Министерство цифрового развития
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

# Отчёт

по лабораторной работе № 1 «Первичный анализ и предобработка данных»

Выполнил:

студент группы ИП-216

Андрущенко Ф.А

Работу проверил: Преподаватель
Сороковых Д.А.

Новосибирск 2025 г.

# Введение

Краткое описание выбранного набора данных:

- Название: House-price
- Объём данных: 187,531 запись
- Количество признаков: 21

Постановка задачи:
Выберите набор данных с различными типами признаков и наличием пропусков. Проведите все этапы разведочного анализа (EDA).

# Основная часть

## 1. Загрузка и первичный осмотр

### Скачиваем набор данных в формате .csv

```
import kagglehub

path = kagglehub.dataset_download("juhibhojani/house-price")
print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/juhibhojani/house-price?dataset_version_number=1...
100%|██████████| 6.61M/6.61M [00:00<00:00, 77.3MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/juhibhojani/house-price/versions/1
```

### Загружаем данные в DataFrame и выводим первые 5 строк

```
import os
import pandas as pd


file_path = '/root/.cache/kagglehub/datasets/juhibhojani/house-price/versions/1/house_prices.csv'
df = pd.read_csv(file_path)
df.head()
```

| | Index | Title | Description | Amount(in rupees) | Price (in rupees) | location | Carpet Area | Status | Floor | Transaction | Furnishing | facing |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 BHK Ready to Occupy Flat for sale in Srushti... | Bhiwandi, Thane has an attractive 1 BHK Flat f... | 42 Lac | 6000.0 | thane | 500 sqft | Ready to Move | 10 out of 11 | Resale | Unfurnished | NaN |
| 1 | 1 | 2 BHK Ready to Occupy Flat for sale in Dosti V... | One can find this stunning 2 BHK flat for sale... | 98 Lac | 13799.0 | thane | 473 sqft | Ready to Move | 3 out of 22 | Resale | Semi-Furnished | East |
| 2 | 2 | 2 BHK Ready to Occupy Flat for sale in Sunrise... | Up for immediate sale is a 2 BHK apartment in ... | 1.40 Cr | 17500.0 | thane | 779 sqft | Ready to Move | 10 out of 29 | Resale | Unfurnished | East |
| 3 | 3 | 1 BHK Ready to Occupy Flat for sale Kasheli | This beautiful 1 BHK Flat is available for sal... | 25 Lac | NaN | thane | 530 sqft | Ready to Move | 1 out of 3 | Resale | Unfurnished | NaN |
| 4 | 4 | 2 BHK Ready to Occupy Flat for sale in TenX Ha... | This lovely 2 BHK Flat in Pokhran Road, Thane ... | 1.60 Cr | 18824.0 | thane | 635 sqft | Ready to Move | 20 out of 42 | Resale | Unfurnished | West |

### Используем методы .info(), .describe(), .shape

```
df.shape
```

```
(187531, 21)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187531 entries, 0 to 187530
Data columns (total 21 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Index              187531 non-null  int64  
 1   Title              187531 non-null  object 
 2   Description        184508 non-null  object 
 3   Amount(in rupees)  187531 non-null  object 
 4   Price (in rupees)  169866 non-null  float64
 5   location           187531 non-null  object 
 6   Carpet Area        106858 non-null  object 
 7   Status             186916 non-null  object 
 8   Floor              180454 non-null  object 
 9   Transaction        187448 non-null  object 
 10  Furnishing         184634 non-null  object 
 11  facing             117298 non-null  object 
 12  overlooking        106095 non-null  object 
 13  Society            77853 non-null   object 
 14  Bathroom           186703 non-null  object 
 15  Balcony            138596 non-null  object 
 16  Car Parking        84174 non-null   object 
 17  Ownership          122014 non-null  object 
 18  Super Area         79846 non-null   object 
 19  Dimensions         0 non-null       float64
 20  Plot Area          0 non-null       float64
dtypes: float64(3), int64(1), object(17)
memory usage: 30.0+ MB
```

```
df.describe()
```

| | Index | Price (in rupees) | Dimensions | Plot Area |
|---|---|---|---|---|
| count | 187531.000000 | 1.698660e+05 | 0.0 | 0.0 |
| mean | 93765.000000 | 7.583772e+03 | NaN | NaN |
| std | 54135.681003 | 2.724171e+04 | NaN | NaN |
| min | 0.000000 | 0.000000e+00 | NaN | NaN |
| 25% | 46882.500000 | 4.297000e+03 | NaN | NaN |
| 50% | 93765.000000 | 6.034000e+03 | NaN | NaN |
| 75% | 140647.500000 | 9.450000e+03 | NaN | NaN |
| max | 187530.000000 | 6.700000e+06 | NaN | NaN |

Количество записей: 187531
Количество признаков: 21

Типы признаков:

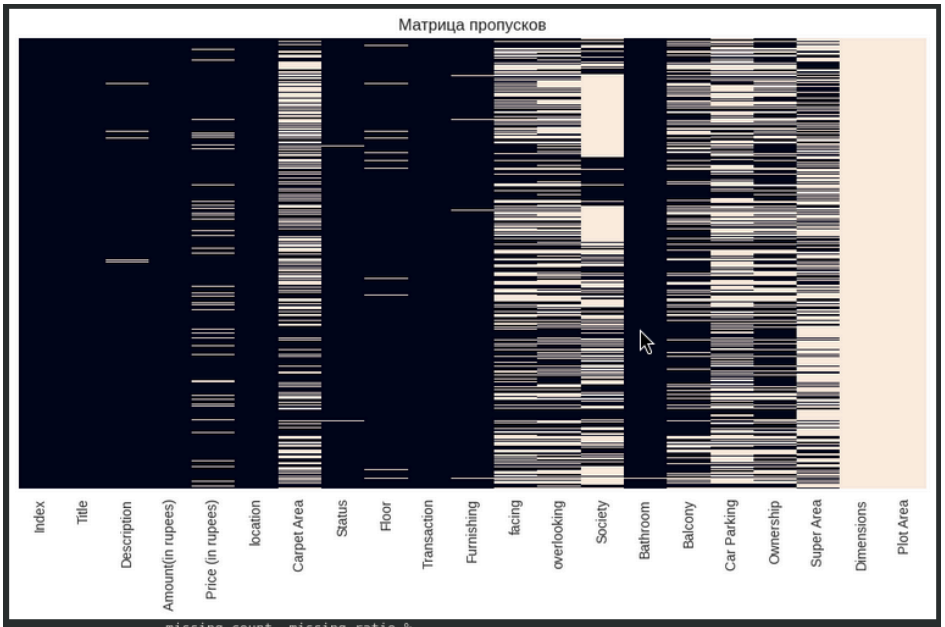
- Числовые (4 признака)
    - Index
    - Price
    - Dimensions
    - Plot Area
- Категориальные (17 признаков) – все имеют тип object

Пропущенные значения:

- Dimensions и Plot Area – 100%
- Super Area – 57%
- Society – 58%
- Car parking – 55%
- Overlooking – 43%
- Facing – 37%
- Balcony – 26%

## 2. Анализ пропусков

```
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar = False, yticklabels = False)
plt.title("Матрица пропусков")
plt.show()

missing_count = df.isnull().sum()
missing_ratio = (missing_count / len(df)) * 100

missing_table = pd.DataFrame({
    'missing_count': missing_count,
    'missing_ratio_%': missing_ratio
})

missing_table = missing_table[missing_table['missing_count'] > 0].sort_values(by='missing_ratio_%', ascending=False)
print(missing_table)
```




| | missing_count | missing_ratio_% |
|---|---|---|
| Plot Area | 187531 | 100.000000 |
| Dimensions | 187531 | 100.000000 |
| Society | 109678 | 58.485264 |
| Super Area | 107685 | 57.422506 |
| Car Parking | 103357 | 55.114621 |
| overlooking | 81436 | 43.425354 |
| Carpet Area | 80673 | 43.018488 |
| facing | 70233 | 37.451408 |
| Ownership | 65517 | 34.936624 |
| Balcony | 48935 | 26.094352 |
| Price (in rupees) | 17665 | 9.419776 |
| Floor | 7077 | 3.773776 |
| Description | 3023 | 1.612000 |
| Furnishing | 2897 | 1.544811 |
| Bathroom | 828 | 0.441527 |
| Status | 615 | 0.327946 |
| Transaction | 83 | 0.044259 |

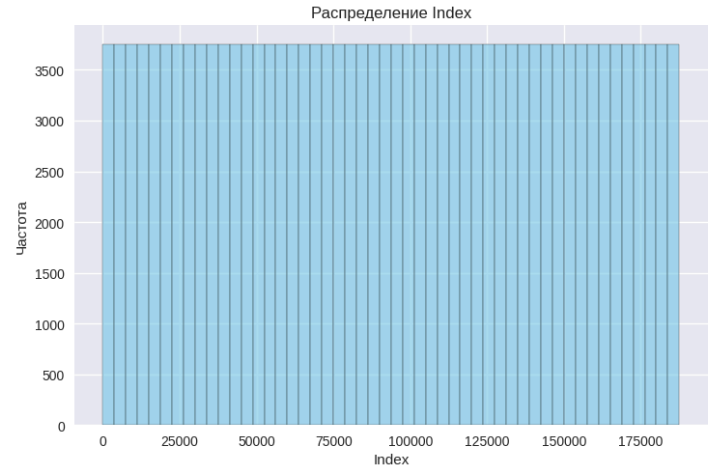
Столбцы с наибольшим процентом пропусков: Dimensions и Plot Area (100%), Super Area (57.4%), Society (58.5%), Car Parking (55.1%)

Стратегия обработки:

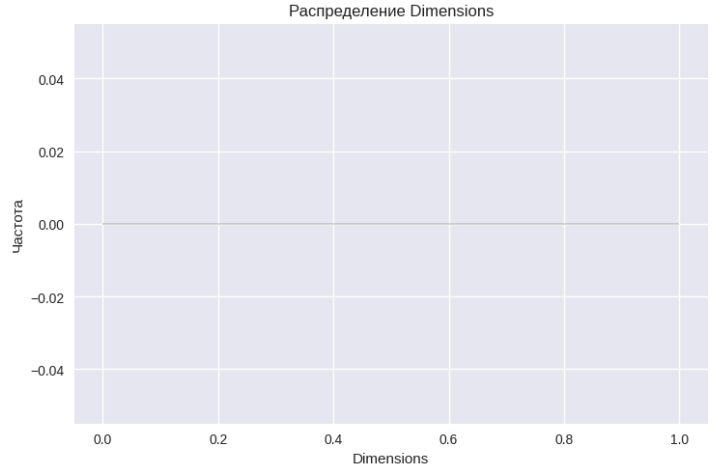
- Удалить полностью пустые столбцы
- Для столбцов с >50% пропусков использовать удаление или заполнение значением «Не указано»
- Для числовых признаков с пропусками заполнить медианной
- Для категориальных признаков заполнить модой

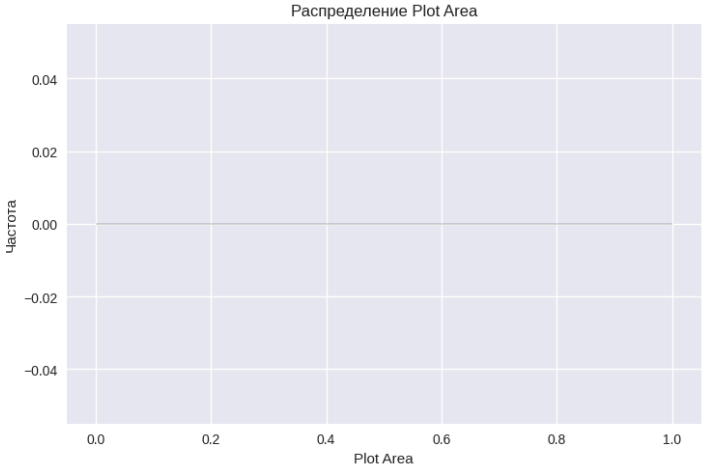
# 3. Анализ числовых признаков

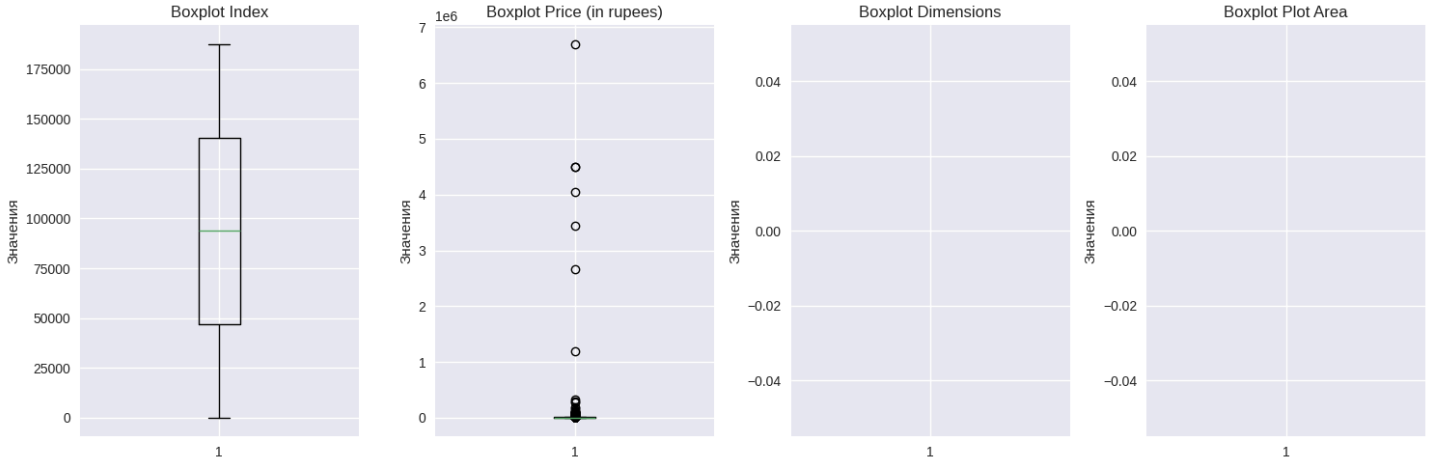
```
numeric_cols = df.select_dtypes(include=np.number).columns

fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.flatten()

for i, col in enumerate(numeric_cols):
    if i < 4:
        axes[i].hist(df[col].dropna(), bins=50, alpha=0.7, color='skyblue', edgecolor='black')
        axes[i].set_title(f'Распределение {col}', fontsize=12)
        axes[i].set_xlabel(col)
        axes[i].set_ylabel('Частота')

plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, len(numeric_cols), figsize=(15, 5))
for i, col in enumerate(numeric_cols):
    axes[i].boxplot(df[col].dropna())
    axes[i].set_title(f'Boxplot {col}')
    axes[i].set_ylabel('Значения')

plt.tight_layout()
plt.show()

stats_df = pd.DataFrame({
    'mean': df[numeric_cols].mean(),
    'median': df[numeric_cols].median(),
    'std': df[numeric_cols].std(),
    'skew': df[numeric_cols].skew()
})
display(stats_df)
```







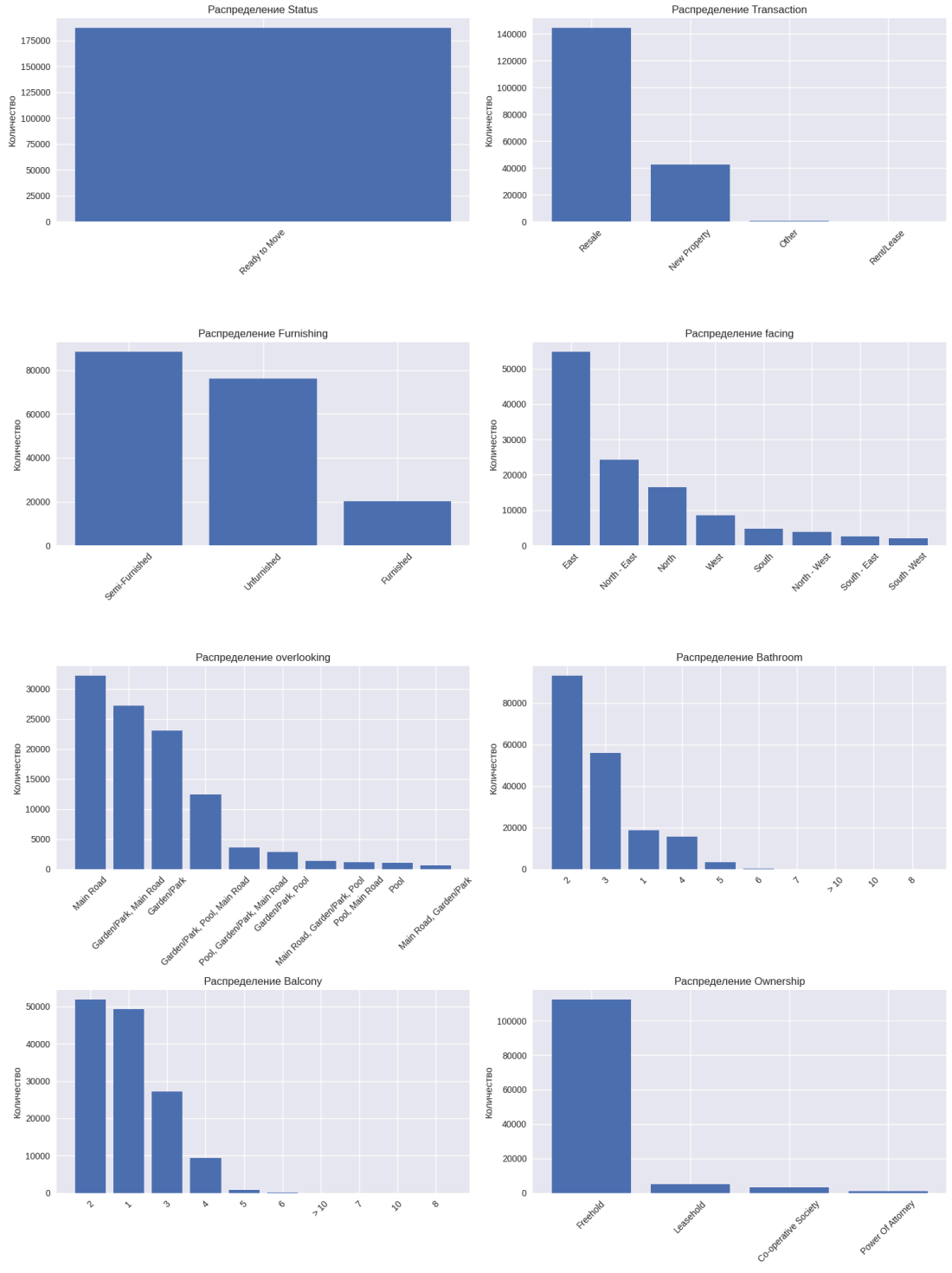



| | mean | median | std | skew |
|---|---|---|---|---|
| **Index** | 93765.000000 | 93765.0 | 54135.681003 | 0.00000 |
| **Price (in rupees)** | 7583.771885 | 6034.0 | 27241.705819 | 177.11337 |
| **Dimensions** | NaN | NaN | NaN | NaN |
| **Plot Area** | NaN | NaN | NaN | NaN |

Распределение цены показывает значительные выбросы (6.7e6 при медиане 6034). Столбец Index равномерно распределён.

# 4. Анализ категориальных признаков

```
categorical_cols = df.select_dtypes(include='object').columns

reasonable_cats = [col for col in categorical_cols if df[col].nunique() <= 20]

fig, axes = plt.subplots(4, 2, figsize=(15, 20))
axes = axes.flatten()

for i, col in enumerate(reasonable_cats[:8]):  # Первые 8 признаков
    value_counts = df[col].value_counts().head(10)  # Топ-10 категорий
    axes[i].bar(value_counts.index.astype(str), value_counts.values)
    axes[i].set_title(f'Распределение {col}', fontsize=12)
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].set_ylabel('Количество')

plt.tight_layout()
plt.show()

print("\nКоличество уникальных категорий:")
print("-" * 35)
for col in categorical_cols:
    unique_count = df[col].nunique()
    print(f"{col}: {unique_count} уникальных значений")
```



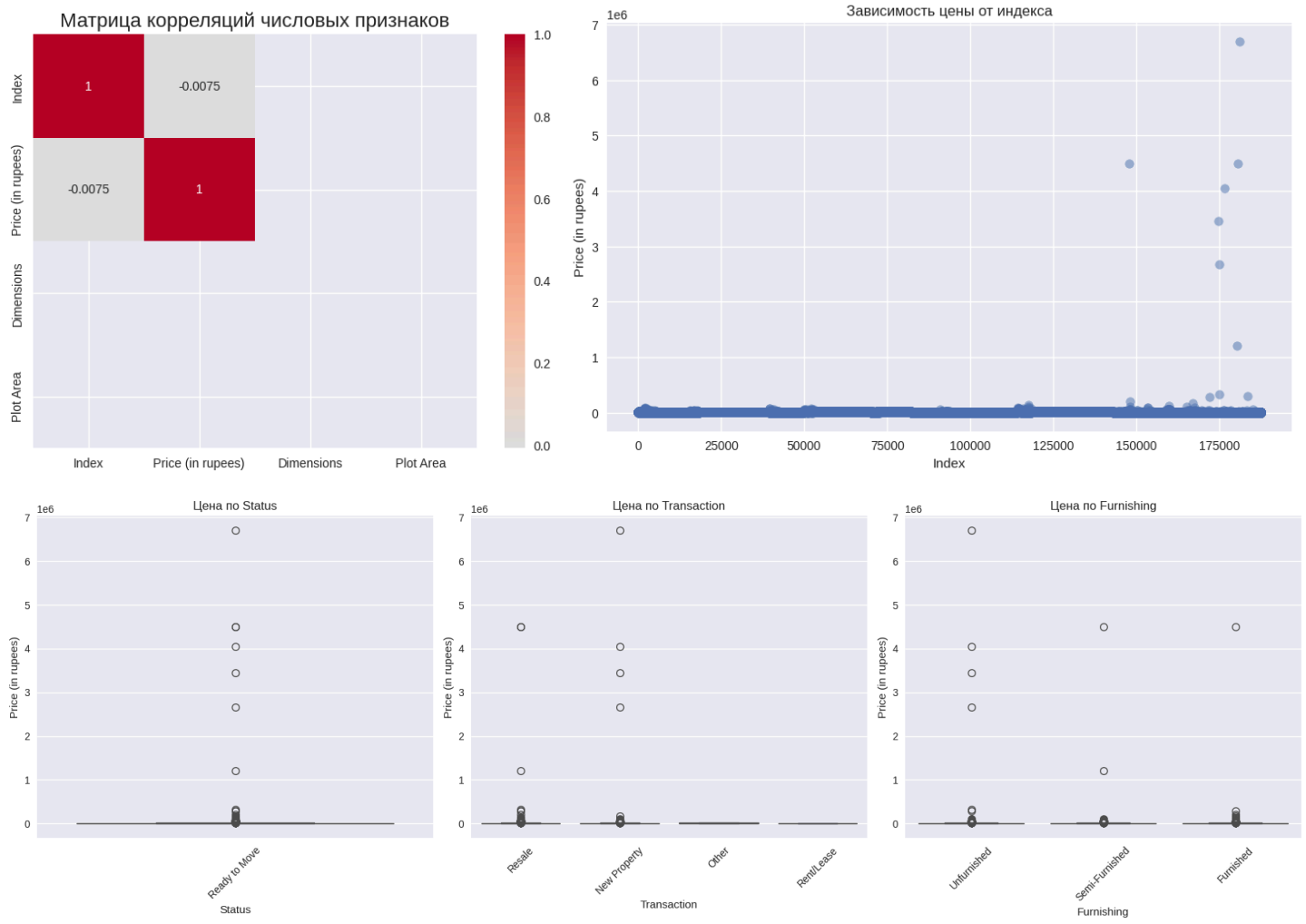
```
Количество уникальных категорий:
----------------------------------
Title: 32446 уникальных значений
Description: 65634 уникальных значений
Amount(in rupees): 1561 уникальных значений
location: 81 уникальных значений
Carpet Area: 2758 уникальных значений
Status: 1 уникальных значений
Floor: 947 уникальных значений
Transaction: 4 уникальных значений
Furnishing: 3 уникальных значений
facing: 8 уникальных значений
overlooking: 19 уникальных значений
Society: 10376 уникальных значений
Bathroom: 11 уникальных значений
Balcony: 11 уникальных значений
Car Parking: 229 уникальных значений
Ownership: 4 уникальных значений
Super Area: 2976 уникальных значений
```

Признаки с высокой кардинальностью:

- Title (высокая уникальность)
- Description
- Location

## 5. Анализ взаимосвязей

```
correlation_matrix = df[numeric_cols].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Матрица корреляций числовых признаков', fontsize=16)
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(df['Index'], df['Price (in rupees)'], alpha=0.5)
plt.title('Зависимость цены от индекса')
plt.xlabel('Index')
plt.ylabel('Price (in rupees)')
plt.show()

cat_for_analysis = [col for col in reasonable_cats if df[col].nunique() <= 5][:3]

fig, axes = plt.subplots(1, len(cat_for_analysis), figsize=(18, 6))
for i, col in enumerate(cat_for_analysis):
    sns.boxplot(x=df[col], y=df['Price (in rupees)'], ax=axes[i])
    axes[i].set_title(f'Цена по {col}')
    axes[i].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```




Корреляция между числовыми признаками слабая из-за малого количества. Зависимость цены от индекса не прослеживается. Boxplot показывают различия в распределении цен по категориям.

## 6. Базовая предобработка

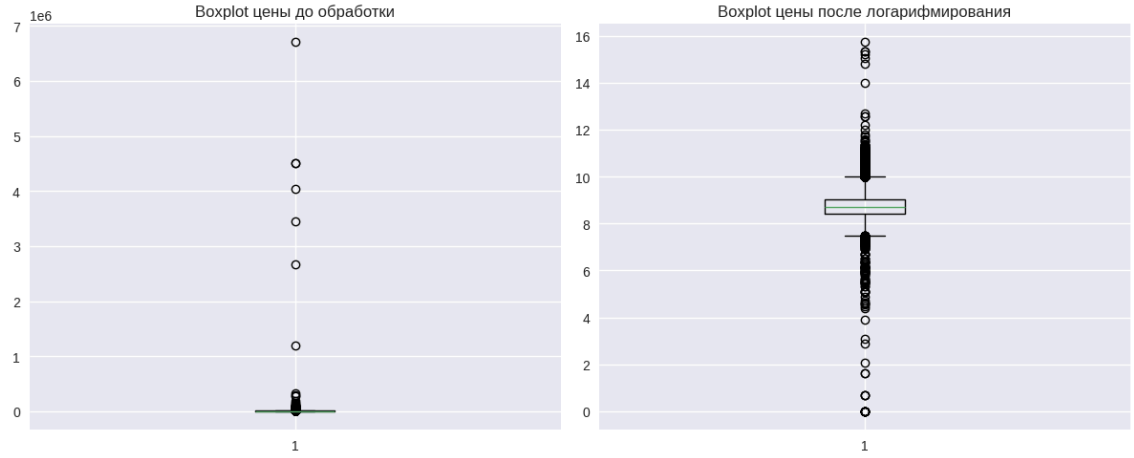
```
print("ИСХОДНЫЕ НАЗВАНИЯ СТОЛБЦОВ:")
print(df.columns.tolist())
print()

df.columns = df.columns.str.lower().str.replace(' ', '_').str.replace('(', '').str.replace(')', '')
print("НОВЫЕ НАЗВАНИЯ СТОЛБЦОВ (после очистки):")
print(df.columns.tolist())
print("\n" + "="*80)

print("АНАЛИЗ ПРОПУСКОВ ДО ОБРАБОТКИ:")
missing_before = df.isnull().sum()
print(missing_before[missing_before > 0])
print()

empty_columns = []
for col in df.columns:
    if df[col].isnull().sum() == len(df):
        empty_columns.append(col)
        print(f"Обнаружен полностью пустой столбец: {col}")

if empty_columns:
    print(f"УДАЛЯЕМ ПОЛНОСТЬЮ ПУСТЫЕ СТОЛБЦЫ: {empty_columns}")
    df = df.drop(empty_columns, axis=1)
    print(f"Размерность после удаления пустых столбцов: {df.shape}")
else:
    print("Полностью пустых столбцов не обнаружено")
print()

numeric_cols_updated = df.select_dtypes(include=[np.number]).columns.tolist()
print("ЧИСЛОВЫЕ ПРИЗНАКИ ДЛЯ ЗАПОЛНЕНИЯ МЕДИАНОЙ:")
for col in numeric_cols_updated:
    missing_count = df[col].isnull().sum()
    if missing_count > 0:
        median_val = df[col].median()
        print(f"  {col}: {missing_count} пропусков → заполняем медианой ({median_val:.2f})")
        df[col] = df[col].fillna(median_val)
    else:
        print(f"  {col}: пропусков нет")
print()

categorical_cols_updated = df.select_dtypes(include=['object']).columns.tolist()
print("КАТЕГОРИАЛЬНЫЕ ПРИЗНАКИ ДЛЯ ЗАПОЛНЕНИЯ МОДОЙ:")
for col in categorical_cols_updated:
    missing_count = df[col].isnull().sum()
    if missing_count > 0:
        mode_val = df[col].mode()[0] if not df[col].mode().empty else 'Unknown'
        print(f"  {col}: {missing_count} пропусков → заполняем модой ('{mode_val}')")
        df[col] = df[col].fillna(mode_val)
    else:
        print(f"  {col}: пропусков нет")
print()

print("ПРОВЕРКА ПОСЛЕ ЗАПОЛНЕНИЯ ПРОПУСКОВ:")
missing_after = df.isnull().sum()
remaining_missing = missing_after[missing_after > 0]
if len(remaining_missing) > 0:
    print("Осталось пропусков:")
    print(remaining_missing)
else:
    print("Все пропуски успешно заполнены!")
print("\n" + "="*80)
```

```
ИСХОДНЫЕ НАЗВАНИЯ СТОЛБЦОВ:
['Index', 'Title', 'Description', 'Amount(in rupees)', 'Price (in rupees)',
'location', 'Carpet Area', 'Status', 'Floor', 'Transaction', 'Furnishing',
'facing', 'overlooking', 'Society', 'Bathroom', 'Balcony', 'Car Parking',
'Ownership', 'Super Area', 'Dimensions', 'Plot Area']
НОВЫЕ НАЗВАНИЯ СТОЛБЦОВ (после очистки):
['index', 'title', 'description', 'amountin_rupees', 'price_in_rupees',
'location', 'carpet_area', 'status', 'floor', 'transaction', 'furnishing',
'facing', 'overlooking', 'society', 'bathroom', 'balcony', 'car_parking',
'ownership', 'super_area', 'dimensions', 'plot_area']
================================================================================
АНАЛИЗ ПРОПУСКОВ ДО ОБРАБОТКИ:
description          3023
price_in_rupees     17665
carpet_area         80673
status                615
floor                7077
transaction            83
```

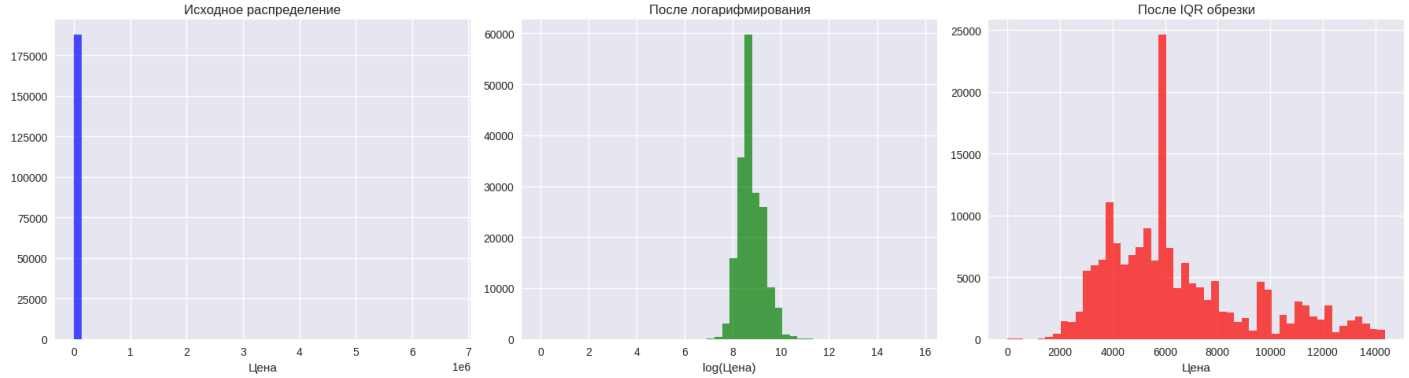
```
furnishing           2897
facing              70233
overlooking         81436
society            109678
bathroom              828
balcony             48935
car_parking        103357
ownership           65517
super_area         107685
dimensions         187531
plot_area          187531
dtype: int64

Обнаружен полностью пустой столбец: dimensions
Обнаружен полностью пустой столбец: plot_area
УДАЛЯЕМ ПОЛНОСТЬЮ ПУСТЫЕ СТОЛБЦЫ: ['dimensions', 'plot_area']
Размерность после удаления пустых столбцов: (187531, 19)

ЧИСЛОВЫЕ ПРИЗНАКИ ДЛЯ ЗАПОЛНЕНИЯ МЕДИАНОЙ:
  index: пропусков нет
  price_in_rupees: 17665 пропусков → заполняем медианой (6034.00)

КАТЕГОРИАЛЬНЫЕ ПРИЗНАКИ ДЛЯ ЗАПОЛНЕНИЯ МОДОЙ:
  title: пропусков нет
  description: 3023 пропусков → заполняем модой ('Multistorey apartment is
available for sale. It is a good location property. Please contact for more
details.')
  amountin_rupees: пропусков нет
  location: пропусков нет
  carpet_area: 80673 пропусков → заполняем модой ('1000 sqft')
  status: 615 пропусков → заполняем модой ('Ready to Move')
  floor: 7077 пропусков → заполняем модой ('2 out of 4')
  transaction: 83 пропусков → заполняем модой ('Resale')
  furnishing: 2897 пропусков → заполняем модой ('Semi-Furnished')
  facing: 70233 пропусков → заполняем модой ('East')
  overlooking: 81436 пропусков → заполняем модой ('Main Road')
  society: 109678 пропусков → заполняем модой ('Hamdam Apartment')
  bathroom: 828 пропусков → заполняем модой ('2')
  balcony: 48935 пропусков → заполняем модой ('2')
  car_parking: 103357 пропусков → заполняем модой ('1 Covered')
  ownership: 65517 пропусков → заполняем модой ('Freehold')
  super_area: 107685 пропусков → заполняем модой ('1100 sqft')

ПРОВЕРКА ПОСЛЕ ЗАПОЛНЕНИЯ ПРОПУСКОВ:
Все пропуски успешно заполнены!
================================================================================
```

# 7. Обработка выбросов

```
price_data = df['price_in_rupees'].dropna()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.boxplot(price_data)
plt.title('Boxplot цены до обработки')

price_log = np.log1p(price_data)

plt.subplot(1, 2, 2)
plt.boxplot(price_log)
plt.title('Boxplot цены после логарифмирования')
plt.tight_layout()
plt.show()

Q1 = price_data.quantile(0.25)
Q3 = price_data.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

price_trimmed = price_data[(price_data >= lower_bound) & (price_data <= upper_bound)]

print(f"До обработки: {len(price_data)} записей")
print(f"После IQR обрезки: {len(price_trimmed)} записей")
print(f"Удалено выбросов: {len(price_data) - len(price_trimmed)}")

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
axes[0].hist(price_data, bins=50, alpha=0.7, color='blue')
axes[0].set_title('Исходное распределение')
axes[0].set_xlabel('Цена')

axes[1].hist(price_log, bins=50, alpha=0.7, color='green')
axes[1].set_title('После логарифмирования')
axes[1].set_xlabel('log(Цена)')

axes[2].hist(price_trimmed, bins=50, alpha=0.7, color='red')
axes[2].set_title('После IQR обрезки')
axes[2].set_xlabel('Цена')

plt.tight_layout()
plt.show()
```




До обработки: 187531 записей
После IQR обрезки: 177086 записей
Удалено выбросов: 10445



# Заключение

1. Проблема с пропущенными значениями
   - В данном наборе данных оказалось 2 полностью пустых столбца dimensions и plot_area.
   - Также был высокий уровень пропусков в ключевых признаках super_area, society и car_parking. В признаке price_in_rupees были умеренные пропуски.

2. Проблема выбросов
   - Сильные выбросы в ценах: максимальное значение 6,700,000 при медиане ~6000.

3. Проблема высокой кардинальности
   - Категориальные признаки с большим количеством уникальных значений: title, description, location.

В данной работе было использовано:
- Удаление полностью пустых столбцов - исключение бесполезных признаков
- Заполнение числовых пропусков медианой – устойчивость к выбросам
- Заполнение категориальных пропусков модой – сохранение наиболее частых значений
- Обработка выбросов осуществлялась с помощью логарифмического преобразования для нормализации распределения цен, а также IQR-обрезка.

Влияние на дальнейшее построение моделей:
- Улучшение качества данных при помощи устранения пропусков и обработки выбросов
- Повышение эффективности моделей
- Улучшение интерпретируемости

Ссылка на датасет:
https://www.kaggle.com/datasets/juhibhojani/house-price

Ссылка на Google Colab:
https://colab.research.google.com/drive/1IVv42XKMGGsQ7Rj56u_p9Qv1-81yY2vb?usp=sharing