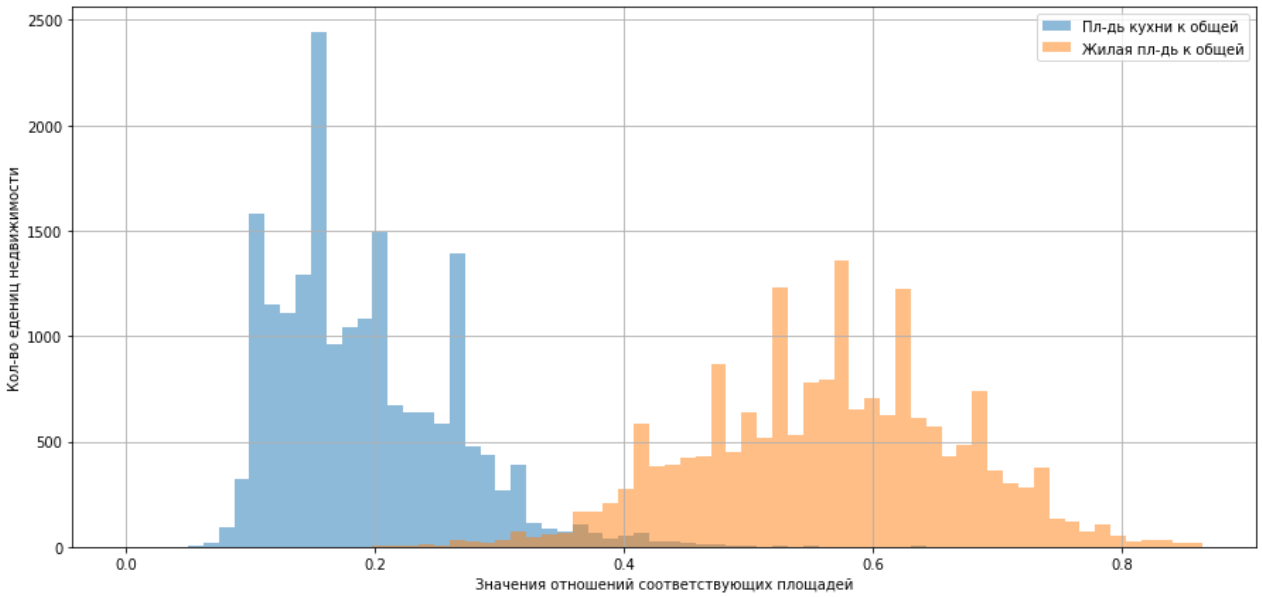
Площадь

In [45]:
```
areas_info = describe_enhanced(data, ['kitchen_total', 'living_total'])
areas_info
```

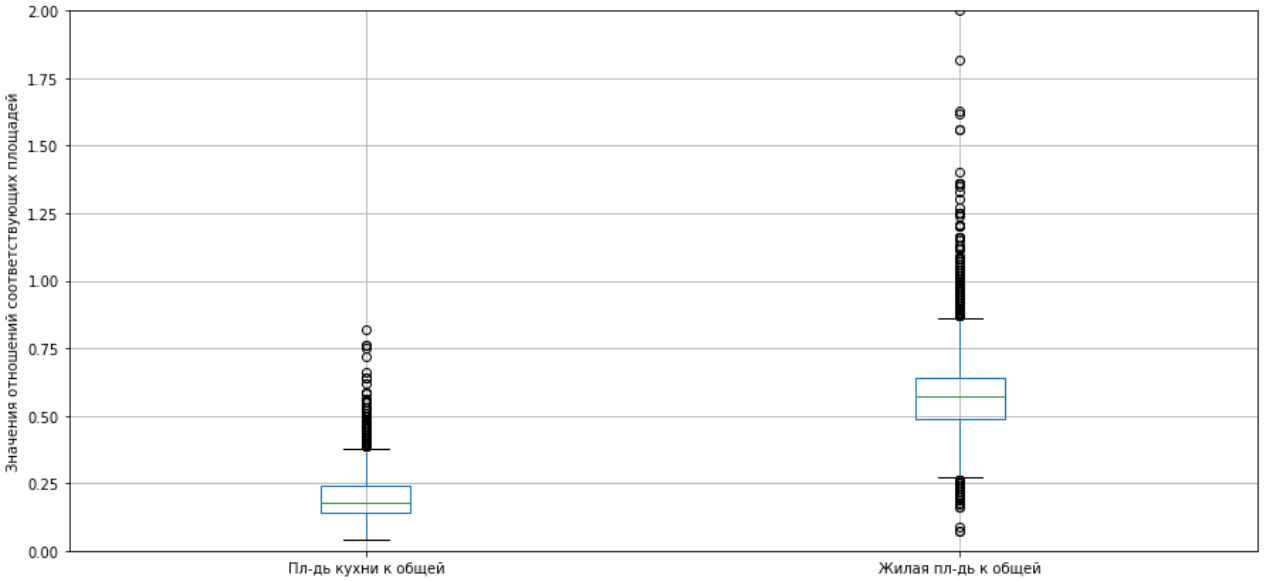
Out[45]:

	kitchen_total	living_total
count	18820.000000	18820.000000
mean	0.192171	0.570481
std	0.070575	0.119018
min	0.040000	0.070000
25%	0.140000	0.490000
50%	0.180000	0.570000
75%	0.240000	0.640000
max	0.820000	2.410000
low_iqr	0.000000	0.265000
up_iqr	0.390000	0.865000

In [46]:
```
to_plot = ['kitchen_total', 'living_total'], ['Пл-дь кухни к общей', 'Жилая пл-дь к общ
range_plot = (areas_info[to_plot[0][0]]['low_iqr'], areas_info[to_plot[0][1]]['up_iqr']
data.plot(kind='hist', y=to_plot[0], bins=70, alpha=0.5, grid=True, legend=True, figsiz
          range=range_plot, label=to_plot[1])
plt.xlabel('Значения отношений соответствующих площадей')
plt.ylabel('Кол-во еденицц недвижимости')
plt.show()
```



In [47]:
```
data.plot(kind='box', y=to_plot[0], \
                label=['Пл-дь кухни к общей', 'Жилая пл-дь к общей'], grid=True, legend=
plt.ylabel('Значения отношений соответствующих площадей')
plt.ylim(0, 2)
plt.show()
```



In [48]:
```
data = del_anomal_values(data, areas_info, to_plot[0])
```

Вывод

Были проанализированы данные столбцов отношений площадей кухни к общей и жилой к общей площади соотв-но.
Было выявлено, что имеют место аномалии (выбросы) в данных. С помощью границы `(Q1 - 1.5 * IQR < values < Q3 + 1.5 * IQR)` были отброшены выбивающиеся значения из данных

Изучение времени продажи квартиры.

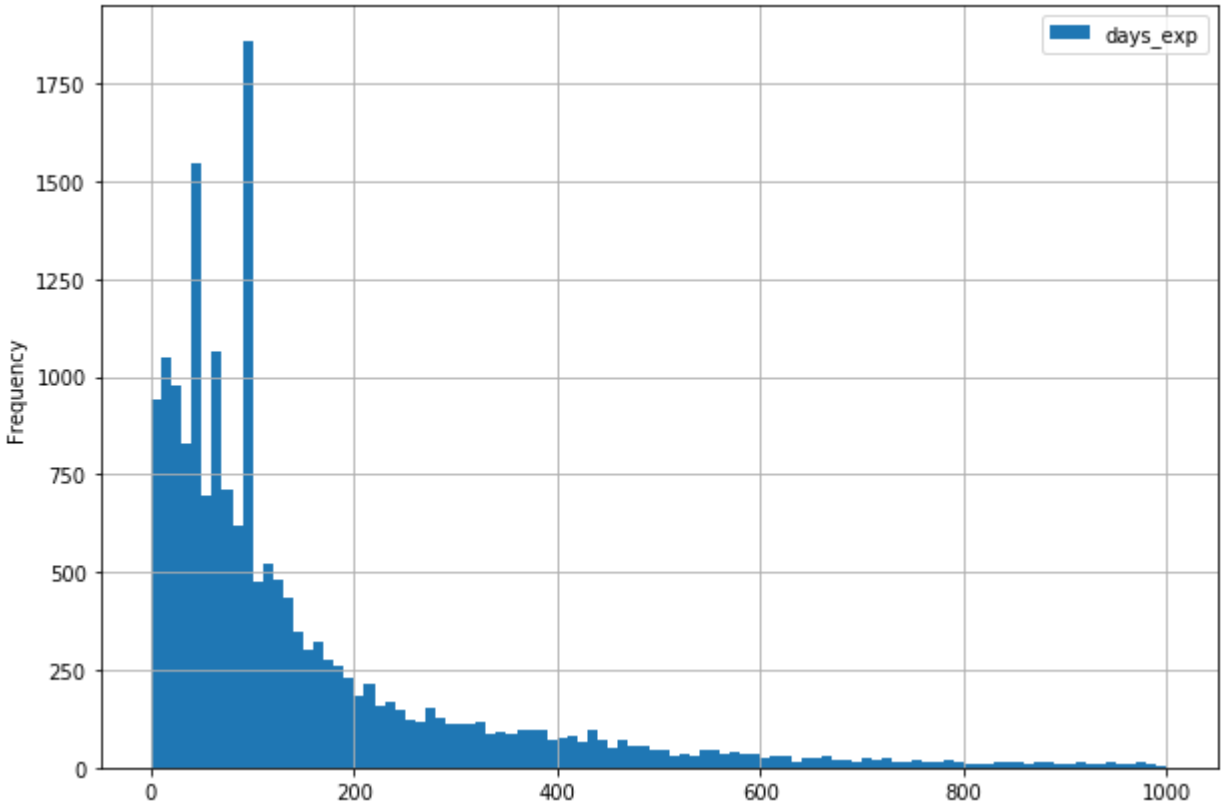
In [49]:
```
times_info = describe_enhanced(data, ['days_exp'])
times_info
```

Out[49]:

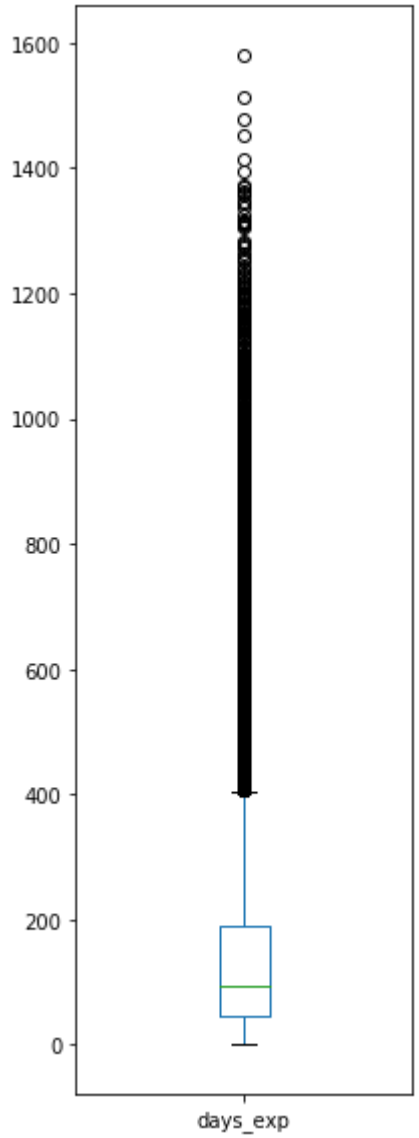
	days_exp
count	18242.000000
mean	160.267953
std	194.779686
min	1.000000
25%	45.000000
50%	95.000000
75%	189.000000
max	1580.000000

	days_exp
low_iqr	0.000000
up_iqr	405.000000

In [50]:
```
data.plot(kind='hist', y='days_exp', bins=100, range=(0, 1000), label='days_exp',\
          legend=True, grid=True, figsize=(10, 7))
plt.show()
```



In [51]:
```
data.plot(kind='box', y='days_exp', figsize=(3, 10))
plt.show()
```



In [52]:
```
data = del_anomal_values(data, times_info, ['days_exp'])
```

In [53]:
```
times_info = describe_enhanced(data, ['days_exp'])
mn_exp, mdn_exp = map(int, (times_info.loc['mean', 'days_exp'], times_info.loc['50%', '
print(f'Days_exposition:\nMean\t{mn_exp}\nMedian\t{mdn_exp}')
```

```
Days_exposition:
Mean    107
Median  86
```

Вывод

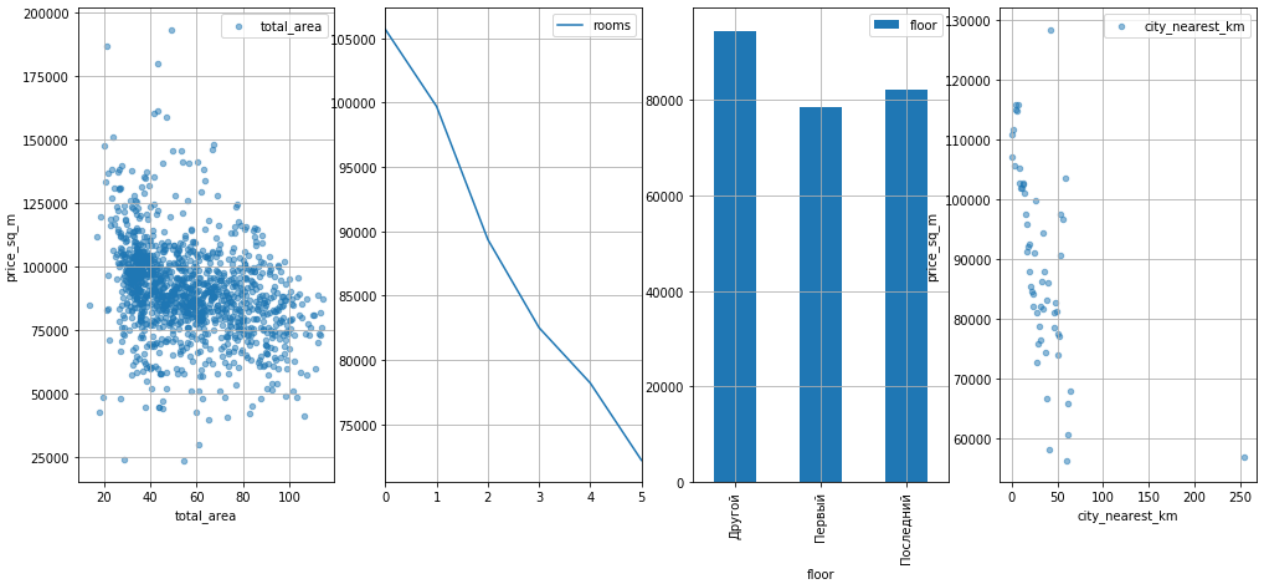
Были проанализированы данные столбца `days_exp` - количество дней доступа к объявлению о недвижимости.
Было выявлено, что имеют место аномалии (выбросы) в данных. С помощью границы `(Q1 - 1.5 * IQR < values < Q3 + 1.5 * IQR)` были отброшены выбивающиеся значения из данных

Какие факторы больше всего влияют на стоимость квартиры?

In [54]:
```
col_parse = 'price_sq_m'
```

```
cols_parse = ['total_area', 'rooms', 'floor', 'city_nearest_km']

fig, axises = plt.subplots(1, len(cols_parse))

for col, axis in zip(cols_parse, axises):
    if col != 'floor':
        correlation = round(data[col_parse].corr(data[col]), 2)
        print('{} ~ {: ^13} ->{: ^10}'.format(col_parse, col, correlation))
    if col == 'floor':
        (data
            .groupby(col)[col_parse].median().reset_index()
            .plot(kind='bar', x='floor', y=col_parse, label=col, legend=True, grid=True
        )
    elif col == 'rooms':
        (data
            .groupby(col)[col_parse].median().reset_index(drop=True)
            .plot(label=col, legend=True, grid=True, figsize=(17, 7), ax=axis)
        )
    else:
        (data
            .groupby(col)[col_parse].median().reset_index()
            .plot(kind='scatter', x=col, y=col_parse, label=col,\
                  legend=True, grid=True, alpha=0.5, figsize=(17, 7), ax=axis)
        )

        plt.xlabel(col)
plt.show()
```

```
price_sq_m ~  total_area   ->  -0.15
price_sq_m ~     rooms     ->  -0.27
price_sq_m ~ city_nearest_km ->  -0.65
```

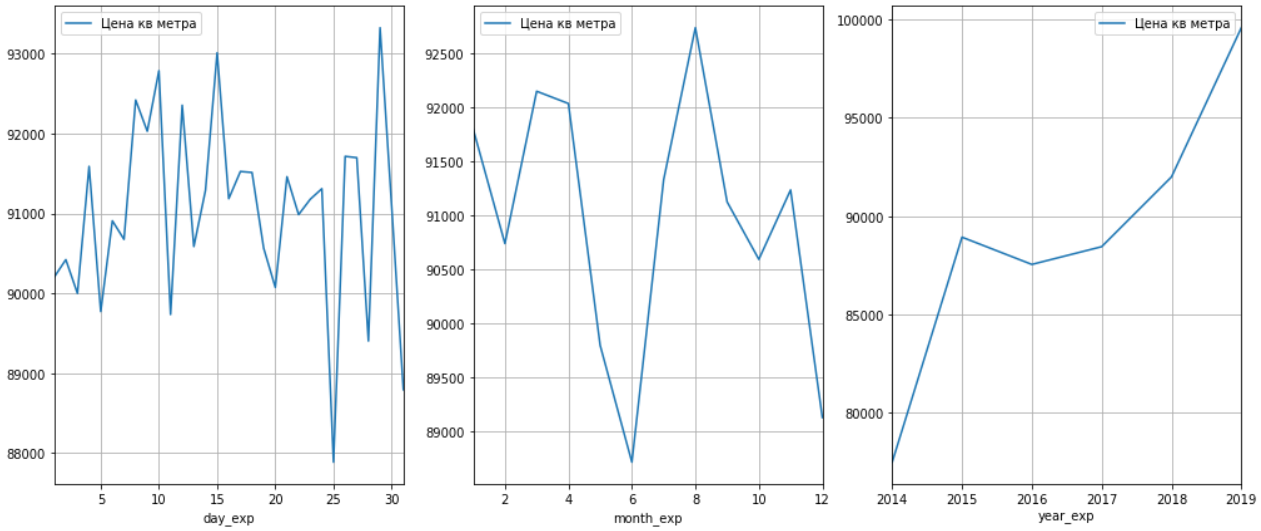


In [55]:

```
col_parse = 'price_sq_m'
cols_parse = ['day_exp', 'month_exp', 'year_exp']

fig, axises = plt.subplots(1, len(cols_parse))

for col, axis in zip(cols_parse, axises):
    correlation = round(data[col_parse].corr(data[col]), 2)
    print('{} ~ {: ^13} ->{: ^10}'.format(col_parse, col, correlation))

    (data
        .groupby(col)[col_parse].median()
```

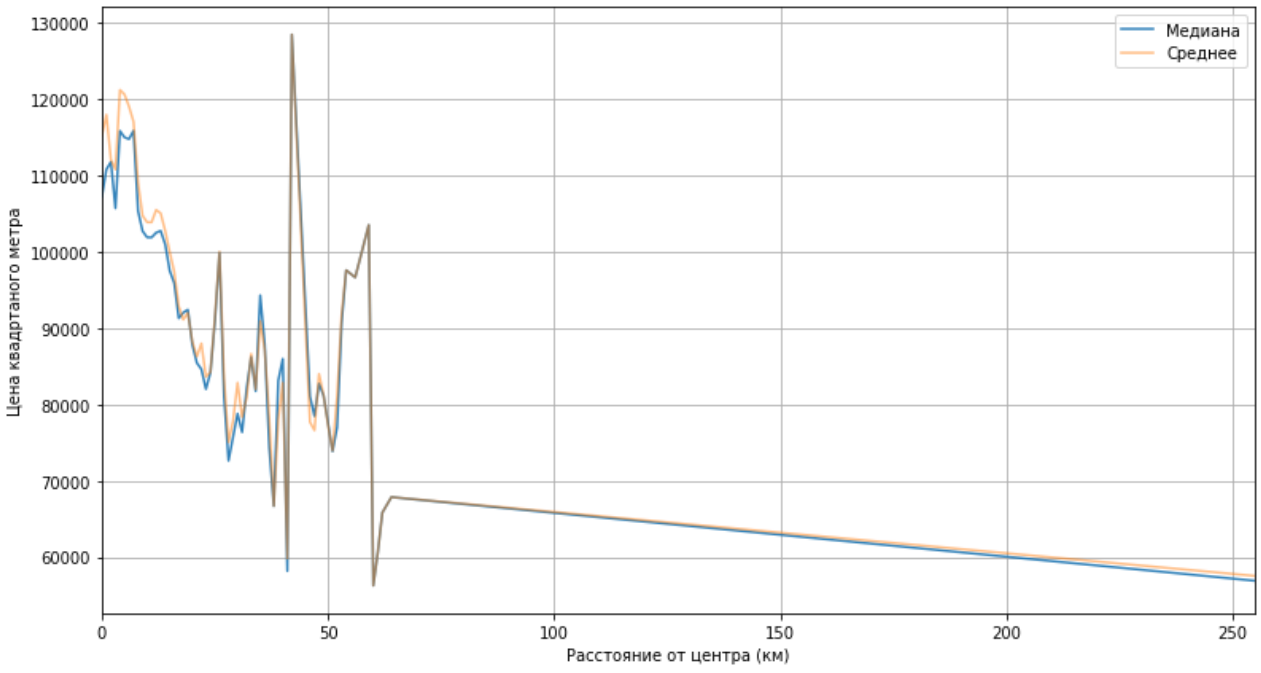
```
        .plot(label='Цена кв метра', legend=True, grid=True, figsize=(17, 7), ax=axis)
    )
plt.show()
```

```
price_sq_m ~    day_exp    ->   0.0
price_sq_m ~   month_exp   ->  -0.01
price_sq_m ~   year_exp    ->   0.08
```



In [56]:

```
ax = (data
    .groupby('city_nearest_km')['price_sq_m'].median().reset_index()
    .plot(x='city_nearest_km', y='price_sq_m', figsize=(13, 7), legend=True, grid=True,
)

(data
    .groupby('city_nearest_km')['price_sq_m'].mean().reset_index()
    .plot(x='city_nearest_km', y='price_sq_m', figsize=(13, 7), legend=True, grid=True,
)

plt.xlabel('Расстояние от центра (км)')
plt.ylabel('Цена квадртаного метра')
plt.show()
```



Вывод

Больше всего на стоимость влияют след параметры: удалённость от города (км) и количество комнат
До ~70км зависимость не является линейной - это можно связать с множеством районов в пределах города, которые отличаются
по уровню и качеству застройки, после граничного значения км жилья +- одинаковое и видная прямая завимисимость между
удалённостью и ценой квадратного метра (обратная зависимость)
Меньшую корреляцию с ценой за кв метр имеет этаж (на первом этаже меньшее значение цены площади кв метра)
Наибольшую цену кв метра, имеют объявления, созданные в 27-28 числах августа. Также наблюдается тенденция в росте
цены кв метра с 2016 по 2019 год.

In [57]:
```
data_top_10_count = (data
    .groupby('location').agg({'rooms':'count', 'price_sq_m':'mean'}).reset_index()
    .sort_values(by='rooms', ascending=False).reset_index(drop=True)
)
data_top_10_count.loc[:, 'price_sq_m'] = data_top_10_count['price_sq_m'].astype(int)
data_top_10_count.columns = ['location', 'count', 'price_sq_m']

print('{:#^70}'.format('Первые 10 нас. пунктов по количеству объявлений'))
display(data_top_10_count.head(10))
```

```
###########Первые 10 нас. пунктов по количеству объявлений#############
```

	location	count	price_sq_m
0	Санкт-Петербург	10142	103506
1	Мурино	459	85595
2	Шушары	377	77829
3	Кудрово	343	95343
4	Всеволожск	318	66557
5	Колпино	284	75273
6	Парголово	270	90435
7	Пушкин	259	99244
8	Гатчина	245	68509
9	Выборг	173	57212

In [58]:
```
data_top_price = (data
#                     .groupby('location')['price_sq_m'].mean().reset_index()
            .groupby('location').agg({'rooms':'count', 'price_sq_m':'mean'}).reset_inde
            .sort_values(by='price_sq_m', ascending=False)
)
data_top_price.loc[:, 'price_sq_m'] = data_top_price['price_sq_m'].astype(int)
data_top_price.columns = ['location', 'count', 'price_sq_m']

display('{:#^70}'.format('Топ по цене (первые 10)'))
```

```
display(data_top_price.head(10).reset_index(drop=True))
display('{:#^70}'.format('Топ по цене (последние 10)'))
display(data_top_price.tail(10).reset_index(drop=True))
```

'########################Топ по цене (первые 10)########################'

	location	count	price_sq_m
0	Лисий Нос	2	113728
1	Санкт-Петербург	10142	103506
2	Сестрорецк	110	101834
3	Зеленогорск	20	100123
4	Пушкин	259	99244
5	Мистолово	8	97145
6	Левашово	1	96997
7	Кудрово	343	95343
8	Парголово	270	90435
9	Стрельна	35	88627

'########################Топ по цене (последние 10)########################'

	location	count	price_sq_m
0	Ефимовский	2	14149
1	Ям-Тесово	2	13711
2	Сижно	1	13709
3	Тёсово-4	1	12931
4	Малая Романовка	1	12724
5	Совхозный	2	12629
6	Выскатка	2	12335
7	Вахнова Кара	1	11688
8	Свирь	2	11481
9	Старополье	3	11206

Вывод

Были найдены первые 10 нас пунктов по кол-ву объявлений (Петербург, Мурино, Шушары). Также были обнаружены первые и последние 10 нас пунктов по цене квадратного метра (см 2 таблицы выше).

Изучение предложений квартир

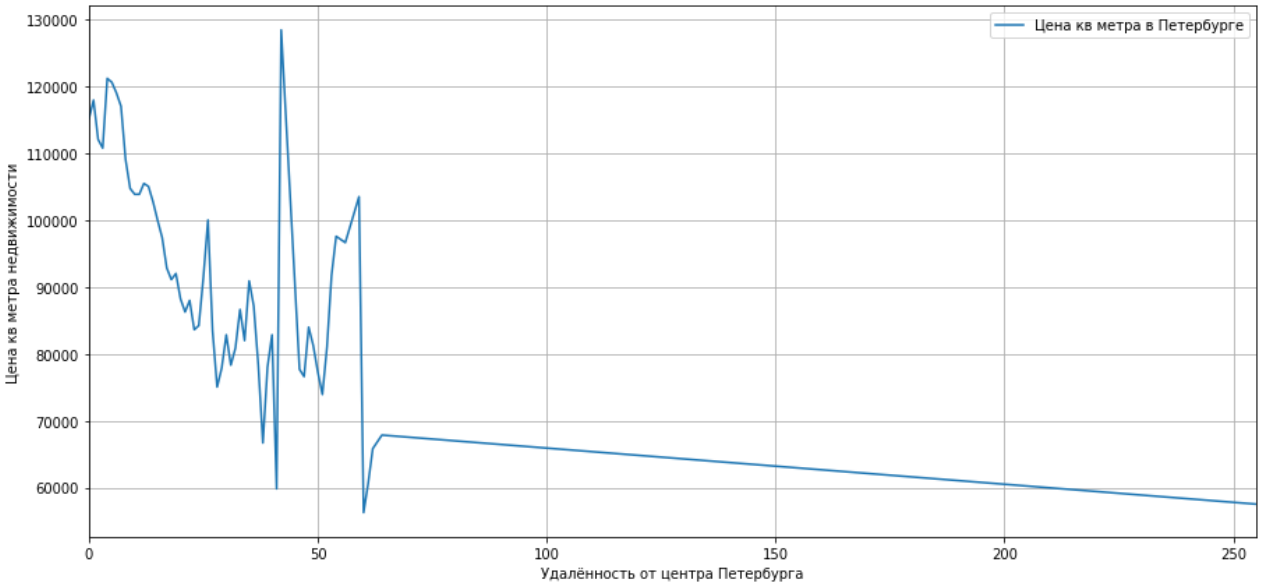
In [59]:
```
city_nearest_info = describe_enhanced(data, ['city_nearest_km'])
city_nearest_info
```

Out[59]:

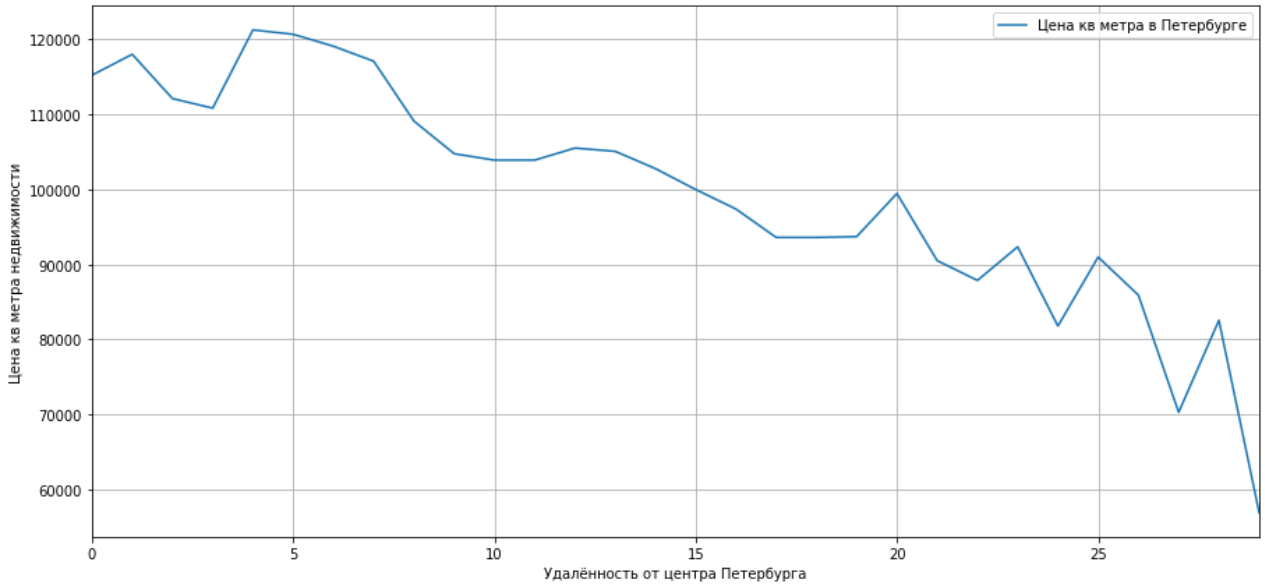
	city_nearest_km

	city_nearest_km
count	16434.000000
mean	71.743154
std	101.703815
min	0.000000
25%	12.000000
50%	16.000000
75%	36.000000
max	255.000000
low_iqr	0.000000
up_iqr	72.000000

In [60]:
```
ax = (data
    .groupby('city_nearest_km')['price_sq_m'].mean()
    .plot(label='Цена кв метра в Петербурге', figsize=(15, 7), grid=True, legend=True)
)

plt.xlabel('Удалённость от центра Петербурга')
plt.ylabel('Цена кв метра недвижимости')
plt.show()
```



In [61]:
```
ax = (data
    .query('location == "Санкт-Петербург"')
    .groupby('city_nearest_km')['price_sq_m'].mean()
    .plot(label='Цена кв метра в Петербурге', figsize=(15, 7), grid=True, legend=True)
)

plt.xlabel('Удалённость от центра Петербурга')
plt.ylabel('Цена кв метра недвижимости')
plt.show()
```



In [62]:
```
border_center = 7
```

Выделиние сегмента квартир в центре и его анализ.

Вывод

Посчитали среднюю цену для каждого километра и постройте график. Чем больше удаляемся от центра, тем цена ниже
(За исключением пары мест на 42 и 59 км - может быть связано с элитными пригородскими коттеджными посёлками,
в которых цены за кв метр существенно выше). Определили границу центральной зоны Петербурга - по графику смогли определить
значение (примерно 7-ой км является границей этой зоны, имеющей форму приблизительно окружности)

In [63]:
```
# cc - city_center
list_cols = 'location', 'total_area', 'price_m', 'rooms', 'ceiling_height', 'price_sq_m
                'day_exp', 'month_exp', 'year_exp'
data_cc = data.query('0 <= city_nearest_km <= @border_center').loc[:, list_cols]
data_cc.describe().T
```

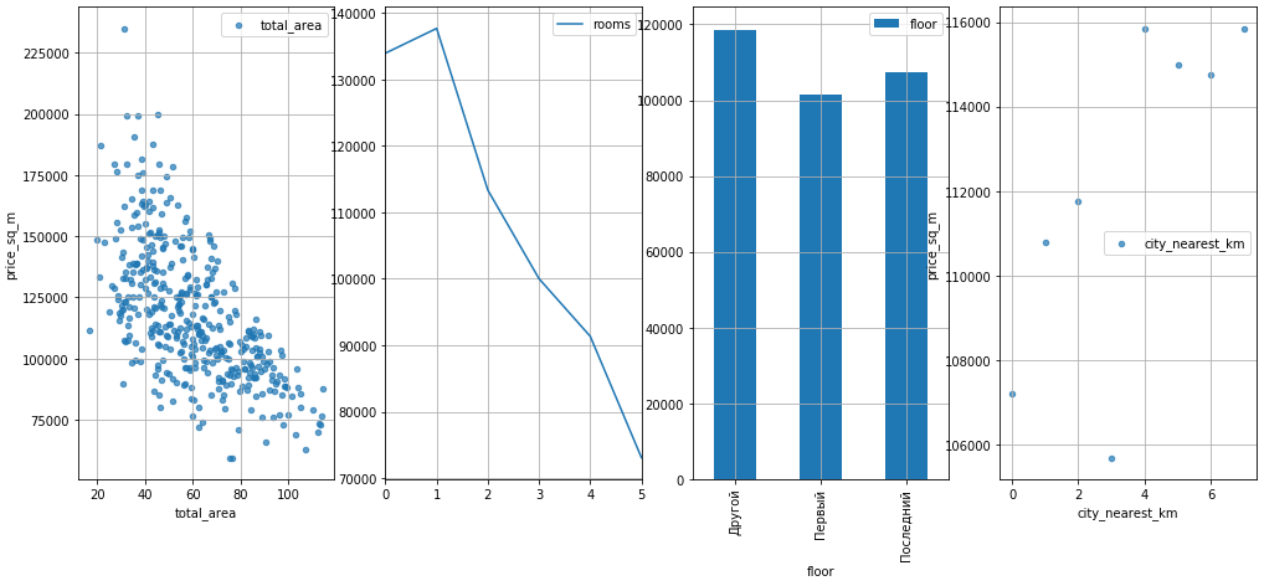
Out[63]:

	count	mean	std	min	25%	50%	75
total_area	1056.0	57.544205	18.327257	17.0	43.375000	55.700000	69.55000
price_m	1056.0	6.021780	1.778358	1.0	5.000000	6.000000	7.00000
rooms	1056.0	2.067235	0.887099	0.0	1.000000	2.000000	3.00000
ceiling_height	1056.0	2.000000	0.000000	2.0	2.000000	2.000000	2.00000
price_sq_m	1056.0	118015.925648	28650.381052	26250.0	97765.151515	113323.890463	134418.10344
city_nearest_km	1056.0	4.682765	1.703280	0.0	4.000000	5.000000	6.00000
day_exp	1056.0	15.047348	8.635957	1.0	8.000000	15.000000	23.00000
month_exp	1056.0	6.661932	3.408146	1.0	4.000000	7.000000	10.00000

	count	mean	std	min	25%	50%	75%
year_exp	1056.0	2017.344697	0.898936	2015.0	2017.000000	2017.000000	2018.00000

In [64]:
```
col_parse = 'price_sq_m'
cols_parse = ['total_area', 'rooms', 'floor', 'city_nearest_km']

fig, axises = plt.subplots(1, len(cols_parse))

for col, axis in zip(cols_parse, axises):
    if col != 'floor':
        correlation = round(data_cc[col_parse].corr(data_cc[col]), 2)
        print('{} ~ {: ^13} ->{: ^10}'.format(col_parse, col, correlation))
    if col == 'floor':
        (data_cc
            .groupby(col)[col_parse].median().reset_index()
            .plot(kind='bar', x='floor', y=col_parse, label=col, legend=True, grid=True
        )
    elif col == 'rooms':
        (data_cc
            .groupby(col)[col_parse].median().reset_index(drop=True)
            .plot(label=col, legend=True, grid=True, figsize=(17, 7), ax=axis)
        )
    else:
        (data_cc
            .groupby(col)[col_parse].median().reset_index()
            .plot(kind='scatter', x=col, y=col_parse, label=col,\
                  legend=True, grid=True, alpha=0.7, figsize=(17, 7), ax=axis)
        )

        plt.xlabel(col)
plt.show()
```

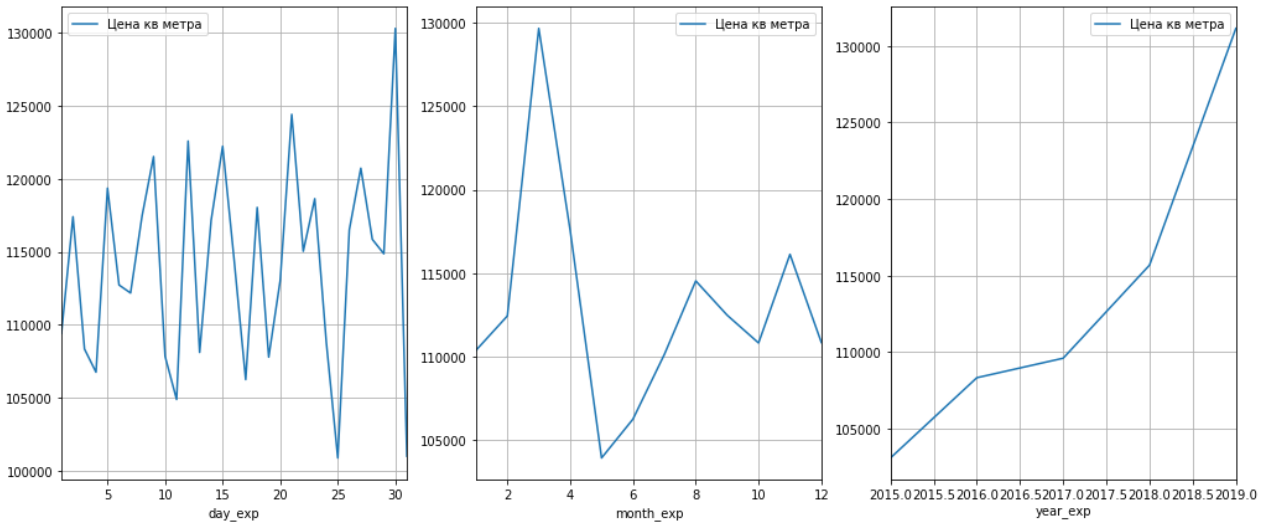
```
price_sq_m ~  total_area   ->  -0.51
price_sq_m ~     rooms     ->  -0.55
price_sq_m ~ city_nearest_km ->   0.04
```



In [65]:
```
col_parse = 'price_sq_m'
cols_parse = ['day_exp', 'month_exp', 'year_exp']
```

```
fig, axises = plt.subplots(1, len(cols_parse))

for col, axis in zip(cols_parse, axises):
    correlation = round(data_cc[col_parse].corr(data_cc[col]), 2)
    print('{} ~ {: ^13} ->{: ^10}'.format(col_parse, col, correlation))

    (data_cc
        .groupby(col)[col_parse].median()
        .plot(label='Цена кв метра', legend=True, grid=True, figsize=(17, 7), ax=axis)
    )
plt.show()
```

```
price_sq_m ~    day_exp    ->   0.02
price_sq_m ~   month_exp   ->  -0.06
price_sq_m ~   year_exp    ->   0.18
```



Вывод

Больше всего на стоимость квадратного метра в центре города влияют след параметры:
количество комнат (превалирует вариант жилья с одной комнатой) и площадь (в отличие от всей области в целом)
Также отличаются от ситуации по области и зависимость цены от времени продажи - 30-ые числа для старта объявления
оказываются наиболее удачными в плане продажи жилья и первой четверти года (март месяц). Что же касаемо тенденции
в росте цены кв метра, то можно утверждать, что цены на кв метр в центре не так подвластны снижению.
Рост наблюдается с 2015 по 2019 год. (нет "обвала" цен в 15 году, как наблюдается для всеё области в целом)