link to my collab: https://colab.research.google.com/drive/1trprNJ9EXHnN56SeX6ciygA3evZO8kM9#scrollTo=H6VElXGUwIPV

Project one

The below project will look at data understanding, data preparation, feature underastanding, feature relationship

```
import pandas
# Get the path to the CSV file
#csv_file_path = '/content/Vehicle_data.csv'
train=pandas.read_csv("/content/sample_data/mine1/train.csv")
print(train)
```

```
      12318  12321  2013-11-20       30     18.0    6.0        9.0       2.0
      12319  12322  2013-11-20       46      NaN   14.0       17.0       1.0
      12320  12323  2013-11-20       45     30.0    3.0        9.0       1.0
      12321  12324  2013-11-20       60     42.0    2.0        9.0       1.0
      12322  12325  2013-11-20       43     26.0    5.0        5.0       2.0

             build_year  num_room  kitch_sq  ...  cafe_count_5000_price_2500  \
      0             NaN       NaN       NaN  ...                         9.0
      1             NaN       NaN       NaN  ...                        15.0
      2             NaN       NaN       NaN  ...                        10.0
      3             NaN       NaN       NaN  ...                        11.0
      4             NaN       NaN       NaN  ...                       319.0
      ...           ...       ...       ...  ...                         ...
      12318      1967.0       1.0       5.0  ...                         5.0
      12319         NaN       1.0       1.0  ...                         1.0
      12320      1973.0       2.0       6.0  ...                        19.0
      12321      1975.0       3.0       6.0  ...                        15.0
      12322      1963.0       2.0       5.0  ...                         NaN

            cafe_count_5000_price_4000 cafe_count_5000_price_high  \
      0                            4.0                        0.0
      1                            3.0                        0.0
      2                            3.0                        0.0
      3                            2.0                        1.0
      4                          108.0                       17.0
      ...                          ...                        ...
      12318                        2.0                        0.0
      12319                        0.0                        0.0
      12320                        5.0                        1.0
      12321                        4.0                        0.0
      12322                        NaN                        NaN

             big_church_count_5000  church_count_5000  mosque_count_5000  \
      0                       13.0               22.0                1.0
      1                       15.0               29.0                1.0
      2                       11.0               27.0                0.0
      3                        4.0                4.0                0.0
      4                      135.0              236.0                2.0
      ...                      ...                ...                ...
      12318                    6.0               12.0                0.0
      12319                    2.0                3.0                0.0
      12320                    8.0               11.0                0.0
      12321                   12.0               28.0                1.0
      12322                    NaN                NaN                NaN

             leisure_count_5000  sport_count_5000  market_count_5000   price_doc
      0                     0.0              52.0                4.0   5850000.0
      1                    10.0              66.0               14.0   6000000.0
      2                     4.0              67.0               10.0   5700000.0
      3                     0.0              26.0                3.0  13100000.0
      4                    91.0             195.0               14.0  16331452.0
      ...                   ...               ...                ...         ...
      12318                 2.0              38.0                5.0   4900000.0
      12319                 0.0               6.0                1.0   4750284.0
      12320                 1.0              56.0                7.0   6400000.0
      12321                 2.0              82.0                9.0   7500000.0
      12322                 NaN               NaN                NaN         NaN

      [12323 rows x 292 columns]
```

Data understanding

Dataframe shape Dtypes Describe Head and tail coloumns

```
# Get the shape of the dataframe
shape = train.shape
# Print the shape of the dataframe
print(shape)
```

```
(12323, 292)
```

```
# Check the data types of the columns in the dataframe
print(train.dtypes)
```

```
id                      int64
timestamp              object
full_sq                 int64
life_sq                float64
floor                  float64
                         ...
mosque_count_5000      float64
leisure_count_5000     float64
sport_count_5000       float64
market_count_5000      float64
price_doc              float64
Length: 292, dtype: object
```

```
# Print the summary of the dataframe
print(train.describe())
```

```
                 id       full_sq      life_sq         floor    max_floor  \
count  12323.000000  12323.000000  9575.000000  12156.000000  2751.000000
mean    6163.795748     53.613568    33.940888      7.922178    12.454744
std     3557.805713     53.209593    21.828396      5.391573     6.683171
min        1.000000      1.000000     0.000000      0.000000     0.000000
25%     3083.500000     38.000000    20.000000      4.000000     9.000000
50%     6164.000000     47.000000    30.000000      7.000000    12.000000
75%     9244.500000     62.000000    42.000000     11.000000    17.000000
max    12325.000000   5326.000000   802.000000     44.000000    40.000000

          material    build_year     num_room     kitch_sq        state  ...  \
count  2751.000000  2.209000e+03  2751.000000  2751.000000  2144.000000  ...
mean      1.890585  1.098305e+04     1.885133     7.145402     2.240672  ...
std       1.503459  4.265981e+05     0.870465    40.344110     1.073547  ...
min       1.000000  0.000000e+00     0.000000     0.000000     1.000000  ...
25%       1.000000  1.966000e+03     1.000000     1.000000     2.000000  ...
50%       1.000000  1.978000e+03     2.000000     6.000000     2.000000  ...
75%       2.000000  2.002000e+03     2.000000     9.000000     3.000000  ...
max       6.000000  2.005201e+07    19.000000  2013.000000    33.000000  ...

       cafe_count_5000_price_2500  cafe_count_5000_price_4000  \
count                12322.00000                12322.000000
mean                    31.02037                   10.353758
std                     71.72763                   27.672551
min                      0.00000                    0.000000
25%                      2.00000                    1.000000
50%                      8.00000                    2.000000
75%                     20.00000                    4.000000
max                    377.00000                  147.000000

       cafe_count_5000_price_high  big_church_count_5000  church_count_5000  \
count                12322.000000           12322.000000       12322.000000
mean                     1.691852              14.662392          29.535222
std                      5.284772              28.412069          46.343634
min                      0.000000               0.000000           0.000000
25%                      0.000000               2.000000           8.000000
50%                      0.000000               7.000000          16.000000
75%                      1.000000              12.000000          28.000000
max                     30.000000             151.000000         250.000000

       mosque_count_5000  leisure_count_5000  sport_count_5000  \
count       12322.000000        12322.000000      12322.000000
mean            0.480441            8.278120         52.428989
std             0.608877           20.137471         45.376008
min             0.000000            0.000000          0.000000
25%             0.000000            0.000000         12.000000
50%             0.000000            2.000000         47.000000
75%             1.000000            6.000000         75.000000
max             2.000000          105.000000        218.000000

       market_count_5000     price_doc
count       12322.000000  1.232200e+04
mean            5.956825  6.493737e+06
std             4.774291  4.389983e+06
min             0.000000  1.900000e+05
```

```
25%            1.000000  4.321635e+06
50%            5.000000  5.756100e+06
75%           10.000000  7.500000e+06
```

```
# Get the list of columns
columns = train.columns

# Print the list of columns
print(columns)
```

```
Index(['id', 'timestamp', 'full_sq', 'life_sq', 'floor', 'max_floor',
       'material', 'build_year', 'num_room', 'kitch_sq',
       ...
       'cafe_count_5000_price_2500', 'cafe_count_5000_price_4000',
       'cafe_count_5000_price_high', 'big_church_count_5000',
       'church_count_5000', 'mosque_count_5000', 'leisure_count_5000',
       'sport_count_5000', 'market_count_5000', 'price_doc'],
      dtype='object', length=292)
```

```
# Identify the quantitative variables
quantitative_variables = [column for column in train.columns if train[column].dtype in ['float', 'int']]

# Identify the qualitative variables
qualitative_variables = [column for column in train.columns if train[column].dtype == 'object']

print('Quantitative variables:', quantitative_variables)
print('Qualitative variables:', qualitative_variables)
```

```
Quantitative variables: ['id', 'full_sq', 'life_sq', 'floor', 'max_floor', 'material', 'build_year', 'num_room', 'kitch_sq', 'state',
Qualitative variables: ['timestamp', 'product_type', 'sub_area', 'culture_objects_top_25', 'thermal_power_plant_raion', 'incineration_
```

```
# Print the summary of the dataframe
print(train.describe())
```

```
std     3557.805713     53.209593    21.828396     5.391573     6.683171
min        1.000000      1.000000     0.000000     0.000000     0.000000
25%     3083.500000     38.000000    20.000000     4.000000     9.000000
50%     6164.000000     47.000000    30.000000     7.000000    12.000000
75%     9244.500000     62.000000    42.000000    11.000000    17.000000
max    12325.000000   5326.000000   802.000000    44.000000    40.000000

          material    build_year     num_room     kitch_sq        state  ...  \
count  2751.000000  2.209000e+03  2751.000000  2751.000000  2144.000000  ...
mean      1.890585  1.098305e+04     1.885133     7.145402     2.240672  ...
std       1.503459  4.265981e+05     0.870465    40.344110     1.073547  ...
min       1.000000  0.000000e+00     0.000000     0.000000     1.000000  ...
25%       1.000000  1.966000e+03     1.000000     1.000000     2.000000  ...
50%       1.000000  1.978000e+03     2.000000     6.000000     2.000000  ...
75%       2.000000  2.002000e+03     2.000000     9.000000     3.000000  ...
max       6.000000  2.005201e+07    19.000000  2013.000000    33.000000  ...

       cafe_count_5000_price_2500  cafe_count_5000_price_4000  \
count                 12322.00000                12322.000000
mean                     31.02037                   10.353758
std                      71.72763                   27.672551
min                       0.00000                    0.000000
25%                       2.00000                    1.000000
50%                       8.00000                    2.000000
75%                      20.00000                    4.000000
max                     377.00000                  147.000000

       cafe_count_5000_price_high  big_church_count_5000  church_count_5000  \
count                12322.000000           12322.000000       12322.000000
mean                     1.691852              14.662392          29.535222
std                      5.284772              28.412069          46.343634
min                      0.000000               0.000000           0.000000
25%                      0.000000               2.000000           8.000000
50%                      0.000000               7.000000          16.000000
75%                      1.000000              12.000000          28.000000
max                     30.000000             151.000000         250.000000

       mosque_count_5000  leisure_count_5000  sport_count_5000  \
count       12322.000000        12322.000000      12322.000000
mean            0.480441            8.278120         52.428989
std             0.608877           20.137471         45.376008
min             0.000000            0.000000          0.000000
```

```
max             2.000000          105.000000        218.000000

       market_count_5000     price_doc
count       12322.000000  1.232200e+04
mean            5.956825  6.493737e+06
std             4.774291  4.389983e+06
min             0.000000  1.900000e+05
25%             1.000000  4.321635e+06
50%             5.000000  5.756100e+06
75%            10.000000  7.500000e+06
max            20.000000  1.111111e+08
```

Data preperation

drop irrelevant columns and rows identify duplicate columns renaming columns feature creation

Method #1: missing data (by columns) heatmap

```
# Missing data
# How to find out?
# Method #1: missing data (by columns) count & percentage
train[qualitative_variables].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12323 entries, 0 to 12322
Data columns (total 16 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   timestamp                 12323 non-null  object
 1   product_type              12323 non-null  object
 2   sub_area                  12323 non-null  object
 3   culture_objects_top_25    12323 non-null  object
 4   thermal_power_plant_raion 12323 non-null  object
 5   incineration_raion        12323 non-null  object
 6   oil_chemistry_raion       12323 non-null  object
 7   radiation_raion           12323 non-null  object
 8   railroad_terminal_raion   12323 non-null  object
 9   big_market_raion          12323 non-null  object
 10  nuclear_reactor_raion     12323 non-null  object
 11  detention_facility_raion  12323 non-null  object
 12  water_1line               12323 non-null  object
 13  big_road1_1line           12323 non-null  object
 14  railroad_1line            12323 non-null  object
 15  ecology                   12323 non-null  object
dtypes: object(16)
memory usage: 1.5+ MB
```

```
# Counts the missing values by columns
num_missing = train.isna().sum()
columns_with_missing = num_missing[num_missing > 0]

print(columns_with_missing)
```

```
life_sq               2748
floor                  167
max_floor             9572
material              9572
build_year           10114
                     ...
mosque_count_5000        1
leisure_count_5000       1
sport_count_5000         1
market_count_5000        1
price_doc                1
Length: 109, dtype: int64
```

```
# Calculate the missing values by columns using mean
num_missing = train.isna().mean()
columns_with_missing = num_missing[num_missing > 0]

print(columns_with_missing)
```

```
life_sq              0.222998
floor                0.013552
max_floor            0.776759
material             0.776759
```

```
build_year            0.820742
                        ...
mosque_count_5000     0.000081
leisure_count_5000    0.000081
sport_count_5000      0.000081
market_count_5000     0.000081
price_doc             0.000081
Length: 109, dtype: float64
```

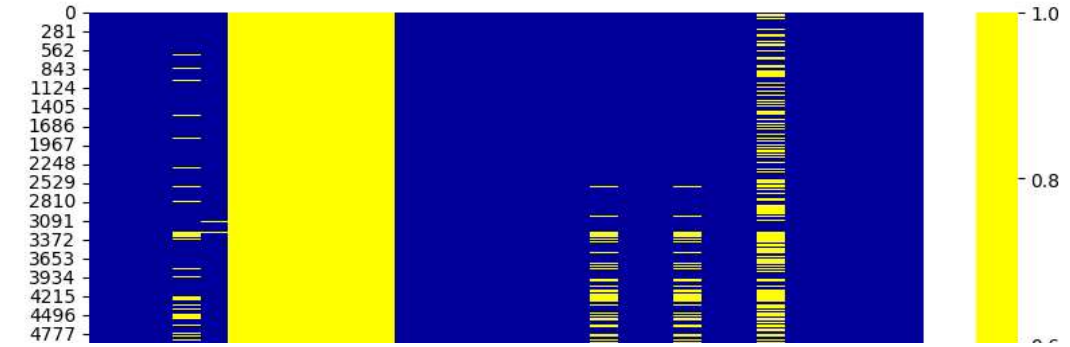
```
# Calculate the percentages of missing values by columns
num_missing = train.isna().mean()*100
columns_with_missing = num_missing[num_missing > 0]

print(columns_with_missing)
```

```
life_sq               22.299765
floor                  1.355189
max_floor             77.675891
material              77.675891
build_year            82.074170
                        ...
mosque_count_5000      0.008115
leisure_count_5000     0.008115
sport_count_5000       0.008115
market_count_5000      0.008115
price_doc              0.008115
Length: 109, dtype: float64
```

Method #2: missing data (by columns) heatmap

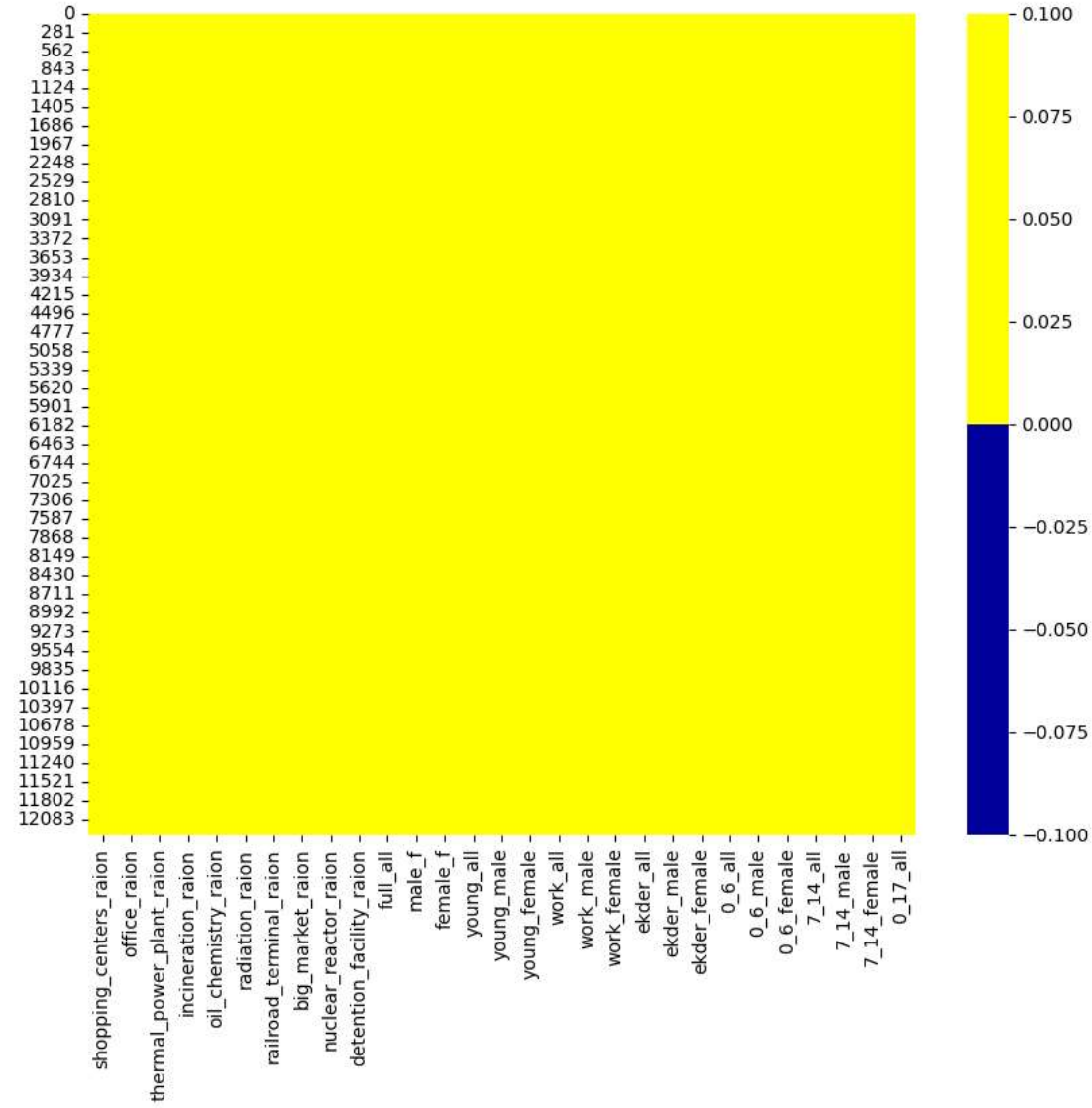
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,8))

cols = train.columns[:30]#uses data from index 1-30
colours = ['#000099', '#ffff00'] # specify colours: yellow - missing. blue - not missing
sns.heatmap(train[cols].isna(), cmap=sns.color_palette(colours))
```

```
<Axes: >
```




```
plt.figure(figsize=(10,8))

cols = train.columns[31:60]#uses data from index 31-60
colours = ['#000099', '#ffff00'] # specify colours: yellow - missing. blue - not missing
sns.heatmap(train[cols].isna(), cmap=sns.color_palette(colours))
```

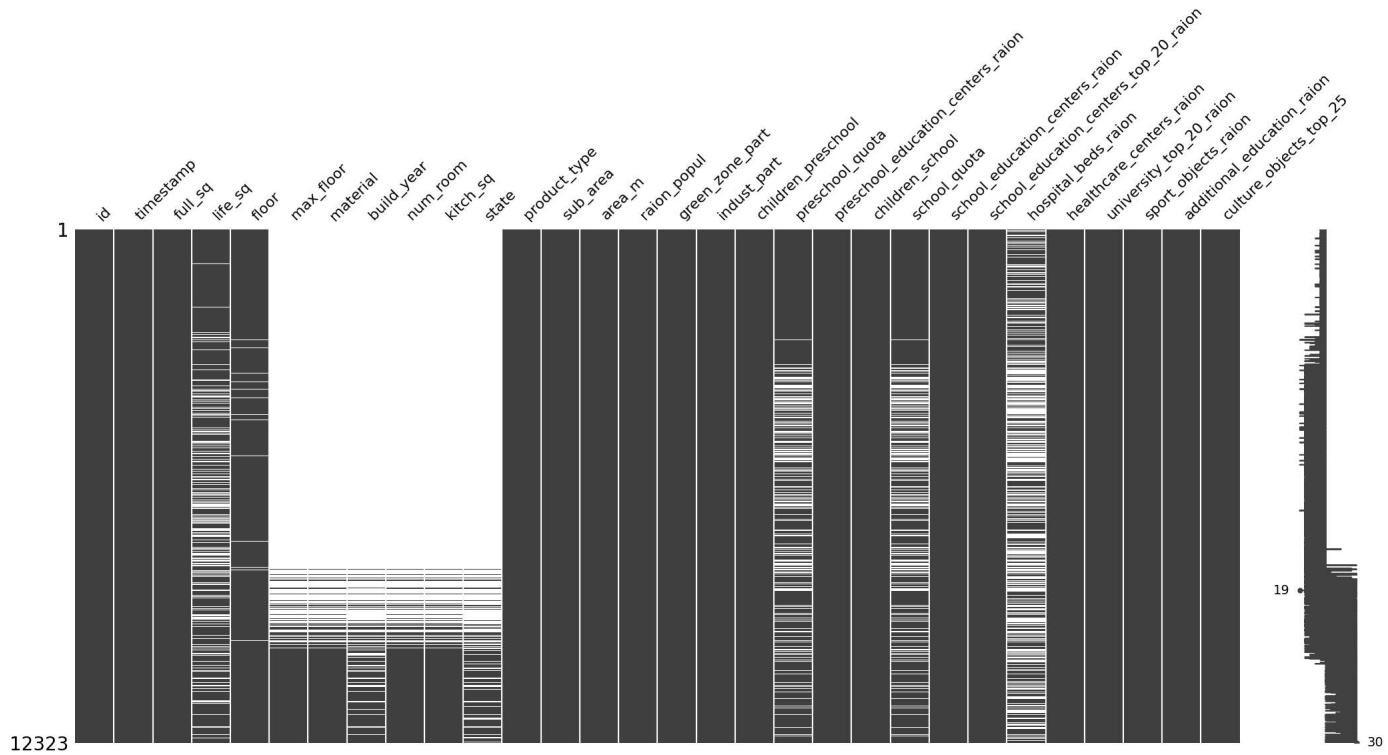
```
<Axes: >
```



Method 3

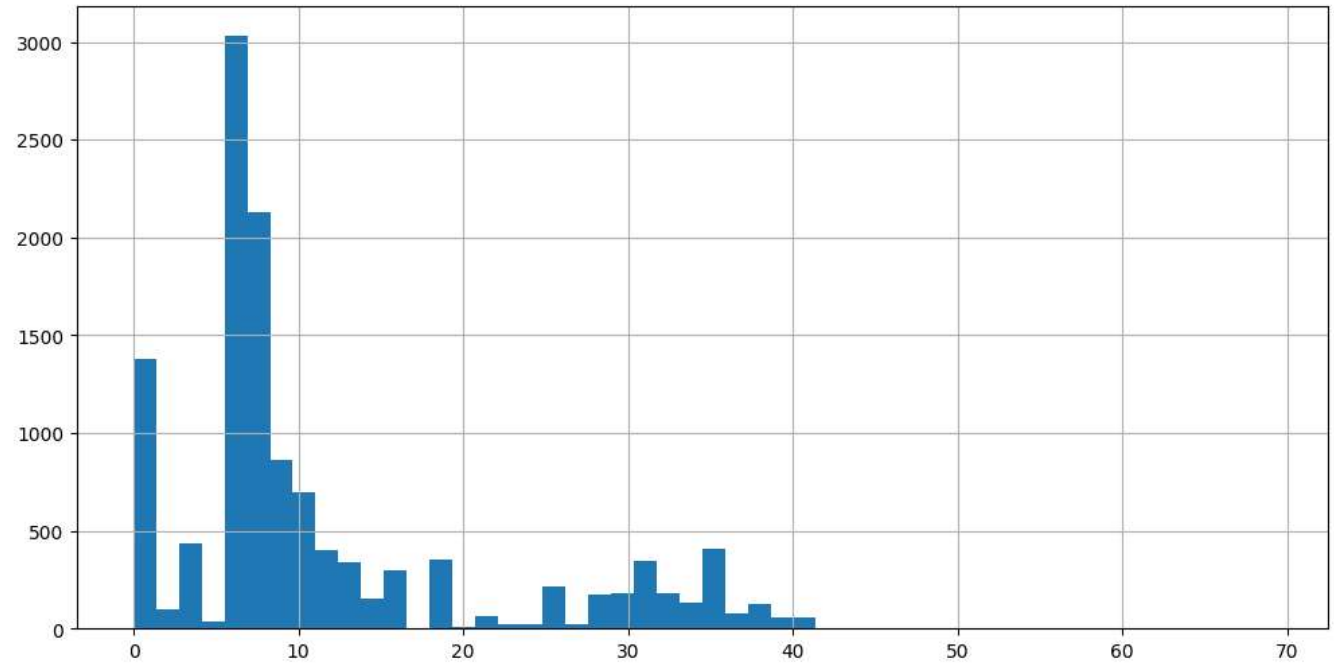
```
# Using missingno library
import missingno as msno
msno.matrix(train.iloc[:, :30])
```

```
<Axes: >
```




```
# Method #3: missing data (by rows) histogram
plt.figure(figsize=(12,6))
missing_by_row = train.isna().sum(axis='columns')
missing_by_row.hist(bins=50)
```

```
<Axes: >
```



DATA CLEANING

Technique #1: drop columns / features

```
#Identifying the numeric data with more than 30% missing values
pct_missing = train.isna().mean()*100
```

```
pct_missing[pct_missing > 30]
```

```
max_floor                    77.675891
material                     77.675891
build_year                   82.074170
num_room                     77.675891
kitch_sq                     77.675891
state                        82.601639
hospital_beds_raion          47.318023
cafe_sum_500_min_price_avg   45.289296
cafe_sum_500_max_price_avg   45.289296
cafe_avg_price_500           45.289296
dtype: float64
```

Technique #2: drop rows / observations

```
#train.dropna(axis='index', thresh=292-35+1).shape
#same as the code above
train_less_missing_rows = train[missing_by_row < 35].copy()
train_less_missing_rows.shape
```

```
(11593, 292)
```

Technique #3: impute the missing with constant values

```
train_copy = train.copy()
train_copy[quantitative_variables] = train_copy[quantitative_variables].fillna(-999)
train_copy[qualitative_variables] = train_copy[qualitative_variables].fillna('MISSING')
train_copy.head(10)# displaying the first ten rows
```

	id	timestamp	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	...	cafe_count_5000_price_2500	cafe_c
0	1	2011-08-20	43	27.0	4.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	9.0	
1	2	2011-08-23	34	19.0	3.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	15.0	
2	3	2011-08-27	43	29.0	2.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	10.0	
3	4	2011-09-01	89	50.0	9.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	11.0	
4	5	2011-09-05	77	77.0	4.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	319.0	
5	6	2011-09-06	67	46.0	14.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	62.0	
6	7	2011-09-08	25	14.0	10.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	81.0	
7	8	2011-09-09	44	44.0	5.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	9.0	
8	9	2011-09-10	42	27.0	5.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	19.0	
9	10	2011-09-13	36	21.0	9.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	19.0	

10 rows × 292 columns

Technique #4: impute the missing with statistics

```
# We can impute the numeric columns with their respective medians.

train_copy = train.copy()
```

```
med = train_copy[quantitative_variables].median()
train_copy[quantitative_variables] = train_copy[quantitative_variables].fillna(med)
print(train_copy)# displaying the whole rows
```

```
      12318  12321  2013-11-20       30     18.0    6.0        9.0       2.0
      12319  12322  2013-11-20       46     30.0   14.0       17.0       1.0
      12320  12323  2013-11-20       45     30.0    3.0        9.0       1.0
      12321  12324  2013-11-20       60     42.0    2.0        9.0       1.0
      12322  12325  2013-11-20       43     26.0    5.0        5.0       2.0

             build_year  num_room  kitch_sq  ...  cafe_count_5000_price_2500  \
      0          1978.0       2.0       6.0  ...                         9.0
      1          1978.0       2.0       6.0  ...                        15.0
      2          1978.0       2.0       6.0  ...                        10.0
      3          1978.0       2.0       6.0  ...                        11.0
      4          1978.0       2.0       6.0  ...                       319.0
      ...           ...       ...       ...  ...                         ...
      12318      1967.0       1.0       5.0  ...                         5.0
      12319      1978.0       1.0       1.0  ...                         1.0
      12320      1973.0       2.0       6.0  ...                        19.0
      12321      1975.0       3.0       6.0  ...                        15.0
      12322      1963.0       2.0       5.0  ...                         8.0

            cafe_count_5000_price_4000 cafe_count_5000_price_high  \
      0                            4.0                        0.0
      1                            3.0                        0.0
      2                            3.0                        0.0
      3                            2.0                        1.0
      4                          108.0                       17.0
      ...                          ...                        ...
      12318                        2.0                        0.0
      12319                        0.0                        0.0
      12320                        5.0                        1.0
      12321                        4.0                        0.0
      12322                        2.0                        0.0

             big_church_count_5000  church_count_5000  mosque_count_5000  \
      0                       13.0               22.0                1.0
      1                       15.0               29.0                1.0
      2                       11.0               27.0                0.0
      3                        4.0                4.0                0.0
      4                      135.0              236.0                2.0
      ...                      ...                ...                ...
      12318                    6.0               12.0                0.0
      12319                    2.0                3.0                0.0
      12320                    8.0               11.0                0.0
      12321                   12.0               28.0                1.0
      12322                    7.0               16.0                0.0

             leisure_count_5000  sport_count_5000  market_count_5000   price_doc
      0                     0.0              52.0                4.0   5850000.0
      1                    10.0              66.0               14.0   6000000.0
      2                     4.0              67.0               10.0   5700000.0
      3                     0.0              26.0                3.0  13100000.0
      4                    91.0             195.0               14.0  16331452.0
      ...                   ...               ...                ...         ...
      12318                 2.0              38.0                5.0   4900000.0
      12319                 0.0               6.0                1.0   4750284.0
      12320                 1.0              56.0                7.0   6400000.0
      12321                 2.0              82.0                9.0   7500000.0
      12322                 2.0              47.0                5.0   5756100.0

      [12323 rows x 292 columns]
```

```
# We can also impute the non-numeric columns with their most frequent values.

most_freq = train_copy[qualitative_variables].describe().loc['top']
train_copy[qualitative_variables] = train_copy[qualitative_variables].fillna(most_freq)
#most_freq.head(10)# displaying the first ten rows
print(most_freq)# displaying the whole rows
```

```
      timestamp                           2013-11-14
      product_type                        Investment
      sub_area                  Poselenie Sosenskoe
      culture_objects_top_25                    no
      thermal_power_plant_raion                 no
      incineration_raion                        no
      oil_chemistry_raion                       no
      radiation_raion                           no
      railroad_terminal_raion                   no
      big_market_raion                          no
      nuclear_reactor_raion                     no
      detention_facility_raion                  no
```

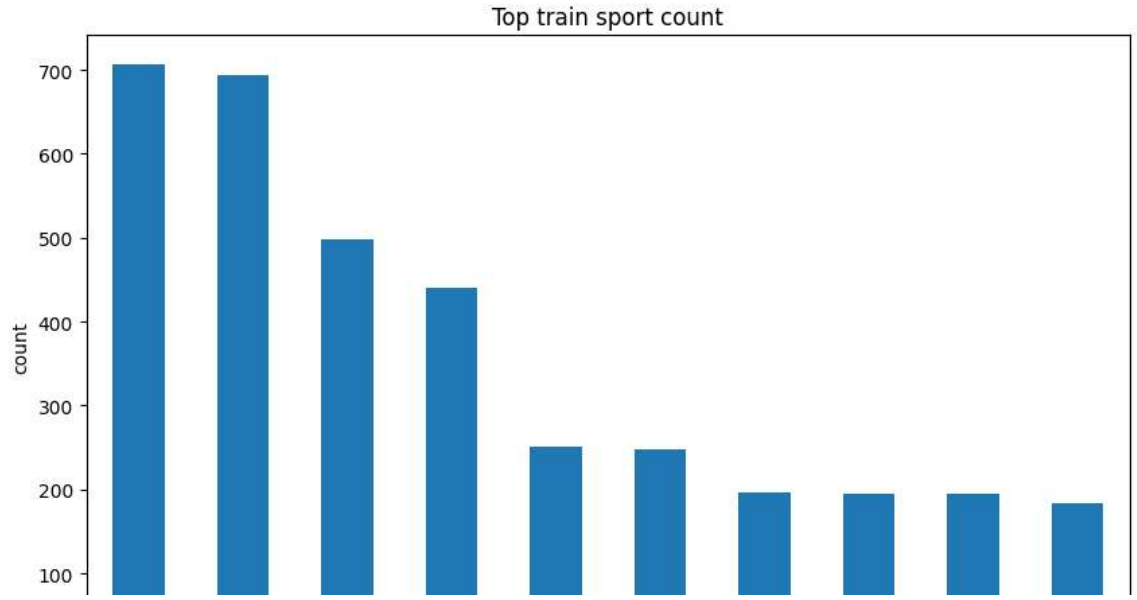
```
water_1line                                 no
big_road1_1line                             no
railroad_1line                              no
ecology                                   poor
Name: top, dtype: object
```

```
print(train_copy)# displaying the whole rows
```

```
      12318  12321  2013-11-20       30     18.0    6.0        9.0       2.0
      12319  12322  2013-11-20       46     30.0   14.0       17.0       1.0
      12320  12323  2013-11-20       45     30.0    3.0        9.0       1.0
      12321  12324  2013-11-20       60     42.0    2.0        9.0       1.0
      12322  12325  2013-11-20       43     26.0    5.0        5.0       2.0

             build_year  num_room  kitch_sq  ...  cafe_count_5000_price_2500  \
      0          1978.0       2.0       6.0  ...                         9.0
      1          1978.0       2.0       6.0  ...                        15.0
      2          1978.0       2.0       6.0  ...                        10.0
      3          1978.0       2.0       6.0  ...                        11.0
      4          1978.0       2.0       6.0  ...                       319.0
      ...           ...       ...       ...  ...                         ...
      12318      1967.0       1.0       5.0  ...                         5.0
      12319      1978.0       1.0       1.0  ...                         1.0
      12320      1973.0       2.0       6.0  ...                        19.0
      12321      1975.0       3.0       6.0  ...                        15.0
      12322      1963.0       2.0       5.0  ...                         8.0

            cafe_count_5000_price_4000 cafe_count_5000_price_high  \
      0                            4.0                        0.0
      1                            3.0                        0.0
      2                            3.0                        0.0
      3                            2.0                        1.0
      4                          108.0                       17.0
      ...                          ...                        ...
      12318                        2.0                        0.0
      12319                        0.0                        0.0
      12320                        5.0                        1.0
      12321                        4.0                        0.0
      12322                        2.0                        0.0

             big_church_count_5000  church_count_5000  mosque_count_5000  \
      0                       13.0               22.0                1.0
      1                       15.0               29.0                1.0
      2                       11.0               27.0                0.0
      3                        4.0                4.0                0.0
      4                      135.0              236.0                2.0
      ...                      ...                ...                ...
      12318                    6.0               12.0                0.0
      12319                    2.0                3.0                0.0
      12320                    8.0               11.0                0.0
      12321                   12.0               28.0                1.0
      12322                    7.0               16.0                0.0

             leisure_count_5000  sport_count_5000  market_count_5000   price_doc
      0                     0.0              52.0                4.0   5850000.0
      1                    10.0              66.0               14.0   6000000.0
      2                     4.0              67.0               10.0   5700000.0
      3                     0.0              26.0                3.0  13100000.0
      4                    91.0             195.0               14.0  16331452.0
      ...                   ...               ...                ...         ...
      12318                 2.0              38.0                5.0   4900000.0
      12319                 0.0               6.0                1.0   4750284.0
      12320                 1.0              56.0                7.0   6400000.0
      12321                 2.0              82.0                9.0   7500000.0
      12322                 2.0              47.0                5.0   5756100.0

      [12323 rows x 292 columns]
```

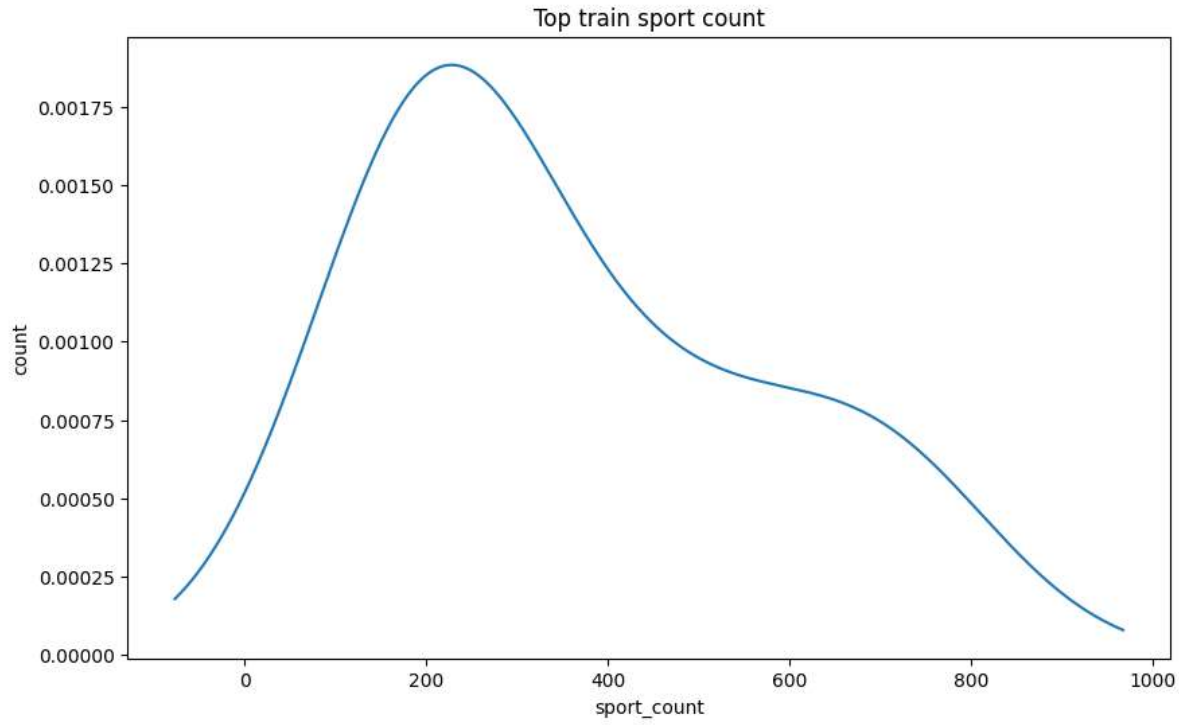
```
plt.figure(figsize=(10,6))
ax=train["sport_count_5000"].value_counts()\
.head(10)\
.plot(kind="bar", title="Top train sport count")
ax.set_xlabel("sport_count")
ax.set_ylabel("count")
```

```
Text(0, 0.5, 'count')
```




```
plt.figure(figsize=(10,6))
ax=train["sport_count_5000"].value_counts()\
.head(10)\
.plot(kind="kde", title="Top train sport count")
ax.set_xlabel("sport_count")
ax.set_ylabel("count")
```

```
Text(0, 0.5, 'count')
```

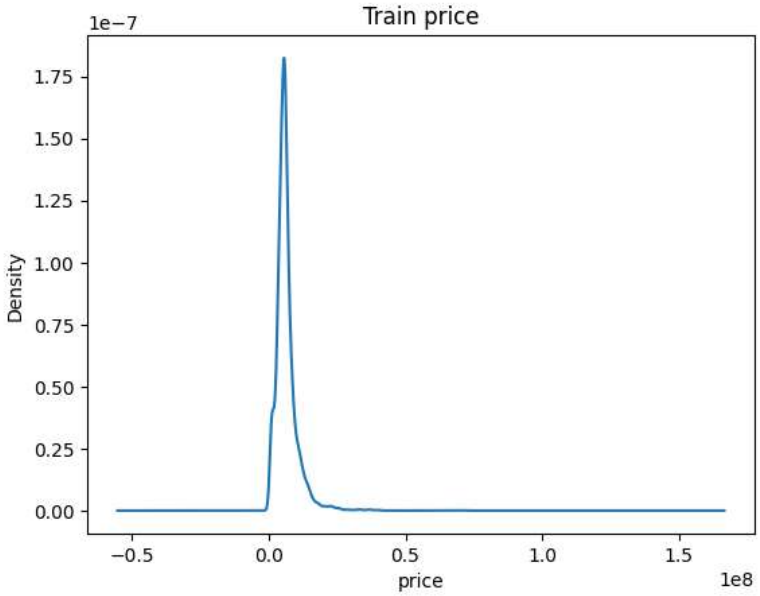



```
ax=train["price_doc"].plot(kind="hist", bins=10, title="Train price")
ax.set_xlabel("price")
```

```
Text(0.5, 0, 'price')
```

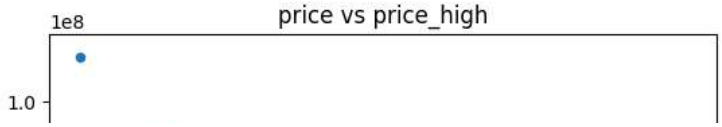

```
ax=train["price_doc"].plot(kind="kde", title="Train price")
ax.set_xlabel("price")
```

```
Text(0.5, 0, 'price')
```



Price vs price_high graph using matplotlib

```
train.plot(kind="scatter",
           x="cafe_count_5000_price_high",
           y="price_doc",
           title="price vs price_high")
plt.show()
```

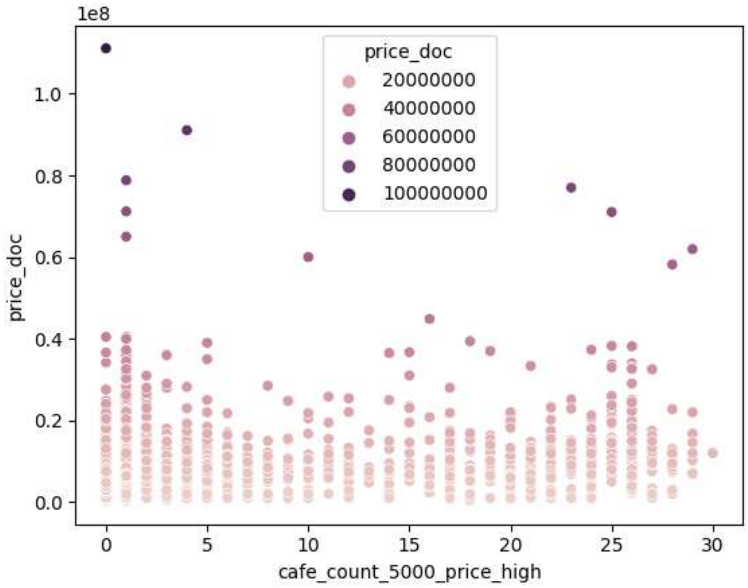


Double-click (or enter) to edit

Price vs price_high graph using seaborn

```
sns.scatterplot(x="cafe_count_5000_price_high",
                y="price_doc",
                hue="price_doc",
                data=train)
```

```
<Axes: xlabel='cafe_count_5000_price_high', ylabel='price_doc'>
```




```
from google.colab import drive
drive.mount('/content/drive')
```

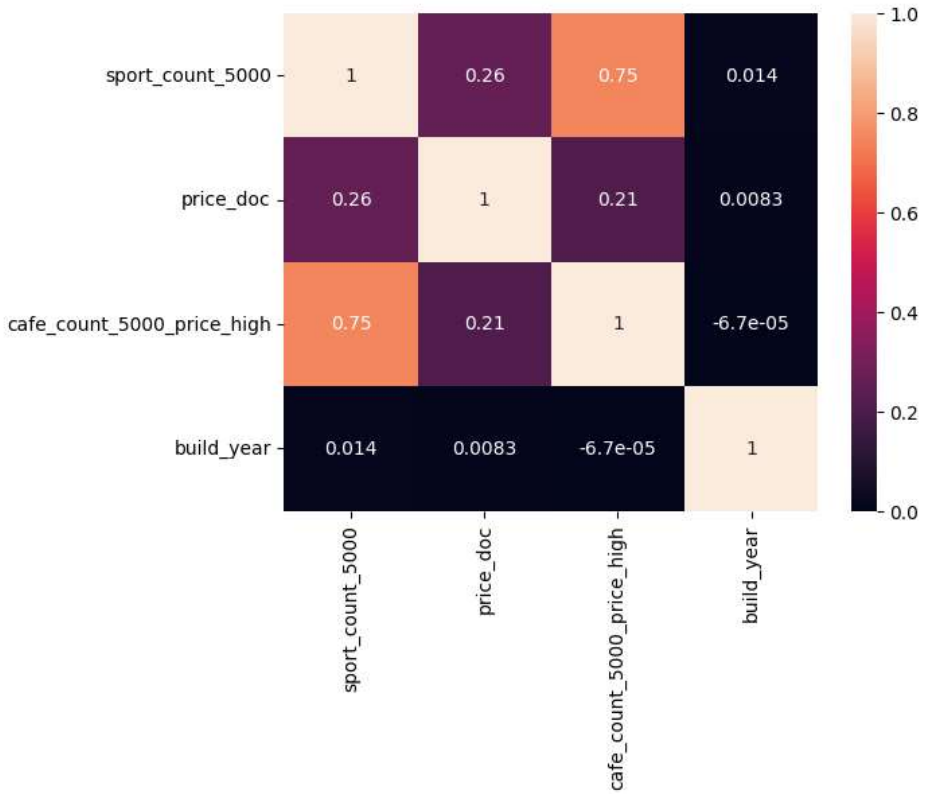
The correlations

```
train[["sport_count_5000", "price_doc", "cafe_count_5000_price_high", "build_year"]].dropna().corr()
```

	sport_count_5000	price_doc	cafe_count_5000_price_high	build_year
sport_count_5000	1.000000	0.261316	0.749006	0.014100
price_doc	0.261316	1.000000	0.209695	0.008270
cafe_count_5000_price_high	0.749006	0.209695	1.000000	-0.000067
build_year	0.014100	0.008270	-0.000067	1.000000

```
train_corr=train[["sport_count_5000", "price_doc", "cafe_count_5000_price_high", "build_year"]].dropna().corr()
sns.heatmap(train_corr, annot=True)
```

```
<Axes: >
```



✓ 0s completed at 5:03 AM