

Импортируем всё необходимое


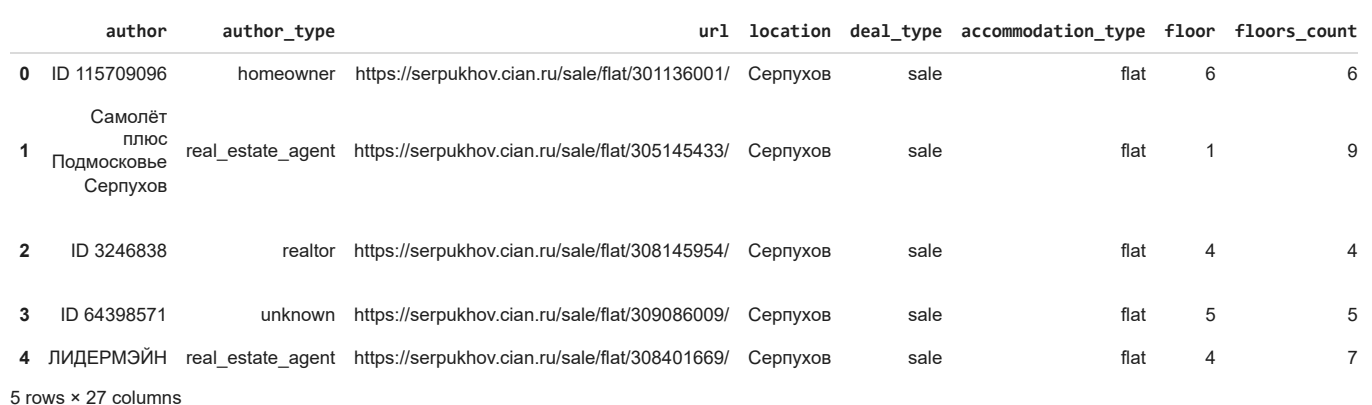
```
pip install wordcloud matplotlib
```

 [Показать скрытые выходные данные](#)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

Начинаем загрузку и работу с данными


```
df = pd.read_csv("/content/omagad.csv") #загружаем датасет
df.head()
```

	author	author_type	url	location	deal_type	accommodation_type	floor	floors_count
0	ID 115709096	homeowner	https://serpukhov.cian.ru/sale/flat/301136001/	Серпухов	sale	flat	6	6
1	Самолёт плюс Подмосковье Серпухов	real_estate_agent	https://serpukhov.cian.ru/sale/flat/305145433/	Серпухов	sale	flat	1	9
2	ID 3246838	realtor	https://serpukhov.cian.ru/sale/flat/308145954/	Серпухов	sale	flat	4	4
3	ID 64398571	unknown	https://serpukhov.cian.ru/sale/flat/309086009/	Серпухов	sale	flat	5	5
4	ЛИДЕРМЭЙН	real_estate_agent	https://serpukhov.cian.ru/sale/flat/308401669/	Серпухов	sale	flat	4	7


5 rows × 27 columns

```
df = df.drop_duplicates()#убираем дубликаты
print(df.columns)
print(df.shape)
```

 Index(['author', 'author_type', 'url', 'location', 'deal_type',
'accommodation_type', 'floor', 'floors_count', 'rooms_count',
'total_meters', 'price', 'year_of_construction', 'object_type',
'have_loggia', 'parking_type', 'house_material_type', 'heating_type',
'finish_type', 'living_meters', 'kitchen_meters', 'phone',
'ceiling_height', 'district', 'street', 'house_number', 'underground',
'residential_complex'],
dtype='object')
(7577, 27)

```
#нам не нужны столбцы автор и автор тайп, сносим
df = df.drop('author', axis=1)
df = df.drop('author_type', axis=1)
```

```
# Теперь узнаём количество нулевых значений
null_counts = df.isna().sum()
print(null_counts)
```

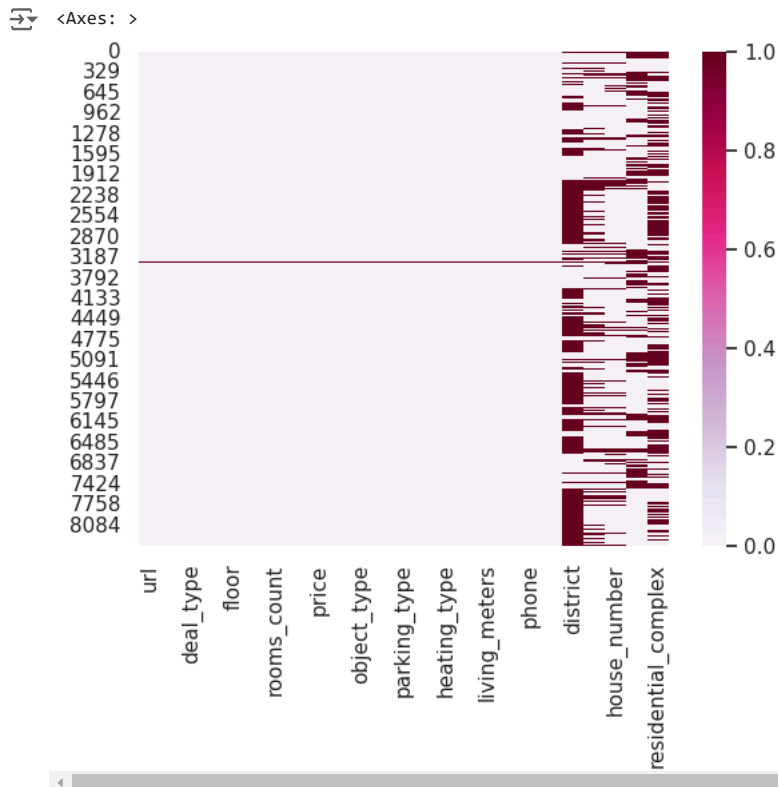
 url 1
location 1
deal_type 1
accommodation_type 1
floor 1
floors_count 1
rooms_count 1
total_meters 1
price 14
year_of_construction 1
object_type 1
have_loggia 1
parking_type 1
house_material_type 1
heating_type 1
finish_type 1

```

living_meters      1
kitchen_meters     1
phone              1
ceiling_height     1
district           4583
street             1575
house_number       1133
underground        2277
residential_complex 3778
dtype: int64

```

```
sns.heatmap(df.isnull(), cmap='PuRd')
```



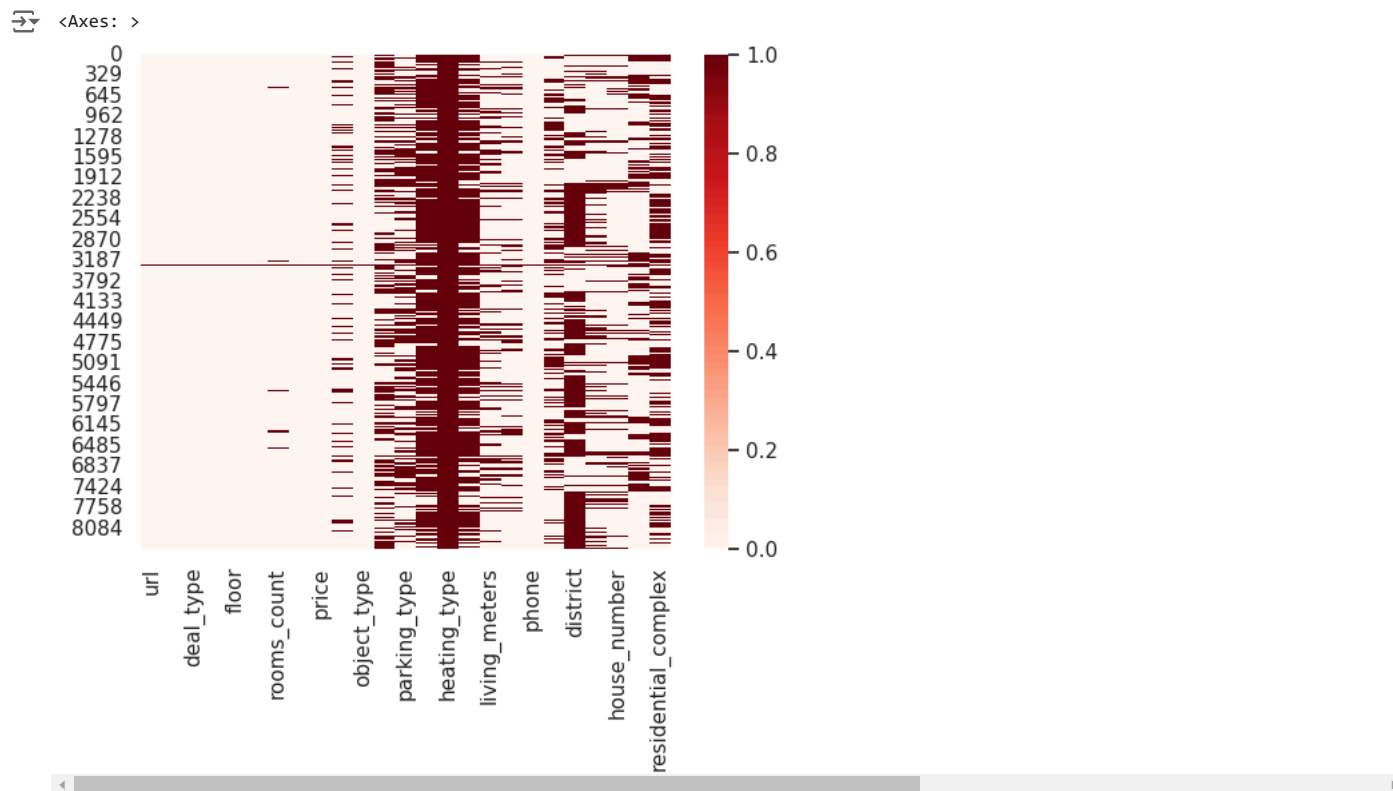
У нас так же много значений равных -1, тк при парсинге то, что мы в итоге не получали проставлялось -1

```

#тут мы заменяем все данные с -1 на нан
df.replace('-1', np.nan, inplace=True)

```

```
sns.heatmap(df.isnull(), cmap='Reds')
```



Удаляем колонки в которых слишком много пропусков

heating_type, house_material_type, finish_type

Так же удаляем колонки которые нам не понадобятся в будущем

phone, deal_type, accommodation_type, house_number, residential_complex


Ещё мы видим, что в лодже, метро и парковке много пустых значений, но это просто их отсутствие

Заменяем нан на нет метро, парковки и лоджи

```
df = df.drop(['heating_type', 'house_material_type', 'finish_type'], axis=1)
df = df.drop(['phone', 'deal_type', 'accommodation_type', 'house_number', 'residential_complex'], axis=1)
```

```
df['have_loggia'] = df['have_loggia'].fillna('нет лоджи')
df['parking_type'] = df['parking_type'].fillna('нет парковки')
df['underground'] = df['underground'].fillna('Нет метро')
```


```
# Тк почти все данные пустые в районе, мы проставляем в него данные из города
df['district'].fillna(df['location'], inplace=True)
```

 <ipython-input-342-973f2e77fe58>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

```
df['district'].fillna(df['location'], inplace=True)
```

```
# очищаем комнаты и улицы и переводим комнаты в инт
df['rooms_count'] = pd.to_numeric(df['rooms_count'], errors='coerce')
df = df.dropna(subset=['rooms_count'])
df['rooms_count'] = df['rooms_count'].astype('int')
df = df.dropna(subset=['street'])
```


 <ipython-input-343-a6883f261fee>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
df['rooms_count'] = df['rooms_count'].astype('int')

```
df['living_meters'] = df['living_meters'].str.replace(r'\xa0м²', '', regex=True).str.replace(',', '.')
df['living_meters'] = df['living_meters'].astype(float)
df['kitchen_meters'] = df['kitchen_meters'].str.replace(r'\xa0м²', '', regex=True).str.replace(',', '.')
df['kitchen_meters'] = df['kitchen_meters'].astype(float)
```

Заполняем пропуски медианой для каждого столбца

```
df['living_meters'].fillna(df['living_meters'].median(), inplace=True)
df['kitchen_meters'].fillna(df['kitchen_meters'].median(), inplace=True)
```

 <ipython-input-345-cfa5ac663253>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]


```
df['living_meters'].fillna(df['living_meters'].median(), inplace=True)
```

<ipython-input-345-cfa5ac663253>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df['kitchen_meters'].fillna(df['kitchen_meters'].median(), inplace=True)
```


```
df['year_of_construction'].unique()
```

 array(['1917', '1975', nan, '1968', '2009', '1980', '1990', '2025',
'2015', '1970', '1984', '1965', '2010', '1964', 'Аукцион', '1992',
'2011', '1986', '1994', '1969', '1956', '1966', '1973', '1971',
'1957', '2013', '1981', '1972', '1959', '2004', '1962', '2019',
'1978', '1977', '2003', '1963', '1952', '2024', '2007', '1995',
'1976', '1982', '1953', '2008', '2026', '2023', '1941', '1993',
'2016', '2027', '2017', '2022', '1932', '2012', '1960', '2006',
'1926', '2020', '1943', '2014', '2021', '1967', '1904', '2028',
'1991', '2018', '1987', '1974', 'Напишите автору', '1979', '1988',
'1930', '1985', '1989', '2005', '1958', '1996', '1940', '1948',
'1961', '1938', '1997', '2002', '2001', '1999', '1983', '1998',
'1902', '2000', '1915', '1910', '1939', '1929', '1954', '1897',
'1955', '1900', 'Позвоните автору', '1949', '1947', '1901', '1909',
'1951', '1936', '1931', '1950', '1934', '1928', '1945', '1937',
'1927', '1896', '1777', '1935'], dtype=object)

```
df['year_of_construction'].replace(['Напишите автору', 'Позвоните автору', '-1', 'Аукцион'], np.nan, inplace=True)
```

Удаляем строки, содержащие NaN

```
df.dropna(subset=['year_of_construction'], inplace=True)
df['year_of_construction'] = df['year_of_construction'].astype('int')
```

 <ipython-input-347-9f4fce767dcf>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting


For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
df['year_of_construction'].replace(['Напишите автору', 'Позвоните автору', '-1', 'Аукцион'], np.nan, inplace=True)
```

```
df = df.drop(df[(df['year_of_construction'] < 1950) | (df['year_of_construction'] > 2025)].index)
```

проверяем на выбросы цена за год

```
df['price'] = pd.to_numeric(df['price'], errors='coerce')
df = df.dropna(subset=['price'])
df['price'] = df['price'].astype(float)
df['year_of_construction'] = df['year_of_construction'].astype('int')
```

 <ipython-input-349-b1e4ebcf2416>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
df['price'] = df['price'].astype(float)

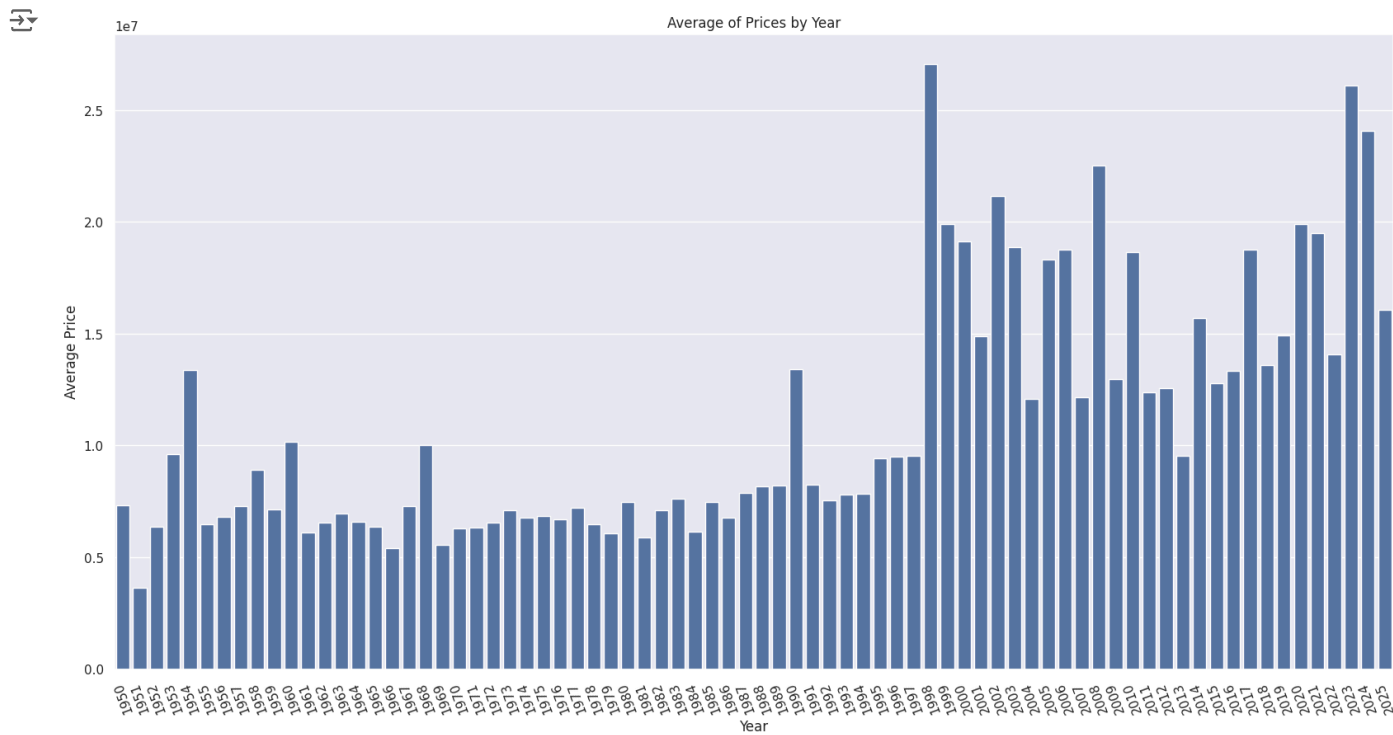
<ipython-input-349-b1e4ebcf2416>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
df['year_of_construction'] = df['year_of_construction'].astype('int')

```
average_prices = df.groupby('year_of_construction')['price'].mean().reset_index()
average_prices
```

```
sns.set(style="darkgrid")
plt.figure(figsize=(20, 10))
sns.barplot(x='year_of_construction', y='price', data=average_prices)
plt.title('Average of Prices by Year')
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.xticks(rotation=110)

plt.show()
```



```
#удаляем 1998
```

```
df = df[df['year_of_construction'] != 1998]
```

```
df = df.dropna(subset=['ceiling_height'])
```

```
df['ceiling_height'] = df['ceiling_height'].str.replace(r'\xa0m', '', regex=True).str.replace(',', '.')
df['ceiling_height'] = df['ceiling_height'].astype(float)
```

```
проверяем на выбросы или аномалии
```

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



ceiling_height

count	3153.000000
mean	2.815046
std	1.005854
min	1.650000
25%	2.650000
50%	2.700000
75%	2.820000
max	52.000000

dtype: float64

```
plt.figure(figsize=(8, 6))
plt.boxplot(df['ceiling_height'], vert=False)

# Добавляем заголовок и метки осей
plt.title('Box Plot of Total Meters with Outliers')
plt.xlabel('ceiling_height')

# Отображаем диаграмму
plt.show()
```



```
#удаляем
df = df[df['ceiling_height'] != 52]

omg = df[df['ceiling_height'] > 8]
omg['url']
```



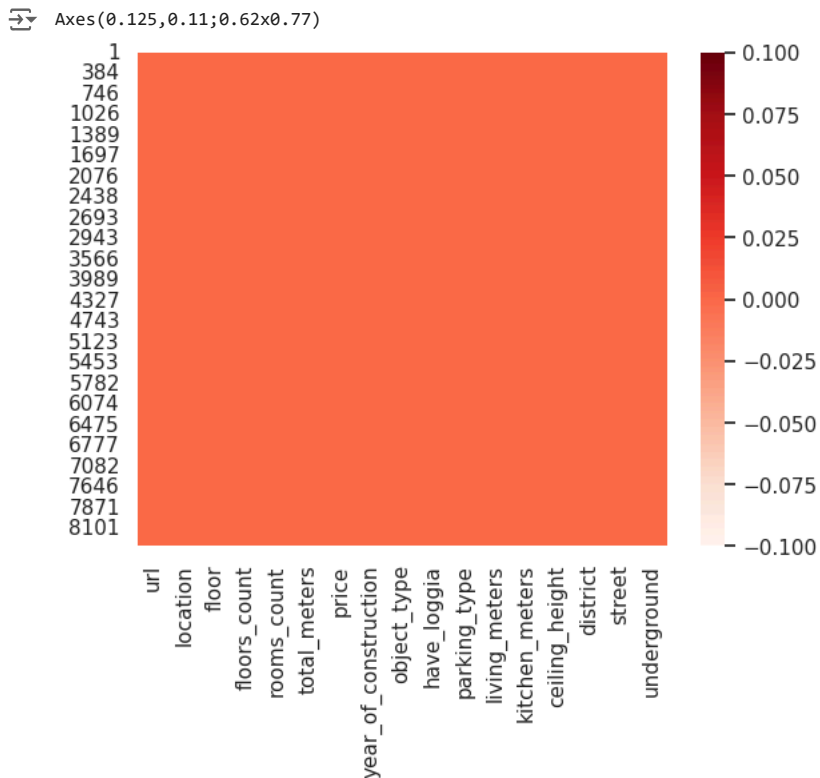
url

5121	https://solnechnogorsk.cian.ru/sale/flat/30038...
5293	https://shchyolkovo.cian.ru/sale/flat/298977261/

dtype: object

```
df = df[df['ceiling_height'] != 25]
df = df[df['ceiling_height'] != 9]
# Удаляем обе ячейки, тк это не так

print(sns.heatmap(df.isnull(), cmap='Reds'))
```



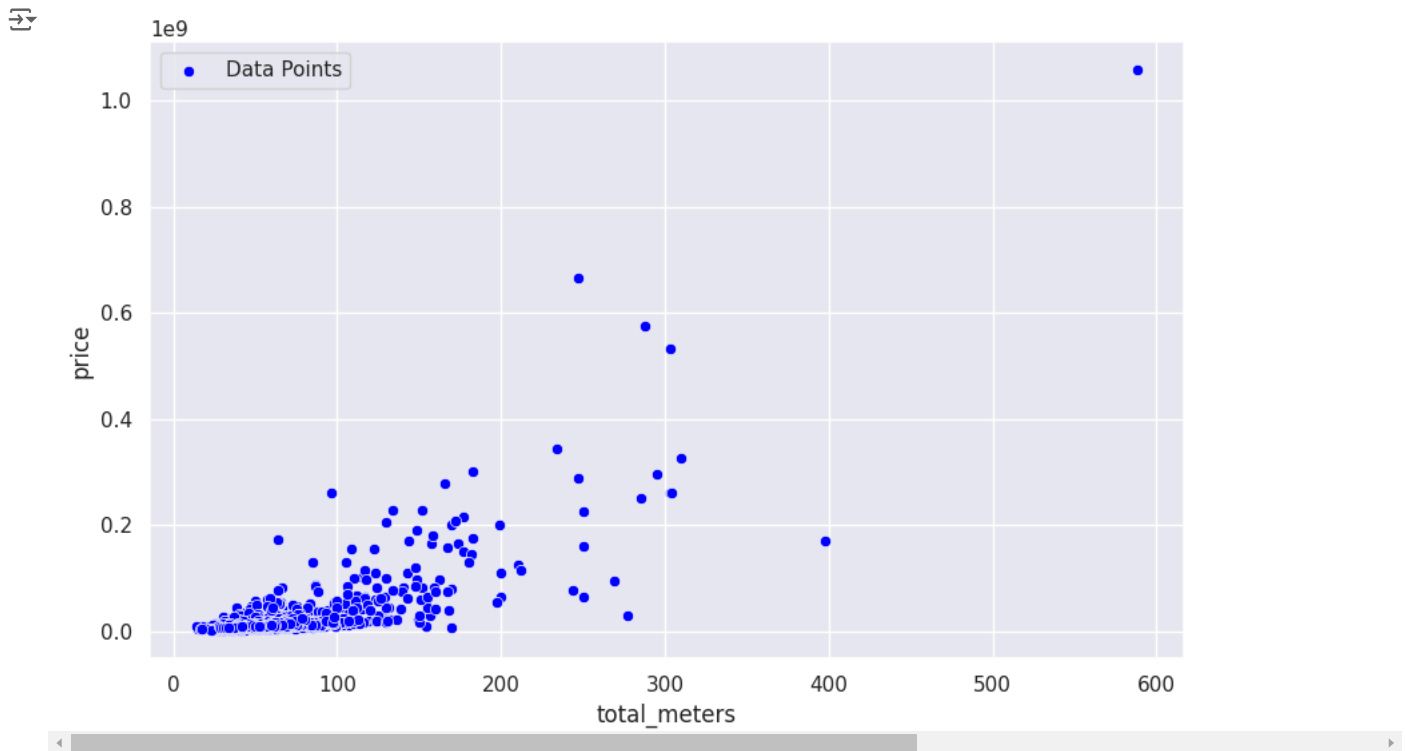
✓ Выводим типы столбцов и начинаем работать над ними

```
df.info()
```

```
→ <class 'pandas.core.frame.DataFrame'>
Index: 3150 entries, 1 to 8391
Data columns (total 17 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   url                                   3150 non-null   object
1   location                             3150 non-null   object
2   floor                                3150 non-null   object
3   floors_count                         3150 non-null   object
4   rooms_count                         3150 non-null   int64
5   total_meters                        3150 non-null   object
6   price                               3150 non-null   float64
7   year_of_construction                3150 non-null   int64
8   object_type                         3150 non-null   object
9   have_loggia                         3150 non-null   object
10  parking_type                        3150 non-null   object
11  living_meters                       3150 non-null   float64
12  kitchen_meters                      3150 non-null   float64
13  ceiling_height                     3150 non-null   float64
14  district                            3150 non-null   object
15  street                              3150 non-null   object
16  underground                         3150 non-null   object
dtypes: float64(4), int64(2), object(11)
memory usage: 443.0+ KB
```

```
df['total_meters'] = df['total_meters'].astype('float')
```

```
#Проверяем на аномалии
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total_meters', y='price', data=df, color='blue', label='Data Points')
plt.show()
```



```
# Удаляем всё что больше 300
```

```
df = df.drop(df[df['total_meters'] > 300].index)
```

```
df['floor'].unique()
```

```
array(['1', '4', '6', '2', '8', '3', '7', '5', '18', '9', '11', '22',  
      '12', '14', '15', '28', '17', '34', '19', '13', '16', '30', '10',  
      '33', '21', '26', '20', '25', '75', '24', '31', '29', '48', '53',  
      '38', '27', '23', '32', '47', '82', '46', '60'], dtype=object)
```

```
# Преобразуем столбец 'floor' в числа, нечисловые значения станут нан
```

```
df['floor'] = pd.to_numeric(df['floor'], errors='coerce')
```

```
# Удаляем строки с нан в столбце 'floor' и меняем флот на инт
```

```
df = df.dropna(subset=['floor'])
```

```
df['floor'] = df['floor'].astype('int')
```

```
# кодируем улицу, метро и тд
```

```
transform = ['street', 'underground', 'object_type', 'district']
```

```
for col in transform:
```

```
    # создание и обучение
```

```
    le = preprocessing.LabelEncoder()
```

```
    df[col] = le.fit_transform(df[col])
```

```
    integer_mapping = {l: i for i, l in enumerate(le.classes_)}
```

```
    print('значения полученные при перекодировке ', col, '-', integer_mapping)
```

```
значения полученные при перекодировке street - {' 1 Мая': 0, ' 1-я Ватутинская': 1, ' 1-я Коммунистическая': 2, ' 1-я Лесная': 3, '  
значения полученные при перекодировке underground - {'Авиамоторная': 0, 'Авиастроительная': 1, 'Автово': 2, 'Автозаводская': 3, 'Ак  
значения полученные при перекодировке object_type - {'Вторичка': 0, 'Вторичка / Апартаменты': 1, 'Вторичка / Пентхаус': 2, 'Новостр  
значения полученные при перекодировке district - {'Авиастроительный': 0, 'Адмиралтейский': 1, 'Акса́й': 2, 'Алексеевский': 3, 'Алуш
```

```
df['floors_count'] = df['floors_count'].astype('int')
```

```
#выводим данные и смотрим на подозрительные выводы
```

```
df['floors_count'].describe()
```



```

floors_count
count    3144.000000
mean      14.166667
std       8.638987
min       1.000000
25%       8.000000
50%      14.000000
75%      18.000000
max       95.000000

```

dtype: float64

```

#омг, ссылки одинаковые
omg = df[df['floors_count'] > 80]
omg['url']

```

```

url
617    https://www.cian.ru/sale/flat/307810501/
923    https://www.cian.ru/sale/flat/307810501/
2984   https://www.cian.ru/sale/flat/301428962/
6688   https://www.cian.ru/sale/flat/307810501/

```

dtype: object

```

#удаляем одинаковые ссылки и заново
df = df.drop_duplicates(subset=['url'], keep='first')
omg = df[df['floors_count'] > 80]
omg['url']
#проверяю ссылки, убеждаюсь что не обман

```

```

url
617    https://www.cian.ru/sale/flat/307810501/
2984   https://www.cian.ru/sale/flat/301428962/

```

dtype: object

```

df = df.drop('url', axis=1)

df['square_price'] = (df['price'] / df['total_meters']).astype(int)

df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Index: 2820 entries, 1 to 8391
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   location              2820 non-null  object  
 1   floor                 2820 non-null  int64   
 2   floors_count          2820 non-null  int64   
 3   rooms_count           2820 non-null  int64   
 4   total_meters          2820 non-null  float64  
 5   price                 2820 non-null  float64  
 6   year_of_construction  2820 non-null  int64   
 7   object_type           2820 non-null  int64   
 8   have_loggia           2820 non-null  object  
 9   parking_type          2820 non-null  object  
10   living_meters         2820 non-null  float64  
11   kitchen_meters        2820 non-null  float64  
12   ceiling_height        2820 non-null  float64  
13   district              2820 non-null  int64   
14   street                2820 non-null  int64   
15   underground           2820 non-null  int64   
16   square_price          2820 non-null  int64   
dtypes: float64(5), int64(9), object(3)
memory usage: 396.6+ KB

```

```
df.head()
```

	location	floor	floors_count	rooms_count	total_meters	price	year_of_construction	object_type	have_loggia	parking_type
1	Серпухов	1	9	1	33.0	4150000.0	1975	0	нет лоджи	нет парковки
4	Серпухов	4	7	1	25.9	3000000.0	2009	0	нет лоджи	Наземная
9	Серпухов	6	9	2	42.7	3990000.0	1980	0	1 лоджия	Наземная
10	Серпухов	1	5	2	42.8	2999000.0	1990	0	1 балкон	Наземная
13	Серпухов	1	3	1	25.3	2500000.0	2015	0	1 лоджия	нет парковки

Далее: [Посмотреть рекомендованные графики](#) [New interactive sheet](#)

✓ **Графики и анализ от чего зависит цена за м²**

✓ **График №1**

На этом графике можно увидеть, что Подземная парковка дороже всех, а остальные +- на равне

```
parking = df.groupby('parking_type')['square_price'].mean().reset_index()
parking
```

	parking_type	square_price
0	Многоуровневая	205065.218391
1	Наземная	169206.042568
2	Открытая	215156.788321
3	Подземная	415238.755418
4	нет парковки	220615.577554

Далее: [Посмотреть рекомендованные графики](#) [New interactive sheet](#)

```
parking_sorted = parking.sort_values(by='square_price', ascending=False)

# Установка размера фигуры
figsize = (12, 1.2 * len(parking_sorted['parking_type'].unique()))
plt.figure(figsize=figsize)

# Создание barplot
sns.barplot(data=parking_sorted, x='square_price', y='parking_type', palette='Dark2', estimator=np.mean)

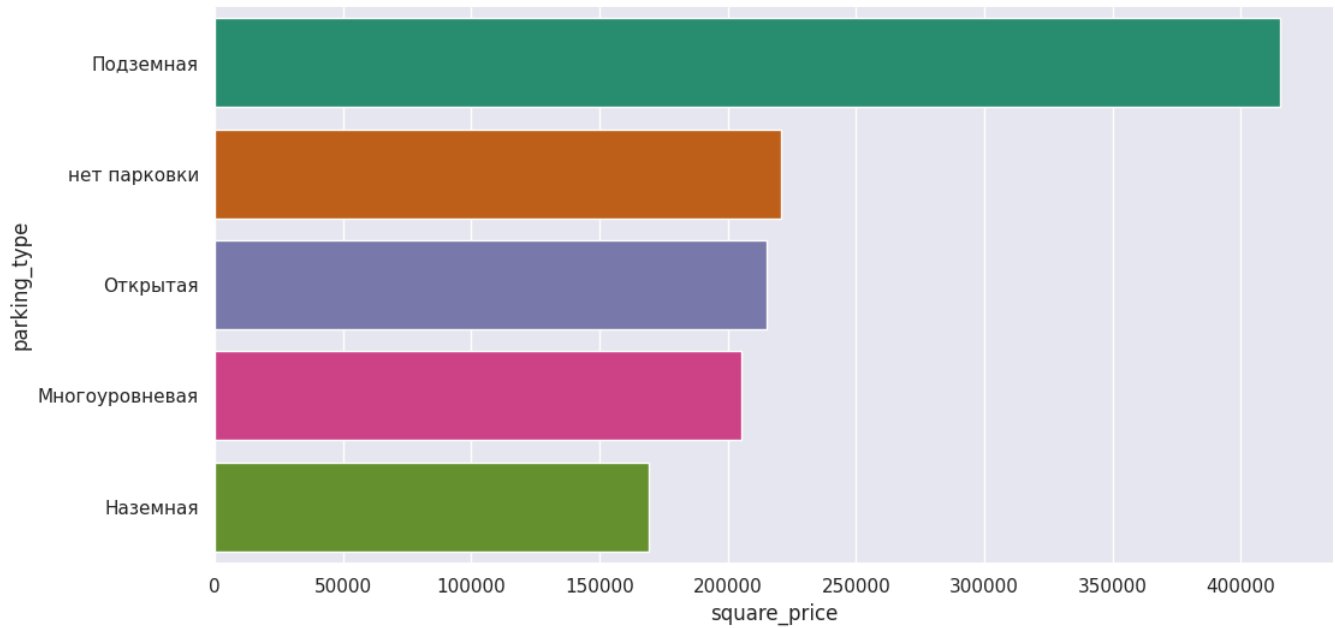
# Удаление лишних осей
sns.despine(top=True, right=True, bottom=True, left=True)

# Вывод графика
plt.show()
```

 <ipython-input-378-f0e5ebaf93f7>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

```
sns.barplot(data=parking_sorted, x='square_price', y='parking_type', palette='Dark2', estimator=np.mean)
```



✓ График №2

На этом графике можно увидеть, что Москва лидирует по цене за метр

```
locations = df.groupby('location')['square_price'].mean().reset_index()
locations
```

 [Показать скрытые выходные данные](#)

Далее: [Посмотреть рекомендованные графики](#) [New interactive sheet](#)

```
top = locations.sort_values(by='square_price', ascending=False).head(5)
```

```
# Устанавливаем размер фигуры
figsize = (12, 1.2 * len(top))
plt.figure(figsize=figsize)
```

```
sns.barplot(data=top, x='square_price', y='location', palette='Dark2', estimator=np.mean)
sns.despine(top=True, right=True, bottom=True, left=True)
```

 <ipython-input-380-9fa66ae80d04>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le`

```
sns.barplot(data=top, x='square_price', y='location', palette='Dark2', estimator=np.mean)
```

Москва

- ✓ Облако(я пыталась)

1. **алудум**

```
import pandas as pd
from collections import Counter
cities = df['location']
```

```
# Подсчет частоты встречаемости каждого города
city_counts = Counter(cities)
```

```
import pandas as pd
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
city_counts = Counter(cities)
```

```
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(city_counts)
```

```
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



- Матрица корреляции

```
def omg(init_df):
    result = init_df.copy()
    encoders = {}
    for column in result.columns:
        if result[column].dtype == object:
            result[column] = result[column].astype(str)
            encoders[column] = preprocessing.LabelEncoder()
            result[column] = encoders[column].fit_transform(result[column])
    return result, encoders
```

```
encoded_data, encoders = omg(df)
```

```
plt.figure(figsize = (15,15))
encoded_data, encoders = omg(df)
```