Survey of the advertisements on real estate sales

Data was provided by Yandex realty - archive of advertisements on sales of apartments in Saint-Petersburg and close cities for the last several years. It's required to learn how to estimate the market value of the realty. Main task - to set the parameters. It allows to develop automatization system which would tracks the anomalies and scammers activity.

For every apartment dataset has two types of data - inserted by users and automatically obtained, based on the map information (such as distance to city center, airport, closest park, water reservoir.

Exploration data analysis

In [1]:
```
import pandas as pd
import pylab as pl
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#  data import and display of it's head and info
try:
    df_aparts = pd.read_csv('/datasets/real_estate_data.csv',sep='\t')
except:
    df_aparts = pd.read_csv('real_estate_data.csv',sep='\t')
display(df_aparts.head(20))
df_aparts.info()
```

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcc
0	20	13000000.0	108.00	2019-03-07T00:00:00	3	2.70	16.0	51.00	8	NaN	...	25.00	N
1	7	3350000.0	40.40	2018-12-04T00:00:00	1	NaN	11.0	18.60	1	NaN	...	11.00	
2	10	5196000.0	56.00	2015-08-20T00:00:00	2	NaN	5.0	34.30	4	NaN	...	8.30	
3	0	64900000.0	159.00	2015-07-24T00:00:00	3	NaN	14.0	NaN	9	NaN	...	NaN	
4	2	10000000.0	100.00	2018-06-19T00:00:00	2	3.03	14.0	32.00	13	NaN	...	41.00	N
5	10	2890000.0	30.40	2018-09-10T00:00:00	1	NaN	12.0	14.40	5	NaN	...	9.10	N
6	6	3700000.0	37.30	2017-11-02T00:00:00	1	NaN	26.0	10.60	6	NaN	...	14.40	
7	5	7915000.0	71.60	2019-04-18T00:00:00	2	NaN	24.0	NaN	22	NaN	...	18.90	
8	20	2900000.0	33.16	2018-05-23T00:00:00	1	NaN	27.0	15.43	26	NaN	...	8.81	N
9	18	5400000.0	61.00	2017-02-26T00:00:00	3	2.50	9.0	43.60	7	NaN	...	6.50	
10	5	5050000.0	39.60	2017-11-16T00:00:00	1	2.67	12.0	20.30	3	NaN	...	8.50	N
11	9	3300000.0	44.00	2018-08-27T00:00:00	2	NaN	5.0	31.00	4	False	...	6.00	
12	10	3890000.0	54.00	2016-06-30T00:00:00	2	NaN	5.0	30.00	5	NaN	...	9.00	
13	20	3550000.0	42.80	2017-07-01T00:00:00	2	2.56	5.0	27.00	5	NaN	...	5.20	
14	1	4400000.0	36.00	2016-06-23T00:00:00	1	NaN	6.0	17.00	1	NaN	...	8.00	
15	16	4650000.0	39.00	2017-11-18T00:00:00	1	NaN	14.0	20.50	5	NaN	...	7.60	

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	kitchen_area	balcc
16	11	6700000.0	82.00	2017-11-23T00:00:00	3	3.05	5.0	55.60	1	NaN	...	9.00	N
17	6	4180000.0	36.00	2016-09-09T00:00:00	1	NaN	17.0	16.50	7	NaN	...	11.00	
18	8	3250000.0	31.00	2017-01-27T00:00:00	1	2.50	5.0	19.40	2	NaN	...	5.60	
19	16	14200000.0	121.00	2019-01-09T00:00:00	3	2.75	16.0	76.00	8	NaN	...	12.00	N

20 rows × 22 columns

```
<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)
```

In [3]:

```
df_aparts[df_aparts['rooms']==3].groupby('locality_name')['rooms'].count()
```

Out[3]:
```
locality_name
Бокситогорск                        3
Волосово                            9
Волхов                             23
Всеволожск                        100
Выборг                             62
                                 ...
садовое товарищество Приладожский   1
село Копорье                        1
село Павлово                        1
село Русско-Высоцкое                1
село Шум                            1
Name: rooms, Length: 199, dtype: int64
```

Conclusion

Based on the preliminary analysis it's possible to conclude the following:

1. Data frame has 23 699 rows и 22 columns;
2. Data frame contains information re: apartments area, city, cost and etc.;
3. A lot of columns has null values, so it's required to analyze such columns and fill up the nulls.

Data preparation

Task - to evaluate data in every column and replace the nulls

Nulls processing in column "ceiling_height"

In [4]:
```
# display the unique value of column
print(df_aparts['ceiling_height'].sort_values().unique())
df_aparts.ceiling_height.describe()
```

```
[  1.     1.2    1.75   2.     2.2    2.25   2.3    2.34   2.4    2.45
   2.46   2.47   2.48   2.49   2.5    2.51   2.52   2.53   2.54   2.55
   2.56   2.57   2.58   2.59   2.6    2.61   2.62   2.63   2.64   2.65
   2.66   2.67   2.68   2.69   2.7    2.71   2.72   2.73   2.74   2.75
   2.76   2.77   2.78   2.79   2.8    2.81   2.82   2.83   2.84   2.85
   2.86   2.87   2.88   2.89   2.9    2.91   2.92   2.93   2.94   2.95
   2.96   2.97   2.98   2.99   3.     3.01   3.02   3.03   3.04   3.05
   3.06   3.07   3.08   3.09   3.1    3.11   3.12   3.13   3.14   3.15
   3.16   3.17   3.18   3.2    3.21   3.22   3.23   3.24   3.25   3.26
   3.27   3.28   3.29   3.3    3.31   3.32   3.33   3.34   3.35   3.36
   3.37   3.38   3.39   3.4    3.42   3.43   3.44   3.45   3.46   3.47
   3.48   3.49   3.5    3.51   3.52   3.53   3.54   3.55   3.56   3.57
   3.58   3.59   3.6    3.62   3.63   3.65   3.66   3.67   3.68   3.69
   3.7    3.75   3.76   3.78   3.8    3.82   3.83   3.84   3.85   3.86
   3.87   3.88   3.9    3.93   3.95   3.98   4.     4.06   4.1    4.14
   4.15   4.19   4.2    4.25   4.3    4.37   4.4    4.45   4.5    4.65
   4.7    4.8    4.9    5.     5.2    5.3    5.5    5.6    5.8    6.
   8.     8.3   10.3   14.    20.    22.6   24.    25.    26.    27.
  27.5   32.   100.      nan]
```

Out[4]:

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

Based on the displayed data - we can conclude that height of ceiling is less than 5 meters but data also contains the anomalies such as 14.25 and 100 m., etc

In [5]:

```
df_distance_range = df_aparts.copy()

#  categorization of realty based on the distance to citycenter
df_distance_range['range_type'] = pd.qcut(df_distance_range['cityCenters_nearest'],3,['centre','regular','subruban'])

# fillna with median value based on the category
df_aparts['ceiling_height'] = df_distance_range.groupby('range_type')['ceiling_height'].apply(lambda x: x.fillna(x.median()))

# display of result
df_aparts.ceiling_height.describe()
```

Out[5]:
```
count    18180.000000
mean         2.759685
std          0.989702
min          1.000000
25%          2.600000
50%          2.600000
75%          2.950000
max        100.000000
Name: ceiling_height, dtype: float64
```

Nulls processing in column "floors_total"

The nulls proposed to fill with median value, it will not affect the price value.

In [6]:
```
# fillna with median value
df_aparts.floors_total = df_aparts.floors_total.fillna(df_aparts.floors_total.median())

# disply of the resullts
print(df_aparts.floors_total.describe())


# loop for replace of values if total floors value is less than floor value
def floors_check (df_name):
    if df_name['floors_total'] < df_name['floor']:
        return (df_name['floor'])
    else:
        return(df_name['floors_total'])

df_aparts['floor_type'] = df_aparts.apply(floors_check,axis=1)

df_aparts.floors_total.describe()
```

```
count    23699.000000
mean        10.667750
std          6.585961
min          1.000000
25%          5.000000
50%          9.000000
75%         16.000000
max         60.000000
Name: floors_total, dtype: float64
```

Out[6]:
```
count    23699.000000
mean        10.667750
std          6.585961
min          1.000000
25%          5.000000
50%          9.000000
75%         16.000000
max         60.000000
Name: floors_total, dtype: float64
```

Nulls processing in column "living_area"

Nulls proposed to fill up with value depends on the quantity of rooms in apartment. If living area value will be above total are? than the coefficient of median living area to median total area will be applied for calculation.

In [7]:
```
# display of information on the column
print(df_aparts.living_area.describe())

# calculation of median value coefficient
koef = round(df_aparts['living_area'].median()/df_aparts['total_area'].median(),2)
print('\n','Коэффициент',koef)

# fill nulls with value depending on the room quantity
df_aparts['living_area'] = df_aparts.groupby(['rooms','locality_name'])['living_area'].apply(lambda x: x.fillna(x.median()))
df_aparts['living_area'] = df_aparts['living_area'].fillna(df_aparts['total_area']*koef)

# dusplay the result
print('\n',df_aparts.living_area.describe())


# checking of the errors in living area value
def living_area_chek (df_name):
    if df_name['total_area'] < df_name['living_area']:
        return ('error')
    else:
        return('ok')

df_liv_area_check = df_aparts.copy()
df_liv_area_check['area_check'] = df_liv_area_check.apply(living_area_chek,axis=1)

# diaplpy the quantity of the errors
print('\n','количество квартир превыщающих площадь: ', df_liv_area_check.query('area_check == "error"')['area_check'].count())
```

```
# replace the error value with coefficient calculation
def living_area_update (df_name):
    if df_name['total_area'] < df_name['living_area']:
        return (df_name['total_area']*koef)
    else:
        return(df_name['living_area'])

df_aparts['living_area'] = df_aparts.apply(living_area_update,axis=1)

# checking of the result
print('\n',df_aparts.living_area.describe())
```

```
count    21796.000000
mean        34.457852
std         22.030445
min          2.000000
25%         18.600000
50%         30.000000
75%         42.300000
max        409.700000
Name: living_area, dtype: float64

 Коэффициент 0.58

 count    23699.000000
mean        34.322076
std         21.707464
min          2.000000
25%         18.500000
50%         30.000000
75%         42.500000
max        409.700000
Name: living_area, dtype: float64

 количество квартир превышающих площадь:  23

 count    23699.000000
mean        34.294980
std         21.679591
min          2.000000
25%         18.485000
50%         30.000000
75%         42.455000
max        409.700000
Name: living_area, dtype: float64
```

Nulls processing in column "is_apartment"

In [8]:

```python
# replace of nulls with False
df_aparts.is_apartment = df_aparts.is_apartment.fillna(False)

# change of datatype to bool
df_aparts.is_apartment = df_aparts.is_apartment.astype('bool')
df_aparts.is_apartment.describe()
```

Out[8]:
```
count        23699
unique           2
top          False
freq         23649
Name: is_apartment, dtype: object
```

Nulls processing in column"kitchen_area"

In [9]:
```
# display the info on the column
print(df_aparts.kitchen_area.describe())

# dataframe copy
df_temp = df_aparts.copy()

# categorization of column based on the total area value
df_temp['total_area_type'] = pd.qcut(df_aparts['total_area'],3,['small','medium','big'])

# fill the nuls based on the category
df_aparts['kitchen_area'] = df_temp.groupby(['total_area_type'])['kitchen_area'].apply(lambda x: x.fillna(x.median()))

# display the results
print('\n',df_aparts.kitchen_area.describe())


# chechking for the errors
def total_area_chek (df_name):
    if df_name['total_area'] < (df_name['living_area']+df_name['kitchen_area)']):
        return ('error')
    else:
        return('ok')

df_total_area_check = df_aparts.copy()
df_total_area_check['area_check'] = df_total_area_check.apply(living_area_chek,axis=1)

# display the quantity of error value
print('\n','количество квартир превыщающих площадь: ', df_total_area_check.query('area_check == "error"')['area_check'].count())
```

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

 count    23699.00000
mean        10.44548
std          5.65161
min          1.30000
25%          7.40000
50%          9.00000
75%         12.00000
max        112.00000
Name: kitchen_area, dtype: float64

 количество квартир превыщающих площадь:  0
```

Nulls processing in column "balcony"

In [10]:
```
# replacing of nulls with zero
df_aparts.balcony = df_aparts.balcony.fillna(0)
df_aparts.balcony.describe()
```

Out[10]:
```
count    23699.000000
mean         0.591080
std          0.959298
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          5.000000
Name: balcony, dtype: float64
```

Nulls processing in column "locality_name"

In [11]:
```
df_local_temp = df_aparts.copy()
print(df_local_temp.dropna().groupby('locality_name')['cityCenters_nearest'].min().sort_values(),
```

```
      '\n\n',df_local_temp.dropna().groupby('locality_name')['cityCenters_nearest'].count().sort_values())

# replacing of nulls with city name
df_aparts.query('cityCenters_nearest < 18006')['locality_name'] = df_aparts.query('cityCenters_nearest < 18006')['locality_name'

print('\n',df_aparts.locality_name.describe())

df_aparts.locality_name = df_aparts.locality_name.fillna('unkown')
df_aparts.locality_name.describe()
```

```
locality_name
Санкт-Петербург        208.0
посёлок Парголово    18006.0
посёлок Стрельна     22589.0
Пушкин               24311.0
Красное Село         28266.0
Петергоф             29815.0
Колпино              30438.0
Павловск             31533.0
Сестрорецк           33605.0
Кронштадт            46657.0
Зеленогорск          52628.0
Ломоносов            52768.0
Name: cityCenters_nearest, dtype: float64

 locality_name
Красное Село            7
посёлок Парголово      10
Ломоносов              14
Зеленогорск            15
посёлок Стрельна       15
Павловск               19
Колпино                61
Сестрорецк             69
Кронштадт              74
Петергоф              103
Пушкин                113
Санкт-Петербург      3606
Name: cityCenters_nearest, dtype: int64

 count               23650
unique                364
top       Санкт-Петербург
freq                15721
Name: locality_name, dtype: object
```

Out[11]:

```
count               23699
unique                365
top       Санкт-Петербург
freq                15721
Name: locality_name, dtype: object
```

Processing of remaining nulls in remaining columns

In [12]:
```python
# selection of columns
columns_to_fill = ['airports_nearest','cityCenters_nearest','parks_around3000','parks_nearest','ponds_around3000','ponds_nearest

# filling up of nulls with '-1'
for column in columns_to_fill:
    df_aparts[column] = df_aparts[column].fillna(-1)

df_aparts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 23 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        18180 non-null  float64
 6   floors_total          23699 non-null  float64
 7   living_area           23699 non-null  float64
 8   floor                 23699 non-null  int64  
 9   is_apartment          23699 non-null  bool   
 10  studio                23699 non-null  bool   
 11  open_plan             23699 non-null  bool   
 12  kitchen_area          23699 non-null  float64
 13  balcony               23699 non-null  float64
 14  locality_name         23699 non-null  object 
 15  airports_nearest      23699 non-null  float64
 16  cityCenters_nearest   23699 non-null  float64
 17  parks_around3000      23699 non-null  float64
 18  parks_nearest         23699 non-null  float64
 19  ponds_around3000      23699 non-null  float64
 20  ponds_nearest         23699 non-null  float64
 21  days_exposition       23699 non-null  float64
 22  floor_type            23699 non-null  float64
dtypes: bool(3), float64(15), int64(3), object(2)
memory usage: 3.7+ MB
```

Changing of data types

In [13]:
```
# selection of columns
columns_int = ['days_exposition', 'ponds_around3000', 'airports_nearest', 'cityCenters_nearest',
               'parks_around3000', 'parks_nearest', 'ponds_around3000', 'ponds_nearest', 'days_exposition',
               'floors_total', 'balcony']

# change of datapye to int
for column in columns_int:
    df_aparts[column] = df_aparts[column].astype('int')

df_aparts['first_day_exposition'] = pd.to_datetime(df_aparts['first_day_exposition'], format ='%Y-%m-%d')
df_aparts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 23 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  datetime64[ns]
 4   rooms                 23699 non-null  int64
 5   ceiling_height        18180 non-null  float64
 6   floors_total          23699 non-null  int32
 7   living_area           23699 non-null  float64
 8   floor                 23699 non-null  int64
 9   is_apartment          23699 non-null  bool
 10  studio                23699 non-null  bool
 11  open_plan             23699 non-null  bool
 12  kitchen_area          23699 non-null  float64
 13  balcony               23699 non-null  int32
 14  locality_name         23699 non-null  object
 15  airports_nearest      23699 non-null  int32
 16  cityCenters_nearest   23699 non-null  int32
 17  parks_around3000      23699 non-null  int32
 18  parks_nearest         23699 non-null  int32
 19  ponds_around3000      23699 non-null  int32
 20  ponds_nearest         23699 non-null  int32
 21  days_exposition       23699 non-null  int32
 22  floor_type            23699 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(6), int32(9), int64(3), object(1)
memory usage: 2.9+ MB
```

In [14]:
```
columns_int = ['days_exposition', 'ponds_around3000', 'airports_nearest', 'cityCenters_nearest',
               'parks_around3000', 'parks_nearest', 'ponds_around3000', 'ponds_nearest', 'days_exposition',
               'floors_total', 'balcony']

for column in columns_int:
    df_aparts[column] = df_aparts[column].astype('int')
```

In [15]:
```
df_aparts.head()
```

Out[15]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	balcony	locality_nai
0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	False	...	0	Сан Петерб
1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	False	...	2	посёл Шуша
2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	False	...	0	Сан Петерб
3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	False	...	0	Сан Петерб
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	False	...	0	Сан Петерб

5 rows × 23 columns

In [16]:
```
df_aparts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 23 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  datetime64[ns]
 4   rooms                 23699 non-null  int64         
 5   ceiling_height        18180 non-null  float64       
 6   floors_total          23699 non-null  int32         
 7   living_area           23699 non-null  float64       
 8   floor                 23699 non-null  int64         
 9   is_apartment          23699 non-null  bool          
 10  studio                23699 non-null  bool          
 11  open_plan             23699 non-null  bool          
 12  kitchen_area          23699 non-null  float64       
 13  balcony               23699 non-null  int32         
 14  locality_name         23699 non-null  object        
 15  airports_nearest      23699 non-null  int32         
 16  cityCenters_nearest   23699 non-null  int32         
 17  parks_around3000      23699 non-null  int32         
 18  parks_nearest         23699 non-null  int32         
 19  ponds_around3000      23699 non-null  int32         
 20  ponds_nearest         23699 non-null  int32         
 21  days_exposition       23699 non-null  int32         
 22  floor_type            23699 non-null  float64       
dtypes: bool(3), datetime64[ns](1), float64(6), int32(9), int64(3), object(1)
memory usage: 2.9+ MB
```

Conclusion

1) dataset had the nulls in following columns:

- 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;

2) For all column the nulls values in all columns were replaced:

- ceiling height with median values
- quantity of total floors with median values
- living area with value depends on the room quantity
- apartment column values with false
- kitchen areas with values calculated by coefficient from total area
- city to unknown
- other columns with zero.

3) changes in data types:

- Data with integer values were change to int
- apartment column to bool
- date of exposition to datetime
- other float data were unchanged due to possible influence of such data on realty price

4) the nulls in data could be lost by different reasons - users could skip it or just haven't got precise information.

Calculation and adding the relevant information to dataset

Calculation of cost per square meter

In [17]:
```
# calculation of square meter cost
df_aparts['price_per_meter'] = round(df_aparts['last_price']/df_aparts['total_area'],2)

# display ofthe results
df_aparts.head()
```

Out[17]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	locality_name	airpo
0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	False	...	Санкт-Петербург	
1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	False	...	посёлок Шушары	
2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	False	...	Санкт-Петербург	
3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	False	...	Санкт-Петербург	
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	False	...	Санкт-Петербург	

5 rows × 24 columns

New columns with year, month and day of exposition

In [18]:
```
# add a new columns do dataframe

df_aparts['exposition_year'] = df_aparts['first_day_exposition'].dt.year
df_aparts['exposition_month'] = df_aparts['first_day_exposition'].dt.month
df_aparts['exposition_weekday'] = df_aparts['first_day_exposition'].dt.weekday
df_aparts.head()
```

Out[18]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	parks_around3000
0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	False	...	1
1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	False	...	0
2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	False	...	1
3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	False	...	2
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	False	...	2

5 rows × 27 columns

Definition of floor of realty

In [19]:

```
# function for floor categorization
def floor_func (df_name):
    if df_name['floor'] == 1:
        return ('first_floor')
    elif df_name['floor'] == df_name['floors_total']:
        return ('last_floor')
    else:
        return('other')

# categorization by floor_type
df_aparts['floor_type'] = df_aparts.apply(floor_func,axis=1)

# display the results
df_aparts.head(20)
```

Out[19]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	parks_around3000
0	20	13000000.0	108.00	2019-03-07	3	2.70	16	51.00	8	False	...	1
1	7	3350000.0	40.40	2018-12-04	1	2.60	11	18.60	1	False	...	0
2	10	5196000.0	56.00	2015-08-20	2	2.60	5	34.30	4	False	...	1
3	0	64900000.0	159.00	2015-07-24	3	2.95	14	45.76	9	False	...	2
4	2	10000000.0	100.00	2018-06-19	2	3.03	14	32.00	13	False	...	2
5	10	2890000.0	30.40	2018-09-10	1	NaN	12	14.40	5	False	...	-1
6	6	3700000.0	37.30	2017-11-02	1	2.60	26	10.60	6	False	...	0
7	5	7915000.0	71.60	2019-04-18	2	2.60	24	31.00	22	False	...	0
8	20	2900000.0	33.16	2018-05-23	1	NaN	27	15.43	26	False	...	-1
9	18	5400000.0	61.00	2017-02-26	3	2.50	9	43.60	7	False	...	0
10	5	5050000.0	39.60	2017-11-16	1	2.67	12	20.30	3	False	...	1
11	9	3300000.0	44.00	2018-08-27	2	2.60	5	31.00	4	False	...	0
12	10	3890000.0	54.00	2016-06-30	2	NaN	5	30.00	5	False	...	-1
13	20	3550000.0	42.80	2017-07-01	2	2.56	5	27.00	5	False	...	1
14	1	4400000.0	36.00	2016-06-23	1	2.60	6	17.00	1	False	...	0
15	16	4650000.0	39.00	2017-11-18	1	2.60	14	20.50	5	False	...	1
16	11	6700000.0	82.00	2017-11-23	3	3.05	5	55.60	1	False	...	3
17	6	4180000.0	36.00	2016-09-09	1	2.60	17	16.50	7	False	...	0
18	8	3250000.0	31.00	2017-01-27	1	2.50	5	19.40	2	False	...	1
19	16	14200000.0	121.00	2019-01-09	3	2.75	16	76.00	8	False	...	0

20 rows × 27 columns

Calculation of ratio of realty areas

In [20]:
```
# calculation of proportion of living area to total
df_aparts['living_to_total_percent'] = round(df_aparts.living_area/df_aparts.total_area,2)

# display the results
df_aparts.head()
```

Out[20]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	parks_nearest	pond
0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	False	...	482	
1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	False	...	-1	
2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	False	...	90	
3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	False	...	84	
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	False	...	112	

5 rows × 28 columns

In [21]:
```
# calculation of proportion of kitchen area to total
df_aparts['kitchen_to_total_percent'] = round(df_aparts.kitchen_area/df_aparts.total_area,2)

# display the results
df_aparts.head()
```

Out[21]:

	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	is_apartment	...	ponds_around3000
0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	False	...	2
1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	False	...	0
2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	False	...	2
3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	False	...	3
4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	False	...	1

5 rows × 29 columns

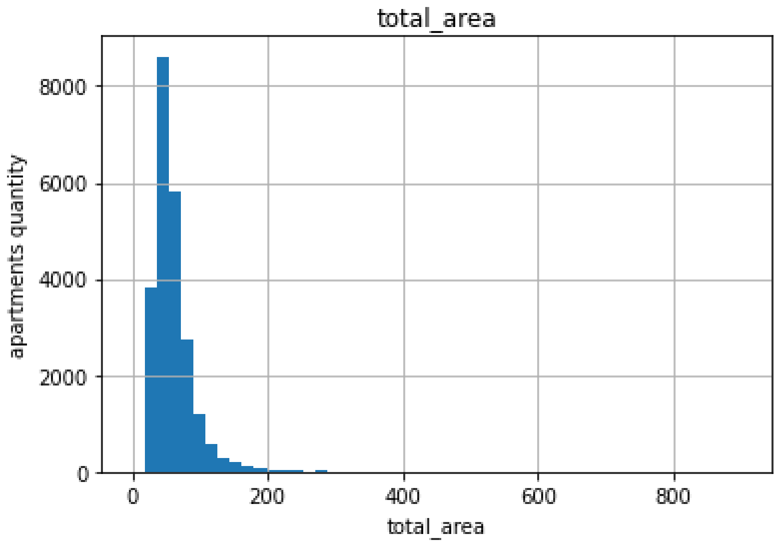
Statistical data analysis

Histogram plotting

Histogram of total area values

In [22]:
```
df_aparts.hist(column = 'total_area', bins=50 ,range = (0,df_aparts['total_area'].max()))
pl.xlabel("total_area")
pl.ylabel("apartments quantity")
```

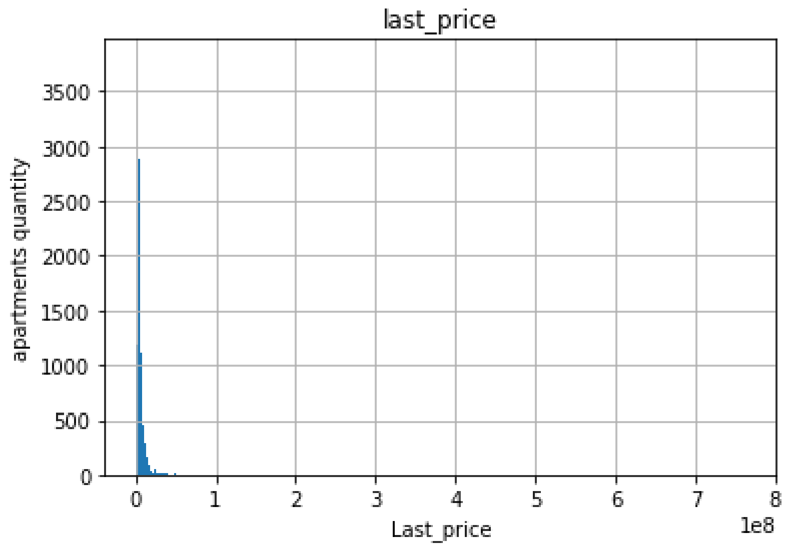
Out[22]:
```
Text(0, 0.5, 'apartments quantity')
```



Histogram of prices

In [23]:
```
df_aparts.hist(column = 'last_price',bins=1000, range=(0,df_aparts['last_price'].max()))
pl.xlabel("Last_price")
pl.ylabel("apartments quantity")
```

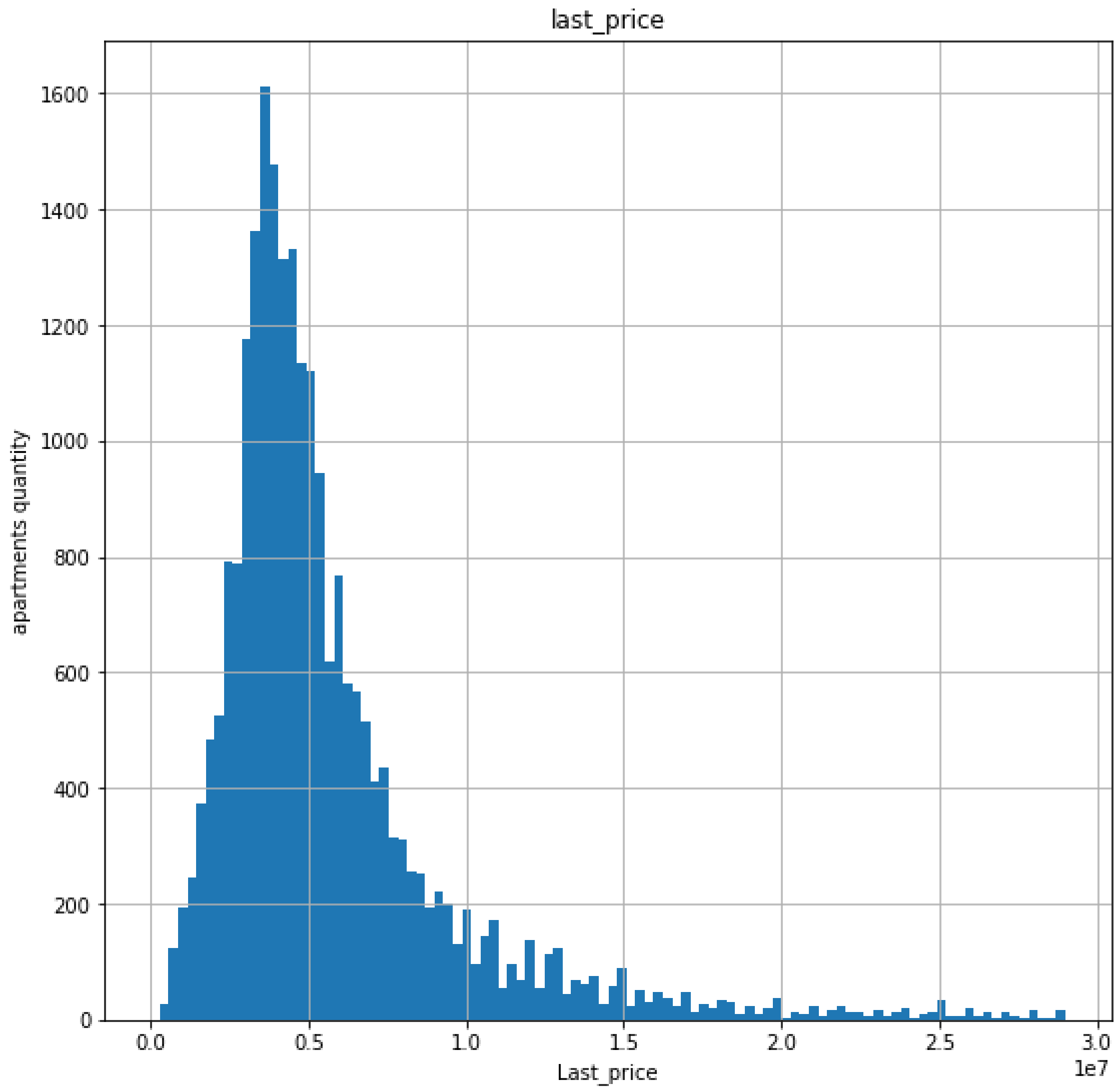
Out[23]:
```
Text(0, 0.5, 'apartments quantity')
```



Rescaling

In [24]:
```
df_aparts.hist(column = 'last_price',bins=100, range=(0,29000000),figsize=(9,9))
pl.xlabel("Last_price")
pl.ylabel("apartments quantity")
```

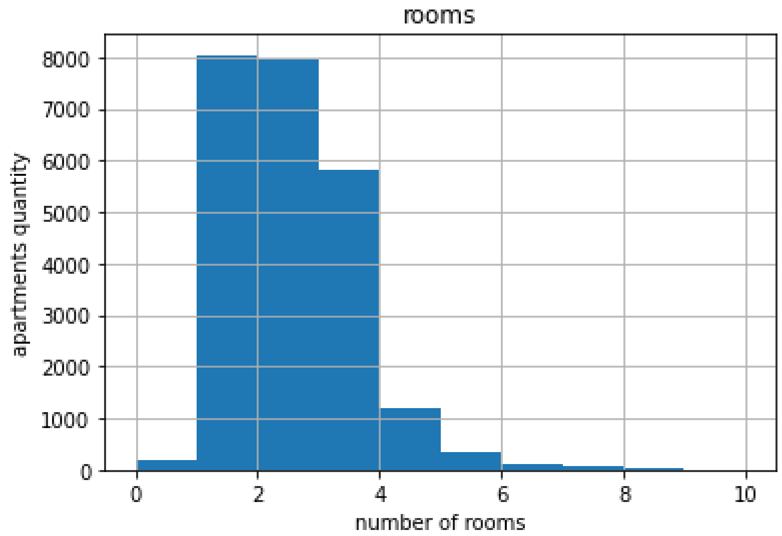
Out[24]:
```
Text(0, 0.5, 'apartments quantity')
```



Histogram of room quantity

In [25]:
```
df_aparts.hist(column='rooms',bins = 10, range =( 0, 10))
pl.xlabel("number of rooms")
pl.ylabel("apartments quantity")
```

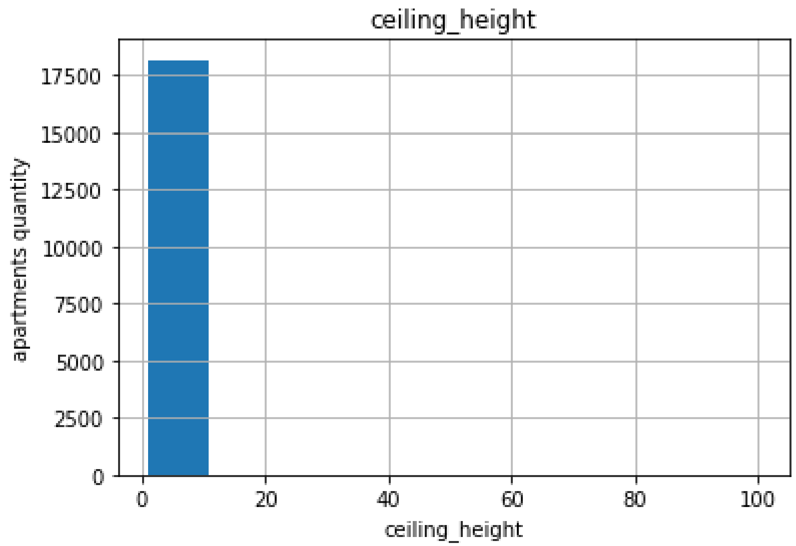
Out[25]:
```
Text(0, 0.5, 'apartments quantity')
```



Ceiling height histogram plotting

In [26]:
```
df_aparts.hist(column='ceiling_height',bins = 10)
pl.xlabel("ceiling_height")
pl.ylabel("apartments quantity")
```

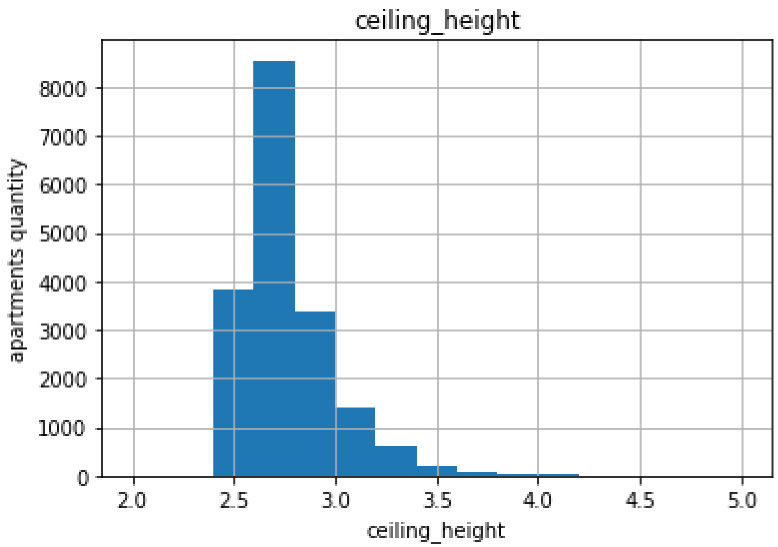
Out[26]:
```
Text(0, 0.5, 'apartments quantity')
```



Rescaling

In [27]:
```
df_aparts.hist(column= 'ceiling_height',bins = 15, range=(2, 5))
pl.xlabel("ceiling_height")
pl.ylabel("apartments quantity")
```

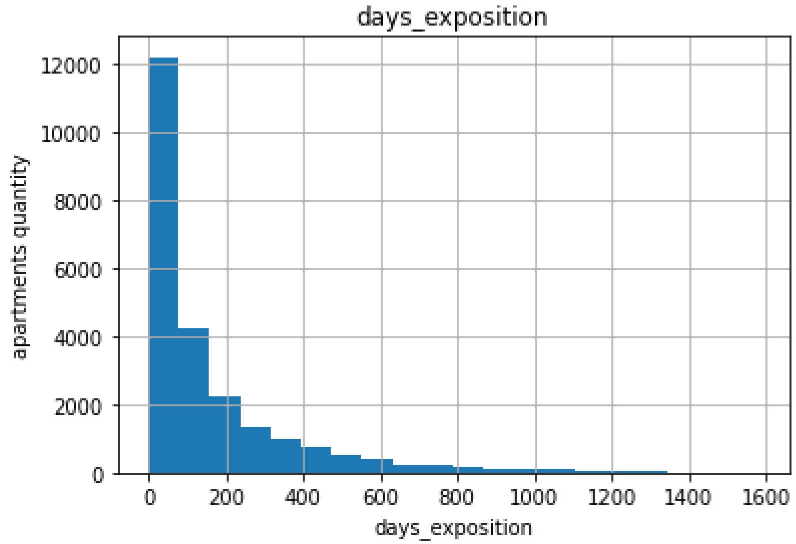
Out[27]:
```
Text(0, 0.5, 'apartments quantity')
```



Advertisement duration histogram

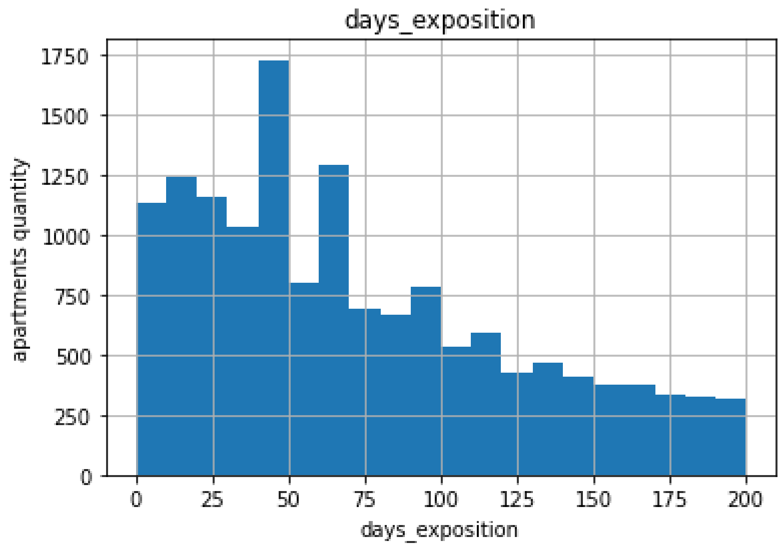
In [28]:
```
df_aparts.hist(column = 'days_exposition', bins=20)
pl.xlabel("days_exposition")
pl.ylabel("apartments quantity")

df_aparts.days_exposition.describe()
```

Out[28]:
```
count    23699.000000
mean       156.474619
std        213.645563
min         -1.000000
25%         22.000000
50%         74.000000
75%        199.000000
max       1580.000000
Name: days_exposition, dtype: float64
```



In [29]:
```
df_aparts.hist(column = 'days_exposition', bins=20,range = (0,200))
pl.xlabel("days_exposition")
pl.ylabel("apartments quantity")
```

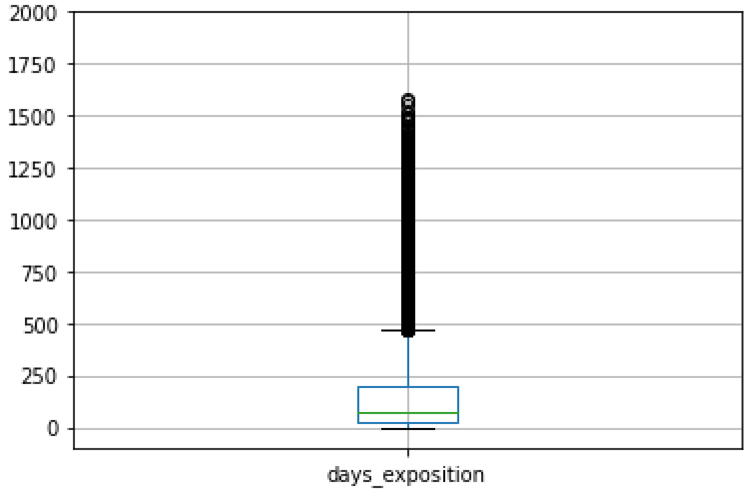
Out[29]:
```
Text(0, 0.5, 'apartments quantity')
```



Histogram shows that the highest quantity of realties were published during 50 and 60 days. Most likely the users were waiting for the exact quantity of days to sold the apartment with higher profit but not loner than 50/60 days.

In [30]:
```
plt.ylim(-100, 2000)
df_aparts.boxplot('days_exposition')
```

Out[30]: <AxesSubplot:>



If realty was sold faster than 22 days - it's too fast.
if longer than 190 days it's too long

Search and deletion of anomalies

During the data preparation some of anomalies were revealed such as ceiling height

Creation of copy of dataset to save the original data and deletion of anomalies

In [31]:
```
# df copy
df_2 = df_aparts.copy()
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 29 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  datetime64[ns]
 4   rooms                   23699 non-null  int64
 5   ceiling_height          18180 non-null  float64
 6   floors_total            23699 non-null  int32
 7   living_area             23699 non-null  float64
 8   floor                   23699 non-null  int64
 9   is_apartment            23699 non-null  bool
 10  studio                  23699 non-null  bool
 11  open_plan               23699 non-null  bool
 12  kitchen_area            23699 non-null  float64
 13  balcony                 23699 non-null  int32
 14  locality_name           23699 non-null  object
 15  airports_nearest        23699 non-null  int32
 16  cityCenters_nearest     23699 non-null  int32
 17  parks_around3000        23699 non-null  int32
 18  parks_nearest           23699 non-null  int32
 19  ponds_around3000        23699 non-null  int32
 20  ponds_nearest           23699 non-null  int32
 21  days_exposition         23699 non-null  int32
 22  floor_type              23699 non-null  object
 23  price_per_meter         23699 non-null  float64
 24  exposition_year         23699 non-null  int64
 25  exposition_month        23699 non-null  int64
 26  exposition_weekday      23699 non-null  int64
 27  living_to_total_percent  23699 non-null  float64
 28  kitchen_to_total_percent  23699 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(8), int32(9), int64(6), object(2)
memory usage: 4.0+ MB
```

In [32]:

```
# deletion of values higher than 4,25 meters
df_2 = df_2.query('ceiling_height <= 4.25').reset_index()
```

Deletion of realties which were sold too fast ot were not sold for a very long time

In [33]:
```
df_2 = df_2.query('(days_exposition >3 or days_exposition <1400) and days_exposition !=0 ').reset_index(drop=True)
```

Deletion of raelty with huge total area

In [34]:
```
df_2 = df_2.query('total_area <550').reset_index(drop=True)
```

Deletion of overpriced realty

In [35]:
```
df_2=df_2.query('last_price < 300000000').reset_index(drop=True)
```

Analysis of parameter which infuence on the realty price

In [36]:
```
# declare function for cagerozation by floor
def floor_func (df_name):
    if df_name['floor'] == 1:
        return (0)
    elif df_name['floor'] == df_name['floors_total']:
        return (2)
    else:
        return(1)

# adding new column with floor category
df_2['floor_type_key'] = df_2.apply(floor_func,axis=1)
df_2.head()
```

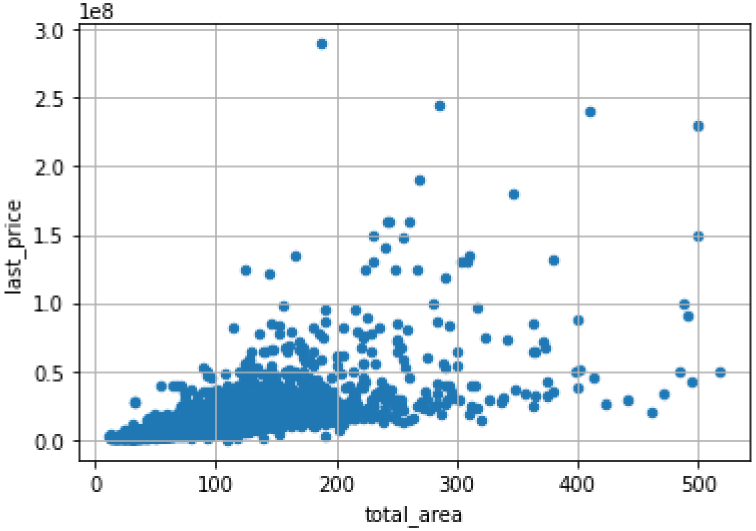
Out[36]:

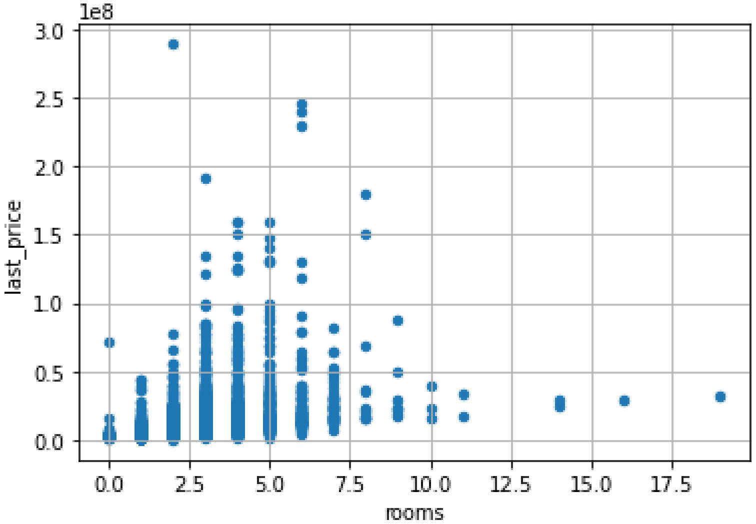
	index	total_images	last_price	total_area	first_day_exposition	rooms	ceiling_height	floors_total	living_area	floor	...	ponds_nearest	days_exposi
0	0	20	13000000.0	108.0	2019-03-07	3	2.70	16	51.00	8	...	755	
1	1	7	3350000.0	40.4	2018-12-04	1	2.60	11	18.60	1	...	-1	
2	2	10	5196000.0	56.0	2015-08-20	2	2.60	5	34.30	4	...	574	
3	3	0	64900000.0	159.0	2015-07-24	3	2.95	14	45.76	9	...	234	
4	4	2	10000000.0	100.0	2018-06-19	2	3.03	14	32.00	13	...	48	

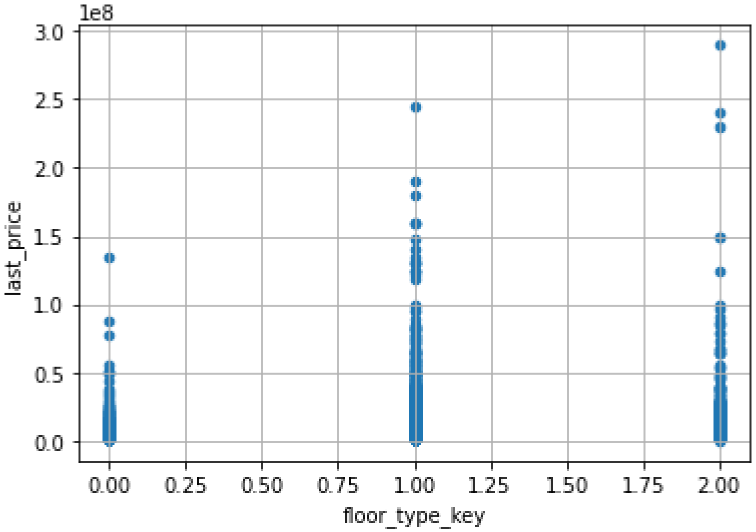
5 rows × 31 columns

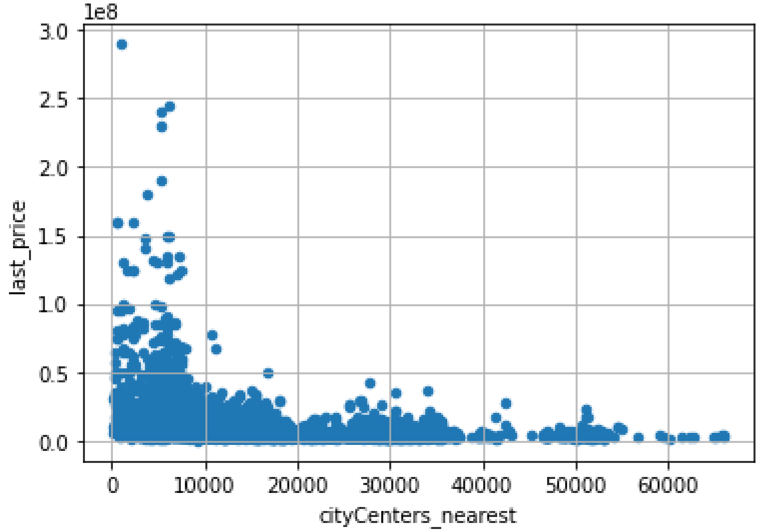
In [37]:
```
# selection of columns with highest affect on the price
data_list = ['total_area','rooms','floor_type_key','cityCenters_nearest','exposition_year','exposition_month','exposition_weekday

# cycle for plottin of histagram of correlation of columns values to the price
for data in data_list:
    df_2.plot(y='last_price', x = data, kind = 'scatter', grid=True)
    print(data, 'coeff:', round(df_2['last_price'].corr(df_2[data]),5))
```

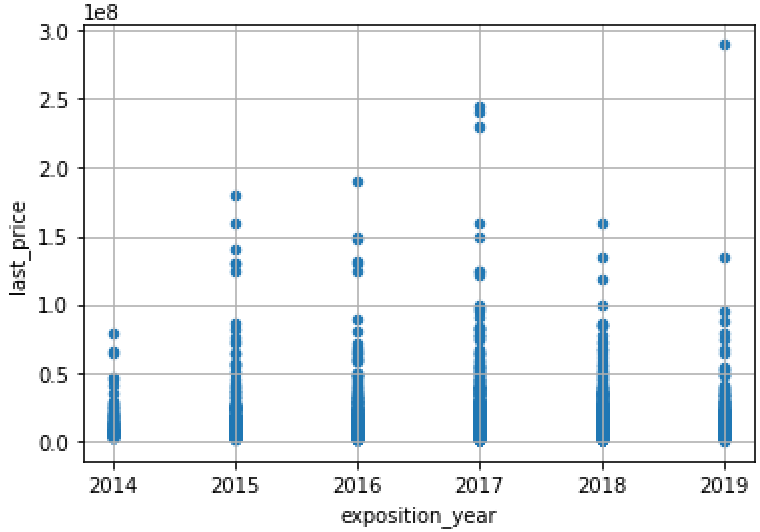
```
total_area coeff: 0.70914
rooms coeff: 0.42109
floor_type_key coeff: 0.06508
cityCenters_nearest coeff: -0.25546
exposition_year coeff: -0.05197
exposition_month coeff: -0.0035
exposition_weekday coeff: -0.0003
```

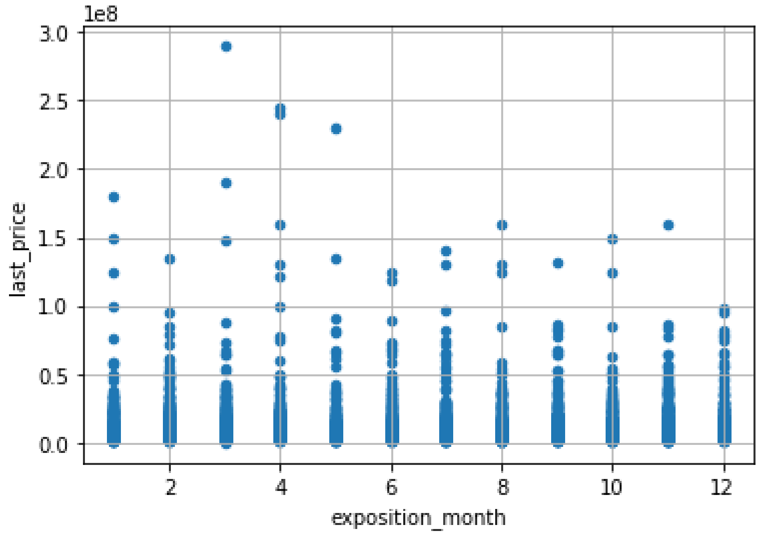


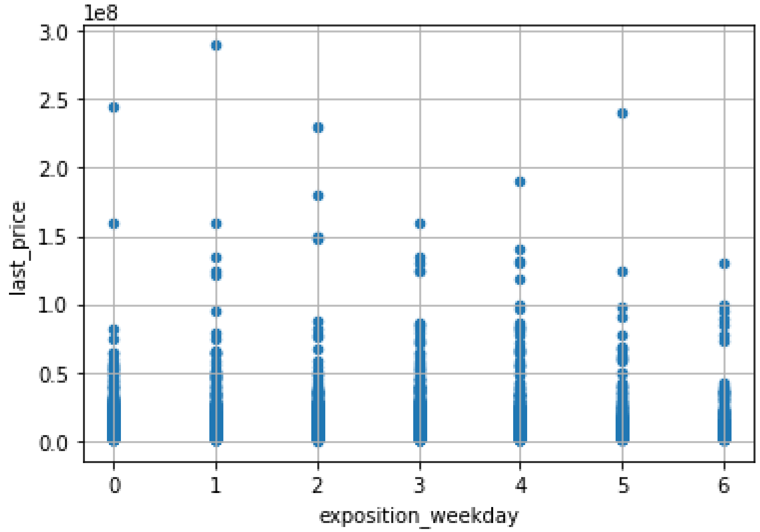

1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
last_price
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
rooms


1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
last_price
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
floor_type_key


1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
last_price
0
10000
20000
30000
40000
50000
60000
cityCenters_nearest


1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
last_price
2014
2015
2016
2017
2018
2019
exposition_year





Conclusions

- Highest dependence on the realty price affect the total area - 70%
- Next one is quantity of bedrooms - 40% dependence
- Dependence of the floor of realty on price is 5.6%
- Dependence of the distance to city center on the price is 5%
- Date and month of publishing has negative dependence (-5%)

- Year of publishing has also negative dependence (-8%)

Search for the cities with maximum quantity of realty and maximum average price

In [38]:
```
df_cities = df_2.groupby('locality_name').count().sort_values(by='last_price',ascending=False)
df_cities = df_cities.query('last_price >= 208')
df_cities_price = df_2.query('locality_name in (@df_cities.index)')
df_cities_price = round(df_cities_price.groupby('locality_name')['price_per_meter'].mean(),2).sort_values(ascending=False)
df_cities_price
```

Out[38]:
```
locality_name
Санкт-Петербург      114180.07
Пушкин               103070.37
посёлок Парголово     90175.91
посёлок Шушары        78474.36
Колпино               75402.50
Name: price_per_meter, dtype: float64
```

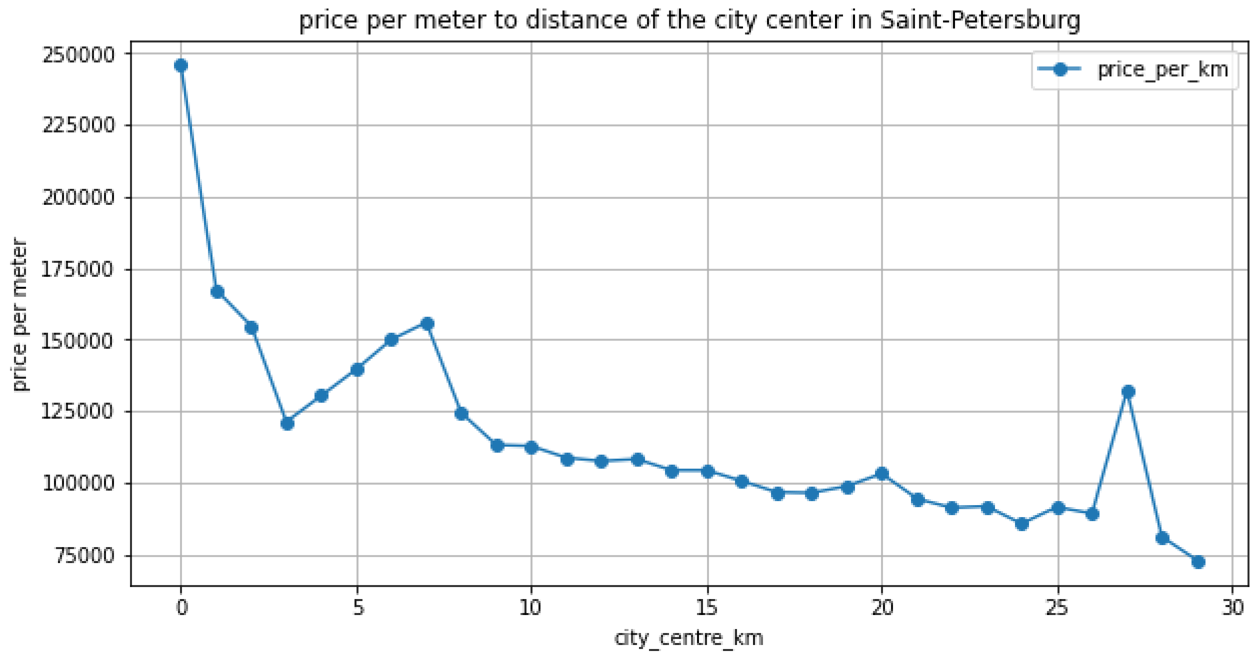
Maximum quantity of advertisements were placed from the following cities:

- Санкт-Петербург
- посёлок Мурино
- посёлок Шушары
- Всеволожск
- Колпино

Definition of apartments price in city center of Saint-Petersburg

In [39]:
```
df_spb = df_2.query('locality_name in "Санкт-Петербург" and cityCenters_nearest !=0').copy()
df_spb['city_centre_km'] = round(df_spb['cityCenters_nearest']/1000,0)
df_spb_average_km = df_spb.groupby('city_centre_km')['price_per_meter'].mean()

df_spb_average_km.plot(style='o-',grid=True,figsize=(10,5), label = 'price_per_km',legend = True)
plt.title("price per meter to distance of the city center in Saint-Petersburg")
pl.ylabel("price per meter")
```

Out[39]:
```
Text(0, 0.5, 'price per meter')
```



Based on the information from graph we assume that realty in city center is with distance equal to 3 km or less

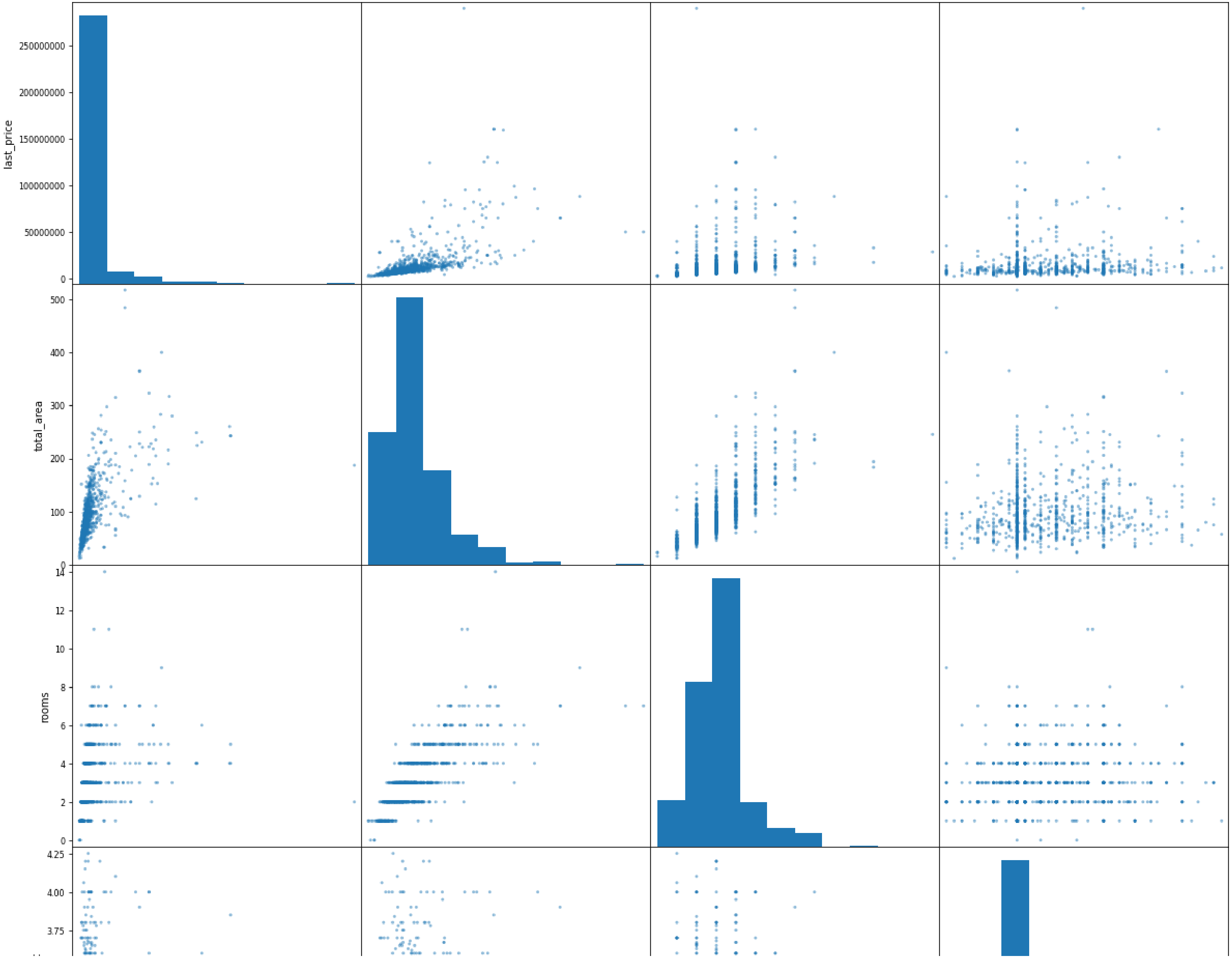
Analysis of parameters of reallty in citycenter

In [40]:

```
df_spb_centre = df_spb.query('city_centre_km <=3')

pd.plotting.scatter_matrix(df_spb_centre[['last_price','total_area','rooms','ceiling_height']],figsize=(20,20))
df_spb_centre[['last_price','total_area','rooms','ceiling_height']].corr()
```

Out[40]:

	last_price	total_area	rooms	ceiling_height
last_price	1.000000	0.595637	0.281825	0.124851
total_area	0.595637	1.000000	0.748886	0.152595
rooms	0.281825	0.748886	1.000000	0.066789
ceiling_height	0.124851	0.152595	0.066789	1.000000



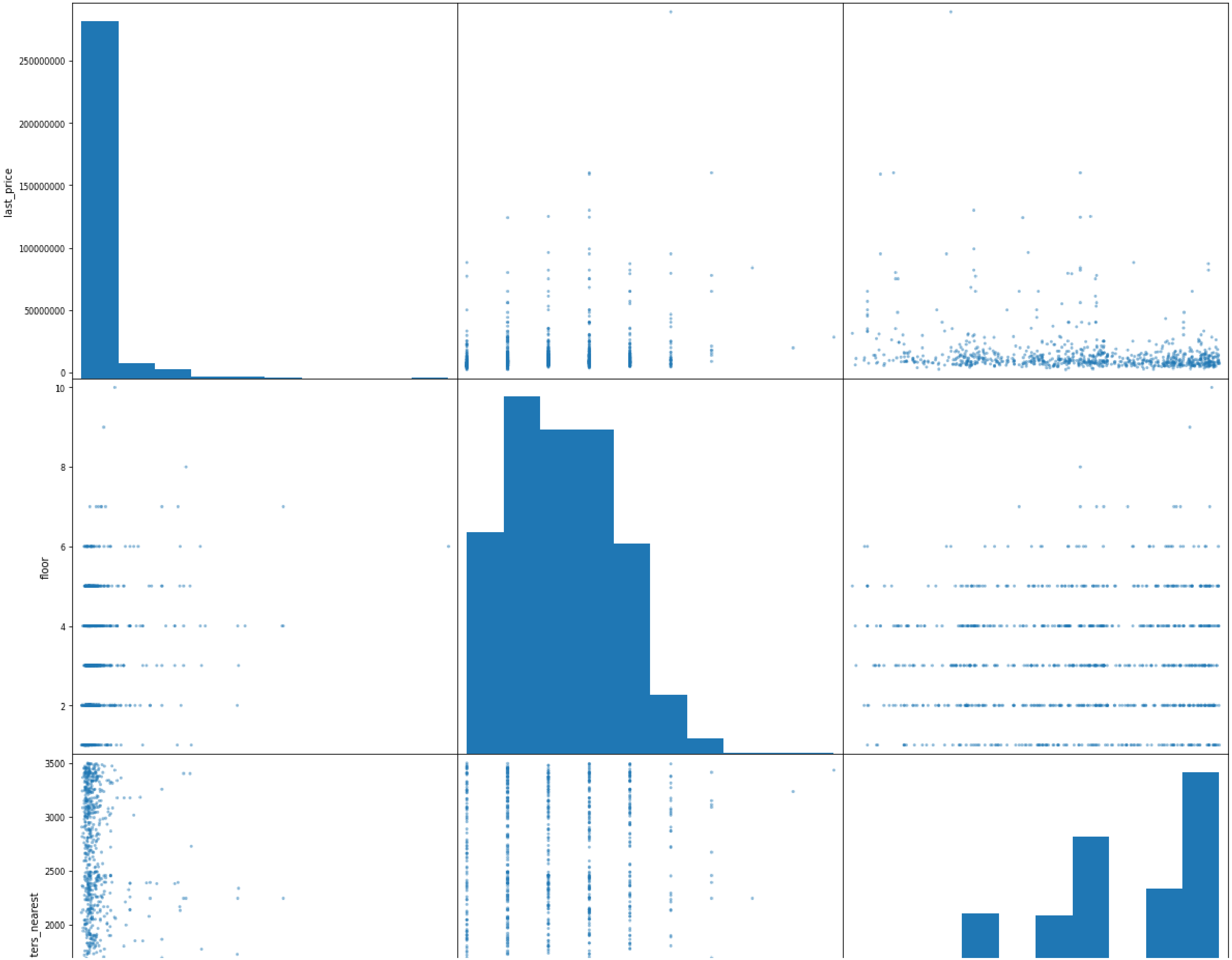
last_price
total_area
rooms
250000000
200000000
150000000
100000000
50000000
0
500
400
300
200
100
0
14
12
10
8
6
4
2
0
4.25
4.00
3.75

In [41]:

```
pd.plotting.scatter_matrix(df_spb_centre[['last_price','floor','cityCenters_nearest']],figsize=(20,20))
```

Out[41]:

```
array([[<AxesSubplot:xlabel='last_price', ylabel='last_price'>,
        <AxesSubplot:xlabel='floor', ylabel='last_price'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='last_price'>],
       [<AxesSubplot:xlabel='last_price', ylabel='floor'>,
        <AxesSubplot:xlabel='floor', ylabel='floor'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='floor'>],
       [<AxesSubplot:xlabel='last_price', ylabel='cityCenters_nearest'>,
        <AxesSubplot:xlabel='floor', ylabel='cityCenters_nearest'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='cityCenters_nearest'>]],
      dtype=object)
```



last_price
250000000
200000000
150000000
100000000
50000000
0
floor
10
8
6
4
2
ters_nearest
3500
3000
2500
2000

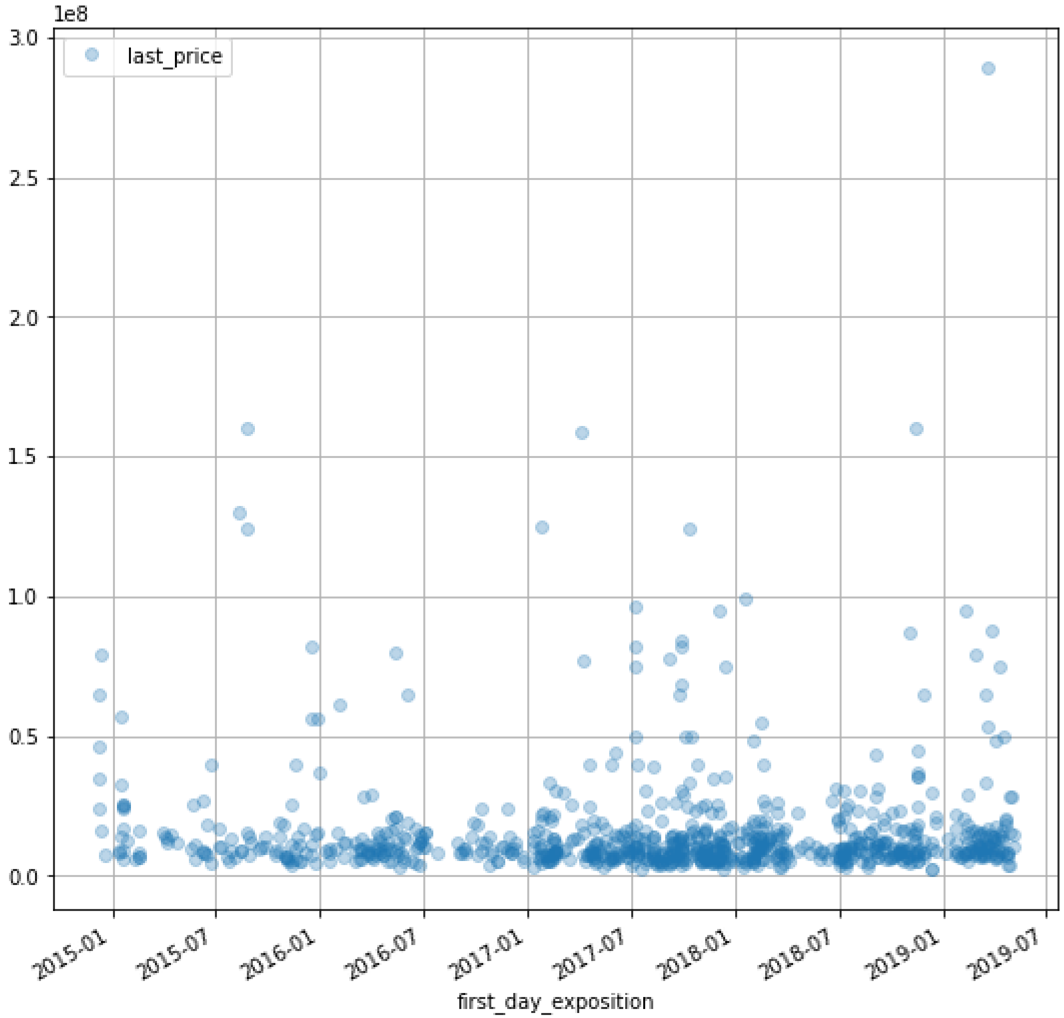
In [42]:
```
display(df_spb_centre[['last_price','floor','cityCenters_nearest']].corr())
```

	last_price	floor	cityCenters_nearest
last_price	1.000000	0.183078	-0.193443
floor	0.183078	1.000000	0.054866
cityCenters_nearest	-0.193443	0.054866	1.000000

In [43]:
```
df_spb_centre.plot(style='o',y='last_price',x='first_day_exposition',grid=True,figsize = (9,9),alpha = 0.3)
df_spb_centre[['last_price','exposition_year']].corr()
```

Out[43]:

	last_price	exposition_year
last_price	1.000000	-0.023326
exposition_year	-0.023326	1.000000



City center realty price dependence conclusion

1) The highest influence on the realty price in city center is caused by total area, the dependance is 64%

2) Second highest parameter is quantity of bedrooms, dependance - 34%

3) Third parameter is realty floor, dependance 20%

4) Fourth - ceiling height, dependance 15%

5) Fifth one is year of publishing, dependance -7% (negative value)

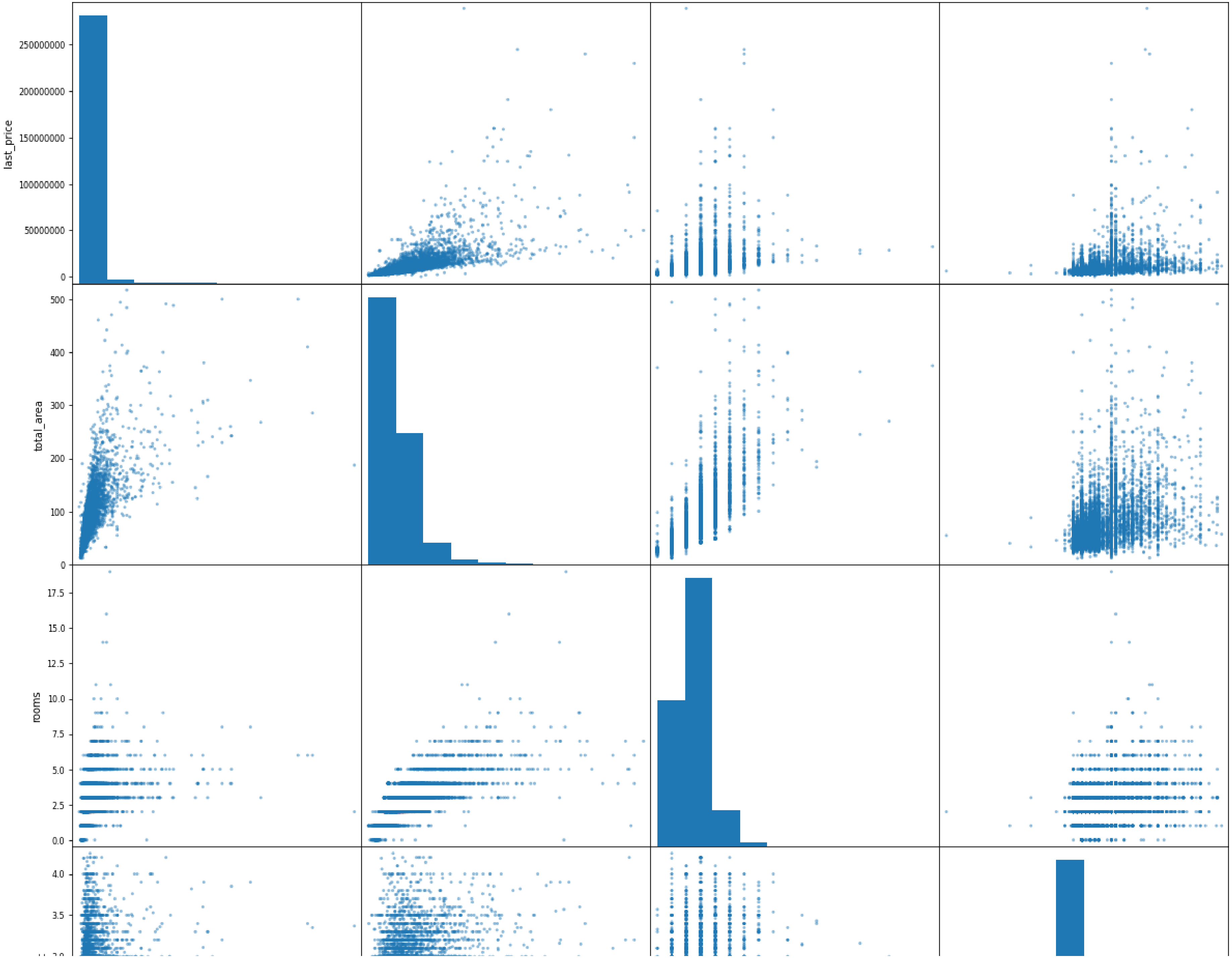
6) Last one is distance to city center , dependance -17% (negative value)

Analysis of apartment in Saint-Petersburg overall

In [44]:
```
display(df_spb[['last_price','floor','cityCenters_nearest']].corr())
pd.plotting.scatter_matrix(df_spb[['last_price','total_area','rooms','ceiling_height']],figsize=(20,20))
```

	last_price	floor	cityCenters_nearest
last_price	1.000000	-0.013061	-0.319131
floor	-0.013061	1.000000	0.228746
cityCenters_nearest	-0.319131	0.228746	1.000000

Out[44]:
```
array([[<AxesSubplot:xlabel='last_price', ylabel='last_price'>,
        <AxesSubplot:xlabel='total_area', ylabel='last_price'>,
        <AxesSubplot:xlabel='rooms', ylabel='last_price'>,
        <AxesSubplot:xlabel='ceiling_height', ylabel='last_price'>],
       [<AxesSubplot:xlabel='last_price', ylabel='total_area'>,
        <AxesSubplot:xlabel='total_area', ylabel='total_area'>,
        <AxesSubplot:xlabel='rooms', ylabel='total_area'>,
        <AxesSubplot:xlabel='ceiling_height', ylabel='total_area'>],
       [<AxesSubplot:xlabel='last_price', ylabel='rooms'>,
        <AxesSubplot:xlabel='total_area', ylabel='rooms'>,
        <AxesSubplot:xlabel='rooms', ylabel='rooms'>,
        <AxesSubplot:xlabel='ceiling_height', ylabel='rooms'>],
       [<AxesSubplot:xlabel='last_price', ylabel='ceiling_height'>,
        <AxesSubplot:xlabel='total_area', ylabel='ceiling_height'>,
        <AxesSubplot:xlabel='rooms', ylabel='ceiling_height'>,
        <AxesSubplot:xlabel='ceiling_height', ylabel='ceiling_height'>]],
      dtype=object)
```

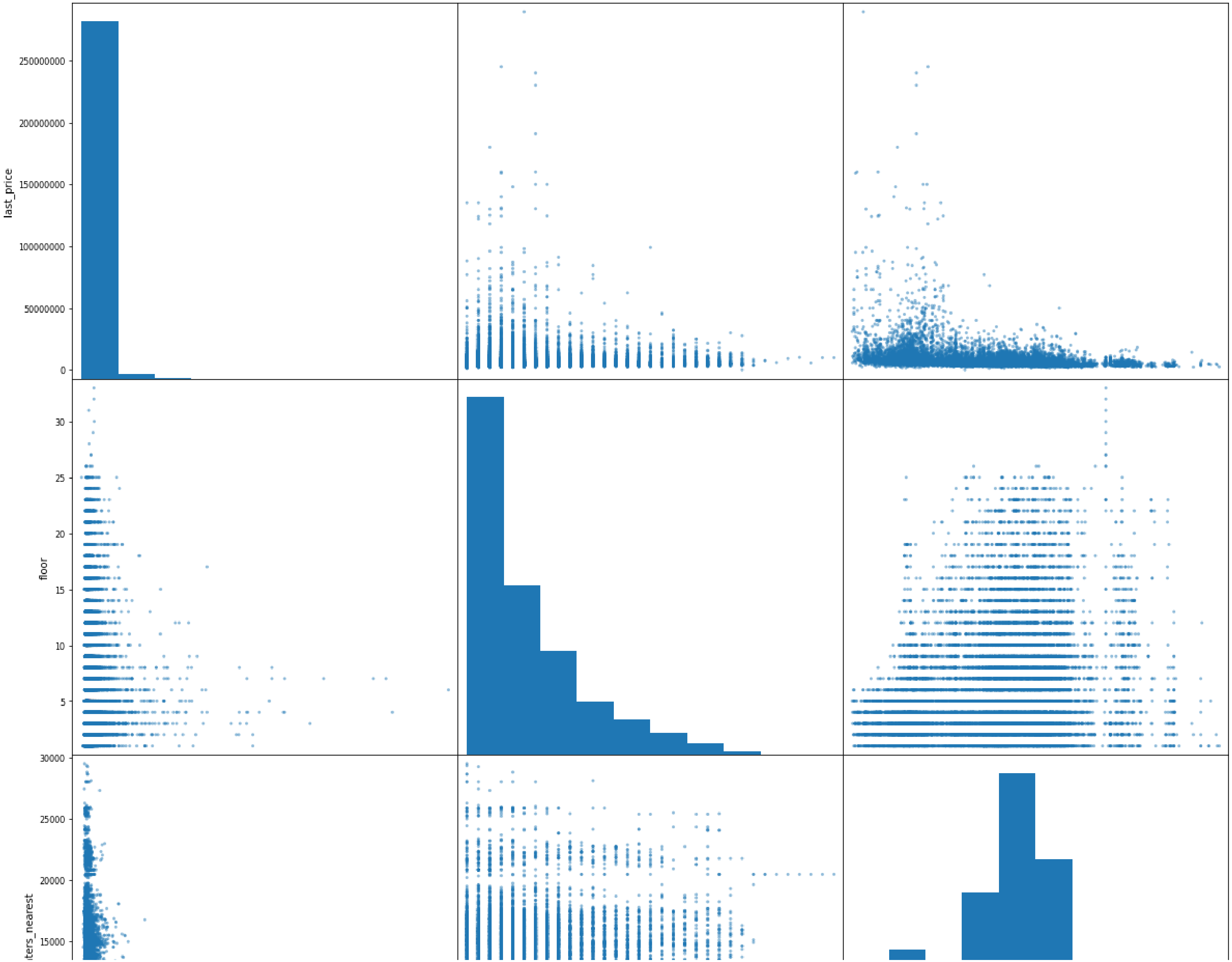


last_price
total_area
rooms
250000000
200000000
150000000
100000000
50000000
0
500
400
300
200
100
0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
4.0
3.5

In [45]:
```
display(df_spb[['last_price','total_area','rooms','ceiling_height']].corr())
pd.plotting.scatter_matrix(df_spb[['last_price','floor','cityCenters_nearest']],figsize=(20,20))
```

	last_price	total_area	rooms	ceiling_height
last_price	1.000000	0.711434	0.420523	0.362052
total_area	0.711434	1.000000	0.765493	0.431705
rooms	0.420523	0.765493	1.000000	0.296858
ceiling_height	0.362052	0.431705	0.296858	1.000000

Out[45]:
```
array([[<AxesSubplot:xlabel='last_price', ylabel='last_price'>,
        <AxesSubplot:xlabel='floor', ylabel='last_price'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='last_price'>],
       [<AxesSubplot:xlabel='last_price', ylabel='floor'>,
        <AxesSubplot:xlabel='floor', ylabel='floor'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='floor'>],
       [<AxesSubplot:xlabel='last_price', ylabel='cityCenters_nearest'>,
        <AxesSubplot:xlabel='floor', ylabel='cityCenters_nearest'>,
        <AxesSubplot:xlabel='cityCenters_nearest', ylabel='cityCenters_nearest'>]],
      dtype=object)
```

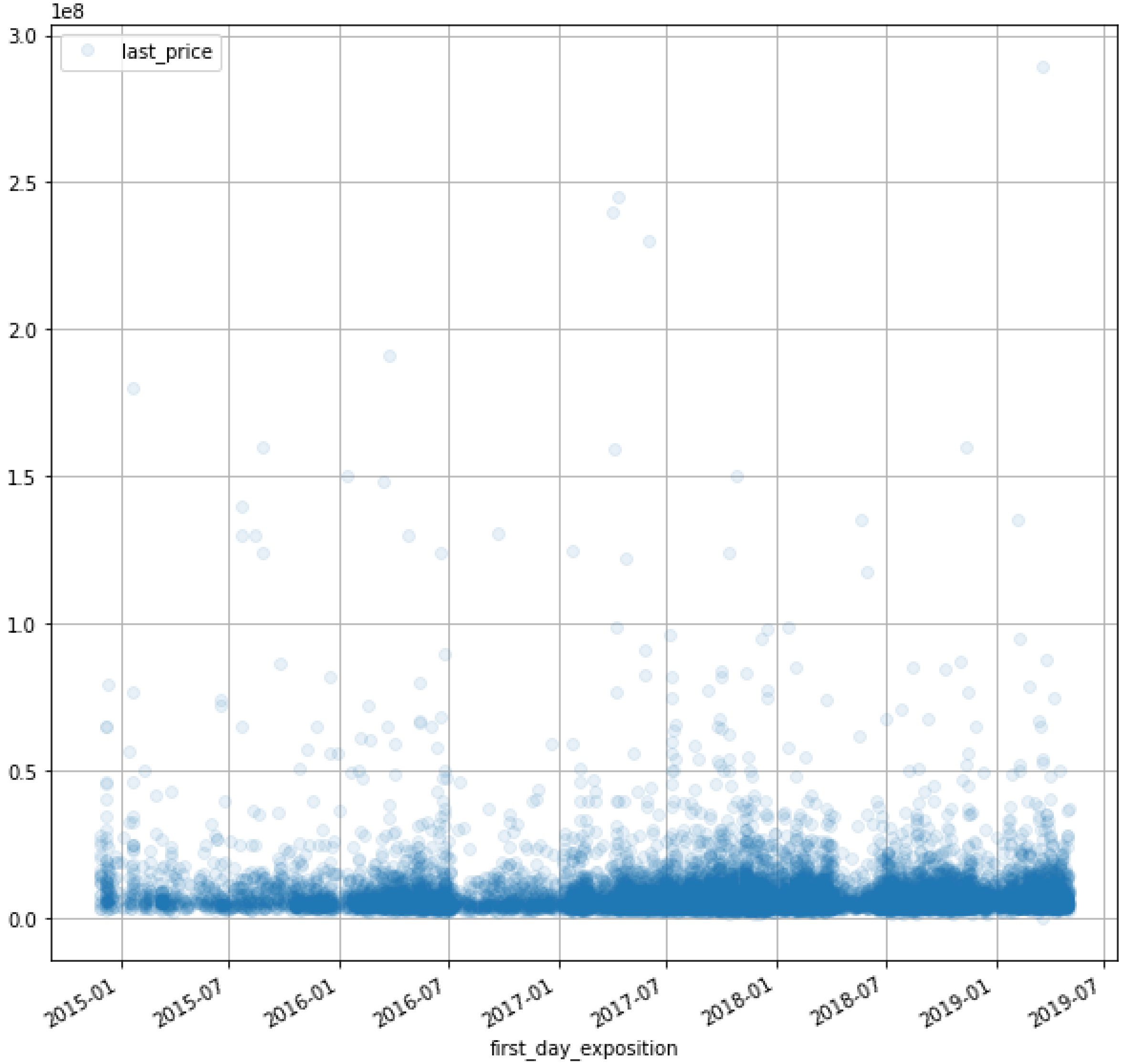


last_price
250000000
200000000
150000000
100000000
50000000
0
floor
30
25
20
15
10
5
30000
25000
20000
15000

In [46]:
```
df_spb.plot(style='o',y='last_price',x='first_day_exposition',grid=True,figsize=(10,10),alpha=0.1)
df_spb[['last_price','exposition_year']].corr()
```

Out[46]:

	last_price	exposition_year
last_price	1.000000	-0.056213
exposition_year	-0.056213	1.000000



Conclusion on the realty price dependency in Saint-Petersburg

1) The highest influence on the realty price is caused by total area, the affect is 72%

2) Second highest parameter is quantity of bedrooms, affect 42%

3) Third parameter ceiling height, affect 33%

4) Fourth is realty floor, affect -1% (negative value)

5) Fifth one is year of publishing, affect - 8% (negative value)

6) Last one is distance to city center , affect - 30% (negative value)

Comparison of price dependency in city center and city overall

The difference between realty price dependency is the the following:

- total area of realty has higher (on 8%) dependency in city overall (72% against 64%).
- quantity of bedrooms also has higher (8%) dependency on the price in city (42% vs 34%).
- ceiling heigh has higher (on 13%) dependency (33% against 20%), most likely due to the fact that most part of realty in city center has high ceilings.
- realty floor has less dependency - on 21% lower in the city overall (-1% against 20%)
- publishing year has less dependency (- 8% vs - 7%)
- distance to city center also has less dependency on 13% (-30% vs -17%)

City center realty price dependency conclusion

1) The highest influence on the realty price in city center is caused by total area, the affect is 64%

2) Second highest parameter is quantity of bedrooms, affect - 34%

3) Third parameter is realty floor, affect 20%

4) Fourth - ceiling height, affect 15%

5) Fifth one is year of publishing, affect -7% (negative value)

6) Last one is distance to city center , affect -17% (negative value)

General Conclusion

1) The higher dependency on the price in data frame has total area (70%), then quantity of bedrooms (40%), floor (5%) and distance to center of the city (5%).

2) Cities with highest quantity of advertisements:

- Санкт-Петербург
- посёлок Мурино
- посёлок Шушары
- Всеволожск
- Колпино
- Пушкин
- посёлок Парголово
- деревня Кудрово
- Гатчина
- Выборг

3) The city center of Saint-Petersburg is considered the area in distance of 3 km from center of the city.

4) Realty in city center has the following parameters with high dependency on the price: total area (64%), then quantity of bedrooms (34%) and floor (20%)

5) In Saint-Petersburg the higher affect on the price has the total area (72%), then quantity of bedrooms (42%) and ceiling height (33%)

6) Total area, quantity of bedrooms and ceiling height have higher affect on the price in Saint-Petersburg, comparing to city center the affect is higher on 8%, 8% and 13% accordingly. Other parameters are losing their affect on the price in city overall comparing to city center area.

*affect percentage specified in brackets