Project_03

September 19, 2021

1 Survey of the advertisements on real estate sales

Data was provided by Yandex realty - archve 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 dataframe has two types of data - insertet by users and automaticly obtained, based on the map information (such as distance to city center, aeroport, closest park, water reservoir.

1.1 Exploration data analysis

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

[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  \
0             20  13000000.0      108.00  2019-03-07T00:00:00      3
1              7   3350000.0       40.40  2018-12-04T00:00:00      1
2             10   5196000.0       56.00  2015-08-20T00:00:00      2
3              0  64900000.0      159.00  2015-07-24T00:00:00      3
4              2  10000000.0      100.00  2018-06-19T00:00:00      2
5             10   2890000.0       30.40  2018-09-10T00:00:00      1
6              6   3700000.0       37.30  2017-11-02T00:00:00      1
7              5   7915000.0       71.60  2019-04-18T00:00:00      2
8             20   2900000.0       33.16  2018-05-23T00:00:00      1
9             18   5400000.0       61.00  2017-02-26T00:00:00      3
10             5   5050000.0       39.60  2017-11-16T00:00:00      1
```

```
11             9   3300000.0       44.00  2018-08-27T00:00:00      2
12            10   3890000.0       54.00  2016-06-30T00:00:00      2
13            20   3550000.0       42.80  2017-07-01T00:00:00      2
14             1   4400000.0       36.00  2016-06-23T00:00:00      1
15            16   4650000.0       39.00  2017-11-18T00:00:00      1
16            11   6700000.0       82.00  2017-11-23T00:00:00      3
17             6   4180000.0       36.00  2016-09-09T00:00:00      1
18             8   3250000.0       31.00  2017-01-27T00:00:00      1
19            16  14200000.0      121.00  2019-01-09T00:00:00      3

    ceiling_height  floors_total  living_area  floor is_apartment  …  \
0             2.70          16.0        51.00      8          NaN  …
1              NaN          11.0        18.60      1          NaN  …
2              NaN           5.0        34.30      4          NaN  …
3              NaN          14.0          NaN      9          NaN  …
4             3.03          14.0        32.00     13          NaN  …
5              NaN          12.0        14.40      5          NaN  …
6              NaN          26.0        10.60      6          NaN  …
7              NaN          24.0          NaN     22          NaN  …
8              NaN          27.0        15.43     26          NaN  …
9             2.50           9.0        43.60      7          NaN  …
10            2.67          12.0        20.30      3          NaN  …
11             NaN           5.0        31.00      4        False  …
12             NaN           5.0        30.00      5          NaN  …
13            2.56           5.0        27.00      5          NaN  …
14             NaN           6.0        17.00      1          NaN  …
15             NaN          14.0        20.50      5          NaN  …
16            3.05           5.0        55.60      1          NaN  …
17             NaN          17.0        16.50      7          NaN  …
18            2.50           5.0        19.40      2          NaN  …
19            2.75          16.0        76.00      8          NaN  …

    kitchen_area  balcony           locality_name  airports_nearest  \
0          25.00      NaN           -                   18863.0
1          11.00      2.0                                12817.0
2           8.30      0.0           -                   21741.0
3            NaN      0.0           -                   28098.0
4          41.00      NaN           -                   31856.0
5           9.10      NaN            -1                  NaN
6          14.40      1.0                               52996.0
7          18.90      2.0           -                   23982.0
8           8.81      NaN                                    NaN
9           6.50      2.0           -                   50898.0
10          8.50      NaN           -                   38357.0
11          6.00      1.0                                  48252.0
12          9.00      0.0                                      NaN
13          5.20      1.0                                   37868.0
14          8.00      0.0                                    20782.0
```

```
15          7.60       1.0                -                12900.0
16          9.00       NaN                -                22108.0
17         11.00       1.0                -                33564.0
18          5.60       1.0                -                44060.0
19         12.00       NaN                -                38900.0

   cityCenters_nearest  parks_around3000  parks_nearest  ponds_around3000  \
0              16028.0               1.0          482.0               2.0
1              18603.0               0.0            NaN               0.0
2              13933.0               1.0           90.0               2.0
3               6800.0               2.0           84.0               3.0
4               8098.0               2.0          112.0               1.0
5                  NaN               NaN            NaN               NaN
6              19143.0               0.0            NaN               0.0
7              11634.0               0.0            NaN               0.0
8                  NaN               NaN            NaN               NaN
9              15008.0               0.0            NaN               0.0
10             13878.0               1.0          310.0               2.0
11             51677.0               0.0            NaN               0.0
12                 NaN               NaN            NaN               NaN
13             33058.0               1.0          294.0               3.0
14             30759.0               0.0            NaN               1.0
15             14259.0               1.0          590.0               1.0
16             10698.0               3.0          420.0               0.0
17             14616.0               0.0            NaN               1.0
18             10842.0               1.0          759.0               0.0
19             12843.0               0.0            NaN               0.0

    ponds_nearest  days_exposition
0           755.0              NaN
1             NaN             81.0
2           574.0            558.0
3           234.0            424.0
4            48.0            121.0
5             NaN             55.0
6             NaN            155.0
7             NaN              NaN
8             NaN            189.0
9             NaN            289.0
10          553.0            137.0
11            NaN              7.0
12            NaN             90.0
13          298.0            366.0
14           96.0            203.0
15          296.0             19.0
16            NaN            397.0
17          859.0            571.0
18            NaN            168.0
```

```
19             NaN             97.0

[20 rows x 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)
memory usage: 3.7+ MB
```

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

```
[3]: locality_name
                                      3
                                        9
                                       23
                                    100
                                       62
                                        …
                              1
                                      1
                                      1
            -                       1
                                        1
```

```
Name: rooms, Length: 199, dtype: int64
```

1.1.1 Conclusion

Based on the preliminary analysis it's possible to conclude the following: 1. Dataframe has 23 699 rows 22 columns; 2. Dataframe contains infromation re: apartments area, city, cost and etc.; 3. A lot of columns has null values, so it's required to analyse such columns and fill up the nulls.

1.2 Data preparation

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

1.2.1 Nulls processing in column "ceiling_height"

[4]:
```
# display the unqique 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]
```

[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 cocnlude that height of ceiling is less than 5 meters but data also contains the anomalies such as 14.25 and 100 m., etc

[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()
```

[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
```

1.2.2 Nulls processing in column "floors_total"

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

[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
```

[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
```

1.2.3 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 coefficicent of median living area to median total area will be applied for calculation.

[7]:

```python
# 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','      ',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','                      : ', 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

        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

                        :  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
```

1.2.4 Nulls processing in column "is_apartment"

[8]:
```
# 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()
```

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

1.2.5 Nulls processing in column"kitchen_area"

[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','                      : ', 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

                       :   0
```

1.2.6 Nulls processing in column "balcony"

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

[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
```

1.2.7 Nulls processing in column "locality_name"

```
[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'].fillna('   -     ')

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

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

```
locality_name
     -             208.0
                 18006.0
                  22589.0
                      24311.0
                    28266.0
                     29815.0
                      30438.0
                     31533.0
                    33605.0
                     46657.0
                    52628.0
                     52768.0
Name: cityCenters_nearest, dtype: float64

 locality_name
                   7
               10
                   14
                  15
                15
                   19
                    61
                   69
                    74
                   103
                     113
   -           3606
Name: cityCenters_nearest, dtype: int64

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

[11]:
```
count                 23699
unique                  365
top          -
freq                  15721
Name: locality_name, dtype: object
```

1.2.8 Processing of remaining nulls in remaining columns

[12]:
```python
# selection of columns
columns_to_fill =␣
 ↪['airports_nearest','cityCenters_nearest','parks_around3000','parks_nearest','ponds_around3

# 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
```

1.2.9 Changing of data types

[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
```

[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')
```

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

[15]:
```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0           2019-03-07      3
1             7   3350000.0        40.4           2018-12-04      1
2            10   5196000.0        56.0           2015-08-20      2
3             0  64900000.0       159.0           2015-07-24      3
4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  is_apartment  …  \
0            2.70            16        51.00      8         False  …
1            2.60            11        18.60      1         False  …
2            2.60             5        34.30      4         False  …
3            2.95            14        45.76      9         False  …
4            3.03            14        32.00     13         False  …

   balcony    locality_name  airports_nearest  cityCenters_nearest  \
0        0     -                      18863                16028
1        2                             12817                18603
2        0     -                      21741                13933
3        0     -                      28098                 6800
4        0     -                      31856                 8098

  parks_around3000  parks_nearest  ponds_around3000  ponds_nearest  \
0                 1           482                 2            755
1                 0            -1                 0             -1
2                 1            90                 2            574
3                 2            84                 3            234
4                 2           112                 1             48

   days_exposition  floor_type
0               -1        16.0
1               81        11.0
2              558         5.0
3              424        14.0
4              121        14.0

[5 rows x 23 columns]
```

[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
```

1.2.10 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
- oher columns with zero.

3) changes in datatypes:

- Data with integer values were change to int
- apartment column to bool
- date of exposition to dadatime
- other float data were uncchanged 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 havn't got precise infomration.

1.3 Calculation and adding the relevant information to dataset

1.3.1 Calculation of cost per square meter

[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()
```

[17]:
```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0           2019-03-07      3
1             7   3350000.0        40.4           2018-12-04      1
2            10   5196000.0        56.0           2015-08-20      2
3             0  64900000.0       159.0           2015-07-24      3
4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  is_apartment  …  \
0            2.70            16        51.00      8         False  …
1            2.60            11        18.60      1         False  …
2            2.60             5        34.30      4         False  …
3            2.95            14        45.76      9         False  …
4            3.03            14        32.00     13         False  …

     locality_name  airports_nearest  cityCenters_nearest  parks_around3000  \
0    -                        18863                16028                 1
1                              12817                18603                 0
2    -                        21741                13933                 1
3    -                        28098                 6800                 2
4    -                        31856                 8098                 2

  parks_nearest  ponds_around3000  ponds_nearest  days_exposition  floor_type  \
0           482                 2            755               -1        16.0
1            -1                 0             -1               81        11.0
2            90                 2            574              558         5.0
3            84                 3            234              424        14.0
4           112                 1             48              121        14.0

   price_per_meter
0        120370.37
1         82920.79
2         92785.71
3        408176.10
4        100000.00

[5 rows x 24 columns]
```

1.3.2 New columns with year, month and day of exposition

[18]:
```python
# 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()
```

[18]:
```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0           2019-03-07      3
1             7   3350000.0        40.4           2018-12-04      1
2            10   5196000.0        56.0           2015-08-20      2
3             0  64900000.0       159.0           2015-07-24      3
4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  is_apartment  …  \
0            2.70            16        51.00      8         False  …
1            2.60            11        18.60      1         False  …
2            2.60             5        34.30      4         False  …
3            2.95            14        45.76      9         False  …
4            3.03            14        32.00     13         False  …

   parks_around3000  parks_nearest  ponds_around3000  ponds_nearest  \
0                 1            482                 2            755
1                 0             -1                 0             -1
2                 1             90                 2            574
3                 2             84                 3            234
4                 2            112                 1             48

  days_exposition  floor_type  price_per_meter  exposition_year  \
0              -1        16.0        120370.37             2019
1              81        11.0         82920.79             2018
2             558         5.0         92785.71             2015
3             424        14.0        408176.10             2015
4             121        14.0        100000.00             2018

   exposition_month  exposition_weekday
0                 3                   3
1                12                   1
2                 8                   3
3                 7                   4
4                 6                   1

[5 rows x 27 columns]
```

1.3.3 Definition of floor of realty

[19]:
```python
# 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)
```

```
[19]:     total_images  last_price  total_area first_day_exposition  rooms  \
     0              20  13000000.0      108.00           2019-03-07      3
     1               7   3350000.0       40.40           2018-12-04      1
     2              10   5196000.0       56.00           2015-08-20      2
     3               0  64900000.0      159.00           2015-07-24      3
     4               2  10000000.0      100.00           2018-06-19      2
     5              10   2890000.0       30.40           2018-09-10      1
     6               6   3700000.0       37.30           2017-11-02      1
     7               5   7915000.0       71.60           2019-04-18      2
     8              20   2900000.0       33.16           2018-05-23      1
     9              18   5400000.0       61.00           2017-02-26      3
     10              5   5050000.0       39.60           2017-11-16      1
     11              9   3300000.0       44.00           2018-08-27      2
     12             10   3890000.0       54.00           2016-06-30      2
     13             20   3550000.0       42.80           2017-07-01      2
     14              1   4400000.0       36.00           2016-06-23      1
     15             16   4650000.0       39.00           2017-11-18      1
     16             11   6700000.0       82.00           2017-11-23      3
     17              6   4180000.0       36.00           2016-09-09      1
     18              8   3250000.0       31.00           2017-01-27      1
     19             16  14200000.0      121.00           2019-01-09      3

         ceiling_height  floors_total  living_area  floor  is_apartment  …  \
     0             2.70            16        51.00      8         False  …
     1             2.60            11        18.60      1         False  …
     2             2.60             5        34.30      4         False  …
     3             2.95            14        45.76      9         False  …
     4             3.03            14        32.00     13         False  …
     5              NaN            12        14.40      5         False  …
     6             2.60            26        10.60      6         False  …
     7             2.60            24        31.00     22         False  …
     8              NaN            27        15.43     26         False  …
     9             2.50             9        43.60      7         False  …
     10            2.67            12        20.30      3         False  …
     11            2.60             5        31.00      4         False  …
     12             NaN             5        30.00      5         False  …
     13            2.56             5        27.00      5         False  …
     14            2.60             6        17.00      1         False  …
     15            2.60            14        20.50      5         False  …
     16            3.05             5        55.60      1         False  …
```

```
17            2.60            17        16.50      7         False  …
18            2.50             5        19.40      2         False  …
19            2.75            16        76.00      8         False  …

   parks_around3000  parks_nearest  ponds_around3000  ponds_nearest  \
0                 1            482                 2            755
1                 0             -1                 0             -1
2                 1             90                 2            574
3                 2             84                 3            234
4                 2            112                 1             48
5                -1             -1                -1             -1
6                 0             -1                 0             -1
7                 0             -1                 0             -1
8                -1             -1                -1             -1
9                 0             -1                 0             -1
10                1            310                 2            553
11                0             -1                 0             -1
12               -1             -1                -1             -1
13                1            294                 3            298
14                0             -1                 1             96
15                1            590                 1            296
16                3            420                 0             -1
17                0             -1                 1            859
18                1            759                 0             -1
19                0             -1                 0             -1

  days_exposition   floor_type  price_per_meter  exposition_year  \
0              -1        other        120370.37             2019
1              81  first_floor         82920.79             2018
2             558        other         92785.71             2015
3             424        other        408176.10             2015
4             121        other        100000.00             2018
5              55        other         95065.79             2018
6             155        other         99195.71             2017
7              -1        other        110544.69             2019
8             189        other         87454.76             2018
9             289        other         88524.59             2017
10            137        other        127525.25             2017
11              7        other         75000.00             2018
12             90   last_floor         72037.04             2016
13            366   last_floor         82943.93             2017
14            203  first_floor        122222.22             2016
15             19        other        119230.77             2017
16            397  first_floor         81707.32             2017
17            571        other        116111.11             2016
18            168        other        104838.71             2017
19             97        other        117355.37             2019
```

```
    exposition_month  exposition_weekday
0                  3                   3
1                 12                   1
2                  8                   3
3                  7                   4
4                  6                   1
5                  9                   0
6                 11                   3
7                  4                   3
8                  5                   2
9                  2                   6
10                11                   3
11                 8                   0
12                 6                   3
13                 7                   5
14                 6                   3
15                11                   5
16                11                   3
17                 9                   4
18                 1                   4
19                 1                   2

[20 rows x 27 columns]
```

1.3.4 Calculation of proportion of realty areas

[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()
```

[20]:
```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0           2019-03-07      3
1             7   3350000.0        40.4           2018-12-04      1
2            10   5196000.0        56.0           2015-08-20      2
3             0  64900000.0       159.0           2015-07-24      3
4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  is_apartment  …  \
0            2.70            16        51.00      8         False  …
1            2.60            11        18.60      1         False  …
2            2.60             5        34.30      4         False  …
3            2.95            14        45.76      9         False  …
4            3.03            14        32.00     13         False  …
```

```
   parks_nearest  ponds_around3000  ponds_nearest  days_exposition  \
0            482                 2            755               -1
1             -1                 0             -1               81
2             90                 2            574              558
3             84                 3            234              424
4            112                 1             48              121

    floor_type  price_per_meter  exposition_year  exposition_month  \
0        other        120370.37             2019                 3
1  first_floor         82920.79             2018                12
2        other         92785.71             2015                 8
3        other        408176.10             2015                 7
4        other        100000.00             2018                 6

   exposition_weekday  living_to_total_percent
0                   3                     0.47
1                   1                     0.46
2                   3                     0.61
3                   4                     0.29
4                   1                     0.32

[5 rows x 28 columns]
```

[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()
```

[21]:
```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0           2019-03-07      3
1             7   3350000.0        40.4           2018-12-04      1
2            10   5196000.0        56.0           2015-08-20      2
3             0  64900000.0       159.0           2015-07-24      3
4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  is_apartment  …  \
0            2.70            16        51.00      8         False  …
1            2.60            11        18.60      1         False  …
2            2.60             5        34.30      4         False  …
3            2.95            14        45.76      9         False  …
4            3.03            14        32.00     13         False  …

   ponds_around3000  ponds_nearest  days_exposition   floor_type  \
0                 2            755               -1        other
```

```
1                  0          -1                81  first_floor
2                  2         574               558        other
3                  3         234               424        other
4                  1          48               121        other

  price_per_meter  exposition_year  exposition_month  exposition_weekday  \
0       120370.37             2019                 3                   3
1        82920.79             2018                12                   1
2        92785.71             2015                 8                   3
3       408176.10             2015                 7                   4
4       100000.00             2018                 6                   1

   living_to_total_percent  kitchen_to_total_percent
0                     0.47                      0.23
1                     0.46                      0.27
2                     0.61                      0.15
3                     0.29                      0.08
4                     0.32                      0.41

[5 rows x 29 columns]
```

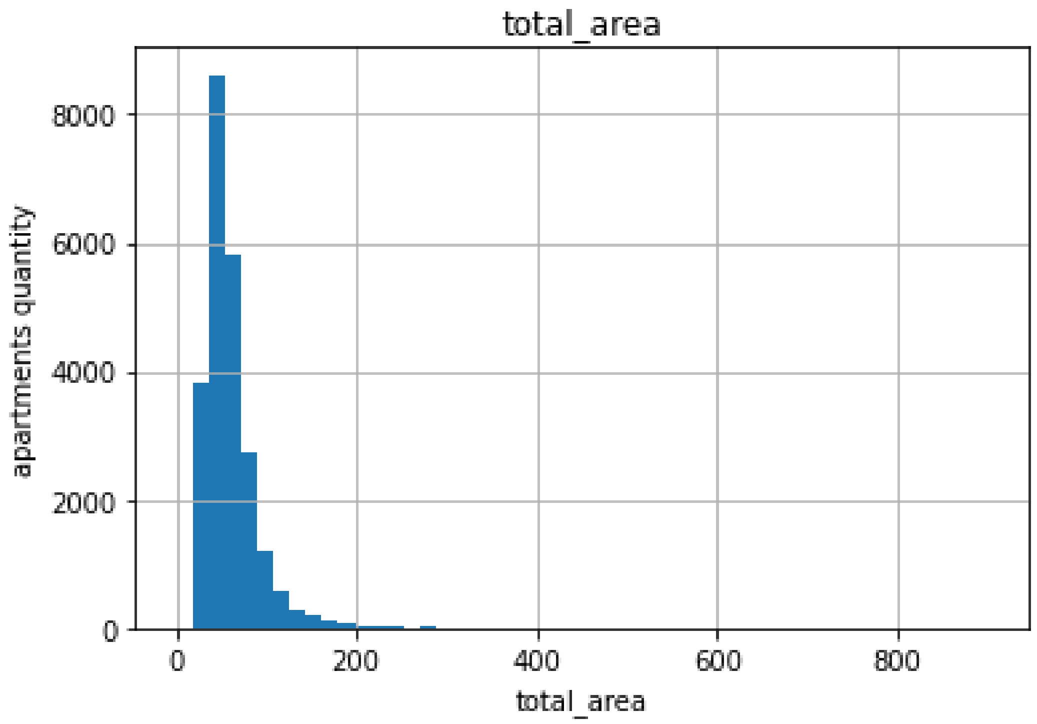
1.4 Statistical data analysis

1.4.1 Histogram plotting

Histogram of total area values

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

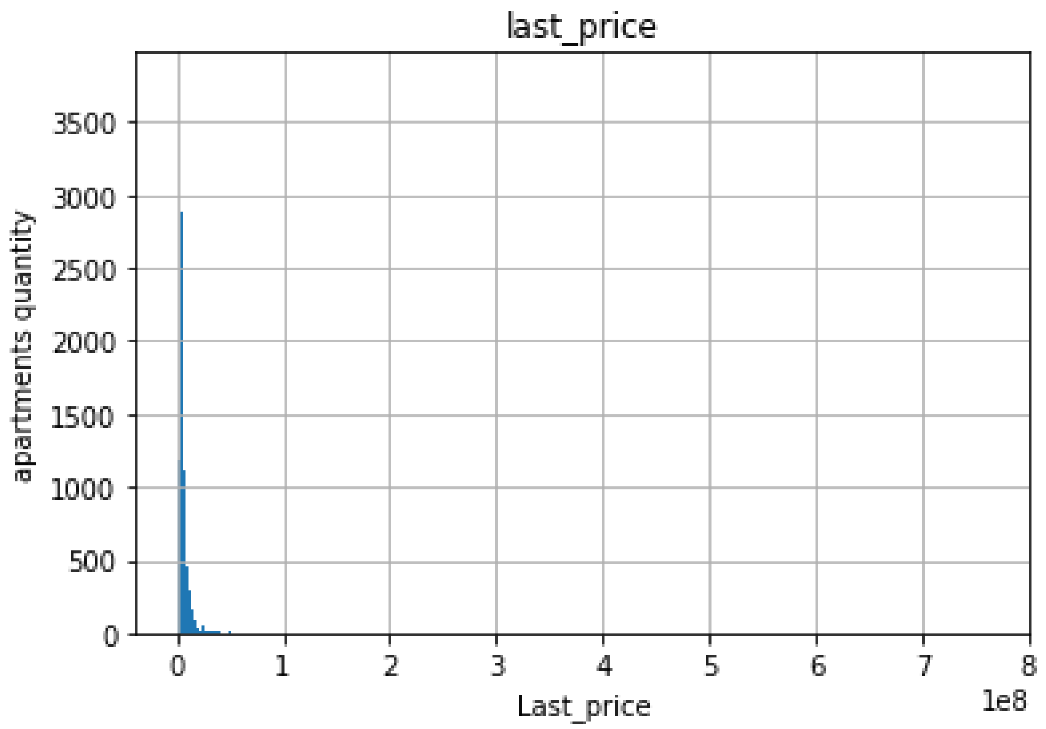
[22]:
```
Text(0, 0.5, 'apartments quantity')
```



Histogram of prices

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

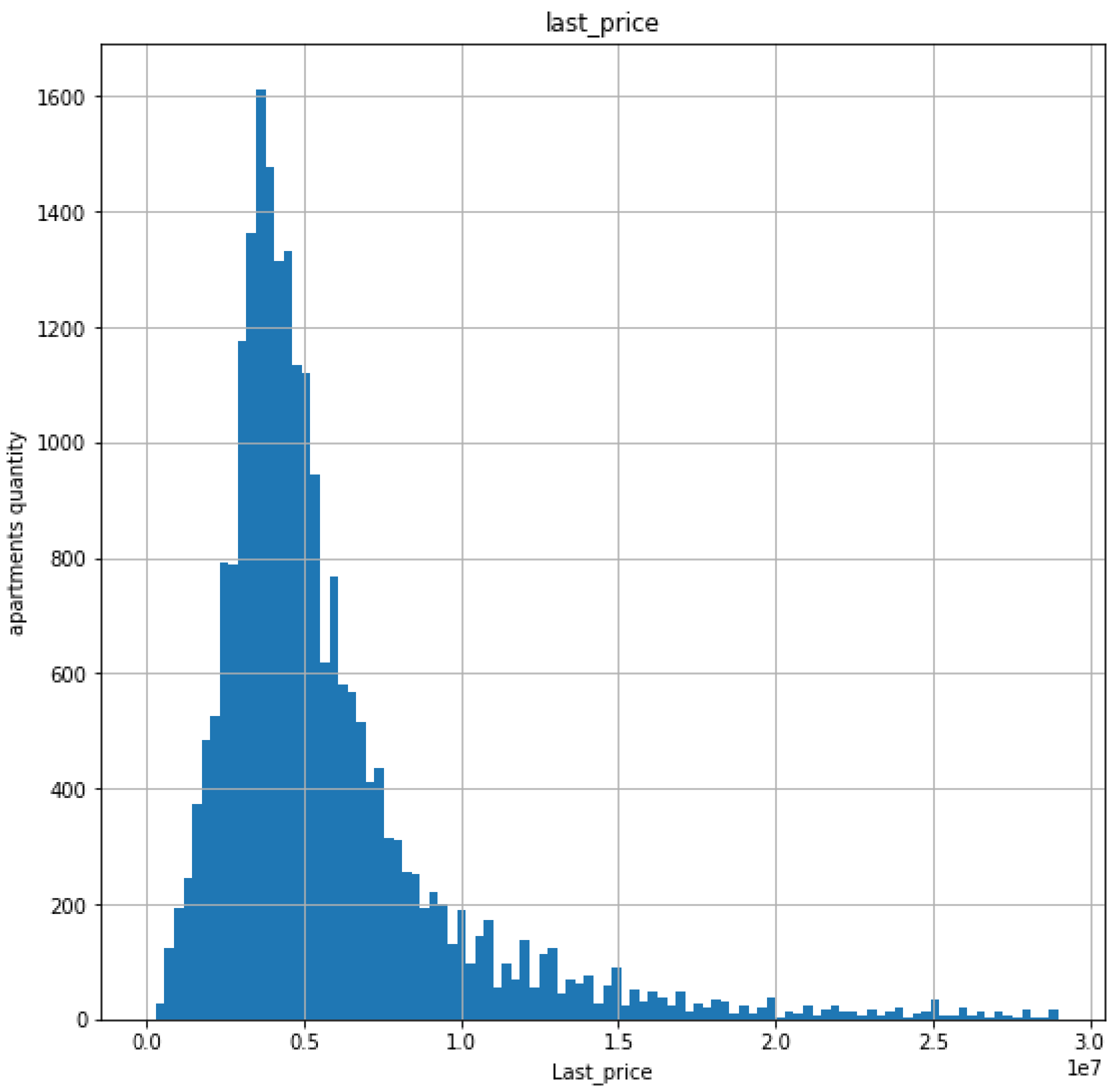
```
[23]: Text(0, 0.5, 'apartments quantity')
```



Rescaling

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

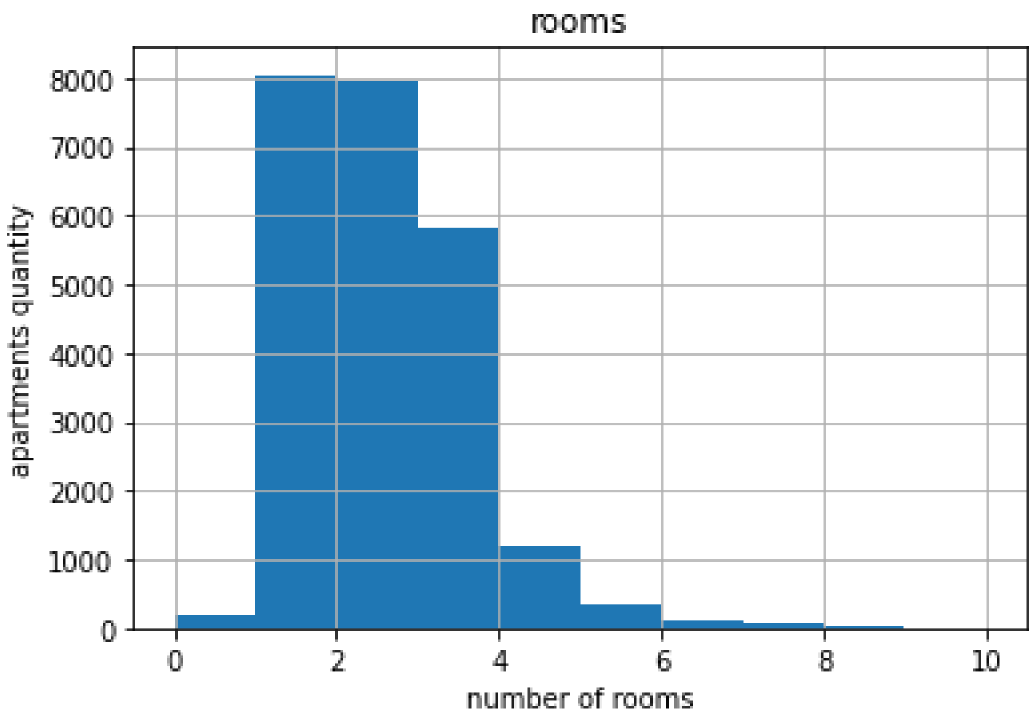
```
[24]: Text(0, 0.5, 'apartments quantity')
```



Histogram of room quantity

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

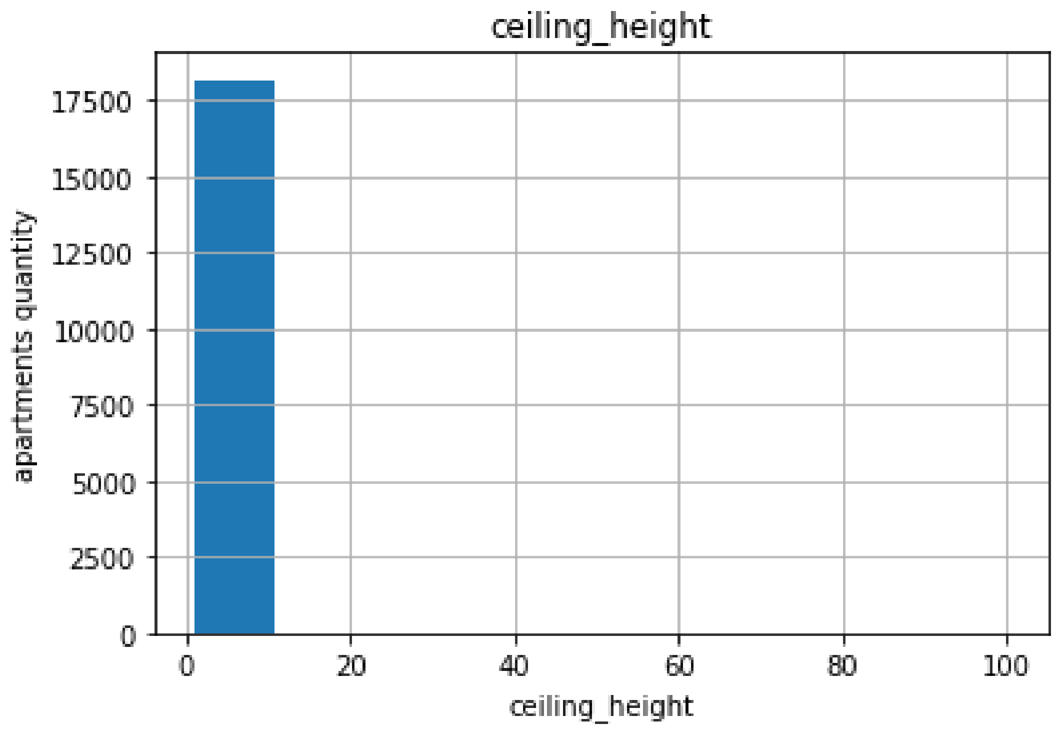
```
[25]: Text(0, 0.5, 'apartments quantity')
```



Ceiling height histogram plotting

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

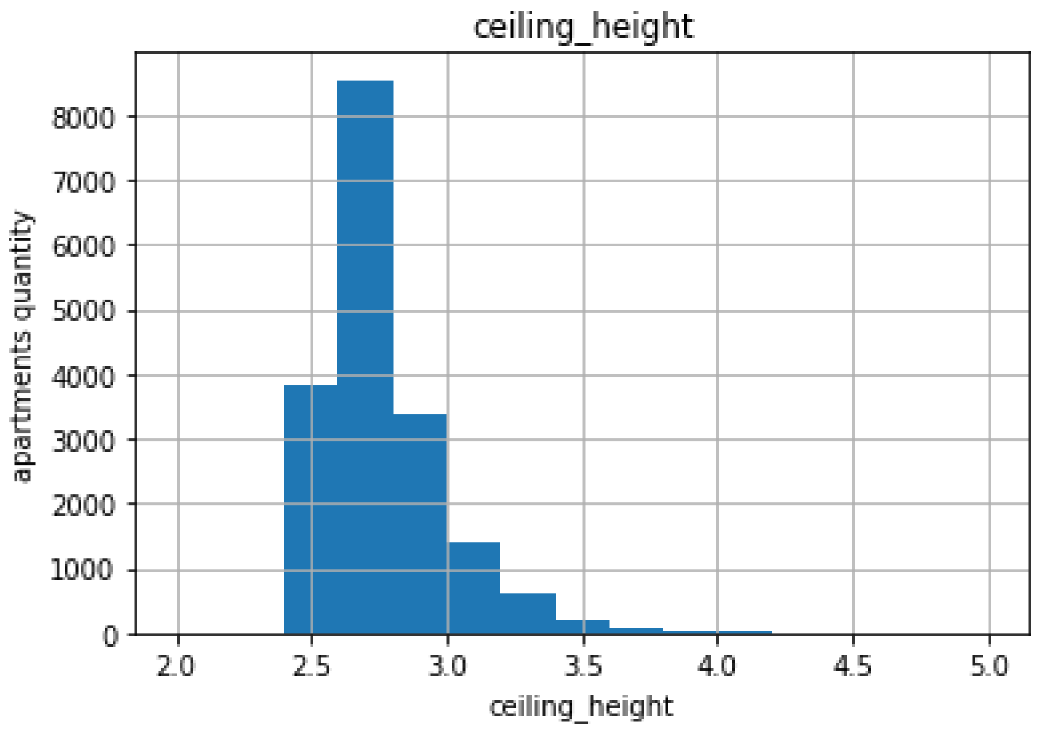
```
[26]: Text(0, 0.5, 'apartments quantity')
```



Rescaling

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

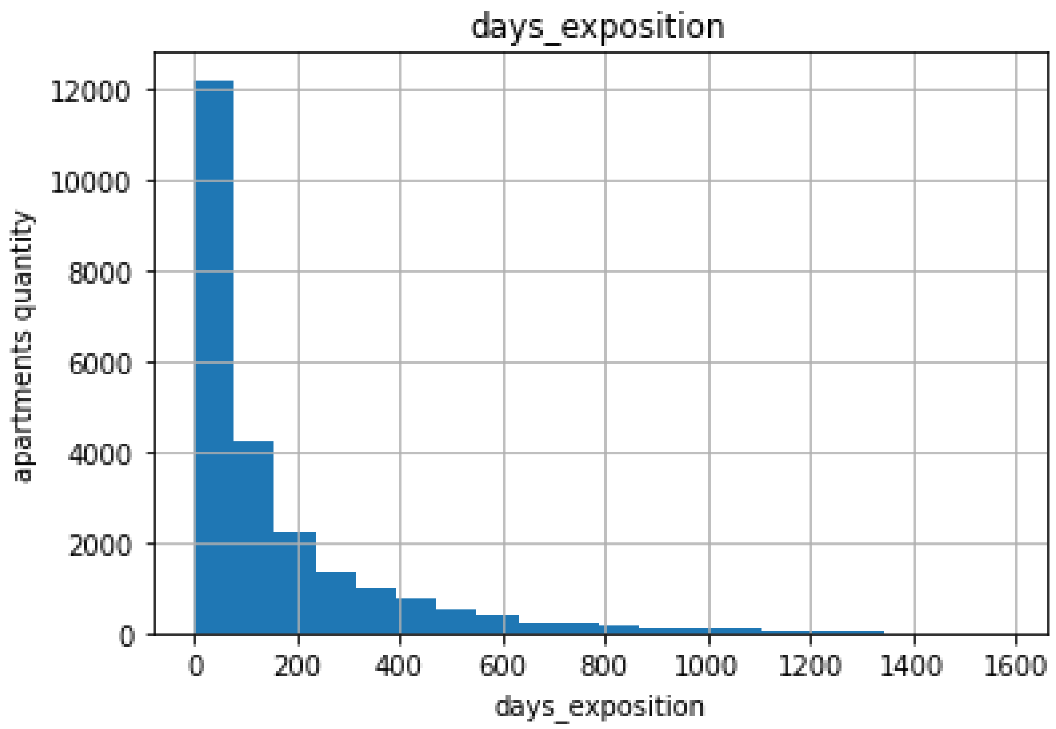
```
[27]: Text(0, 0.5, 'apartments quantity')
```



Advertisement duration histogram

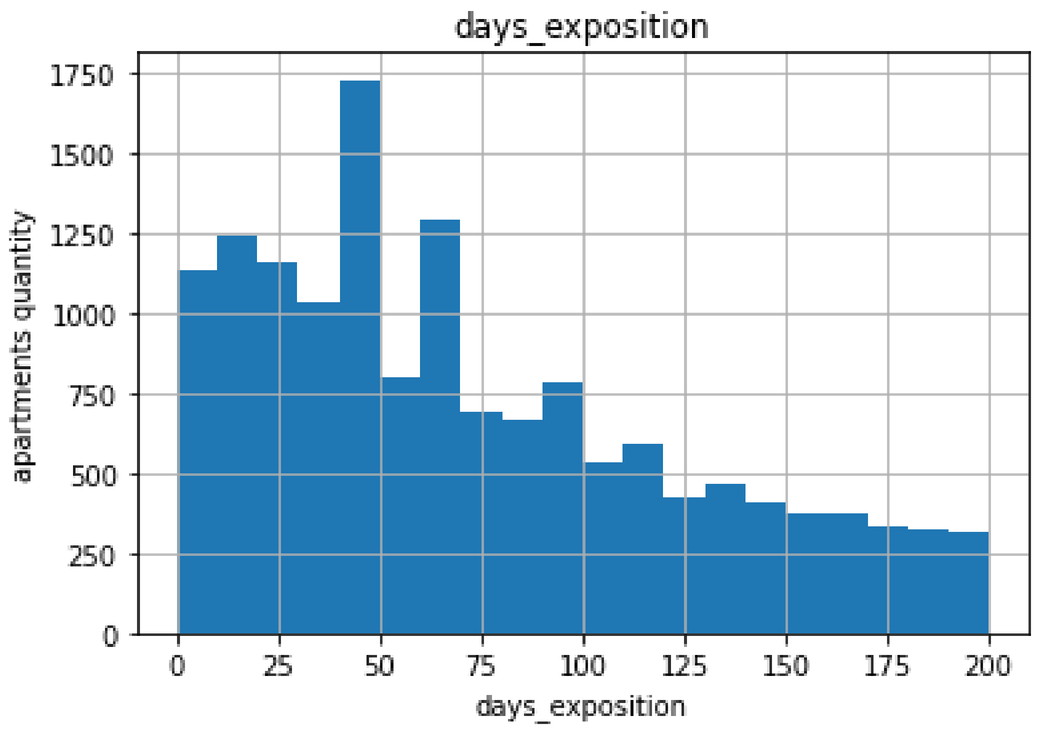
```
[28]: df_aparts.hist(column = 'days_exposition', bins=20)
      pl.xlabel("days_exposition")
      pl.ylabel("apartments quantity")

      df_aparts.days_exposition.describe()
```

```
[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
```



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

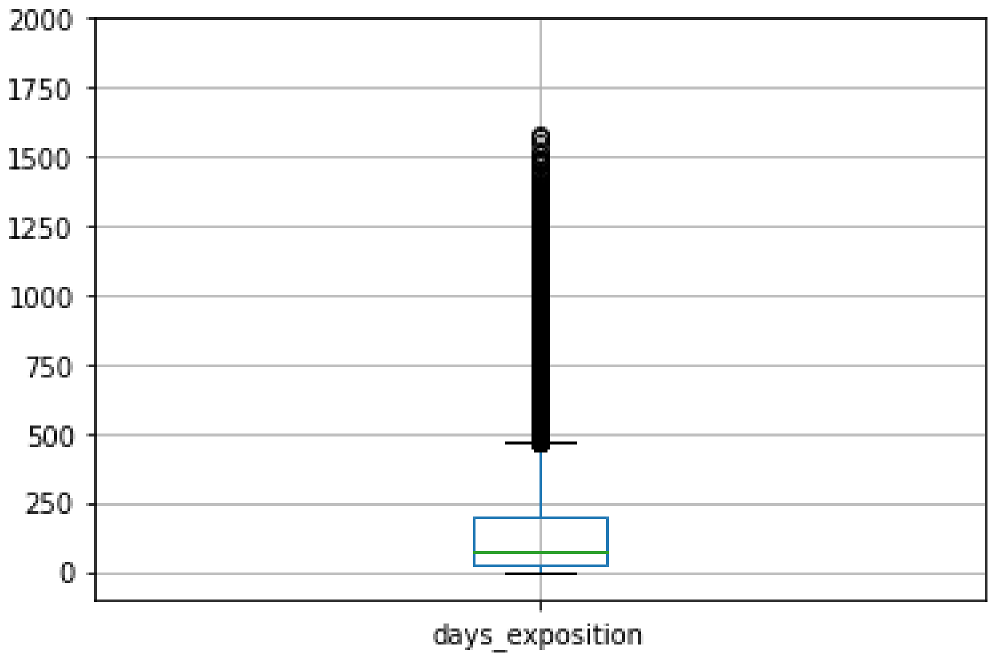
```
[29]: Text(0, 0.5, 'apartments quantity')
```



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

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

[30]:
```
<AxesSubplot:>
```



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

1.4.2 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

[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
```

[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

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

Deletion of ralty with huge total area

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

Deletion of overpriced realty

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

1.4.3 Analysis of parameter which infuence on the realty price

[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)
```

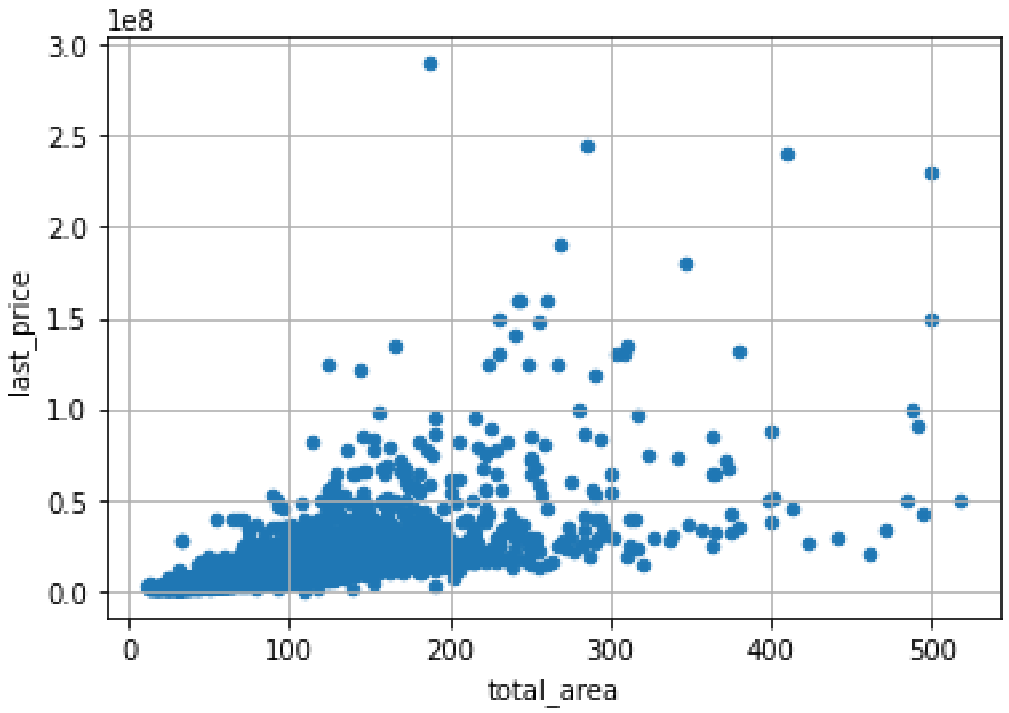
```
# adding new column with floor category
df_2['floor_type_key'] = df_2.apply(floor_func,axis=1)
df_2.head()
```

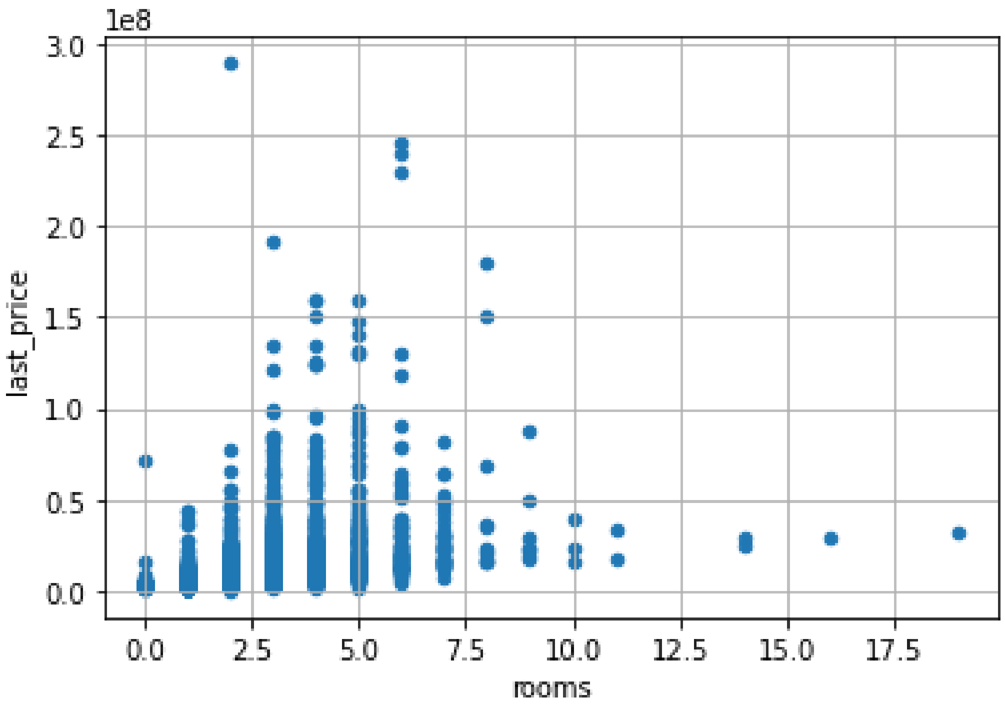
[36]:
```
   index  total_images  last_price  total_area first_day_exposition  rooms  \
0      0            20  13000000.0       108.0           2019-03-07      3
1      1             7   3350000.0        40.4           2018-12-04      1
2      2            10   5196000.0        56.0           2015-08-20      2
3      3             0  64900000.0       159.0           2015-07-24      3
4      4             2  10000000.0       100.0           2018-06-19      2

   ceiling_height  floors_total  living_area  floor  …  ponds_nearest  \
0            2.70            16        51.00      8  …            755
1            2.60            11        18.60      1  …             -1
2            2.60             5        34.30      4  …            574
3            2.95            14        45.76      9  …            234
4            3.03            14        32.00     13  …             48

   days_exposition   floor_type  price_per_meter  exposition_year  \
0               -1        other        120370.37             2019
1               81  first_floor         82920.79             2018
2              558        other         92785.71             2015
3              424        other        408176.10             2015
4              121        other        100000.00             2018

  exposition_month  exposition_weekday  living_to_total_percent  \
0                3                   3                     0.47
1               12                   1                     0.46
2                8                   3                     0.61
3                7                   4                     0.29
4                6                   1                     0.32

   kitchen_to_total_percent  floor_type_key
0                      0.23               1
1                      0.27               0
2                      0.15               1
3                      0.08               1
4                      0.41               1

[5 rows x 31 columns]
```

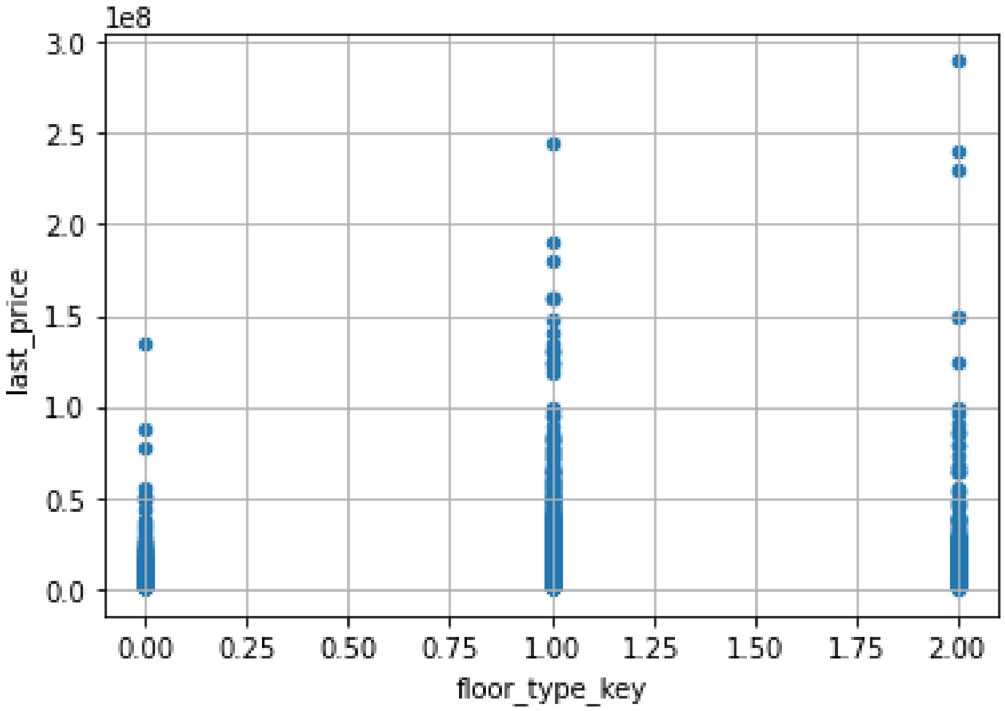
[37]:
```
# selection of columns with highest affect on the price
data_list =␣
 ↪['total_area','rooms','floor_type_key','cityCenters_nearest','exposition_year','exposition_

# cycle for plottin of histagram of correlation of columns values to the price
```

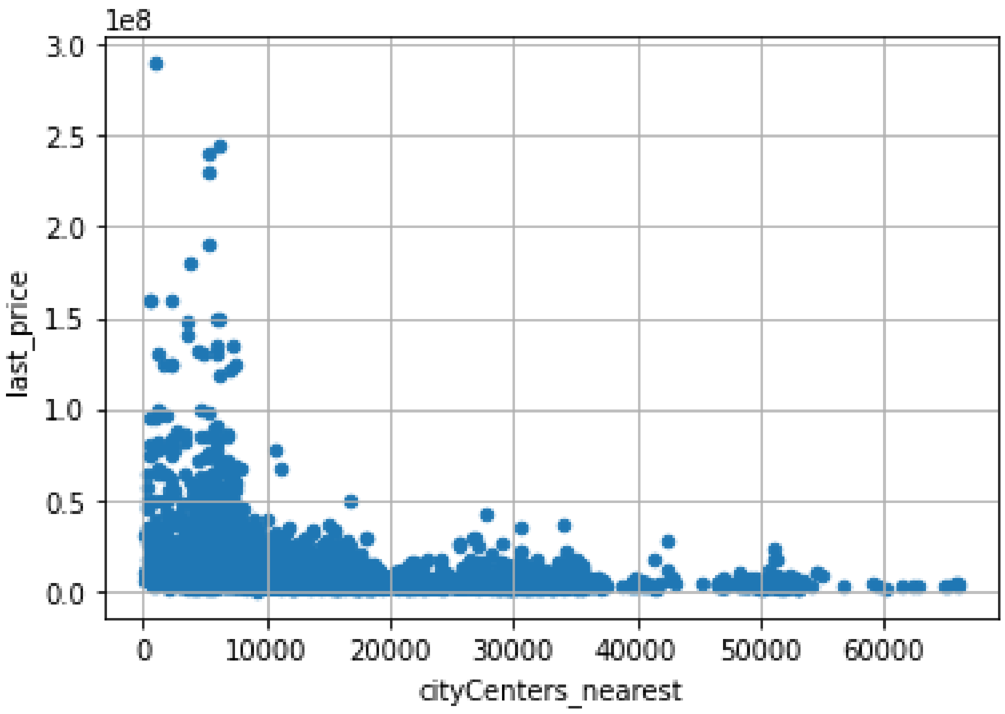
```
for data in data_list:
    df_2.plot(y='last_price', x = data, kind = 'scatter', grid=True)
    print(data, 'coeff:', round(df_2['last_price'].corr(df_2[data]),5))
```

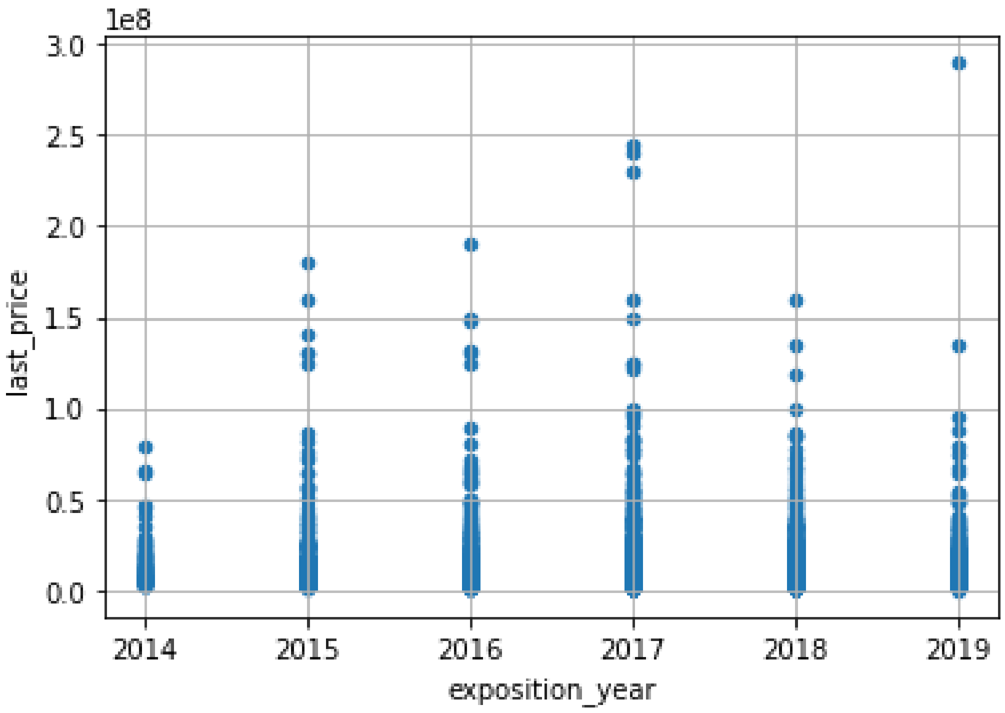
```
total_area coeff: 0.70914
rooms coeff: 0.42109
floor_type_key coeff: 0.06508
cityCenters_nearest coeff: -0.25546
exposition_year coeff: -0.05197
exposition_month coeff: -0.0035
exposition_weekday coeff: -0.0003
```

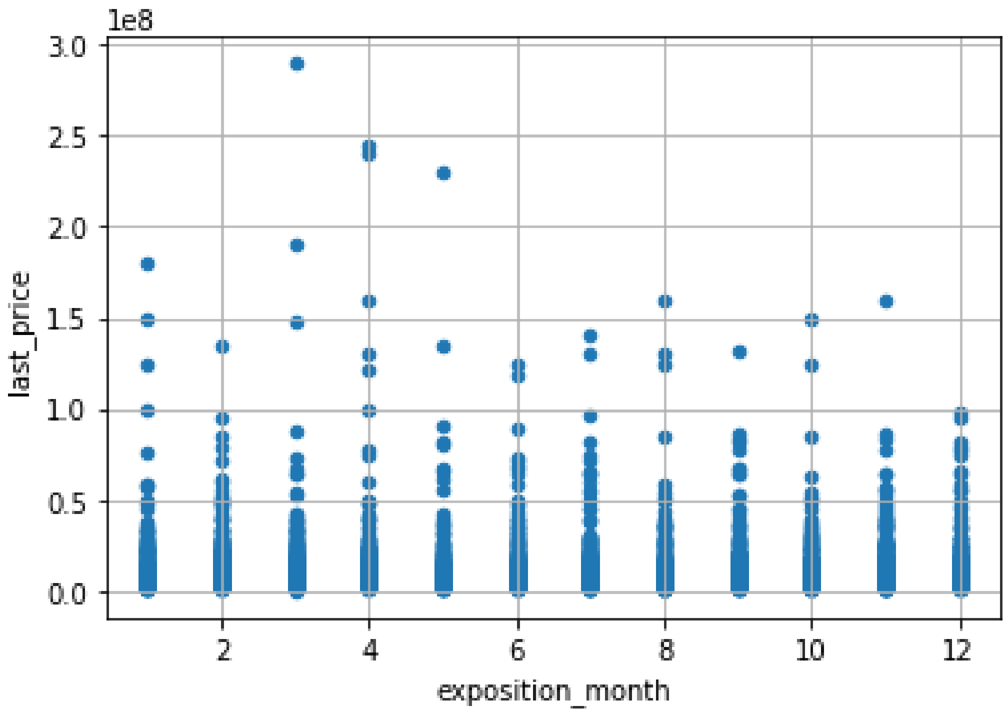


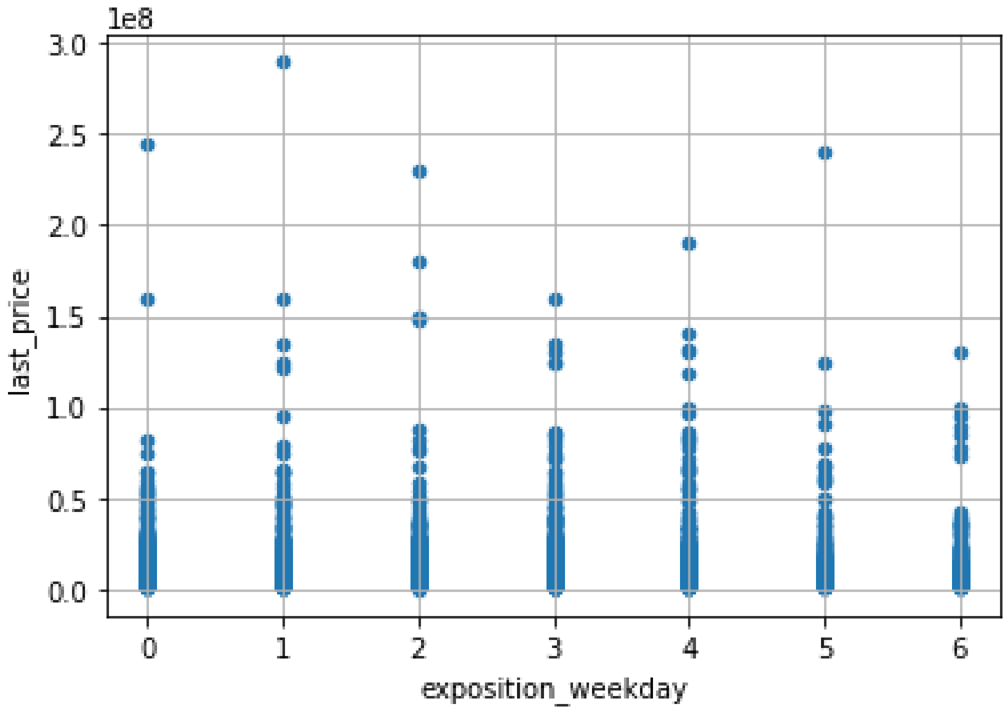

1e8
last_price
rooms
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


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


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 publishinh has negative dependence (-5%)
- Year of publishing has also negative dependence (-8%)

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

[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
```

[38]:
```
locality_name
    -          114180.07
                   103070.37
               90175.91
                 78474.36
                     75402.50
Name: price_per_meter, dtype: float64
```

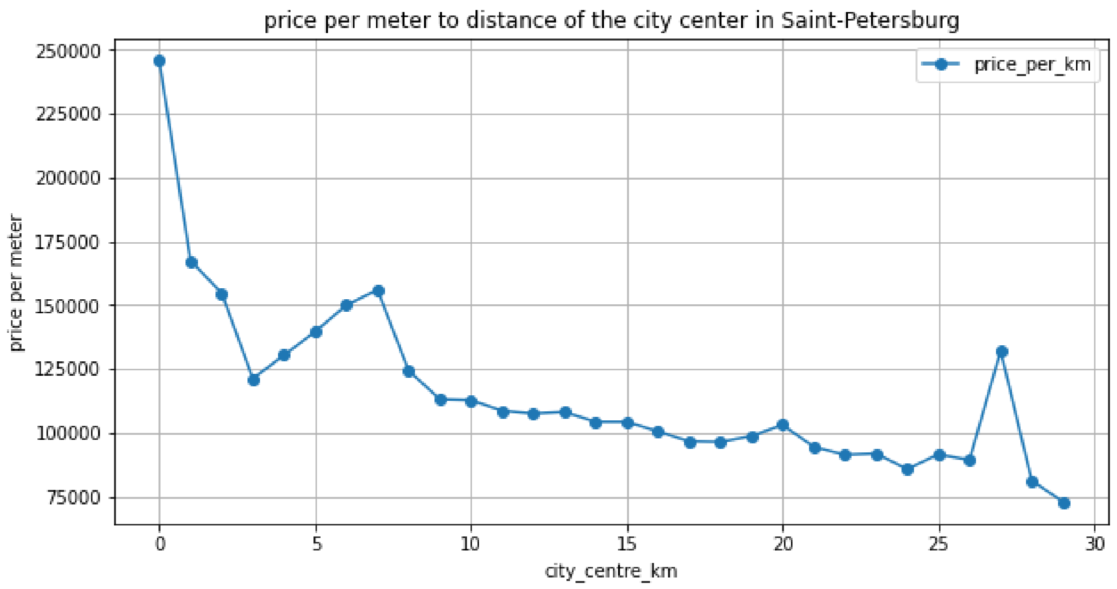
Maximum quantity of advertisements were placed from the folowing cities:

- -
-
-
-
-

1.4.5 Definition of apartments price in citycenter of Saint-Petersburg

[39]:
```
df_spb = df_2.query('locality_name in "   -     " 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")
```

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

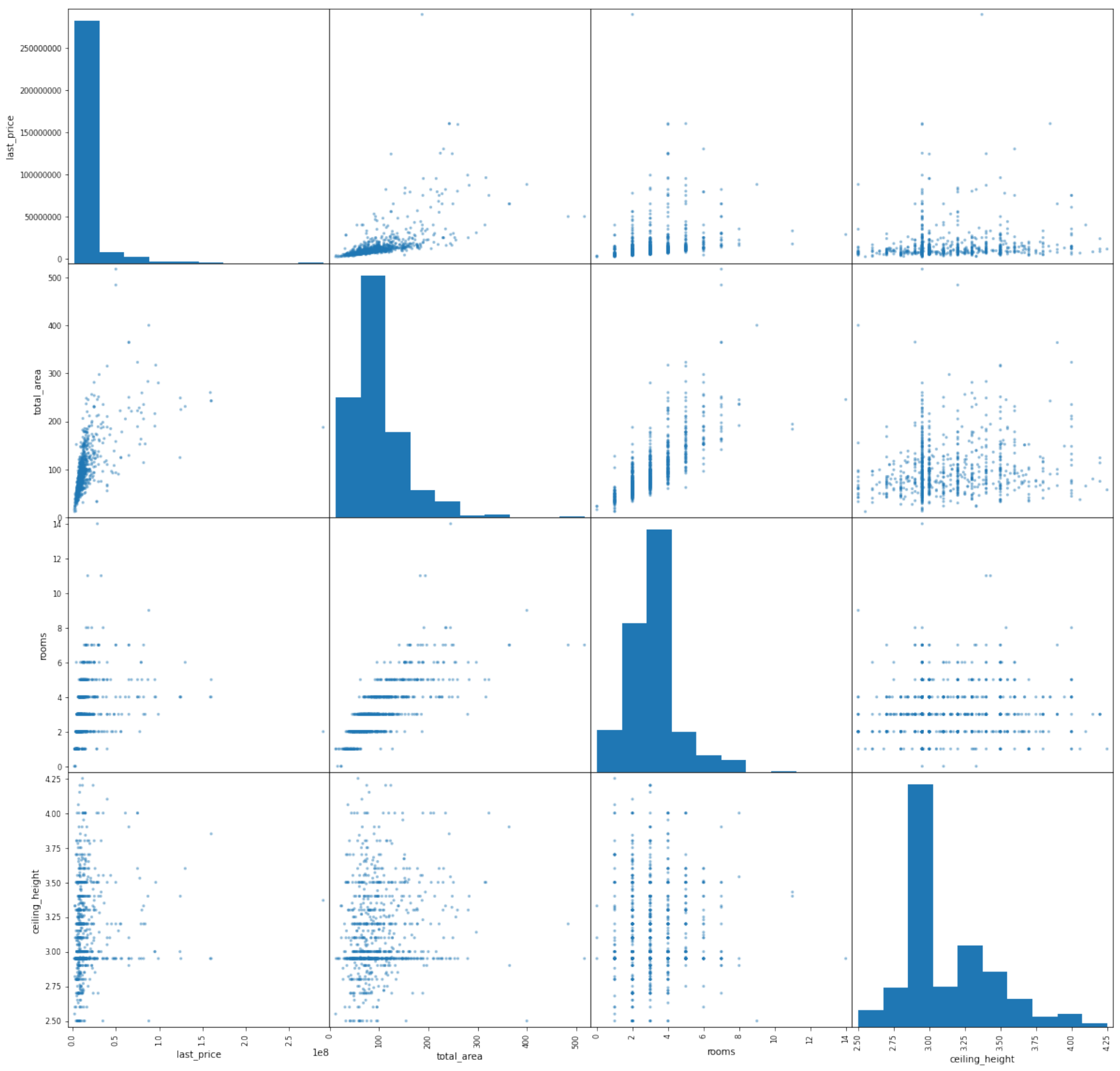


Based on the information from graph we assume that ralty in citycenter is with distance equal to 3 km or less

1.4.6 Analysis of parameters of reallty in citycenter

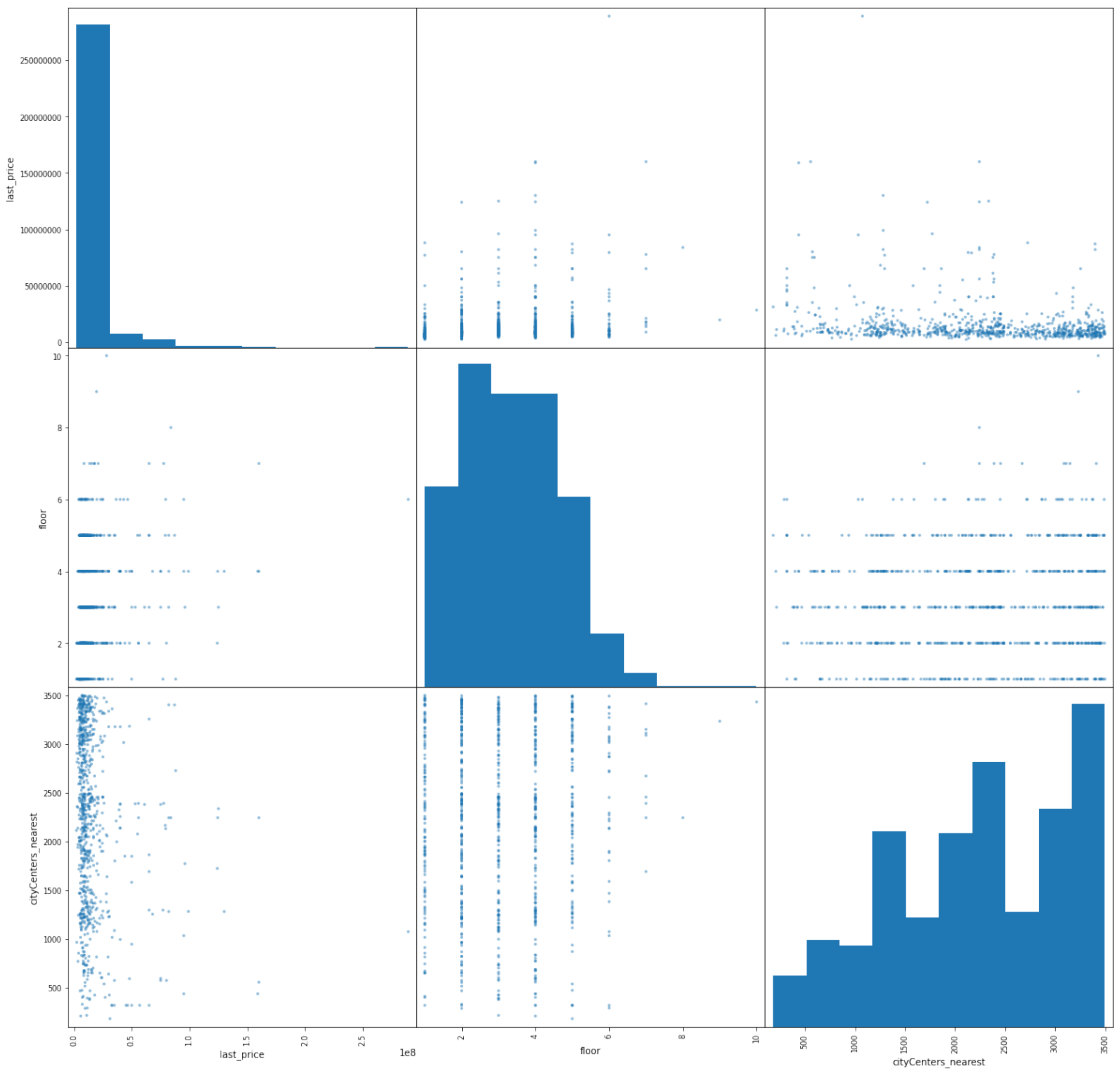
[40]:
```
df_spb_centre = df_spb.query('city_centre_km <=3')

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

[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
```



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

```
[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)
```

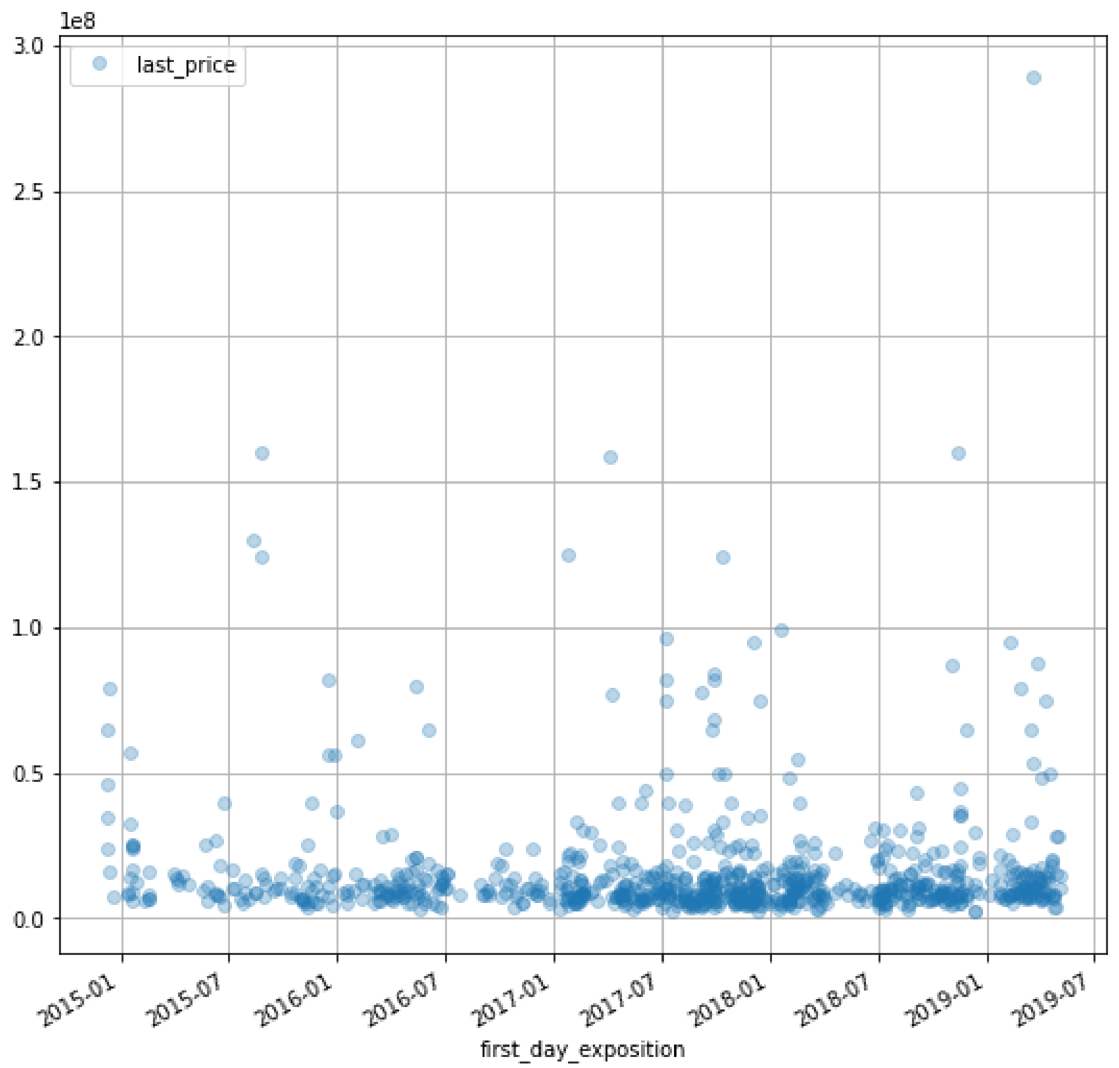


```
[42]: display(df_spb_centre[['last_price','floor','cityCenters_nearest']].corr())
```

```
                     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
```

```
[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()
```

```
[43]:                  last_price  exposition_year
      last_price         1.000000        -0.023326
      exposition_year   -0.023326         1.000000
```



1.4.7 Citycenter realty price dependence conclusion

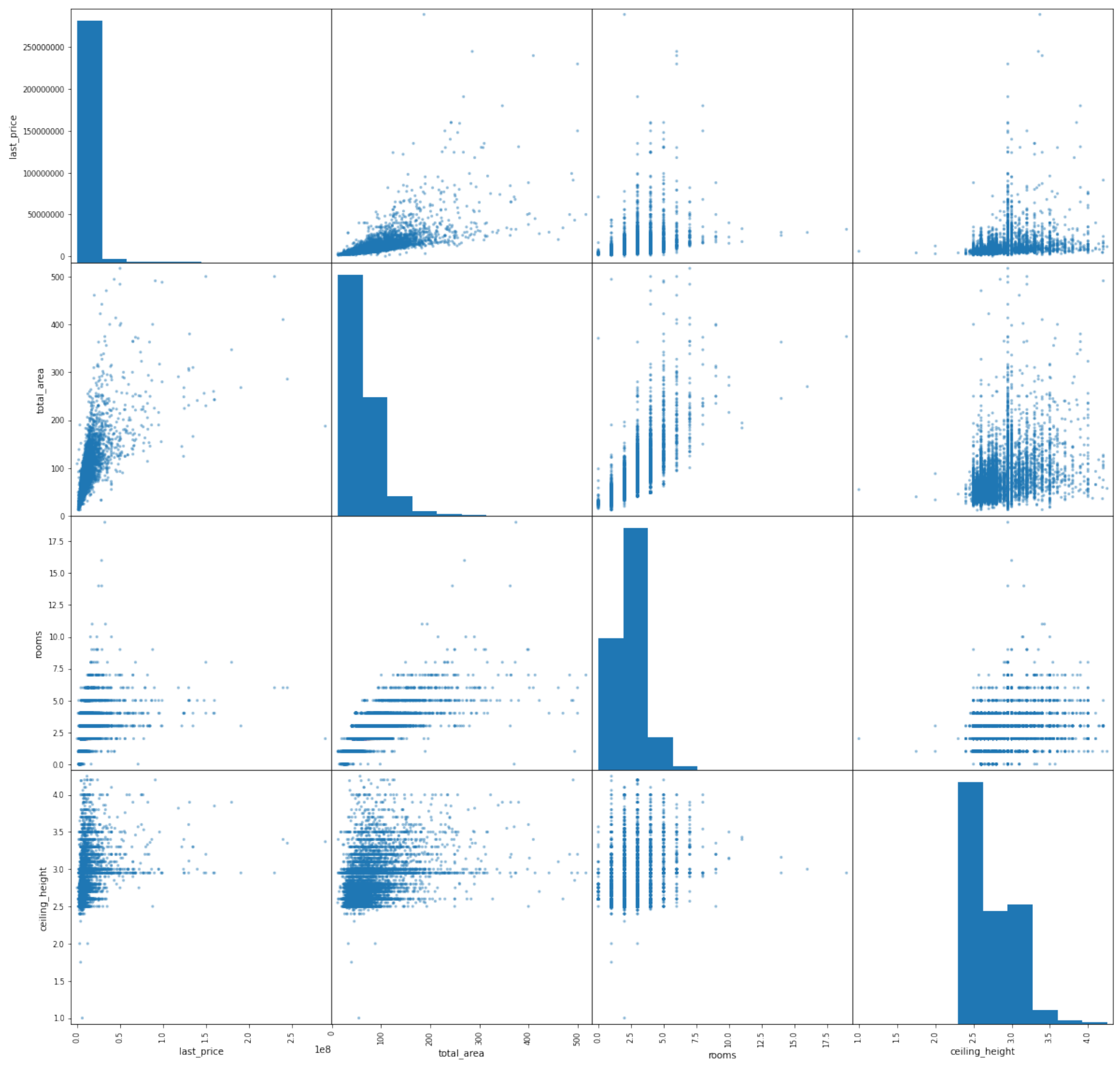
1) The highest influence on the realty price in citycenter is caused by total area, the dependance is 64%
2) Second highest parameter is quantity of bedrooms, dependance - 34%
3) Third parameter is realty floor, dependance 20%
4) Fourth - ceiling height, dependance 15%
5) Fifth one is year of publishing, dependance -7% (negative value)
6) Last one is distance to city center , dependance -17% (negative value)

1.4.8 Analysis of apartment in Saint-Petersburg overall

[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
```

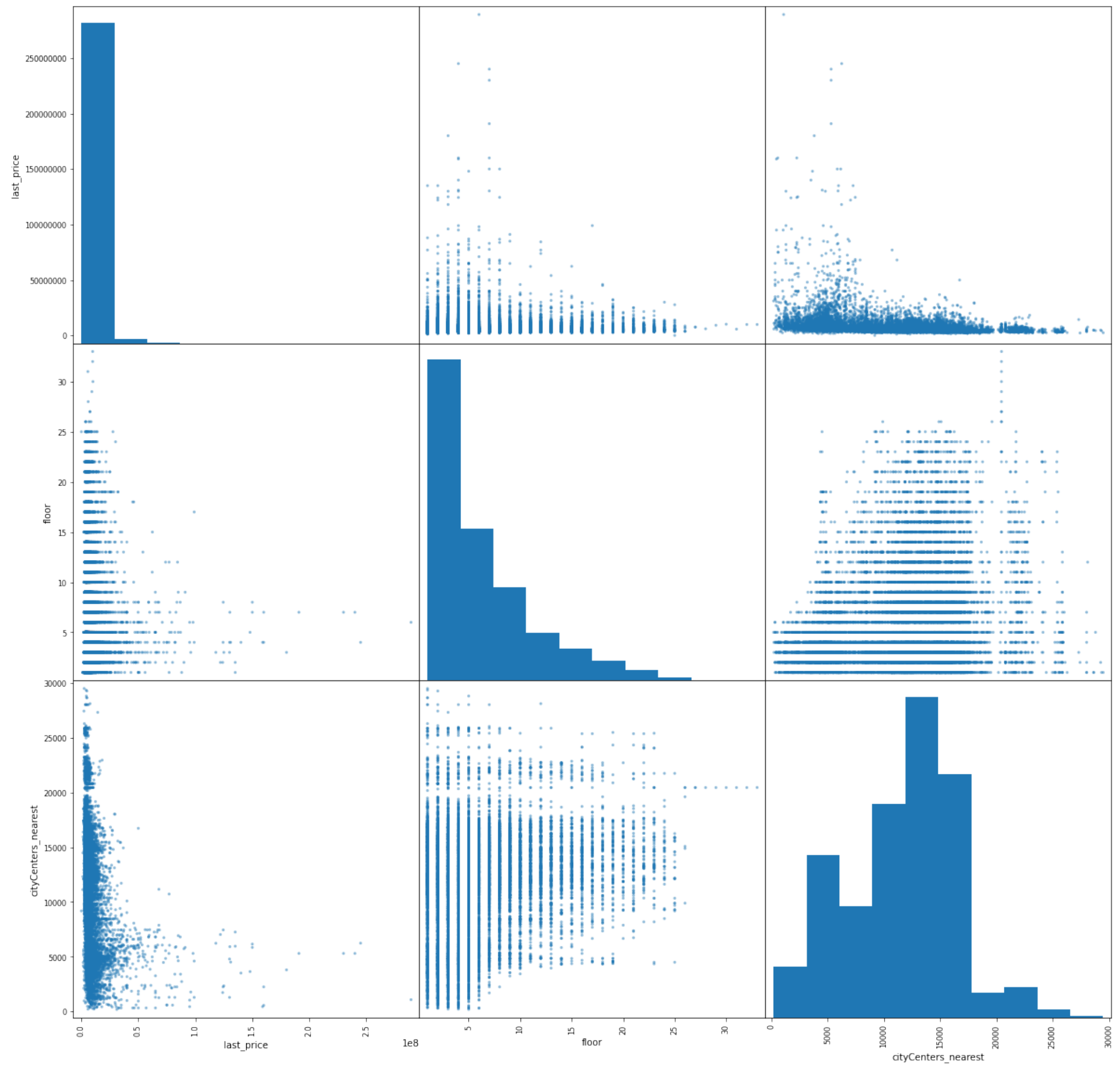
[44]:
```
array([[<AxesSubplot:xlabel='last_price', ylabel='last_price'>,
        <AxesSubplot:xlabel='total_area', ylabel='last_price'>,
        <AxesSubplot:xlabel='rooms', ylabel='last_price'>,
        <AxesSubplot:xlabel='ceiling_height', ylabel='last_price'>],
       [<AxesSubplot:xlabel='last_price',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



```
[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
```

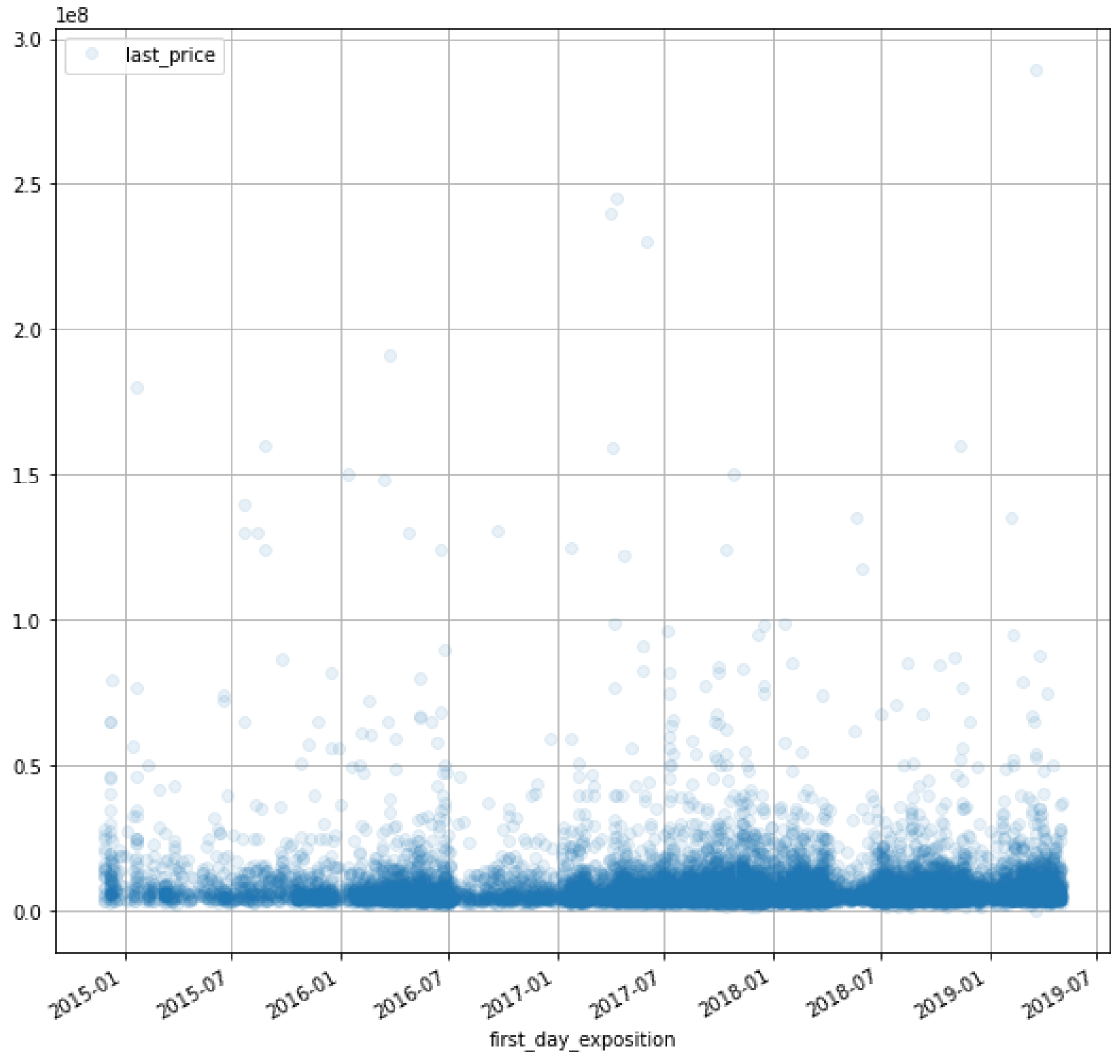
```
[45]: array([[<AxesSubplot:xlabel='last_price', ylabel='last_price'>,
              <AxesSubplot:xlabel='floor', ylabel='last_price'>,
              <AxesSubplot:xlabel='cityCenters_nearest', ylabel='last_price'>],
             [<AxesSubplot:xlabel='last_price', ylabel='floor'>,
              <AxesSubplot:xlabel='floor',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



[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()
```

[46]:
```
                 last_price  exposition_year
last_price         1.000000        -0.056213
```

```
exposition_year   -0.056213         1.000000
```



1.4.9 Cocnlusion on the realty price dependency in Saint-Petersburg

1) The highest influence on the realty price is caused by total area, the dependancy is 72%
2) Second highest parameter is quantity of bedrooms, dependancy 42%
3) Third parameter ceiling height, dependancy 33%
4) Fourth is realty floor, dependancy -1% (negative value)
5) Fifth one is year of publishing, dependancy - 8% (negative value)
6) Last one is distance to city center , dependancy - 30% (negative value)

1.4.10 Comparison of price dependency in citycenter 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 citycenter has high ceilings.
- realty floor has less dependency - on 21% lower in th 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%)

1.4.11 Citycenter realty price dependency conclusion

1) The highest influence on the realty price in citycenter is caused by total area, the dependancy is 64%
2) Second highest parameter is quantity of bedrooms, dependancy - 34%
3) Third parameter is realty floor, dependancy 20%
4) Fourth - ceiling height, dependancy 15%
5) Fifth one is year of publishing, dependancy -7% (negative value)
6) Last one is distance to city center , dependancy -17% (negative value)

1.5 General Conclusion

1) The higher dependency on the price in dataframe has total area (70%), then quantity of bedrooms (40%), floor (5%) and distnce to center of the city (5%).
2) Cities with highest quantity of advertisements:

- -
-
-
-
-
-
-
-
-
-

3) The city center of Saint-Petersburg is considered the area in distance of 3 km from center of the city.
4) Realty in citycenter 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% accordintly. Other parameters are losing their affect on the price in city overall comparing to citycenter area.

*dependency precentage specified in brackets

[ ]:
```
```