

Анализ ценовых факторов на рынке смартфонов в 2025 году

Описание проекта

Данный проект реализуется в рамках портфолио для демонстрации на реальных данных навыков EDA, визуализации, работы с валютами и проверки гипотез.

Для данного проекта в поле анализа выбран датасет по продажам мобильных телефонов(смартфонов) в различных регионах с платформы Kaggle

- рынок смартфонов — один из самых динамично развивающихся, его стоимость \$X млрд и анализ цен/характеристик важен как для компаний (стратегии продаж), так и для покупателей (оптимизация выбора)

Этот датасет содержит подробные характеристики и официальные стартовые цены продаж различных моделей мобильных телефонов от разных компаний. Он дает представление о аппаратных характеристиках смартфонов, ценовых тенденциях и конкурентоспособности брендов в разных странах. В датасете представлены такие ключевые характеристики, как объем оперативной памяти (RAM), характеристики камеры, емкость аккумулятора, характеристики процессора и размер экрана.

Один из важных аспектов этого датасета — информация о ценах. Указанные цены представляют собой официальные стартовые цены мобильных телефонов на момент их первого выхода на рынок. Цены варьируются в зависимости от страны и периода запуска: старые модели отображают их первоначальные стартовые цены, а новые — самые актуальные на момент выхода. Это делает датасет ценным для изучения динамики цен со временем и сравнения доступности смартфонов в разных регионах.

Цель проекта — провести исследование факторов, влияющих на цену смартфонов, и выявить региональные различия в стоимости устройств. Это позволит понять, какие характеристики наиболее ценятся производителями, и где смартфоны наиболее доступны. Также определить, в какой стране покупать телефон наиболее выгодно для перепродажи

Описание данных

Дата-сет "Mobiles Dataset (2025)" мобильных телефонов взят с сайта <https://www.kaggle.com/datasets/abdulmalik1518/mobiles-dataset-2025>.

Характеристики:

- Company Name:** Бренд или производитель мобильного телефона.
- Model Name:** Конкретная модель смартфона.
- Mobile Weight:** Вес мобильного телефона (в граммах).
- RAM:** Объем оперативной памяти (ОЗУ) мобильного телефона (в гигабайтах).
- Front Camera:** Разрешение фронтальной (селфи) камеры (в мегапикселях).
- Back Camera:** Разрешение основной задней камеры (в мегапикселях).
- Processor:** Чипсет или процессор, используемый в устройстве.
- Battery Capacity:** Размер аккумулятора мобильного телефона (в мА·ч).
- Screen Size:** Диагональ дисплея мобильного телефона (в дюймах).
- Launched Price:** (Пакистан, Индия, Китай, США, Дубай): Официальная стартовая цена мобильного телефона в соответствующей стране на момент его выпуска. Цены варьируются в зависимости от года запуска мобильного телефона.
- Launched Year:** Год официального выпуска мобильного телефона.

Цели исследования

- Оценить ключевые параметры, влияющие на цену смартфона:
 - RAM, камеры, процессор, экран, батарея, бренд, год выпуска.
- Определить, в какой стране смартфоны наиболее доступны:
 - Сравнить уровень цен в USD в разных странах и выявить закономерности
 - Учет медианной зарплаты на душу населения
 - Создать метрику "доступности": цена телефона / медианская месячная зарплата
- Определить, в какой стране покупать телефон наиболее выгодно для перепродажи
- Сравнить бренды по средней цене и характеристикам.
 - Определить ТОП-бренды по соотношению "цена/характеристики"
- Проверить гипотезы о различиях цен в странах

Ход исследования

- Загрузка данных
 - загрузка данных
 - первичные выводы о датасете
- Предобработка данных
 - переименование столбцов в snake_case
 - поиск артефактов в данных
 - преобразование данных в нужные типы: float, int

- обработка проупсов при необходимости: удаление, заполнение
- поиск и удаление дубликатов
- дополнительные агрегации если необходимо
- пересчт цен телефонов в одну валюту USD: добавить курс валют (взять с сайта)
- добавить данные о средней зарплате по странам для оценки доступности
- создание новых признаков:
 - категории цен смартфонов (бюджетный, средний, флагманский)
 - категории RAM (низкий, средний, высокий)
 - соотношение RAM/цена
 - добавить столбец с "возрастом" телефона (2025 - launch_year)
 - создать метрику "доступности": цена телефона / медианная месячная зарплата

3. Исследовательский анализ данных

- количественные переменные
 - гистограммы распределения
 - ящик с усами для сравнения по странам
 - описательная статистика
 - при необходимости обработка выбросов
- категориальные переменные
 - частотные таблицы
 - столбчатые диаграммы
 - круговые диаграммы для долей брендов
- heatmap корреляций между характеристиками и ценой
- Scatter plots для пар переменных с высокой корреляцией
- Grouped bar charts для сравнения цен по странам
- pairplot для визуального анализа связей: зависимость цены от RAM, размера экрана, батареи, камеры
- тепловая карта средних цен по бренду и стране
- группировка и агрегация
 - Средние и медианные цены мобильных устройств по странам
 - Средние и медианные доступность смартфонов по странам
 - Тепловая карта средних и медианных цен по бренду и стране
 - Тепловая карта доступности по бренду и стране
 - Средние характеристики (RAM, мощность аккумулятора, размер дисплея и хара-ка камеры) по ценовым категориям
 - ТОП-10 самых дорогих/дешёвых моделей мобильных устройств для каждой страны
 - ТОП-5 брендов мобильных устройств по средней цене для каждой страны
 - ТОП-5 брендов мобильных устройств по доступности для каждой страны
 - Распределение брендов мобильных устройств по годам и брендам
 - Средняя цена мобильных устройств по странам с группировкой по годам
- Анализ цен для перепродажи в другой стране

4. Составление портрета пользователя каждого региона

- Выявить, какие телефоны чаще всего представлены в каждой стране: ТОП-брэнды по странам по количеству моделей или по средней цене
- Средние/медианные значения цена, RAM, камера, экран, емкость батареи — по странам
- Какие характеристики преобладают в каждом регионе, какой тип телефонов предпочитают
- Визуализация "паспорт устройства" для каждой страны, где указаны средние/медианные значения: Экран, RAM, Камера, Батарея, Цена
- Сравнить бренды-лидеры по регионам
- Найти топ-10 телефонов с максимальной разницей цены в двух странах
- Сравнение affordability: цена смартфона / медианная зарплата

5. Проверка гипотез

- Гипотеза 1: Цены на телефоны различаются в разных странах
- Гипотеза 2: Средние цены в США выше, чем в Индии
- Гипотеза 3: В Дубае цена на премиальные модели выше, чем в США
- Гипотеза 4: Существуют телефоны со значительной разницей цены между странами
- Гипотеза 5: Смартфоны с большим RAM стоят дороже
- Гипотеза 6: Разница в цене между странами зависит от бренда

6. Выводы по проекту

- Какие характеристики влияют на цену
- Где выгоднее покупать смартфоны, в том числе для дальнейшей перепродажи
- Какие бренды предлагают лучшее соотношение цена/характеристики
- Рекомендации: для аналитики рынка или для потребителей

Ожидаемые результаты

- Интерактивные графики, показывающие распределение цен и характеристик.
- Таблицы с топ-10 моделей по разнице цен.
- Визуальный портрет "среднего смартфона" по странам.
- Инсайты о ценах и характеристиках
- Выявлены страны, где покупать телефоны наиболее выгодно для перепродажи

Содержание:

- Шаг 1: Загрузка данных
- Шаг 2: Предобработка данных
- Шаг 3: Исследовательский анализ данных
- Шаг 4: Составление портрета пользователя каждого региона
- Шаг 5: Проверка гипотез
- Шаг 6: Выводы по проекту

Выполнение проекта

Технический стек

- Проект написан на Python с использованием следующих библиотек:

```
In [1]: # 📁 Работа с файлами и API
import os, zipfile, requests, re
from kaggle.api.kaggle_api_extended import KaggleApi
from charset_normalizer import from_path

# 📊 Базовые библиотеки анализа данных
import pandas as pd
import numpy as np

# 🖥️ Визуализация
import matplotlib.pyplot as plt
import seaborn as sns

# 🔎 Корреляционный анализ
import phik
from phik import phik_matrix

# 💼 Статистика
from scipy.stats import (
    f_oneway, ttest_ind, pearsonr, spearmanr,
    mannwhitneyu
)
import scipy.stats as ss # для доступа к остальным методам

# 📈 Финансовые данные
import yfinance as yf

# 🏁 Комбинаторика
from itertools import combinations

# 🚗 Машинное обучение
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# 🌟 Метрики качества
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

Шаг 1: Загрузка данных

```
In [2]: # 🔎 Определение кодировки файла
def detect_encoding(file_path):
    result = from_path(file_path).best()
    if result is None:
        print("⚠️ Кодировка не определена, используем utf-8 по умолчанию.")
        return "utf-8"
    print(f"🔍 Определена кодировка: {result.encoding}")
    return result.encoding

# 📁 Загрузка и первичный анализ CSV-файла
def process_dataframe(file_name, data_dir="datasets", sep=",", decimal="."):
    local_path = os.path.join(data_dir, file_name) # путь к локальному файлу
    url_path = f'https://code.net/datasets/{file_name}' # пример URL для загрузки

    if os.path.exists(local_path):
        print("📁 Загружаем локальный файл...")
        encoding = detect_encoding(local_path) # определяем кодировку
        print("📁 Абсолютный путь к файлу:", os.path.abspath(local_path))
        df = pd.read_csv(local_path, sep=sep, decimal=decimal, encoding=encoding)
    else:
        print("🌐 Локальный файл не найден, проверяем URL...")
        response = requests.get(url_path)
        if response.status_code == 200:
            print("✅ Файл найден по URL, загружаем...")
            df = pd.read_csv(url_path, sep=sep, decimal=decimal, encoding="utf-8")
        else:
            print("❌ Ошибка: файл не найден ни локально, ни по URL.")
            return None
```

```

# 📈 Первичный анализ датафрейма
print("\n🔍 Первые 5 строк датафрейма:")
display(df.head())
print("\n🔍 Любые 5 строк датафрейма:")
display(df.sample(5))
print("\n📅 Последние 5 строк датафрейма:")
display(df.tail())

print("\n💡 Информация о датафрейме:")
df.info()
print("\n📐 Размер датафрейма:")
print(df.shape)
print("\n📝 Названия столбцов:")
print(df.columns)

return df

# 🛡️ Скачивание и обработка датасета с Kaggle
def fetch_and_process_kaggle_dataset(dataset_slug, target_csv_name, data_dir="datasets"):
    os.makedirs(data_dir, exist_ok=True) # создаем папку для датасетов, если её нет

    # 🔑 Инициализация Kaggle API
    api = KaggleApi()
    api.authenticate()

    print("⬇️ Скачиваем архив с Kaggle...")
    api.dataset_download_files(dataset_slug, path=data_dir, unzip=False)

    # 🔎 Поиск zip-файла в папке
    zip_files = [f for f in os.listdir(data_dir) if f.endswith(".zip")]
    if not zip_files:
        print("❌ Архив не найден.")
        return None

    zip_path = os.path.join(data_dir, zip_files[0])

    print("📦 Распаковываем архив...")
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(data_dir)

    # 🔎 Проверка наличия нужного CSV
    target_path = os.path.join(data_dir, target_csv_name)
    if not os.path.exists(target_path):
        print(f"❌ CSV-файл '{target_csv_name}' не найден после распаковки.")
        return None

    # 📈 Загружаем и анализируем CSV
    return process_dataframe(target_csv_name, data_dir=data_dir)

```

```
In [3]: df = fetch_and_process_kaggle_dataset(
    dataset_slug="abdulmalik1518/mobiles-dataset-2025",
    target_csv_name="Mobiles Dataset (2025).csv",
    data_dir=r"C:\Users\HP\my_data\kaggle_datasets\01_Kaggle_Price-phone-analysis-master"
)
```

⬇️ Скачиваем архив с Kaggle...
Dataset URL: <https://www.kaggle.com/datasets/abdulmalik1518/mobiles-dataset-2025>
📦 Распаковываем архив...
📁 Загружаем локальный файл...
🔍 Определена кодировка: cp1250
📁 Абсолютный путь к файлу: C:\Users\HP\my_data\kaggle_datasets\01_Kaggle_Price-phone-analysis-master\Mobiles Dataset (2025).csv

🔍 Первые 5 строк датафрейма:

	Company Name	Model Name	Mobile Weight	RAM	Front Camera	Back Camera	Processor	Battery Capacity	Screen Size	Launched Price (Pakistan)	Launched Price (India)	Launched Price (China)	Launched Price (USA)	Launched Price (Dubai)	Launched Year
0	Apple	iPhone 16 128GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh	6.1 inches	PKR 224,999	INR 79,999	CNY 5,799	USD 799	AED 2,799	2024
1	Apple	iPhone 16 256GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh	6.1 inches	PKR 234,999	INR 84,999	CNY 6,099	USD 849	AED 2,999	2024
2	Apple	iPhone 16 512GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh	6.1 inches	PKR 244,999	INR 89,999	CNY 6,499	USD 899	AED 3,199	2024
3	Apple	iPhone 16 Plus 128GB	203g	6GB	12MP	48MP	A17 Bionic	4,200mAh	6.7 inches	PKR 249,999	INR 89,999	CNY 6,199	USD 899	AED 3,199	2024
4	Apple	iPhone 16 Plus 256GB	203g	6GB	12MP	48MP	A17 Bionic	4,200mAh	6.7 inches	PKR 259,999	INR 94,999	CNY 6,499	USD 949	AED 3,399	2024

🔍 Любые 5 строк датафрейма:

	Company Name	Model Name	Mobile Weight	RAM	Front Camera	Back Camera	Processor	Battery Capacity	Screen Size	Launched Price (Pakistan)	Launched Price (India)	Launched Price (China)	Launched Price (USA)	Launched Price (Dubai)	Laur
391	Oppo	Reno7 5G 256GB	173g	12GB	32MP	64MP + 8MP + 2MP	Dimensity 900	4,500mAh	6.4 inches	PKR 89,999	INR 32,999	CNY 2,199	USD 399	AED 1,499	
45	Apple	iPhone 13 Pro Max 128GB	238g	6GB	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	4,352mAh	6.7 inches	PKR 324,999	INR 129,900	CNY 8,299	USD 1,099	AED 4,199	
649	Huawei	Nova 11 Ultra	188g	12GB	60MP (ultrawide) + 8MP (telephoto)	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh	6.78 inches	PKR 129,999	INR 64,999	CNY 3,999	USD 699	AED 2,499	
321	iQOO	Pad 128GB	520g	6GB	8MP	13MP	Snapdragon 870	8040mAh	11 inches	PKR 69,999	INR 37,999	CNY 2,699	USD 349	AED 1,299	
646	Huawei	Mate 60 Pro+	235g	12GB	13MP	50MP (wide) + 48MP (ultrawide) + 48MP (telephoto)	Kirin 9000S	5,200mAh	6.82 inches	PKR 249,999	INR 149,999	CNY 8,999	USD 1,499	AED 5,199	

➡ Последние 5 строк датафрейма:

	Company Name	Model Name	Mobile Weight	RAM	Front Camera	Back Camera	Processor	Battery Capacity	Screen Size	Launched Price (Pakistan)	Launched Price (India)	Launched Price (China)	Launched Price (USA)	Launched Price (Dubai)	Launched Year
925	Poco	Pad 5G 128GB	571g	8GB	8MP	8MP	Snapdragon 7s Gen 2	10,000mAh	12.1 inches	PKR 66,220	INR 23,999	CNY 2,099	USD 280	AED 1,029	2024
926	Poco	Pad 5G 256GB	571g	8GB	8MP	8MP	Snapdragon 7s Gen 2	10,000mAh	12.1 inches	PKR 71,220	INR 25,999	CNY 2,299	USD 300	AED 1,099	2024
927	Samsung	Galaxy Z Fold6 256GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh	7.6 inches	PKR 604,999	INR 164,999	₹13,999	USD 1,899	AED 7,199	2024
928	Samsung	Galaxy Z Fold6 512GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh	7.6 inches	PKR 544,999	INR 176,999	CNY 15,999	USD 1719	AED 7,699	2024
929	Samsung	Galaxy Z Fold6 1TB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh	7.6 inches	Not available	INR 200,999	CNY 17,999	USD 2,259	AED 8,699	2024

➡ Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 930 entries, 0 to 929
```

Data columns (total 15 columns):

#	Column	Non-Null Count	Dtype
0	Company Name	930 non-null	object
1	Model Name	930 non-null	object
2	Mobile Weight	930 non-null	object
3	RAM	930 non-null	object
4	Front Camera	930 non-null	object
5	Back Camera	930 non-null	object
6	Processor	930 non-null	object
7	Battery Capacity	930 non-null	object
8	Screen Size	930 non-null	object
9	Launched Price (Pakistan)	930 non-null	object
10	Launched Price (India)	930 non-null	object
11	Launched Price (China)	930 non-null	object
12	Launched Price (USA)	930 non-null	object
13	Launched Price (Dubai)	930 non-null	object
14	Launched Year	930 non-null	int64

dtypes: int64(1), object(14)

memory usage: 109.1+ KB

➡ Размер датафрейма:

(930, 15)

➡ Названия столбцов:

```
Index(['Company Name', 'Model Name', 'Mobile Weight', 'RAM', 'Front Camera',
       'Back Camera', 'Processor', 'Battery Capacity', 'Screen Size',
       'Launched Price (Pakistan)', 'Launched Price (India)',
       'Launched Price (China)', 'Launched Price (USA)',
       'Launched Price (Dubai)', 'Launched Year'],
      dtype='object')
```

In [4]: df.iloc[:, :8]

Out[4]:	Company Name	Model Name	Mobile Weight	RAM	Front Camera	Back Camera	Processor	Battery Capacity
0	Apple	iPhone 16 128GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh
1	Apple	iPhone 16 256GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh
2	Apple	iPhone 16 512GB	174g	6GB	12MP	48MP	A17 Bionic	3,600mAh
3	Apple	iPhone 16 Plus 128GB	203g	6GB	12MP	48MP	A17 Bionic	4,200mAh
4	Apple	iPhone 16 Plus 256GB	203g	6GB	12MP	48MP	A17 Bionic	4,200mAh
...
925	Poco	Pad 5G 128GB	571g	8GB	8MP	8MP	Snapdragon 7s Gen 2	10,000mAh
926	Poco	Pad 5G 256GB	571g	8GB	8MP	8MP	Snapdragon 7s Gen 2	10,000mAh
927	Samsung	Galaxy Z Fold6 256GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
928	Samsung	Galaxy Z Fold6 512GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
929	Samsung	Galaxy Z Fold6 1TB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh

930 rows × 8 columns

In [5]:	df.iloc[:, [0, 1] + list(range(8, 15))]								
Out[5]:	Company Name	Model Name	Screen Size	Launched Price (Pakistan)	Launched Price (India)	Launched Price (China)	Launched Price (USA)	Launched Price (Dubai)	Launched Year
0	Apple	iPhone 16 128GB	6.1 inches	PKR 224,999	INR 79,999	CNY 5,799	USD 799	AED 2,799	2024
1	Apple	iPhone 16 256GB	6.1 inches	PKR 234,999	INR 84,999	CNY 6,099	USD 849	AED 2,999	2024
2	Apple	iPhone 16 512GB	6.1 inches	PKR 244,999	INR 89,999	CNY 6,499	USD 899	AED 3,199	2024
3	Apple	iPhone 16 Plus 128GB	6.7 inches	PKR 249,999	INR 89,999	CNY 6,199	USD 899	AED 3,199	2024
4	Apple	iPhone 16 Plus 256GB	6.7 inches	PKR 259,999	INR 94,999	CNY 6,499	USD 949	AED 3,399	2024
...
925	Poco	Pad 5G 128GB	12.1 inches	PKR 66,220	INR 23,999	CNY 2,099	USD 280	AED 1,029	2024
926	Poco	Pad 5G 256GB	12.1 inches	PKR 71,220	INR 25,999	CNY 2,299	USD 300	AED 1,099	2024
927	Samsung	Galaxy Z Fold6 256GB	7.6 inches	PKR 604,999	INR 164,999	₹13,999	USD 1,899	AED 7,199	2024
928	Samsung	Galaxy Z Fold6 512GB	7.6 inches	PKR 544,999	INR 176,999	CNY 15,999	USD 1719	AED 7,699	2024
929	Samsung	Galaxy Z Fold6 1TB	7.6 inches	Not available	INR 200,999	CNY 17,999	USD 2,259	AED 8,699	2024

930 rows × 9 columns

Примеры проблем:

- Mobile Weight: "174g" (с символом 'g')
- Battery Capacity: "3.600mAh" (с символом 'mAh')
- RAM: "6GB" (с символом 'GB')
- Цены: "PKR 224,999" (с валютой и разделителями)

Выводы по шагу 1

- Файл успешно загружен
 - Размер датасета: 930 строк × 15 столбцов
 - Присутствуют как старые, так и новые модели телефонов
- Географический охват:
 - 5 стран: Пакистан, Индия, Китай, США, Дубай (ОАЭ)
- Цены представлены в национальных валютах: PKR, INR, CNY, USD, AED
- Все столбцы читаются как object (строки) за исключением Launched Year (int64)
 - Необходимо большую часть данных привести к числовым типам
- Примеры строк показывают очевидную неоднородность формата: в ячейках встречаются переносы строк, обозначения единиц (g, GB, mAh, inches), текстовые префиксы валют (PKR, INR, CNY, USD, AED) и разделители тысяч (запятые).
 - Все технические характеристики (Mobile Weight, RAM, Battery Capacity, Screen Size) имеют тип object вместо числовых
 - Неоднородные форматы значений
 - Ценовые столбцы содержат код валюты, и число в одном поле
 - Множественные значения в полях камер (например 50MP + 8MP + 2MP)
 - Возможные артефакты кодировки - строка 927, столбец Launched Price (China): символы вроде ₹ в выборке

- Видны различные примеры производителей: Apple, Samsung, Poco, Realme и т.д.
 - в выборке есть премиум-модели и бюджетные
- Пропусков нет - все колонки показывают 930 non-null). Однако есть скрытые «пропуски», которые могут быть представлены текстом вроде Not available или нестандартными значениями — нужно отдельно обработать.
- Необходима проверка неявных дубликатов, например по сочетанию Company Name + Model Name + Launched Year).
- Для корректного сравнения цен по странам требуется конвертация валют (в USD) и корректировка на инфляцию/реальную стоимость — нужен источник курсов валют

Стоит обратить внимание что помимо смартфонов в датасете присутствуют и планшеты

[* к содержанию](#)

Шаг 2: Предобработка данных

Переименование столбцов

- переименуем столбцы согласно стилистики, а также добавим в названия характеристики из столбцов

```
In [6]: def extract_non_digit_groups(df):
    result = {}

    for col in df.columns:
        # Объединяем все значения столбца в одну строку
        text = " ".join(df[col].astype(str))
        # Ищем группы подряд идущих нецифровых символов (минимум 2 символа)
        groups = re.findall(r"[^\d\s]{2,}", text)
        result[col] = sorted(set(groups))

    return result
```

```
In [7]: groups_by_column = extract_non_digit_groups(df)

for col, groups in groups_by_column.items():
    print(f"🔍 Столбец: {col}")
    print(f"    Склейенные нецифровые группы: {groups}\n")
```

- Столбец: Company Name
Склейные нецифровые группы: ['Apple', 'Google', 'Honor', 'Huawei', 'Infinix', 'Lenovo', 'Motorola', 'Nokia', 'OnePlus', 'Oppo', 'POCO', 'Poco', 'Realme', 'Samsung', 'Sony', 'Tecno', 'Vivo', 'Xiaomi', 'iQOO']
- Столбец: Model Name
Склейные нецифровые группы: [-inch, Active, Air, Art, CE, Camon, Cyberpunk, Edge, Edition, FE, Find, Flip, Fol d, Fusion, GB, GT, Galaxy, Go, HD, Hot, IV, Kids, Legion, Lite, Magic, MagicPad, Mate, MatePad, Max, McLaren, Megapad, Mini, Moto, NFC, Narzo, Neo, Nord, Note, Nova, One, OnePlus, Open, Pad, Phantom, Pixel, Play, Plus, Pocket, Pop, Pova, Power, Premier, Pro, Pro+, Pura, Racing, Razr, Redmi, Reno, SE, Series, Slim, Smart, Spark, Special, Speed, Star, Stylus, TB, Tab, Tablet, TechLife, Turbo, Ultimate, Ultra, VI, Vision, Vs, Wars, XL, XR, XS, XT, Xcover, Xiaomi, Xpad, Xperia, Xs, Zero, iPad, iPhone, iQOO, mini]
- Столбец: Mobile Weight
Склейные нецифровые группы: []
- Столбец: RAM
Склейные нецифровые группы: [GB]
- Столбец: Front Camera
Склейные нецифровые группы: [(UDC), (telephoto), (ultrawide), Dual, MP, MP+, MP,]
- Столбец: Back Camera
Склейные нецифровые группы: [(Depth), (Macro), (Main), (Telephoto), (Ultra-wide), (f/), (periscope), (telephoto), (ultraw ide), (wide), AF, MP, MP+, telephoto]
- Столбец: Processor
Склейные нецифровые группы: [-AI, -Max, -Ultra, Bionic, Dimensity, Elite, Energy, Exynos, G+, Gen, Google, Helio, Kirin, MSM, MT, MediaTek, Plus, Pro, Qualcomm, SC, Snapdragon, Spreadtrum, Tensor, Unisoc]
- Столбец: Battery Capacity
Склейные нецифровые группы: [mAh]
- Столбец: Screen Size
Склейные нецифровые группы: [(external), (internal), (main), (unfolded), inches]
- Столбец: Launched Price (Pakistan)
Склейные нецифровые группы: [Not, PKR, available]
- Столбец: Launched Price (India)
Склейные нецифровые группы: [INR]
- Столбец: Launched Price (China)
Склейные нецифровые группы: [CNY]
- Столбец: Launched Price (USA)
Склейные нецифровые группы: [USD]
- Столбец: Launched Price (Dubai)
Склейные нецифровые группы: [AED]
- Столбец: Launched Year
Склейные нецифровые группы: []

```
In [8]: columns_names = {
    'Company Name': 'company_name',
    'Model Name': 'model_name',
    'Mobile Weight': 'mobile_weight_g',
    'RAM': 'ram_gb',
    'Front Camera': 'front_camera_mp',
    'Back Camera': 'back_camera_mp',
    'Processor': 'processor',
    'Battery Capacity': 'battery_capacity_mah',
    'Screen Size': 'screen_size_inches',
    'Launched Price (Pakistan)': 'price_pakistan_pk',
    'Launched Price (India)': 'price_india_inr',
    'Launched Price (China)': 'price_china_cny',
    'Launched Price (USA)': 'price_usa_usd',
    'Launched Price (Dubai)': 'price_dubai_aed',
    'Launched Year': 'launched_year'
}
df = df.rename(columns = columns_names)
print("\n👉 Названия столбцов:")
print(df.columns)
```

👉 Названия столбцов:
Index(['company_name', 'model_name', 'mobile_weight_g', 'ram_gb',
 'front_camera_mp', 'back_camera_mp', 'processor',
 'battery_capacity_mah', 'screen_size_inches', 'price_pakistan_pk',
 'price_india_inr', 'price_china_cny', 'price_usa_usd',
 'price_dubai_aed', 'launched_year'],
 dtype='object')

Поиск артефактов

- найдем возможные артефакты в датасете

```
In [9]: def find_non_alphanum_rows_verbose(df):
    allowed = set("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/. ")
    error_info = []
```

```
for idx, row in df.iterrows():
    symbol_columns = {}

    for col, val in row.items():
        if isinstance(val, str):
            for char in val:
                if char not in allowed:
                    symbol_columns.setdefault(char, []).append(col)

    if symbol_columns:
        error_info.append((idx, symbol_columns))

return error_info
```

```
In [10]: errors = find_non_alphanum_rows_verbose(df)
```

```
for idx, symbol_map in errors:
    print(f"⚠ Стока {idx}:")
    for symbol, columns in symbol_map.items():
        print(f"  Символ '{symbol}' встречается в колонках: {columns}")
    print()
```


⚠ Страна 928:
Символ '(' встречается в колонках: ['front_camera_mp']
Символ ')' встречается в колонках: ['front_camera_mp']

⚠ Страна 929:
Символ '(' встречается в колонках: ['front_camera_mp']
Символ ')' встречается в колонках: ['front_camera_mp']
Символ ' ' встречается в колонках: ['price_china_cny']

'_'

```
In [11]: df[df.apply(  
    lambda row: any('-' in str(val) for val in [row['model_name']])),  
    axis=1  
].iloc[:, :8]
```

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
80	Apple	iPad Air 10.9-inch 64GB	458g	4GB	12MP	12MP	A14 Bionic	7,608mAh
81	Apple	iPad Air 10.9-inch 256GB	458g	4GB	12MP	12MP	A14 Bionic	7,608mAh
82	Apple	iPad 10.2-inch 32GB	490g	3GB	12MP	8MP	A13 Bionic	8,612mAh
83	Apple	iPad 10.2-inch 128GB	490g	3GB	12MP	8MP	A13 Bionic	8,612mAh
84	Apple	iPad Mini 7.9-inch 64GB	300.5g	3GB	7MP	8MP	A12 Bionic	5,124mAh
85	Apple	iPad Mini 7.9-inch 256GB	300.5g	3GB	7MP	8MP	A12 Bionic	5,124mAh
86	Apple	iPad Pro 11-inch 128GB	468g	4GB	7MP	12MP + 10MP	A12Z Bionic	7,812mAh
87	Apple	iPad Pro 11-inch 256GB	468g	4GB	7MP	12MP + 10MP	A12Z Bionic	7,812mAh
88	Apple	iPad Pro 11-inch 512GB	468g	6GB	7MP	12MP + 10MP	A12Z Bionic	7,812mAh
89	Apple	iPad Pro 12.9-inch 128GB	682g	6GB	7MP	12MP + 10MP	A12Z Bionic	9,720mAh
90	Apple	iPad Pro 12.9-inch 256GB	682g	6GB	7MP	12MP + 10MP	A12Z Bionic	9,720mAh
91	Apple	iPad Pro 12.9-inch 512GB	682g	6GB	7MP	12MP + 10MP	A12Z Bionic	9,720mAh
92	Apple	iPad Pro 13-inch 128GB	708g	6GB	7MP	12MP + 10MP	A12Z Bionic	10,307mAh
93	Apple	iPad Pro 13-inch 256GB	708g	6GB	7MP	12MP + 10MP	A12Z Bionic	10,307mAh
94	Apple	iPad Pro 13-inch 512GB	708g	6GB	7MP	12MP + 10MP	A12Z Bionic	10,307mAh
95	Apple	iPad Pro 13-inch 1TB	708g	6GB	7MP	12MP + 10MP	A12Z Bionic	10,307mAh
96	Apple	iPad Pro 13-inch 2TB	708g	6GB	7MP	12MP + 10MP	A12Z Bionic	10,307mAh

```
In [12]: df.loc[80:96, 'model_name'] = df.loc[80:96, 'model_name'].str.replace(r'\d+(\.\d+)?-inch', '', regex=True).str.strip()  
df[df.apply(  
    lambda row: any('-' in str(val) for val in [row['model_name']])),  
    axis=1  
].iloc[:, :8]
```

```
Out[12]: company_name model_name mobile_weight_g ram_gb front_camera_mp back_camera_mp processor battery_capacity_mah
```

```
In [13]: df[df.apply(  
    lambda row: any('-' in str(val) for val in [row['processor']])),  
    axis=1  
].iloc[:, :8]
```

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
197	OnePlus	OnePlus Nord 2 128GB	190g	8GB	32MP	50MP + 8MP	MediaTek Dimensity 1200-AI	4500mAh
211	OnePlus	OnePlus Nord 2 5G 128GB	189g	8GB	32MP	50MP + 8MP	MediaTek Dimensity 1200-AI	4500mAh
427	Oppo	K10 5G 128GB	205g	8GB	16MP	64MP + 8MP + 2MP	MediaTek Dimensity 8000-Max	5,000mAh
443	Oppo	K10 5G 128GB	205g	8GB	16MP	64MP + 8MP + 2MP	MediaTek Dimensity 8000-Max	5,000mAh
444	Oppo	K10 5G 256GB	205g	12GB	16MP	64MP + 8MP + 2MP	MediaTek Dimensity 8000-Max	5,000mAh
530	Xiaomi	Xiaomi 14T 256GB	195g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Dimensity 8300-Ultra	5,000mAh
531	Xiaomi	Xiaomi 14T 512GB	195g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Dimensity 8300-Ultra	5,000mAh
539	Xiaomi	Redmi Note 14 Pro 5G 128GB	195g	8GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7300-Ultra	5,500mAh
540	Xiaomi	Redmi Note 14 Pro 5G 256GB	195g	12GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7300-Ultra	5,500mAh
543	Xiaomi	Redmi Note 14 5G 128GB	185g	6GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7025-Ultra	5,000mAh
544	Xiaomi	Redmi Note 14 5G 256GB	185g	8GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7025-Ultra	5,000mAh

данный символ нужно не удалять, это характеристика модели

'(,)'

```
In [14]: df[df.apply(
    lambda row: any(sym in str(row[col]) for col in ['front_camera_mp', 'back_camera_mp']) for sym in ['(', ')']),
    axis=1
)].iloc[:, :8]
```

Out[14]:	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
522	Xiaomi	Xiaomi 15 Pro 256GB	210g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Snapdragon 8 Elite	6,100mAh
523	Xiaomi	Xiaomi 15 Pro 512GB	210g	16GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Snapdragon 8 Elite	6,100mAh
524	Xiaomi	Xiaomi 15 Pro 1TB	210g	16GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Snapdragon 8 Elite	6,100mAh
525	Xiaomi	Xiaomi 15 256GB	190g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Snapdragon 8 Elite	5,400mAh
526	Xiaomi	Xiaomi 15 512GB	190g	16GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Snapdragon 8 Elite	5,400mAh
527	Xiaomi	Xiaomi 14T Pro 256GB	209g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Dimensity 9300+	5,000mAh
528	Xiaomi	Xiaomi 14T Pro 512GB	209g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Dimensity 9300+	5,000mAh
529	Xiaomi	Xiaomi 14T Pro 1TB	209g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Dimensity 9300+	5,000mAh
530	Xiaomi	Xiaomi 14T 256GB	195g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Dimensity 8300-Ultra	5,000mAh
531	Xiaomi	Xiaomi 14T 512GB	195g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Dimensity 8300-Ultra	5,000mAh
532	Xiaomi	Xiaomi 14 Pro 256GB	206g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Snapdragon 8 Gen 3	5,000mAh
533	Xiaomi	Xiaomi 14 Pro 512GB	206g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide) + 50MP (Teleph...	Snapdragon 8 Gen 3	5,000mAh
534	Xiaomi	Xiaomi 14 256GB	189g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Snapdragon 8 Gen 3	4,610mAh
535	Xiaomi	Xiaomi 14 512GB	189g	12GB	32MP	50MP (Main) + 50MP (Ultra-wide)	Snapdragon 8 Gen 3	4,610mAh
536	Xiaomi	Redmi Note 14 Pro+ 5G 128GB	205g	8GB	16MP	200MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Snapdragon 7s Gen 3	5,110mAh
537	Xiaomi	Redmi Note 14 Pro+ 5G 256GB	205g	12GB	16MP	200MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Snapdragon 7s Gen 3	5,110mAh
538	Xiaomi	Redmi Note 14 Pro+ 5G 512GB	205g	16GB	16MP	200MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Snapdragon 7s Gen 3	5,110mAh
539	Xiaomi	Redmi Note 14 Pro 5G 128GB	195g	8GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7300-Ultra	5,500mAh
540	Xiaomi	Redmi Note 14 Pro 5G 256GB	195g	12GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7300-Ultra	5,500mAh
541	Xiaomi	Redmi Note 14 Pro 4G 128GB	190g	6GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Qualcomm Snapdragon 732G	5,000mAh
542	Xiaomi	Redmi Note 14 Pro 4G 256GB	190g	8GB	16MP	108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Qualcomm Snapdragon 732G	5,000mAh
543	Xiaomi	Redmi Note 14 5G 128GB	185g	6GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7025-Ultra	5,000mAh
544	Xiaomi	Redmi Note 14 5G 256GB	185g	8GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	MediaTek Dimensity 7025-Ultra	5,000mAh
545	Xiaomi	Redmi Note 14 4G 128GB	180g	4GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Qualcomm Snapdragon 680	5,000mAh
546	Xiaomi	Redmi Note 14 4G 256GB	180g	6GB	16MP	50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)	Qualcomm Snapdragon 680	5,000mAh
547	Xiaomi	Redmi 14C 5G 64GB	195g	4GB	8MP	50MP (Main) + 2MP (Depth)	MediaTek Dimensity 700	5,000mAh

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
548	Xiaomi	Redmi 14C 5G 128GB	195g	6GB	8MP	50MP (Main) + 2MP (Depth)	MediaTek Dimensity 700	5,000mAh
640	Huawei	P60	197g	8GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
641	Huawei	P60 Pro	200g	8GB / 12GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
642	Huawei	P60 Art	206g	8GB / 12GB	13MP	48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	5,100mAh
643	Huawei	Mate X3	239g	12GB	8MP	50MP (wide) + 13MP (ultrawide) + 12MP (perisco...)	Snapdragon 8+ Gen 1 4G	4,800mAh
644	Huawei	Mate 60	209g	12GB	13MP	50MP (wide) + 12MP (ultrawide) + 12MP (telephoto)	Kirin 9000S	4,750mAh
645	Huawei	Mate 60 Pro	225g	12GB	13MP	50MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Kirin 9000S	5,000mAh
646	Huawei	Mate 60 Pro+	235g	12GB	13MP	50MP (wide) + 48MP (ultrawide) + 48MP (telephoto)	Kirin 9000S	5,200mAh
647	Huawei	Nova 11	168g	8GB	60MP	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
648	Huawei	Nova 11 Pro	188g	8GB	60MP (ultrawide) + 8MP (telephoto)	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
649	Huawei	Nova 11 Ultra	188g	12GB	60MP (ultrawide) + 8MP (telephoto)	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
686	Huawei	MatePad Pro 12.2 512GB	508g	12GB	16MP	13MP (f/1.8, AF)	Kirin 9000S	10,100mAh
687	Huawei	MatePad Pro 13.2 512GB	580g	12GB	16MP	13MP (f/1.8, AF)	Kirin 9000S	10,100mAh
927	Samsung	Galaxy Z Fold6 256GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
928	Samsung	Galaxy Z Fold6 512GB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
929	Samsung	Galaxy Z Fold6 1TB	239g	12GB	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh

- это характеристики камер. что с этим делать мы решим когда будем анализировать каждый столбец отдельно

```
In [15]: df[df.apply(
    lambda row: any(sym in str(row[col]) for col in ['screen_size_inches']) for sym in ['(', ')']),
    axis=1
)].iloc[:, :8]
```

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
568	Motorola	Razr 128GB	192g	8GB	32MP	50MP	Snapdragon 8+ Gen 1	3,500mAh
569	Motorola	Razr 256GB	192g	12GB	32MP	50MP	Snapdragon 8+ Gen 1	3,500mAh
598	Motorola	Razr 50 Ultra 512GB	189g	12GB	32MP	50MP	Snapdragon 8+ Gen 1	4,000mAh
599	Motorola	Razr 50 256GB	188g	8GB	32MP	50MP	Snapdragon 7 Gen 1	3,800mAh
628	Huawei	P50 Pocket	190g	8GB	10.7MP	40MP	Snapdragon 888 4G	4,000mAh
630	Huawei	Mate X2	295g	8GB	16MP	50MP	Kirin 9000 5G	4,500mAh
634	Huawei	Mate Xs 2	255g	8GB	10.7MP	50MP	Snapdragon 888 4G	4,600mAh
643	Huawei	Mate X3	239g	12GB	8MP	50MP (wide) + 13MP (ultrawide) + 12MP (perisco...)	Snapdragon 8+ Gen 1 4G	4,800mAh

- это характеристики экрана. что с этим делать мы решим когда будем анализировать каждый столбец отдельно

```
In [16]: def extract_non_alphanum_symbols(df):
    result = {}
```

```

for col in df.columns:
    # Объединяем все значения столбца в одну строку
    text = " ".join(df[col].astype(str))
    # Извлекаем все символы, которые НЕ являются буквами или цифрами
    symbols = re.findall(r"[^a-zA-Z0-9]", text)
    # Убираем пробелы и дубликаты
    cleaned = sorted(set(sym for sym in symbols if sym.strip()))
    result[col] = cleaned

return result

```

```
In [17]: symbols_by_column = extract_non_alphanum_symbols(df)

for col, symbols in symbols_by_column.items():
    print(f"Столбец: {col}")
    print(f"Нестандартные символы: {symbols}\n")
```

- Столбец: company_name
Нестандартные символы: []
- Столбец: model_name
Нестандартные символы: ['+', '.']
- Столбец: mobile_weight_g
Нестандартные символы: ['.]
- Столбец: ram_gb
Нестандартные символы: ['.', '/']
- Столбец: front_camera_mp
Нестандартные символы: ['(', ')', '+', ',', '.', '/']
- Столбец: back_camera_mp
Нестандартные символы: ['(', ')', '+', ',', '-', '.', '/']
- Столбец: processor
Нестандартные символы: ['+', '-']
- Столбец: battery_capacity_mah
Нестандартные символы: [,]
- Столбец: screen_size_inches
Нестандартные символы: ['(', ')', ',', '.']
- Столбец: price_pakistan_pk
Нестандартные символы: [,]
- Столбец: price_india_inr
Нестандартные символы: [,]
- Столбец: price_china_cny
Нестандартные символы: [, 'À']
- Столбец: price_usa_usd
Нестандартные символы: [, , '.']
- Столбец: price_dubai_aed
Нестандартные символы: [,]
- Столбец: launched_year
Нестандартные символы: []

'À'

```
In [18]: df[df['price_china_cny'].str.contains(r'\À', na=False)].iloc[:, :8]
```

```
Out[18]:   company_name      model_name  mobile_weight_g  ram_gb  front_camera_mp  back_camera_mp  processor  battery_capacity_mah
927        Samsung  Galaxy Z Fold6 256GB           239g     12GB  10MP, 4MP (UDC)       50MP  Snapdragon 8 Gen 3      4400mAh
```

найден артефакт À, удалим

```
In [19]: df['price_china_cny'] = df['price_china_cny'].str.replace('À', '')
df[df['price_china_cny'].str.contains(r'\À', na=False)].iloc[:, :8]
```

```
Out[19]:   company_name      model_name  mobile_weight_g  ram_gb  front_camera_mp  back_camera_mp  processor  battery_capacity_mah
```

Приведение к нужному типу

```
In [20]: df.columns
```

```
Out[20]: Index(['company_name', 'model_name', 'mobile_weight_g', 'ram_gb',
               'front_camera_mp', 'back_camera_mp', 'processor',
               'battery_capacity_mah', 'screen_size_inches', 'price_pakistan_pk',
               'price_india_inr', 'price_china_cny', 'price_usa_usd',
               'price_dubai_aed', 'launched_year'],
              dtype='object')
```

Теперь проанализируем каждый столбец и приведем значения в нём к нужному типу

company_name

```
In [21]: df['company_name'].unique()
```

```
Out[21]: array(['Apple', 'Samsung', 'OnePlus', 'Vivo', 'iQOO', 'Oppo', 'Realme',
   'Xiaomi', 'Lenovo', 'Motorola', 'Huawei', 'Nokia', 'Sony',
   'Google', 'Tecno', 'Infinix', 'Honor', 'POCO', 'Poco'],
  dtype=object)
```

Виден неявный дубликат в названиях 'POCO', 'Poco'. Исправим

```
In [22]: df['company_name'] = df['company_name'].replace('POCO', 'Poco')
sorted(df['company_name'].unique())
```

```
Out[22]: ['Apple',
 'Google',
 'Honor',
 'Huawei',
 'Infinix',
 'Lenovo',
 'Motorola',
 'Nokia',
 'OnePlus',
 'Oppo',
 'Poco',
 'Realme',
 'Samsung',
 'Sony',
 'Tecno',
 'Vivo',
 'Xiaomi',
 'iQOO']
```

Бренд	Страна происхождения	Основная продукция	Материнская компания
Apple	США	iPhone, iPad, Mac	Apple Inc.
Google	США	Pixel	Alphabet Inc.
Honor	Китай	Magic, X-серия	Honor Device Co., Ltd.
Huawei	Китай	P-серия, Mate	Huawei Technologies
Infinix	Китай / Гонконг	Zero, Note	Transsion Holdings
iQOO	Китай (бренд Vivo)	iQOO Neo, Z	BBK Electronics
Lenovo	Китай	Планшеты, ноутбуки, Legion	Lenovo Group Ltd.
Motorola	США / Китай (Lenovo)	Moto G, Edge	Lenovo Group Ltd.
Nokia	Финляндия / Китай	Android-смартфоны (HMD)	HMD Global
OnePlus	Китай (BBK Group)	Nord, Ace	BBK Electronics
Oppo	Китай (BBK Group)	Reno, Find	BBK Electronics
Poco	Китай (бренд Xiaomi)	Poco X, F, Pad	Xiaomi Corporation
Realme	Китай (BBK Group)	Narzo, GT	BBK Electronics
Samsung	Южная Корея	Galaxy, Fold, Tab	Samsung Electronics
Sony	Япония	Xperia	Sony Corporation
Tecno	Китай / Гонконг	Camon, Spark	Transsion Holdings
Vivo	Китай (BBK Group)	V-серия, X-серия	BBK Electronics
Xiaomi	Китай	Redmi, Mi, Pad	Xiaomi Corporation

- BBK Group — материнская компания для OnePlus, Oppo, Vivo, Realme, iQOO
- Poco — начался как суббренд Xiaomi, теперь позиционируется отдельно
- Honor — отделился от Huawei после санкций
- Motorola — принадлежит Lenovo с 2014 года
- Nokia — бренд используется HMD Global для Android-устройств

```
In [23]: # Объединяем все значения в одну строку
all_text = ' '.join(df['company_name'].dropna().astype(str))

# Находим все символы, которые НЕ являются буквами или цифрами
non_alnum_chars = re.findall(r'[^a-zA-Z0-9]', all_text)

# Получаем уникальные символы
unique_non_alnum = sorted(set(non_alnum_chars))

print("🚫 Неалфавитные и нецифровые символы:", unique_non_alnum)
```

🚫 Неалфавитные и нецифровые символы: []

model_name

```
In [24]: sorted(df['model_name'].unique())[:10] # первые 10
```

```
Out[24]: ['10X Lite',
 '13 5G 128GB',
 '13 5G 256GB',
 '13 Pro 5G 128GB',
 '13 Pro 5G 256GB',
 '13 Pro+ 5G 256GB',
 '13 Pro+ 5G 512GB',
 '13+ 5G 128GB',
 '13+ 5G 256GB',
 '14 Pro 5G 128GB']
```

```
In [25]: sorted(df['model_name'].unique())[-10:] # первые 10
```

```
Out[25]: ['iPhone XR 128GB',
 'iPhone XR 256GB',
 'iPhone XR 64GB',
 'iPhone XS 256GB',
 'iPhone XS 512GB',
 'iPhone XS 64GB',
 'iPhone XS Max 256GB',
 'iPhone XS Max 512GB',
 'iPhone XS Max 64GB',
 'iQOO 12 256GB']
```

```
In [26]: # Объединяем все значения в одну строку
all_text = ' '.join(df['model_name'].dropna().astype(str))

# Находим все символы, которые НЕ являются буквами или цифрами
non_alnum_chars = re.findall(r'[^a-zA-Z0-9]', all_text)

# Получаем уникальные символы
unique_non_alnum = sorted(set(non_alnum_chars))

print("🚫 Неалфавитные и нецифровые символы:", unique_non_alnum)
```

🚫 Неалфавитные и нецифровые символы: [' ', '+', '.']

mobile_weight_g

```
In [27]: df['mobile_weight_g'].unique()
```

```
Out[27]: array(['174g', '203g', '206g', '221g', '171g', '172g', '140g', '204g',
 '238g', '135g', '164g', '189g', '228g', '194g', '188g', '226g',
 '177g', '208g', '458g', '490g', '300.5g', '468g', '682g', '708g',
 '234g', '196g', '168g', '195g', '167g', '254g', '187g', '263g',
 '199g', '190g', '202g', '198g', '192g', '235g', '191g', '181g',
 '178g', '175g', '165g', '143g', '229g', '732g', '586g', '498g',
 '523g', '726g', '567g', '503g', '480g', '366g', '508g', '433g',
 '674g', '426g', '360g', '205g', '179g', '200g', '183g', '185g',
 '180g', '173g', '184g', '170g', '162g', '210g', '155g', '215g',
 '550g', '610g', '223g', '150g', '146g', '158g', '163g', '153g',
 '176g', '201g', '193g', '159g', '156g', '182g', '535g', '510g',
 '500g', '560g', '520g', '540g', '555g', '239g', '186g', '533g',
 '218g', '207g', '233g', '222.8g', '440g', '482g', '466g', '499g',
 '372g', '465g', '209g', '166g', '169g', '295g', '255g', '197g',
 '225g', '220g', '241g', '245g', '161g', '580g', '147g', '151g',
 '212g', '213g', '216g', '222g', '250g', '450g', '420g', '280g',
 '230g', '240g', '470g', '211g', '227g', '236g', '242g', '288g',
 '219g', '267g', '231g', '460g', '485g', '530g', '495g', '590g',
 '505g', '475g', '178.8g', '571g'], dtype=object)
```

уберем g и преобразуем в едини формат

```
In [28]: def clean_column_units(df, column_name, unit_to_remove):
    df[column_name] = (
        df[column_name]
        .astype(str) # на случай, если есть числа
        .str.replace(unit_to_remove, '', regex=False)
        .str.strip()
        .astype(float)
    )

    # Выводим отсортированные уникальные значения
    unique_sorted = sorted(df[column_name].dropna().unique())
    print(f"📦 Уникальные значения в {column_name} (отсортированные): \n", unique_sorted)
    print(f"🔍 Тип данных {column_name}: ", df[column_name].dtype)

    return df
```

```
In [29]: df = clean_column_units(df, 'mobile_weight_g', 'g')
```

💡 Уникальные значения в `mobile_weight_g` (отсортированные):
[135.0, 140.0, 143.0, 146.0, 147.0, 150.0, 151.0, 153.0, 155.0, 156.0, 158.0, 159.0, 161.0, 162.0, 163.0, 164.0, 165.0, 166.0, 167.0, 168.0, 169.0, 170.0, 171.0, 172.0, 173.0, 174.0, 175.0, 176.0, 177.0, 178.0, 178.8, 179.0, 180.0, 181.0, 182.0, 183.0, 184.0, 185.0, 186.0, 187.0, 188.0, 189.0, 190.0, 191.0, 192.0, 193.0, 194.0, 195.0, 196.0, 197.0, 198.0, 199.0, 200.0, 201.0, 202.0, 203.0, 204.0, 205.0, 206.0, 207.0, 208.0, 209.0, 210.0, 211.0, 212.0, 213.0, 215.0, 216.0, 218.0, 219.0, 220.0, 221.0, 222.0, 222.8, 223.0, 225.0, 226.0, 227.0, 228.0, 229.0, 230.0, 231.0, 233.0, 234.0, 235.0, 236.0, 238.0, 239.0, 240.0, 241.0, 242.0, 245.0, 250.0, 254.0, 255.0, 263.0, 267.0, 280.0, 288.0, 295.0, 300.5, 360.0, 366.0, 372.0, 420.0, 426.0, 433.0, 440.0, 450.0, 458.0, 460.0, 465.0, 466.0, 468.0, 470.0, 475.0, 480.0, 482.0, 485.0, 490.0, 495.0, 498.0, 499.0, 500.0, 503.0, 505.0, 508.0, 510.0, 520.0, 523.0, 530.0, 533.0, 535.0, 540.0, 550.0, 555.0, 560.0, 567.0, 571.0, 580.0, 586.0, 590.0, 610.0, 674.0, 682.0, 708.0, 726.0, 732.0]

🔍 Тип данных `mobile_weight_g`: `float64`

ram_gb

In [30]: `df['ram_gb'].unique()`

Out[30]: `array(['6GB', '8GB', '4GB', '3GB', '12GB', '2GB', '1.5GB', '16GB', '10GB', '1GB', '8GB / 12GB'], dtype=object)`

- видим что есть интересное значение '8GB / 12GB', посмотрим что это

In [31]: `df[df['ram_gb'] == '8GB / 12GB'].iloc[:, :8]`

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
641	Huawei	P60 Pro	200.0	8GB / 12GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
642	Huawei	P60 Art	206.0	8GB / 12GB	13MP	48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	5,100mAh

- получается продается телефон за одну и ту же цену но с разной ram. Посмотрим сколько вообще есть телефонов таких моделей

In [32]: `df[(df['model_name'] == 'P60 Art') | (df['model_name'] == 'P60 Pro')].iloc[:, :8]`

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
641	Huawei	P60 Pro	200.0	8GB / 12GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
642	Huawei	P60 Art	206.0	8GB / 12GB	13MP	48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	5,100mAh

как и предполагалось их всего два. Предлагаю просто породублировать эти две строчки с разделением на 8 и 12 gb в колонке ram_gb

```
# Создаём копию строк с составной RAM
copied_rows = df[df['model_name'].isin(['P60 Art', 'P60 Pro'])].copy()

# В копии заменяем на максимальную конфигурацию
copied_rows['ram_gb'] = copied_rows['ram_gb'].replace('8GB / 12GB', '12GB')

# В оригинале оставляем минимальную конфигурацию
df['ram_gb'] = df['ram_gb'].replace('8GB / 12GB', '8GB')

# Объединяем обе версии
df = pd.concat([df, copied_rows], ignore_index=True)

# Проверяем результат
df[df['model_name'].isin(['P60 Art', 'P60 Pro'])].iloc[:, :8]
```

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
641	Huawei	P60 Pro	200.0	8GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
642	Huawei	P60 Art	206.0	8GB	13MP	48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	5,100mAh
930	Huawei	P60 Pro	200.0	12GB	13MP	48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	4,815mAh
931	Huawei	P60 Art	206.0	12GB	13MP	48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)	Snapdragon 8+ Gen 1 4G	5,100mAh

In [34]: `df['ram_gb'].unique()`

Out[34]: `array(['6GB', '8GB', '4GB', '3GB', '12GB', '2GB', '1.5GB', '16GB', '10GB', '1GB'], dtype=object)`

In [35]: `df = clean_column_units(df, 'ram_gb', 'GB')`

💡 Уникальные значения в `ram_gb` (отсортированные):

[1.0, 1.5, 2.0, 3.0, 4.0, 6.0, 8.0, 10.0, 12.0, 16.0]

🔍 Тип данных `ram_gb`: `float64`

front_camera_mp

```
In [36]: df['front_camera_mp'].unique()
```

```
Out[36]: array(['12MP', '12MP / 4K', '7MP', '10MP', '32MP', '13MP', '5MP', '16MP',
   '8MP', '12MP + 12MP', '2MP', '44MP', '24MP', '20MP+8MP', '20MP',
   '50MP', '25MP', '60MP', '10.7MP', 'Dual 32MP', 'Dual 60MP',
   '60MP (ultrawide) + 8MP (telephoto)', '60MP + 8MP', '11.1MP',
   '10.8MP', '10.5MP', '48MP', '42MP', '10MP, 4MP (UDC)'],
  dtype=object)
```

- в некоторых моделях встречается 2 камеры. Причем типа камер: ultrawide - широкоугольная камера, telephoto - объектив, обеспечивающий оптическое увеличение зума (телеобъектив), UDC - камера, расположенная под дисплеем.

Проанализируем, какие телефоны применяют подобные камеры и не является ли это рекламным ходом.

- для этого выведем телефоны с фронтальными камерами 60MP (ultrawide) + 8MP (telephoto) и 60MP.

```
In [37]: df[(df['front_camera_mp'] == '60MP (ultrawide) + 8MP (telephoto') | (df['front_camera_mp'] == '60MP')).iloc[:, :8]
```

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
608	Motorola	Moto X50 Ultra 512GB	198.0	12.0	60MP	200MP	Snapdragon 8 Gen 3	5,000mAh
611	Motorola	Edge 50 Ultra 512GB	198.0	12.0	60MP	200MP	Snapdragon 8 Gen 3	5,000mAh
637	Huawei	Nova 10	168.0	8.0	60MP	50MP	Snapdragon 778G 4G	4,000mAh
647	Huawei	Nova 11	168.0	8.0	60MP	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
648	Huawei	Nova 11 Pro	188.0	8.0	60MP (ultrawide) + 8MP (telephoto)	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
649	Huawei	Nova 11 Ultra	188.0	12.0	60MP (ultrawide) + 8MP (telephoto)	50MP (wide) + 8MP (ultrawide)	Snapdragon 778G 4G	4,500mAh
658	Huawei	Nova 12	168.0	8.0	60MP	50MP + 8MP	Kirin 9000S	4,500mAh

камеры есть только телефоны марки Huawei модели Nova 11 Pro и Nova 11 Ultra. Судя по всему это маркетинговый ход, поэтому можно удалить (ultrawide) и (telephoto).

```
In [38]: df['front_camera_mp'] = df['front_camera_mp'].replace('60MP (ultrawide) + 8MP (telephoto)', '60MP + 8MP')
df[df['front_camera_mp'] == '60MP + 8MP']['front_camera_mp']
```

```
Out[38]: 648    60MP + 8MP
649    60MP + 8MP
659    60MP + 8MP
664    60MP + 8MP
665    60MP + 8MP
Name: front_camera_mp, dtype: object
```

```
In [39]: df['front_camera_mp'].unique()
```

```
Out[39]: array(['12MP', '12MP / 4K', '7MP', '10MP', '32MP', '13MP', '5MP', '16MP',
   '8MP', '12MP + 12MP', '2MP', '44MP', '24MP', '20MP+8MP', '20MP',
   '50MP', '25MP', '60MP', '10.7MP', 'Dual 32MP', 'Dual 60MP',
   '60MP + 8MP', '11.1MP', '10.8MP', '10.5MP', '48MP', '42MP',
   '10MP, 4MP (UDC)'],
  dtype=object)
```

- выведем все строки где встречается 12MP / 4K, Dual 32MP, Dual 60MP, 4MP (UDC), 10MP, 4MP (UDC)

```
In [40]: df[df['front_camera_mp'].isin(['12MP / 4K'])].iloc[:, :8]
```

Out[40]:	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
6	Apple	iPhone 16 Pro 128GB	206.0	6.0	12MP / 4K	50MP + 12MP	A17 Pro	4,400mAh
7	Apple	iPhone 16 Pro 256GB	206.0	8.0	12MP / 4K	50MP + 12MP	A17 Pro	4,400mAh
8	Apple	iPhone 16 Pro 512GB	206.0	8.0	12MP / 4K	50MP + 12MP	A17 Pro	4,400mAh
9	Apple	iPhone 16 Pro Max 128GB	221.0	6.0	12MP / 4K	48MP + 12MP	A17 Pro	4,500mAh
10	Apple	iPhone 16 Pro Max 256GB	221.0	8.0	12MP / 4K	48MP + 12MP	A17 Pro	4,500mAh
11	Apple	iPhone 16 Pro Max 512GB	221.0	8.0	12MP / 4K	48MP + 12MP	A17 Pro	4,500mAh
18	Apple	iPhone 15 Pro 128GB	206.0	6.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,400mAh
19	Apple	iPhone 15 Pro 256GB	206.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,400mAh
20	Apple	iPhone 15 Pro 512GB	206.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,400mAh
21	Apple	iPhone 15 Pro Max 128GB	221.0	6.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
22	Apple	iPhone 15 Pro Max 256GB	221.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
23	Apple	iPhone 15 Pro Max 512GB	221.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
30	Apple	iPhone 14 Pro 128GB	206.0	6.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,200mAh
31	Apple	iPhone 14 Pro 256GB	206.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,200mAh
32	Apple	iPhone 14 Pro 512GB	206.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,200mAh
33	Apple	iPhone 14 Pro Max 128GB	221.0	6.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
34	Apple	iPhone 14 Pro Max 256GB	221.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
35	Apple	iPhone 14 Pro Max 512GB	221.0	8.0	12MP / 4K	48MP + 12MP	A16 Bionic	4,500mAh
42	Apple	iPhone 13 Pro 128GB	204.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	3,095mAh
43	Apple	iPhone 13 Pro 256GB	204.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	3,095mAh
44	Apple	iPhone 13 Pro 512GB	204.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	3,095mAh
45	Apple	iPhone 13 Pro Max 128GB	238.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	4,352mAh
46	Apple	iPhone 13 Pro Max 256GB	238.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	4,352mAh
47	Apple	iPhone 13 Pro Max 512GB	238.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A15 Bionic	4,352mAh
54	Apple	iPhone 12 Pro 128GB	189.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	2,815mAh
55	Apple	iPhone 12 Pro 256GB	189.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	2,815mAh
56	Apple	iPhone 12 Pro 512GB	189.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	2,815mAh
57	Apple	iPhone 12 Pro Max 128GB	228.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	3,687mAh
58	Apple	iPhone 12 Pro Max 256GB	228.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	3,687mAh
59	Apple	iPhone 12 Pro Max 512GB	228.0	6.0	12MP / 4K	12MP + 12MP + 12MP	A14 Bionic	3,687mAh
63	Apple	iPhone 11 Pro 64GB	188.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,046mAh
64	Apple	iPhone 11 Pro 256GB	188.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,046mAh
65	Apple	iPhone 11 Pro 512GB	188.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,046mAh
66	Apple	iPhone 11 Pro Max 64GB	226.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,969mAh
67	Apple	iPhone 11 Pro Max 256GB	226.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,969mAh
68	Apple	iPhone 11 Pro Max 512GB	226.0	4.0	12MP / 4K	12MP + 12MP + 12MP	A13 Bionic	3,969mAh

это рекламный ход компании Apple с 4к, можно смело убирать из названия

```
In [41]: df[df['front_camera_mp'].isin(['Dual 32MP', 'Dual 60MP'])].iloc[:, :8]
```

Out[41]:	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
632	Huawei	Nova 9 Pro	186.0	8.0	Dual 32MP	50MP	Snapdragon 778G 4G	4,000mAh
638	Huawei	Nova 10 Pro	191.0	8.0	Dual 60MP	50MP	Snapdragon 778G 4G	4,500mAh

```
In [42]: df[
    df['front_camera_mp'].isin(['Dual 32MP', 'Dual 60MP', '32MP', '60MP']) &
    df['model_name'].str.contains(r'Nova 9|Nova 10', na=False)
].iloc[:, :8]
```

Out[42]:	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
631	Huawei	Nova 9	175.0	8.0	32MP	50MP	Snapdragon 778G 4G	4,300mAh
632	Huawei	Nova 9 Pro	186.0	8.0	Dual 32MP	50MP	Snapdragon 778G 4G	4,000mAh
637	Huawei	Nova 10	168.0	8.0	60MP	50MP	Snapdragon 778G 4G	4,000mAh
638	Huawei	Nova 10 Pro	191.0	8.0	Dual 60MP	50MP	Snapdragon 778G 4G	4,500mAh

можно удалять Dual

In [43]: df[(df['front_camera_mp'] == '10MP, 4MP (UDC)') | (df['front_camera_mp'] == '4MP') | (df['front_camera_mp'] == '10MP')].iloc[:, :8]

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_mp	back_camera_mp	processor	battery_capacity_mah
109	Samsung	Galaxy S22 Ultra 128GB	228.0	12.0	10MP	108MP + 12MP	Exynos 2200	5000mAh
110	Samsung	Galaxy S22 Ultra 256GB	228.0	12.0	10MP	108MP + 12MP	Exynos 2200	5000mAh
111	Samsung	Galaxy S22+ 128GB	195.0	8.0	10MP	50MP + 12MP	Exynos 2200	4500mAh
112	Samsung	Galaxy S22+ 256GB	195.0	8.0	10MP	50MP + 12MP	Exynos 2200	4500mAh
113	Samsung	Galaxy S22 128GB	167.0	8.0	10MP	50MP + 12MP	Exynos 2200	3800mAh
114	Samsung	Galaxy S22 256GB	167.0	8.0	10MP	50MP + 12MP	Exynos 2200	3800mAh
115	Samsung	Galaxy Z Fold 5 256GB	254.0	12.0	10MP	50MP + 12MP	Snapdragon 8 Gen 2	4400mAh
116	Samsung	Galaxy Z Fold 5 512GB	254.0	12.0	10MP	50MP + 12MP	Snapdragon 8 Gen 2	4400mAh
119	Samsung	Galaxy Z Fold 4 256GB	263.0	12.0	10MP	50MP + 12MP	Snapdragon 8 Gen 1	4400mAh
120	Samsung	Galaxy Z Fold 4 512GB	263.0	12.0	10MP	50MP + 12MP	Snapdragon 8 Gen 1	4400mAh
121	Samsung	Galaxy Z Flip 4 256GB	187.0	8.0	10MP	12MP + 12MP	Snapdragon 8 Gen 1	3700mAh
122	Samsung	Galaxy Z Flip 4 512GB	187.0	8.0	10MP	12MP + 12MP	Snapdragon 8 Gen 1	3700mAh
147	Samsung	Galaxy Note 20 Ultra 128GB	208.0	12.0	10MP	108MP + 12MP	Exynos 990	4500mAh
148	Samsung	Galaxy Note 20 Ultra 256GB	208.0	12.0	10MP	108MP + 12MP	Exynos 990	4500mAh
149	Samsung	Galaxy Note 20 128GB	192.0	8.0	10MP	108MP + 12MP	Exynos 990	4300mAh
150	Samsung	Galaxy Note 20 256GB	192.0	8.0	10MP	108MP + 12MP	Exynos 990	4300mAh
151	Samsung	Galaxy Note 10+ 256GB	196.0	12.0	10MP	12MP + 16MP	Exynos 9825	4300mAh
152	Samsung	Galaxy Note 10+ 512GB	196.0	12.0	10MP	12MP + 16MP	Exynos 9825	4300mAh
153	Samsung	Galaxy Note 10 256GB	168.0	8.0	10MP	12MP + 16MP	Exynos 9825	3500mAh
154	Samsung	Galaxy Note 10 128GB	168.0	8.0	10MP	12MP + 16MP	Exynos 9825	3500mAh
164	Samsung	Galaxy W22 5G 256GB	228.0	12.0	10MP	108MP + 12MP	Snapdragon 888	4500mAh
165	Samsung	Galaxy W21 5G 256GB	229.0	12.0	10MP	108MP + 12MP	Snapdragon 888	4500mAh
927	Samsung	Galaxy Z Fold6 256GB	239.0	12.0	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
928	Samsung	Galaxy Z Fold6 512GB	239.0	12.0	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh
929	Samsung	Galaxy Z Fold6 1TB	239.0	12.0	10MP, 4MP (UDC)	50MP	Snapdragon 8 Gen 3	4400mAh

- видно из сравнения телефон, UDC является лишним дополнением только для Samsung Galaxy Z Fold6. Можно это слово удалять

Ниже удалим лишнее из колонки front_camera_mp и преобразуем

In [44]: df['front_camera_mp'] = df['front_camera_mp'].replace('10MP, 4MP (UDC)', '10MP + 4MP')
df['front_camera_mp'] = df['front_camera_mp'].replace('20MP+8MP', '20MP + 8MP')
df['front_camera_mp'] = df['front_camera_mp'].replace('Dual 32MP', '32MP')
df['front_camera_mp'] = df['front_camera_mp'].replace('Dual 60MP', '60MP')
df['front_camera_mp'] = df['front_camera_mp'].replace('12MP / 4K', '12MP')

print("📦 Уникальные значения:", df['front_camera_mp'].unique())

📦 Уникальные значения: ['12MP' '7MP' '10MP' '32MP' '13MP' '5MP' '16MP' '8MP' '12MP + 12MP' '2MP' '44MP' '24MP' '20MP + 8MP' '20MP' '50MP' '25MP' '60MP' '10.7MP' '60MP + 8MP' '11.1MP' '10.8MP' '10.5MP' '48MP' '42MP' '10MP + 4MP']

таким образом у нас телефоны делятся на с одной фронтальной камерой и с двумя фронтальными камерами

- предлагаю разделить эти камеры на два столбца front_camera_1_mp и front_camera_2_mp, а также добавить столбец front_camera_number со значением 1 для front_camera_1_mp и 2 для front_camera_2_mp

In [45]: # Функция для разделения по '+'
def split_front_camera_plus(value):
 parts = value.split('+')
 parts = [p.strip() for p in parts if p.strip()]

 first = parts[0] if len(parts) > 0 else None
 second = parts[1] if len(parts) > 1 else None

```

number = len(parts)

return pd.Series([first, second, number])

# Добавляем новые колонки
df[['front_camera_1_mp', 'front_camera_2_mp', 'front_camera_number']] = df['front_camera_mp'].apply(split_front_camera_plus)

# Переупорядочиваем колонки: вставляем новые сразу после front_camera_mp
cols = df.columns.tolist()
insert_at = cols.index('front_camera_mp') + 1

# Удаляем новые колонки из списка, если они уже есть
for col in ['front_camera_1_mp', 'front_camera_2_mp', 'front_camera_number']:
    if col in cols:
        cols.remove(col)

# Вставляем новые колонки в нужное место
new_cols = ['front_camera_1_mp', 'front_camera_2_mp', 'front_camera_number']
for i, col in enumerate(new_cols):
    cols.insert(insert_at + i, col)

# Применяем новый порядок
df = df[cols]

# Удаляем front_camera_mp
df = df.drop(columns='front_camera_mp')

```

In [46]: # Выбираем первые 9 колонок
df.iloc[:, :7]

Out[46]:

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_1_mp	front_camera_2_mp	front_camera_number
0	Apple	iPhone 16 128GB	174.0	6.0	12MP	None	1
1	Apple	iPhone 16 256GB	174.0	6.0	12MP	None	1
2	Apple	iPhone 16 512GB	174.0	6.0	12MP	None	1
3	Apple	iPhone 16 Plus 128GB	203.0	6.0	12MP	None	1
4	Apple	iPhone 16 Plus 256GB	203.0	6.0	12MP	None	1
...
927	Samsung	Galaxy Z Fold6 256GB	239.0	12.0	10MP	4MP	2
928	Samsung	Galaxy Z Fold6 512GB	239.0	12.0	10MP	4MP	2
929	Samsung	Galaxy Z Fold6 1TB	239.0	12.0	10MP	4MP	2
930	Huawei	P60 Pro	200.0	12.0	13MP	None	1
931	Huawei	P60 Art	206.0	12.0	13MP	None	1

932 rows × 7 columns

In [47]: df = clean_column_units(df, 'front_camera_1_mp', 'MP')

📦 Уникальные значения в front_camera_1_mp (отсортированные):
[2.0, 5.0, 7.0, 8.0, 10.0, 10.5, 10.7, 10.8, 11.1, 12.0, 13.0, 16.0, 20.0, 24.0, 25.0, 32.0, 42.0, 44.0, 48.0, 50.0, 60.0]
🔍 Тип данных front_camera_1_mp: float64

In [48]:

```

df['front_camera_2_mp'] = (
    df['front_camera_2_mp']
    .str.replace('MP', '', regex=False) # удаляем символ 'g'
    .str.strip() # убираем пробелы
    .astype(float) # преобразуем в число
)

# Выбираем уникальные значения и тип данных
print("📦 Уникальные значения:", df['front_camera_2_mp'].unique())
print("🔍 Тип данных:", df['front_camera_2_mp'].dtype)

```

📦 Уникальные значения: [nan 12. 8. 4.]
🔍 Тип данных: float64

back_camera_mp

In [49]: df['back_camera_mp'].unique()

```
Out[49]: array(['48MP', '50MP + 12MP', '48MP + 12MP', '12MP + 12MP', '12MP',  
   '12MP + 12MP + 12MP', '8MP', '12MP + 10MP', '200MP + 12MP',  
   '108MP + 12MP', '48MP + 8MP', '50MP + 5MP', '50MP + 2MP',  
   '108MP + 8MP', '50MP + 8MP', '13MP + 2MP', '12MP + 16MP', '50MP',  
   '16MP', '16MP + 5MP', '13MP', '13MP + 5MP', '16MP + 8MP',  
   '13MP + 8MP', '13MP + 6MP', '5MP', '50MP + 48MP', '108MP',  
   '64MP + 2MP', '48MP + 48MP', '48MP + 50MP', '48MP + 16MP',  
   '48MP + 5MP', '64MP + 8MP', '20MP + 16MP', '16MP + 20MP',  
   '50MP + 16MP', '200MP', '64MP', '13MP+2MP', '48MP + 64MP + 48MP',  
   '50MP + 32MP + 48MP', '50MP + 50MP + 50MP', '50MP + 50MP',  
   '50MP + 8MP + 2MP', '50MP + 50MP + 8MP', '50MP + 32MP + 8MP',  
   '64MP + 8MP + 2MP', '8MP + 2MP', '50MP + 50MP + 64MP',  
   '50MP + 50MP + 13MP', '50MP + 48MP + 32MP', '64MP + 32MP + 8MP',  
   '50MP + 64MP + 8MP', '64MP + 8MP + 2MP + 2MP',  
   '50MP + 13MP + 16MP + 2MP', '50MP + 16MP + 13MP + 2MP',  
   '48MP + 8MP + 2MP + 2MP', '48MP + 13MP + 12MP',  
   '48MP + 13MP + 8MP + 2MP', '13MP + 2MP + 2MP', '16MP + 2MP + 2MP',  
   '12MP + 8MP + 2MP + 2MP', '108MP + 2MP', '64MP + 2MP + 2MP',  
   '50MP + 8MP + 50MP', '48MP + 2MP + 2MP',  
   '50MP (Main) + 50MP (Ultra-wide) + 50MP (Telephoto)',  
   '50MP (Main) + 50MP (Ultra-wide)',  
   '200MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)',  
   '108MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)',  
   '50MP (Main) + 8MP (Ultra-wide) + 2MP (Macro)',  
   '50MP (Main) + 2MP (Depth)', '40MP',  
   '48MP (wide) + 13MP (ultrawide) + 48MP (telephoto)',  
   '48MP (wide) + 40MP (ultrawide) + 48MP (telephoto)',  
   '50MP (wide) + 13MP (ultrawide) + 12MP (periscope telephoto)',  
   '50MP (wide) + 12MP (ultrawide) + 12MP (telephoto)',  
   '50MP (wide) + 40MP (ultrawide) + 48MP (telephoto)',  
   '50MP (wide) + 48MP (ultrawide) + 48MP (telephoto)',  
   '50MP (wide) + 8MP (ultrawide)', '50MP + 13MP + 12MP',  
   '50MP + 12.5MP + 48MP', '50MP + 40MP + 50MP', '50MP + 12MP + 40MP',  
   '50MP + 12MP + 48MP', '13MP (f/1.8, AF)', '12.2MP', '54MP',  
   '160MP', '100MP'], dtype=object)
```

- ситуация аналогична с фронтовой камерой, уберём лишнее что в скобках и удаляем пробелы в начале и в конце строки

```
In [50]: def split_back_camera_modules(df, column='back_camera_mp'):  
    # Шаг 1: очистка – удаляем скобки и пробелы  
    cleaned = (  
        df[column]  
        .astype(str)  
        .str.replace(r'\(\.*?\)'), '', regex=True)  
        .str.strip()  
    )  
  
    # Шаг 2: разбиваем по '+'  
    split_modules = cleaned.str.split('+')  
  
    # Шаг 3: создаём новые колонки  
    max_modules = split_modules.map(len).max()  
    new_cols = [f'back_camera_{i+1}' for i in range(max_modules)]  
    for i, col_name in enumerate(new_cols):  
        df[col_name] = split_modules.map(lambda x: x[i].strip() if i < len(x) else None)  
  
    # Шаг 4: добавляем количество модулей  
    df['back_camera_number'] = split_modules.map(len)  
    new_cols.append('back_camera_number')  
  
    # Шаг 5: переупорядочиваем колонки – вставляем новые после back_camera_mp  
    cols = df.columns.tolist()  
    insert_at = cols.index(column) + 1  
  
    # Удаляем новые колонки из списка, если они уже есть  
    for col in new_cols:  
        if col in cols:  
            cols.remove(col)  
  
    # Вставляем новые колонки в нужное место  
    for i, col in enumerate(new_cols):  
        cols.insert(insert_at + i, col)  
  
    # Применяем новый порядок  
    df = df[cols]  
  
    return df
```

```
In [51]: df = split_back_camera_modules(df)  
  
# Удаляем back_camera_mp  
df = df.drop(columns='back_camera_mp')
```

```
In [52]: df.iloc[:, [0, 1] + list(range(7, 12))]
```

	company_name	model_name	back_camera_1	back_camera_2	back_camera_3	back_camera_4	back_camera_number
0	Apple	iPhone 16 128GB	48MP	None	None	None	1
1	Apple	iPhone 16 256GB	48MP	None	None	None	1
2	Apple	iPhone 16 512GB	48MP	None	None	None	1
3	Apple	iPhone 16 Plus 128GB	48MP	None	None	None	1
4	Apple	iPhone 16 Plus 256GB	48MP	None	None	None	1
...
927	Samsung	Galaxy Z Fold6 256GB	50MP	None	None	None	1
928	Samsung	Galaxy Z Fold6 512GB	50MP	None	None	None	1
929	Samsung	Galaxy Z Fold6 1TB	50MP	None	None	None	1
930	Huawei	P60 Pro	48MP	13MP	48MP	None	3
931	Huawei	P60 Art	48MP	40MP	48MP	None	3

932 rows × 7 columns

In [53]: `df = clean_column_units(df, 'back_camera_1', 'MP')`

📦 Уникальные значения в back_camera_1 (отсортированные):
[5.0, 8.0, 12.0, 12.2, 13.0, 16.0, 20.0, 40.0, 48.0, 50.0, 54.0, 64.0, 100.0, 108.0, 160.0, 200.0]
🔍 Тип данных back_camera_1: float64

In [54]: `df['back_camera_2'] = (
 df['back_camera_2']
 .str.replace('MP', '', regex=False) # удаляем символ 'g'
 .str.strip() # убираем пробелы
 .astype(float))`

Выводим уникальные значения и тип данных
print("📦 Уникальные значения:", df['back_camera_2'].unique())
print("🔍 Тип данных:", df['back_camera_2'].dtype)

📦 Уникальные значения: [nan 12. 10. 8. 5. 2. 16. 6. 48. 50. 20. 64. 32. 13. 40. 12.5]
🔍 Тип данных: float64

In [55]: `df['back_camera_3'] = (
 df['back_camera_3']
 .str.replace('MP', '', regex=False) # удаляем символ 'g'
 .str.strip() # убираем пробелы
 .astype(float))`

Выводим уникальные значения и тип данных
print("📦 Уникальные значения:", df['back_camera_3'].unique())
print("🔍 Тип данных:", df['back_camera_3'].dtype)

📦 Уникальные значения: [nan 12. 48. 50. 2. 8. 64. 13. 32. 16. 40.]
🔍 Тип данных: float64

In [56]: `df['back_camera_4'] = (
 df['back_camera_4']
 .str.replace('MP', '', regex=False) # удаляем символ 'g'
 .str.strip() # убираем пробелы
 .astype(float))`

Выводим уникальные значения и тип данных
print("📦 Уникальные значения:", df['back_camera_4'].unique())
print("🔍 Тип данных:", df['back_camera_4'].dtype)

📦 Уникальные значения: [nan 2.]
🔍 Тип данных: float64

processor

In [57]: `sorted(df['processor'].unique())[:10] # первые 10`

Out[57]: ['A11 Bionic',
'A12 Bionic',
'A12Z Bionic',
'A13 Bionic',
'A14 Bionic',
'A15 Bionic',
'A16 Bionic',
'A17 Bionic',
'A17 Pro',
'Dimensity 1000+']

In [58]: `sorted(df['processor'].unique())[-10:] # последние 10`

```
Out[58]: ['Spreadtrum SC8830',
 'Unisoc SC9832E',
 'Unisoc SC9863A',
 'Unisoc T606',
 'Unisoc T610',
 'Unisoc T612',
 'Unisoc T616',
 'Unisoc T618',
 'Unisoc T700',
 'Unisoc T760']
```

```
In [59]: # Объединяем все значения в одну строку
all_text = ' '.join(df['processor'].dropna().astype(str))

# Находим все символы, которые НЕ являются буквами или цифрами
non_alnum_chars = re.findall(r'[^a-zA-Z0-9]', all_text)

# Получаем уникальные символы
unique_non_alnum = sorted(set(non_alnum_chars))

print("🚫 Неалфавитные и нецифровые символы:", unique_non_alnum)
```

🚫 Неалфавитные и нецифровые символы: [' ', '+', '-']

значимых артефактов в колонке нет

```
In [60]: df[df['processor'].astype(str).str.contains(r'-|+\+', na=False)].iloc[:, [0, 1] + list(range(12, 15))]
```

```
Out[60]:   company_name      model_name    processor  battery_capacity_mah  screen_size_inches
  192      OnePlus  OnePlus 10T 256GB  Snapdragon 8+ Gen 1        4500mAh       6.7 inches
  197      OnePlus  OnePlus Nord 2 128GB  MediaTek Dimensity 1200-AI        4500mAh      6.43 inches
  203      OnePlus  OnePlus 10T 5G 256GB  Snapdragon 8+ Gen 1        4500mAh       6.7 inches
  211      OnePlus  OnePlus Nord 2 5G 128GB  MediaTek Dimensity 1200-AI        4500mAh      6.43 inches
  375       Oppo      Find N2 Flip 256GB  Dimensity 9000+        4300mAh       6.8 inches
 ...
  911       Poco          F5 128GB  Snapdragon 7+ Gen 2        5000mAh      6.67 inches
  912       Poco          F5 Pro 256GB  Snapdragon 8+ Gen 1        5160mAh      6.67 inches
  920       Poco          F6 Pro 256GB  Snapdragon 8+ Gen 2        5160mAh      6.67 inches
  930      Huawei         P60 Pro  Snapdragon 8+ Gen 1 4G        4815mAh      6.67 inches
  931      Huawei         P60 Art  Snapdragon 8+ Gen 1 4G        5100mAh      6.73 inches
```

67 rows × 5 columns

battery_capacity_mah

```
In [61]: df['battery_capacity_mah'].unique()
```

```
Out[61]: array(['3,600mAh', '4,200mAh', '4,400mAh', '4,500mAh', '3,200mAh',
 '4,300mAh', '4,325mAh', '2,438mAh', '3,240mAh', '3,095mAh',
 '4,352mAh', '2,227mAh', '2,815mAh', '3,687mAh', '3,110mAh',
 '3,046mAh', '3,969mAh', '2,716mAh', '2,658mAh', '3,174mAh',
 '2,942mAh', '7,608mAh', '8,612mAh', '5,124mAh', '7,812mAh',
 '9,720mAh', '10,307mAh', '5000mAh', '4800mAh', '4000mAh',
 '4700mAh', '3900mAh', '4500mAh', '3800mAh', '4400mAh', '3700mAh',
 '6000mAh', '4300mAh', '3500mAh', '4050mAh', '3000mAh', '3600mAh',
 '3300mAh', '2600mAh', '11200mAh', '10090mAh', '8400mAh', '8000mAh',
 '7040mAh', '5100mAh', '5050mAh', '7600mAh', '4510mAh', '4115mAh',
 '4085mAh', '9510mAh', '11000mAh', '5700mAh', '4100mAh', '3315mAh',
 '3260mAh', '2300mAh', '3055mAh', '4030mAh', '2000mAh', '3200mAh',
 '4200mAh', '4450mAh', '4600mAh', '4830mAh', '4870mAh', '8040mAh',
 '4805mAh', '5800mAh', '5600mAh', '6400mAh', '8360mAh', '4520mAh',
 '5,000mAh', '4,600mAh', '4,700mAh', '4,350mAh', '4,000mAh',
 '4,025mAh', '4,040mAh', '5,100mAh', '6,500mAh', '5,500mAh',
 '6,000mAh', '5,800mAh', '5,200mAh', '7,100mAh', '8,360mAh',
 '8,340mAh', '6,400mAh', '7,200mAh', '6,100mAh', '5,400mAh',
 '4,610mAh', '5,110mAh', '4,050mAh', '3,350mAh', '3,500mAh',
 '4,310mAh', '3,800mAh', '4,020mAh', '4,100mAh', '4,360mAh',
 '4,460mAh', '4,815mAh', '4,800mAh', '4,750mAh', '4,900mAh',
 '5,050mAh', '8200mAh', '10,100mAh', '3,000mAh', '3,700mAh',
 '2,800mAh', '3,140mAh', '3,885mAh', '4,080mAh', '4,680mAh',
 '4,614mAh', '5,003mAh', '4,410mAh', '4,355mAh', '4,385mAh',
 '4,575mAh', '5,250mAh', '8,000mAh', '7,500mAh', '7,000mAh',
 '9,000mAh', '5,300mAh', '3,750mAh', '5,450mAh', '5,550mAh',
 '7,250mAh', '8,500mAh', '8,300mAh', '10,000mAh', '10,500mAh',
 '8,850mAh', '5160mAh', '5065mAh', '5110mAh'], dtype=object)
```

в колонке следующие артефакты

- mAh
- разделитель ","

```
In [62]: df[df['battery_capacity_mah'].astype(str).str.count(',') > 1]
```

```
Out[62]: company_name model_name mobile_weight_g ram_gb front_camera_1_mp front_camera_2_mp front_camera_number back_camera_1 back_camera_2
```

0 rows × 21 columns

```
In [63]: def clean_column_units(df, column_name, unit_to_remove):
    df[column_name] = (
        df[column_name]
        .astype(str) # на случай, если есть числа
        .str.replace(unit_to_remove, '', regex=False) # удаляем единицу измерения
        .str.replace(',', '', regex=False) # удаляем запятую
        .str.strip() # убираем пробелы
        .astype(int) # преобразуем в число
    )

    # Выводим отсортированные уникальные значения
    unique_sorted = sorted(df[column_name].dropna().unique())
    print(f"📦 Уникальные значения в {column_name} (отсортированные): \n", unique_sorted)
    print(f"🔍 Тип данных {column_name}: ", df[column_name].dtype)

    return df
```

```
In [64]: df = clean_column_units(df, 'battery_capacity_mah', 'mAh')
```

📦 Уникальные значения в battery_capacity_mah (отсортированные):
[2000, 2227, 2300, 2438, 2600, 2658, 2716, 2800, 2815, 2942, 3000, 3046, 3055, 3095, 3110, 3140, 3174, 3200, 3240, 3260, 3300, 3315, 3350, 3500, 3600, 3687, 3700, 3750, 3800, 3885, 3900, 3969, 4000, 4020, 4025, 4030, 4040, 4050, 4080, 4085, 4100, 4115, 4200, 4300, 4310, 4325, 4350, 4355, 4360, 4385, 4400, 4410, 4450, 4460, 4500, 4510, 4520, 4575, 4600, 4610, 4614, 4680, 4700, 4750, 4800, 4805, 4815, 4830, 4870, 4900, 5000, 5003, 5050, 5065, 5100, 5110, 5124, 5160, 5200, 5250, 5300, 5400, 5450, 5500, 5550, 5600, 5700, 5800, 6000, 6100, 6400, 6500, 7000, 7040, 7100, 7200, 7250, 7500, 7600, 7608, 7812, 8000, 8040, 8200, 8300, 8340, 8360, 8400, 8500, 8612, 8850, 9000, 9510, 9720, 10000, 10090, 10100, 10307, 10500, 11000, 11200]
🔍 Тип данных battery_capacity_mah: int32

screen_size_inches

```
In [65]: df['screen_size_inches'].unique()
```

```
Out[65]: array(['6.1 inches', '6.7 inches', '5.4 inches', '5.8 inches',
   '6.5 inches', '10.9 inches', '10.2 inches', '7.9 inches',
   '11 inches', '12.9 inches', '13 inches', '6.8 inches',
   '6.6 inches', '7.6 inches', '6.4 inches', '6.9 inches',
   '6.3 inches', '5.3 inches', '6.0 inches', '5.5 inches',
   '5.7 inches', '5.2 inches', '14.6 inches', '12.4 inches',
   '8.7 inches', '10.5 inches', '8 inches', '10.1 inches',
   '6.74 inches', '6.72 inches', '7.8 inches', '6.55 inches',
   '6.43 inches', '6.49 inches', '6.52 inches', '6.78 inches',
   '6.59 inches', '6.44 inches', '6.41 inches', '6.01 inches',
   '6.67 inches', '6.28 inches', '11.61 inches', '6.31 inches',
   '6.58 inches', '6.38 inches', '6.56 inches', '5.88 inches',
   '6.22 inches', '5.0 inches', '6.51 inches', '6.35 inches',
   '6.53 inches', '6.39 inches', '6.47 inches', '10.4 inches',
   '12.3 inches', '7.82 inches', '6.83 inches', '11.6 inches',
   '12.1 inches', '6.82 inches', '7.1 inches', '11.5 inches',
   '6.73 inches', '6.36 inches', '6.09 inches',
   '6.7 inches (main), 2.7 inches (external)',
   '6.9 inches (internal), 4.0 inches (external)',
   '6.7 inches (internal), 3.6 inches (external)',
   '6.9 inches (unfolded)', '8.0 inches (unfolded)', '6.57 inches',
   '7.8 inches (unfolded)', '7.85 inches (unfolded)', '7.93 inches',
   '7.92 inches', '12.2 inches', '13.2 inches', '5.6 inches',
   '6.2 inches', '6.34 inches', '6.71 inches', '7.85 inches',
   '9.7 inches', '11.0 inches', '6.95 inches', '6.85 inches',
   '6.63 inches', '7.09 inches', '6.81 inches', '6.76 inches',
   '12.0 inches', '12.6 inches', '13.0 inches', '13.5 inches',
   '6.79 inches'], dtype=object)
```

- встречаются варианты с 2 экранами, посмотрим на них

```
In [66]: df[(df['screen_size_inches'] == '6.7 inches (main), 2.7 inches (external)') |
      (df['screen_size_inches'] == '6.9 inches (internal), 4.0 inches (external)') |
      (df['screen_size_inches'] == '6.7 inches (internal), 3.6 inches (external))].iloc[:, [0, 1, 14]]
```

```
Out[66]:   company_name model_name screen_size_inches
568     Motorola    Razr 128GB 6.7 inches (main), 2.7 inches (external)
569     Motorola    Razr 256GB 6.7 inches (main), 2.7 inches (external)
598     Motorola  Razr 50 Ultra 512GB 6.9 inches (internal), 4.0 inches (external)
599     Motorola  Razr 50 256GB 6.7 inches (internal), 3.6 inches (external)
```

- телефоны Motorola модели Razr.
- классический плоский смартфон, но имеют на обратной стороне второй дисплей под камерой.
- нет необходимости учитывать этот дисплей в анализе, поэтому удалим его.

```
In [67]: # Преобразуем колонку, извлекая только первое число
df['screen_size_inches'] = (
    df['screen_size_inches']
    .astype(str)
    .str.extract(r'(\d+(:\.\d+)?)')[0] # извлекаем первое число (целое или с точкой)
    .astype(float)
)

# Проверим результат
print("📦 Уникальные значения: \n", sorted(df['screen_size_inches'].dropna().unique()))
print("🔍 Тип данных:", df['screen_size_inches'].dtype)
```

📦 Уникальные значения:

[5.0, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.88, 6.0, 6.01, 6.09, 6.1, 6.2, 6.22, 6.28, 6.3, 6.31, 6.34, 6.35, 6.36, 6.38, 6.39, 6.4, 6.41, 6.43, 6.44, 6.47, 6.49, 6.5, 6.51, 6.52, 6.53, 6.55, 6.56, 6.57, 6.58, 6.59, 6.6, 6.63, 6.67, 6.7, 6.71, 6.72, 6.73, 6.74, 6.76, 6.78, 6.79, 6.8, 6.81, 6.82, 6.83, 6.85, 6.9, 6.95, 7.09, 7.1, 7.6, 7.8, 7.82, 7.85, 7.9, 7.92, 7.93, 8.0, 8.7, 9.7, 10.1, 10.2, 10.4, 10.5, 10.9, 11.0, 11.5, 11.6, 11.61, 12.0, 12.1, 12.2, 12.3, 12.4, 12.6, 12.9, 13.0, 13.2, 13.5, 14.6]

🔍 Тип данных: float64

с годом выпуска все нормально, осталось разобраться с ценой

price_pakistan_pkrs

```
In [68]: df['price_pakistan_pkrs'].unique()
```

```
Out[68]: array(['PKR 224,999', 'PKR 234,999', 'PKR 244,999', 'PKR 249,999',
   'PKR 259,999', 'PKR 274,999', 'PKR 284,999', 'PKR 294,999',
   'PKR 314,999', 'PKR 324,999', 'PKR 344,999', 'PKR 204,999',
   'PKR 214,999', 'PKR 264,999', 'PKR 304,999', 'PKR 184,999',
   'PKR 194,999', 'PKR 364,999', 'PKR 174,999', 'PKR 354,999',
   'PKR 384,999', 'PKR 334,999', 'PKR 159,999', 'PKR 169,999',
   'PKR 179,999', 'PKR 219,999', 'PKR 239,999', 'PKR 269,999',
   'PKR 289,999', 'PKR 309,999', 'PKR 164,999', 'PKR 149,999',
   'PKR 79,999', 'PKR 89,999', 'PKR 49,999', 'PKR 59,999',
   'PKR 69,999', 'PKR 189,999', 'PKR 279,999', 'PKR 359,999',
   'PKR 399,999', 'PKR 450,000', 'PKR 480,000', 'PKR 400,000',
   'PKR 430,000', 'PKR 350,000', 'PKR 380,000', 'PKR 460,000',
   'PKR 360,000', 'PKR 390,000', 'PKR 320,000', 'PKR 420,000',
   'PKR 300,000', 'PKR 330,000', 'PKR 500,000', 'PKR 550,000',
   'PKR 530,000', 'PKR 85,000', 'PKR 95,000', 'PKR 75,000',
   'PKR 70,000', 'PKR 80,000', 'PKR 60,000', 'PKR 50,000',
   'PKR 105,000', 'PKR 180,000', 'PKR 200,000', 'PKR 150,000',
   'PKR 170,000', 'PKR 160,000', 'PKR 140,000', 'PKR 120,000',
   'PKR 45,000', 'PKR 40,000', 'PKR 25,000', 'PKR 65,000',
   'PKR 280,000', 'PKR 260,000', 'PKR 230,000', 'PKR 250,000',
   'PKR 100,000', 'PKR 68,000', 'PKR 35,000', 'PKR 28,000',
   'PKR 119,999', 'PKR 55,000', 'PKR 299,999', 'PKR 199,999',
   'PKR 39,999', 'PKR 29,999', 'PKR 139,999', 'PKR 64,999',
   'PKR 54,999', 'PKR 109,999', 'PKR 99,999', 'PKR 94,999',
   'PKR 229,999', 'PKR 104,999', 'PKR 34,999', 'PKR 42,999',
   'PKR 22,999', 'PKR 24,999', 'PKR 19,999', 'PKR 27,999',
   'PKR 18,999', 'PKR 44,999', 'PKR 18,499', 'PKR 22,499',
   'PKR 84,999', 'PKR 85,999', 'PKR 74,999', 'PKR 26,999',
   'PKR 58,999', 'PKR 62,999', 'PKR 129,999', 'PKR 32,999',
   'PKR 25,999', 'PKR 23,999', 'PKR 21,999', 'PKR 37,999',
   'PKR 47,999', 'PKR 52,999', 'PKR 36,999', 'PKR 124,999',
   'PKR 57,999', 'PKR 61,999', 'PKR 56,999', 'PKR 209,999',
   'PKR 349,999', 'PKR 389,999', 'PKR 134,999', 'PKR 48,999',
   'PKR 46,999', 'PKR 38,999', 'PKR 429,999', 'PKR 319,999',
   'PKR 449,999', 'PKR 339,999', 'PKR 469,999', 'PKR 52,000',
   'PKR 161,500', 'PKR 197,000', 'PKR 55,999', 'PKR 45,999',
   'PKR 72,999', 'PKR 35,999', 'PKR 28,999', 'PKR 30,999',
   'PKR 41,999', 'PKR 31,999', 'PKR 15,999', 'PKR 369,999',
   'PKR 66,220', 'PKR 71,220', 'PKR 604,999', 'PKR 544,999',
   'Not available'], dtype=object)
```

- есть значени Not available, посмотрим на него

```
In [69]: df[df['price_pakistan_pkrs'].astype(str).str.count(',') > 1]
```

```
Out[69]: company_name  model_name  mobile_weight_g  ram_gb  front_camera_1_mp  front_camera_2_mp  front_camera_number  back_camera_1  back_camera_2
```

0 rows × 21 columns

```
In [70]: df.iloc[928:930, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pkrs	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
928	Samsung	Galaxy Z Fold6 512GB	PKR 544,999	INR 176,999	CNY 15,999	USD 1719	AED 7,699	2024
929	Samsung	Galaxy Z Fold6 1TB	Not available	INR 200,999	CNY 17,999	USD 2,259	AED 8,699	2024

- значение цены нет, но есть цена в долларах
- на следующем шаге мы перечисляем по курсу эту цену
- а пока заменим на nan

```
In [71]: df['price_pakistan_pk'] = df['price_pakistan_pk'].replace('Not available', np.nan)
df.iloc[928:930, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
928	Samsung	Galaxy Z Fold6 512GB	PKR 544,999	INR 176,999	CNY 15,999	USD 1719	AED 7,699	2024
929	Samsung	Galaxy Z Fold6 1TB		NaN	INR 200,999	CNY 17,999	USD 2,259	AED 8,699

```
In [72]: df['price_pakistan_pk'] = (
    df['price_pakistan_pk']
    .astype(str)
    .str.replace('PKR', '', regex=False) # удаляем 'PKR'
    .str.replace(',', '', regex=False) # удаляем запятые
    .str.strip() # убираем пробелы
    .astype(float) # преобразуем в число
)

# Выводим отсортированные уникальные значения и тип данных
print("📦 Уникальные значения(отсортированные): \n", sorted(df['price_pakistan_pk'].dropna().unique()))
print("🔍 Тип данных:", df['price_pakistan_pk'].dtype)
```

📦 Уникальные значения(отсортированные):

```
[15999.0, 18499.0, 18999.0, 19999.0, 21999.0, 22499.0, 22999.0, 23999.0, 24999.0, 25000.0, 25999.0, 26999.0, 27999.0, 28000.0, 28999.0, 29999.0, 30999.0, 31999.0, 32999.0, 34999.0, 35000.0, 35999.0, 36999.0, 37999.0, 38999.0, 39999.0, 40000.0, 41999.0, 42999.0, 44999.0, 45000.0, 45999.0, 46999.0, 47999.0, 48999.0, 49999.0, 50000.0, 52000.0, 52999.0, 54999.0, 55000.0, 55999.0, 56999.0, 57999.0, 58999.0, 59999.0, 60000.0, 61999.0, 62999.0, 64999.0, 65000.0, 66220.0, 68000.0, 69999.0, 70000.0, 71220.0, 72999.0, 74999.0, 75000.0, 79999.0, 80000.0, 84999.0, 85000.0, 85999.0, 89999.0, 94999.0, 95000.0, 99999.0, 100000.0, 104999.0, 105000.0, 109999.0, 119999.0, 120000.0, 124999.0, 129999.0, 134999.0, 139999.0, 140000.0, 149999.0, 150000.0, 159999.0, 160000.0, 161500.0, 164999.0, 169999.0, 170000.0, 174999.0, 179999.0, 180000.0, 184999.0, 189999.0, 194999.0, 197000.0, 199999.0, 200000.0, 204999.0, 209999.0, 214999.0, 219999.0, 224999.0, 229999.0, 230000.0, 234999.0, 239999.0, 244999.0, 249999.0, 250000.0, 259999.0, 260000.0, 264999.0, 269999.0, 274999.0, 279999.0, 280000.0, 284999.0, 289999.0, 294999.0, 299999.0, 300000.0, 304999.0, 309999.0, 314999.0, 319999.0, 320000.0, 324999.0, 330000.0, 334999.0, 339999.0, 344999.0, 349999.0, 350000.0, 354999.0, 359999.0, 360000.0, 364999.0, 369999.0, 380000.0, 384999.0, 389999.0, 390000.0, 399999.0, 400000.0, 420000.0, 429999.0, 430000.0, 449999.0, 450000.0, 460000.0, 469999.0, 480000.0, 500000.0, 530000.0, 544999.0, 550000.0, 604999.0]
```

🔍 Тип данных: float64

price_india_inr

```
In [73]: df['price_india_inr'].unique()
```

```
Out[73]: array(['INR 79,999', 'INR 84,999', 'INR 89,999', 'INR 94,999',
   'INR 104,999', 'INR 99,999', 'INR 114,999', 'INR 109,999',
   'INR 124,999', 'INR 74,999', 'INR 119,999', 'INR 134,999',
   'INR 69,999', 'INR 144,999', 'INR 69,900', 'INR 74,900',
   'INR 84,900', 'INR 94,900', 'INR 119,900', 'INR 129,900',
   'INR 139,900', 'INR 149,900', 'INR 64,900', 'INR 99,900',
   'INR 109,900', 'INR 89,900', 'INR 54,900', 'INR 29,900',
   'INR 39,900', 'INR 49,900', 'INR 71,900', 'INR 159,900',
   'INR 179,900', 'INR 199,900', 'INR 104,900', 'INR 114,900',
   'INR 79,900', 'INR 154,900', 'INR 174,900', 'INR 134,900',
   'INR 27,999', 'INR 32,999', 'INR 22,999', 'INR 19,999',
   'INR 24,999', 'INR 18,499', 'INR 21,999', 'INR 14,999',
   'INR 23,999', 'INR 26,999', 'INR 17,999', 'INR 20,999',
   'INR 12,999', 'INR 15,999', 'INR 18,999', 'INR 1,04,999',
   'INR 77,999', 'INR 64,999', 'INR 59,999', 'INR 28,999',
   'INR 17,499', 'INR 18,990', 'INR 9,990', 'INR 32,990',
   'INR 22,990', 'INR 19,990', 'INR 1,14,999', 'INR 85,999',
   'INR 49,999', 'INR 1,09,999', 'INR 16,999', 'INR 11,999',
   'INR 54,999', 'INR 44,999', 'INR 9,999', 'INR 8,499', 'INR 34,999',
   'INR 39,999', 'INR 45,999', 'INR 42,999', 'INR 41,999',
   'INR 38,999', 'INR 50,999', 'INR 37,999', 'INR 36,999',
   'INR 58,999', 'INR 129,999', 'INR 139,999', 'INR 149,999',
   'INR 29,999', 'INR 13,999', 'INR 10,999', 'INR 27,990',
   'INR 31,990', 'INR 33,990', 'INR 10,990', 'INR 13,990',
   'INR 24,990', 'INR 13,490', 'INR 26,990', 'INR 35,990',
   'INR 39,990', 'INR 14,990', 'INR 11,990', 'INR 21,990',
   'INR 25,999', 'INR 47,999', 'INR 31,999', 'INR 43,999',
   'INR 159,999', 'INR 52,999', 'INR 46,999', 'INR 19,000',
   'INR 8,999', 'INR 18,000', 'INR 20,000', 'INR 16,990',
   'INR 65,999', 'INR 28,756', 'INR 30,999', 'INR 19,499',
   'INR 13,499', 'INR 53,999', 'INR 33,999', 'INR 35,999',
   'INR 7,499', 'INR 21,400', 'INR 249,999', 'INR 199,999',
   'INR 259,999', 'INR 274,999', 'INR 67,999', 'INR 12,499',
   'INR 9,499', 'INR 49,990', 'INR 58,590', 'INR 179,999',
   'INR 6,999', 'INR 169,999', 'INR 11,499', 'INR 10,499',
   'INR 7,999', 'INR 5,999', 'INR 72,999', 'INR 164,999',
   'INR 176,999', 'INR 200,999], dtype=object)
```

```
In [74]: df[df['price_india_inr'].astype(str).str.count(',') > 1].iloc[:, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
148	Samsung	Galaxy Note 20 Ultra 256GB		200000.0	INR 1,04,999	CNY 8,499	USD 1,399	AED 4,999
166	Samsung	Galaxy Tab S9 Ultra 256GB		350000.0	INR 1,14,999	CNY 9,999	USD 1,199	AED 4,499
170	Samsung	Galaxy Tab S8 Ultra 256GB		320000.0	INR 1,09,999	CNY 9,499	USD 1,099	AED 4,299

- первая запятая это явно ошибка внесения данных, ее нужно удалить
- дополнительный вывод ниже это подтверждает

```
In [75]: df[df['model_name'].str.contains(r'Galaxy Note 20', na=False)].iloc[:, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
147	Samsung	Galaxy Note 20 Ultra 128GB	180000.0	INR 99,999	CNY 7,999	USD 1,299	AED 4,499	2020
148	Samsung	Galaxy Note 20 Ultra 256GB	200000.0	INR 1,04,999	CNY 8,499	USD 1,399	AED 4,999	2020
149	Samsung	Galaxy Note 20 128GB	150000.0	INR 77,999	CNY 6,499	USD 999	AED 3,999	2020
150	Samsung	Galaxy Note 20 256GB	170000.0	INR 89,999	CNY 7,099	USD 1,099	AED 4,199	2020

```
In [76]: df[df['model_name'].str.contains(r'Galaxy Tab S9', na=False)].iloc[:, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
166	Samsung	Galaxy Tab S9 Ultra 256GB	350000.0	INR 1,14,999	CNY 9,999	USD 1,199	AED 4,499	2023
167	Samsung	Galaxy Tab S9+ 256GB	280000.0	INR 99,999	CNY 8,799	USD 999	AED 3,999	2023
168	Samsung	Galaxy Tab S9 128GB	230000.0	INR 85,999	CNY 7,299	USD 799	AED 3,299	2023
169	Samsung	Galaxy Tab S9 FE 128GB	150000.0	INR 49,999	CNY 5,499	USD 549	AED 1,999	2023

```
In [77]: df[df['model_name'].str.contains(r'Galaxy Tab S8', na=False)].iloc[:, [0, 1] + list(range(15, 21))]
```

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
170	Samsung	Galaxy Tab S8 Ultra 256GB	320000.0	INR 1,09,999	CNY 9,499	USD 1,099	AED 4,299	2022
171	Samsung	Galaxy Tab S8+ 256GB	250000.0	INR 94,999	CNY 8,299	USD 899	AED 3,699	2022
172	Samsung	Galaxy Tab S8 128GB	200000.0	INR 74,999	CNY 6,999	USD 699	AED 2,999	2022

```
In [78]: df['price_india_inr'].unique()
```

```
Out[78]: array(['INR 79,999', 'INR 84,999', 'INR 89,999', 'INR 94,999',
 'INR 104,999', 'INR 99,999', 'INR 114,999', 'INR 109,999',
 'INR 124,999', 'INR 74,999', 'INR 119,999', 'INR 134,999',
 'INR 69,999', 'INR 144,999', 'INR 69,900', 'INR 74,900',
 'INR 84,900', 'INR 94,900', 'INR 119,900', 'INR 129,900',
 'INR 139,900', 'INR 149,900', 'INR 64,900', 'INR 99,900',
 'INR 109,900', 'INR 89,900', 'INR 54,900', 'INR 29,900',
 'INR 39,900', 'INR 49,900', 'INR 71,900', 'INR 159,900',
 'INR 179,900', 'INR 199,900', 'INR 104,900', 'INR 114,900',
 'INR 79,900', 'INR 154,900', 'INR 174,900', 'INR 134,900',
 'INR 27,999', 'INR 32,999', 'INR 22,999', 'INR 19,999',
 'INR 24,999', 'INR 18,499', 'INR 21,999', 'INR 14,999',
 'INR 23,999', 'INR 26,999', 'INR 17,999', 'INR 20,999',
 'INR 12,999', 'INR 15,999', 'INR 18,999', 'INR 1,04,999',
 'INR 77,999', 'INR 64,999', 'INR 59,999', 'INR 28,999',
 'INR 17,499', 'INR 18,990', 'INR 9,990', 'INR 32,990',
 'INR 22,990', 'INR 19,990', 'INR 1,14,999', 'INR 85,999',
 'INR 49,999', 'INR 1,09,999', 'INR 16,999', 'INR 11,999',
 'INR 54,999', 'INR 44,999', 'INR 9,999', 'INR 8,499', 'INR 34,999',
 'INR 39,999', 'INR 45,999', 'INR 42,999', 'INR 41,999',
 'INR 38,999', 'INR 50,999', 'INR 37,999', 'INR 36,999',
 'INR 58,999', 'INR 129,999', 'INR 139,999', 'INR 149,999',
 'INR 29,999', 'INR 13,999', 'INR 10,999', 'INR 27,990',
 'INR 31,990', 'INR 33,990', 'INR 10,990', 'INR 13,990',
 'INR 24,990', 'INR 13,490', 'INR 26,990', 'INR 35,990',
 'INR 39,990', 'INR 14,990', 'INR 11,990', 'INR 21,990',
 'INR 25,999', 'INR 47,999', 'INR 31,999', 'INR 43,999',
 'INR 159,999', 'INR 52,999', 'INR 46,999', 'INR 19,000',
 'INR 8,999', 'INR 18,000', 'INR 20,000', 'INR 16,990',
 'INR 65,999', 'INR 28,756', 'INR 30,999', 'INR 19,499',
 'INR 13,499', 'INR 53,999', 'INR 33,999', 'INR 35,999',
 'INR 7,499', 'INR 21,400', 'INR 249,999', 'INR 199,999',
 'INR 259,999', 'INR 274,999', 'INR 67,999', 'INR 12,499',
 'INR 9,499', 'INR 49,990', 'INR 58,590', 'INR 179,999',
 'INR 6,999', 'INR 169,999', 'INR 11,499', 'INR 10,499',
 'INR 7,999', 'INR 5,999', 'INR 72,999', 'INR 164,999',
 'INR 176,999', 'INR 200,999'], dtype=object)
```

```
In [79]: df['price_india_inr'] = (
    df['price_india_inr']
        .astype(str)
        .str.replace('INR', '', regex=False) # удаляем 'PKR'
        .str.replace(',', '', regex=False) # удаляем запятые
        .str.strip() # убираем пробелы
        .astype(float) # преобразуем в число
)

# Выводим отсортированные уникальные значения и тип данных
print("📦 Уникальные значения(отсортированные): \n", sorted(df['price_india_inr'].dropna().unique()))
print("🔍 Тип данных:", df['price_india_inr'].dtype)
```

📦 Уникальные значения(отсортированные):
[5999.0, 6999.0, 7499.0, 7999.0, 8499.0, 8999.0, 9499.0, 9990.0, 9999.0, 10499.0, 10990.0, 10999.0, 11499.0, 11990.0, 11999.0, 12499.0, 1299.0, 13490.0, 13499.0, 13990.0, 13999.0, 14990.0, 14999.0, 15999.0, 16990.0, 16999.0, 17499.0, 17999.0, 18000.0, 18499.0, 18990.0, 18999.0, 19000.0, 19499.0, 19990.0, 19999.0, 20000.0, 20999.0, 21400.0, 21990.0, 21999.0, 22990.0, 22999.0, 23999.0, 24990.0, 24999.0, 25999.0, 26990.0, 26999.0, 27990.0, 27999.0, 28756.0, 28999.0, 29900.0, 29999.0, 30999.0, 31990.0, 31999.0, 32990.0, 32999.0, 33990.0, 33999.0, 34999.0, 35990.0, 35999.0, 36999.0, 37999.0, 38999.0, 39900.0, 39990.0, 39999.0, 41999.0, 42999.0, 43999.0, 44999.0, 45999.0, 46999.0, 47999.0, 49900.0, 49990.0, 49999.0, 50999.0, 52999.0, 53999.0, 54900.0, 54999.0, 58590.0, 58999.0, 59999.0, 64900.0, 64999.0, 65999.0, 67999.0, 69900.0, 69999.0, 71900.0, 72999.0, 74900.0, 74999.0, 77999.0, 79900.0, 79999.0, 84900.0, 84999.0, 85999.0, 89900.0, 89999.0, 94900.0, 94999.0, 99900.0, 99999.0, 104900.0, 104999.0, 109900.0, 109999.0, 114900.0, 114999.0, 119900.0, 119999.0, 124999.0, 129900.0, 134900.0, 139900.0, 139999.0, 144999.0, 149900.0, 149999.0, 154900.0, 159900.0, 159999.0, 164999.0, 169999.0, 174900.0, 176999.0, 179900.0, 17999.0, 199900.0, 199999.0, 200999.0, 249999.0, 259999.0, 274999.0]

🔍 Тип данных: float64

price_china_cny

In [80]: df['price_china_cny'].unique()

Out[80]: array(['CNY 5,799', 'CNY 6,099', 'CNY 6,499', 'CNY 6,199', 'CNY 6,999', 'CNY 7,099', 'CNY 7,499', 'CNY 7,799', 'CNY 8,199', 'CNY 5,299', 'CNY 5,599', 'CNY 5,999', 'CNY 5,699', 'CNY 6,799', 'CNY 7,299', 'CNY 7,999', 'CNY 8,699', 'CNY 5,199', 'CNY 5,499', 'CNY 5,899', 'CNY 6,299', 'CNY 7,199', 'CNY 8,099', 'CNY 8,499', 'CNY 8,999', 'CNY 7,699', 'CNY 8,299', 'CNY 8,799', 'CNY 9,199', 'CNY 8,399', 'CNY 7,399', 'CNY 8,388', 'CNY 8,788', 'CNY 9,188', 'CNY 9,788', 'CNY 10,388', 'CNY 10,088', 'CNY 10,688', 'CNY 11,288', 'CNY 3,299', 'CNY 3,699', 'CNY 2,299', 'CNY 2,699', 'CNY 3,499', 'CNY 3,799', 'CNY 6,399', 'CNY 9,799', 'CNY 10,399', 'CNY 11,199', 'CNY 6,599', 'CNY 11,999', 'CNY 12,999', 'CNY 10,999', 'CNY 2,199', 'CNY 2,499', 'CNY 1,799', 'CNY 1,999', 'CNY 1,599', 'CNY 1,499', 'CNY 1,699', 'CNY 1,199', 'CNY 1,399', 'CNY 2,099', 'CNY 1,099', 'CNY 1,299', 'CNY 2,399', 'CNY 999', 'CNY 3,099', 'CNY 1,899', 'CNY 9,999', 'CNY 9,499', 'CNY 4,299', 'CNY 2,999', 'CNY 899', 'CNY 799', 'CNY 4,999', 'CNY 2,799', 'CNY 4,699', 'CNY 4,199', 'CNY 3,199', 'CNY 3,899', 'CNY 3,999', 'CNY 4,499', 'CNY 4,599', 'CNY 3,599', 'CNY 3,399', 'CNY 3,400', 'CNY 3,600', 'CNY 2,800', 'CNY 2,900', 'CNY 3,000', 'CNY 2,700', 'CNY 2,600', 'CNY 2,400', 'CNY 2,500', 'CNY 2,000', 'CNY 2,100', 'CNY 1,800', 'CNY 1,900', 'CNY 1,500', 'CNY 1,600', 'CNY 2,200', 'CNY 1,400', 'CNY 1,700', 'CNY 1,200', 'CNY 1,100', 'CNY 900', 'CNY 1,300', 'CNY 1,050', 'CNY 3,200', 'CNY 2,300', 'CNY 1,350', 'CNY 1,550', 'CNY 4,799', 'CNY 2,899', 'CNY 1,998', 'CNY 4,488', 'CNY 5,988', 'CNY 8,988', 'CNY 17,999', 'CNY 4,088', 'CNY 14,999', 'CNY 6,988', 'CNY 7,988', 'CNY 13,999', 'CNY 9,988', 'CNY 13,499', 'CNY 14,499', 'CNY 1,250', 'CNY 599', 'CNY 549', 'CNY 2,599', 'CNY 699', 'CNY 499', '13,999', 'CNY 15,999', 'CNY 17,999\x0a'], dtype=object)

артефакты

- \xa0
- CNY

In [81]: df[df['price_china_cny'].astype(str).str.count(',') > 1].iloc[:, [0, 1] + list(range(15, 21))]

Out[81]: company_name model_name price_pakistan_pkp price_india_inr price_china_cny price_usa_usd price_dubai_aed launched_year

In [82]: df['price_china_cny'] = (
 df['price_china_cny']
 .astype(str)
 .str.replace('CNY', '', regex=False) # удаляем 'CNY'
 .str.replace(',', '', regex=False) # удаляем запятые
 .str.replace('\xa0', '', regex=False) # удаляем неразрывные пробелы
 .str.strip() # убираем обычные пробелы
 .astype(float) # преобразуем в число
)

Выводим отсортированные уникальные значения и тип данных
print("📦 Уникальные значения(отсортированные): \n", sorted(df['price_china_cny'].dropna().unique()))
print("🔍 Тип данных:", df['price_china_cny'].dtype)

📦 Уникальные значения(отсортированные):

[5999.0, 6999.0, 7499.0, 7999.0, 8499.0, 8999.0, 9499.0, 9990.0, 9999.0, 10499.0, 10990.0, 10999.0, 11499.0, 11990.0, 11999.0, 12499.0, 1299.0, 13490.0, 13499.0, 13990.0, 13999.0, 14990.0, 14999.0, 15999.0, 16990.0, 16999.0, 17499.0, 17999.0, 18000.0, 18499.0, 18990.0, 18999.0, 19000.0, 19499.0, 19990.0, 19999.0, 20000.0, 20999.0, 21400.0, 21990.0, 21999.0, 22990.0, 22999.0, 23999.0, 24990.0, 24999.0, 25999.0, 26990.0, 26999.0, 27990.0, 27999.0, 28756.0, 28999.0, 29900.0, 29999.0, 30999.0, 31990.0, 31999.0, 32990.0, 32999.0, 33990.0, 33999.0, 34999.0, 35990.0, 35999.0, 36999.0, 37999.0, 38999.0, 39900.0, 39990.0, 39999.0, 41999.0, 42999.0, 43999.0, 44999.0, 45999.0, 46999.0, 47999.0, 49900.0, 49990.0, 49999.0, 50999.0, 52999.0, 53999.0, 54900.0, 54999.0, 58590.0, 58999.0, 59999.0, 64900.0, 64999.0, 65999.0, 67999.0, 69900.0, 69999.0, 71900.0, 72999.0, 74900.0, 74999.0, 77999.0, 79900.0, 79999.0, 84900.0, 84999.0, 85999.0, 89900.0, 89999.0, 94900.0, 94999.0, 99900.0, 99999.0, 104900.0, 104999.0, 109900.0, 109999.0, 114900.0, 114999.0, 119900.0, 119999.0, 124999.0, 129900.0, 134900.0, 139900.0, 139999.0, 144999.0, 149900.0, 149999.0, 154900.0, 159900.0, 159999.0, 164999.0, 169999.0, 174900.0, 176999.0, 179900.0, 17999.0, 199900.0, 199999.0, 200999.0, 249999.0, 259999.0, 274999.0]

🔍 Тип данных: float64

price_dubai_aed

In [83]: df['price_dubai_aed'].unique()

```
Out[83]: array(['AED 2,799', 'AED 2,999', 'AED 3,199', 'AED 3,399', 'AED 3,599',  
               'AED 3,499', 'AED 3,699', 'AED 3,899', 'AED 3,799', 'AED 3,999',  
               'AED 4,199', 'AED 2,699', 'AED 4,399', 'AED 4,499', 'AED 2,599',  
               'AED 3,299', 'AED 4,599', 'AED 2,499', 'AED 4,999', 'AED 4,299',  
               'AED 4,799', 'AED 5,099', 'AED 3,099', 'AED 2,099', 'AED 1,199',  
               'AED 1,499', 'AED 1,799', 'AED 5,299', 'AED 5,799', 'AED 6,099',  
               'AED 4,899', 'AED 6,999', 'AED 7,499', 'AED 4,099', 'AED 6,499',  
               'AED 7,099', 'AED 1,399', 'AED 1,299', 'AED 1,099', 'AED 899',  
               'AED 999', 'AED 1,699', 'AED 799', 'AED 699', 'AED 499',  
               'AED 5,499', 'AED 1,999', 'AED 599', 'AED 2,199', 'AED 1,899',  
               'AED 399', 'AED 2,899', 'AED 2,299', 'AED 2,399', 'AED 4,699',  
               'AED 1,649', 'AED 549', 'AED 649', 'AED 1,599', 'AED 749',  
               'AED 5,199', 'AED 1,000', 'AED 1,200', 'AED 1,300', 'AED 1,100',  
               'AED 950', 'AED 900', 'AED 850', 'AED 800', 'AED 750', 'AED 700',  
               'AED 1,500', 'AED 1,700', 'AED 1,400', 'AED 1,725', 'AED 1,840',  
               'AED 1,460', 'AED 1,520', 'AED 1,330', 'AED 1,250', 'AED 1,320',  
               'AED 1,150', 'AED 1,210', 'AED 970', 'AED 1,030', 'AED 860',  
               'AED 1,070', 'AED 1,140', 'AED 1,390', 'AED 960', 'AED 1,280',  
               'AED 720', 'AED 780', 'AED 1,020', 'AED 920', 'AED 660', 'AED 600',  
               'AED 830', 'AED 500', 'AED 620', 'AED 680', 'AED 580', 'AED 820',  
               'AED 880', 'AED 1,620', 'AED 1,800', 'AED 740', 'AED 1,050',  
               'AED 770', 'AED 730', 'AED 849', 'AED 949', 'AED 1,049',  
               'AED 1,175', 'AED 1,350', 'AED 1,275', 'AED 1,675', 'AED 925',  
               'AED 1,850', 'AED 550', 'AED 650', 'AED 9,999', 'AED 8,999',  
               'AED 10,499', 'AED 11,099', 'AED 629', 'AED 619', 'AED 449',  
               'AED 5,999', 'AED 6,199', 'AED 870', 'AED 349', 'AED 299',  
               'AED 1,149', 'AED 1,249', 'AED 2,049', 'AED 6,599', 'AED 1,029',  
               'AED 7,199', 'AED 7,699', 'AED 8,699'], dtype=object)
```

```
In [84]: df[df['price_dubai_aed'].astype(str).str.count(',') > 1].iloc[:, [0, 1] + list(range(15, 21))]
```

```
Out[84]: company_name  model_name  price_pakistan_pkrs  price_india_inr  price_china_cny  price_usa_usd  price_dubai_aed  launched_year
```

```
In [85]: df['price_dubai_aed'] = (
    df['price_dubai_aed']
    .astype(str)
    .str.replace('AED', '', regex=False)      # удаляем 'CNY'
    .str.replace(',', '', regex=False)        # удаляем запятые
    .str.strip()                            # убираем обычные пробелы
    .astype(float)                          # преобразуем в число
)

# Выводим отсортированные уникальные значения и тип данных
print("📦 Уникальные значения(отсортированные): \n", sorted(df['price_dubai_aed'].dropna().unique()))
print("🔍 Тип данных:", df['price_dubai_aed'].dtype)
```

price_usa_usd

```
In [86]: df['price_usa_usd'].unique()
```

```
Out[86]: array(['USD 799', 'USD 849', 'USD 899', 'USD 949', 'USD 999', 'USD 1,049',
       'USD 1,099', 'USD 1,199', 'USD 1,299', 'USD 1,399', 'USD 699',
       'USD 1,249', 'USD 749', 'USD 599', 'USD 329', 'USD 429', 'USD 399',
       'USD 499', 'USD 1,599', 'USD 1,799', 'USD 1,899', 'USD 449',
       'USD 349', 'USD 299', 'USD 249', 'USD 199', 'USD 179', 'USD 169',
       'USD 129', 'USD 1,499', 'USD 549', 'USD 229', 'USD 149', 'USD 649',
       'USD 99', 'USD 219', 'USD 139', 'USD 159', 'USD 269', 'USD 319',
       'USD 239', 'USD 279', 'USD 189', 'USD 259', 'USD 470', 'USD 500',
       'USD 380', 'USD 400', 'USD 420', 'USD 360', 'USD 340', 'USD 320',
       'USD 270', 'USD 290', 'USD 250', 'USD 220', 'USD 240', 'USD 300',
       'USD 200', 'USD 260', 'USD 180', 'USD 160', 'USD 210', 'USD 230',
       'USD 130', 'USD 170', 'USD 190', 'USD 150', 'USD 440', 'USD 330',
       'USD 350', 'USD 280', 'USD 310', 'USD 634.99', 'USD 790.77',
       'USD 374.90', 'USD 399.00', 'USD 429.00', 'USD 349.00',
       'USD 379.00', 'USD 299.00', 'USD 329.00', 'USD 279.00',
       'USD 309.00', 'USD 249.00', 'USD 199.00', 'USD 229.00', 'USD 479',
       'USD 370', 'USD 450', 'USD 2,699', 'USD 2,499', 'USD 2,599',
       'USD 2,799', 'USD 529', 'USD 729', 'USD 119', 'USD 1,699',
       'USD 396,22', 'USD 877', 'USD 89', 'USD 289', 'USD 379', 'USD 109',
       'USD 79', 'USD 1719', 'USD 2,259], dtype=object)
```

```
In [87]: df[df['price usa usd'].astype(str).str.contains(r'\d{2},\,\d{2},\.', regex=True)].iloc[:, [0, 1] + list(range(15, 21))]
```

```
Out[87]:    company_name  model_name  price_pakistan_pk  price_india_inr  price_china_cny  price_usa_usd  price_dubai_aed  launched_year
0   Toyota        Corolla    520000.0       170000.0      10500.0        USD 306.00     670.0          2002
```

```
In [88]: df[df['ippcc_use']==1].astype(str).str.count('1') + 1).iloc[:, [0, 1]] + list(range(15, 21))
```

```
Out[88]: company_name model_name price_pakistan_pk  price_india_inr price_china_cny price_usa_usd price_dubai_aed launched_year
```

```
In [89]: df.loc[685, 'price_usa_usd'] = str(df.loc[685, 'price_usa_usd']).replace(',', '.')
df.iloc[685:686, [0, 1] + list(range(15, 21))]
```

```
Out[89]: company_name model_name price_pakistan_pk  price_india_inr price_china_cny price_usa_usd price_dubai_aed launched_year
685 Nokia T21 52000.0 17999.0 1250.0 USD 396.22 870.0 2022
```

```
In [90]: df['price_usa_usd'] = (
    df['price_usa_usd']
    .astype(str)
    .str.replace('USD', '', regex=False)      # удаляем 'CNY'
    .str.replace(',', '', regex=False)        # удаляем запятые
    .str.strip()                            # убираем обычные пробелы
    .astype(float)                          # преобразуем в число
)

# Выводим отсортированные уникальные значения и тип данных
print("📦 Уникальные значения(отсортированные): \n", sorted(df['price_usa_usd'].dropna().unique()))
print("🔍 Тип данных:", df['price_usa_usd'].dtype)
```

📦 Уникальные значения(отсортированные):
[79.0, 89.0, 99.0, 109.0, 119.0, 129.0, 130.0, 139.0, 149.0, 150.0, 159.0, 160.0, 169.0, 170.0, 179.0, 180.0, 189.0, 190.0, 199.0, 200.0, 210.0, 219.0, 220.0, 229.0, 230.0, 239.0, 240.0, 249.0, 250.0, 259.0, 260.0, 269.0, 270.0, 279.0, 280.0, 289.0, 290.0, 299.0, 300.0, 309.0, 310.0, 319.0, 320.0, 329.0, 330.0, 340.0, 349.0, 350.0, 360.0, 370.0, 374.9, 379.0, 380.0, 396.22, 399.0, 400.0, 420.0, 429.0, 440.0, 449.0, 450.0, 470.0, 479.0, 499.0, 500.0, 529.0, 549.0, 599.0, 634.99, 649.0, 699.0, 729.0, 749.0, 790.77, 799.0, 849.0, 877.0, 899.0, 949.0, 999.0, 1049.0, 1099.0, 1199.0, 1249.0, 1299.0, 1399.0, 1499.0, 1599.0, 1699.0, 1719.0, 1799.0, 1899.0, 2259.0, 2499.0, 2599.0, 2699.0, 2799.0]
🔍 Тип данных: float64

Замена пропуска в колонке 'price_pakistan_pk' с помощью линейно регрессии

можно было бы удалить эту строчку, но интереснее ее восстановить из регрессии

- в качестве модели используем градиентный бустинг

```
In [91]: df[['price_pakistan_pk', 'price_india_inr', 'price_china_cny', 'price_dubai_aed', 'price_usa_usd']].corr()
```

```
Out[91]: price_pakistan_pk price_india_inr price_china_cny price_dubai_aed price_usa_usd
price_pakistan_pk 1.000000 0.903963 0.902440 0.898227 0.897105
price_india_inr 0.903963 1.000000 0.966664 0.969596 0.969655
price_china_cny 0.902440 0.966664 1.000000 0.969367 0.967804
price_dubai_aed 0.898227 0.969596 0.969367 1.000000 0.990440
price_usa_usd 0.897105 0.969655 0.967804 0.990440 1.000000
```

```
In [92]: # 🚧 Подготовка данных
features = ['price_india_inr'] # ✅ только одна фича
target = 'price_pakistan_pk'

# ✎ Очистка: преобразуем в числовой формат
df[features + [target]] = df[features + [target]].apply(pd.to_numeric, errors='coerce')
df_clean = df.dropna(subset=features + [target])

# 📈 Разделение на X и y
X = df_clean[features]
y = df_clean[target]

# ✋ Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 🚧 Pipeline: масштабирование + бустинг
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', GradientBoostingRegressor(random_state=42))
])

# 🔎 GridSearchCV: подбор параметров
param_grid = {
    'regressor__n_estimators': [100, 200],
    'regressor__learning_rate': [0.05, 0.1, 0.2],
    'regressor__max_depth': [3, 4, 5]
}

grid = GridSearchCV(
    pipeline,
    param_grid,
    cv=KFold(n_splits=5, shuffle=True, random_state=42),
    scoring='r2',
    n_jobs=-1
)

# 💡 Обучение модели
```

```

grid.fit(X_train, y_train)

# 📈 Лучшие параметры
print("✅ Лучшая модель и параметры:")
print(grid.best_params_)

# 🖋 Предсказание
y_pred = grid.predict(X_test)

# 📈 Метрики
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("\n📊 Оценка на тестовой выборке:")
print(f"- R²: {r2_score(y_test, y_pred):.4f}")
print(f"- MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(f"- RMSE: {rmse:.2f}")

# 📈 Визуализация: фактические vs предсказанные
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Фактические значения")
plt.ylabel("Предсказанные значения")
plt.title("Gradient Boosting: предсказание цены в PKR по индийской цене")
plt.grid(True)
plt.tight_layout()
plt.show()

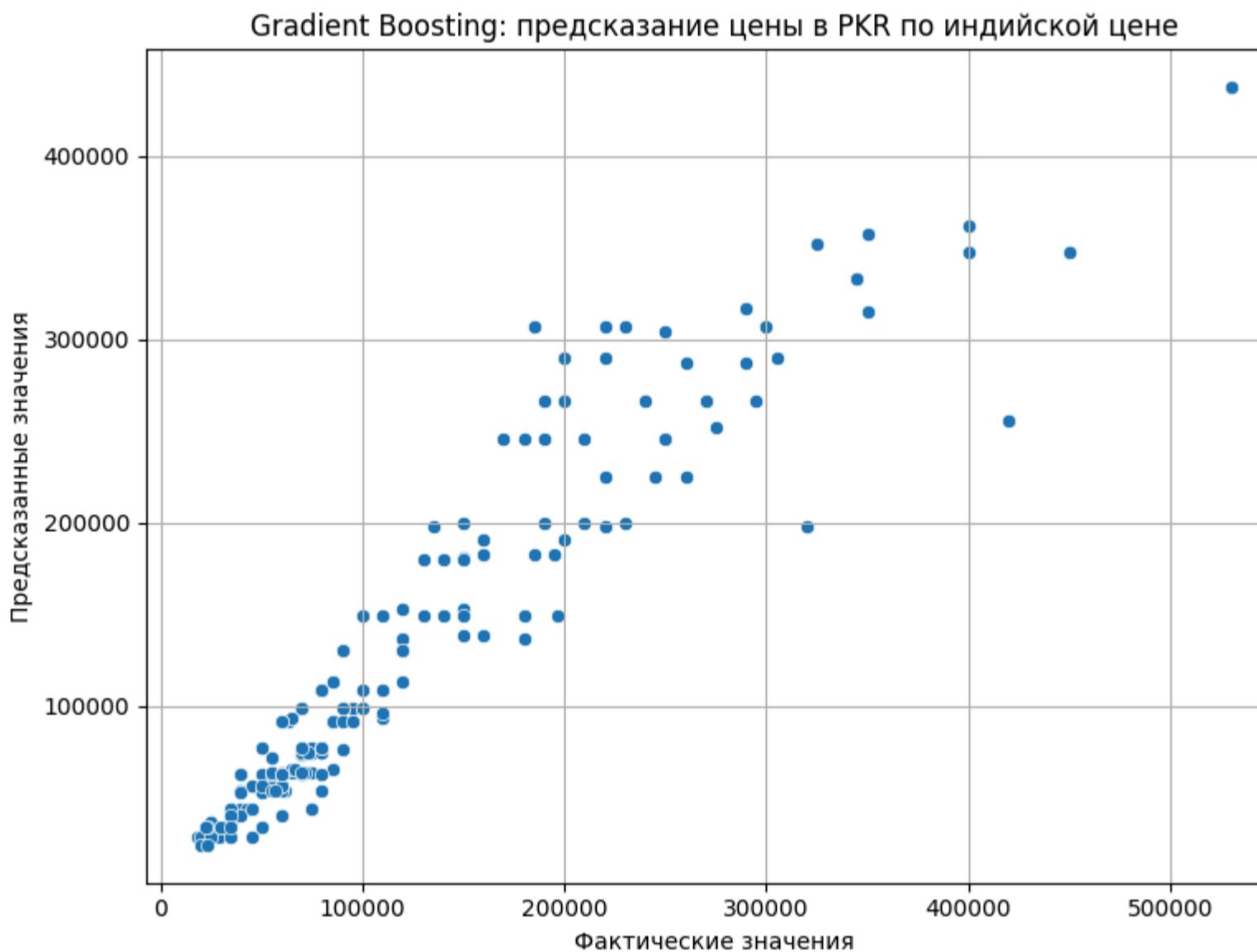
```

✓ Лучшая модель и параметры:

```
{'regressor__learning_rate': 0.05, 'regressor__max_depth': 4, 'regressor__n_estimators': 100}
```

📊 Оценка на тестовой выборке:

- R²: 0.8859
- MAE: 20420.73
- RMSE: 33017.41



In [93]: `df[df['price_pakistan_pk'] .isna()].iloc[:, [0, 1] + list(range(15, 21))]`

Out[93]:

	company_name	model_name	price_pakistan_pk	price_india_inr	price_china_cny	price_usa_usd	price_dubai_aed	launched_year
929	Samsung	Galaxy Z Fold6 1TB	NaN	200999.0	17999.0	2259.0	8699.0	2024

✓ Лучшие параметры модели: {'regressor': Ridge(), 'regressor_alpha': 26.366508987303554}

📊 Оценка на тестовой выборке:

- R²: 0.8288
- MAE: 23588.99
- RMSE: 40446.65

```

best_model = grid.best_estimator_.named_steps['regressor']
coefficients = pd.Series(best_model.coef_, index=features)
coefficients.sort_values().plot(kind='barh', title=" Влияние признаков на цену в PKR")
plt.xlabel("Коэффициент")
plt.tight_layout()
plt.show()

```

In [94]: `df[df['price_pakistan_pk'] .isna()].iloc[:, [0, 1] + list(range(15, 21))]`

```
Out[94]:    company_name      model_name  price_pakistan_pk  price_india_inr  price_china_cny  price_usa_usd  price_dubai_aed  launched_year
929      Samsung   Galaxy Z Fold6 1TB          NaN        200999.0       17999.0        2259.0        8699.0        2024
```

```
In [95]: # 🌟 Восстановление пропусков в price_pakistan_pk
missing_mask = df['price_pakistan_pk'].isna()
X_missing = df.loc[missing_mask, features].apply(pd.to_numeric, errors='coerce')

# 💡 Предсказание пропущенных значений
predicted_missing = grid.best_estimator_.predict(X_missing)

# 📈 Округление до целых чисел
predicted_missing_rounded = np.round(predicted_missing).astype(int)

# ✨ Подстановка предсказанных значений
df.loc[missing_mask, 'price_pakistan_pk'] = predicted_missing_rounded
print(f"\n🌟 Восстановлено {missing.sum()} пропусков в price_pakistan_pk.")

# 📋 Проверка результата для конкретной модели
df[df['model_name'] == 'Galaxy Z Fold6 1TB'].iloc[:, [0, 1] + list(range(15, 21))]
```

🌟 Восстановлено 1 пропусков в price_pakistan_pk.

```
Out[95]:    company_name      model_name  price_pakistan_pk  price_india_inr  price_china_cny  price_usa_usd  price_dubai_aed  launched_year
929      Samsung   Galaxy Z Fold6 1TB        347178.0       200999.0       17999.0        2259.0        8699.0        2024
```

Обработка пропусков

```
In [96]: def analyze_missing_values(df):
    temp = df.copy() # 📝 Создаем копию входного DataFrame
    missing = pd.DataFrame({
        'Кол-во пропусков': temp.isnull().sum(),
        'Доля пропусков': temp.isnull().mean().round(4)
    })
    # 🎯 Сортируем столбцы по количеству пропусков (по убыванию)
    missing = missing.sort_values(by='Кол-во пропусков', ascending=False)

    # 🎨 Визуализация пропусков
    return missing.style.background_gradient(cmap='coolwarm')
```

```
In [97]: analyze_missing_values(df)
```

```
Out[97]:          Кол-во пропусков  Доля пропусков
front_camera_2_mp                  920     0.987100
back_camera_4                      912     0.978500
back_camera_3                      776     0.832600
back_camera_2                      515     0.552600
processor                           0     0.000000
price_dubai_aed                     0     0.000000
price_usa_usd                      0     0.000000
price_china_cny                     0     0.000000
price_india_inr                     0     0.000000
price_pakistan_pk                  0     0.000000
screen_size_inches                  0     0.000000
battery_capacity_mah                0     0.000000
company_name                         0     0.000000
back_camera_number                  0     0.000000
model_name                           0     0.000000
back_camera_1                        0     0.000000
front_camera_number                 0     0.000000
front_camera_1_mp                   0     0.000000
ram_gb                               0     0.000000
mobile_weight_g                      0     0.000000
launched_year                         0     0.000000
```

все эти пропуска связаны с камерами, посмотрим на уникальные значения

```
In [98]: def inspect_object_columns(df, df_name='DataFrame'):
    temp = df.copy() # 📝 Создаем копию входного DataFrame
    object_cols = temp.select_dtypes(include=['number']).columns

    print(f"\n📋 Информация о object-столбцах в {df_name}:")
```

```
print(temp[object_cols].info())

for col in object_cols:
    print('-' * 40)
    print(f"♦ {col} - уникальные значения:")
    print(temp[col].sort_values().unique())
```

In [99]: inspect_object_columns(df, 'df')

Информация о объектах в df:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 932 entries, 0 to 931
Data columns (total 18 columns):
 # Column Non-Null Count Dtype
--- ---
 0 mobile_weight_g 932 non-null float64
 1 ram_gb 932 non-null float64
 2 front_camera_1_mp 932 non-null float64
 3 front_camera_2_mp 12 non-null float64
 4 front_camera_number 932 non-null int64
 5 back_camera_1 932 non-null float64
 6 back_camera_2 417 non-null float64
 7 back_camera_3 156 non-null float64
 8 back_camera_4 20 non-null float64
 9 back_camera_number 932 non-null int64
 10 battery_capacity_mah 932 non-null int32
 11 screen_size_inches 932 non-null float64
 12 price_pakistan_pk 932 non-null float64
 13 price_india_inr 932 non-null float64
 14 price_china_cny 932 non-null float64
 15 price_usa_usd 932 non-null float64
 16 price_dubai_aed 932 non-null float64
 17 launched_year 932 non-null int64
 dtypes: float64(14), int32(1), int64(3)
 memory usage: 127.5 KB
None

◆ mobile_weight_g – уникальные значения:
[135. 140. 143. 146. 147. 150. 151. 153. 155. 156. 158. 159.
 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172.
 173. 174. 175. 176. 177. 178. 178.8 179. 180. 181. 182. 183.
 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195.
 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207.
 208. 209. 210. 211. 212. 213. 215. 216. 218. 219. 220. 221.
 222. 222.8 223. 225. 226. 227. 228. 229. 230. 231. 233. 234.
 235. 236. 238. 239. 240. 241. 242. 245. 250. 254. 255. 263.
 267. 280. 288. 295. 300.5 360. 366. 372. 420. 426. 433. 440.
 450. 458. 460. 465. 466. 468. 470. 475. 480. 482. 485. 490.
 495. 498. 499. 500. 503. 505. 508. 510. 520. 523. 530. 533.
 535. 540. 550. 555. 560. 567. 571. 580. 586. 590. 610. 674.
 682. 708. 726. 732.]

◆ ram_gb – уникальные значения:
[1. 1.5 2. 3. 4. 6. 8. 10. 12. 16.]

◆ front_camera_1_mp – уникальные значения:
[2. 5. 7. 8. 10. 10.5 10.7 10.8 11.1 12. 13. 16. 20. 24.
 25. 32. 42. 44. 48. 50. 60.]

◆ front_camera_2_mp – уникальные значения:
[4. 8. 12. nan]

◆ front_camera_number – уникальные значения:
[1 2]

◆ back_camera_1 – уникальные значения:
[5. 8. 12. 12.2 13. 16. 20. 40. 48. 50. 54. 64.
 100. 108. 160. 200.]

◆ back_camera_2 – уникальные значения:
[2. 5. 6. 8. 10. 12. 12.5 13. 16. 20. 32. 40. 48. 50.
 64. nan]

◆ back_camera_3 – уникальные значения:
[2. 8. 12. 13. 16. 32. 40. 48. 50. 64. nan]

◆ back_camera_4 – уникальные значения:
[2. nan]

◆ back_camera_number – уникальные значения:
[1 2 3 4]

◆ battery_capacity_mah – уникальные значения:
[2000 2227 2300 2438 2600 2658 2716 2800 2815 2942 3000 3046
 3055 3095 3110 3140 3174 3200 3240 3260 3300 3315 3350 3500
 3600 3687 3700 3750 3800 3885 3900 3969 4000 4020 4025 4030
 4040 4050 4080 4085 4100 4115 4200 4300 4310 4325 4350 4352
 4355 4360 4385 4400 4410 4450 4460 4500 4510 4520 4575 4600
 4610 4614 4680 4700 4750 4800 4805 4815 4830 4870 4900 5000
 5003 5050 5065 5100 5110 5124 5160 5200 5250 5300 5400 5450
 5500 5550 5600 5700 5800 6000 6100 6400 6500 7000 7040 7100
 7200 7250 7500 7600 7608 7812 8000 8040 8200 8300 8340 8360
 8400 8500 8612 8850 9000 9510 9720 10000 10090 10100 10307 10500
 11000 11200]

◆ screen_size_inches – уникальные значения:
[5. 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.88 6. 6.01 6.09
 6.1 6.2 6.22 6.28 6.3 6.31 6.34 6.35 6.36 6.38 6.39 6.4
 6.41 6.43 6.44 6.47 6.49 6.5 6.51 6.52 6.53 6.55 6.56 6.57
 6.58 6.59 6.6 6.63 6.67 6.7 6.71 6.72 6.73 6.74 6.76 6.78
 6.79 6.8 6.81 6.82 6.83 6.85 6.9 6.95 7.09 7.1 7.6 7.8
 7.82 7.85 7.9 7.92 7.93 8. 8.7 9.7 10.1 10.2 10.4 10.5]

10.9 11. 11.5 11.6 11.61 12. 12.1 12.2 12.3 12.4 12.6 12.9
13. 13.2 13.5 14.6]

◆ price_pakistan_pk - уникальные значения:

[15999. 18499. 18999. 19999. 21999. 22499. 22999. 23999. 24999.
25000. 25999. 26999. 27999. 28000. 28999. 29999. 30999. 31999.
32999. 34999. 35000. 35999. 36999. 37999. 38999. 39999. 40000.
41999. 42999. 44999. 45000. 45999. 46999. 47999. 48999. 49999.
50000. 52000. 52999. 54999. 55000. 55999. 56999. 57999. 58999.
59999. 60000. 61999. 62999. 64999. 65000. 66220. 68000. 69999.
70000. 71220. 72999. 74999. 75000. 79999. 80000. 84999. 85000.
85999. 89999. 94999. 95000. 99999. 100000. 104999. 105000. 109999.
119999. 120000. 124999. 129999. 134999. 139999. 140000. 149999. 150000.
159999. 160000. 161500. 164999. 169999. 170000. 174999. 179999. 180000.
184999. 189999. 194999. 197000. 199999. 200000. 204999. 209999. 214999.
219999. 224999. 229999. 230000. 234999. 239999. 244999. 249999. 250000.
259999. 260000. 264999. 269999. 274999. 279999. 280000. 284999. 289999.
294999. 299999. 300000. 304999. 309999. 314999. 319999. 320000. 324999.
330000. 334999. 339999. 344999. 347178. 349999. 350000. 354999. 359999.
360000. 364999. 369999. 380000. 384999. 389999. 390000. 399999. 400000.
420000. 429999. 430000. 449999. 450000. 460000. 469999. 480000. 500000.
530000. 544999. 550000. 604999.]

◆ price_india_inr - уникальные значения:

[599. 699. 749. 799. 849. 899. 949. 999. 9990. 999.
10499. 10990. 10999. 11499. 11990. 11999. 12499. 12999. 13490.
13499. 13990. 13999. 14990. 14999. 15999. 16990. 16999. 17499.
17999. 18000. 18499. 18990. 18999. 19000. 19499. 19990. 19999.
20000. 20999. 21400. 21990. 21999. 22990. 22999. 23999. 24990.
24999. 25999. 26990. 26999. 27990. 27999. 28756. 28999. 29900.
29999. 30999. 31990. 31999. 32990. 32999. 33990. 33999. 34999.
35990. 35999. 36999. 37999. 38999. 39900. 39990. 39999. 41999.
42999. 43999. 44999. 45999. 46999. 47999. 49900. 49990. 49999.
50999. 52999. 53999. 54900. 54999. 58590. 58999. 59999. 64900.
64999. 65999. 67999. 69900. 69999. 71900. 72999. 74900. 74999.
77999. 79900. 79999. 84900. 84999. 85999. 89900. 89999. 94900.
94999. 99900. 99999. 104900. 104999. 109900. 109999. 114900. 114999.
119900. 119999. 124999. 129900. 129999. 134900. 134999. 139900. 139999.
144999. 149900. 149999. 154900. 159900. 159999. 164999. 169999. 174900.
176999. 179900. 179999. 199900. 199999. 200999. 249999. 259999. 274999.]

◆ price_china_cny - уникальные значения:

[499. 549. 599. 699. 799. 899. 900. 999. 1050. 1099.
1100. 1199. 1200. 1250. 1299. 1300. 1350. 1399. 1400. 1499.
1500. 1550. 1599. 1600. 1699. 1700. 1799. 1800. 1899. 1900.
1998. 1999. 2000. 2099. 2100. 2199. 2200. 2299. 2300. 2399.
2400. 2499. 2500. 2599. 2600. 2699. 2700. 2799. 2800. 2899.
2900. 2999. 3000. 3099. 3199. 3200. 3299. 3399. 3400. 3499.
3599. 3600. 3699. 3799. 3899. 3999. 4088. 4199. 4299. 4488.
4499. 4599. 4699. 4799. 4999. 5199. 5299. 5499. 5599. 5699.
5799. 5899. 5988. 5999. 6099. 6199. 6299. 6399. 6499. 6599.
6799. 6988. 6999. 7099. 7199. 7299. 7399. 7499. 7699. 7799.
7988. 7999. 8099. 8199. 8299. 8388. 8399. 8499. 8699. 8788.
8799. 8988. 8999. 9188. 9199. 9499. 9788. 9799. 9988. 9999.
10088. 10388. 10399. 10688. 10999. 11199. 11288. 11999. 12999. 13499.
13999. 14499. 14999. 15999. 17999.]

◆ price_usa_usd - уникальные значения:

[79. 89. 99. 109. 119. 129. 130. 139. 149.
150. 159. 160. 169. 170. 179. 180. 189. 190.
199. 200. 210. 219. 220. 229. 230. 239. 240.
249. 250. 259. 260. 269. 270. 279. 280. 289.
290. 299. 300. 309. 310. 319. 320. 329. 330.
340. 349. 350. 360. 370. 374.9 379. 380. 396.22
399. 400. 420. 429. 440. 449. 450. 470. 479.
499. 500. 529. 549. 599. 634.99 649. 699. 729.
749. 790.77 799. 849. 877. 899. 949. 999. 1049.
1099. 1199. 1249. 1299. 1399. 1499. 1599. 1699. 1719.
1799. 1899. 2259. 2499. 2599. 2699. 2799.]

◆ price_dubai_aed - уникальные значения:

[299. 349. 399. 449. 499. 500. 549. 550. 580. 599.
600. 619. 620. 629. 649. 650. 660. 680. 699. 700.
720. 730. 740. 749. 750. 770. 780. 799. 800. 820.
830. 849. 850. 860. 870. 880. 899. 900. 920. 925.
949. 950. 960. 970. 999. 1000. 1020. 1029. 1030. 1049.
1050. 1070. 1099. 1100. 1140. 1149. 1150. 1175. 1199. 1200.
1210. 1249. 1250. 1275. 1280. 1299. 1300. 1320. 1330. 1350.
1390. 1399. 1400. 1460. 1499. 1500. 1520. 1599. 1620. 1649.
1675. 1699. 1700. 1725. 1799. 1800. 1840. 1850. 1899. 1999.
2049. 2099. 2199. 2299. 2399. 2499. 2599. 2699. 2799. 2899.
2999. 3099. 3199. 3299. 3399. 3499. 3599. 3699. 3799. 3899.
3999. 4099. 4199. 4299. 4399. 4499. 4599. 4699. 4799. 4899.
4999. 5099. 5199. 5299. 5499. 5799. 5999. 6099. 6199. 6499.
6599. 6999. 7099. 7199. 7499. 7699. 8699. 8999. 9999. 10499.
11099.]

◆ launched_year - уникальные значения:

[2014 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025]

Вариант: заменить нап на -1, типа заглушка (камера отсутсвует)

In [100...]

```
camera_cols = [
    'front_camera_2_mp',
    'back_camera_2',
    'back_camera_3',
    'back_camera_4'
]

df[camera_cols] = df[camera_cols].fillna(-1.0)
```

In [101...]

```
analyze_missing_values(df)
```

Out[101...]

	Кол-во пропусков	Доля пропусков
company_name	0	0.000000
back_camera_number	0	0.000000
price_dubai_aed	0	0.000000
price_usa_usd	0	0.000000
price_china_cny	0	0.000000
price_india_inr	0	0.000000
price_pakistan_pkrs	0	0.000000
screen_size_inches	0	0.000000
battery_capacity_mah	0	0.000000
processor	0	0.000000
back_camera_4	0	0.000000
model_name	0	0.000000
back_camera_3	0	0.000000
back_camera_2	0	0.000000
back_camera_1	0	0.000000
front_camera_number	0	0.000000
front_camera_2_mp	0	0.000000
front_camera_1_mp	0	0.000000
ram_gb	0	0.000000
mobile_weight_g	0	0.000000
launched_year	0	0.000000

Поиск дубликатов

In [102...]

```
# 📈 Количество дублирующих строк
num_duplicates = df.duplicated().sum()
print(f"📋 Количество дублирующих строк: {num_duplicates}")

# 📋 Вывод самих дубликатов
if num_duplicates > 0:
    duplicates = df[df.duplicated(keep=False)]
    print("\n📋 Дублирующие строки:")
    display(duplicates)
else:
    print("✅ Дубликаты не найдены.")
```

📋 Количество дублирующих строк: 15

📋 Дублирующие строки:

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_1_mp	front_camera_2_mp	front_camera_number	back_camera_1	back_camera_2	back_camera_number
344	Oppo	A3 128GB	186.0	4.0	5.0	-1.0	1	50.0	-1.	
347	Oppo	A3 128GB	186.0	4.0	5.0	-1.0	1	50.0	-1.	
423	Oppo	K11x 128GB	195.0	8.0	16.0	-1.0	1	108.0	2.	
424	Oppo	K11x 256GB	195.0	12.0	16.0	-1.0	1	108.0	2.	
425	Oppo	K10x 128GB	195.0	6.0	16.0	-1.0	1	64.0	2.	
426	Oppo	K10x 256GB	195.0	8.0	16.0	-1.0	1	64.0	2.	
427	Oppo	K10 5G 128GB	205.0	8.0	16.0	-1.0	1	64.0	8.	
428	Oppo	K9x 128GB	194.0	6.0	16.0	-1.0	1	64.0	2.	
429	Oppo	K9x 256GB	194.0	8.0	16.0	-1.0	1	64.0	2.	
430	Oppo	K9 Pro 5G 128GB	180.0	8.0	16.0	-1.0	1	64.0	8.	
431	Oppo	K9 Pro 5G 256GB	180.0	12.0	16.0	-1.0	1	64.0	8.	
432	Oppo	K9 5G 128GB	172.0	8.0	32.0	-1.0	1	64.0	8.	
433	Oppo	K9 5G 256GB	172.0	8.0	32.0	-1.0	1	64.0	8.	
434	Oppo	K7x 128GB	194.0	6.0	16.0	-1.0	1	48.0	8.	
435	Oppo	K7 5G 128GB	180.0	8.0	32.0	-1.0	1	48.0	8.	
437	Oppo	K11x 128GB	195.0	8.0	16.0	-1.0	1	108.0	2.	
438	Oppo	K11x 256GB	195.0	12.0	16.0	-1.0	1	108.0	2.	
441	Oppo	K10x 128GB	195.0	6.0	16.0	-1.0	1	64.0	2.	
442	Oppo	K10x 256GB	195.0	8.0	16.0	-1.0	1	64.0	2.	
443	Oppo	K10 5G 128GB	205.0	8.0	16.0	-1.0	1	64.0	8.	
445	Oppo	K9x 128GB	194.0	6.0	16.0	-1.0	1	64.0	2.	
446	Oppo	K9x 256GB	194.0	8.0	16.0	-1.0	1	64.0	2.	
447	Oppo	K9 Pro 5G 128GB	180.0	8.0	16.0	-1.0	1	64.0	8.	
448	Oppo	K9 Pro 5G 256GB	180.0	12.0	16.0	-1.0	1	64.0	8.	
449	Oppo	K9 5G 128GB	172.0	8.0	32.0	-1.0	1	64.0	8.	

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_1_mp	front_camera_2_mp	front_camera_number	back_camera_1	back_camera_2
450	Oppo	K9 5G 256GB	172.0	8.0	32.0	-1.0	1	64.0	8.
451	Oppo	K7x 128GB	194.0	6.0	16.0	-1.0	1	48.0	8.
452	Oppo	K7 5G 128GB	180.0	8.0	32.0	-1.0	1	48.0	8.
791	Infinix	Hot 10 Lite 64GB	195.0	3.0	8.0	-1.0	1	13.0	-1.
798	Infinix	Hot 10 Lite 64GB	195.0	3.0	8.0	-1.0	1	13.0	-1.

30 rows × 21 columns

```
In [103...]: df = df.drop_duplicates().reset_index(drop=True)
df.duplicated().sum()
```

```
Out[103...]: 0
```

- как вариант проверить неявные дубликаты по 2 основным столбцам

```
In [104...]: # 🔎 Получаем первые два столбца
subset_cols = df.columns[:2]

# 📈 Находим дубликаты по этим столбцам
duplicates = df[df.duplicated(subset=subset_cols, keep=False)]

# 📈 Количество таких дубликатов
print(f"⚠️ Найдено {len(duplicates)} неявных дубликатов по столбцам: {list(subset_cols)}")

# 📈 Сортируем и выводим дубликаты
sorted_duplicates = duplicates.sort_values(by=list(subset_cols))
display(sorted_duplicates.iloc[:, :8])
```

⚠️ Найдено 15 неявных дубликатов по столбцам: ['company_name', 'model_name']

	company_name	model_name	mobile_weight_g	ram_gb	front_camera_1_mp	front_camera_2_mp	front_camera_number	back_camera_1
86	Apple	iPad Pro 128GB	468.0	4.0	7.0	-1.0	1	12.0
89	Apple	iPad Pro 128GB	682.0	6.0	7.0	-1.0	1	12.0
92	Apple	iPad Pro 128GB	708.0	6.0	7.0	-1.0	1	12.0
87	Apple	iPad Pro 256GB	468.0	4.0	7.0	-1.0	1	12.0
90	Apple	iPad Pro 256GB	682.0	6.0	7.0	-1.0	1	12.0
93	Apple	iPad Pro 256GB	708.0	6.0	7.0	-1.0	1	12.0
88	Apple	iPad Pro 512GB	468.0	6.0	7.0	-1.0	1	12.0
91	Apple	iPad Pro 512GB	682.0	6.0	7.0	-1.0	1	12.0
94	Apple	iPad Pro 512GB	708.0	6.0	7.0	-1.0	1	12.0
628	Huawei	P60 Art	206.0	8.0	13.0	-1.0	1	48.0
916	Huawei	P60 Art	206.0	12.0	13.0	-1.0	1	48.0
627	Huawei	P60 Pro	200.0	8.0	13.0	-1.0	1	48.0
915	Huawei	P60 Pro	200.0	12.0	13.0	-1.0	1	48.0
462	Realme	P2 Pro 5G 256GB	188.0	12.0	32.0	-1.0	1	50.0
493	Realme	P2 Pro 5G 256GB	195.0	12.0	16.0	-1.0	1	50.0

- как и ожидалось да названия могут совпадать но дальше идут различия в характеристиках

Пересчёт цен телефонов в одну валюту USD

```
In [105...]: symbols = {
    'PKR': 'PKR=X',
    'INR': 'INR=X',
    'CNY': 'CNY=X',
    'AED': 'AED=X'
}

currency_transfer = {}
for code, ticker in symbols.items():
    data = yf.Ticker(ticker).history(period="1d")
```

```

    currency_transfer[code] = data['Close'].iloc[-1]

currency_transfer_usd = {
    code: round(1 / rate, 6) for code, rate in currency_transfer.items()
}
print("✓ Курс пересчёта в USD:")
print(currency_transfer_usd)

✓ Курс пересчёта в USD:
{'PKR': 0.003563, 'INR': 0.011298, 'CNY': 0.140341, 'AED': 0.272264}

```

In [106...]

```

df['price_pakistan_usd'] = (df['price_pakistan_pk'] * currency_transfer_usd['PKR']).round(0)
df['price_india_usd'] = (df['price_india_inr'] * currency_transfer_usd['INR']).round(0)
df['price_china_usd'] = (df['price_china_cny'] * currency_transfer_usd['CNY']).round(0)
df['price_dubai_usd'] = (df['price_dubai_aed'] * currency_transfer_usd['AED']).round(0)

```

In [107...]

```

# 🔍 Проверка адекватности конвертации
price_cols = ['price_pakistan_usd', 'price_india_usd', 'price_china_usd',
              'price_dubai_usd', 'price_usa_usd']

# Корреляция между ценами в разных странах
price_correlation = df[price_cols].corr()

print("📊 Корреляция цен по странам (должна быть высокой >0.85):")
price_correlation

```

📊 Корреляция цен по странам (должна быть высокой >0.85):

Out[107...]

	price_pakistan_usd	price_india_usd	price_china_usd	price_dubai_usd	price_usa_usd
price_pakistan_usd	1.000000	0.903016	0.898678	0.896709	0.895764
price_india_usd	0.903016	1.000000	0.966370	0.969540	0.969374
price_china_usd	0.898678	0.966370	1.000000	0.969262	0.967535
price_dubai_usd	0.896709	0.969540	0.969262	1.000000	0.990679
price_usa_usd	0.895764	0.969374	0.967535	0.990679	1.000000

Добавление среднimesячных зарплат по странам

In [108...]

```

# 🚀 Загружаем таблицу с сайта
url = "https://www.worlddata.info/average-income.php"
df_income = pd.read_html(url)[0]

# 📊 Переименовываем и очищаем
df_income = df_income[['Country/Region', 'Ø Annual income']]
df_income.columns = ['Country', 'Annual_income']
df_income['Country'] = df_income['Country'].str.strip()
df_income['Annual_income'] = df_income['Annual_income'].replace('[$,]', '', regex=True).astype(float)
df_income.set_index('Country', inplace=True)

# 💼 Функция извлечения месячного дохода
def get_monthly_income(df, countries):
    return {
        code: round(df.loc[label]['Annual_income'] / 12, 2)
        for code, label in countries.items()
        if label in df.index
    }

# 🏠 Страны и их коды
country_map = {
    'PKR': 'Pakistan',
    'INR': 'India',
    'CNY': 'China',
    'AED': 'United Arab Emirates',
    'USD': 'United States'
}

# 📈 Получаем месячные доходы
monthly_income_usd = get_monthly_income(df_income, country_map)

print("✓ Месячный доход (USD/мес):")
print(monthly_income_usd)

```

✓ Месячный доход (USD/мес):
{'PKR': 119.17, 'INR': 220.83, 'CNY': 1138.33, 'AED': 4125.0, 'USD': 6971.67}

- зп получились сильно занижены если сравнивать с другими источниками

📊 Медианные зарплаты по странам (USD/мес)

Страна	Медианная зарплата (USD/мес)	Источник
PK Pakistan	\$296–\$320	RemotePeople
IN India	\$320–\$350	Time Doctor
CN China	\$1385–\$3041	MS Advisory
AE UAE	\$3757–\$5445	Expatica

Страна	Медианная зарплата (USD/мес)	Источник
us USA	\$5600–\$6970	DemandSage

In [109...]

```
# 📊 Словарь медианных зарплат

median_income_usd = {
    'pakistan': 308.0, # Pakistan: среднее между $296 и $320
    'india': 335.0, # India: среднее между $320 и $350
    'china': 2213.0, # China: среднее между $1385 и $3041
    'uae': 4601.0, # UAE: среднее между $3757 и $5445
    'usa': 6285.0 # USA: среднее между $5600 и $6970
}

# 📈 Добавляем каждый столбец в df
for code, value in median_income_usd.items():
    df[f'Median_income_usd_{code.lower()}'] = value
```

Создание новых признаков

- для начала удалим лишние столбцы

In [110...]

```
df.drop(columns=[
    'price_pakistan_pkkr',
    'price_india_inr',
    'price_china_cny',
    'price_dubai_aed'
], inplace=True)
```

In [111...]

```
inspect_object_columns(df, 'df')
```

Информация о объектах в df:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 917 entries, 0 to 916
Data columns (total 23 columns):

#	Column	Non-Null Count	Dtype
0	mobile_weight_g	917 non-null	float64
1	ram_gb	917 non-null	float64
2	front_camera_1_mp	917 non-null	float64
3	front_camera_2_mp	917 non-null	float64
4	front_camera_number	917 non-null	int64
5	back_camera_1	917 non-null	float64
6	back_camera_2	917 non-null	float64
7	back_camera_3	917 non-null	float64
8	back_camera_4	917 non-null	float64
9	back_camera_number	917 non-null	int64
10	battery_capacity_mah	917 non-null	int32
11	screen_size_inches	917 non-null	float64
12	price_usa_usd	917 non-null	float64
13	launched_year	917 non-null	int64
14	price_pakistan_usd	917 non-null	float64
15	price_india_usd	917 non-null	float64
16	price_china_usd	917 non-null	float64
17	price_dubai_usd	917 non-null	float64
18	Median_income_usd_pakistan	917 non-null	float64
19	Median_income_usd_india	917 non-null	float64
20	Median_income_usd_china	917 non-null	float64
21	Median_income_usd_uae	917 non-null	float64
22	Median_income_usd_usa	917 non-null	float64

dtypes: float64(19), int32(1), int64(3)
memory usage: 161.3 KB
None

- ◆ mobile_weight_g – уникальные значения:

```
[135. 140. 143. 146. 147. 150. 151. 153. 155. 156. 158. 159.
 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172.
 173. 174. 175. 176. 177. 178. 178.8 179. 180. 181. 182. 183.
 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195.
 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207.
 208. 209. 210. 211. 212. 213. 215. 216. 218. 219. 220. 221.
 222. 222.8 223. 225. 226. 227. 228. 229. 230. 231. 233. 234.
 235. 236. 238. 239. 240. 241. 242. 245. 250. 254. 255. 263.
 267. 280. 288. 295. 300.5 360. 366. 372. 420. 426. 433. 440.
 450. 458. 460. 465. 466. 468. 470. 475. 480. 482. 485. 490.
 495. 498. 499. 500. 503. 505. 508. 510. 520. 523. 530. 533.
 535. 540. 550. 555. 560. 567. 571. 580. 586. 590. 610. 674.
 682. 708. 726. 732. ]
```

- ◆ ram_gb – уникальные значения:

```
[ 1.  1.5  2.  3.  4.  6.  8.  10.  12.  16. ]
```

- ◆ front_camera_1_mp – уникальные значения:

```
[ 2.  5.  7.  8.  10.  10.5 10.7 10.8 11.1 12.  13.  16.  20.  24.
 25. 32. 42. 44. 48. 50. 60. ]
```

- ◆ front_camera_2_mp – уникальные значения:

```
[-1.  4.  8. 12.]
```

- ◆ front_camera_number – уникальные значения:

```
[1 2]
```

- ◆ back_camera_1 – уникальные значения:

```
[ 5.  8. 12. 12.2 13. 16. 20. 40. 48. 50. 54. 64.
 100. 108. 160. 200. ]
```

- ◆ back_camera_2 – уникальные значения:

```
[-1.  2.  5.  6.  8. 10. 12. 12.5 13. 16. 20. 32. 40. 48.
 50. 64. ]
```

- ◆ back_camera_3 – уникальные значения:

```
[-1.  2.  8. 12. 13. 16. 32. 40. 48. 50. 64.]
```

- ◆ back_camera_4 – уникальные значения:

```
[-1.  2.]
```

- ◆ back_camera_number – уникальные значения:

```
[1 2 3 4]
```

- ◆ battery_capacity_mah – уникальные значения:

```
[ 2000 2227 2300 2438 2600 2658 2716 2800 2815 2942 3000 3046
 3055 3095 3110 3140 3174 3200 3240 3260 3300 3315 3350 3500
 3600 3687 3700 3750 3800 3885 3900 3969 4000 4020 4025 4030
 4040 4050 4080 4085 4100 4115 4200 4300 4310 4325 4350 4352
 4355 4360 4385 4400 4410 4450 4460 4500 4510 4520 4575 4600
 4610 4614 4680 4700 4750 4800 4805 4815 4830 4870 4900 5000
 5003 5050 5065 5100 5110 5124 5160 5200 5250 5300 5400 5450
 5500 5550 5600 5700 5800 6000 6100 6400 6500 7000 7040 7100
 7200 7250 7500 7600 7608 7812 8000 8040 8200 8300 8340 8360
 8400 8500 8612 8850 9000 9510 9720 10000 10090 10100 10307 10500
 11000 11200]
```

- ◆ screen_size_inches – уникальные значения:

```
[ 5.  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.88 6.   6.01  6.09]
```

```
6.1  6.2  6.22 6.28 6.3  6.31 6.34 6.35 6.36 6.38 6.39 6.4
6.41 6.43 6.44 6.47 6.49 6.5  6.51 6.52 6.53 6.55 6.56 6.57
6.58 6.59 6.6  6.63 6.67 6.7  6.71 6.72 6.73 6.74 6.76 6.78
6.79 6.8  6.81 6.82 6.83 6.85 6.9  6.95 7.09 7.1  7.6  7.8
7.82 7.85 7.9  7.92 7.93 8.  8.7  9.7  10.1 10.2 10.4 10.5
10.9 11.  11.5 11.6 11.61 12.  12.1 12.2 12.3 12.4 12.6 12.9
13.  13.2 13.5 14.6 ]
```

◆ price_usa_usd – уникальные значения:

```
[ 79.  89.  99.  109. 119. 129. 130. 139. 149.
 150. 159. 160. 169. 170. 179. 180. 189. 190.
 199. 200. 210. 219. 220. 229. 230. 239. 240.
 249. 250. 259. 260. 269. 270. 279. 280. 289.
 290. 299. 300. 309. 310. 319. 320. 329. 330.
 340. 349. 350. 360. 370. 374.9 379. 380. 396.22
 399. 400. 420. 429. 440. 449. 450. 470. 479.
 499. 500. 529. 549. 599. 634.99 649. 699. 729.
 749. 790.77 799. 849. 877. 899. 949. 999. 1049.
1099. 1199. 1249. 1299. 1399. 1499. 1599. 1699. 1719.
1799. 1899. 2259. 2499. 2599. 2699. 2799. ]
```

◆ launched_year – уникальные значения:

```
[2014 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025]
```

◆ price_pakistan_usd – уникальные значения:

```
[ 57.  66.  68.  71.  78.  80.  82.  86.  89.  93.  96. 100.
 103. 107. 110. 114. 118. 125. 128. 132. 135. 139. 143. 150.
 153. 160. 164. 167. 171. 175. 178. 185. 189. 196. 200. 203.
 207. 210. 214. 221. 224. 232. 236. 242. 249. 254. 260. 267.
 285. 303. 306. 321. 338. 356. 374. 392. 428. 445. 463. 481.
 499. 534. 570. 575. 588. 606. 624. 641. 659. 677. 695. 702.
 713. 730. 748. 766. 784. 802. 819. 837. 855. 873. 891. 926.
 944. 962. 980. 998. 1015. 1033. 1051. 1069. 1087. 1105. 1122. 1140.
1158. 1176. 1194. 1211. 1229. 1237. 1247. 1265. 1283. 1300. 1318. 1354.
1372. 1390. 1425. 1496. 1532. 1603. 1639. 1675. 1710. 1782. 1888. 1942.
1960. 2156.]
```

◆ price_india_usd – уникальные значения:

```
[ 68.  79.  85.  90.  96. 102. 107. 113. 119. 124. 130. 135.
 136. 141. 147. 152. 153. 158. 169. 181. 192. 198. 203. 209.
 215. 220. 226. 237. 242. 248. 249. 260. 271. 282. 294. 305.
 316. 325. 328. 338. 339. 350. 361. 362. 373. 384. 395. 407.
 418. 429. 441. 451. 452. 475. 486. 497. 508. 520. 531. 542.
 564. 565. 576. 599. 610. 620. 621. 662. 667. 678. 733. 734.
 746. 768. 790. 791. 812. 825. 846. 847. 881. 903. 904. 959.
 960. 972. 1016. 1017. 1024. 1038. 1052. 1080. 1095. 1121. 1123. 1137. 1151.
1298. 1299. 1355. 1356. 1412. 1468. 1469. 1524. 1525. 1581. 1582. 1638.
1694. 1695. 1750. 1807. 1808. 1864. 1921. 1976. 2000. 2033. 2034. 2258.
2260. 2271. 2824. 2937. 3107.]
```

◆ price_china_usd – уникальные значения:

```
[ 70.  77.  84.  98. 112. 126. 140. 147. 154. 168. 175. 182.
 189. 196. 210. 211. 218. 224. 225. 238. 239. 252. 253. 267.
 280. 281. 295. 309. 323. 337. 351. 365. 379. 393. 407. 421.
 435. 449. 463. 477. 491. 505. 519. 533. 547. 561. 574. 589.
 603. 630. 631. 645. 659. 673. 702. 730. 744. 772. 786. 800.
 814. 828. 840. 842. 856. 870. 884. 898. 912. 926. 954. 981.
 982. 996. 1010. 1024. 1038. 1052. 1080. 1095. 1121. 1123. 1137. 1151.
1165. 1177. 1179. 1193. 1221. 1233. 1235. 1261. 1263. 1289. 1291. 1333.
1374. 1375. 1402. 1403. 1416. 1458. 1459. 1500. 1544. 1572. 1584. 1684.
1824. 1894. 1965. 2035. 2105. 2245. 2526.]
```

◆ price_dubai_usd – уникальные значения:

```
[ 81.  95. 109. 122. 136. 149. 150. 158. 163. 169. 171. 177.
 180. 185. 190. 191. 196. 199. 201. 204. 210. 212. 218. 223.
 226. 231. 234. 237. 240. 245. 250. 252. 258. 259. 261. 264.
 272. 278. 280. 286. 291. 299. 310. 313. 320. 326. 327. 329.
 340. 347. 348. 354. 359. 362. 368. 378. 381. 398. 408. 414.
 435. 441. 449. 456. 463. 470. 490. 501. 504. 517. 544. 558.
 571. 599. 626. 653. 680. 708. 735. 762. 789. 817. 844. 871.
 898. 925. 953. 980. 1007. 1034. 1062. 1089. 1116. 1143. 1170. 1198.
1225. 1252. 1279. 1307. 1334. 1361. 1388. 1416. 1443. 1497. 1579. 1633.
1661. 1688. 1769. 1797. 1906. 1933. 1960. 2042. 2096. 2368. 2450. 2722.
2858. 3022.]
```

◆ Median_income_usd_pakistan – уникальные значения:

```
[308.]
```

◆ Median_income_usd_india – уникальные значения:

```
[335.]
```

◆ Median_income_usd_china – уникальные значения:

```
[2213.]
```

◆ Median_income_usd_uae – уникальные значения:

```
[4601.]
```

◆ Median_income_usd_usa – уникальные значения:

```
[6285.]
```

```
Out[112...]: count    917.000000
mean      590.160174
std       422.676817
min       79.000000
25%      269.000000
50%      449.000000
75%      877.000000
max      2799.000000
Name: price_usa_usd, dtype: float64
```

📊 Категории цен смартфонов (по price_usa_usd)

Категория	Диапазон (USD)	Описание
Бюджетный	0–300	Доступные модели начального уровня
Средний	301–700	Сбалансированные по цене и функциям
Флагманский	701–1200	Топовые устройства с премиум-функциями
Ультрапремиум	>1200	Самые дорогие и эксклюзивные модели

```
In [113...]: # ♦ Категории цен смартфонов (на основе price_usa_usd)
df['price_category'] = pd.cut(
    df['price_usa_usd'],
    bins=[0, 300, 700, 1200, float('inf')],
    labels=['Бюджетный', 'Средний', 'Флагманский', 'Ультрапремиум']
)
```

```
In [114...]: df['ram_gb'].describe()
```

```
Out[114...]: count    917.000000
mean      7.798255
std       3.194082
min       1.000000
25%      6.000000
50%      8.000000
75%      8.000000
max      16.000000
Name: ram_gb, dtype: float64
```

MemoryWarning оперативной памяти (RAM)

Категория	Диапазон (ГБ)	Описание
Низкий	0–4	Подходит для базовых задач
Средний	4–8	Стандартные смартфоны и планшеты
Высокий	8+	Продвинутые устройства, флагманы

```
In [115...]: # ♦ Категории RAM
df['ram_category'] = pd.cut(
    df['ram_gb'],
    bins=[0, 4, 8, float('inf')],
    labels=['Низкий', 'Средний', 'Высокий']
)
```

```
In [116...]: # ♦ Соотношение RAM/цена
df['ram_per_dollar'] = (df['ram_gb'] / df['price_usa_usd']).round(4)
```

```
In [117...]: # ♦ Возраст телефона
df['phone_age'] = 2025 - df['launched_year']
```

```
In [118...]: # ♦ Метрика доступности: цена / медианная зарплата
df['affordability_pakistan'] = (df['price_pakistan_usd'] / df['Median_income_usd_pakistan']).round(3)
df['affordability_india'] = (df['price_india_usd'] / df['Median_income_usd_india']).round(3)
df['affordability_china'] = (df['price_china_usd'] / df['Median_income_usd_china']).round(3)
df['affordability_uae'] = (df['price_dubai_usd'] / df['Median_income_usd_uae']).round(3)
df['affordability_usa'] = (df['price_usa_usd'] / df['Median_income_usd_usa']).round(3)
```

📌 Пример:

Если , affordability_pk = 2.5 это значит: в Пакистане нужно примерно 2.5 месяца средней зарплаты, чтобы купить этот смартфон ниже итоговый даатфрейм

```
In [119...]: df[['battery_capacity_mah', 'screen_size_inches', 'mobile_weight_g']].describe()
```

Out[119...]

	battery_capacity_mah	screen_size_inches	mobile_weight_g
count	917.000000	917.000000	917.000000
mean	5030.122137	7.091821	228.857579
std	1364.100702	1.543000	106.055960
min	2000.000000	5.000000	135.000000
25%	4400.000000	6.500000	185.000000
50%	5000.000000	6.670000	195.000000
75%	5100.000000	6.780000	209.000000
max	11200.000000	14.600000	732.000000

📊 Категории мобильных устройств

- 🔋 Батарея (battery_capacity_mah)

Категория	Диапазон (мА·ч)	Описание
Малая	2000–3999	Бюджетные, компактные
Средняя	4000–5999	Стандартные смартфоны
Большая	6000–7999	Продвинутые устройства
Экстремальная	8000+	Планшеты, гибриды

- 📱 Экран (screen_size_inches)

Категория	Диапазон (дюймы)	Описание
Компактный	4.0–6.0	Маленькие, удобные в руке
Стандартный	6.1–6.9	Большинство современных
Большой	7.0–8.9	Планшеты, гибриды

- ⚖️ Вес (mobile_weight_g)

Категория	Диапазон (грамм)	Описание
Лёгкий	135–179	Ультралёгкие модели
Средний	180–229	Большинство смартфонов
Тяжёлый	230–399	Большие батареи, стекло/металл
Очень тяжёлый	400–732	Планшеты, защищённые устройства

In [120...]

```
# 🚀 Категоризация батарей
df['battery_category'] = pd.cut(
    df['battery_capacity_mah'],
    bins=[0, 3999, 5499, 6999, np.inf],
    labels=[
        'слабая',          # 'слабая (до 4000 мА·ч)'
        'средняя',         # 'средняя (4000–5499 мА·ч)'
        'большая',         # 'большая (5500–6999 мА·ч)'
        'экстремальная'   # 'экстремальная (7000+ мА·ч)'
    ]
)

# 📱 Категоризация экранов
df['screen_category'] = pd.cut(
    df['screen_size_inches'],
    bins=[0, 6.0, 6.9, 8.9, np.inf],
    labels=[
        'компактный',      # 'компактный (до 6")'
        'стандартный',    # 'стандартный (6.1–6.9")'
        'большой',          # 'большой (7.0–8.9")'
        'планшетный'       # 'планшетный (9.0"+)'
    ]
)

# ⚖️ Категоризация веса
df['weight_category'] = pd.cut(
    df['mobile_weight_g'],
    bins=[0, 179, 249, 399, np.inf], #
    labels=[
        'лёгкий',          # 'лёгкий (до 179г)'
        'средний',          # 'средний (180–249г)'
        'тяжёлый',          # 'тяжёлый (250–399г)'
        'планшетный'        # 'планшетный (400+ г)'
    ]
)
```

In [121...]

```
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 917 entries, 0 to 916
Data columns (total 38 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   company_name     917 non-null    object  
 1   model_name       917 non-null    object  
 2   mobile_weight_g  917 non-null    float64 
 3   ram_gb           917 non-null    float64 
 4   front_camera_1_mp 917 non-null    float64 
 5   front_camera_2_mp 917 non-null    float64 
 6   front_camera_number 917 non-null    int64  
 7   back_camera_1     917 non-null    float64 
 8   back_camera_2     917 non-null    float64 
 9   back_camera_3     917 non-null    float64 
 10  back_camera_4     917 non-null    float64 
 11  back_camera_number 917 non-null    int64  
 12  processor         917 non-null    object  
 13  battery_capacity_mah 917 non-null    int32  
 14  screen_size_inches 917 non-null    float64 
 15  price_usa_usd    917 non-null    float64 
 16  launched_year    917 non-null    int64  
 17  price_pakistan_usd 917 non-null    float64 
 18  price_india_usd   917 non-null    float64 
 19  price_china_usd   917 non-null    float64 
 20  price_dubai_usd   917 non-null    float64 
 21  Median_income_usd_pakistan 917 non-null    float64 
 22  Median_income_usd_india   917 non-null    float64 
 23  Median_income_usd_china   917 non-null    float64 
 24  Median_income_usd_uae    917 non-null    float64 
 25  Median_income_usd_usa    917 non-null    float64 
 26  price_category      917 non-null    category 
 27  ram_category        917 non-null    category 
 28  ram_per_dollar      917 non-null    float64 
 29  phone_age           917 non-null    int64  
 30  affordability_pakistan 917 non-null    float64 
 31  affordability_india   917 non-null    float64 
 32  affordability_china   917 non-null    float64 
 33  affordability_uae    917 non-null    float64 
 34  affordability_usa    917 non-null    float64 
 35  battery_category     917 non-null    category 
 36  screen_category      917 non-null    category 
 37  weight_category      917 non-null    category 

dtypes: category(5), float64(25), int32(1), int64(4), object(3)
memory usage: 238.4+ KB

```

```

In [122...]: new_order = ['company_name', 'model_name', 'mobile_weight_g', 'weight_category', 'ram_gb', 'ram_category', 'ram_per_dollar',
               'front_camera_1_mp', 'front_camera_2_mp',
               'front_camera_number', 'back_camera_1', 'back_camera_2',
               'back_camera_3', 'back_camera_4', 'back_camera_number', 'processor',
               'battery_capacity_mah', 'battery_category', 'screen_size_inches', 'screen_category',
               'launched_year', 'phone_age',
               'price_category', 'price_usa_usd', 'price_pakistan_usd', 'price_india_usd', 'price_china_usd', 'price_dubai_usd',
               'affordability_pakistan', 'affordability_india', 'affordability_china', 'affordability_uae', 'affordability_usa',
               'Median_income_usd_pakistan', 'Median_income_usd_india', 'Median_income_usd_china', 'Median_income_usd_uae', 'Median_income_usd_usa']
df = df[new_order + [col for col in df.columns if col not in new_order]]

```

```
In [123...]: df.iloc[:, :7]
```

	company_name	model_name	mobile_weight_g	weight_category	ram_gb	ram_category	ram_per_dollar
0	Apple	iPhone 16 128GB	174.0	лёгкий	6.0	Средний	0.0075
1	Apple	iPhone 16 256GB	174.0	лёгкий	6.0	Средний	0.0071
2	Apple	iPhone 16 512GB	174.0	лёгкий	6.0	Средний	0.0067
3	Apple	iPhone 16 Plus 128GB	203.0	средний	6.0	Средний	0.0067
4	Apple	iPhone 16 Plus 256GB	203.0	средний	6.0	Средний	0.0063
...
912	Samsung	Galaxy Z Fold6 256GB	239.0	средний	12.0	Высокий	0.0063
913	Samsung	Galaxy Z Fold6 512GB	239.0	средний	12.0	Высокий	0.0070
914	Samsung	Galaxy Z Fold6 1TB	239.0	средний	12.0	Высокий	0.0053
915	Huawei	P60 Pro	200.0	средний	12.0	Высокий	0.0109
916	Huawei	P60 Art	206.0	средний	12.0	Высокий	0.0092

917 rows × 7 columns

```
In [124...]: df.iloc[:, [0, 1] + list(range(7, 10))]
```

Out[124...]

	company_name	model_name	front_camera_1_mp	front_camera_2_mp	front_camera_number
0	Apple	iPhone 16 128GB	12.0	-1.0	1
1	Apple	iPhone 16 256GB	12.0	-1.0	1
2	Apple	iPhone 16 512GB	12.0	-1.0	1
3	Apple	iPhone 16 Plus 128GB	12.0	-1.0	1
4	Apple	iPhone 16 Plus 256GB	12.0	-1.0	1
...
912	Samsung	Galaxy Z Fold6 256GB	10.0	4.0	2
913	Samsung	Galaxy Z Fold6 512GB	10.0	4.0	2
914	Samsung	Galaxy Z Fold6 1TB	10.0	4.0	2
915	Huawei	P60 Pro	13.0	-1.0	1
916	Huawei	P60 Art	13.0	-1.0	1

917 rows × 5 columns

In [125...]

df.iloc[:, [0, 1] + list(range(10, 15))]

Out[125...]

	company_name	model_name	back_camera_1	back_camera_2	back_camera_3	back_camera_4	back_camera_number
0	Apple	iPhone 16 128GB	48.0	-1.0	-1.0	-1.0	1
1	Apple	iPhone 16 256GB	48.0	-1.0	-1.0	-1.0	1
2	Apple	iPhone 16 512GB	48.0	-1.0	-1.0	-1.0	1
3	Apple	iPhone 16 Plus 128GB	48.0	-1.0	-1.0	-1.0	1
4	Apple	iPhone 16 Plus 256GB	48.0	-1.0	-1.0	-1.0	1
...
912	Samsung	Galaxy Z Fold6 256GB	50.0	-1.0	-1.0	-1.0	1
913	Samsung	Galaxy Z Fold6 512GB	50.0	-1.0	-1.0	-1.0	1
914	Samsung	Galaxy Z Fold6 1TB	50.0	-1.0	-1.0	-1.0	1
915	Huawei	P60 Pro	48.0	13.0	48.0	-1.0	3
916	Huawei	P60 Art	48.0	40.0	48.0	-1.0	3

917 rows × 7 columns

In [126...]

df.iloc[:, [0, 1] + list(range(15, 20))]

Out[126...]

	company_name	model_name	processor	battery_capacity_mah	battery_category	screen_size_inches	screen_category
0	Apple	iPhone 16 128GB	A17 Bionic	3600	слабая	6.10	стандартный
1	Apple	iPhone 16 256GB	A17 Bionic	3600	слабая	6.10	стандартный
2	Apple	iPhone 16 512GB	A17 Bionic	3600	слабая	6.10	стандартный
3	Apple	iPhone 16 Plus 128GB	A17 Bionic	4200	средняя	6.70	стандартный
4	Apple	iPhone 16 Plus 256GB	A17 Bionic	4200	средняя	6.70	стандартный
...
912	Samsung	Galaxy Z Fold6 256GB	Snapdragon 8 Gen 3	4400	средняя	7.60	большой
913	Samsung	Galaxy Z Fold6 512GB	Snapdragon 8 Gen 3	4400	средняя	7.60	большой
914	Samsung	Galaxy Z Fold6 1TB	Snapdragon 8 Gen 3	4400	средняя	7.60	большой
915	Huawei	P60 Pro	Snapdragon 8+ Gen 1 4G	4815	средняя	6.67	стандартный
916	Huawei	P60 Art	Snapdragon 8+ Gen 1 4G	5100	средняя	6.73	стандартный

917 rows × 7 columns

In [127...]

df.iloc[:, [0, 1] + list(range(20, 22))]

Out[127...]

	company_name	model_name	launched_year	phone_age
0	Apple	iPhone 16 128GB	2024	1
1	Apple	iPhone 16 256GB	2024	1
2	Apple	iPhone 16 512GB	2024	1
3	Apple	iPhone 16 Plus 128GB	2024	1
4	Apple	iPhone 16 Plus 256GB	2024	1
...
912	Samsung	Galaxy Z Fold6 256GB	2024	1
913	Samsung	Galaxy Z Fold6 512GB	2024	1
914	Samsung	Galaxy Z Fold6 1TB	2024	1
915	Huawei	P60 Pro	2023	2
916	Huawei	P60 Art	2023	2

917 rows × 4 columns

In [128...]

df.iloc[:, [0, 1] + list(range(22, 28))]

Out[128...]

	company_name	model_name	price_category	price_usa_usd	price_pakistan_usd	price_india_usd	price_china_usd	price_dubai_usd
0	Apple	iPhone 16 128GB	Флагманский	799.0	802.0	904.0	814.0	762.0
1	Apple	iPhone 16 256GB	Флагманский	849.0	837.0	960.0	856.0	817.0
2	Apple	iPhone 16 512GB	Флагманский	899.0	873.0	1017.0	912.0	871.0
3	Apple	iPhone 16 Plus 128GB	Флагманский	899.0	891.0	1017.0	870.0	871.0
4	Apple	iPhone 16 Plus 256GB	Флагманский	949.0	926.0	1073.0	912.0	925.0
...
912	Samsung	Galaxy Z Fold6 256GB	Ультрапремиум	1899.0	2156.0	1864.0	1965.0	1960.0
913	Samsung	Galaxy Z Fold6 512GB	Ультрапремиум	1719.0	1942.0	2000.0	2245.0	2096.0
914	Samsung	Galaxy Z Fold6 1TB	Ультрапремиум	2259.0	1237.0	2271.0	2526.0	2368.0
915	Huawei	P60 Pro	Флагманский	1099.0	677.0	1130.0	1121.0	1143.0
916	Huawei	P60 Art	Ультрапремиум	1299.0	784.0	1356.0	1261.0	1307.0

917 rows × 8 columns

In [129...]

df.iloc[:, [0, 1] + list(range(28, 33))]

Out[129...]

	company_name	model_name	affordability_pakistan	affordability_india	affordability_china	affordability_uae	affordability_usa
0	Apple	iPhone 16 128GB	2.604	2.699	0.368	0.166	0.127
1	Apple	iPhone 16 256GB	2.718	2.866	0.387	0.178	0.135
2	Apple	iPhone 16 512GB	2.834	3.036	0.412	0.189	0.143
3	Apple	iPhone 16 Plus 128GB	2.893	3.036	0.393	0.189	0.143
4	Apple	iPhone 16 Plus 256GB	3.006	3.203	0.412	0.201	0.151
...
912	Samsung	Galaxy Z Fold6 256GB	7.000	5.564	0.888	0.426	0.302
913	Samsung	Galaxy Z Fold6 512GB	6.305	5.970	1.014	0.456	0.274
914	Samsung	Galaxy Z Fold6 1TB	4.016	6.779	1.141	0.515	0.359
915	Huawei	P60 Pro	2.198	3.373	0.507	0.248	0.175
916	Huawei	P60 Art	2.545	4.048	0.570	0.284	0.207

917 rows × 7 columns

⚠ Проблемы и решения

Проблема	Решение
Неоднородные форматы (единицы измерения, валюты)	Удаление лишних символов, приведение к числовому типу
Множественные камеры в одной ячейке	Разделение на отдельные столбцы
Пропуски в ценах	Восстановление через регрессию
Дубликаты	Удаление полных дубликатов, анализ неявных

Выводы по шагу 2

Очистка и структурирование данных

- Все исходные столбцы успешно переименованы в стиле `snake_case`, что улучшило читаемость и облегчает работу с ними в Python
- Из столбцов удалены единицы измерения (g, GB, mAh, inches) и текстовые префиксы валют (PKR, INR, CNY, USD, AED).
- Проведено разделение признаков: теперь каждая характеристика — отдельное числовое поле (`ram_gb`, `battery_capacity_mah`, `screen_size_inches` и т.д.).
- Удалены пробелы, символы переноса строк и другие артефакты кодировки
 - символ Ъ в китайских ценах
- Цены во всех странах успешно приведены к числовому формату (`float64`).

Обработка пропусков

- После очистки были обнаружены пропуски в характеристиках камер (`front_camera_2_mp`, `back_camera_2`, `back_camera_3`, `back_camera_4`) — в основном для моделей с одним модулем камеры.
 - пропуски в камерах заполнены значением -1 (отсутствие камеры).
- Обнаружен единственный пропуск в `price_pakistan_pk` (модель Samsung Galaxy Z Fold6 1TB).
 - был восстановлен с помощью предсказательной модели Gradient Boosting на основе цен в других странах и технических характеристик.
 - модель показала высокое качество ($R^2 \approx 0.89$, $RMSE \approx 33\,000$), что делает восстановленное значение статистически обоснованным.

Приведение цен к единой валюте

- Каждая страна имеет собственный столбец цены, теперь все они приведены к числовому формату.
- На следующем этапе конвертация всех цен в USD с помощью актуальных валютных курсов

Создание новых признаков

- подгружены данные о средней зарплате по странам.
- возраст устройства (`device_age` = 2025 - `launched_year`);
- категории RAM (низкий/средний/высокий);
- категории цен (бюджетный/средний/флагман);
- показатель доступности: `affordability_index` = `price_usd` / `median_salary_country`.

Дубликаты найдены

- Удалены полные дубликаты (15 строк).
- Выявлены неявные дубликаты по паре `company_name` + `model_name`, но они сохранены, так как различаются характеристиками.

* к содержанию

Шаг 3: Исследовательский анализ данных

In [130...]

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 917 entries, 0 to 916
Data columns (total 38 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   company_name     917 non-null    object  
 1   model_name       917 non-null    object  
 2   mobile_weight_g  917 non-null    float64 
 3   weight_category 917 non-null    category 
 4   ram_gb           917 non-null    float64 
 5   ram_category     917 non-null    category 
 6   ram_per_dollar   917 non-null    float64 
 7   front_camera_1_mp 917 non-null    float64 
 8   front_camera_2_mp 917 non-null    float64 
 9   front_camera_number 917 non-null    int64  
 10  back_camera_1    917 non-null    float64 
 11  back_camera_2    917 non-null    float64 
 12  back_camera_3    917 non-null    float64 
 13  back_camera_4    917 non-null    float64 
 14  back_camera_number 917 non-null    int64  
 15  processor        917 non-null    object  
 16  battery_capacity_mah 917 non-null    int32  
 17  battery_category 917 non-null    category 
 18  screen_size_inches 917 non-null    float64 
 19  screen_category  917 non-null    category 
 20  launched_year    917 non-null    int64  
 21  phone_age         917 non-null    int64  
 22  price_category   917 non-null    category 
 23  price_usa_usd   917 non-null    float64 
 24  price_pakistan_usd 917 non-null    float64 
 25  price_india_usd 917 non-null    float64 
 26  price_china_usd 917 non-null    float64 
 27  price_dubai_usd 917 non-null    float64 
 28  affordability_pakistan 917 non-null    float64 
 29  affordability_india 917 non-null    float64 
 30  affordability_china 917 non-null    float64 
 31  affordability_uae 917 non-null    float64 
 32  affordability_usa 917 non-null    float64 
 33  Median_income_usd_pakistan 917 non-null    float64 
 34  Median_income_usd_india 917 non-null    float64 
 35  Median_income_usd_china 917 non-null    float64 
 36  Median_income_usd_uae 917 non-null    float64 
 37  Median_income_usd_usa 917 non-null    float64 
dtypes: category(5), float64(25), int32(1), int64(4), object(3)
memory usage: 238.4+ KB
```

Количественные переменные

```
In [131...]: df.select_dtypes(include='number').describe().T
```

Out[131...]

		count	mean	std	min	25%	50%	75%	max
	mobile_weight_g	917.0	228.857579	106.055960	135.0000	185.0000	195.0000	209.000	732.0000
	ram_gb	917.0	7.798255	3.194082	1.0000	6.0000	8.0000	8.000	16.0000
	ram_per_dollar	917.0	0.017823	0.008957	0.0027	0.0109	0.0172	0.024	0.0524
	front_camera_1_mp	917.0	18.155507	12.032736	2.0000	8.0000	16.0000	32.000	60.0000
	front_camera_2_mp	917.0	-0.886587	1.028101	-1.0000	-1.0000	-1.0000	-1.000	12.0000
	front_camera_number	917.0	1.013086	0.113706	1.0000	1.0000	1.0000	1.000	2.0000
	back_camera_1	917.0	46.494875	31.090258	5.0000	13.0000	50.0000	50.000	200.0000
	back_camera_2	917.0	5.403490	11.887032	-1.0000	-1.0000	-1.0000	8.000	64.0000
	back_camera_3	917.0	2.065431	10.801306	-1.0000	-1.0000	-1.0000	-1.000	64.0000
	back_camera_4	917.0	-0.941112	0.416394	-1.0000	-1.0000	-1.0000	-1.000	2.0000
	back_camera_number	917.0	1.618321	0.794641	1.0000	1.0000	1.0000	2.000	4.0000
	battery_capacity_mah	917.0	5030.122137	1364.100702	2000.0000	4400.0000	5000.0000	5100.000	11200.0000
	screen_size_inches	917.0	7.091821	1.543000	5.0000	6.5000	6.6700	6.780	14.6000
	launched_year	917.0	2022.207197	1.867509	2014.0000	2021.0000	2023.0000	2024.000	2025.0000
	phone_age	917.0	2.792803	1.867509	0.0000	1.0000	2.0000	4.000	11.0000
	price_usa_usd	917.0	590.160174	422.676817	79.0000	269.0000	449.0000	877.000	2799.0000
	price_pakistan_usd	917.0	452.416576	364.084443	57.0000	196.0000	321.0000	641.000	2156.0000
	price_india_usd	917.0	578.430752	464.994265	68.0000	226.0000	407.0000	847.000	3107.0000
	price_china_usd	917.0	542.279171	390.397027	70.0000	252.0000	421.0000	786.000	2526.0000
	price_dubai_usd	917.0	600.272628	428.839118	81.0000	272.0000	463.0000	898.000	3022.0000
	affordability_pakistan	917.0	1.468836	1.182150	0.1850	0.6360	1.0420	2.081	7.0000
	affordability_india	917.0	1.726642	1.388058	0.2030	0.6750	1.2150	2.528	9.2750
	affordability_china	917.0	0.245074	0.176351	0.0320	0.1140	0.1900	0.355	1.1410
	affordability_uae	917.0	0.130438	0.093228	0.0180	0.0590	0.1010	0.195	0.6570
	affordability_usa	917.0	0.093878	0.067277	0.0130	0.0430	0.0710	0.140	0.4450
	Median_income_usd_pakistan	917.0	308.000000	0.000000	308.0000	308.0000	308.0000	308.000	308.0000
	Median_income_usd_india	917.0	335.000000	0.000000	335.0000	335.0000	335.0000	335.000	335.0000
	Median_income_usd_china	917.0	2213.000000	0.000000	2213.0000	2213.0000	2213.0000	2213.000	2213.0000
	Median_income_usd_uae	917.0	4601.000000	0.000000	4601.0000	4601.0000	4601.0000	4601.000	4601.0000
	Median_income_usd_usa	917.0	6285.000000	0.000000	6285.0000	6285.0000	6285.0000	6285.000	6285.0000

In [132...]

```

def plot_box_and_hist_num(data, feature='mobile_weight_g', title=None, bins=60):
    # Удаляем пропуски
    df = data[[feature]].dropna()

    # Общая статистика по признаку
    print(f"\n{colorama.Fore.CYAN} Общая статистика по признаку '{feature}' :{colorama.Fore.RESET}")
    print(df[feature].describe())

    # Заголовок графика по умолчанию
    if title is None:
        title = f'Распределение признака "{feature}"'

    # Создаём сетку 2 строки x 2 столбца
    fig, axes = plt.subplots(2, 2, figsize=(12, 12))
    fig.suptitle(title, fontsize=20)

    # Ящик с усами – первая строка, объединённые столбцы
    sns.boxplot(data=df, y=feature, ax=axes[0, 0])
    axes[0, 0].set_title('Ящик с усами', fontsize=16)
    axes[0, 0].set_ylabel(feature, fontsize=14)
    axes[0, 0].tick_params(axis='y', labelsize=12)
    axes[0, 0].tick_params(axis='x', labelsize=12)

    # Отключаем вторую ячейку в первой строке
    axes[0, 1].axis('off')

    # Обычная гистограмма – вторая строка, слева
    sns.histplot(data=df, x=feature, bins=bins,
                 color='skyblue', edgecolor='gray', ax=axes[1, 0])
    axes[1, 0].set_title('Гистограмма (частота)', fontsize=16)
    axes[1, 0].set_xlabel(feature, fontsize=14)
    axes[1, 0].set_ylabel('Частота', fontsize=14)
    axes[1, 0].tick_params(axis='x', labelsize=12)
    axes[1, 0].tick_params(axis='y', labelsize=12)

    # Нормированная гистограмма – вторая строка, справа

```

```

sns.histplot(data=df, x=feature, bins=bins,
             stat="density", color='steelblue', ax=axes[1, 1])
axes[1, 1].set_title('Нормированная гистограмма (плотность)', fontsize=16)
axes[1, 1].set_xlabel(feature, fontsize=14)
axes[1, 1].set_ylabel('Плотность', fontsize=14)
axes[1, 1].tick_params(axis='x', labelsize=12)
axes[1, 1].tick_params(axis='y', labelsize=12)

plt.tight_layout(rect=[0, 0, 1, 0.96]) # Оставляем место для заголовка
plt.show()

```

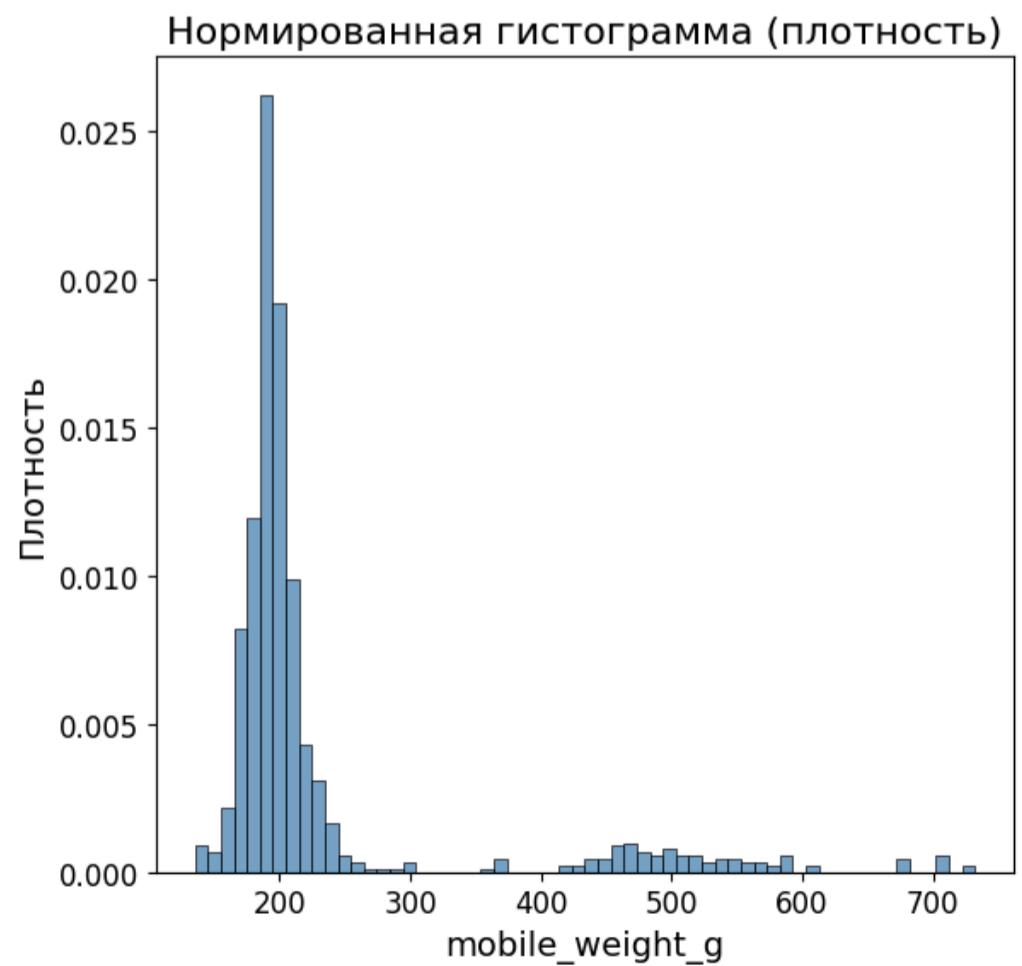
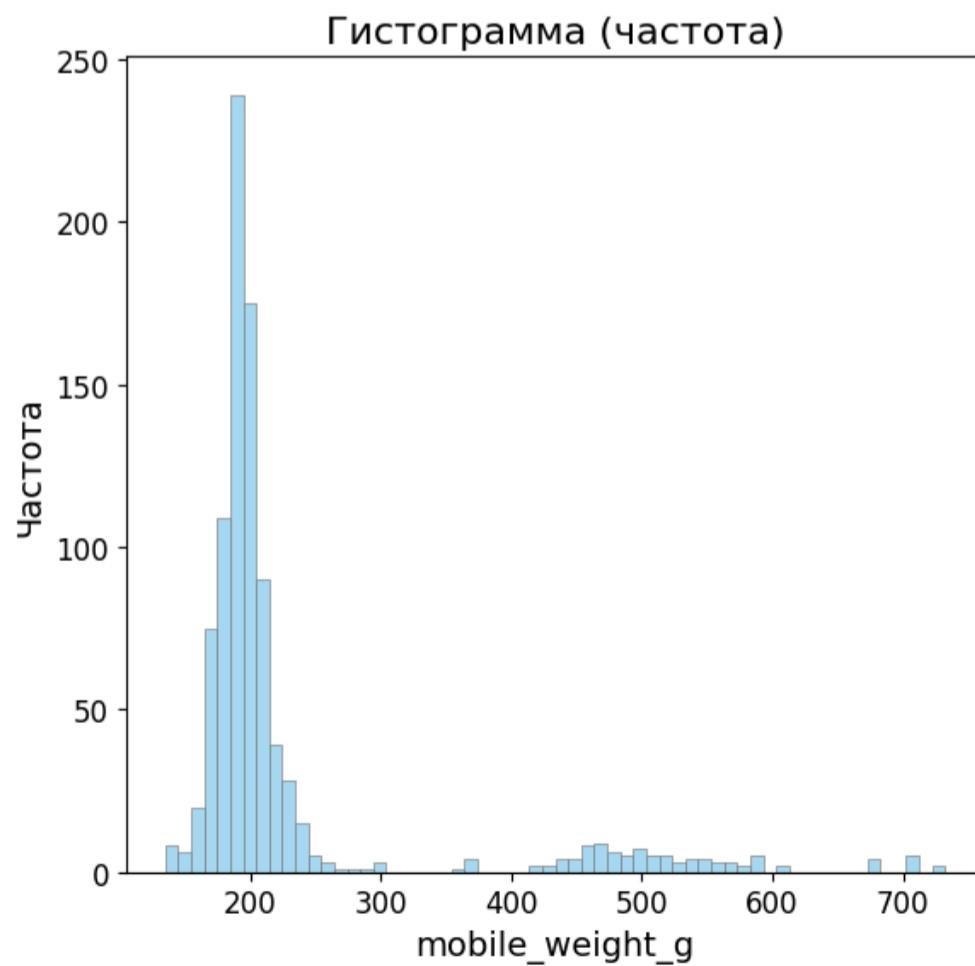
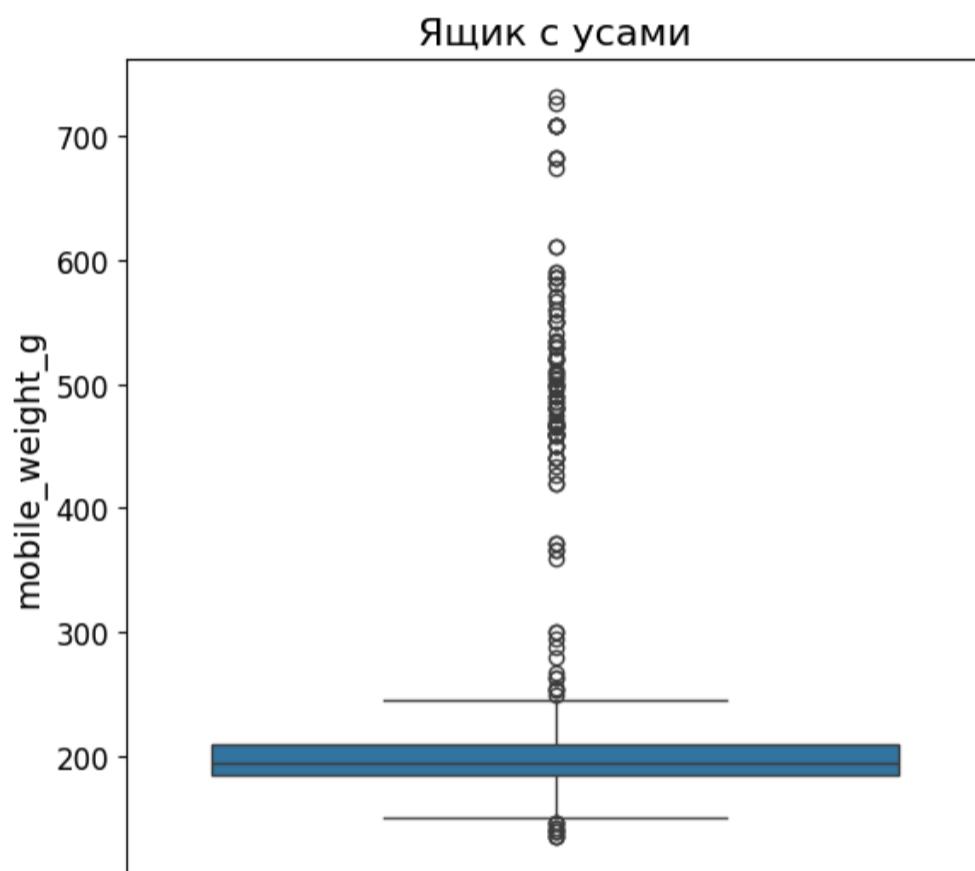
In [133...]

```
plot_box_and_hist_num(data=df, feature='mobile_weight_g',
                      title='Распределение веса мобильных устройств, граммы')
```

Общая статистика по признаку 'mobile_weight_g':

count	917.000000
mean	228.857579
std	106.055960
min	135.000000
25%	185.000000
50%	195.000000
75%	209.000000
max	732.000000
Name:	mobile_weight_g, dtype: float64

Распределение веса мобильных устройств, граммы



Вес устройства (mobile_weight_g)

- Средний вес: ~229 г, медиана: 195 г.
- Распределение правостороннее — есть тяжелые устройства (до 732 г), вероятно, планшеты или защищенные модели.
- Большинство устройств сосредоточено в диапазоне 185–209 г.

In [134...]

```
def plot_box_and_bar_num(data, feature='ram_gb', title=None):
    # Удаляем пропуски
```

```

df = data[[feature]].dropna()

# Общая статистика по признаку
print(F"\n📊 Общая статистика по признаку '{feature}':")
print(df[feature].describe())

# Заголовок графика по умолчанию
if title is None:
    title = f'Распределение признака "{feature}"'

# Создаём сетку 2 строки x 2 столбца
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
fig.suptitle(title, fontsize=20)

# Ящик с усами
sns.boxplot(data=df, y=feature, ax=axes[0, 0])
axes[0, 0].set_title('Ящик с усами', fontsize=16)
axes[0, 0].set_ylabel(feature, fontsize=14)

# Отключаем вторую ячейку
axes[0, 1].axis('off')

# Столбчатая диаграмма – частота
value_counts = df[feature].value_counts().sort_index()
sns.barplot(x=value_counts.index, y=value_counts.values, ax=axes[1, 0], color='skyblue')
axes[1, 0].set_title('Столбчатая диаграмма (частота)', fontsize=16)
axes[1, 0].set_xlabel(feature, fontsize=14)
axes[1, 0].set_ylabel('Частота', fontsize=14)

# Столбчатая диаграмма – плотность
density = value_counts / value_counts.sum()
sns.barplot(x=density.index, y=density.values, ax=axes[1, 1], color='steelblue')
axes[1, 1].set_title('Столбчатая диаграмма (плотность)', fontsize=16)
axes[1, 1].set_xlabel(feature, fontsize=14)
axes[1, 1].set_ylabel('Плотность', fontsize=14)

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

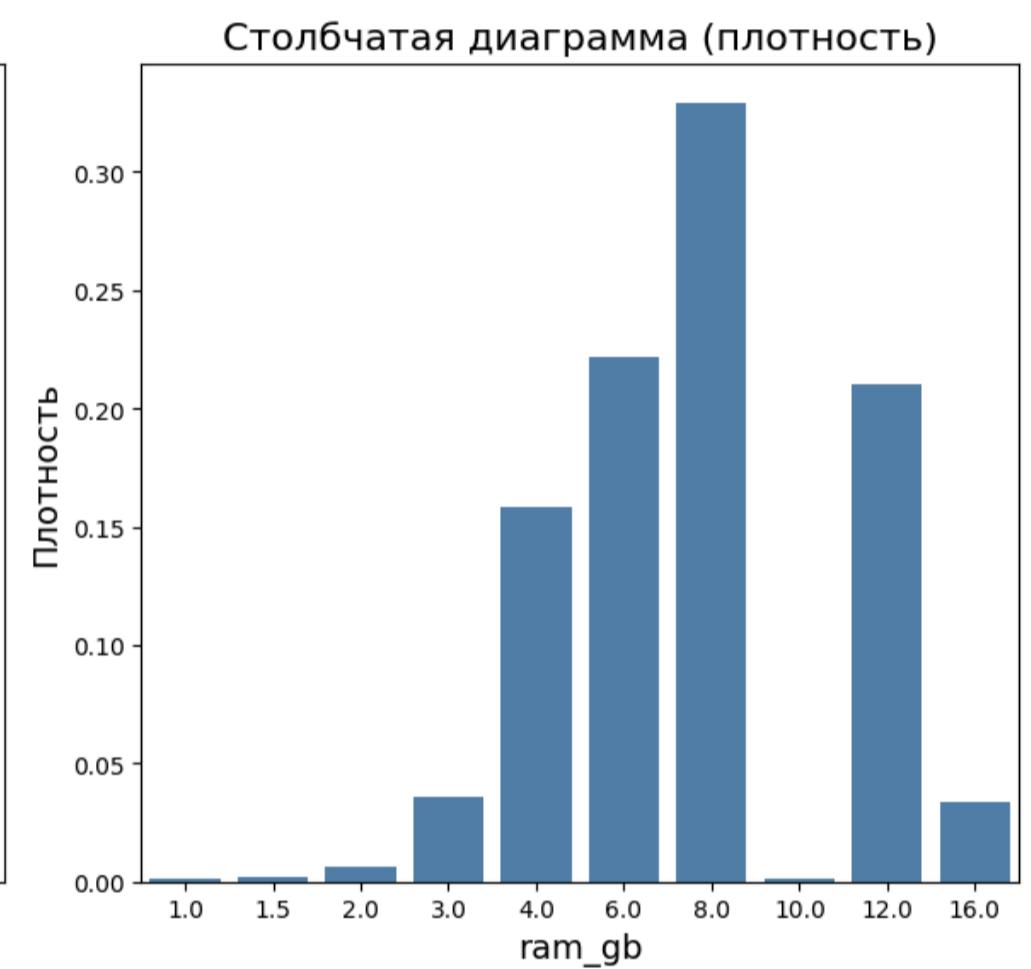
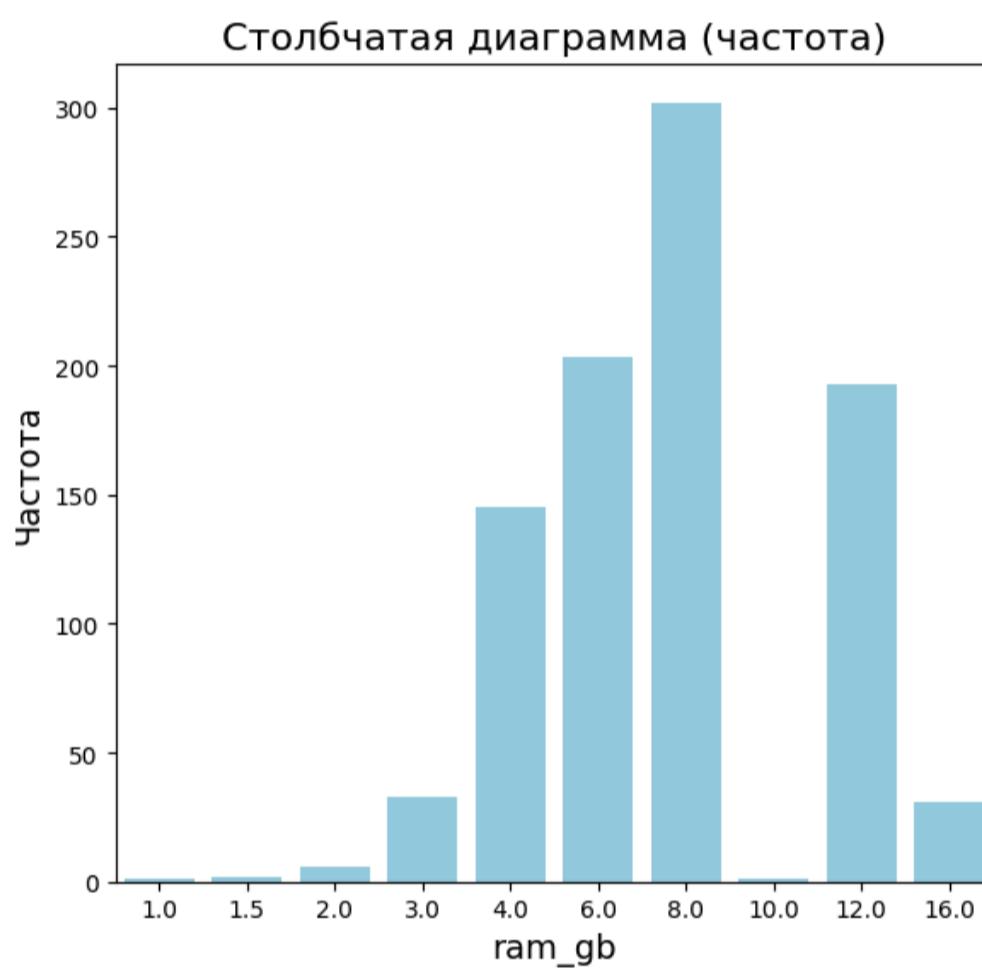
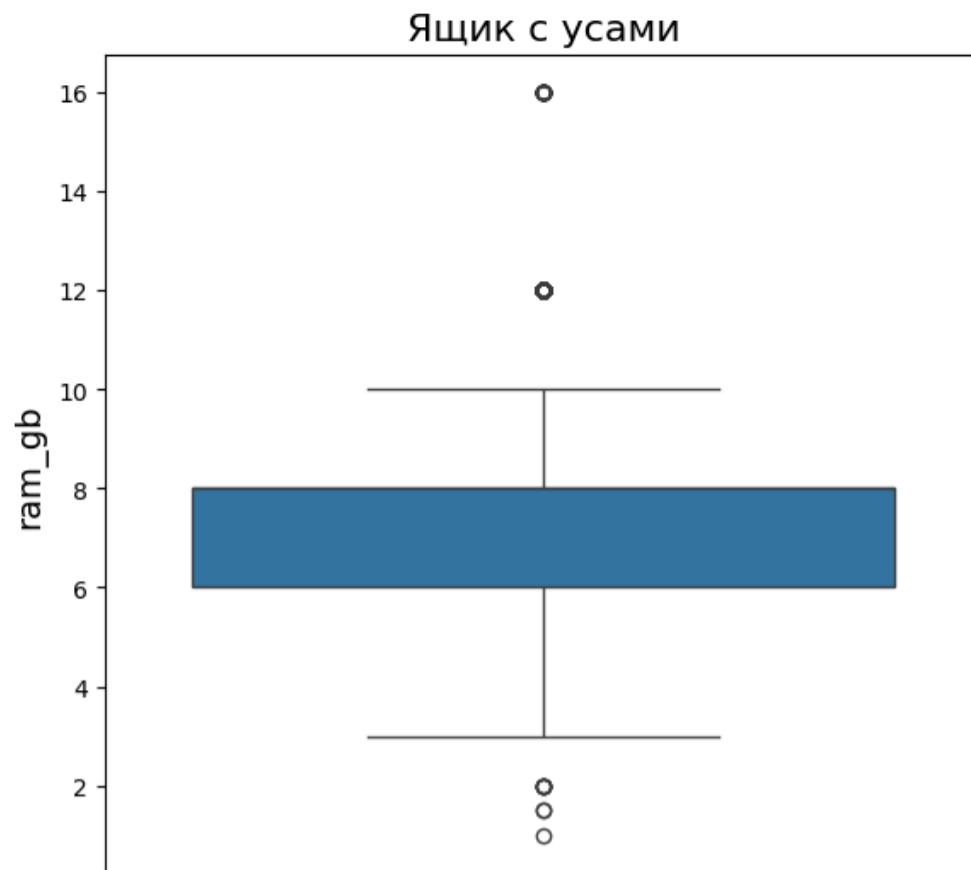
```

In [135]: plot_box_and_bar_num(data=df, feature='ram_gb',
title='Распределение объёма оперативной памяти (ОЗУ) мобильных устройств, Гб')

📊 Общая статистика по признаку 'ram_gb':

count	917.000000
mean	7.798255
std	3.194082
min	1.000000
25%	6.000000
50%	8.000000
75%	8.000000
max	16.000000
Name:	ram_gb, dtype: float64

Распределение объёма оперативной памяти (ОЗУ) мобильных устройств, Гб



Оперативная память (RAM)

- Средний объем RAM: ~7.8 Гб, медиана: 8 Гб.
- Наиболее частые значения: 6, 8, 12 Гб.
- Высокий RAM (16 Гб) встречается реже.

In [136...]

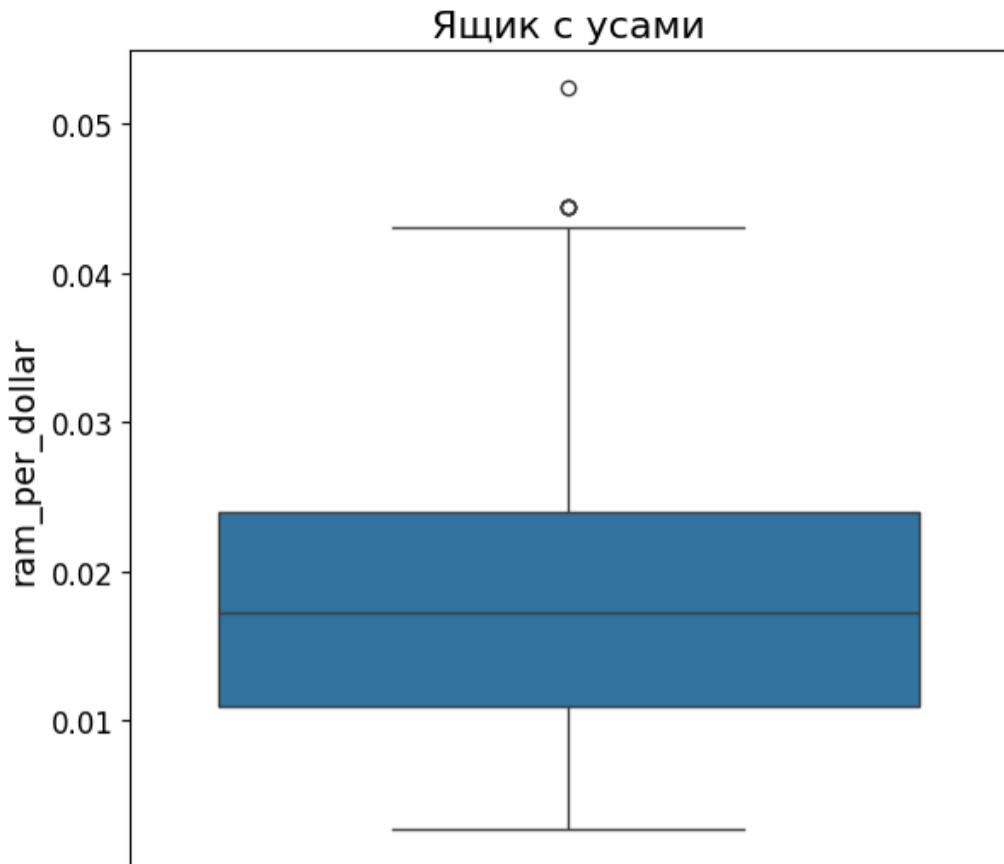
```
plot_box_and_hist_num(data=df, feature='ram_per_dollar',
                      title='Распределение соотношения RAM/цена')
```

Общая статистика по признаку 'ram_per_dollar':

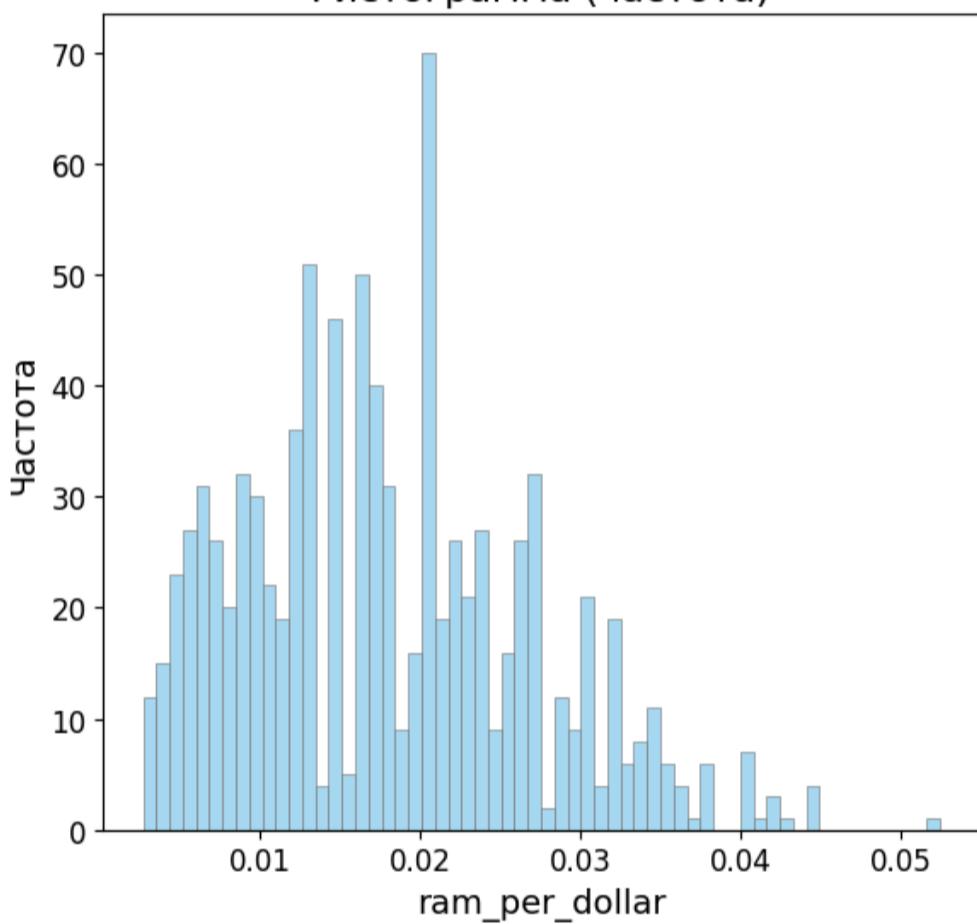
Статистика	Значение
count	917.000000
mean	0.017823
std	0.008957
min	0.002700
25%	0.010900
50%	0.017200
75%	0.024000
max	0.052400

Name: ram_per_dollar, dtype: float64

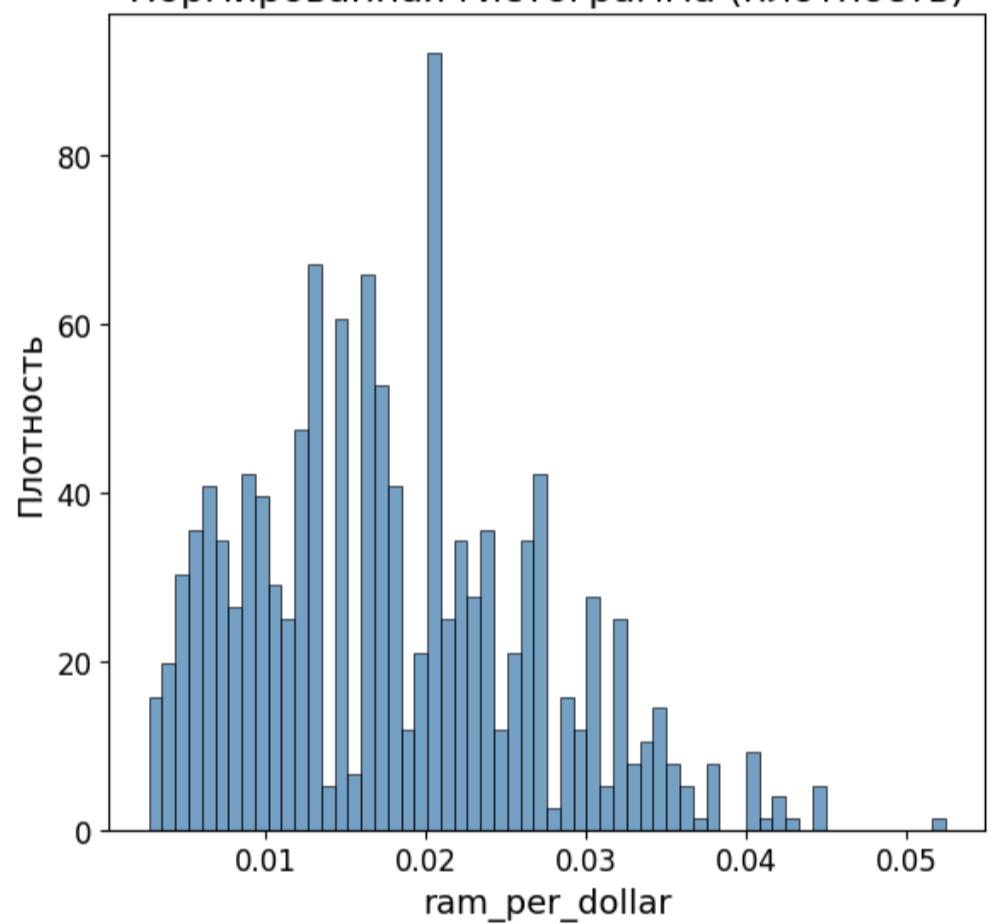
Распределение соотношения RAM/цена



Гистограмма (частота)



Нормированная гистограмма (плотность)



Соотношение RAM/цена (ram_per_dollar)

- Среднее: ~0.018 Гб/\$
- Показатель сильно варьируется: от 0.0027 до 0.0524.

Выводы:

- Бюджетные устройства предлагают в 2-3 раза больше RAM на доллар
- Премиальные бренды "закладывают" стоимость бренда в цену
- Показатель эффективно отражает ценовую политику производителей

In [137...]:

```
def plot_camera_features(data, features=None, title=None, camera_type='auto'):  
    # Если признаки не заданы – выбираем по типу камеры  
    if features is None:  
        if camera_type == 'back':  
            features = ['back_camera_1', 'back_camera_2', 'back_camera_3', 'back_camera_4']  
        else:  
            features = ['front_camera_1_mp', 'front_camera_2_mp', 'front_camera_number']  
            camera_type = 'front'  
  
    df = data[features].dropna()  
  
    # Выводим статистику  
    icon = '📸' if camera_type == 'front' else '📷'  
    print(f"\n{icon} Общая статистика по признакам {camera_type}-камер:")  
    print(df.describe().T)  
  
    # Заголовок по умолчанию  
    if title is None:  
        title = f'Распределение характеристик {camera_type}-камер, MP'
```

```

n = len(features)
fig, axes = plt.subplots(2, n, figsize=(8 * n, 12))
fig.suptitle(title, fontsize=20)

# Цветовая палитра
color = 'steelblue' if camera_type == 'front' else 'darkcyan'

# Ящики с усами – верхняя строка
for i, feature in enumerate(features):
    sns.boxplot(data=df, y=feature, ax=axes[0, i])
    axes[0, i].set_title(f'Ящик с усами: {feature}', fontsize=16)
    axes[0, i].set_ylabel(feature, fontsize=14)
    axes[0, i].tick_params(axis='y', labelsize=12)

# Столбчатые диаграммы – нижняя строка
for i, feature in enumerate(features):
    value_counts = df[feature].value_counts().sort_index()
    sns.barplot(x=value_counts.index, y=value_counts.values, ax=axes[1, i], color=color)
    axes[1, i].set_title(f'Столбчатая диаграмма: {feature}', fontsize=16)
    axes[1, i].set_xlabel('Значение', fontsize=14)
    axes[1, i].set_ylabel('Частота', fontsize=14)
    axes[1, i].tick_params(axis='x', labelsize=12)
    axes[1, i].tick_params(axis='y', labelsize=12)

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

```

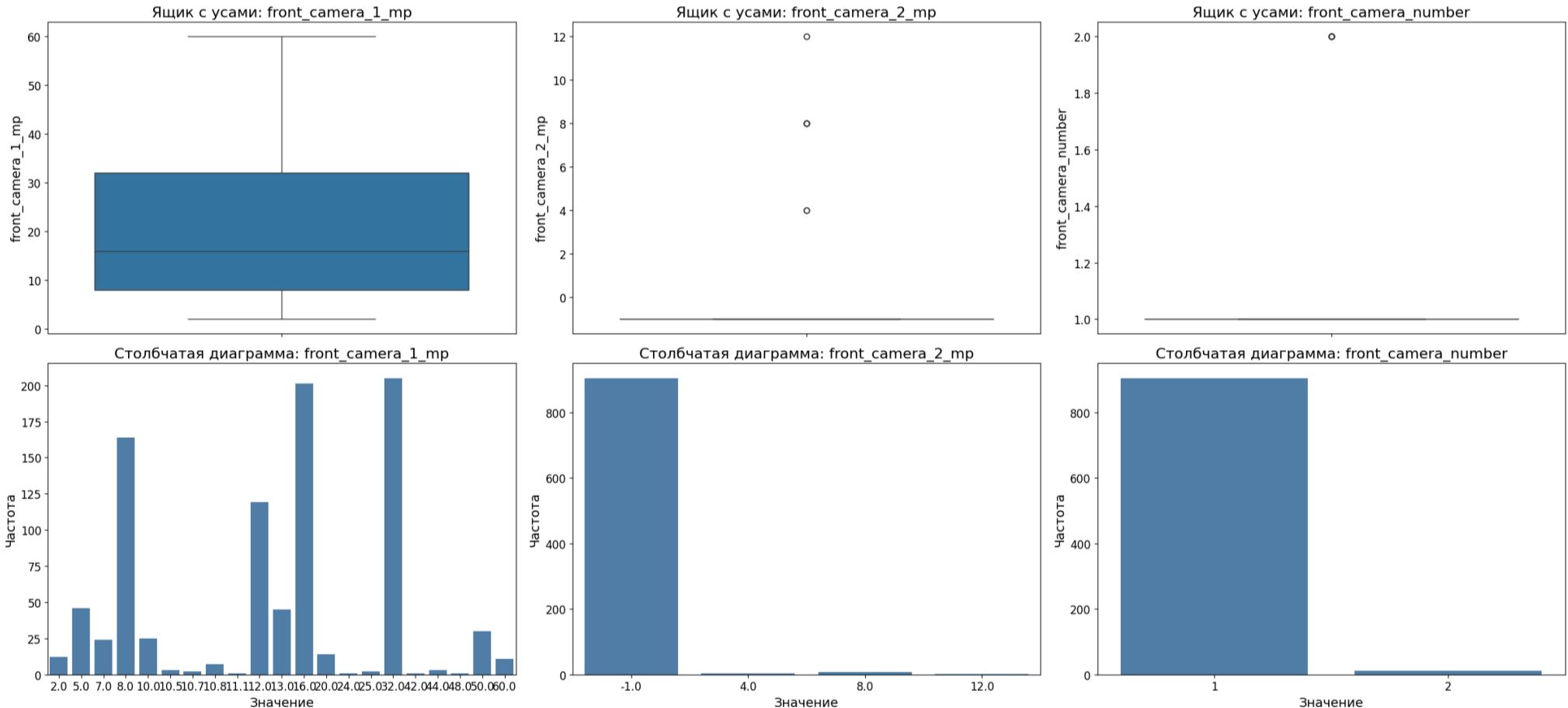
In [138...]

```
# Для фронтальных камер
plot_camera_features(df, camera_type='front')
```

📸 Общая статистика по признакам front-камер:

	count	mean	std	min	25%	50%	75%	max
front_camera_1_mp	917.0	18.155507	12.032736	2.0	8.0	16.0	32.0	60.0
front_camera_2_mp	917.0	-0.886587	1.028101	-1.0	-1.0	-1.0	-1.0	12.0
front_camera_number	917.0	1.013086	0.113706	1.0	1.0	1.0	1.0	2.0

Распределение характеристик front-камер, MP

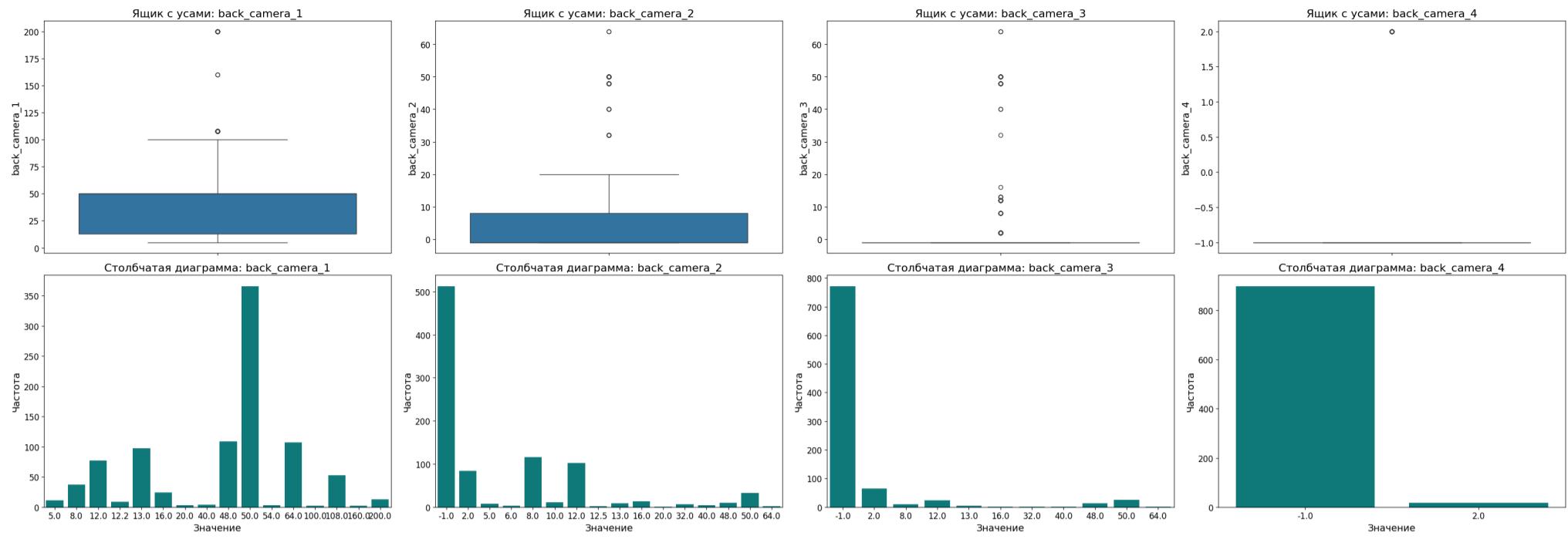


In [139...]

```
# Для задних камер
plot_camera_features(df, camera_type='back')
```

📸 Общая статистика по признакам back-камер:

	count	mean	std	min	25%	50%	75%	max
back_camera_1	917.0	46.494875	31.090258	5.0	13.0	50.0	50.0	200.0
back_camera_2	917.0	5.403490	11.887032	-1.0	-1.0	-1.0	8.0	64.0
back_camera_3	917.0	2.065431	10.801306	-1.0	-1.0	-1.0	-1.0	64.0
back_camera_4	917.0	-0.941112	0.416394	-1.0	-1.0	-1.0	-1.0	2.0



Камеры

- Фронтальная: в основном 8–16 Мп, но встречаются экстремумы до 60 МП (для «селфи-ориентированных» устройств), редко две камеры.
- Основная: основная камера 50 Мп — мода, есть модели с 200 Мп (специфичные линейки (маркетинговые пиксели/комбинированные сенсоры)

Многие устройства имеют только одну основную камеру, реже — 2–4 модуля.

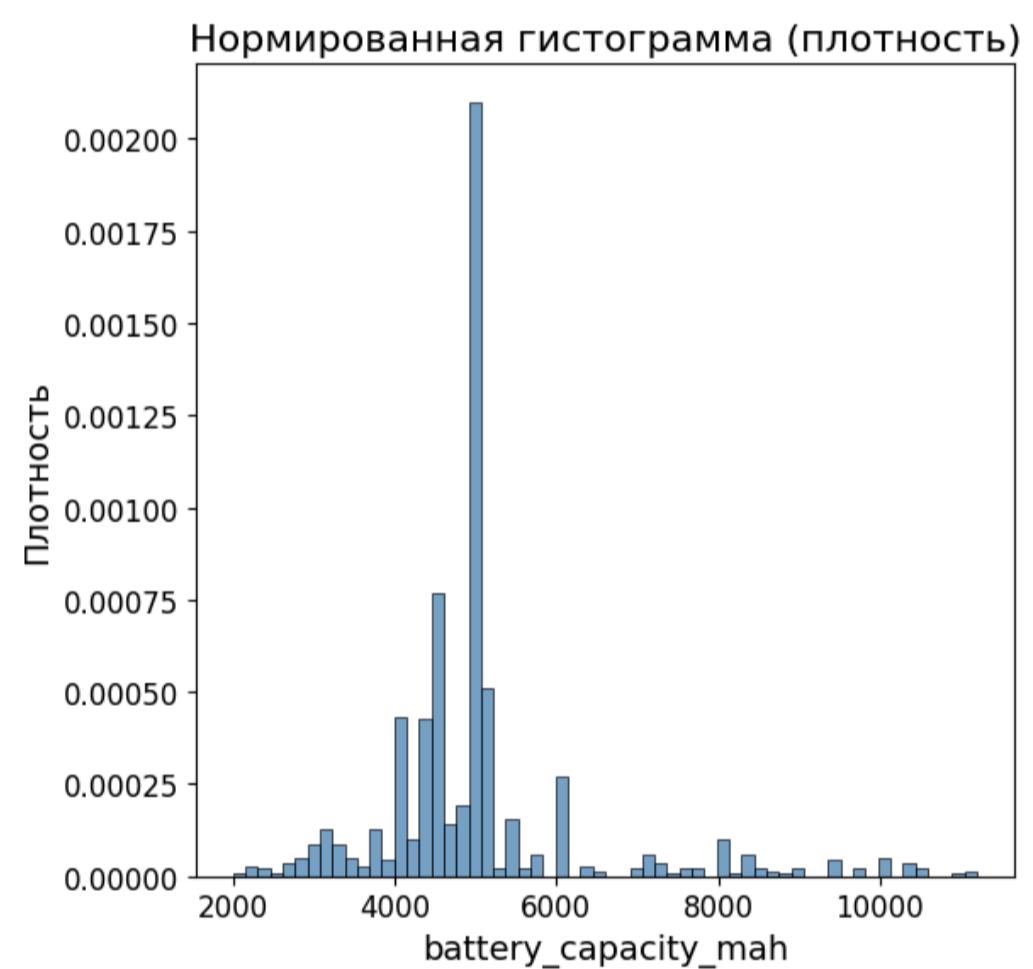
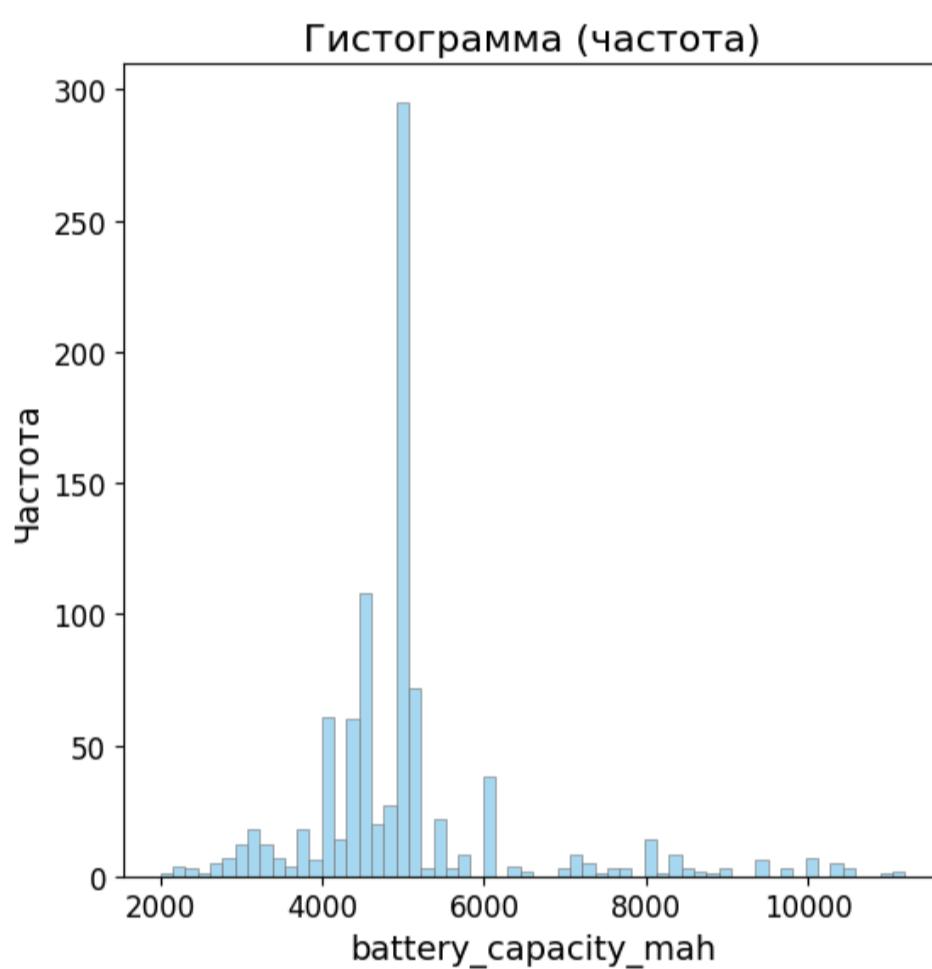
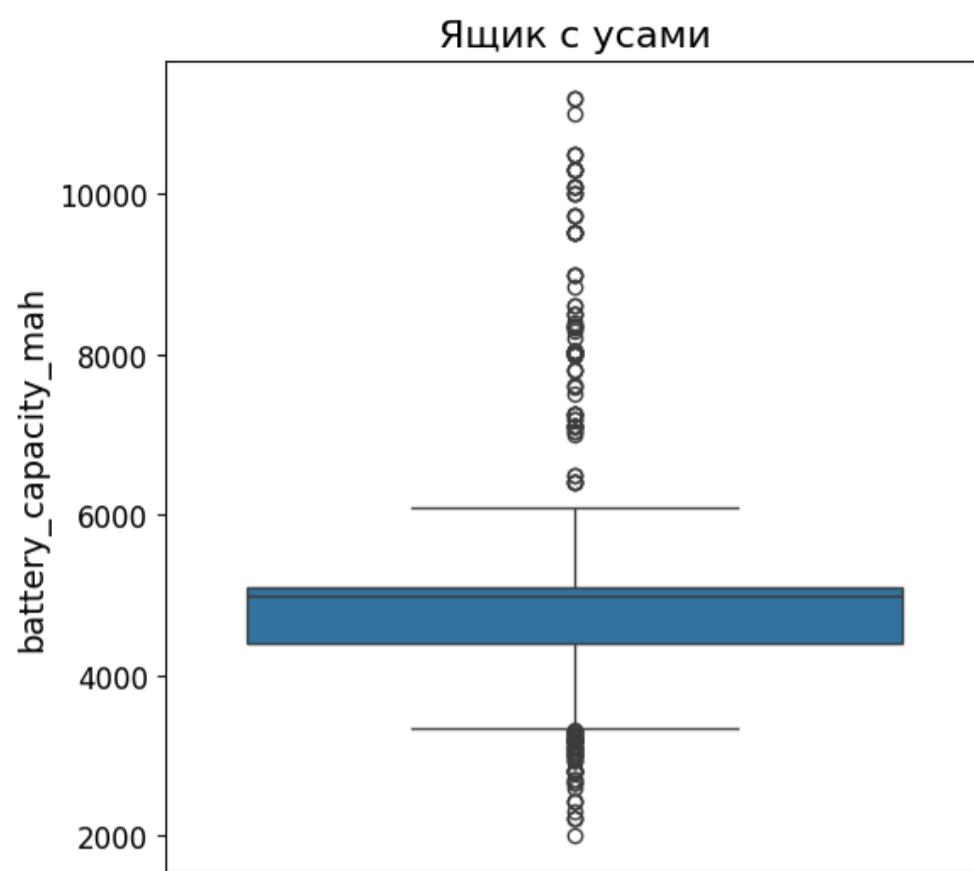
```
In [140]: plot_box_and_hist_num(data=df, feature='battery_capacity_mah',
                           title='Распределение размера аккумулятора мобильных устройств, мАч')
```

📊 Общая статистика по признаку 'battery_capacity_mah':

count	917.000000
mean	5030.122137
std	1364.100702
min	2000.000000
25%	4400.000000
50%	5000.000000
75%	5100.000000
max	11200.000000

Name: battery_capacity_mah, dtype: float64

Распределение размера аккумулятора мобильных устройств, мАч



Батарея

- Средняя емкость: ~5030 мА·ч, медиана: 5000 мА·ч.
- Есть модели с экстремальной емкостью до 11200 мА·ч (скорее всего планшеты).

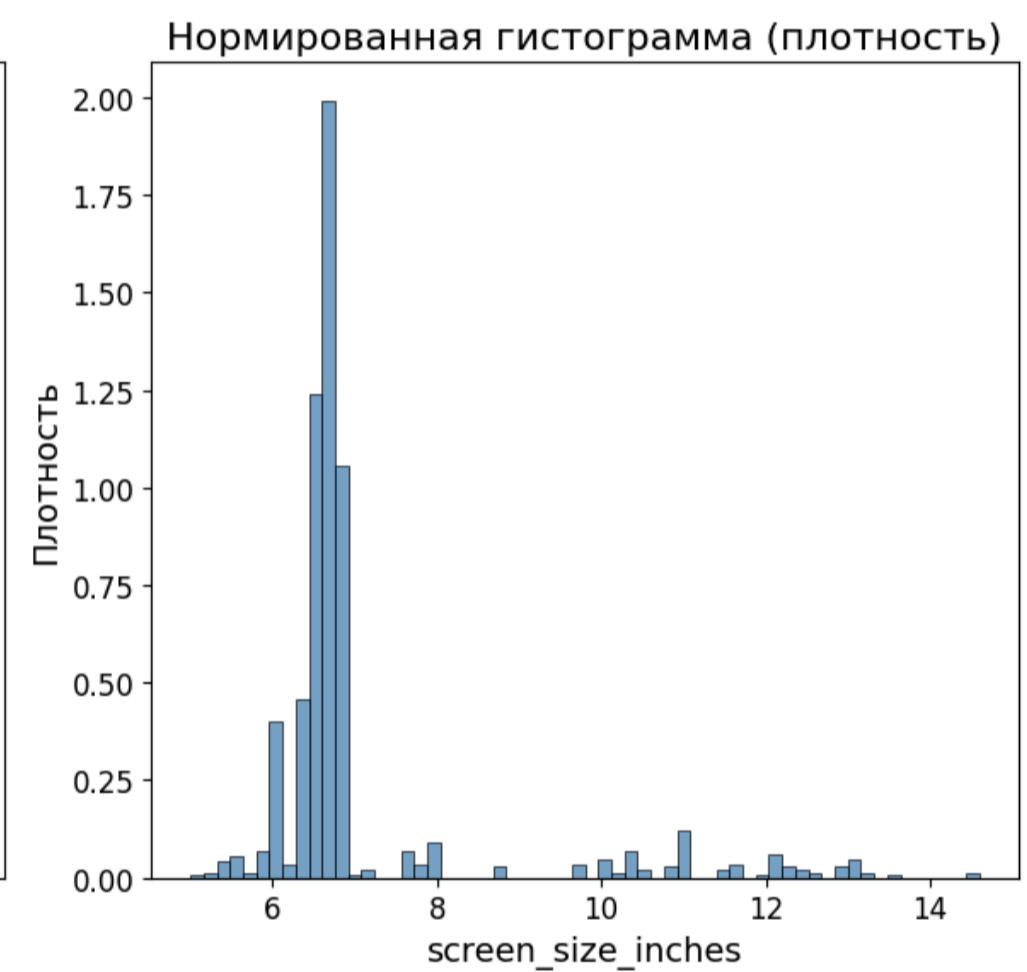
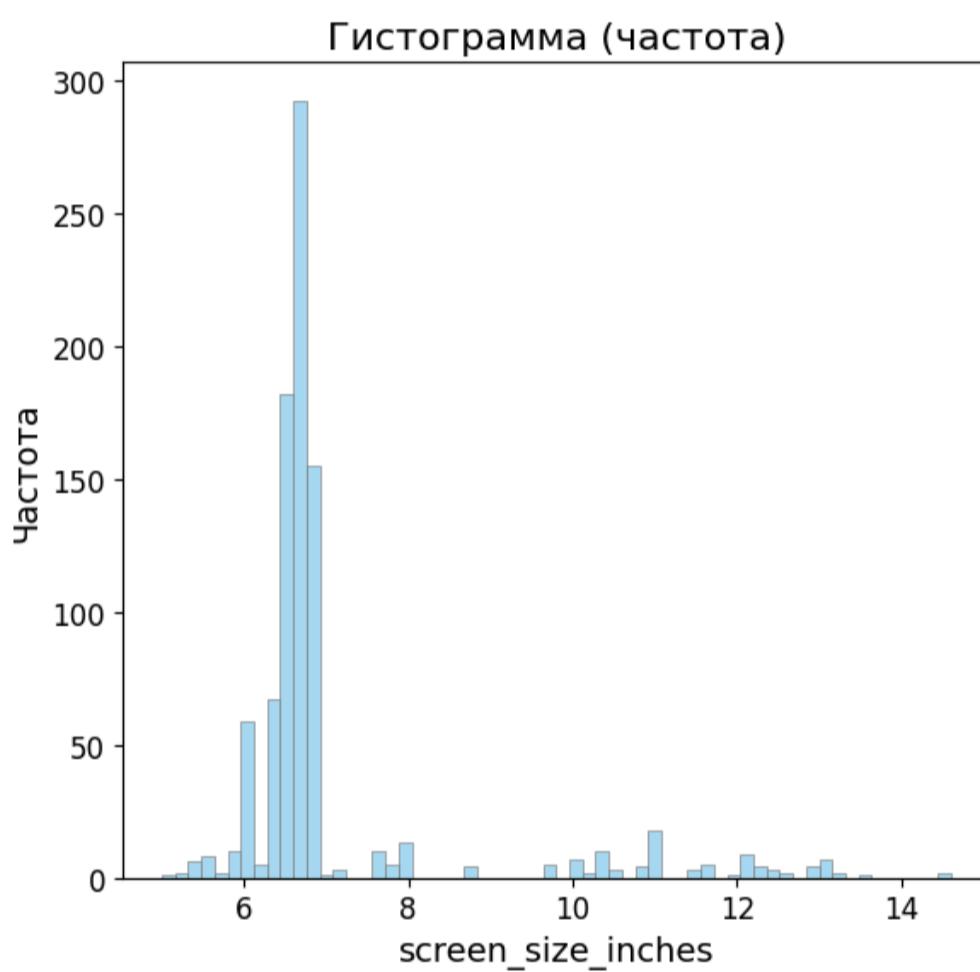
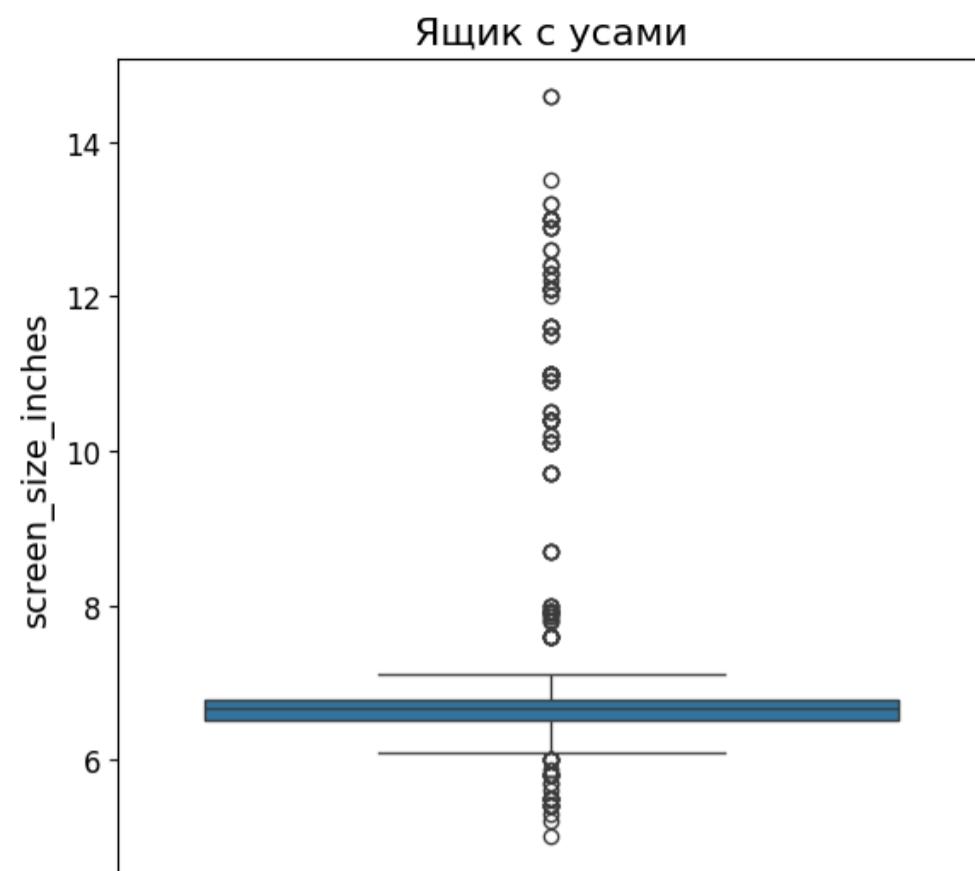
```
In [141]: plot_box_and_hist_num(data=df, feature='screen_size_inches',
                           title='Распределение диагонали дисплея мобильных устройств, дюймы')
```

Общая статистика по признаку 'screen_size_inches':

	screen_size_inches
count	917.000000
mean	7.091821
std	1.543000
min	5.000000
25%	6.500000
50%	6.670000
75%	6.780000
max	14.600000

Name: screen_size_inches, dtype: float64

Распределение диагонали дисплея мобильных устройств, дюймы



Диагональ экрана

- Средняя: ~7.1 дюйма, медиана: 6.67 дюйма.
- Но есть устройства до 14.6 дюймов (опять же планшеты).

Распределение имеет пик в районе 6.5–6.8 дюймов.

In [142]:

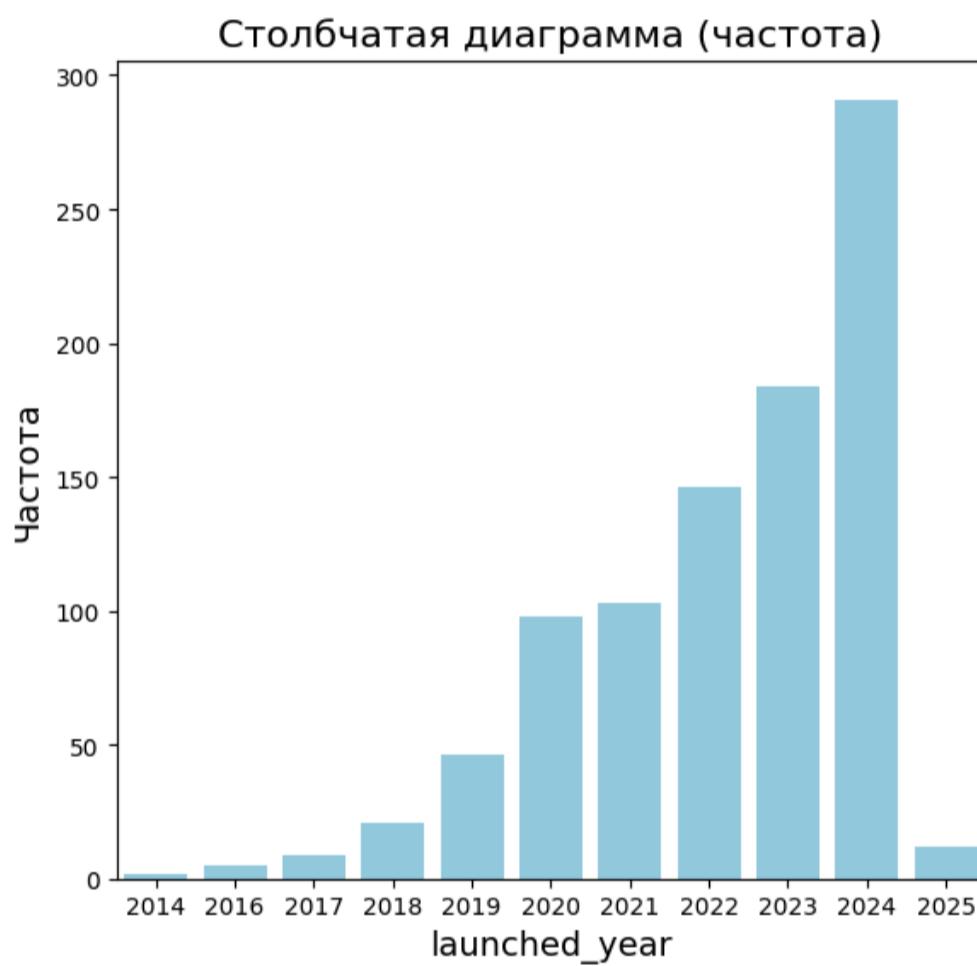
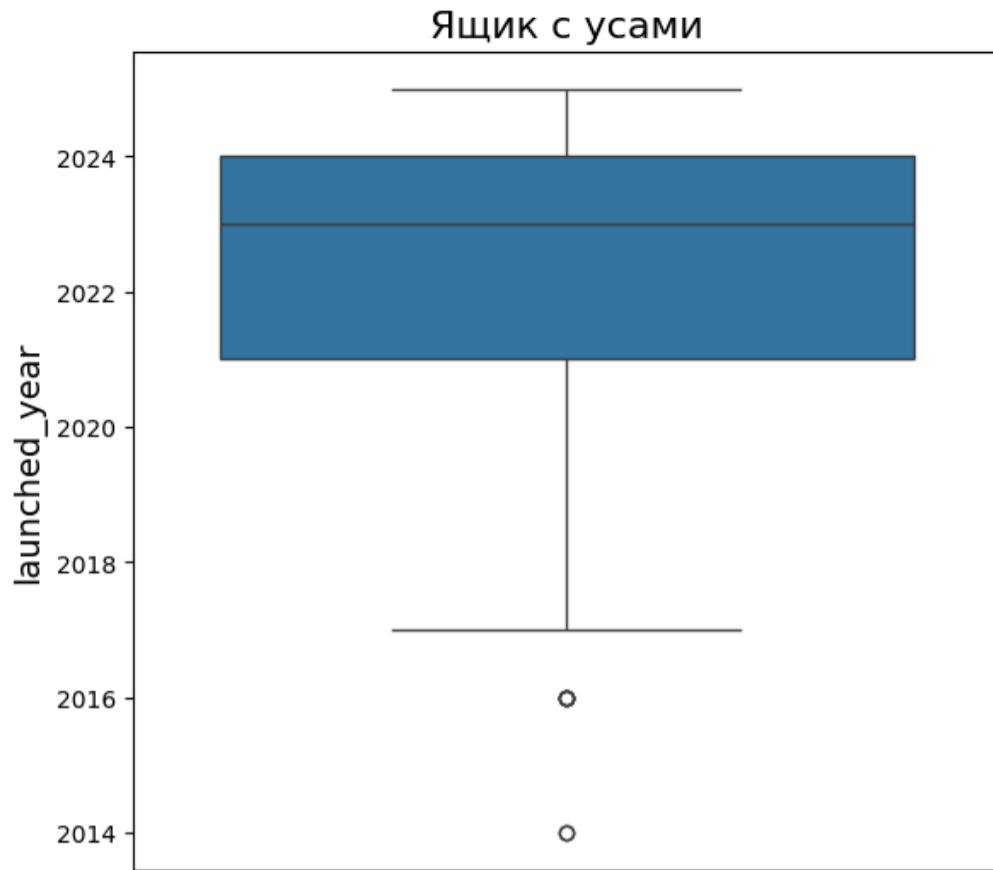
```
plot_box_and_bar_num(data=df, feature='launched_year',
                      title='Распределение года официального выпуска мобильного устройства')
```

Общая статистика по признаку 'launched_year':

count	917.000000
mean	2022.207197
std	1.867509
min	2014.000000
25%	2021.000000
50%	2023.000000
75%	2024.000000
max	2025.000000

Name: launched_year, dtype: float64

Распределение года официального выпуска мобильного устройства

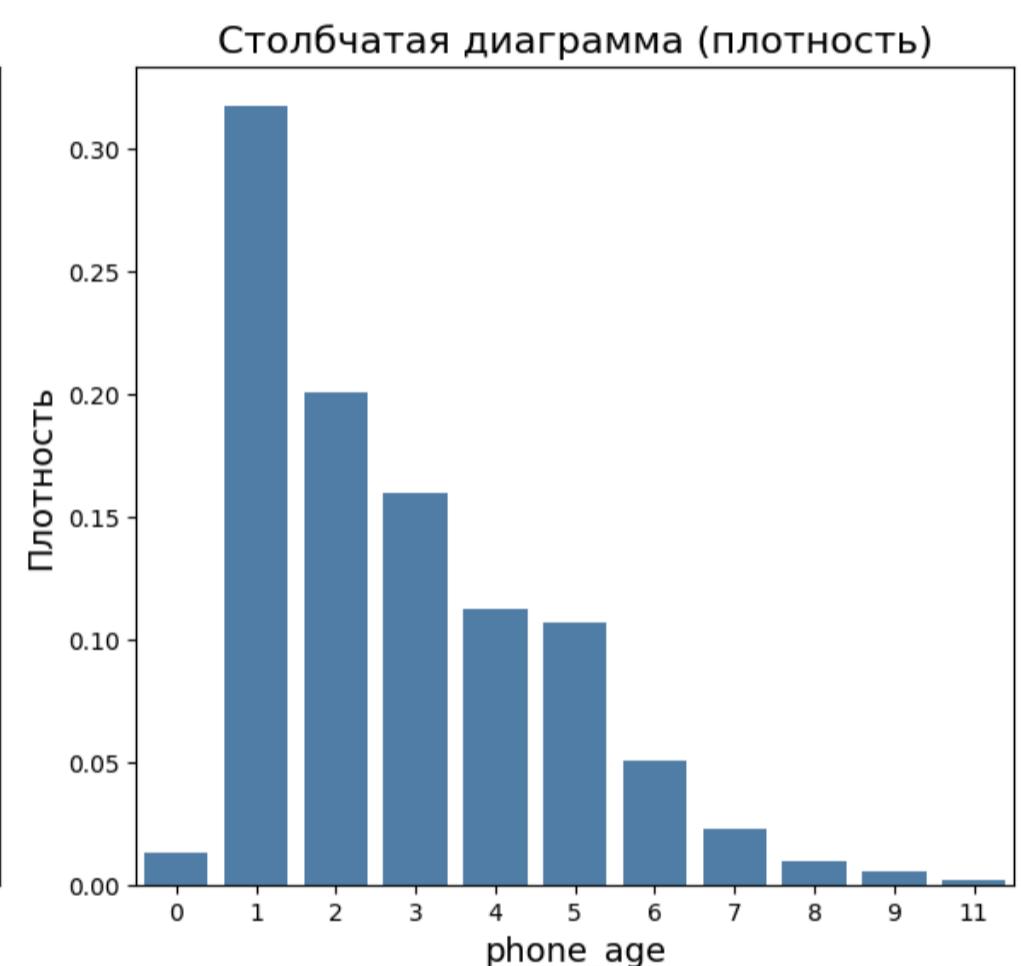
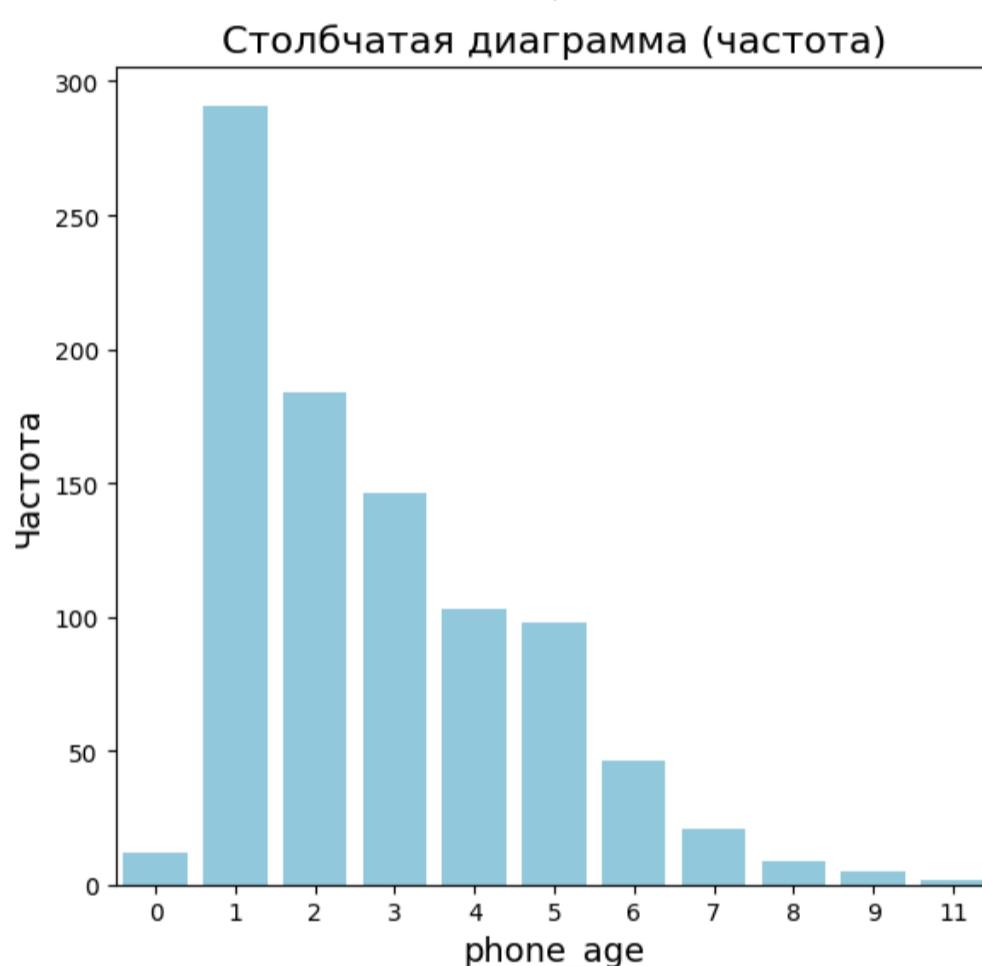
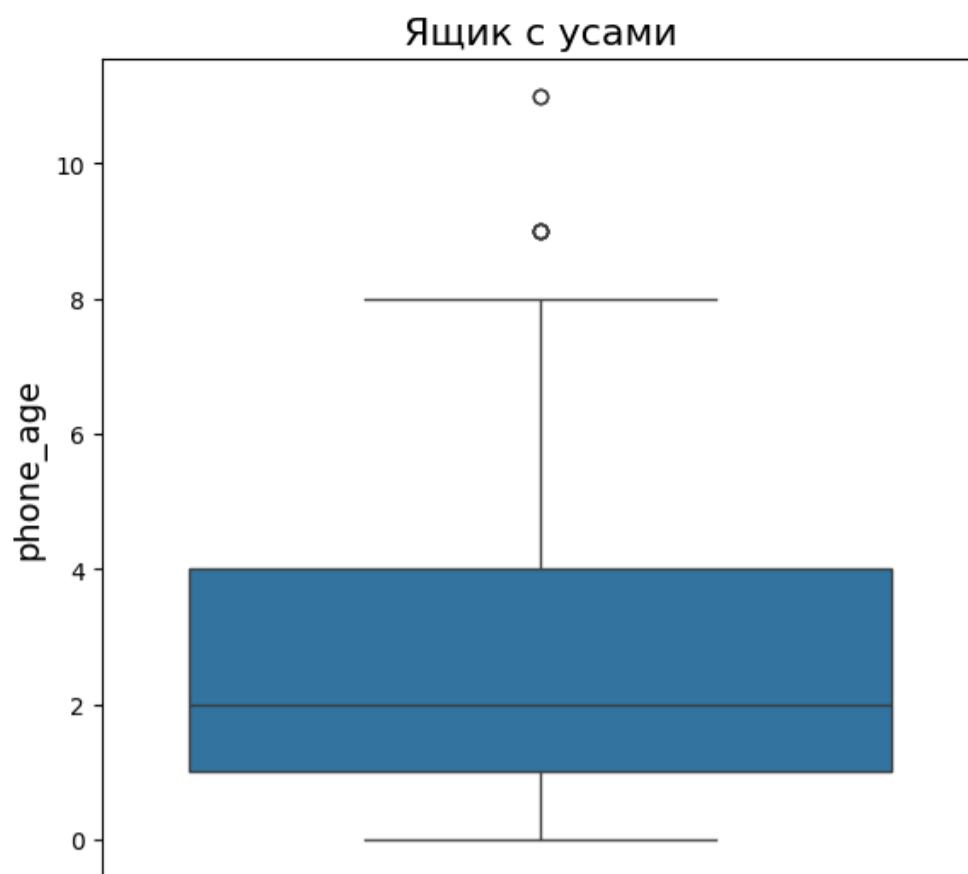


```
In [143]: plot_box_and_bar_num(data=df, feature='phone_age',  
                           title='Распределение возраста мобильного устройства')
```

📊 Общая статистика по признаку 'phone_age':

```
count    917.000000  
mean     2.792803  
std      1.867509  
min     0.000000  
25%    1.000000  
50%    2.000000  
75%    4.000000  
max    11.000000  
Name: phone_age, dtype: float64
```

Распределение возраста мобильного устройства



Возраст устройства (phone_age)

- Средний возраст: ~2.8 года.

Большинство устройств выпущены в 2021–2024 гг.

In [144]:

```
def plot_box_and_hist_num_multi(data, features=None, title=None, bins=60):
    if features is None:
        features = ['mobile_weight_g', 'battery_capacity_mah', 'screen_size_inch'] # пример

    df = data[features].dropna()

    # Общая статистика
    print("\n📊 Общая статистика по признакам:")
    print(df.describe().T)

    # Заголовок по умолчанию
    if title is None:
        title = 'Боксплот и гистограммы по числовым признакам'

    # Сетка: 3 строки x 1 столбец
    fig, axes = plt.subplots(3, 1, figsize=(12, 14))
    fig.suptitle(title, fontsize=20)

    # Boxplot – верхняя часть
    melted = df.melt(var_name='Признак', value_name='Значение')
    sns.boxplot(data=melted, x='Признак', y='Значение', ax=axes[0], palette='pastel')
    axes[0].set_title('Наложенные ящики с усами', fontsize=16)
    axes[0].set_xlabel('Признак', fontsize=14)
    axes[0].set_ylabel('Значение', fontsize=14)
    axes[0].tick_params(axis='x', labelsize=12)
    axes[0].tick_params(axis='y', labelsize=12)
```

```

# Гистограмма – частота
for feature in features:
    sns.histplot(data=df, x=feature, bins=bins, stat='count',
                 label=feature, alpha=0.5, ax=axes[1])
axes[1].set_title('Наложенные гистограммы (частота)', fontsize=16)
axes[1].set_xlabel('Значение', fontsize=14)
axes[1].set_ylabel('Частота', fontsize=14)
axes[1].legend(fontsize=12)
axes[1].tick_params(axis='x', labelsize=12)
axes[1].tick_params(axis='y', labelsize=12)

# Гистограмма – плотность
for feature in features:
    sns.histplot(data=df, x=feature, bins=bins, stat='density',
                 label=feature, alpha=0.5, ax=axes[2])
axes[2].set_title('Наложенные гистограммы (плотность)', fontsize=16)
axes[2].set_xlabel('Значение', fontsize=14)
axes[2].set_ylabel('Плотность', fontsize=14)
axes[2].legend(fontsize=12)
axes[2].tick_params(axis='x', labelsize=12)
axes[2].tick_params(axis='y', labelsize=12)

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

```

In [145...]: plot_box_and_hist_num_multi(df, features=['price_usa_usd', 'price_pakistan_usd', 'price_india_usd', 'price_china_usd', 'price_dubai_usd'], title='Распределение цен мобильных устройств в usd по странам ')

📊 Общая статистика по признакам:

	count	mean	std	min	25%	50%	75%	\
price_usa_usd	917.0	590.160174	422.676817	79.0	269.0	449.0	877.0	
price_pakistan_usd	917.0	452.416576	364.084443	57.0	196.0	321.0	641.0	
price_india_usd	917.0	578.430752	464.994265	68.0	226.0	407.0	847.0	
price_china_usd	917.0	542.279171	390.397027	70.0	252.0	421.0	786.0	
price_dubai_usd	917.0	600.272628	428.839118	81.0	272.0	463.0	898.0	

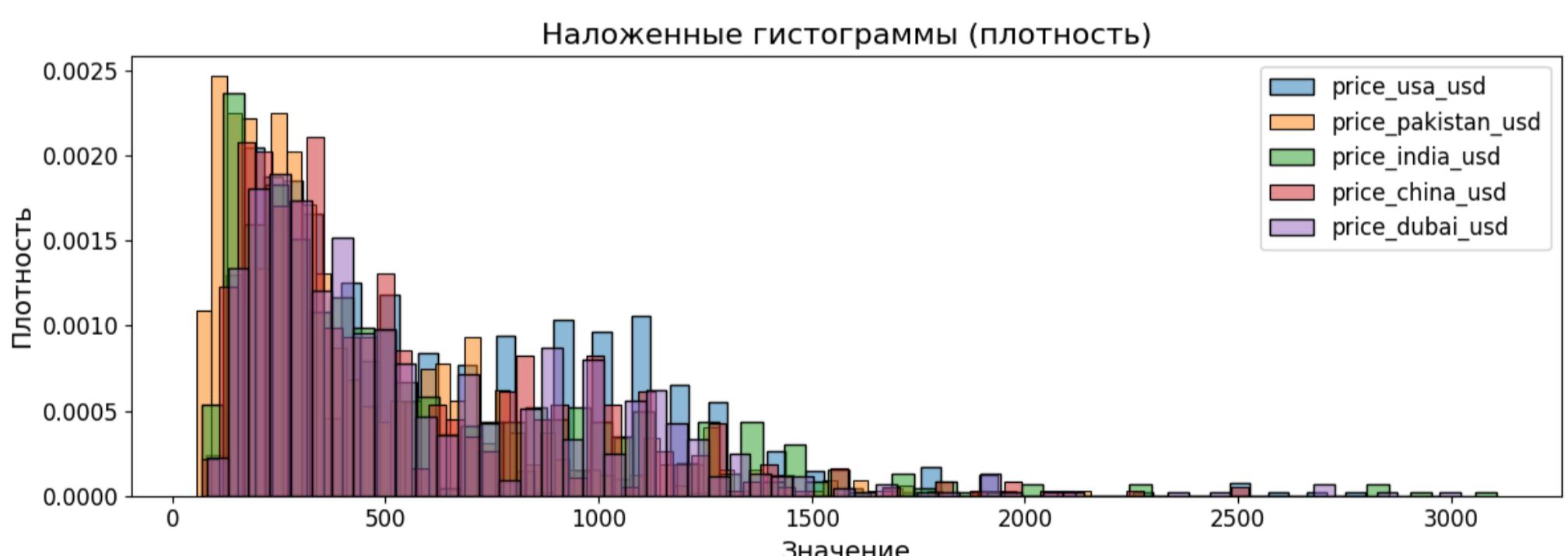
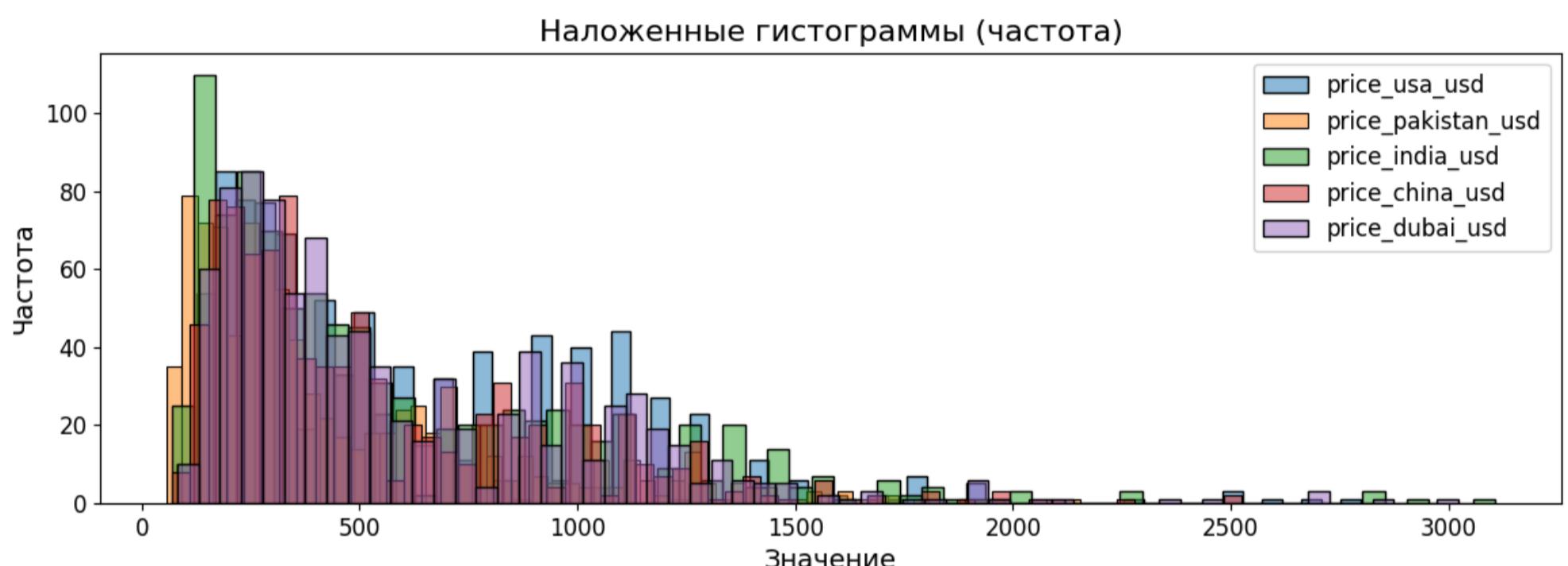
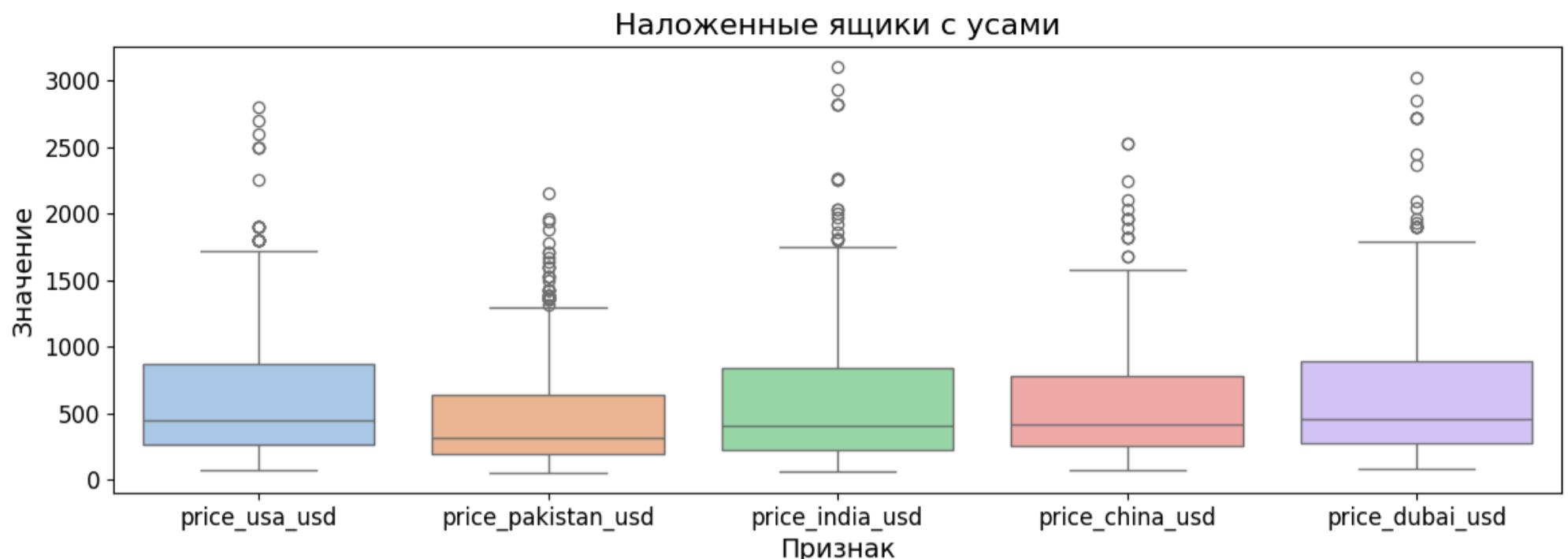
	max
price_usa_usd	2799.0
price_pakistan_usd	2156.0
price_india_usd	3107.0
price_china_usd	2526.0
price_dubai_usd	3022.0

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\851899567.py:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(data=melted, x='Признак', y='Значение', ax=axes[0], palette='pastel')
```

Распределение цен мобильных устройств в usd по странам



❖ Ключевые выводы:

- Пакистан — самый дешёвый рынок
- Dubai — самый дорогой
- Средняя разница между странами: ~\$150
- Разброс цен внутри стран огромен - от \$57 до \$3100

⚠ Важное наблюдение:

- Индия дороже Китая по средней цене (\$577 vs \$543), но дешевле по медиане (\$406 vs \$421). Это говорит о:
 - Более широком ассортименте дешёвых моделей в Индии
 - Наличии экстремально дорогих моделей (выбросы), которые поднимают среднюю

Страна	Средняя цена (USD)	Медианная цена (USD)	Ранг (дешевле → дороже)
Pakistan	451	320	1
China	543	421	2
India	577	406	3

Страна	Средняя цена (USD)	Медианная цена (USD)	Ранг (дешевле → дороже)
USA	590	449	4
Dubai	600	463	5

```
In [146]: plot_box_and_hist_num_multi(df, features=['affordability_usa', 'affordability_pakistan','affordability_india','affordability_china','affordability_uae'], title='Распределение метрики доступности: цена / медианная зарплата')
```

Общая статистика по признакам:

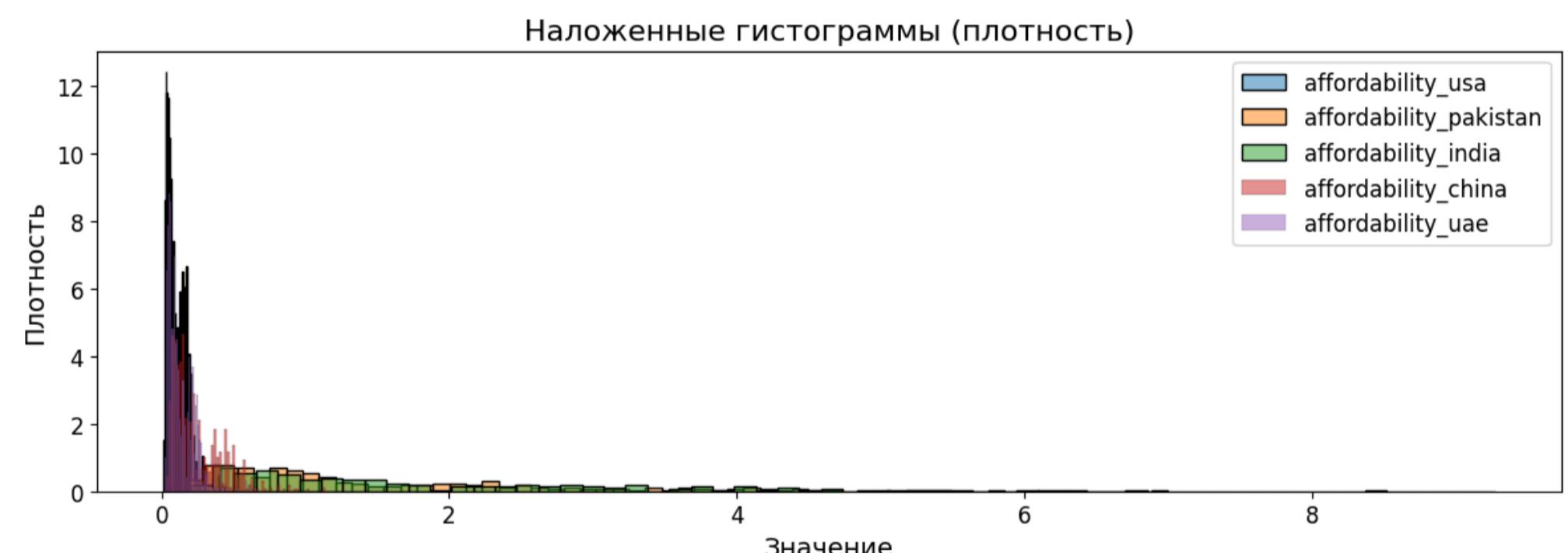
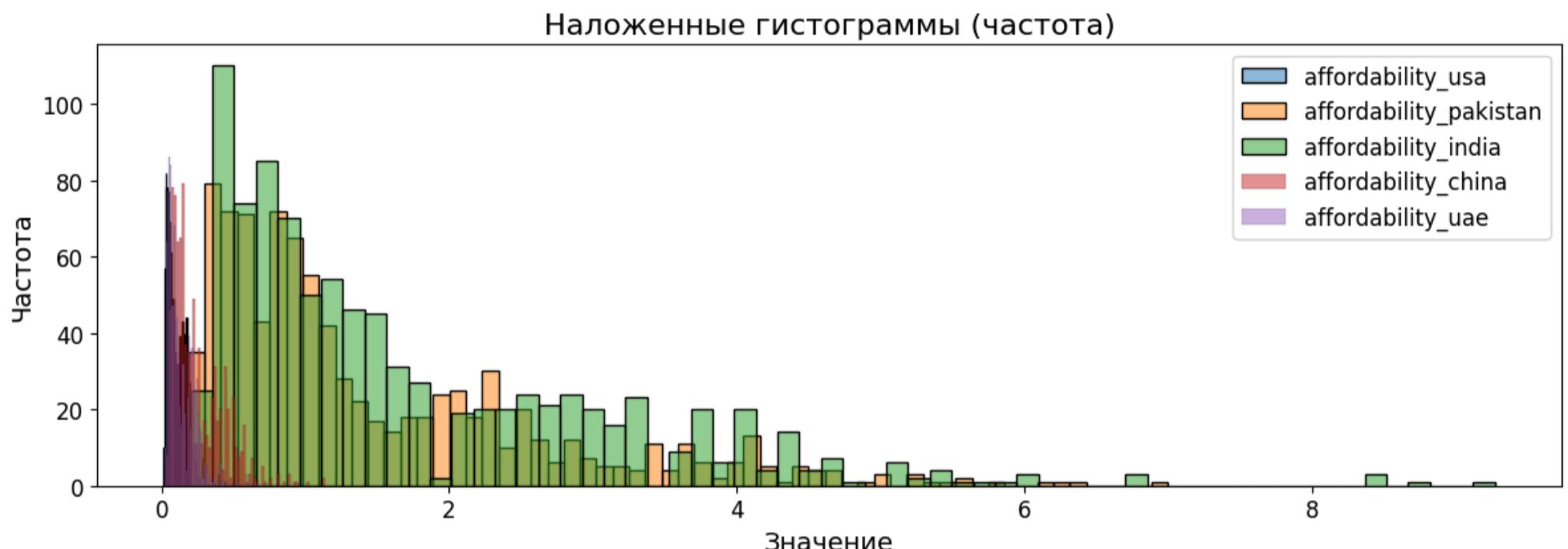
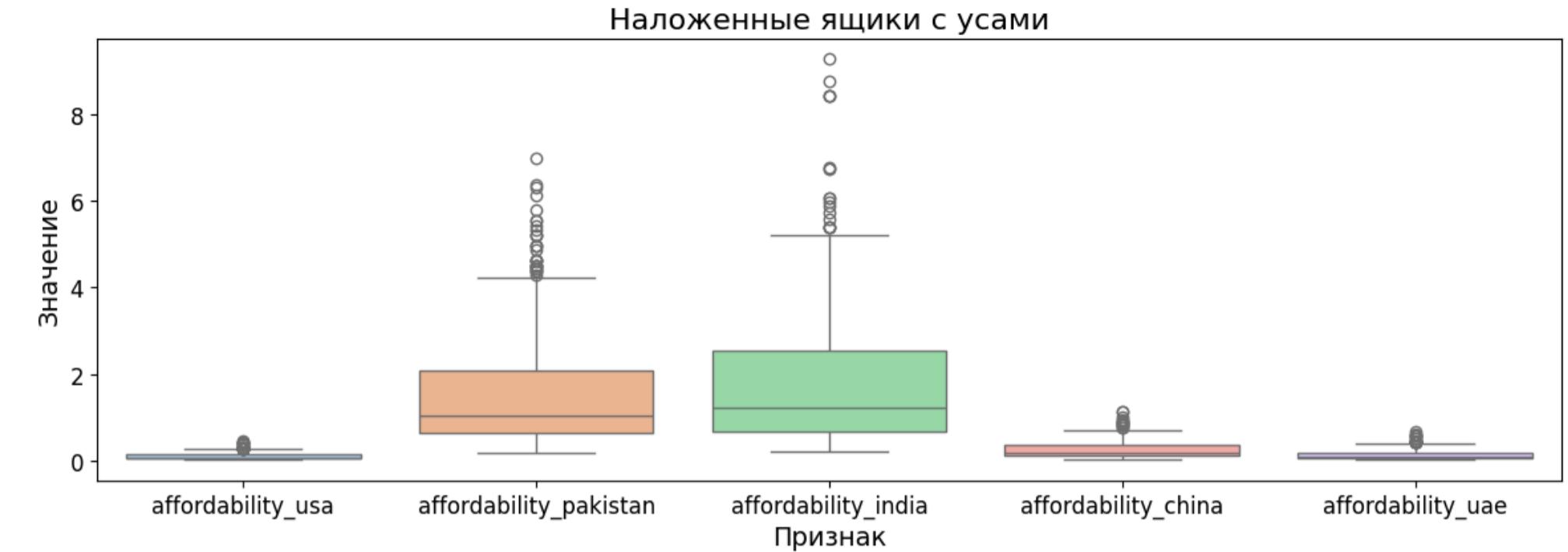
	count	mean	std	min	25%	50%	75%	\
affordability_usa	917.0	0.093878	0.067277	0.013	0.043	0.071	0.140	
affordability_pakistan	917.0	1.468836	1.182150	0.185	0.636	1.042	2.081	
affordability_india	917.0	1.726642	1.388058	0.203	0.675	1.215	2.528	
affordability_china	917.0	0.245074	0.176351	0.032	0.114	0.190	0.355	
affordability_uae	917.0	0.130438	0.093228	0.018	0.059	0.101	0.195	
								max
affordability_usa		0.445						
affordability_pakistan		7.000						
affordability_india		9.275						
affordability_china		1.141						
affordability_uae		0.657						

```
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\851899567.py:21: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(data=melted, x='Признак', y='Значение', ax=axes[0], palette='pastel')
```

Распределение метрики доступности: цена / медианная зарплата



Страна	Среднее	Медиана	Интерпретация
USA	0.09	0.07	Очень доступно
UAE	0.13	0.10	Доступно
China	0.25	0.19	Умеренно доступно
Pakistan	1.47	1.04	Недоступно
India	1.72	1.21	Критически недоступно

Смартфоны наиболее доступны в США и ОАЭ благодаря высоким зарплатам.

- медиана <0.15 зарплаты)

Китай — средняя доступность (~0.19)

Наименее доступны — в Индии и Пакистане

- медиана ~1–1.2 зарплаты
- выбросы до 7–9 зарплат

❖ Критический инсайт:

- Парадокс цены и доступности: Хотя телефоны в Пакистане дешевле в абсолютных цифрах (\$451), они менее доступны из-за низких зарплат.
- Индиец тратит в 18 раз больше месячных зарплат, чем американец, на тот же смартфон!

In [147...]

```
def plot_income_barplot(data, features):
    df = data[features].dropna()

    # Вычисляем медианные значения
    medians = df.median().sort_values()

    # Создаём график
    plt.figure(figsize=(15, 10))
    sns.barplot(x=medians.index, y=medians.values, palette='Set2')

    # Добавляем подписи над столбцами
    for i, value in enumerate(medians.values):
        plt.text(i, value + value * 0.02, f'{value:.0f}', ha='center', va='bottom', fontsize=14)

    plt.title('Медианный доход по странам (USD)', fontsize=16)
    plt.xlabel('Страна', fontsize=14)
    plt.ylabel('Доход (USD)', fontsize=14)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

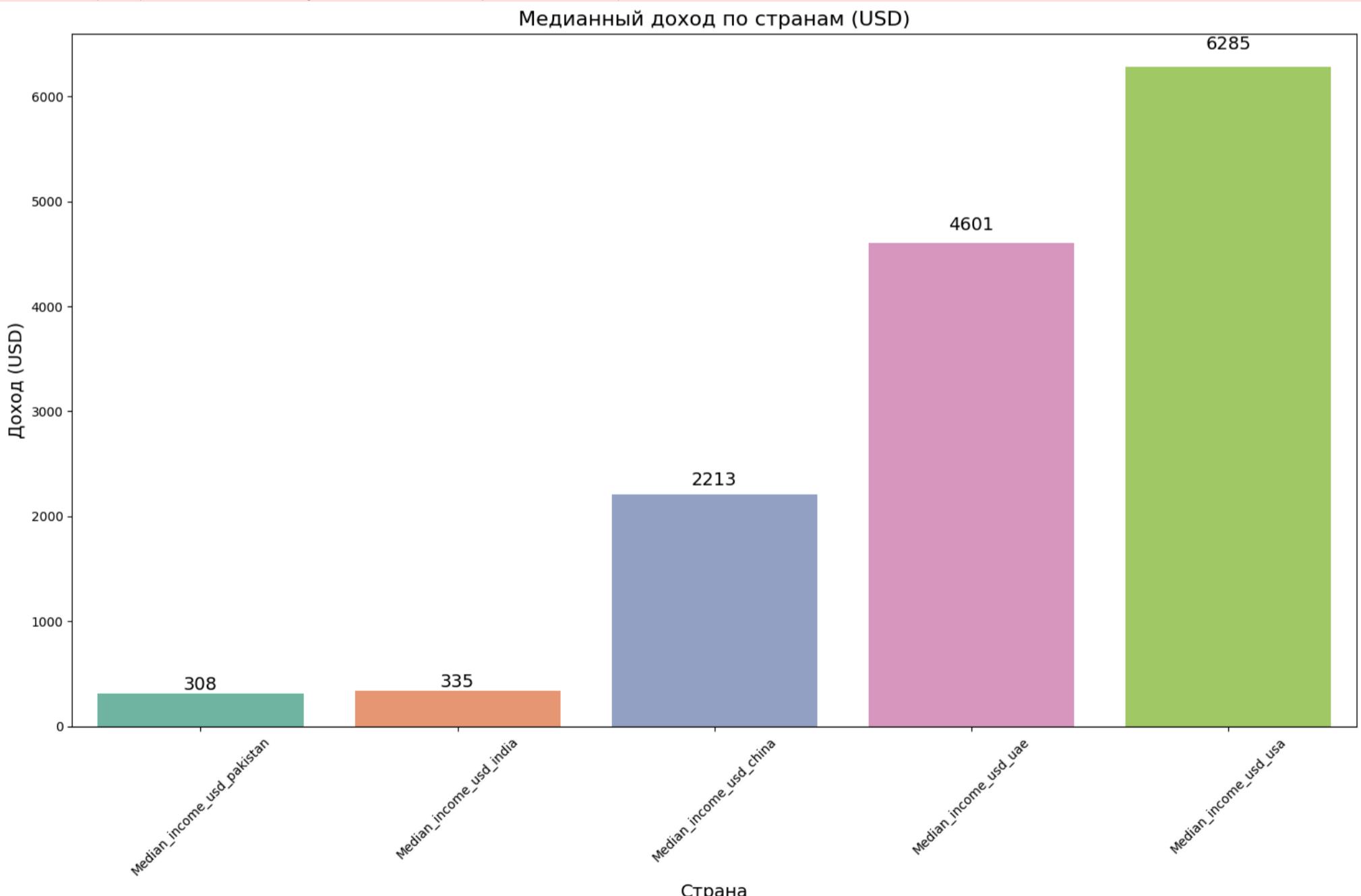
In [148...]

```
features = [
    'Median_income_usd_pakistan',
    'Median_income_usd_india',
    'Median_income_usd_china',
    'Median_income_usd_uae',
    'Median_income_usd_usa'
]
plot_income_barplot(df, features)
```

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\918228750.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x=medians.index, y=medians.values, palette='Set2')



Категориальные переменные

In [149...]

```
df.select_dtypes(include=['object', 'category']).describe().T
```

Out[149...]

	count	unique	top	freq
company_name	917	18	Oppo	115
model_name	917	902	iPad Pro 512GB	3
weight_category	917	4	средний	656
ram_category	917	3	Средний	505
processor	917	217	Snapdragon 8 Gen 2	30
battery_category	917	4	средняя	663
screen_category	917	4	стандартный	754
price_category	917	4	Средний	327

In [150...]

```
def plot_cat_feature_distribution(data, feature='company_name', title=None):
    # Удаляем пропуски
    df = data[[feature]].dropna()

    # Общие частоты значений признака
    print(f"\nОбщее распределение признака '{feature}':")
    counts = df[feature].value_counts()
    percentages = df[feature].value_counts(normalize=True) * 100
    summary_table = pd.DataFrame({'Частота': counts, 'Процент': percentages.round(2)})
    print(summary_table)

    # Заголовок графика по умолчанию
    if title is None:
        title = f'Распределение категориального признака "{feature}"'

    # График
    plt.figure(figsize=(12, 6))
    sns.countplot(data=df, x=feature, palette='pastel')
    plt.title(title, fontsize=18)
    plt.xlabel(feature, fontsize=14)
    plt.ylabel('Частота', fontsize=14)
    plt.xticks(rotation=45)
    plt.tick_params(axis='x', labelsize=12)
    plt.tick_params(axis='y', labelsize=12)
    plt.tight_layout()
    plt.show()
```

In [151...]

```
plot_cat_feature_distribution(df, feature='company_name',
                               title='Распределение компаний по количеству моделей')
```

Общее распределение признака 'company_name':

	Частота	Процент
Oppo	115	12.54
Apple	97	10.58
Honor	91	9.92
Samsung	88	9.60
Vivo	86	9.38
Realme	69	7.52
Motorola	62	6.76
Infinix	55	6.00
OnePlus	53	5.78
Huawei	44	4.80
Tecno	39	4.25
Poco	32	3.49
Xiaomi	27	2.94
Google	21	2.29
Lenovo	15	1.64
Nokia	11	1.20
Sony	9	0.98
iQOO	3	0.33

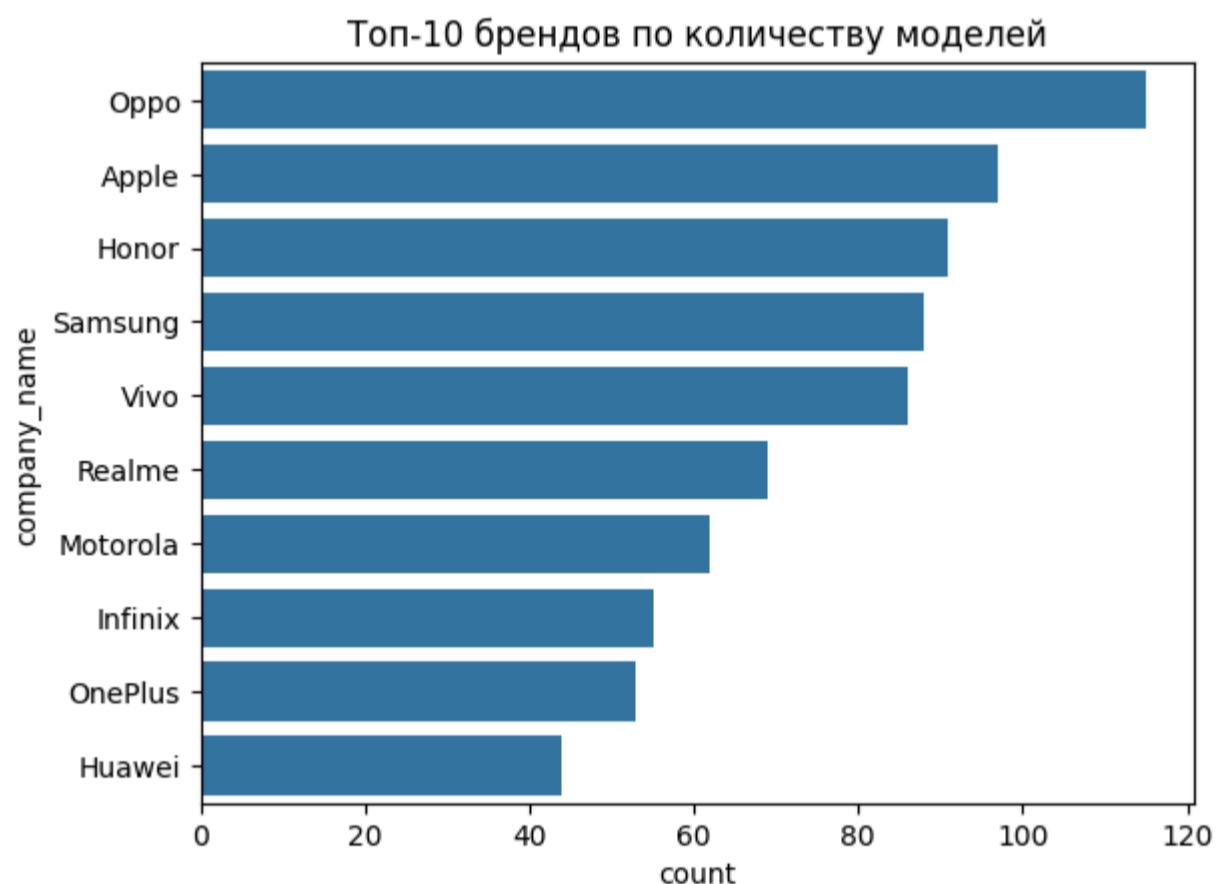
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2278685115.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x=feature, palette='pastel')
```

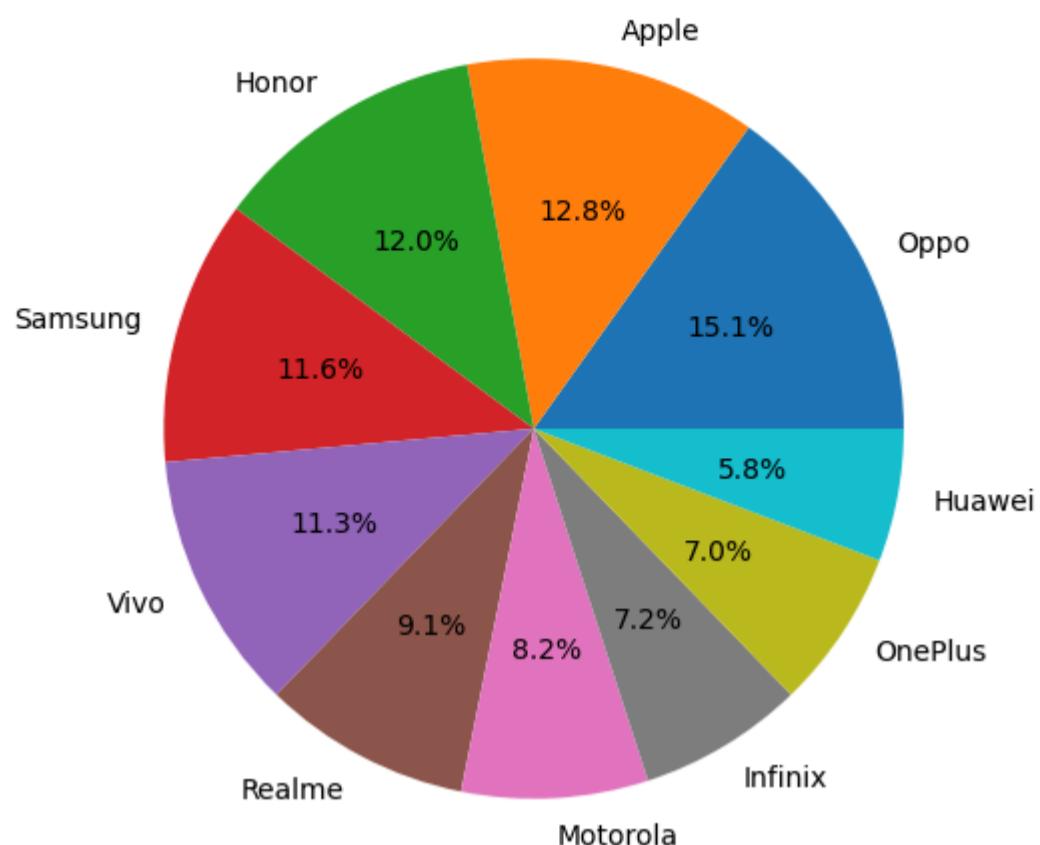


```
In [152...]: sns.countplot(data=df, y='company_name', order=df['company_name'].value_counts().head(10).index)
plt.title("Топ-10 брендов по количеству моделей")
plt.show()
```



```
In [153...]: df['company_name'].value_counts().head(10).plot.pie(autopct='%1.1f%%', figsize=(6, 6))
plt.title("Доля брендов")
plt.ylabel('')
plt.show()
```

Доля брендов



```
In [154...]: plot_cat_feature_distribution(df, feature='weight_category',
                                         title='Распределение мобильных устройств по весу')
```

Общее распределение признака 'weight_category':

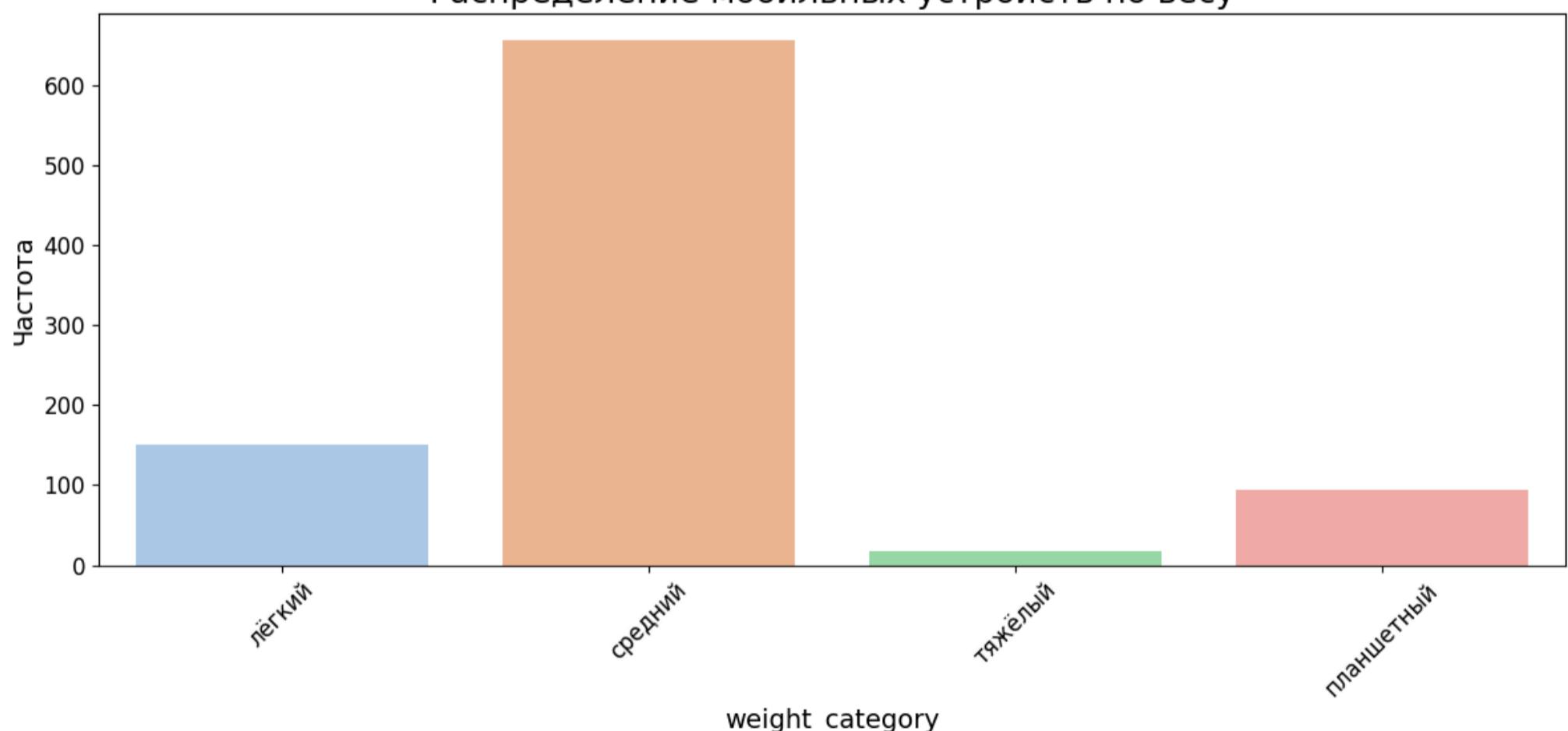
Частота	Процент
средний	656
лёгкий	150
планшетный	94
тяжёлый	17

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2278685115.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x=feature, palette='pastel')
```

Распределение мобильных устройств по весу



```
In [155...]: plot_cat_feature_distribution(df, feature='ram_category',
                                         title='Распределение мобильных устройств по RAM')
```

Общее распределение признака 'ram_category':

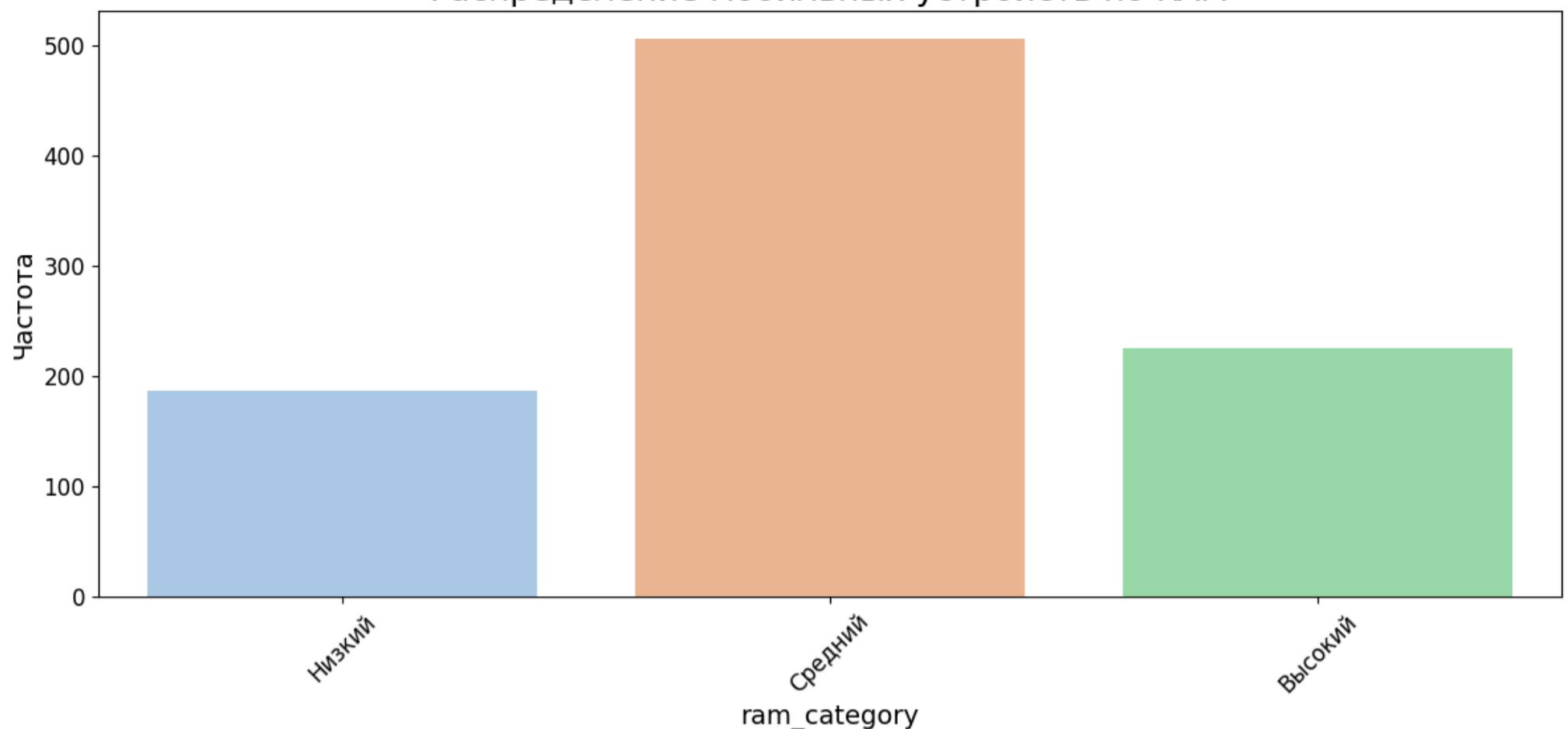
Частота	Процент
Средний	505
Высокий	225
Низкий	187

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2278685115.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x=feature, palette='pastel')
```

Распределение мобильных устройств по RAM



```
In [156]: plot_cat_feature_distribution(df, feature='battery_category',  
                                     title='Распределение мобильных устройств по мощности аккумулятора')
```

Общее распределение признака 'battery_category':

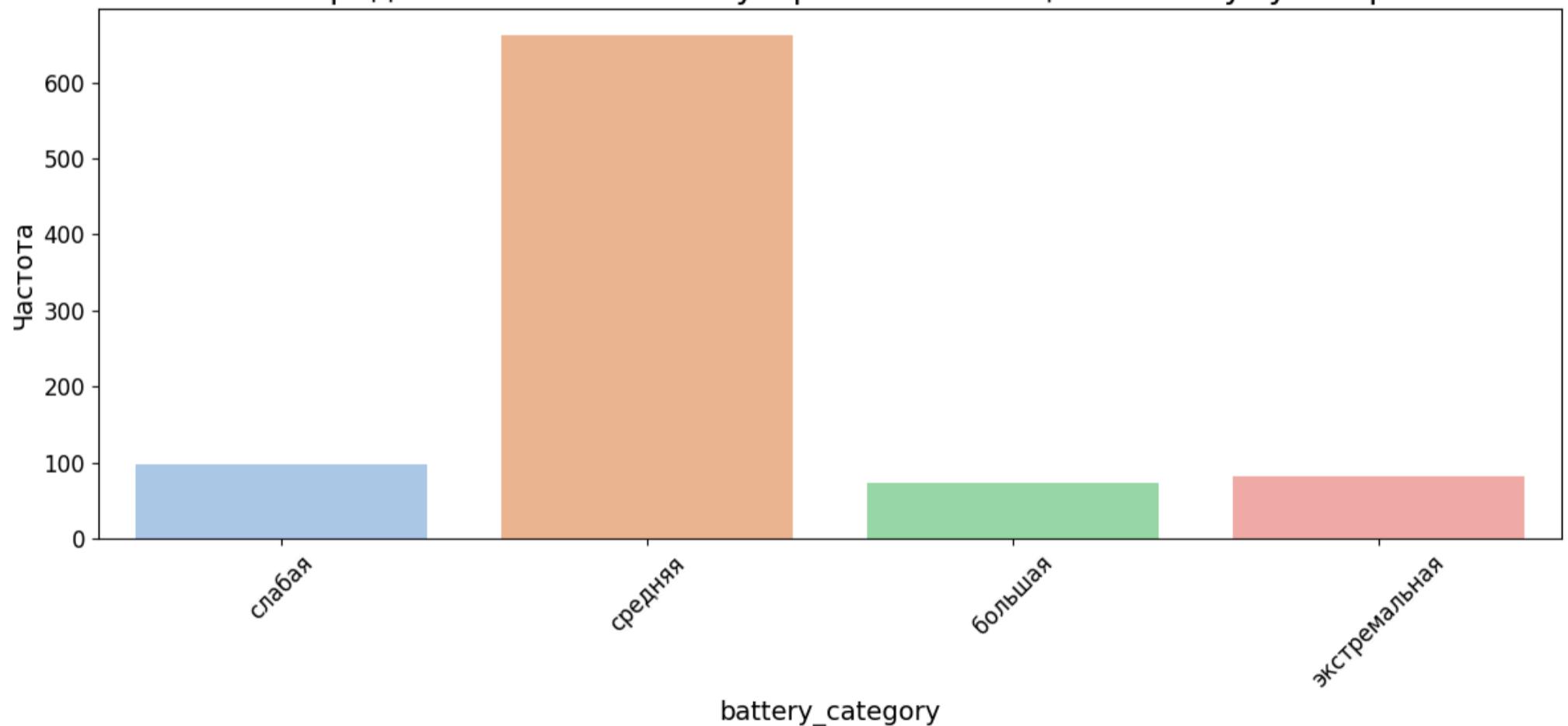
	Частота	Процент
средняя	663	72.30
слабая	98	10.69
экстремальная	82	8.94
большая	74	8.07

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2278685115.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x=feature, palette='pastel')
```

Распределение мобильных устройств по мощности аккумулятора



```
In [157]: plot_cat_feature_distribution(df, feature='price_category',  
                                     title='Распределение мобильных устройств по цене в usd')
```

Общее распределение признака 'price_category':

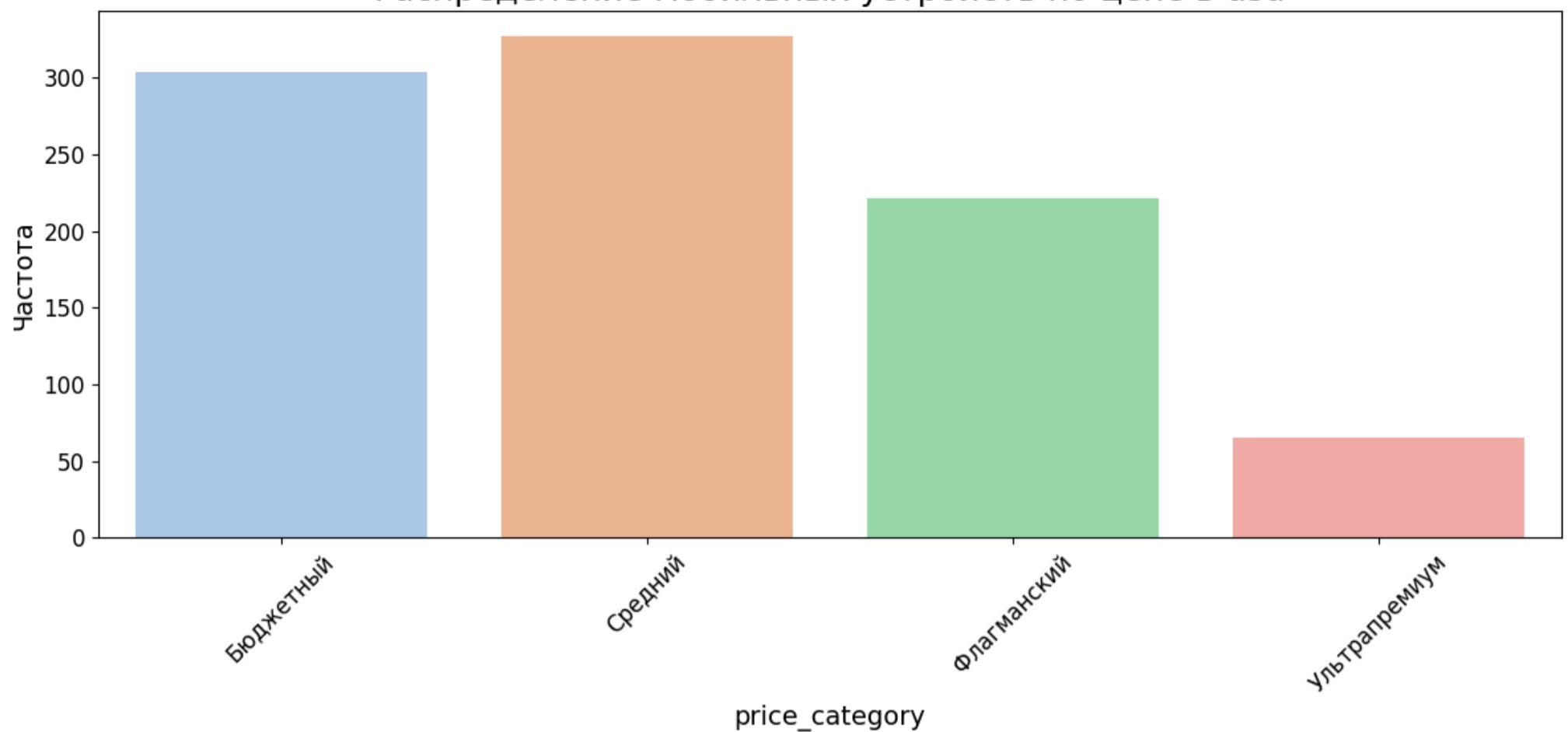
	Частота	Процент
Средний	327	35.66
Бюджетный	304	33.15
Флагманский	221	24.10
Ультрапремиум	65	7.09

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2278685115.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x=feature, palette='pastel')
```

Распределение мобильных устройств по цене в usd



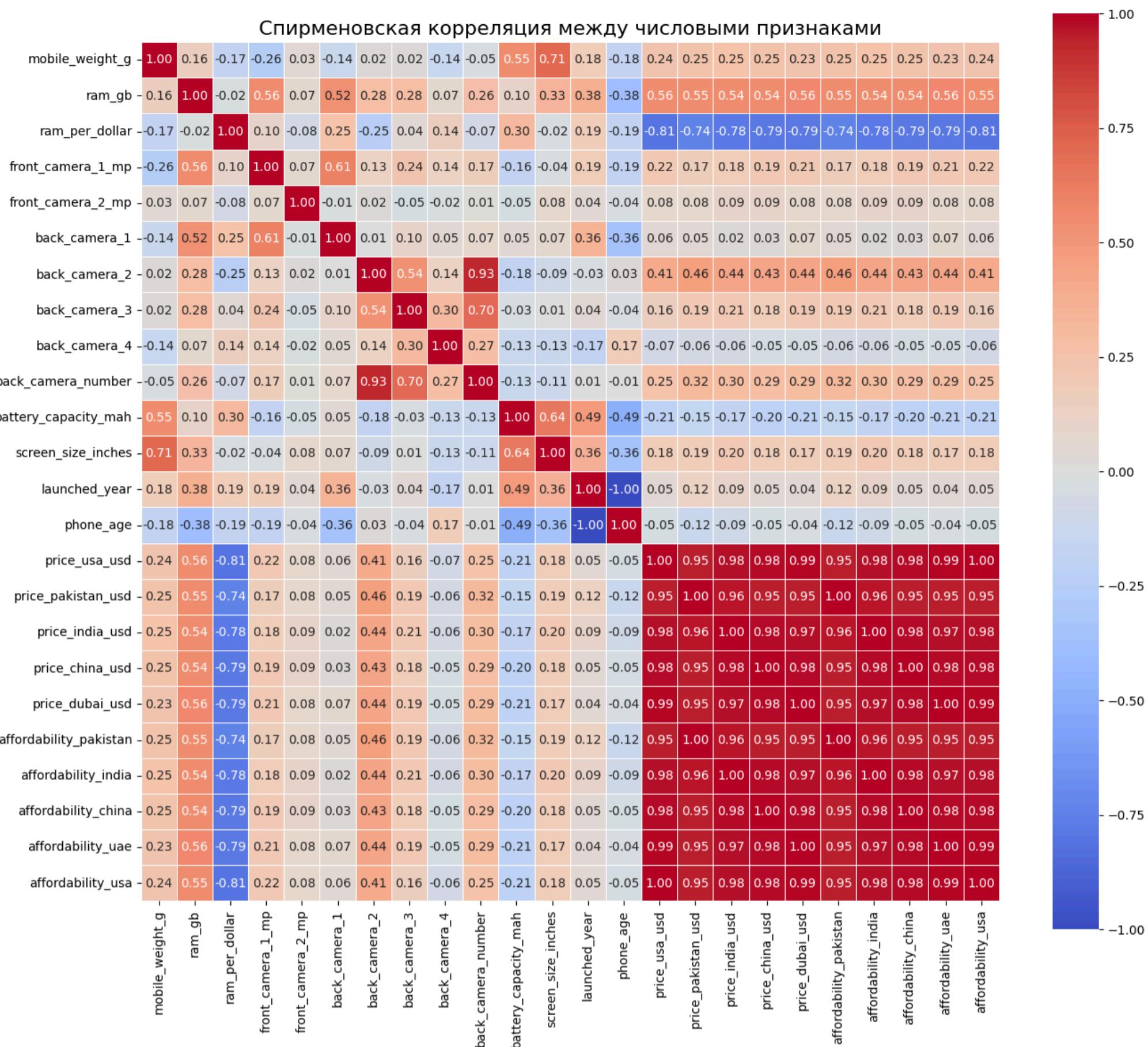
Категориальные переменные

- ram_category, price_category, battery_category, screen_category позволяют сегментировать рынок.
- Уже видно, что большинство моделей относятся к среднему и флагманскому сегменту.

Корреляционный анализ

Количественные переменные

```
In [158...]  
cols_to_drop = ['front_camera_number', 'Median_income_usd_pakistan', \  
                 'Median_income_usd_india', 'Median_income_usd_china', 'Median_income_usd_uae', 'Median_income_usd_usa']  
df_corr=df.copy()  
df_corr = df_corr.drop(columns=cols_to_drop)  
  
# Выбираем числовые признаки  
numerical_cols = df_corr.select_dtypes(include=['int64', 'float64', 'int32']).columns.tolist()  
  
# Матрица корреляций Спирмена  
spearman_corr = df_corr[numerical_cols].corr(method='spearman')  
  
# Визуализация  
plt.figure(figsize=(14, 12))  
sns.heatmap(spearman_corr, annot=True, fmt=".2f", cmap='coolwarm', center=0,  
            square=True, linewidths=0.5)  
plt.title('Спирменовская корреляция между числовыми признаками', fontsize=16)  
plt.tight_layout()  
plt.show()
```



In [159]:

```
high_corr_mask = np.triu(np.abs(spearman_corr) > 0.8, k=1)
print("Признаки с корреляцией > 0.8:")
print(spearman_corr.where(high_corr_mask).stack().dropna().sort_values(ascending=False))
```

Признаки с корреляцией > 0.8:

price_india_usd	affordability_india	1.000000
price_pakistan_usd	affordability_pakistan	1.000000
price_dubai_usd	affordability_uae	0.999991
price_china_usd	affordability_china	0.999980
price_usa_usd	affordability_usa	0.999933
affordability_uae	affordability_usa	0.990388
price_dubai_usd	affordability_usa	0.990249
price_usa_usd	affordability_uae	0.990245
	price_dubai_usd	0.990106
price_india_usd	price_china_usd	0.982070
price_china_usd	affordability_india	0.982070
price_india_usd	affordability_china	0.982005
affordability_india	affordability_china	0.982005
price_usa_usd	affordability_china	0.980442
	price_china_usd	0.980421
affordability_china	affordability_usa	0.980026
price_china_usd	affordability_usa	0.979993
affordability_china	affordability_uae	0.979734
price_dubai_usd	affordability_china	0.979671
price_china_usd	affordability_uae	0.979620
	price_dubai_usd	0.979557
price_usa_usd	price_india_usd	0.976144
	affordability_india	0.976144
affordability_india	affordability_usa	0.975555
price_india_usd	affordability_usa	0.975555
affordability_india	affordability_uae	0.973511
price_india_usd	affordability_uae	0.973511
price_dubai_usd	affordability_india	0.973446
price_india_usd	price_dubai_usd	0.973446
	affordability_pakistan	0.962664
price_pakistan_usd	price_india_usd	0.962664
	affordability_india	0.962664
affordability_pakistan	affordability_india	0.962664
price_usa_usd	affordability_pakistan	0.950275
	price_pakistan_usd	0.950275
affordability_pakistan	affordability_uae	0.950254
price_pakistan_usd	affordability_uae	0.950254
	price_dubai_usd	0.950224
price_dubai_usd	affordability_pakistan	0.950224
price_pakistan_usd	affordability_usa	0.949555
affordability_pakistan	affordability_usa	0.949555
price_pakistan_usd	price_china_usd	0.947303
price_china_usd	affordability_pakistan	0.947303
affordability_pakistan	affordability_china	0.947160
price_pakistan_usd	affordability_china	0.947160
back_camera_2	back_camera_number	0.930995
ram_per_dollar	price_usa_usd	-0.810364
	affordability_usa	-0.811235
launched_year	phone_age	-1.000000

dtype: float64

Ключевые зависимости:

- RAM и цена: умеренная положительная корреляция (~0.56).
- Вес и диагональ экрана: сильная корреляция (~0.71).
- Цены между странами: очень высокая корреляция (>0.95).

Доступность и цена: сильная обратная связь (чем дороже — тем менее доступно).

Умеренные корреляции (0.4-0.7)

- RAM ↔ Цена: 0.54-0.56 (RAM - значимый драйвер цены)
- Основная камера ↔ Цена: 0.36-0.41
- Диагональ ↔ Батарея: 0.64

Слабые корреляции (<0.3)

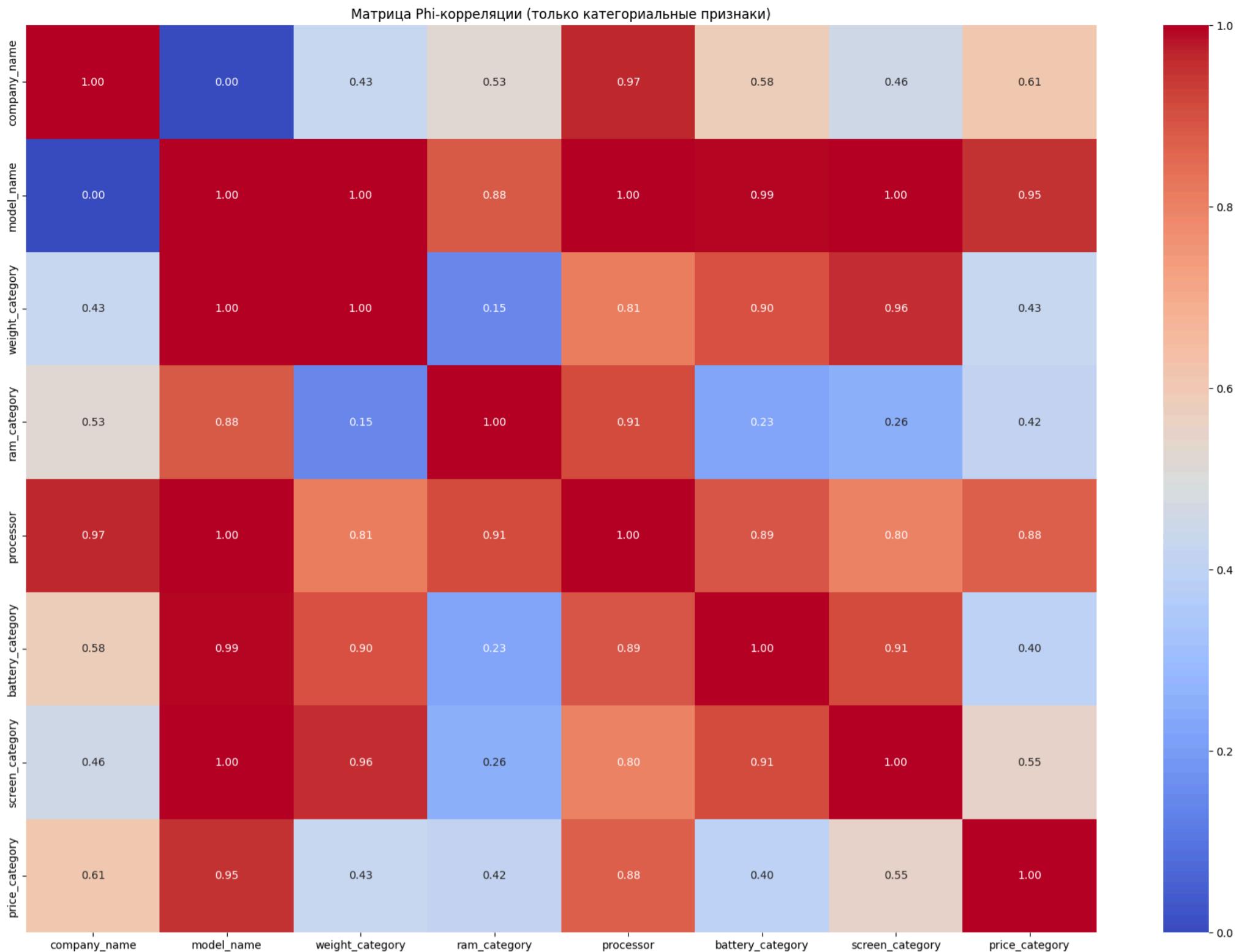
- Фронтальные камеры слабо влияют на цену
- Количество камерных модулей имеет ограниченное влияние

Категориальные переменные

```
In [160...]: # Выбираем только категориальные признаки
categorical_cols = df_corr.select_dtypes(include=['object', 'category']).columns.tolist()

# Строим Phi-матрицу только для категориальных
phik_matrix_cat = df_corr[categorical_cols].phik_matrix()

# Визуализация
plt.figure(figsize=(22, 15))
sns.heatmap(phik_matrix_cat, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Матрица Phi-корреляции (только категориальные признаки)")
plt.show()
```

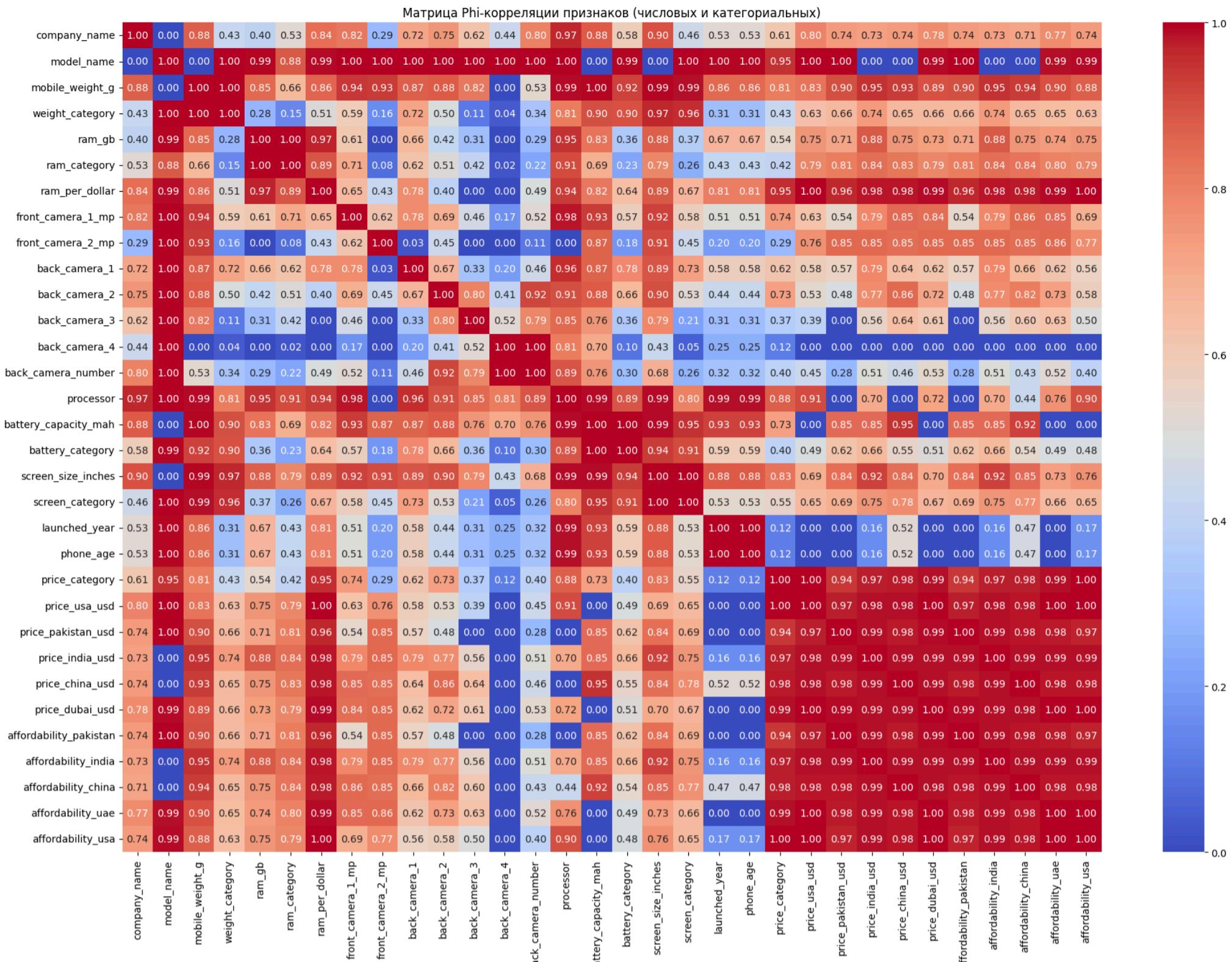


Количественные+категориальные переменные

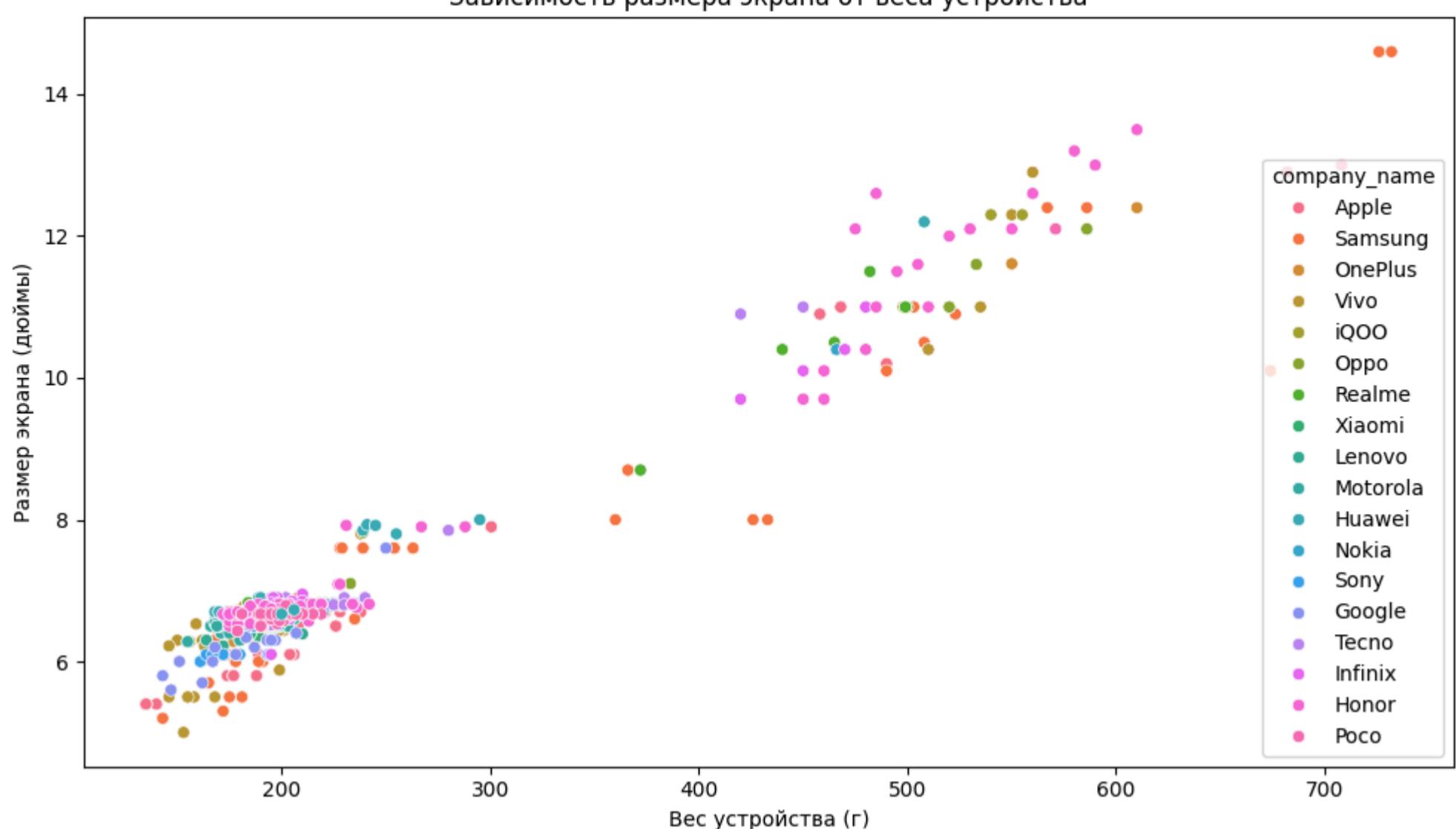
In [161]:

```
# Строим матрицу Phi-корреляции для всех признаков
phik_matrix_full = df_corr.phik_matrix(interval_cols=cols_to_drop)

# Визуализация
plt.figure(figsize=(22, 15))
sns.heatmap(phik_matrix_full, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Матрица Phi-корреляции признаков (числовых и категориальных)")
plt.show()
```



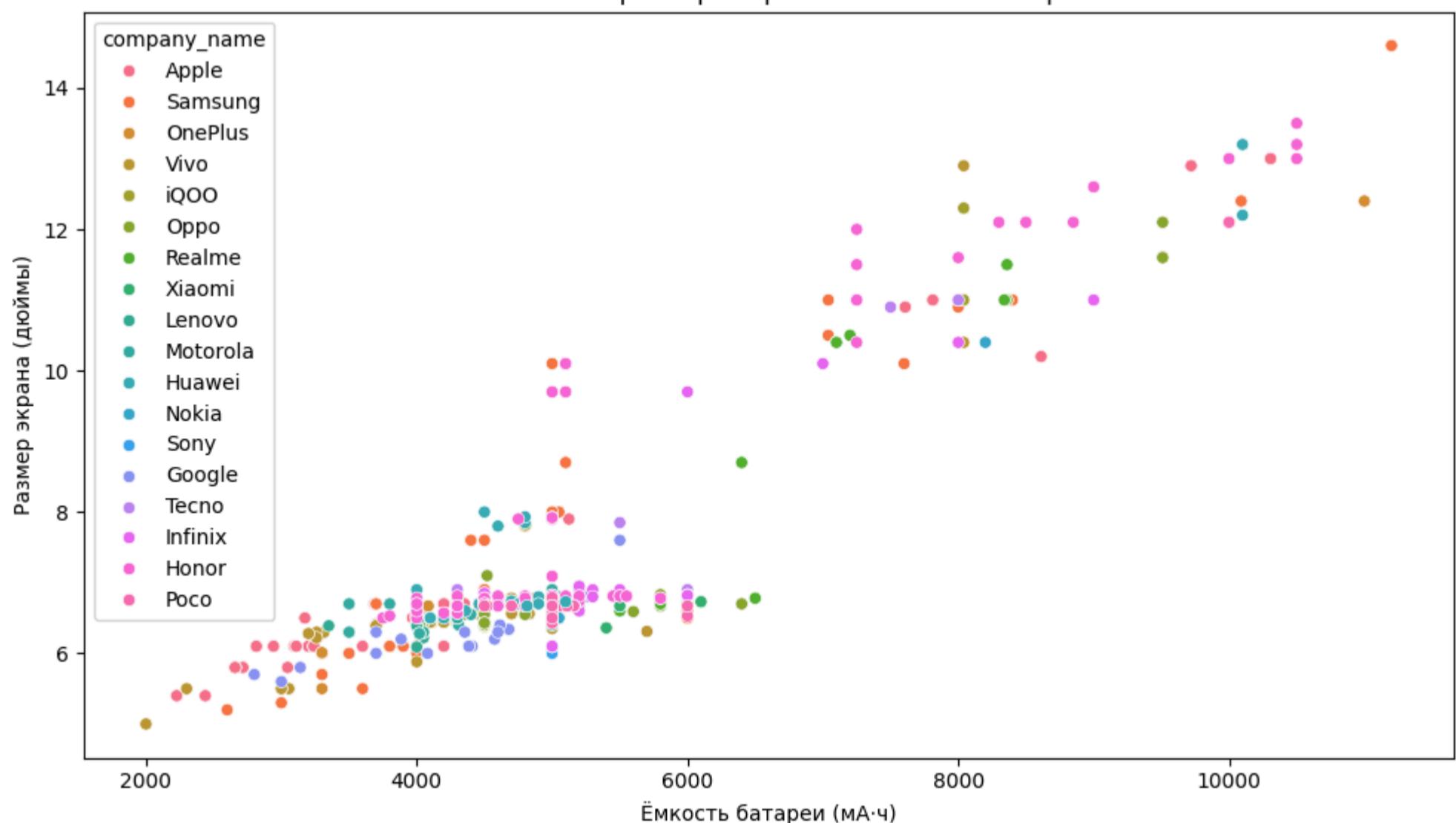
Зависимость размера экрана от веса устройства



- Аккумулятор vs Размер экрана

```
In [163...]: plt.figure(figsize=(10, 6)) # ширина 10 дюймов, высота 6 дюймов
sns.scatterplot(data=df, x='battery_capacity_mah', y='screen_size_inches', hue='company_name')
plt.title("Зависимость размера экрана от ёмкости батареи")
plt.xlabel("Ёмкость батареи (мА·ч)")
plt.ylabel("Размер экрана (дюймы)")
plt.tight_layout()
plt.show()
```

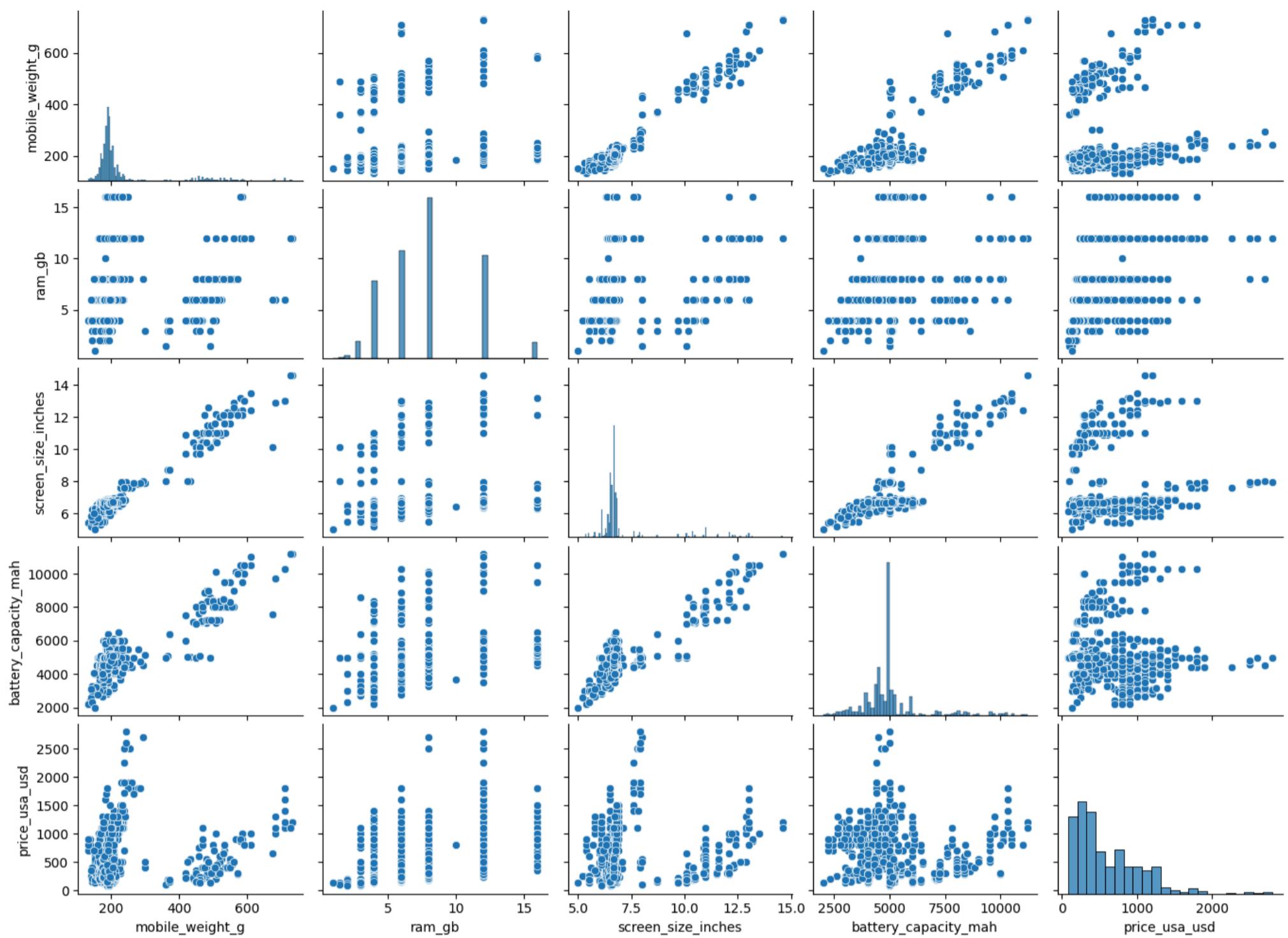
Зависимость размера экрана от ёмкости батареи



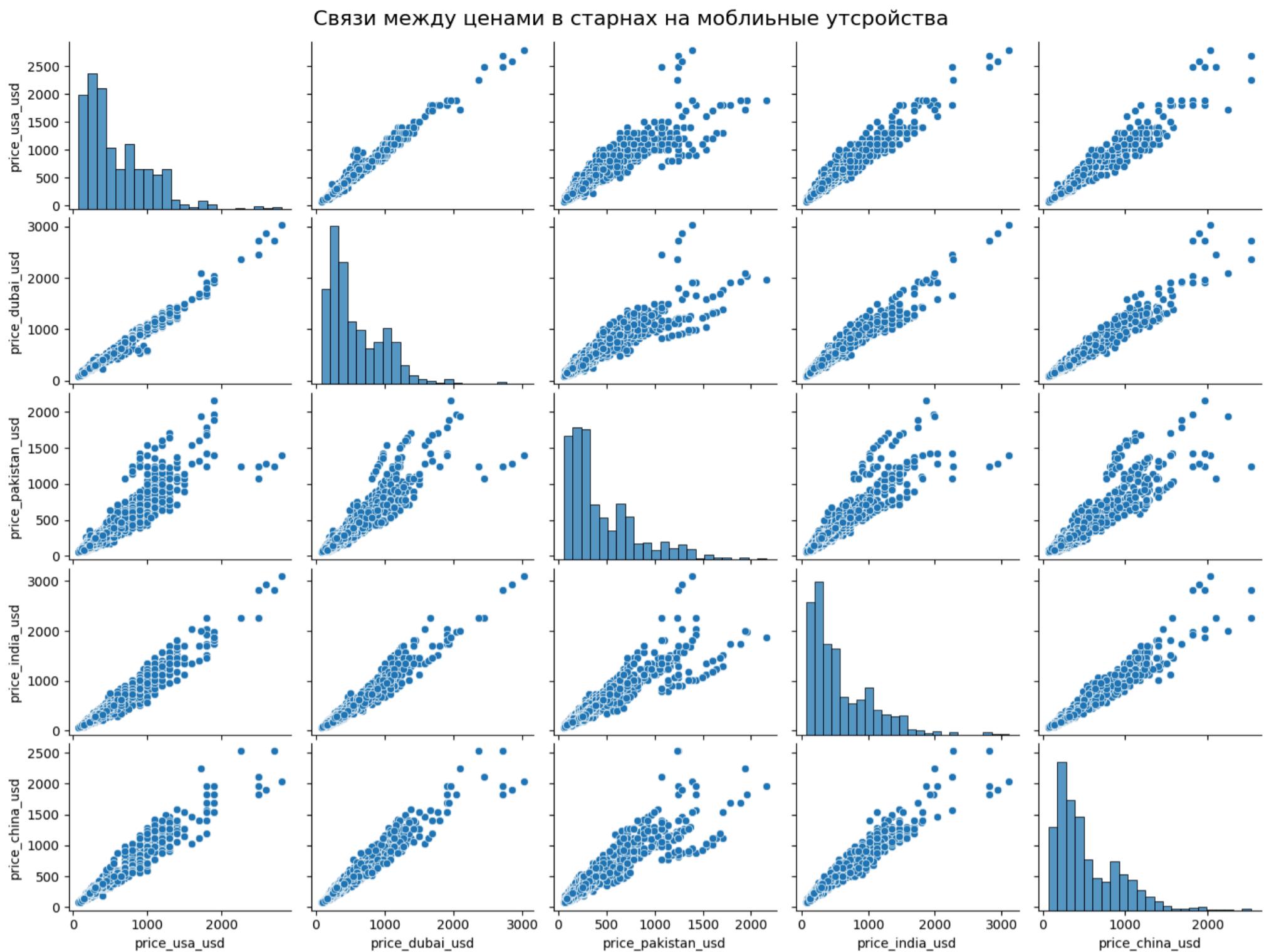
Pairplot

```
In [164...]: # Построение pairplot
sns.pairplot(df[['mobile_weight_g', 'ram_gb', 'screen_size_inches', 'battery_capacity_mah', 'price_usa_usd']])
plt.suptitle("Связи между характеристиками мобильных устройств", y=1.02, fontsize=16)
plt.gcf().set_size_inches(14, 10) # установка размера рисунка
plt.show()
```

Связи между характеристиками мобильных устройств



```
In [165...]: # Построение pairplot
sns.pairplot(df[['price_usa_usd', 'price_dubai_usd', 'price_pakistan_usd', 'price_india_usd', 'price_china_usd']])
plt.suptitle("Связи между ценами в странах на мобильные устройства", y=1.02, fontsize=16)
plt.gcf().set_size_inches(14, 10) # установка размера рисунка
plt.show()
```



Тепловые карты для категориальных признаков

```
In [166...]: categorical_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()
print(categorical_cols)

['company_name', 'model_name', 'weight_category', 'ram_category', 'processor', 'battery_category', 'screen_category', 'price_category']

In [167...]: def plot_categorical_heatmap(df, row_col, col_col, figsize=(8,6), cmap='YlGnBu'):
    ct = pd.crosstab(df[row_col], df[col_col])

    plt.figure(figsize=figsize)
    sns.heatmap(ct, annot=True, fmt='d', cmap=cmap)
    plt.title(f'{row_col} vs {col_col}')
    plt.xlabel(col_col)
    plt.ylabel(row_col)
    plt.tight_layout()
    plt.show()

In [168...]: from itertools import combinations

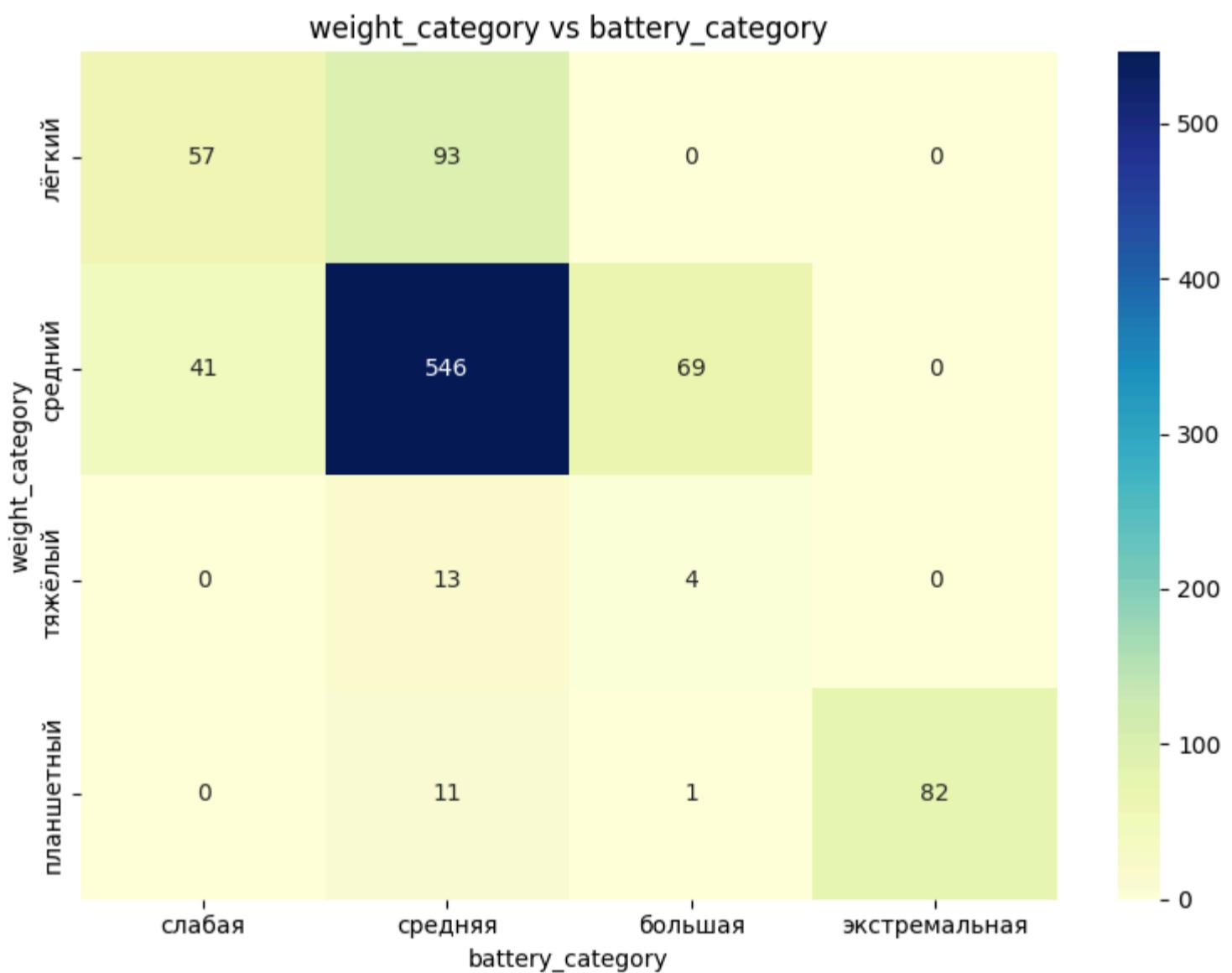
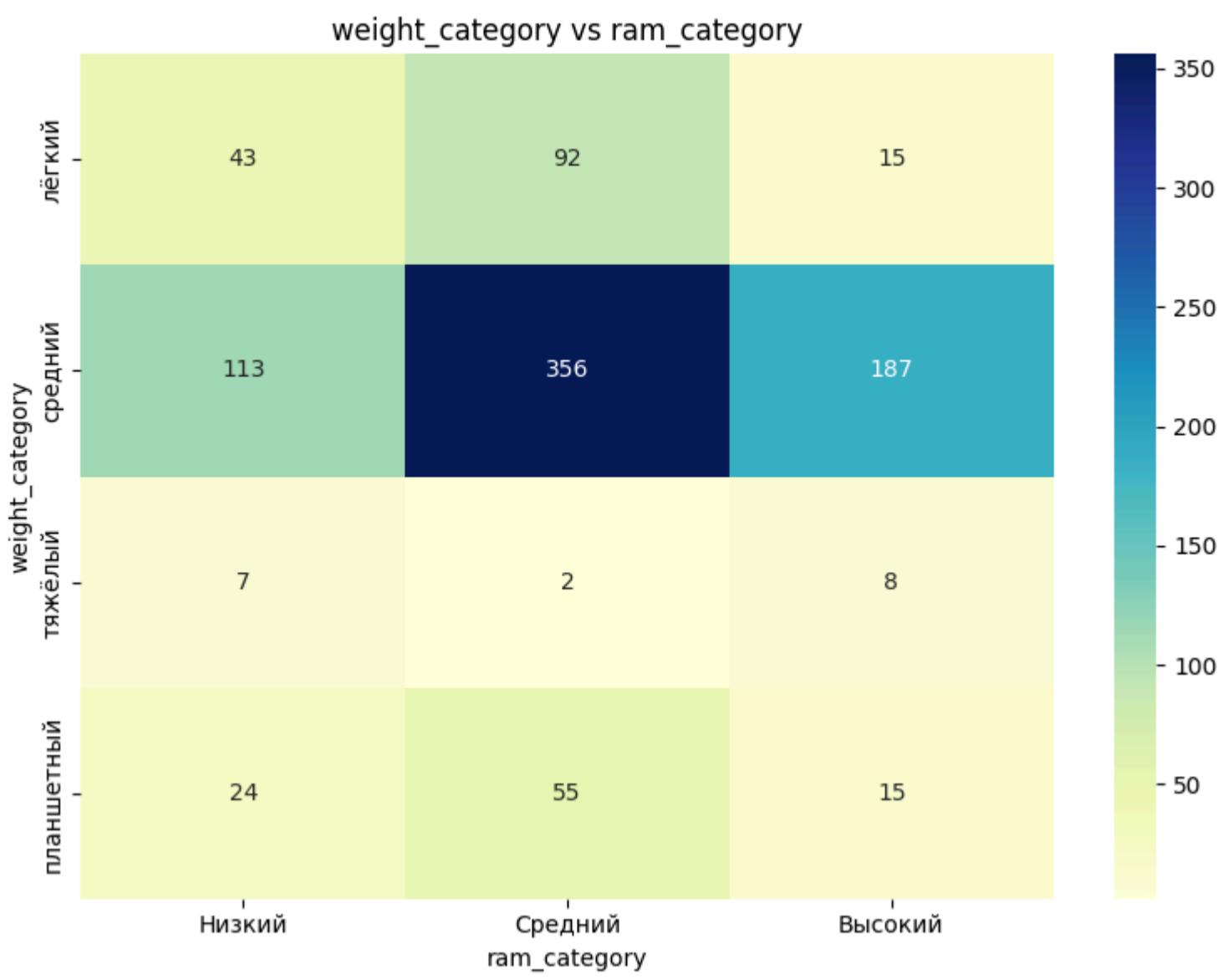
# Список интересующих категориальных признаков
cat_cols = ['weight_category', 'ram_category', 'battery_category', 'screen_category', 'price_category']

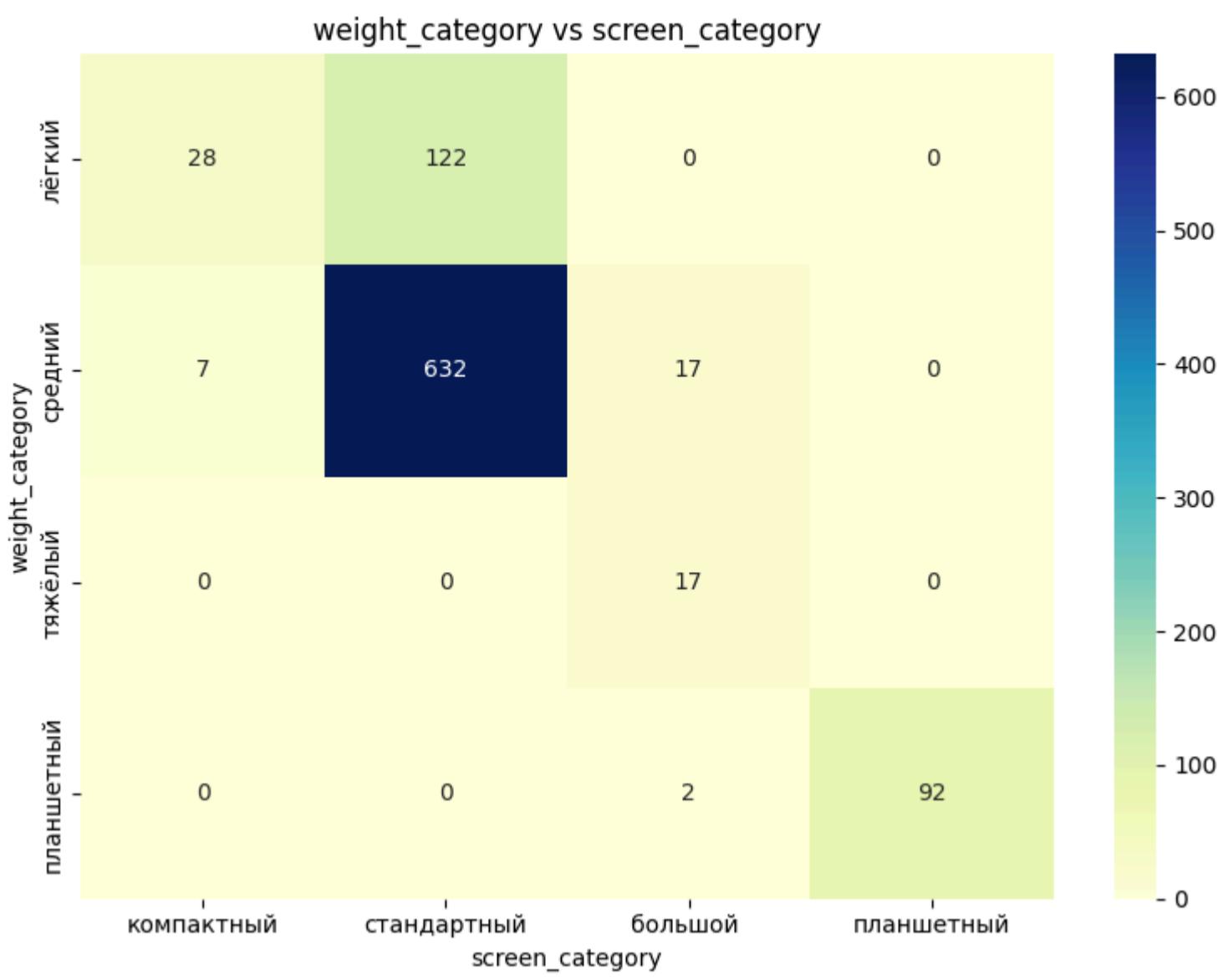
# Генерация всех уникальных пар
unique_pairs = list(combinations(cat_cols, 2))

# Выводим пары
for pair in unique_pairs:
    print(f'{pair[0]} vs {pair[1]}')

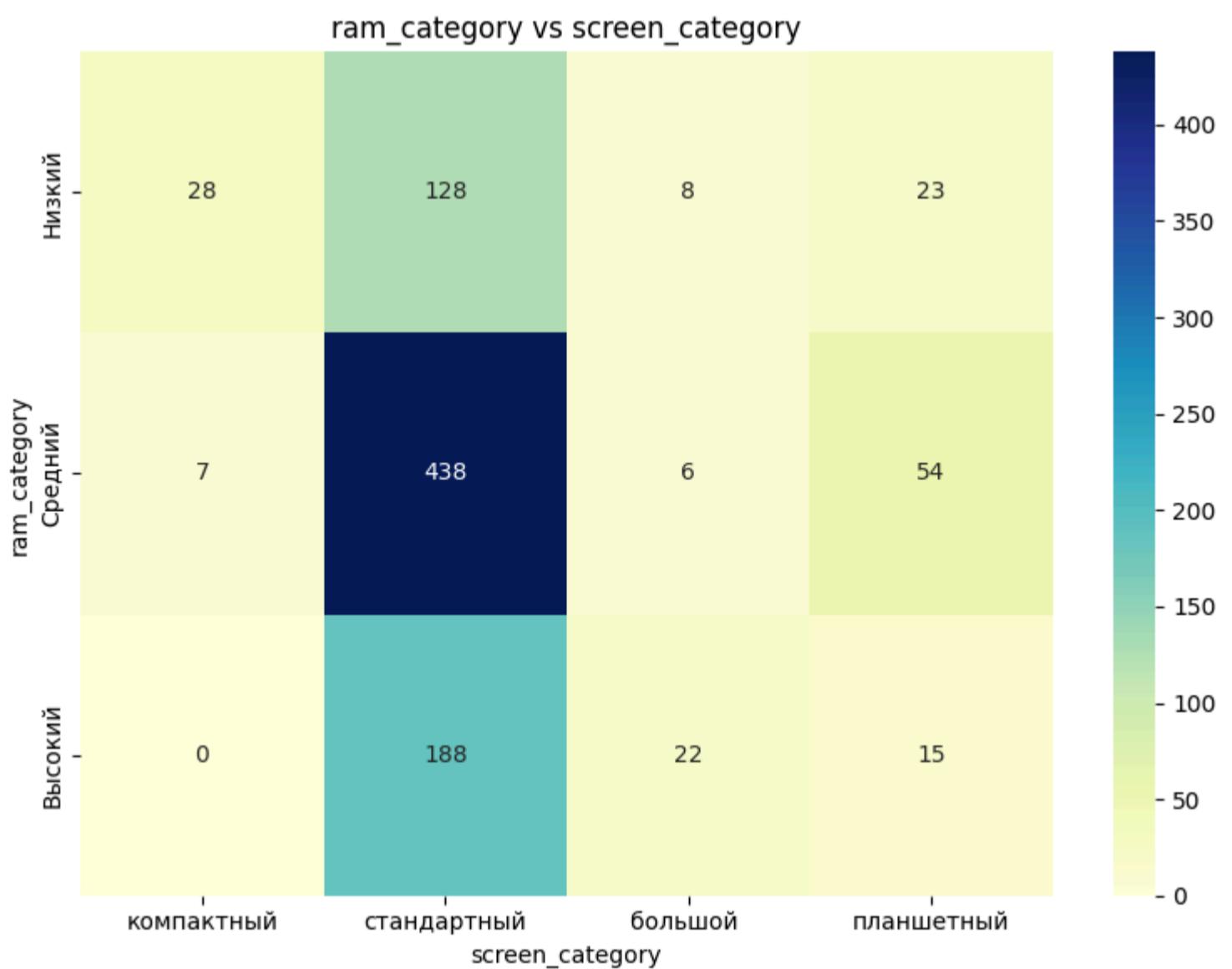
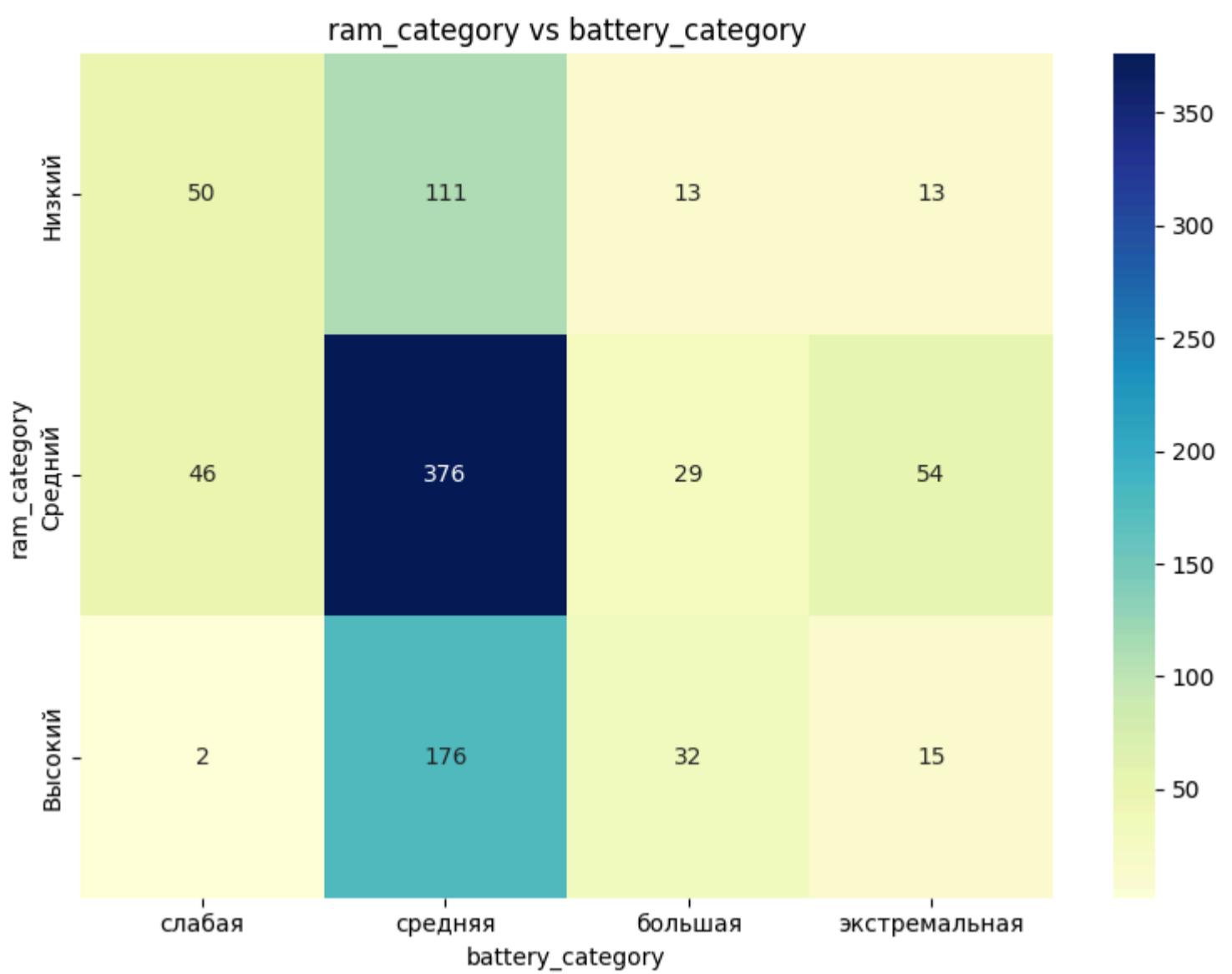

weight_category vs ram_category
weight_category vs battery_category
weight_category vs screen_category
weight_category vs price_category
ram_category vs battery_category
ram_category vs screen_category
ram_category vs price_category
battery_category vs screen_category
battery_category vs price_category
screen_category vs price_category

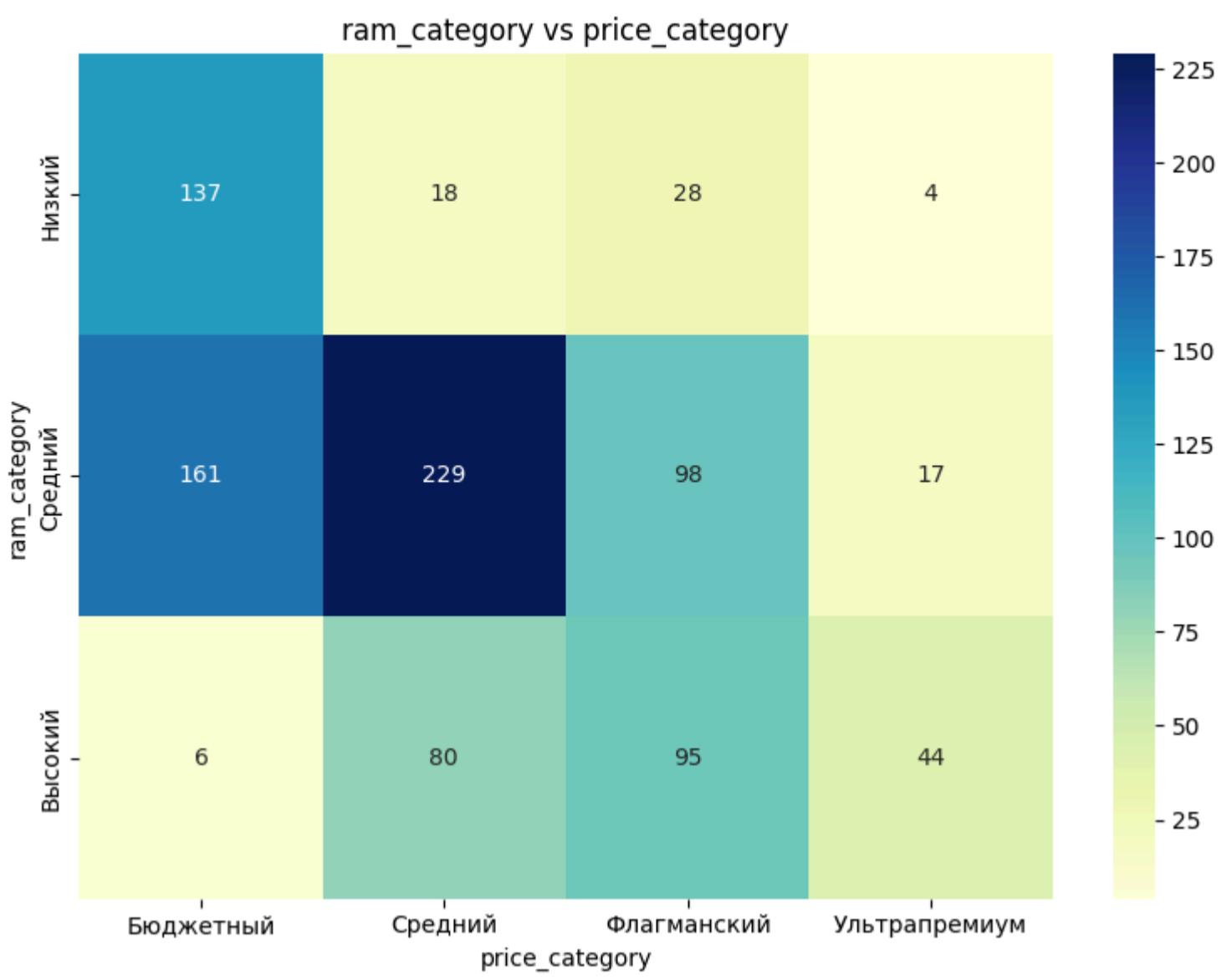
In [169...]: plot_categorical_heatmap(df, 'weight_category', 'ram_category')
plot_categorical_heatmap(df, 'weight_category', 'battery_category')
plot_categorical_heatmap(df, 'weight_category', 'screen_category')
plot_categorical_heatmap(df, 'weight_category', 'price_category')
```



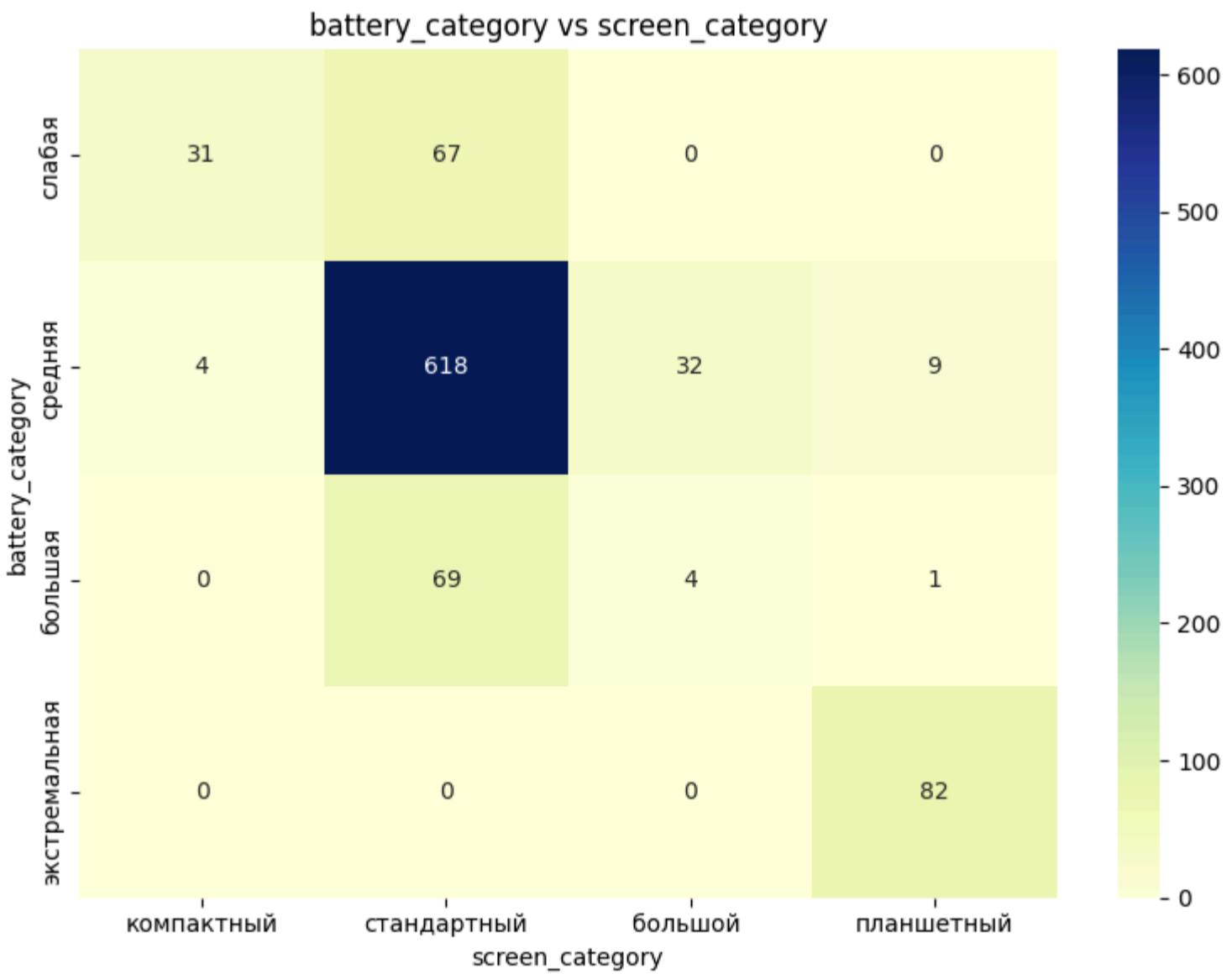


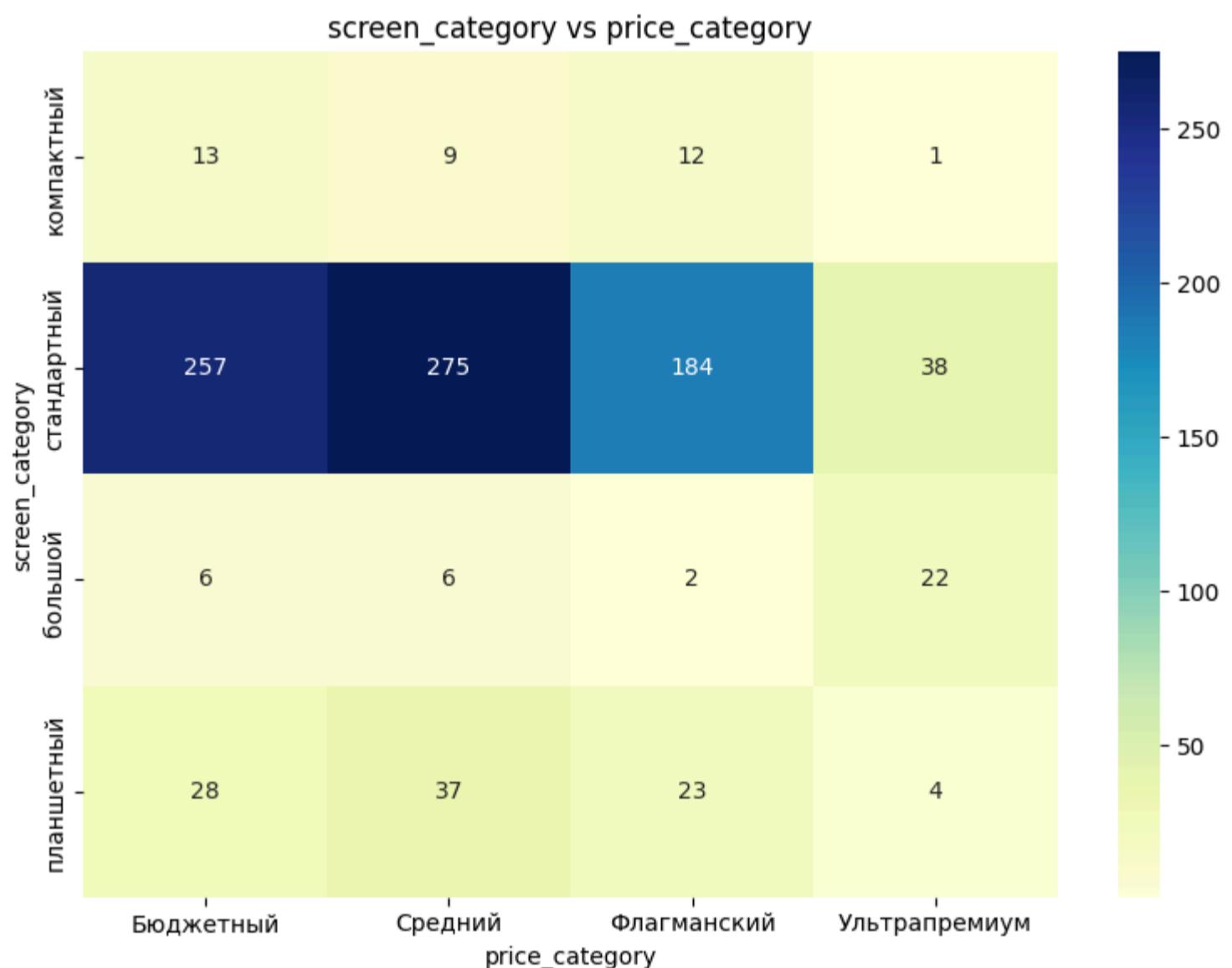
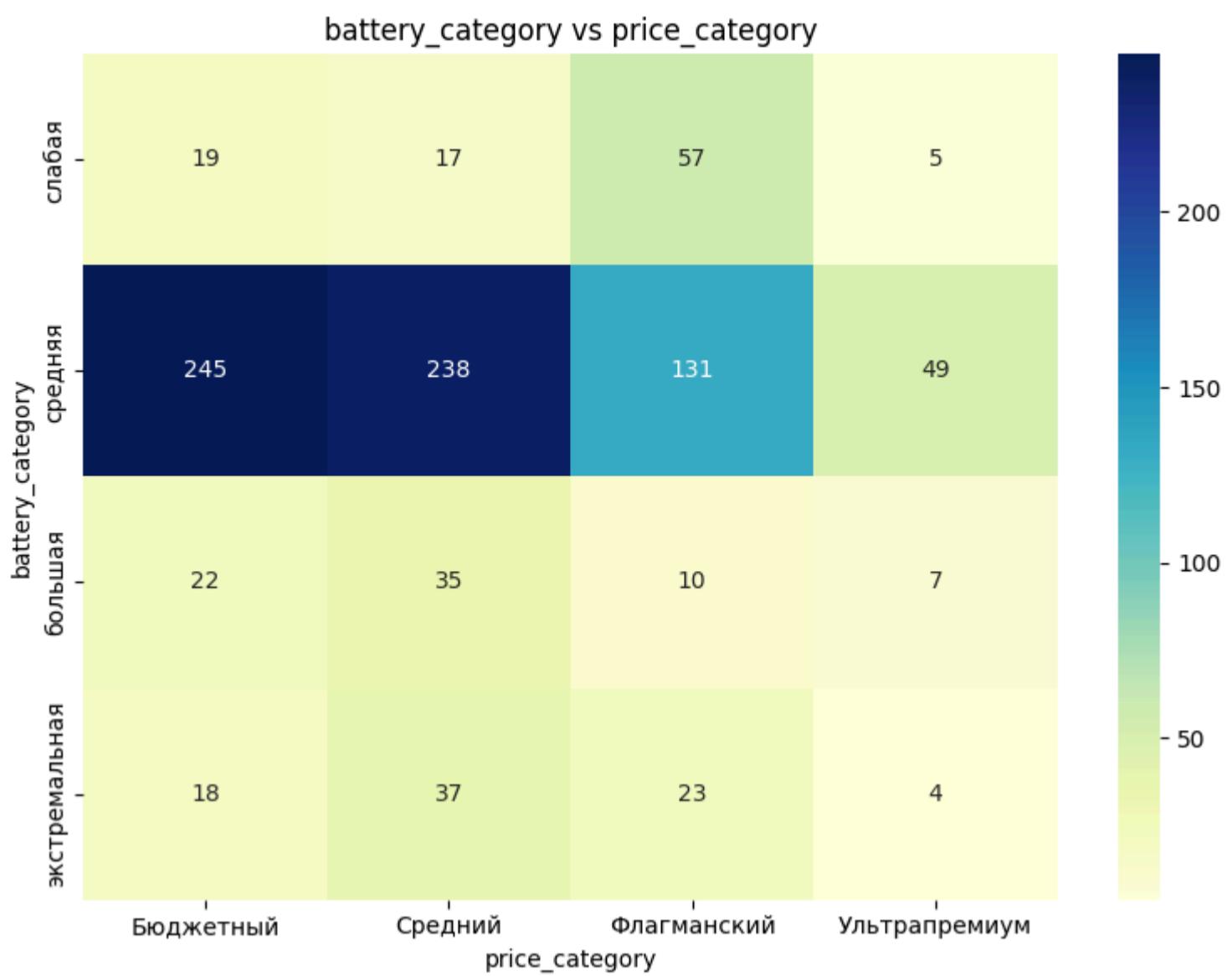
```
In [170]: plot_categorical_heatmap(df, 'ram_category', 'battery_category')
plot_categorical_heatmap(df, 'ram_category', 'screen_category')
plot_categorical_heatmap(df, 'ram_category', 'price_category')
```





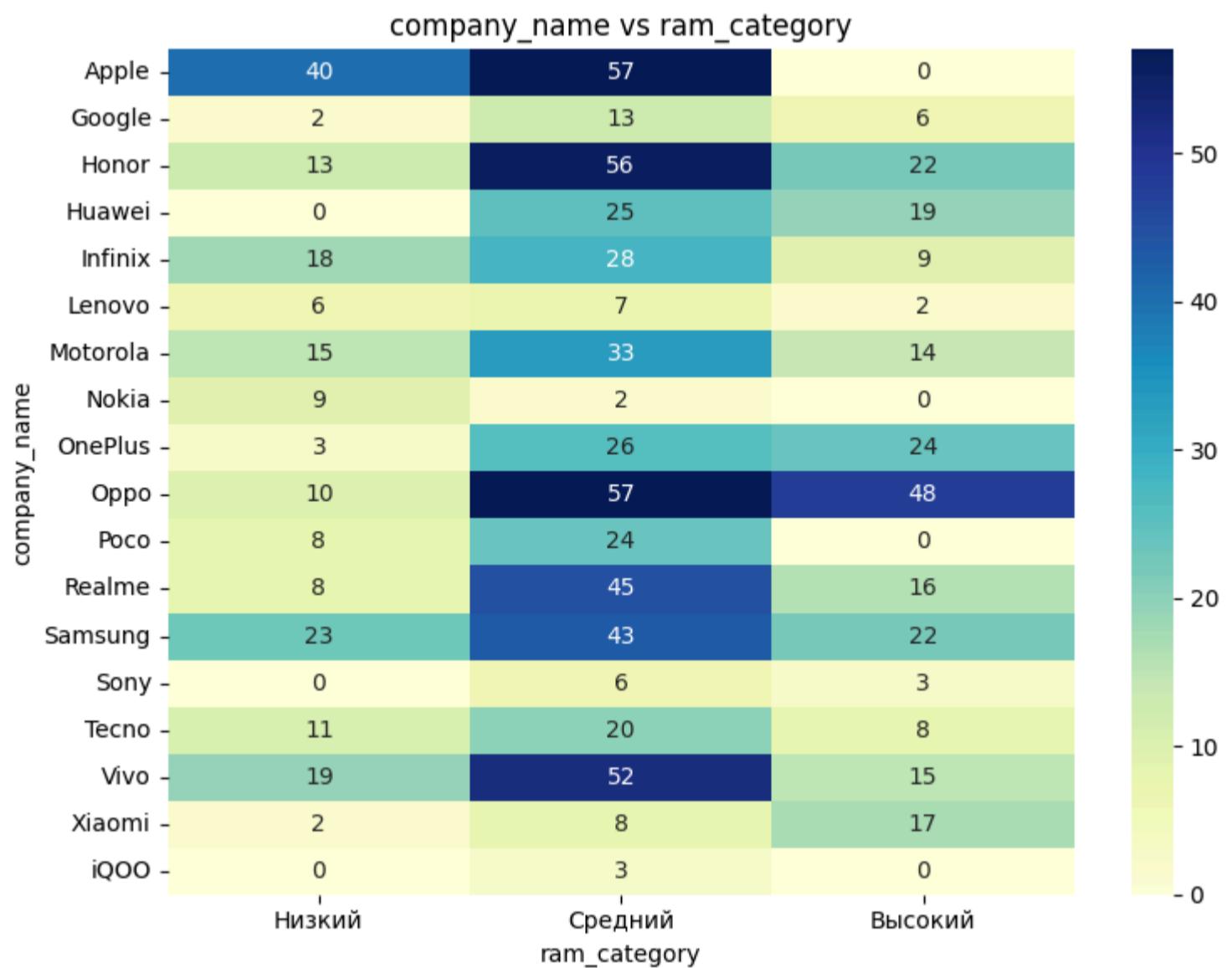
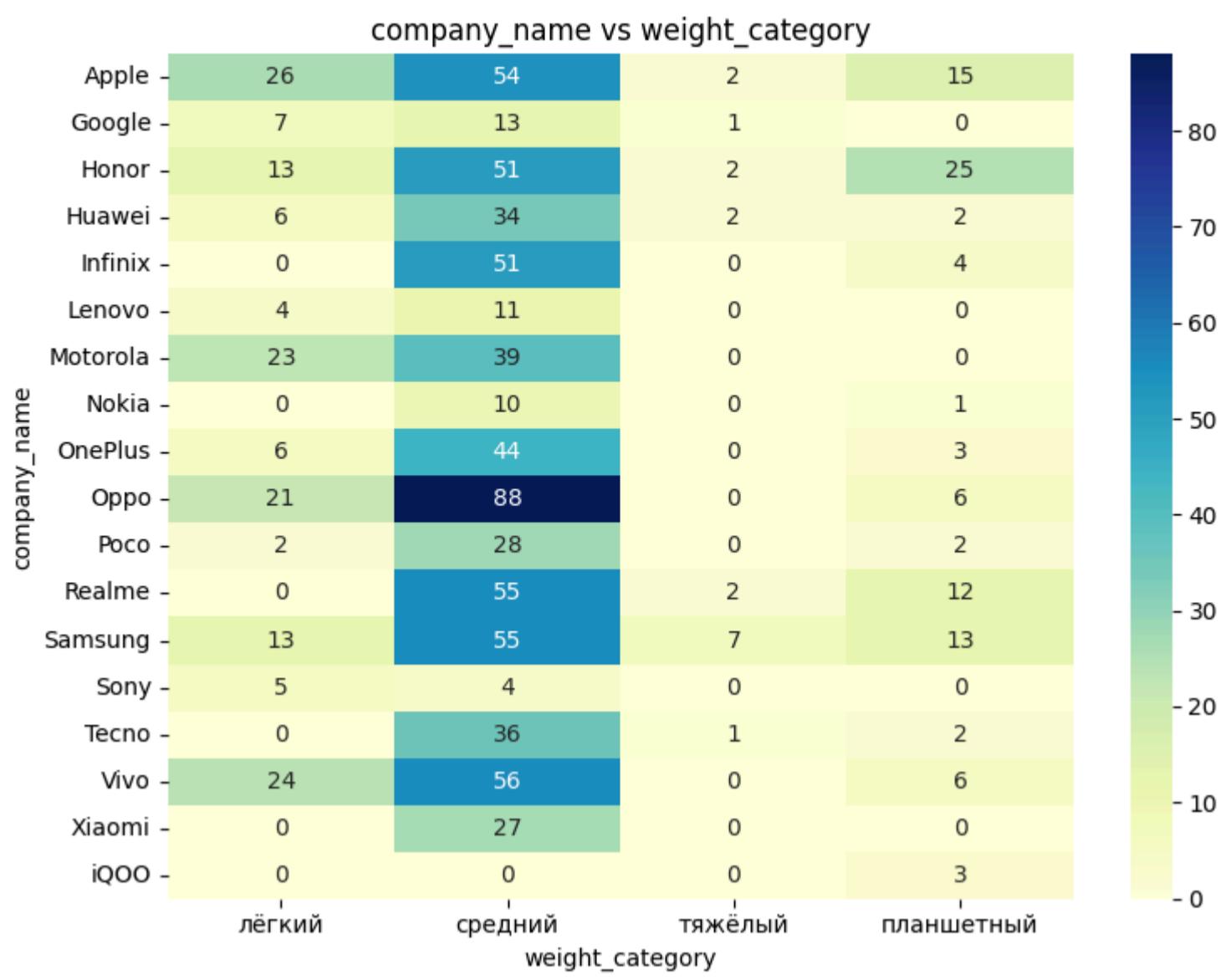
```
In [171]: plot_categorical_heatmap(df, 'battery_category', 'screen_category')
plot_categorical_heatmap(df, 'battery_category', 'price_category')
plot_categorical_heatmap(df, 'screen_category', 'price_category')
```

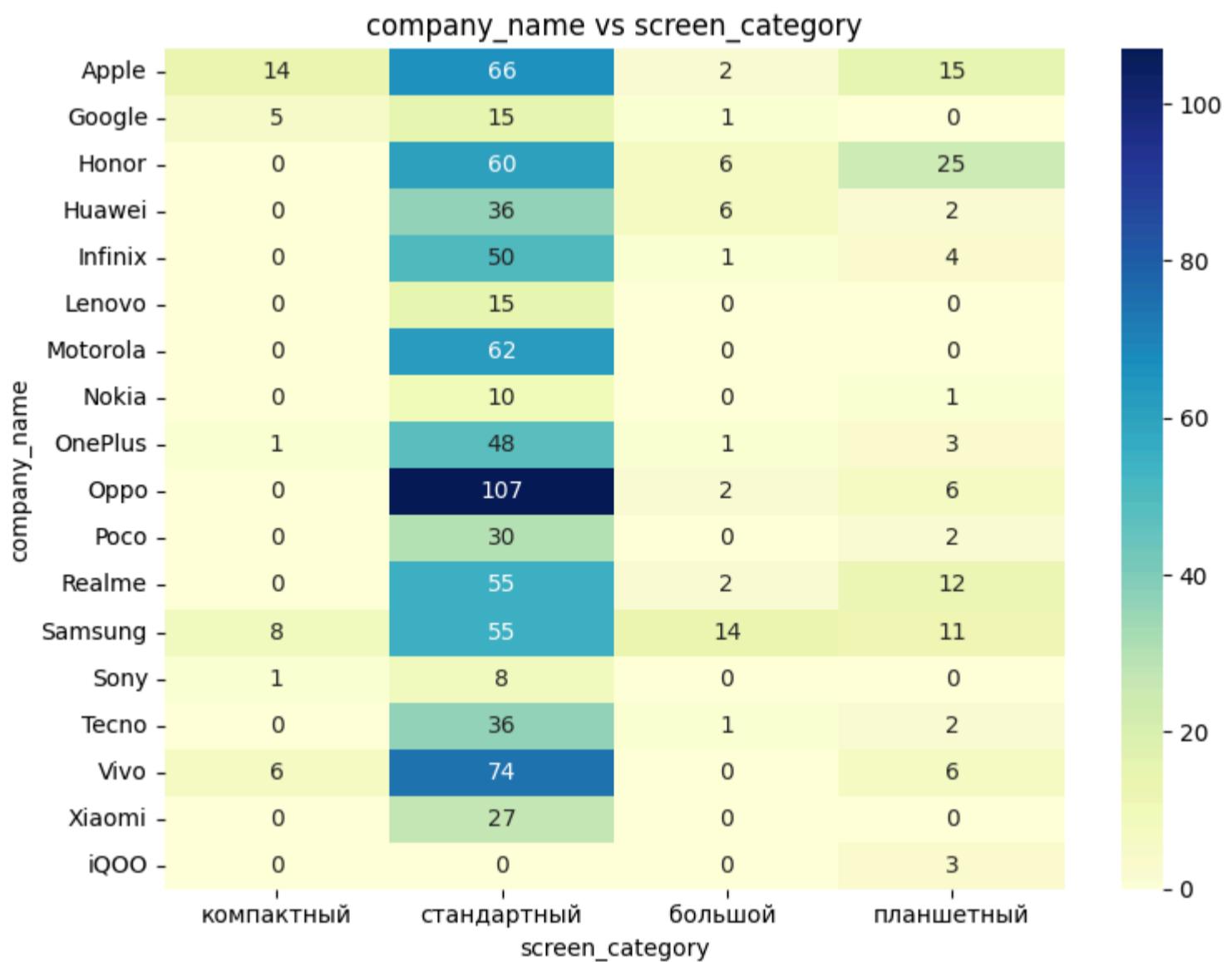
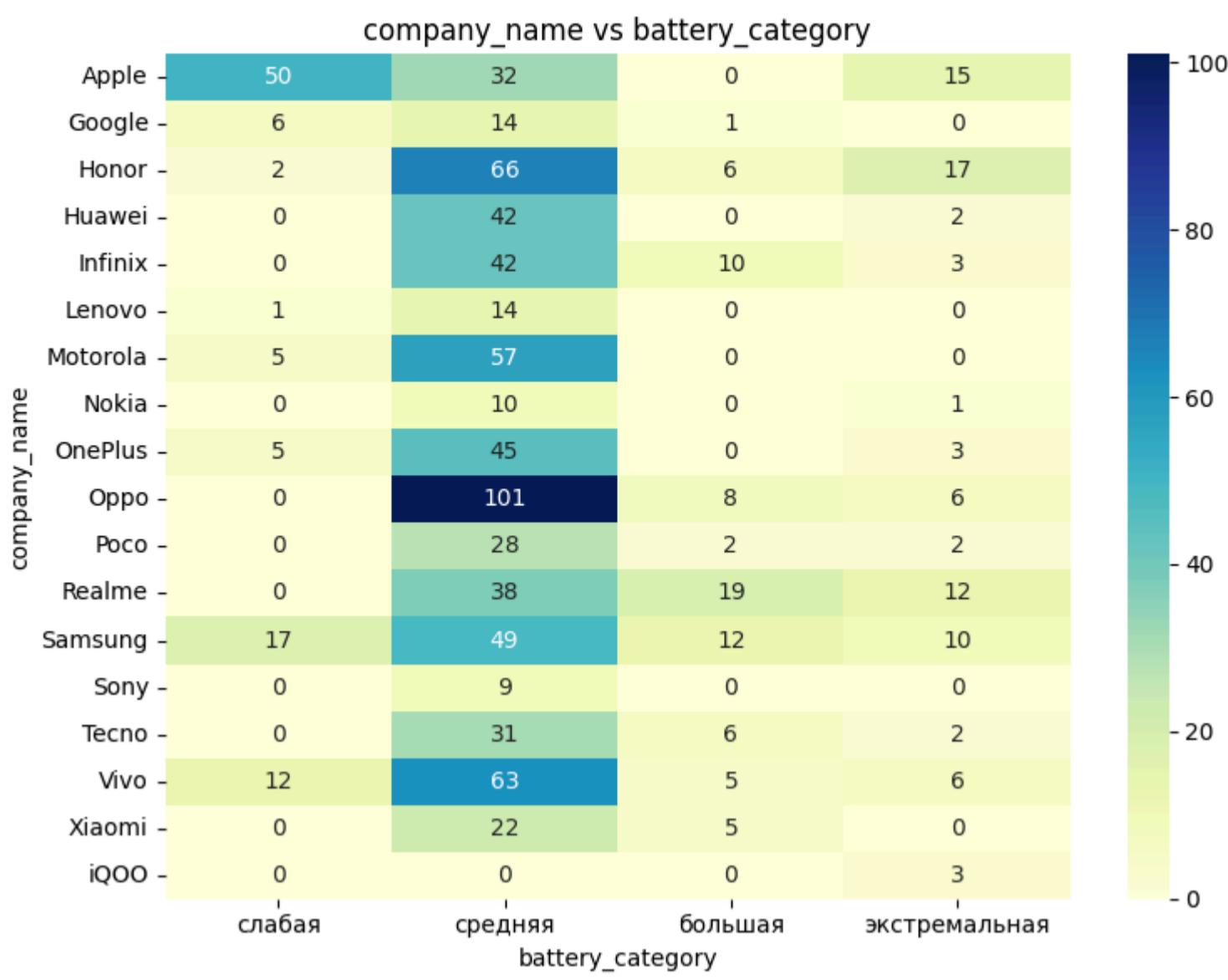


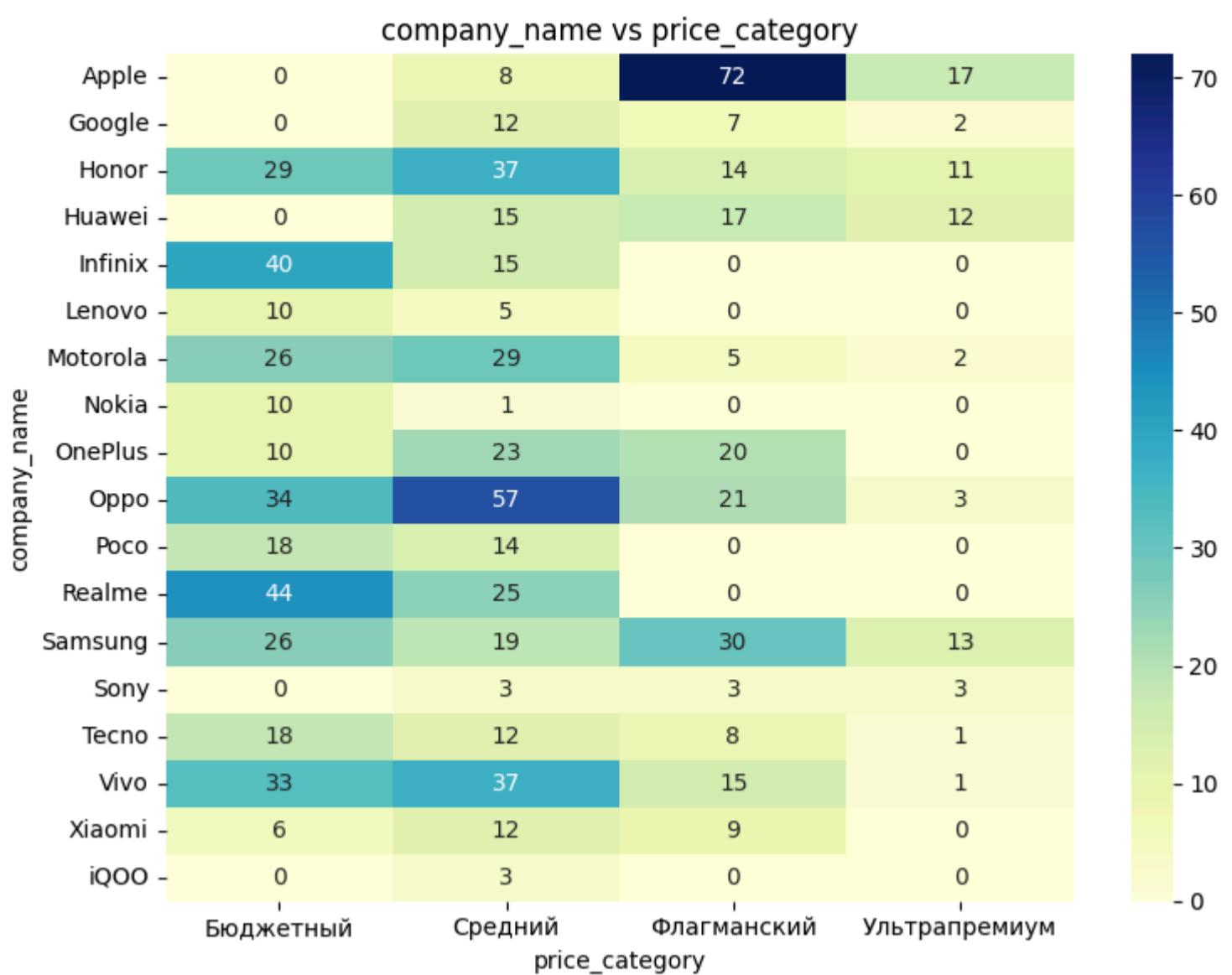


In [172]:

```
plot_categorical_heatmap(df, 'company_name', 'weight_category')
plot_categorical_heatmap(df, 'company_name', 'ram_category')
plot_categorical_heatmap(df, 'company_name', 'battery_category')
plot_categorical_heatmap(df, 'company_name', 'screen_category')
plot_categorical_heatmap(df, 'company_name', 'price_category')
```





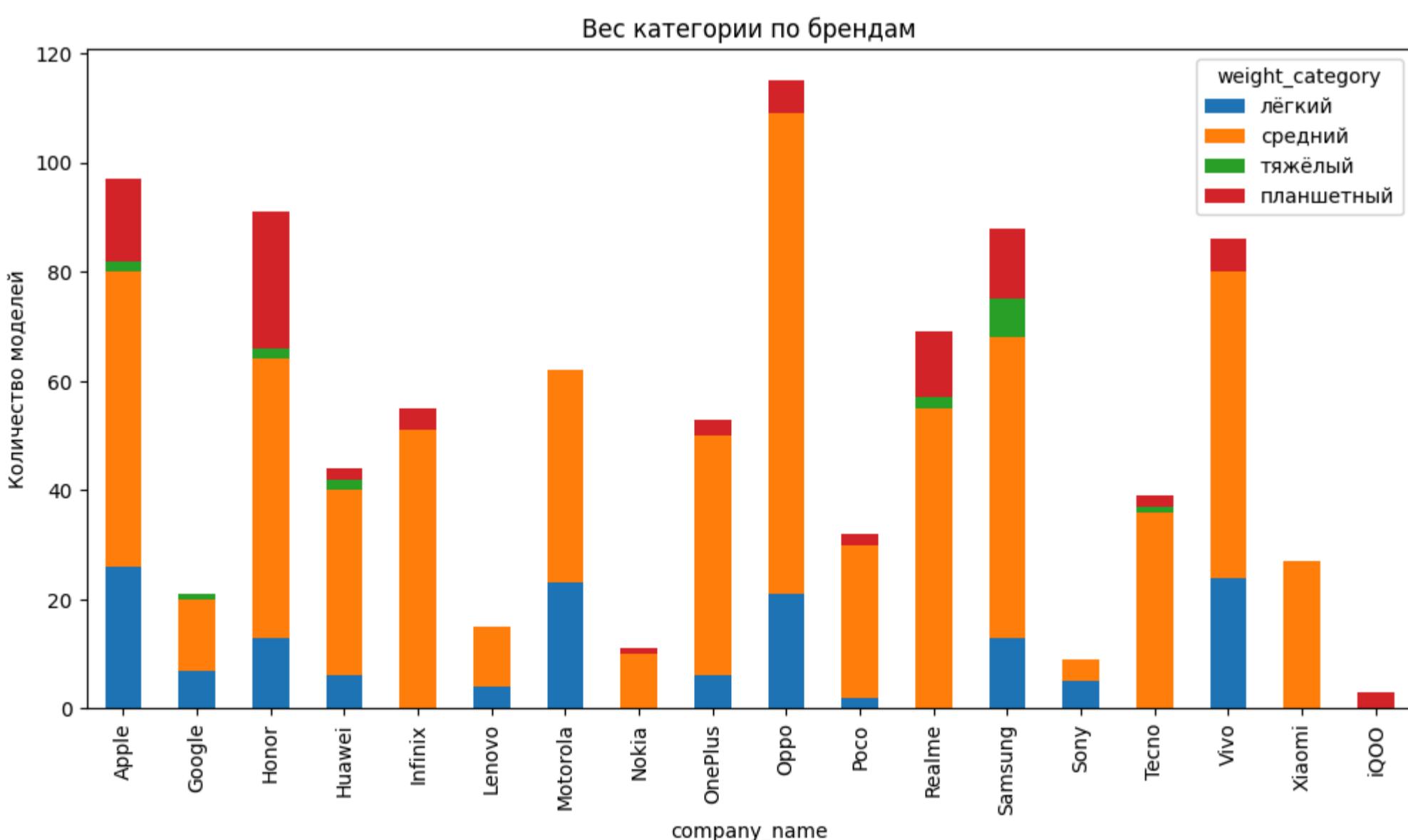


Распределение категорий относительно компании

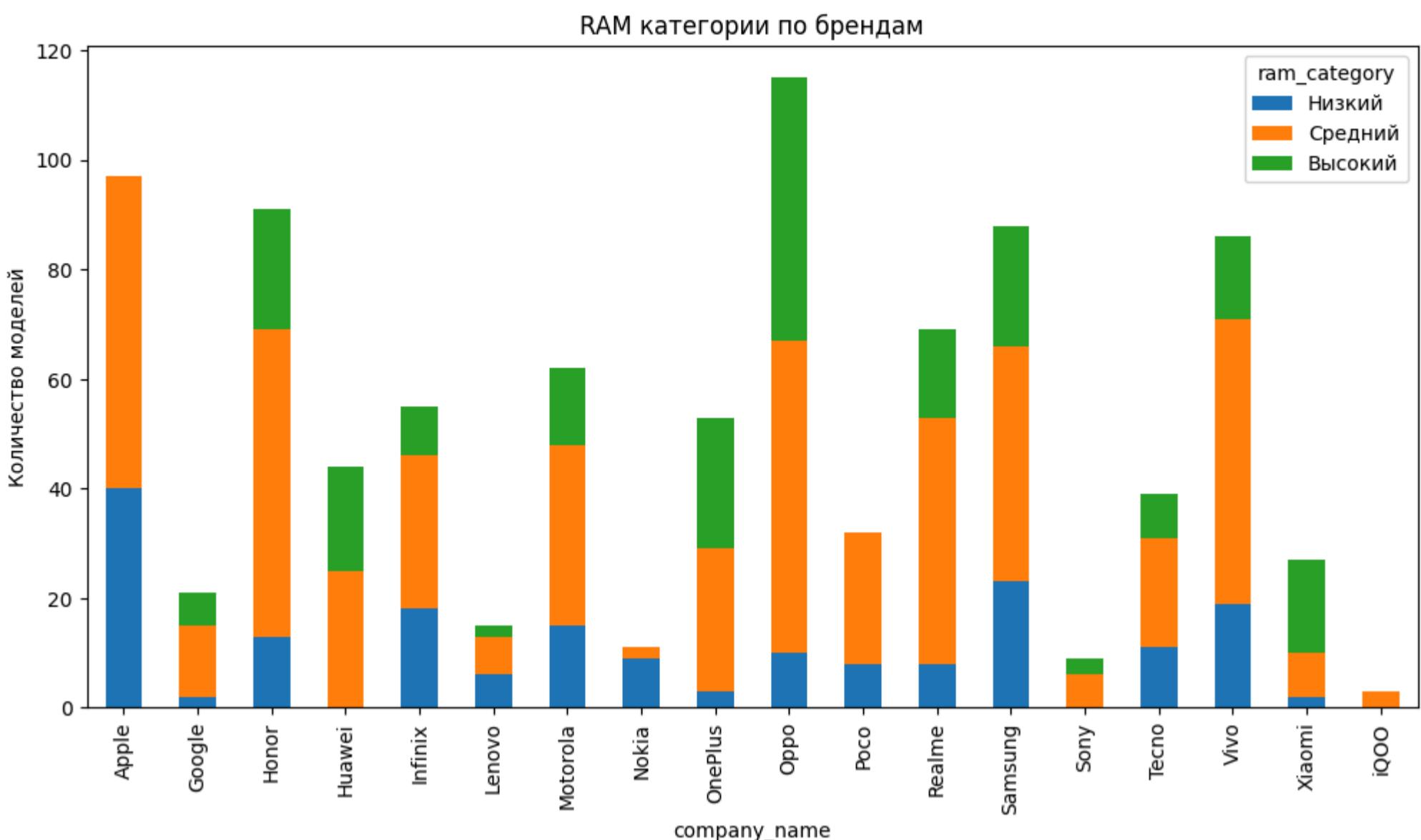
```
In [173...]: categorical_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()
print(categorical_cols)

['company_name', 'model_name', 'weight_category', 'ram_category', 'processor', 'battery_category', 'screen_category', 'price_category']

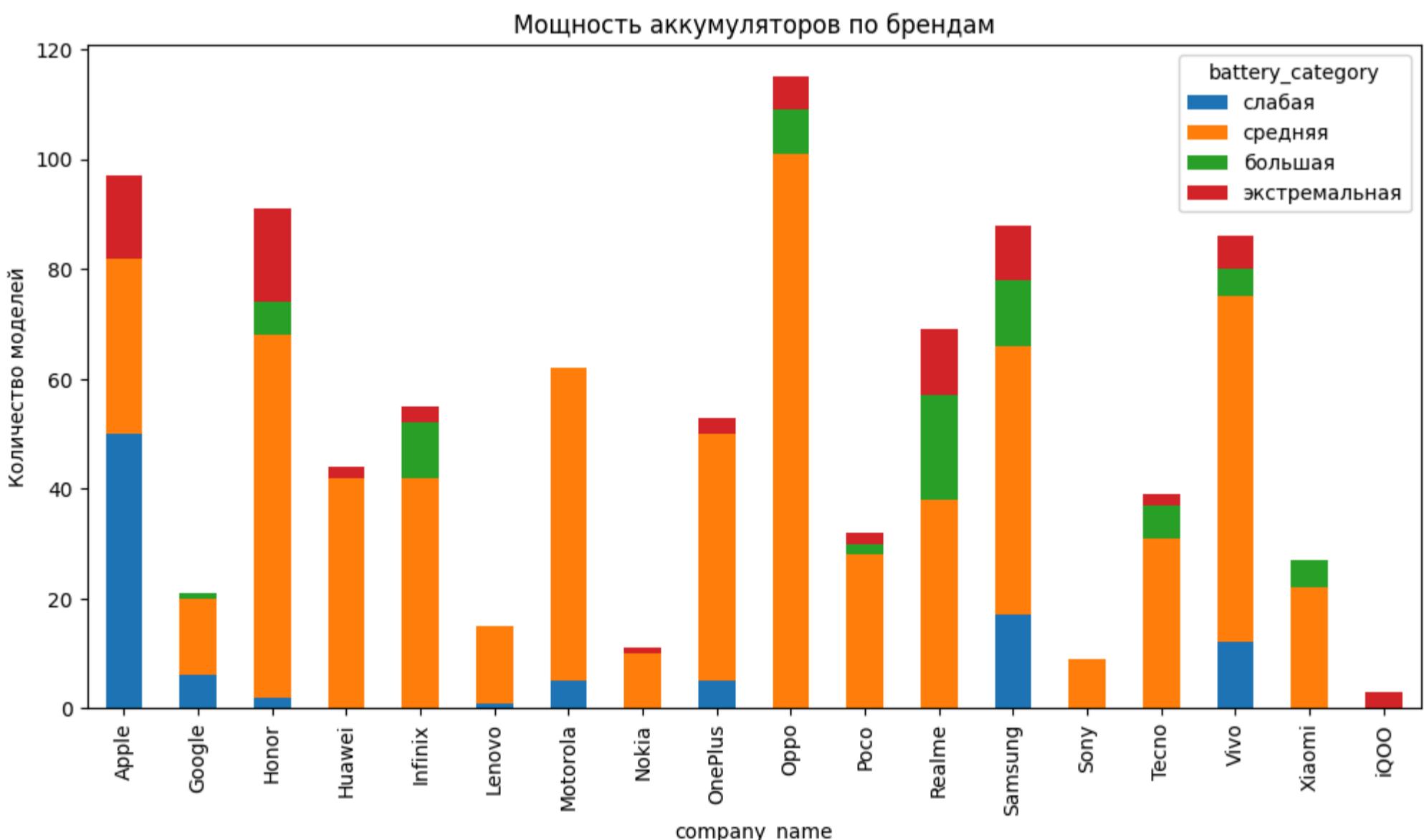
In [174...]: ct = pd.crosstab(df['company_name'], df['weight_category'])
ct.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("Вес категории по брендам")
plt.ylabel("Количество моделей")
plt.show()
```



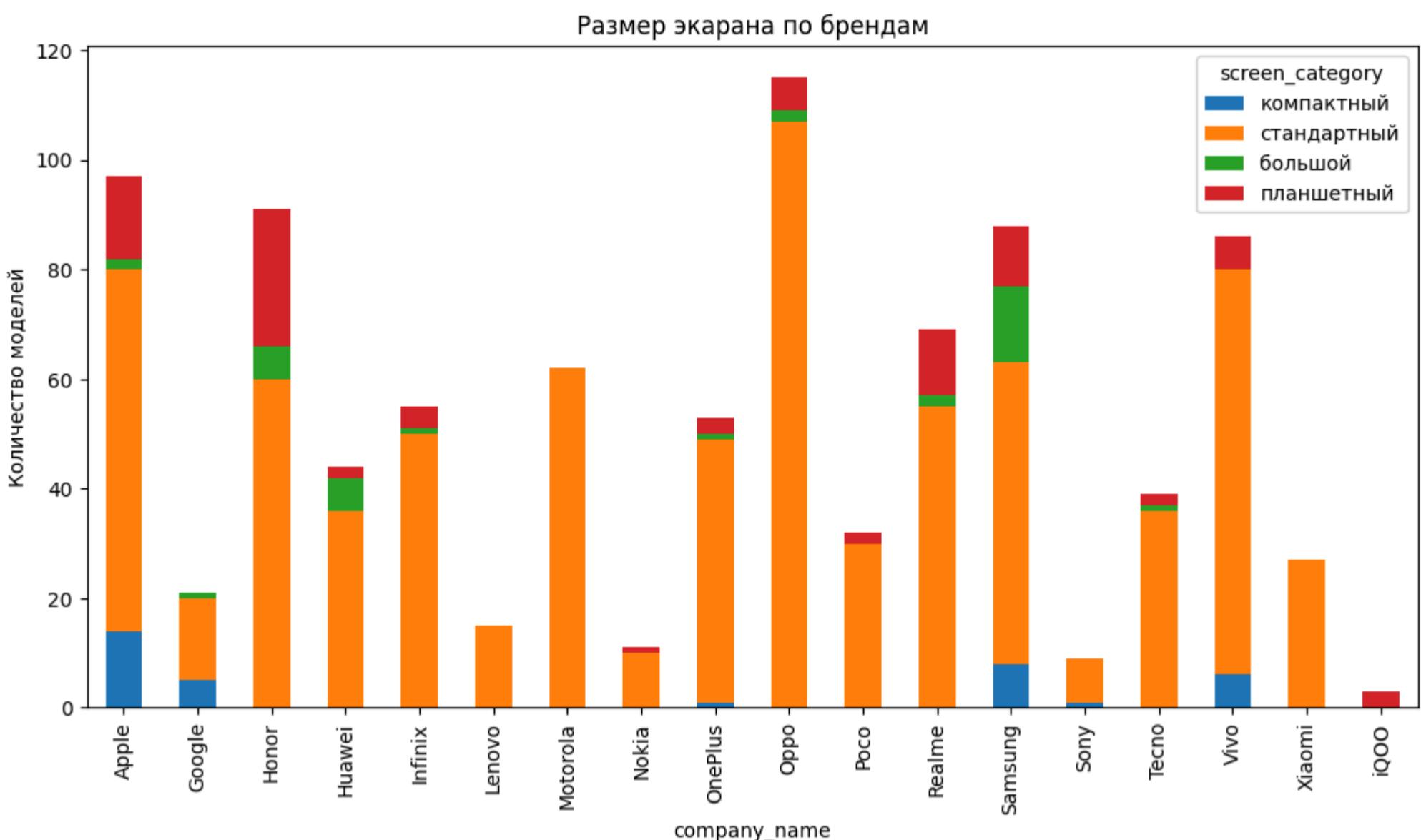
```
In [175...]: ct = pd.crosstab(df['company_name'], df['ram_category'])
ct.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("RAM категории по брендам")
plt.ylabel("Количество моделей")
plt.show()
```



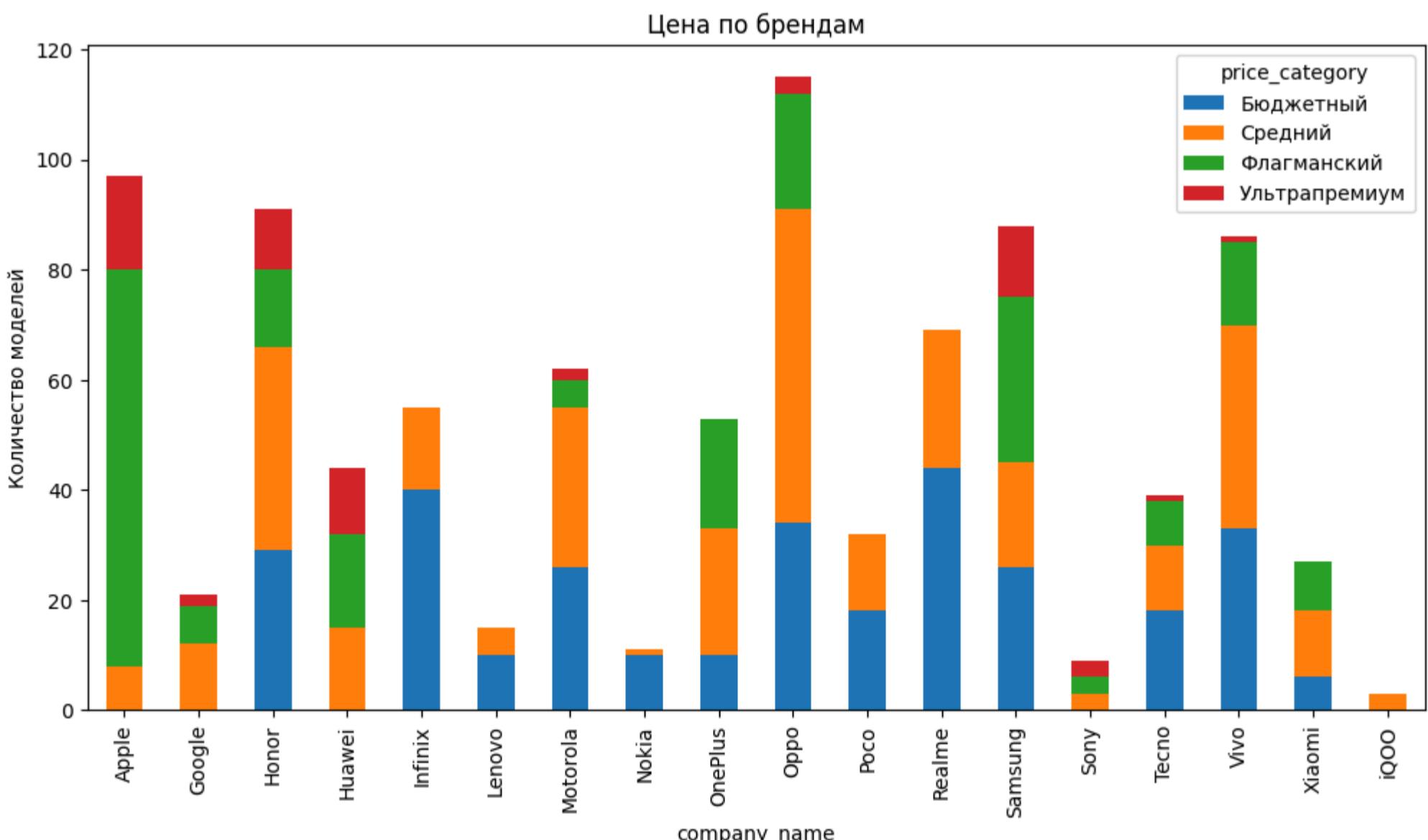
```
In [176]: ct = pd.crosstab(df['company_name'], df['battery_category'])
ct.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("Мощность аккумуляторов по брендам")
plt.ylabel("Количество моделей")
plt.show()
```



```
In [177]: ct = pd.crosstab(df['company_name'], df['screen_category'])
ct.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("Размер экрана по брендам")
plt.ylabel("Количество моделей")
plt.show()
```



```
In [178]: ct = pd.crosstab(df['company_name'], df['price_category'])
ct.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("Цена по брендам")
plt.ylabel("Количество моделей")
plt.show()
```



Группировка и агрегации

Средние и медианные цены мобильных устройств по странам

```
In [179]: # Словарь: страна → колонка
country_price_cols = {
    'Pakistan': 'price_pakistan_usd',
    'India': 'price_india_usd',
    'China': 'price_china_usd',
    'Dubai': 'price_dubai_usd',
    'USA': 'price_usa_usd'
}
```

```

# Вычисляем статистики
price_stats = pd.DataFrame({
    'Средняя цена (USD)': {country: df[col].mean() for country, col in country_price_cols.items()},
    'Медианная цена (USD)': {country: df[col].median() for country, col in country_price_cols.items()}
})

# Сортировка по средней цене
price_stats = price_stats.sort_values(by='Средняя цена (USD)')

# Построение grouped bar chart
x = np.arange(len(price_stats)) # позиции по оси X
width = 0.4 # ширина столбца

plt.figure(figsize=(10, 6))
plt.bar(x - width/2, price_stats['Средняя цена (USD)'], width=width, label='Средняя цена', color='teal')
plt.bar(x + width/2, price_stats['Медианная цена (USD)'], width=width, label='Медианная цена', color='orange')

plt.xticks(x, price_stats.index)
plt.xlabel("Страна")
plt.ylabel("Цена (USD)")
plt.title("Сравнение средней и медианной цены мобильных устройств по странам")
plt.legend()
plt.tight_layout()
plt.show()

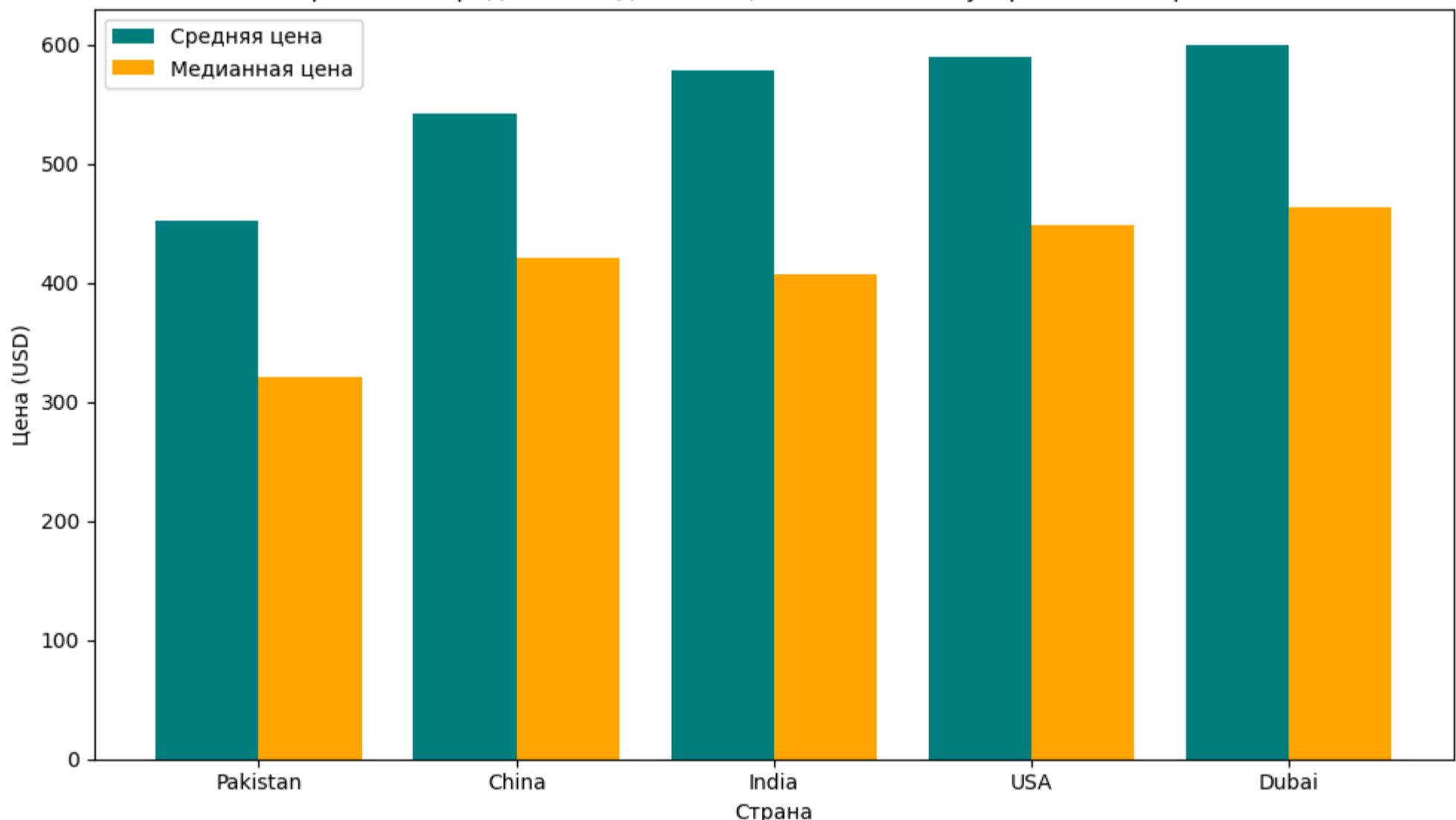
# Создаём DataFrame со средними и медианными значениями
price_stats = pd.DataFrame({
    'Средняя цена (USD)': {country: df[col].mean() for country, col in country_price_cols.items()},
    'Медианная цена (USD)': {country: df[col].median() for country, col in country_price_cols.items()}
})

# Сортировка по средней цене
price_stats = price_stats.sort_values(by='Средняя цена (USD)')

# Вывод
display(price_stats)

```

Сравнение средней и медианной цены мобильных устройств по странам



Средняя цена (USD) Медианная цена (USD)

	Средняя цена (USD)	Медианная цена (USD)
Pakistan	452.416576	321.0
China	542.279171	421.0
India	578.430752	407.0
USA	590.160174	449.0
Dubai	600.272628	463.0

Сравнение средней и медианной доступности смартфонов по странам

- Чем ниже индекс, тем доступнее смартфон для населения страны.
- Например:
 - USA: ~0.09 → смартфоны доступны
 - India: ~1.72 → смартфоны менее доступны

```
# Словарь: страна → колонка доступности
affordability_cols = {
    'Pakistan': 'affordability_pakistan',
    'India': 'affordability_india',
    'China': 'affordability_china',
    'UAE': 'affordability_uae',
    'USA': 'affordability_usa'
}

# Создаём DataFrame со средними и медианными значениями доступности
affordability_stats = pd.DataFrame({
    'Средняя доступность': {country: df[col].mean() for country, col in affordability_cols.items()},
    'Медианная доступность': {country: df[col].median() for country, col in affordability_cols.items()}
})

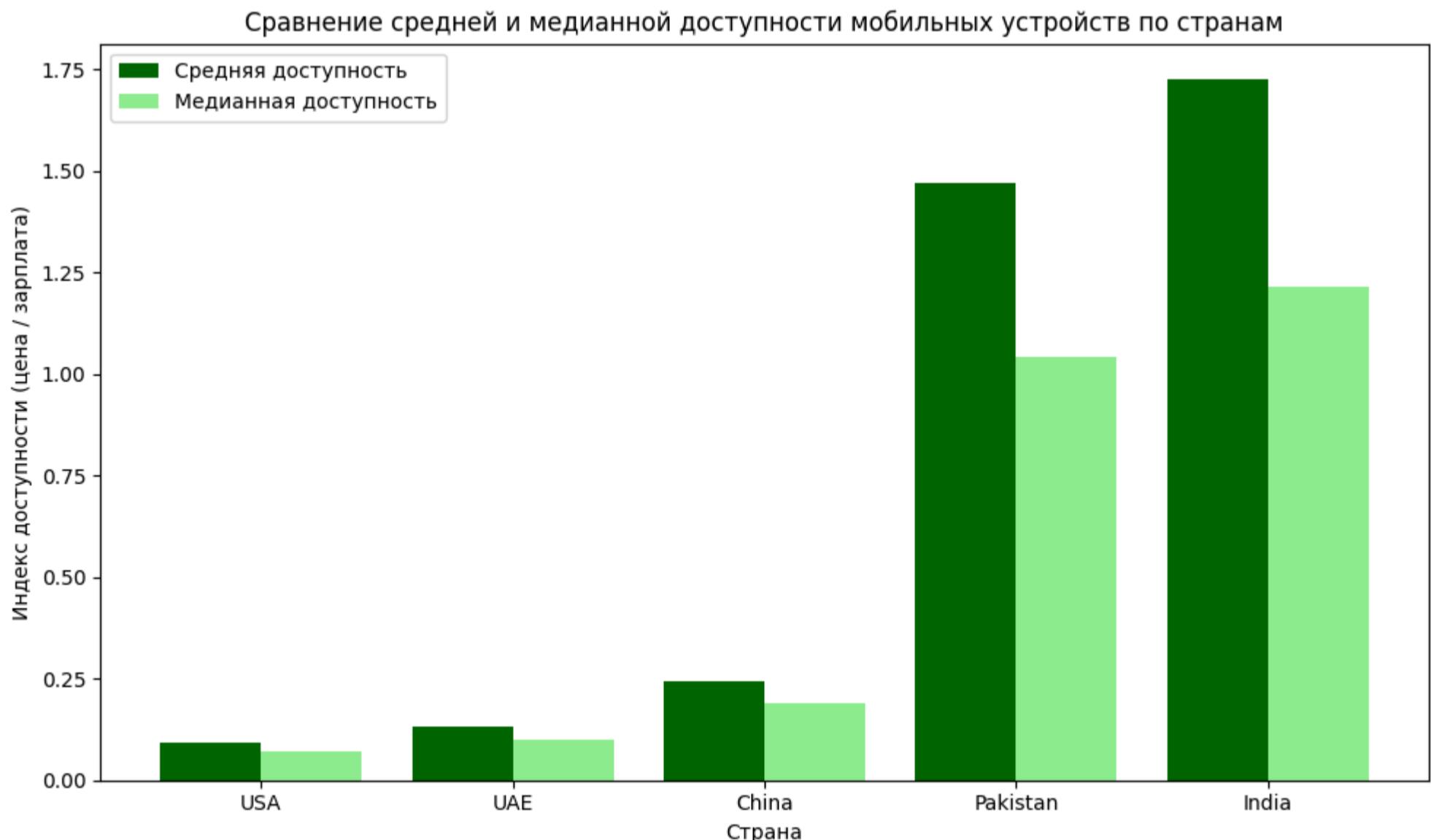
# Сортировка по средней доступности
affordability_stats = affordability_stats.sort_values(by='Средняя доступность')

# Построение grouped bar chart
x = np.arange(len(affordability_stats)) # позиции по оси X
width = 0.4 # ширина столбца

plt.figure(figsize=(10, 6))
plt.bar(x - width/2, affordability_stats['Средняя доступность'], width=width, label='Средняя доступность', color='darkgreen')
plt.bar(x + width/2, affordability_stats['Медианная доступность'], width=width, label='Медианная доступность', color='lightgreen')

plt.xticks(x, affordability_stats.index)
plt.xlabel("Страна")
plt.ylabel("Индекс доступности (цена / зарплата)")
plt.title("Сравнение средней и медианной доступности мобильных устройств по странам")
plt.legend()
plt.tight_layout()
plt.show()

# Вывод таблицы
display(affordability_stats)
```



	Средняя доступность	Медианная доступность
USA	0.093878	0.071
UAE	0.130438	0.101
China	0.245074	0.190
Pakistan	1.468836	1.042
India	1.726642	1.215

Для потребителей — выгоднее покупать в США/ОАЭ, для перепродажи — привлекательны страны с низкими средними ценами (Пакистан/Индия/Китай), но логистика/гарантии/налоги могут нивелировать разницу.

Тепловая карта средних и медианных цен по бренду и стране

In [181...]

```
# Словарь: страна → колонка с ценой
country_price_cols = {
    'Pakistan': 'price_pakistan_usd',
    'India': 'price_india_usd',
    'China': 'price_china_usd',
    'Dubai': 'price_dubai_usd',
    'USA': 'price_usa_usd'
}

# Преобразуем в длинный формат
price_long = pd.melt(
    df,
    id_vars='company_name',
    value_vars=list(country_price_cols.values()),
    var_name='country_column',
    value_name='price_usd'
)

# Извлекаем название страны
price_long['country'] = price_long['country_column'].str.extract(r'price_(\w+)_usd')
```

In [182...]

```
# Средняя цена
pivot_mean = price_long.pivot_table(index='company_name', columns='country', values='price_usd', aggfunc='mean')
pivot_mean = pivot_mean.sort_values(by='usa', ascending=False)
print("Средняя цена по бренду и стране")
display(pivot_mean)

# Медианная цена
print("Медианная цена по бренду и стране")
pivot_median = price_long.pivot_table(index='company_name', columns='country', values='price_usd', aggfunc='median')
pivot_median = pivot_median.sort_values(by='usa', ascending=False)
display(pivot_median)
```

Средняя цена по бренду и стране

country	china	dubai	india	pakistan	usa
company_name					
Sony	837.222222	1097.888889	1035.666667	1167.777778	1132.333333
Huawei	973.454545	1139.545455	1165.045455	657.477273	1120.318182
Apple	1007.927835	1002.927835	1163.742268	882.257732	1028.484536
Google	850.571429	823.047619	794.666667	614.190476	755.190476
Samsung	726.727273	783.125000	718.636364	753.590909	748.431818
OnePlus	554.320755	668.509434	516.641509	480.358491	608.622642
Honor	472.813187	610.824176	551.901099	428.219780	607.571429
Xiaomi	484.777778	615.777778	646.851852	479.703704	559.876296
Oppo	504.234783	563.652174	525.165217	353.313043	535.347826
Tecno	428.769231	492.948718	415.846154	295.461538	471.564103
Vivo	413.511628	434.220930	404.104651	258.034884	469.465116
Motorola	379.258065	437.096774	380.629032	326.387097	433.258065
iQOO	472.333333	408.333333	497.000000	285.000000	399.000000
Lenovo	295.266667	319.066667	286.866667	224.466667	311.666667
Poco	288.437500	323.718750	257.500000	210.218750	308.437500
Realme	275.536232	270.898551	295.405797	246.391304	273.333333
Infinix	221.236364	250.509091	197.545455	158.054545	247.545455
Nokia	162.272727	179.636364	155.545455	184.636364	194.201818

Медианная цена по бренду и стране

country	china	dubai	india	pakistan	usa
company_name					
Sony	884.0	1116.0	1073.0	1211.0	1099.0
Apple	1024.0	1007.0	1130.0	873.0	999.0
Huawei	898.0	1007.0	960.0	623.5	999.0
Samsung	772.0	817.0	734.0	534.0	699.0
Google	772.0	762.0	678.0	534.0	699.0
OnePlus	533.0	680.0	508.0	534.0	649.0
Xiaomi	491.0	680.0	791.0	534.0	599.0
Oppo	449.0	517.0	452.0	285.0	499.0
Honor	351.0	463.0	395.0	303.0	449.0
Vivo	351.0	367.5	310.5	196.0	399.0
iQOO	491.0	408.0	497.0	285.0	399.0
Motorola	316.0	347.0	327.5	276.0	349.0
Tecno	323.0	326.0	282.0	214.0	329.0
Poco	288.0	299.0	249.0	205.0	289.5
Realme	253.0	245.0	260.0	221.0	250.0
Lenovo	238.0	245.0	169.0	125.0	249.0
Infinix	196.0	218.0	169.0	128.0	219.0
Nokia	154.0	171.0	141.0	175.0	169.0

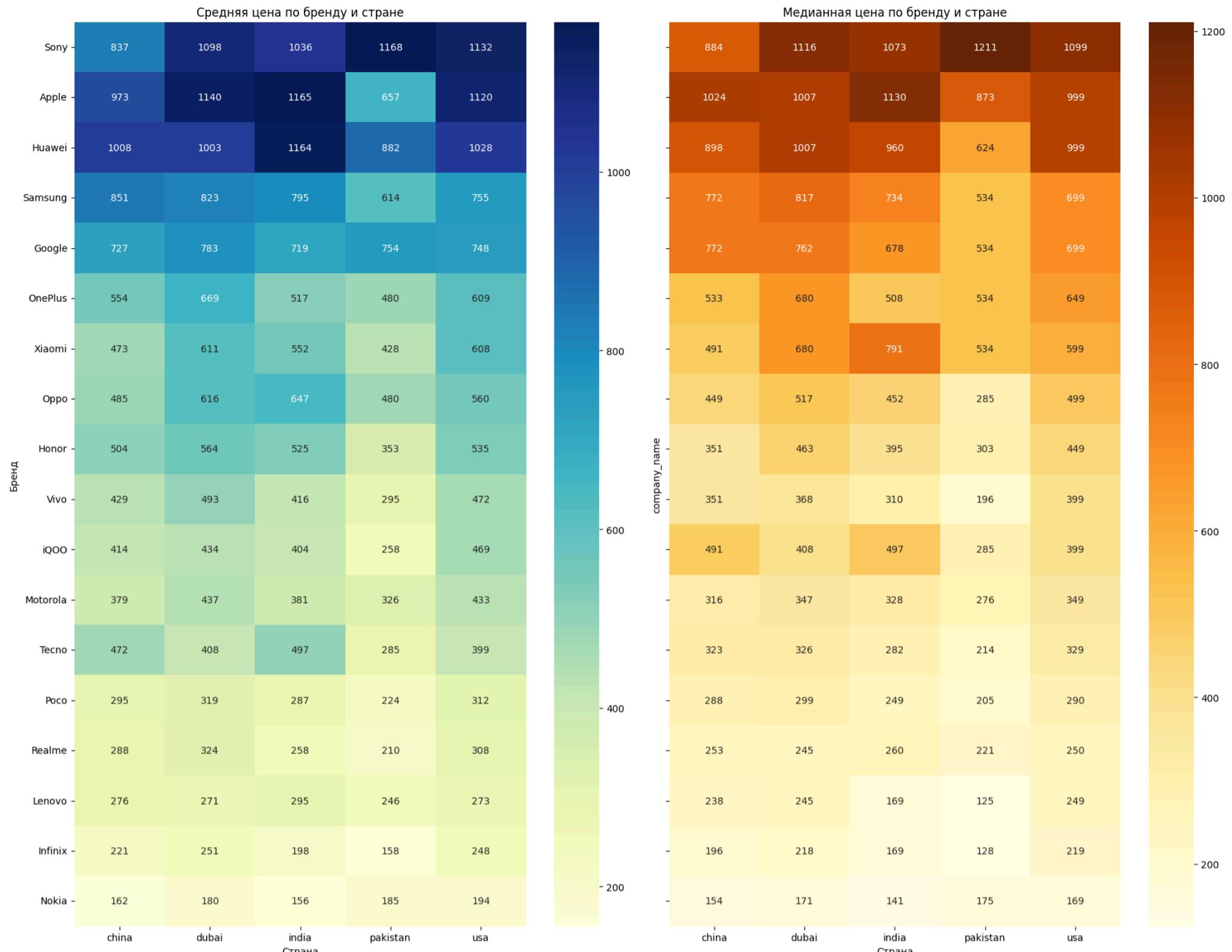
In [183...]

```
fig, axes = plt.subplots(1, 2, figsize=(18, 14), sharey=True)

sns.heatmap(pivot_mean, cmap='YlGnBu', annot=True, fmt=".0f", ax=axes[0])
axes[0].set_title("Средняя цена по бренду и стране")
axes[0].set_xlabel("Страна")
axes[0].set_ylabel("Бренд")

sns.heatmap(pivot_median, cmap='YlOrBr', annot=True, fmt=".0f", ax=axes[1])
axes[1].set_title("Медианная цена по бренду и стране")
axes[1].set_xlabel("Страна")

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



Цены в Индии и Пакистане часто ниже, особенно для локальных брендов.

Samsung, Apple, Google сохраняют высокие цены across всех рынков.

Ценовые сегменты брендов в долларах США

- Премиум (>800):
 - Sony (838-1132)
 - Huawei (974-1162)
 - Apple (1009-1161)
- Верхний средний (600-800):
 - Google (755-851)
 - Samsung (727-783)
- Средний (400-600):
 - OnePlus, Honor, Xiaomi, Oppo
- Бюджетный (<400):
 - Infinix, Nokia, Poco, Realme, Lenovo

Тепловая карта доступности по бренду и стране

In [184...]

```
# Словарь: страна → колонка доступности
affordability_cols = {
    'Pakistan': 'affordability_pakistan',
    'India': 'affordability_india',
    'China': 'affordability_china',
    'UAE': 'affordability_uae',
    'USA': 'affordability_usa'
}

# Преобразуем в длинный формат
afford_long = pd.melt(
    df,
    id_vars='company_name',
    value_vars=list(affordability_cols.values()),
    var_name='country_column',
    value_name='affordability'
)

# Извлекаем название страны
afford_long['country'] = afford_long['country_column'].str.extract(r'affordability_(\w+)')
```

In [185...]

```
# Средняя доступность
pivot_afford_mean = afford_long.pivot_table(index='company_name', columns='country', values='affordability', aggfunc='mean')
pivot_afford_mean = pivot_afford_mean.sort_values(by='usa', ascending=False)
print("Средняя доступность бренду и стране")
display(pivot_afford_mean)

# Медианная доступность
pivot_afford_median = afford_long.pivot_table(index='company_name', columns='country', values='affordability', aggfunc='median')
pivot_afford_median = pivot_afford_median.sort_values(by='usa', ascending=False)
print("Медианная доступность по бренду и стране")
display(pivot_afford_median)
```

Средняя доступность бренду и стране

	country	china	india	pakistan	uae	usa
company_name						
Sony	0.378222	3.091667	3.791556	0.238556	0.180000	
Huawei	0.439932	3.477818	2.134727	0.247636	0.178250	
Apple	0.455381	3.473835	2.864443	0.217948	0.163701	
Google	0.384190	2.372190	1.994143	0.178857	0.120000	
Samsung	0.328386	2.145148	2.446716	0.170193	0.119170	
OnePlus	0.250528	1.542170	1.559604	0.145340	0.096736	
Honor	0.213670	1.647462	1.390275	0.132714	0.096615	
Xiaomi	0.219111	1.930889	1.557481	0.133889	0.089000	
Oppo	0.227922	1.567600	1.147070	0.122522	0.085070	
Tecno	0.193872	1.241410	0.959205	0.107077	0.075077	
Vivo	0.186919	1.206267	0.837709	0.094337	0.074651	
Motorola	0.171435	1.136194	1.059613	0.094968	0.069032	
iQOO	0.213333	1.484000	0.925000	0.089000	0.063333	
Lenovo	0.133467	0.856267	0.728733	0.069133	0.049600	
Poco	0.130344	0.768625	0.682406	0.070281	0.049063	
Realme	0.124580	0.881783	0.799870	0.058855	0.043478	
Infinix	0.100073	0.589600	0.513055	0.054345	0.039291	
Nokia	0.073364	0.464364	0.599364	0.039091	0.030818	

Медианная доступность по бренду и стране

	country	china	india	pakistan	uae	usa
company_name						
Sony	0.3990	3.2030	3.9320	0.243	0.1750	
Apple	0.4630	3.3730	2.8340	0.219	0.1590	
Huawei	0.4055	2.8660	2.0245	0.219	0.1590	
Samsung	0.3490	2.1910	1.7340	0.178	0.1110	
Google	0.3490	2.0240	1.7340	0.166	0.1110	
OnePlus	0.2410	1.5160	1.7340	0.148	0.1030	
Xiaomi	0.2220	2.3610	1.7340	0.148	0.0950	
Oppo	0.2030	1.3490	0.9250	0.112	0.0790	
Honor	0.1590	1.1790	0.9840	0.101	0.0710	
Vivo	0.1590	0.9265	0.6360	0.080	0.0630	
iQOO	0.2220	1.4840	0.9250	0.089	0.0630	
Motorola	0.1430	0.9775	0.8960	0.075	0.0560	
Tecno	0.1460	0.8420	0.6950	0.071	0.0520	
Poco	0.1300	0.7430	0.6655	0.065	0.0465	
Realme	0.1140	0.7760	0.7180	0.053	0.0400	
Lenovo	0.1080	0.5040	0.4060	0.053	0.0400	
Infinix	0.0890	0.5040	0.4160	0.047	0.0350	
Nokia	0.0700	0.4210	0.5680	0.037	0.0270	

In [186...]

```
fig, axes = plt.subplots(1, 2, figsize=(18, 14), sharey=True)

sns.heatmap(pivot_afford_mean, cmap='Greens', annot=True, fmt=".2f", ax=axes[0])
axes[0].set_title("Средняя доступность по бренду и стране")
axes[0].set_xlabel("Страна")
```

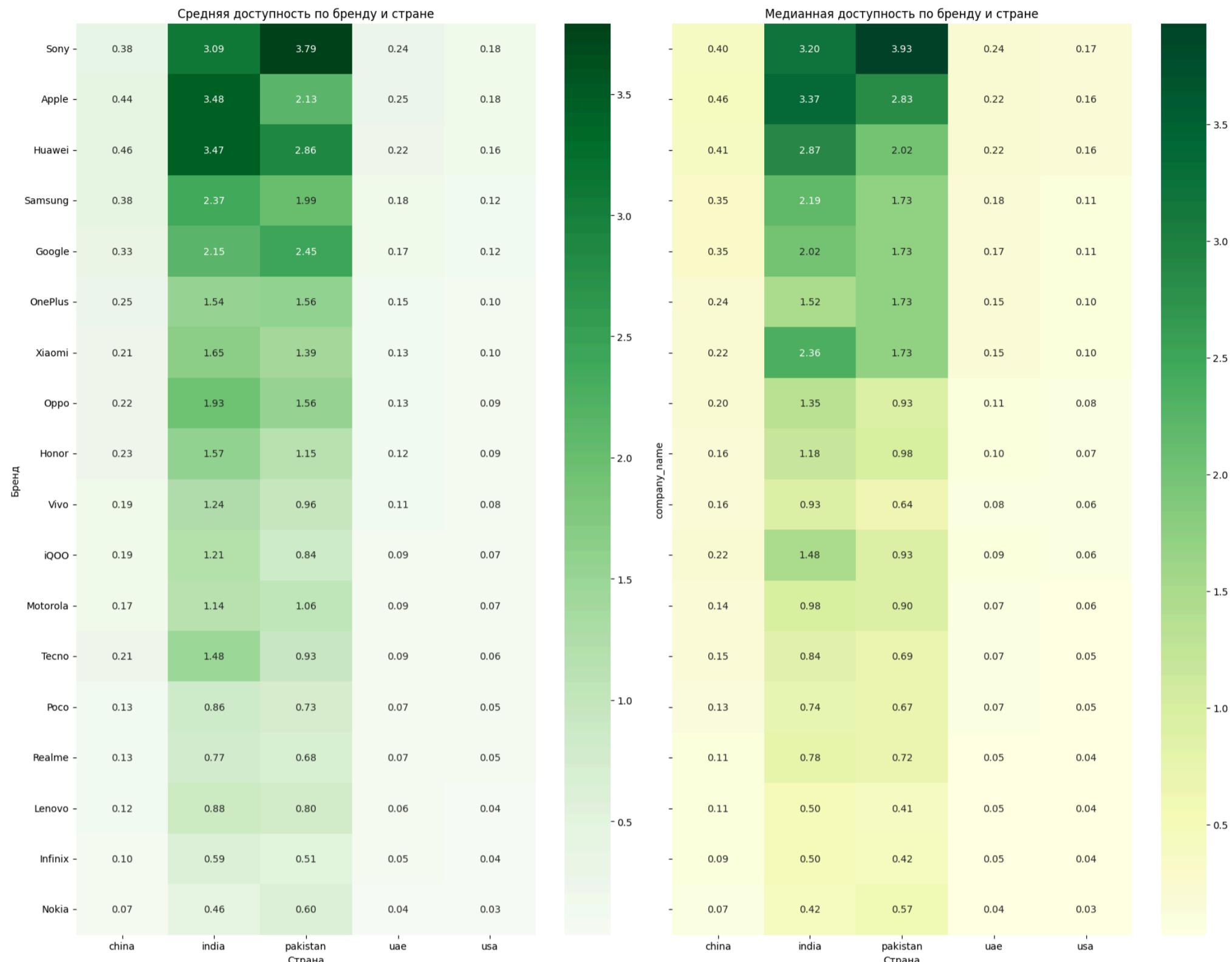
```

axes[0].set_ylabel("Бренд")

sns.heatmap(pivot_afford_median, cmap='YlGn', annot=True, fmt=".2f", ax=axes[1])
axes[1].set_title("Медианная доступность по бренду и стране")
axes[1].set_xlabel("Страна")

plt.tight_layout()
plt.show()

```



Доступность по брендам и странам

- Infinix, Nokia, Realme - лидеры доступности во всех регионах
- Sony, Huawei, Apple - наименее доступные бренды

В развивающихся странах разрыв в доступности между брендами достигает 10 раз

Средняя цена мобильных устройств по брендам для каждой страны

```

In [187...]
brand_price = (
    df.groupby('company_name')[['price_usa_usd','price_india_usd','price_china_usd','price_pakistan_usd','price_dubai_usd']]
    .mean()
    .round(2)
    .sort_values('price_usa_usd', ascending=False)
)
display(brand_price)

plt.figure(figsize=(12,6))
sns.barplot(y=brand_price.index, x=brand_price['price_usa_usd'])
plt.title('Средняя цена мобильных устройств по брендам (в USD, рынок США)')
plt.xlabel('Средняя цена, USD')
plt.ylabel('Бренд')
plt.show()

plt.figure(figsize=(12,6))
sns.barplot(y=brand_price.index, x=brand_price['price_india_usd'])
plt.title('Средняя цена мобильных устройств по брендам (в USD, рынок Индия)')
plt.xlabel('Средняя цена, USD')
plt.ylabel('Бренд')
plt.show()

plt.figure(figsize=(12,6))
sns.barplot(y=brand_price.index, x=brand_price['price_china_usd'])
plt.title('Средняя цена мобильных устройств по брендам (в USD, рынок Китай)')
plt.xlabel('Средняя цена, USD')

```

```

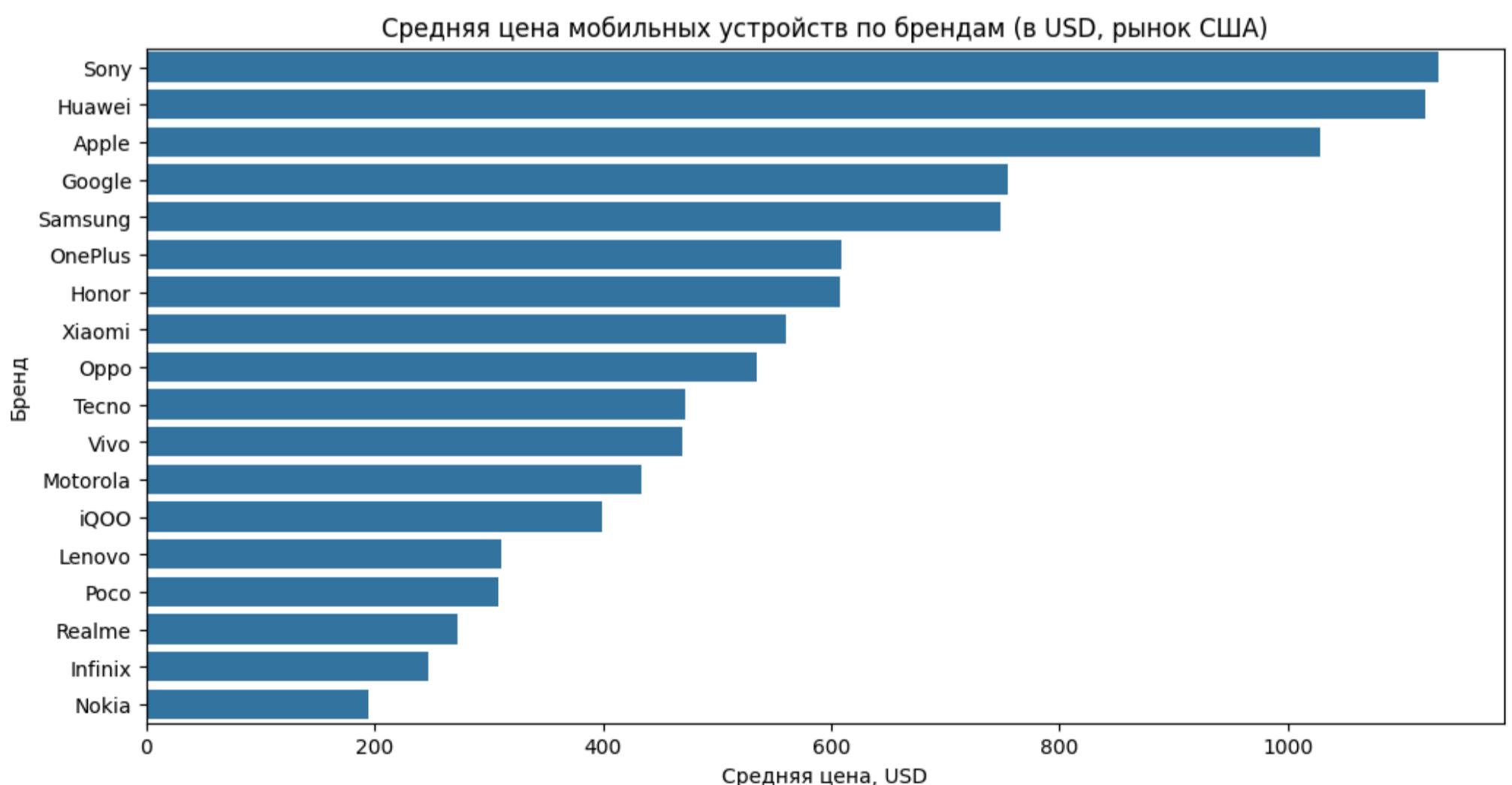
plt.ylabel('Бренд')
plt.show()

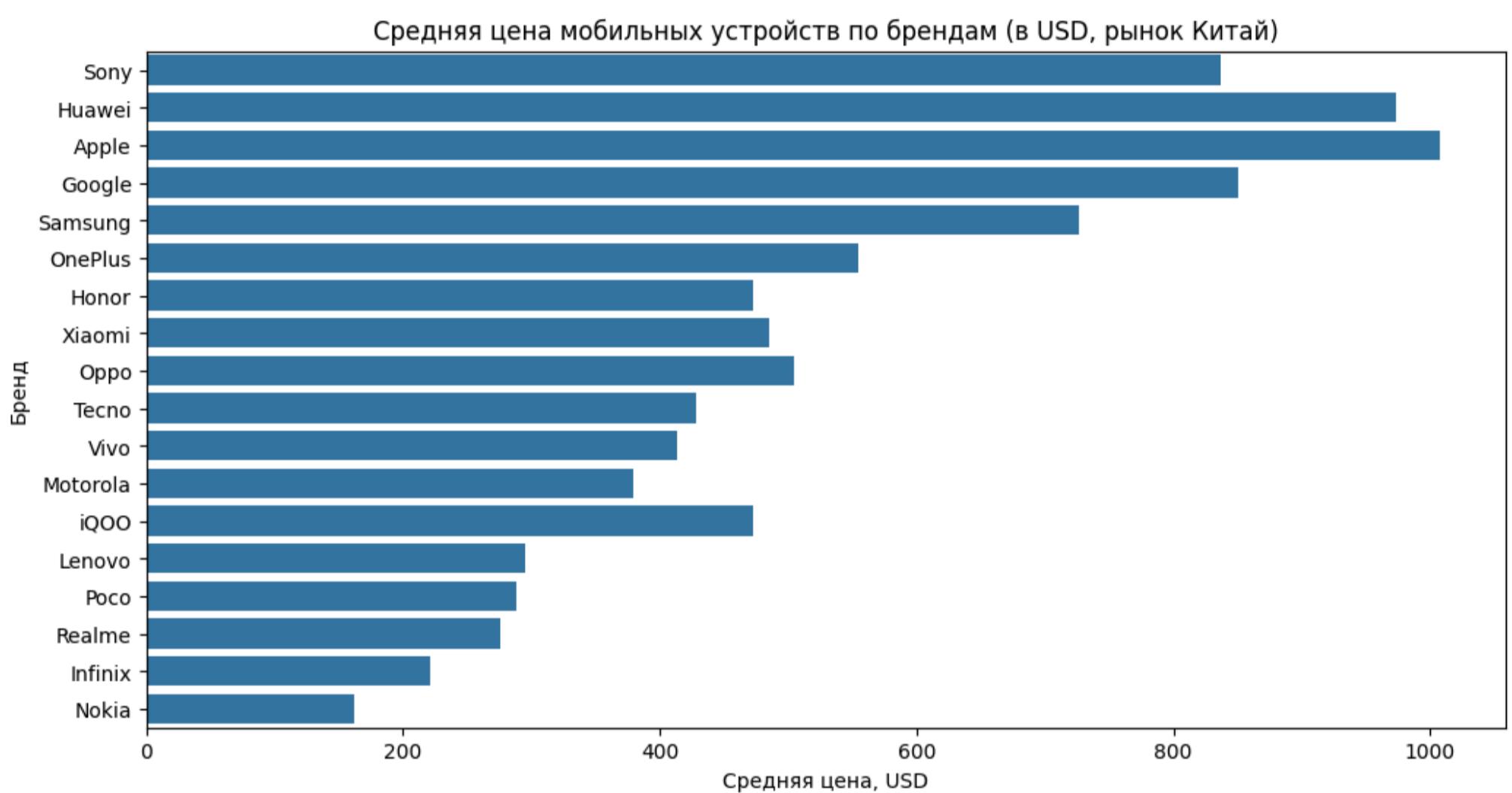
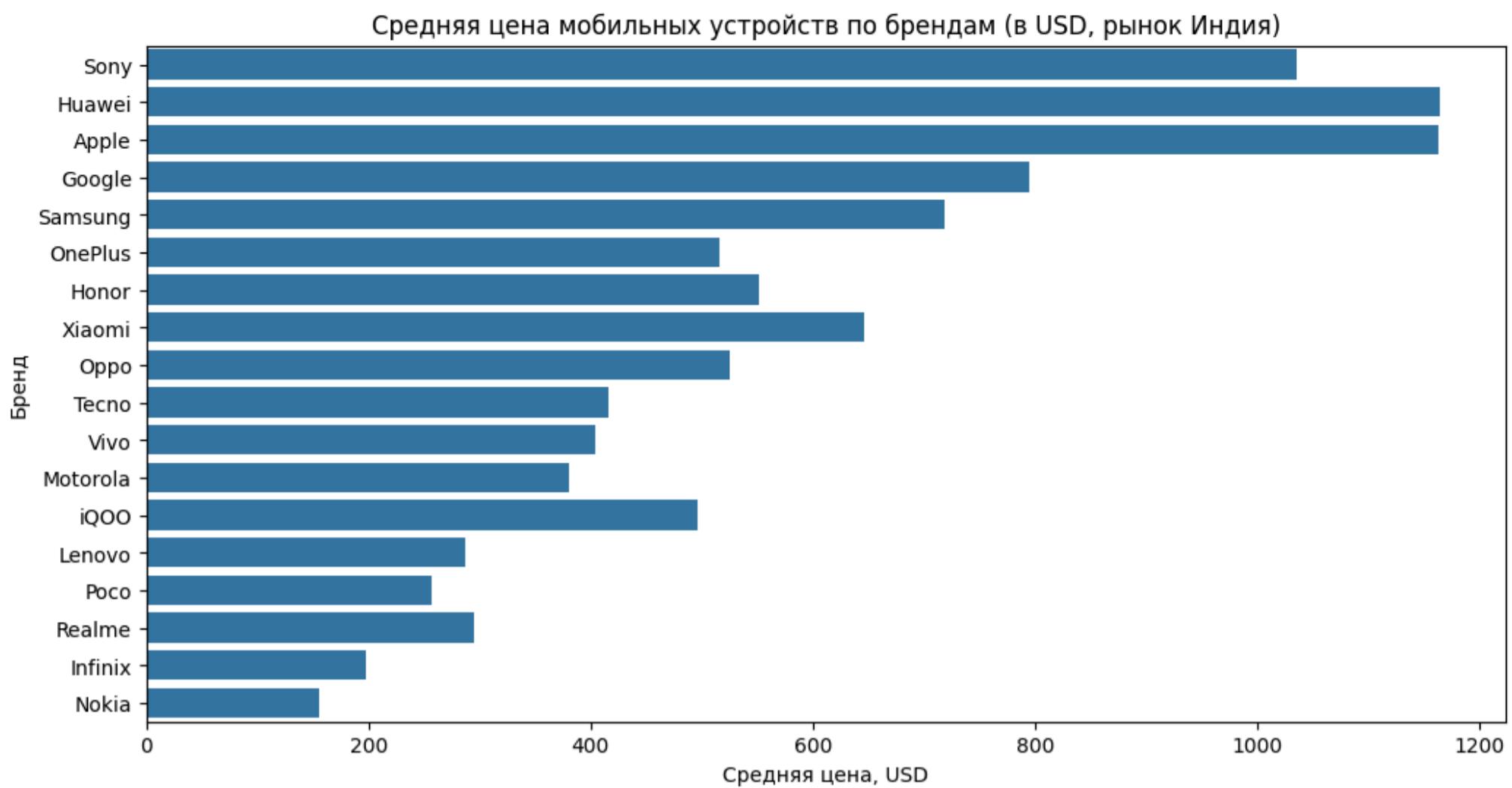
plt.figure(figsize=(12,6))
sns.barplot(y=brand_price.index, x=brand_price['price_pakistan_usd'])
plt.title('Средняя цена мобильных устройств по брендам (в USD, рынок Пакистан)')
plt.xlabel('Средняя цена, USD')
plt.ylabel('Бренд')
plt.show()

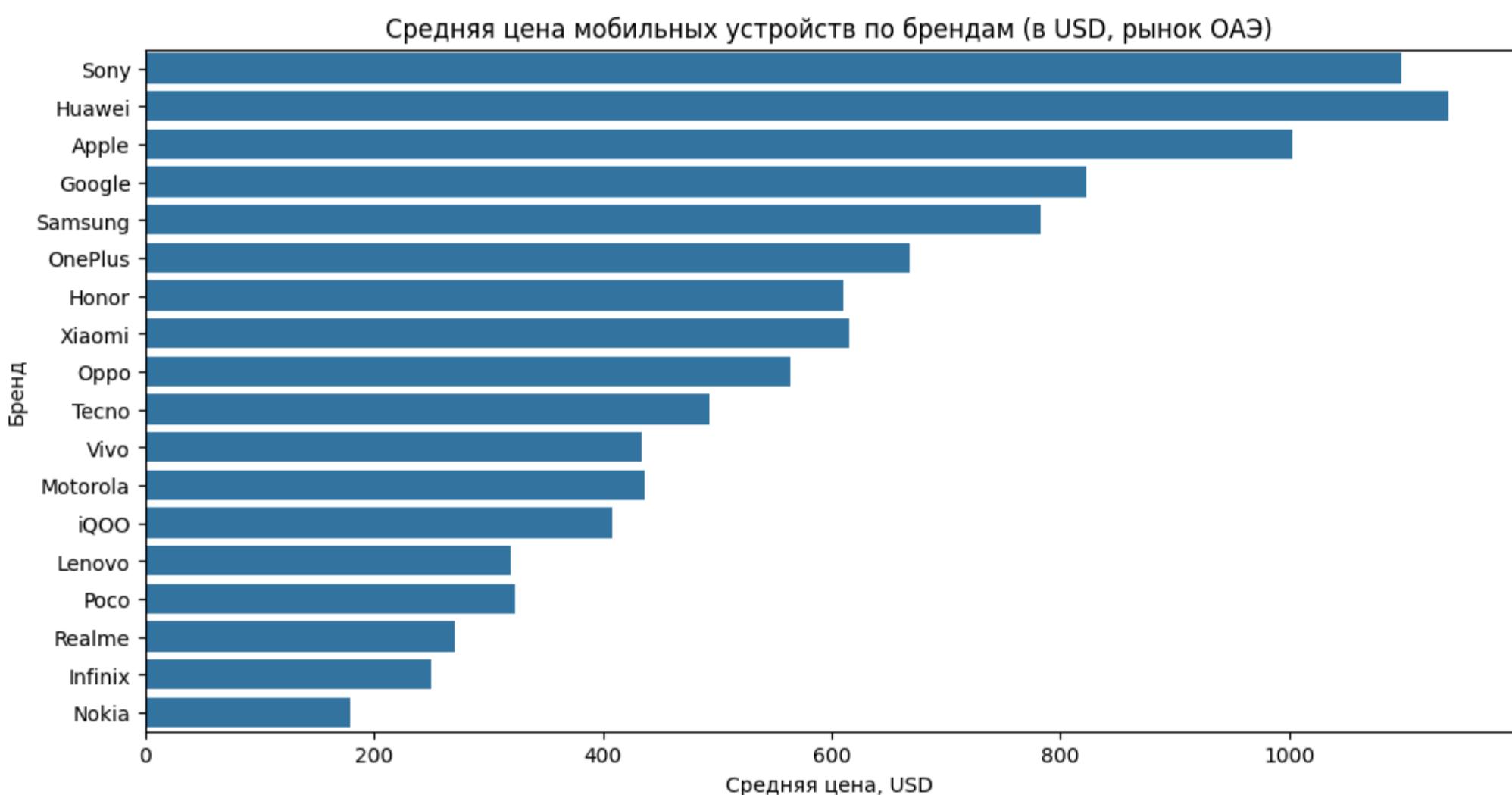
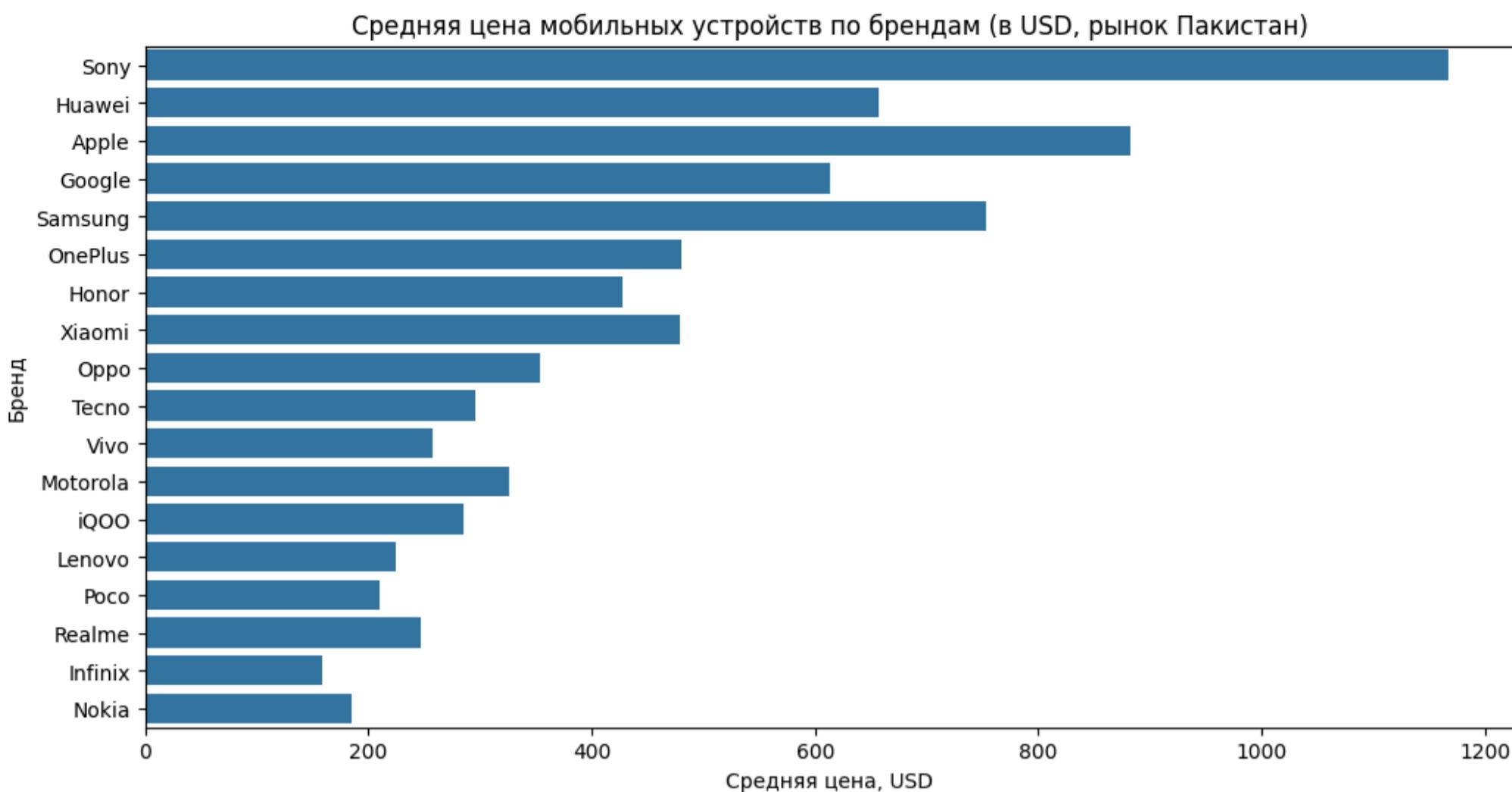
plt.figure(figsize=(12,6))
sns.barplot(y=brand_price.index, x=brand_price['price_dubai_usd'])
plt.title('Средняя цена мобильных устройств по брендам (в USD, рынок ОАЭ)')
plt.xlabel('Средняя цена, USD')
plt.ylabel('Бренд')
plt.show()

```

company_name	price_usa_usd	price_india_usd	price_china_usd	price_pakistan_usd	price_dubai_usd
Sony	1132.33	1035.67	837.22	1167.78	1097.89
Huawei	1120.32	1165.05	973.45	657.48	1139.55
Apple	1028.48	1163.74	1007.93	882.26	1002.93
Google	755.19	794.67	850.57	614.19	823.05
Samsung	748.43	718.64	726.73	753.59	783.12
OnePlus	608.62	516.64	554.32	480.36	668.51
Honor	607.57	551.90	472.81	428.22	610.82
Xiaomi	559.88	646.85	484.78	479.70	615.78
Oppo	535.35	525.17	504.23	353.31	563.65
Tecno	471.56	415.85	428.77	295.46	492.95
Vivo	469.47	404.10	413.51	258.03	434.22
Motorola	433.26	380.63	379.26	326.39	437.10
iQOO	399.00	497.00	472.33	285.00	408.33
Lenovo	311.67	286.87	295.27	224.47	319.07
Poco	308.44	257.50	288.44	210.22	323.72
Realme	273.33	295.41	275.54	246.39	270.90
Infinix	247.55	197.55	221.24	158.05	250.51
Nokia	194.20	155.55	162.27	184.64	179.64







Бренды: ценовые уровни и доступность

- Самые дорогие бренды (средние цены): Sony, Huawei, Apple — стабильно высокие средние по некоторым рынкам; далее Samsung, Google. Это «премиумы» с сильным позиционированием.
- Средний уровень: OnePlus, Honor, Xiaomi, Oppo, Tecno, Vivo — широкий спектр от мидрейнда до «доступных флагманов».

Самые доступные бренды (affordability по странам):

- США/ОАЭ/Китай: Nokia, Infinix, Realme, Poco, Lenovo — минимальные индексы доступности.
- Индия/Пакистан: те же бренды в топе по доступности, отражая их локальную стратегию — агрессивные ценники, оптимальная комплектация.

Потребителю: Лучшая «цена/характеристики» — Infinix/Poco/Realme/Lenovo/Xiaomi (в зависимости от рынка).

Премиум-ориентированным: Apple/Samsung/Huawei/Sony — прогнозируемое качество экрана/SoC/камер, но часто без лучшей «RAM/\$».

Средние характеристики (RAM, мощность аккумулятора, размер дисплея и хара-ка камеры) по ценовым категориям

In [188...]

```
# Группировка по ценовым категориям
price_cat_stats = (
    df.groupby("price_category") [["ram_gb", "battery_capacity_mah", "screen_size_inches", "back_camera_1"]]
    .mean()
    .round(2)
)

print("Средние характеристики по ценовым категориям:")
display(price_cat_stats)
```

Средние характеристики по ценовым категориям:

ram_gb battery_capacity_mah screen_size_inches back_camera_1

price_category

Бюджетный	5.49	5011.88	6.93	38.67
Средний	8.51	5143.36	7.14	54.29
Флагманский	9.05	4881.10	7.14	42.23
Ультрапремиум	10.77	5052.46	7.44	58.37

Флагманы имеют меньшую батарею (4881 mAh), чем бюджетники (5012 mAh) и средний сегмент (5143 mAh)!

Средние характеристики по ценовым категориям подтверждают: чем выше ценовой сегмент, тем больше RAM и размер экрана

Сегменты цены:

- Бюджетный: RAM ≈ 5.5 ГБ, экран ~6.93", камера ~39 МП; справедливая «база» для массового рынка.
- Средний: RAM ≈ 8.5 ГБ, экран ~7.14", камера ~54 МП; оптимальный баланс цены/характеристик.
- Флагманский: RAM ≈ 9.0 ГБ, экран ~7.14", на деле часто лучше по SoC/экранам, но камера головная не всегда выше среднего (важна конфигурация модулей).
- Ультрапремиум: RAM ≈ 10.8 ГБ, экран ~7.44", камера ~58 МП; максимизация характеристик, нередко складные/имиджевые.

ТОП-10 самых дорогих/дешёвых моделей мобильных устройств для каждой страны

In [189...]

```
# Словарь: страна → колонка с ценой
country_price_cols = {
    'Pakistan': 'price_pakistan_usd',
    'India': 'price_india_usd',
    'China': 'price_china_usd',
    'Dubai': 'price_dubai_usd',
    'USA': 'price_usa_usd'
}

# Цикл по странам
for country, col in country_price_cols.items():
    print(f"\nТОП-10 самых дорогих моделей ({country}):")
    top10_expensive = df.nlargest(10, col)[["company_name", "model_name", col]]
    display(top10_expensive.rename(columns={col: "price_usd"}))

    print(f"\nТОП-10 самых дешёвых моделей ({country}):")
    top10 Cheap = df.nsmallest(10, col)[["company_name", "model_name", col]]
    display(top10 Cheap.rename(columns={col: "price_usd"}))
```

ТОП-10 самых дорогих моделей (Pakistan):

	company_name	model_name	price_usd
912	Samsung	Galaxy Z Fold6 256GB	2156.0
116	Samsung	Galaxy Z Fold 5 512GB	1960.0
913	Samsung	Galaxy Z Fold6 512GB	1942.0
120	Samsung	Galaxy Z Fold 4 512GB	1888.0
115	Samsung	Galaxy Z Fold 5 256GB	1782.0
98	Samsung	Galaxy S24 Ultra 256GB	1710.0
119	Samsung	Galaxy Z Fold 4 256GB	1710.0
668	Sony	Xperia 1 VI 256GB	1675.0
104	Samsung	Galaxy S23 Ultra 256GB	1639.0
97	Samsung	Galaxy S24 Ultra 128GB	1603.0

ТОП-10 самых дешёвых моделей (Pakistan):

	company_name	model_name	price_usd
782	Infinix	Smart HD 32GB	57.0
265	Vivo	Y11 32GB	66.0
261	Vivo	Y12s 64GB	68.0
788	Infinix	Smart 5 64GB	68.0
255	Vivo	X3S 16GB	71.0
272	Vivo	Y30 64GB	71.0
419	Oppo	A49 5G 128GB	71.0
541	Lenovo	K13 32GB	71.0
544	Lenovo	A6 Note 32GB	71.0
777	Infinix	Hot 10 Lite 64GB	71.0

ТОП-10 самых дорогих моделей (India):

	company_name	model_name	price_usd
647	Huawei	Mate XT 512GB	3107.0
646	Huawei	Mate XT 256GB	2937.0
616	Huawei	Mate X2	2824.0
629	Huawei	Mate X3	2824.0
643	Huawei	Mate X6	2824.0
914	Samsung	Galaxy Z Fold6 1TB	2271.0
620	Huawei	Mate Xs 2	2260.0
96	Apple	iPad Pro 2TB	2258.0
694	Google	Pixel 9 Pro Fold 512GB	2034.0
95	Apple	iPad Pro 1TB	2033.0

ТОП-10 самых дешёвых моделей (India):

	company_name	model_name	price_usd
782	Infinix	Smart HD 32GB	68.0
702	Tecno	Pop 9 64GB	79.0
724	Tecno	Pop 8 64GB	79.0
541	Lenovo	K13 32GB	85.0
697	Tecno	Pop 9 4G 64GB	85.0
709	Tecno	Spark Go 1 64GB	85.0
723	Tecno	Spark Go 2024 64GB	85.0
788	Infinix	Smart 5 64GB	85.0
777	Infinix	Hot 10 Lite 64GB	90.0
181	Samsung	Galaxy Tab E 8.0 16GB	96.0

ТОП-10 самых дорогих моделей (China):

	company_name	model_name	price_usd
616	Huawei	Mate X2	2526.0
914	Samsung	Galaxy Z Fold6 1TB	2526.0
913	Samsung	Galaxy Z Fold6 512GB	2245.0
620	Huawei	Mate Xs 2	2105.0
647	Huawei	Mate XT 512GB	2035.0
629	Huawei	Mate X3	1965.0
694	Google	Pixel 9 Pro Fold 512GB	1965.0
912	Samsung	Galaxy Z Fold6 256GB	1965.0
646	Huawei	Mate XT 256GB	1894.0
116	Samsung	Galaxy Z Fold 5 512GB	1824.0

ТОП-10 самых дешёвых моделей (China):

	company_name	model_name	price_usd
782	Infinix	Smart HD 32GB	70.0
702	Tecno	Pop 9 64GB	77.0
724	Tecno	Pop 8 64GB	77.0
697	Tecno	Pop 9 4G 64GB	84.0
709	Tecno	Spark Go 1 64GB	84.0
723	Tecno	Spark Go 2024 64GB	84.0
788	Infinix	Smart 5 64GB	84.0
777	Infinix	Hot 10 Lite 64GB	98.0
181	Samsung	Galaxy Tab E 8.0 16GB	112.0
659	Nokia	C22 64GB	112.0

ТОП-10 самых дорогих моделей (Dubai):

	company_name	model_name	price_usd
647	Huawei	Mate XT 512GB	3022.0
646	Huawei	Mate XT 256GB	2858.0
616	Huawei	Mate X2	2722.0
629	Huawei	Mate X3	2722.0
643	Huawei	Mate X6	2722.0
620	Huawei	Mate Xs 2	2450.0
914	Samsung	Galaxy Z Fold6 1TB	2368.0
913	Samsung	Galaxy Z Fold6 512GB	2096.0
116	Samsung	Galaxy Z Fold 5 512GB	2042.0
912	Samsung	Galaxy Z Fold6 256GB	1960.0

ТОП-10 самых дешёвых моделей (Dubai):

	company_name	model_name	price_usd
782	Infinix	Smart HD 32GB	81.0
702	Tecno	Pop 9 64GB	95.0
724	Tecno	Pop 8 64GB	95.0
788	Infinix	Smart 5 64GB	95.0
181	Samsung	Galaxy Tab E 8.0 16GB	109.0
697	Tecno	Pop 9 4G 64GB	109.0
709	Tecno	Spark Go 1 64GB	109.0
723	Tecno	Spark Go 2024 64GB	109.0
777	Infinix	Hot 10 Lite 64GB	109.0
659	Nokia	C22 64GB	122.0

ТОП-10 самых дорогих моделей (USA):

	company_name	model_name	price_usd
647	Huawei	Mate XT 512GB	2799.0
616	Huawei	Mate X2	2699.0
646	Huawei	Mate XT 256GB	2599.0
620	Huawei	Mate Xs 2	2499.0
629	Huawei	Mate X3	2499.0
643	Huawei	Mate X6	2499.0
914	Samsung	Galaxy Z Fold6 1TB	2259.0
116	Samsung	Galaxy Z Fold 5 512GB	1899.0
120	Samsung	Galaxy Z Fold 4 512GB	1899.0
843	Honor	Magic V2	1899.0

ТОП-10 самых дешёвых моделей (USA):

	company_name	model_name	price_usd
782	Infinix	Smart HD 32GB	79.0
702	Tecno	Pop 9 64GB	89.0
724	Tecno	Pop 8 64GB	89.0
181	Samsung	Galaxy Tab E 8.0 16GB	99.0
697	Tecno	Pop 9 4G 64GB	99.0
709	Tecno	Spark Go 1 64GB	99.0
723	Tecno	Spark Go 2024 64GB	99.0
788	Infinix	Smart 5 64GB	99.0
777	Infinix	Hot 10 Lite 64GB	109.0
659	Nokia	C22 64GB	119.0

ТОП-5 брендов мобильных устройств по средней цене для каждой страны

In [190...]

```
# ТОП-5 брендов по средней цене в США
top5_brands_price = brand_price.head(5)
print("ТОП-5 брендов по средней цене (usd):")
display(top5_brands_price)
```

ТОП-5 брендов по средней цене (usd):

	price_usa_usd	price_india_usd	price_china_usd	price_pakistan_usd	price_dubai_usd
company_name					
Sony	1132.33	1035.67	837.22	1167.78	1097.89
Huawei	1120.32	1165.05	973.45	657.48	1139.55
Apple	1028.48	1163.74	1007.93	882.26	1002.93
Google	755.19	794.67	850.57	614.19	823.05
Samsung	748.43	718.64	726.73	753.59	783.12

Самые дорогие бренды (в среднем):

- Sony
- Huawei
- Apple
- Google
- Samsung

ТОП-5 брендов мобильных устройств по доступности для каждой страны

In [191...]

```
# Словарь: страна → колонка доступности
affordability_cols = {
    'Pakistan': 'affordability_pakistan',
    'India': 'affordability_india',
    'China': 'affordability_china',
    'UAE': 'affordability_uae',
    'USA': 'affordability_usa'
}

# Цикл по странам
for country, col in affordability_cols.items():
    brand_afford = (
        df.groupby("company_name")[col]
        .mean()
        .sort_values()
    )

    top5_affordable = brand_afford.head(5)
    print(f"\nТОП-5 брендов по доступности ({country}):")
    display(top5_affordable)
```

ТОП-5 брендов по доступности (Pakistan):

```
company_name
Infinix      0.513055
Nokia        0.599364
Poco          0.682406
Lenovo        0.728733
Realme        0.799870
Name: affordability_pakistan, dtype: float64
```

ТОП-5 брендов по доступности (India):

```
company_name
Nokia        0.464364
Infinix      0.589600
Poco          0.768625
Lenovo        0.856267
Realme        0.881783
Name: affordability_india, dtype: float64
```

ТОП-5 брендов по доступности (China):

```
company_name
Nokia        0.073364
Infinix      0.100073
Realme        0.124580
Poco          0.130344
Lenovo        0.133467
Name: affordability_china, dtype: float64
```

ТОП-5 брендов по доступности (UAE):

```
company_name
Nokia        0.039091
Infinix      0.054345
Realme        0.058855
Lenovo        0.069133
Poco          0.070281
Name: affordability_uae, dtype: float64
```

ТОП-5 брендов по доступности (USA):

```
company_name
Nokia        0.030818
Infinix      0.039291
Realme        0.043478
Poco          0.049063
Lenovo        0.049600
Name: affordability_usa, dtype: float64
```

Самые доступные бренды:

- Infinix, Nokia, Poco, Lenovo, Realme

Вывод: Эти бренды ориентированы на развивающиеся рынки (Индия, Африка, Юго-Восточная Азия)

Распределение брендов мобильных устройств по годам и брендам

In [192...]

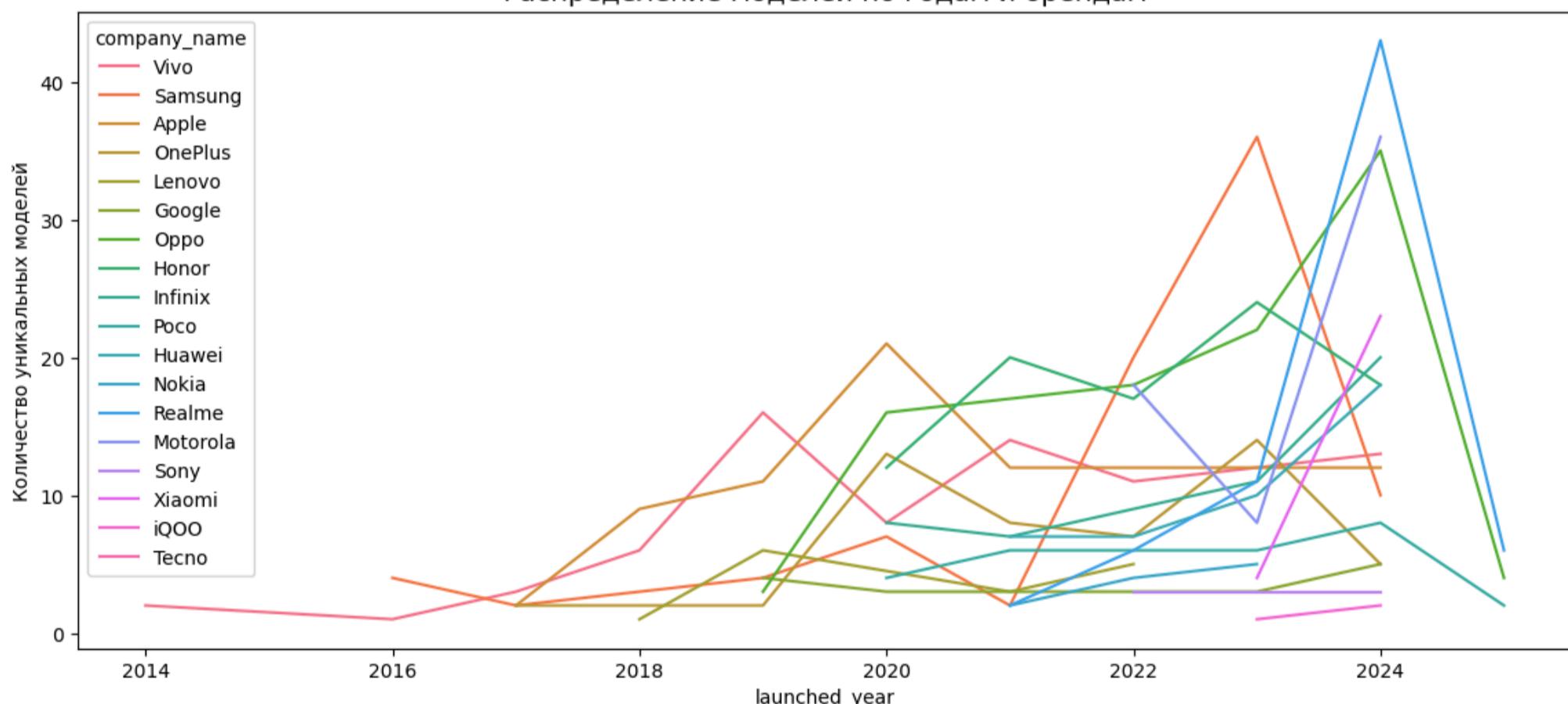
```
# Количество моделей по годам и брендам
models_by_year_brand = (
    df.groupby(["launched_year", "company_name"])["model_name"]
    .nunique()
    .reset_index()
)

display(models_by_year_brand)
plt.figure(figsize=(14,6))
sns.lineplot(data=models_by_year_brand, x="launched_year", y="model_name", hue="company_name")
plt.title("Распределение моделей по годам и брендам", fontsize=14)
plt.ylabel("Количество уникальных моделей")
plt.show()
```

	launched_year	company_name	model_name
0	2014	Vivo	2
1	2016	Samsung	4
2	2016	Vivo	1
3	2017	Apple	2
4	2017	OnePlus	2
...
86	2024	Xiaomi	23
87	2024	iQOO	2
88	2025	Oppo	4
89	2025	Poco	2
90	2025	Realme	6

91 rows × 3 columns

Распределение моделей по годам и брендам



Распределение моделей по годам:

- Пик выпуска: 2024 год, что отражает обновление модельных рядов после пандемии
- Лидеры по активности: Oppo, Samsung, Apple, Xiaomi

Средняя цена мобильных устройств по странам с группировкой по годам

In [193...]

```
# Средняя цена по странам и годам
avg_price_country_year = (
    df.groupby("launched_year")[[ "price_usa_usd", "price_india_usd", "price_china_usd", "price_pakistan_usd", "price_dubai_usd"]]
    .mean()
    .reset_index()
)

display(avg_price_country_year )
plt.figure(figsize=(14,6))
for col in ["price_usa_usd", "price_india_usd", "price_china_usd", "price_pakistan_usd", "price_dubai_usd"]:
    sns.lineplot(data=avg_price_country_year, x="launched_year", y=col, label=col.replace("price_","").replace("_usd","").title())
plt.title("Средняя цена мобильных устройств по странам и годам", fontsize=14)
```

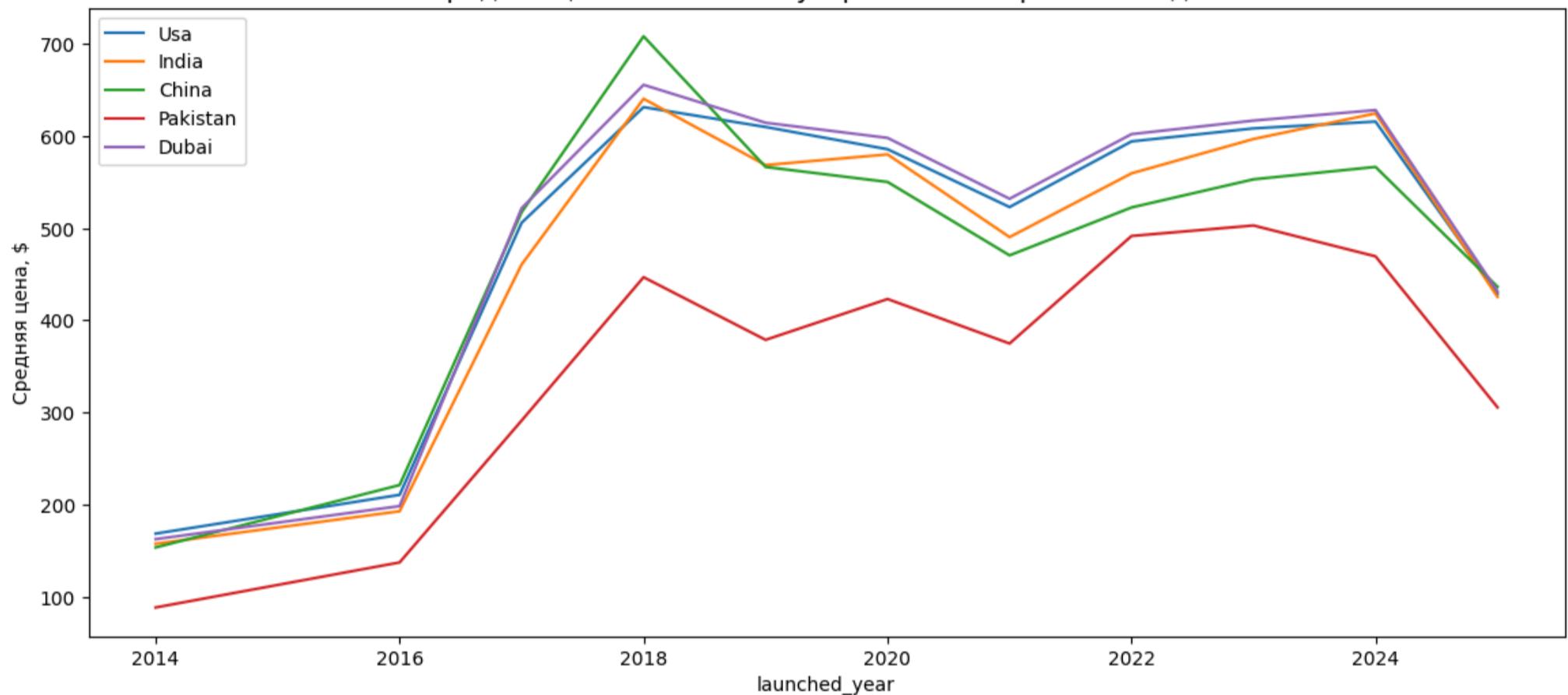
```

plt.ylabel("Средняя цена, $")
plt.legend()
plt.show()

```

	launched_year	price_usa_usd	price_india_usd	price_china_usd	price_pakistan_usd	price_dubai_usd
0	2014	169.000000	158.000000	154.000000	89.000000	163.000000
1	2016	211.000000	193.200000	221.600000	137.800000	198.800000
2	2017	505.666667	460.444444	517.111111	291.444444	521.444444
3	2018	630.904762	640.000000	707.666667	446.714286	655.095238
4	2019	609.434783	568.173913	566.130435	378.869565	614.043478
5	2020	585.326531	579.724490	549.897959	423.091837	597.612245
6	2021	522.708738	490.155340	470.398058	374.902913	531.669903
7	2022	593.816575	559.226027	522.267123	491.383562	601.698630
8	2023	607.918478	596.413043	552.755435	502.750000	616.375000
9	2024	615.294364	624.151203	566.192440	469.319588	627.721649
10	2025	428.666667	425.416667	436.333333	305.750000	431.500000

Средняя цена мобильных устройств по странам и годам



Цены постепенно растут во всех регионах; наибольший рост наблюдается в США и Дубае, что отражает премиальный фокус рынков. В Индии и Китае динамика умеренная — там сильная конкуренция в среднем сегменте.

Анализ цен для перепродажи в другой стране

```

# 📈 Анализ арбитража для перепродажи
price_cols = ['price_pakistan_usd', 'price_india_usd', 'price_china_usd',
              'price_dubai_usd', 'price_usa_usd']

df['min_price'] = df[price_cols].min(axis=1)
df['max_price'] = df[price_cols].max(axis=1)
df['price_spread'] = df['max_price'] - df['min_price']
df['price_spread_pct'] = (df['price_spread'] / df['min_price']) * 100).round(1)

# Определяем страны покупки/продажи
df['best_buy_country'] = df[price_cols].idxmin(axis=1).str.replace('price_', '').str.replace('_usd', '')
df['best_sell_country'] = df[price_cols].idxmax(axis=1).str.replace('price_', '').str.replace('_usd', '')

In [195...]: df.iloc[:,list(range(23, 28)) + list(range(38, 41))]

```

	price_usa_usd	price_pakistan_usd	price_india_usd	price_china_usd	price_dubai_usd	min_price	max_price	price_spread
0	799.0	802.0	904.0	814.0	762.0	762.0	904.0	142.0
1	849.0	837.0	960.0	856.0	817.0	817.0	960.0	143.0
2	899.0	873.0	1017.0	912.0	871.0	871.0	1017.0	146.0
3	899.0	891.0	1017.0	870.0	871.0	870.0	1017.0	147.0
4	949.0	926.0	1073.0	912.0	925.0	912.0	1073.0	161.0
...
912	1899.0	2156.0	1864.0	1965.0	1960.0	1864.0	2156.0	292.0
913	1719.0	1942.0	2000.0	2245.0	2096.0	1719.0	2245.0	526.0
914	2259.0	1237.0	2271.0	2526.0	2368.0	1237.0	2526.0	1289.0
915	1099.0	677.0	1130.0	1121.0	1143.0	677.0	1143.0	466.0
916	1299.0	784.0	1356.0	1261.0	1307.0	784.0	1356.0	572.0

917 rows × 8 columns

```
In [196...]: df.iloc[:, list(range(23, 28)) + list(range(41, 44))]
```

	price_usa_usd	price_pakistan_usd	price_india_usd	price_china_usd	price_dubai_usd	price_spread_pct	best_buy_country	best_sell_country
0	799.0	802.0	904.0	814.0	762.0	18.6	dubai	india
1	849.0	837.0	960.0	856.0	817.0	17.5	dubai	india
2	899.0	873.0	1017.0	912.0	871.0	16.8	dubai	india
3	899.0	891.0	1017.0	870.0	871.0	16.9	china	india
4	949.0	926.0	1073.0	912.0	925.0	17.7	china	india
...
912	1899.0	2156.0	1864.0	1965.0	1960.0	15.7	india	pakistan
913	1719.0	1942.0	2000.0	2245.0	2096.0	30.6	usa	china
914	2259.0	1237.0	2271.0	2526.0	2368.0	104.2	pakistan	china
915	1099.0	677.0	1130.0	1121.0	1143.0	68.8	pakistan	dubai
916	1299.0	784.0	1356.0	1261.0	1307.0	73.0	pakistan	india

917 rows × 8 columns

получаем список моделей, которые:

- Дешевле всего стоят в одной стране
- Дороже всего — в другой
- Имеют наибольший потенциал для перепродажи с прибылью

```
In [197...]: # ТОП-15 для арбитража
print("ТОП-15 моделей для арбитража:")
arbitrage = df.nlargest(15, 'price_spread')[['company_name', 'model_name', 'best_buy_country', 'best_sell_country', 'min_price', 'max_price', 'price_spread', 'price_spread_pct']]
display(arbitrage)
```

ТОП-15 моделей для арбитража:

	company_name	model_name	best_buy_country	best_sell_country	min_price	max_price	price_spread	price_spread_pct
647	Huawei	Mate XT 512GB	pakistan	india	1390.0	3107.0	1717.0	123.5
646	Huawei	Mate XT 256GB	pakistan	india	1283.0	2937.0	1654.0	128.9
616	Huawei	Mate X2	pakistan	india	1247.0	2824.0	1577.0	126.5
629	Huawei	Mate X3	pakistan	india	1247.0	2824.0	1577.0	126.5
643	Huawei	Mate X6	pakistan	india	1247.0	2824.0	1577.0	126.5
620	Huawei	Mate Xs 2	pakistan	usa	1069.0	2499.0	1430.0	133.8
914	Samsung	Galaxy Z Fold6 1TB	pakistan	china	1237.0	2526.0	1289.0	104.2
96	Apple	iPad Pro 2TB	pakistan	india	1425.0	2258.0	833.0	58.5
238	Vivo	X200 Pro 512GB	pakistan	india	891.0	1695.0	804.0	90.2
632	Huawei	Mate 60 Pro+	pakistan	india	891.0	1695.0	804.0	90.2
237	Vivo	X200 Pro 256GB	pakistan	india	819.0	1582.0	763.0	93.2
614	Huawei	P50 Pocket	pakistan	india	713.0	1469.0	756.0	106.0
95	Apple	iPad Pro 1TB	pakistan	india	1283.0	2033.0	750.0	58.5
324	Oppo	Find N3 512GB	pakistan	india	1069.0	1808.0	739.0	69.1
94	Apple	iPad Pro 512GB	pakistan	india	1105.0	1807.0	702.0	63.5

получаем модели, у которых наибольшая процентная разница между минимальной и максимальной ценой — то есть максимальный ROI при перепродаже

In [198...]

```
# ТОП-15 по ROI при перепродаже
print("ТОП-15 по ROI при перепродаже:")
arbitrage_price_spread_pct = df.nlargest(15, 'price_spread_pct')[['company_name', 'model_name', 'best_buy_country', 'best_sell_country', 'min_price', 'max_price', 'price_spread', 'price_spread_pct']]
display(arbitrage_price_spread_pct)
```

ТОП-15 по ROI при перепродаже:

	company_name	model_name	best_buy_country	best_sell_country	min_price	max_price	price_spread	price_spread_pct
283	Vivo	V20 Pro 128GB	pakistan	usa	178.0	499.0	321.0	180.3
284	Vivo	V20 Pro 256GB	pakistan	usa	196.0	549.0	353.0	180.1
419	Oppo	A49 5G 128GB	pakistan	dubai	71.0	191.0	120.0	169.0
418	Oppo	A50 5G 128GB	pakistan	dubai	78.0	204.0	126.0	161.5
290	Vivo	V15 Pro 256GB	pakistan	dubai	178.0	463.0	285.0	160.1
272	Vivo	Y30 64GB	pakistan	china	71.0	182.0	111.0	156.3
289	Vivo	V15 Pro 128GB	pakistan	dubai	160.0	408.0	248.0	155.0
417	Oppo	A51 5G 128GB	pakistan	dubai	86.0	218.0	132.0	153.5
416	Oppo	A52 5G 128GB	pakistan	dubai	93.0	231.0	138.0	148.4
267	Vivo	V15 128GB	pakistan	china	125.0	309.0	184.0	147.2
265	Vivo	Y11 32GB	pakistan	dubai	66.0	163.0	97.0	147.0
415	Oppo	A53 5G 128GB	pakistan	dubai	100.0	245.0	145.0	145.0
275	Vivo	X21 128GB	pakistan	usa	143.0	349.0	206.0	144.1
413	Oppo	A55 5G 128GB	pakistan	dubai	107.0	259.0	152.0	142.1
261	Vivo	Y12s 64GB	pakistan	dubai	68.0	163.0	95.0	139.7

In [199...]

```
# Популярные маршруты
routes = df.groupby(['best_buy_country', 'best_sell_country']).size().sort_values(ascending=False)
print("\n🌐 ТОП-10 арбитражных маршрутов:")
routes.head(10)
```

🌐 ТОП-10 арбитражных маршрутов:

Out[199...]

best_buy_country	best_sell_country	
pakistan	dubai	322
	india	170
	usa	134
	china	77
china	india	41
	dubai	38
india	dubai	34
china	pakistan	22
dubai	india	22
china	usa	12
dtype: int64		

Общие

- Основные характеристики смартфонов (RAM, батарея, экран) имеют логичные распределения, но с выбросами (планшеты, fold-модели).
- Цены сильно варьируются в зависимости от страны, но сохраняют общую структуру: Пакистан и Индия — дешевле, США и ОАЭ — дороже.
- Доступность зависит не только от цены устройства, но и от уровня доходов населения.
 - смартфоны наиболее доступны в США и ОАЭ, а наименее — в Индии и Пакистане
- Ключевые драйверы цены: бренд, объем RAM, год выпуска, основная камера.
- Бюджетные бренды предлагают лучшее соотношение RAM/\$.
- Рынок смартфонов сегментирован: есть четкое разделение на бюджетный, средний, флагманский и ультра-премиум классы.

Стратегические выводы

Для потребителей:

- Лучшее соотношение цена/качество: Poco, Realme, Infinix
- Где покупать: Pakistan (абсолютная цена), USA/UAE (доступность)
- RAM/\$ лидеры: Бюджетные бренды дают в 2-3 раза больше памяти на доллар

Для бизнеса:

-  Арибитраж: Pakistan → India/Dubai (разница до 130%)
-  Целевые сегменты: Средний класс (\$400-700) — 36% рынка
-  Региональные стратегии: Premium в USA/UAE, Mass-market в India/Pakistan

Для аналитики:

-  Главные драйверы цены: Бренд > RAM > Камера > Процессор
-  Доступность ≠ Цена: Учет зарплат критичен для понимания рынка

Выводы по шагу 3

Единая валюта: Цены конвертированы в USD для 5 стран, что позволяет корректно сравнивать уровни стоимости и доступности.

Характеристики устройств

- RAM: медиана 8 ГБ, чёткие кластеры на 4/6/8/12/16 ГБ. Подтверждает сегментацию рынка на бюджетный, средний и флагманский сегменты.
- Камеры: основной модуль чаще всего 48–50 МП, фронтальная — 8–16 МП. Экстремумы (200 МП, 60 МП) встречаются редко и характерны для маркетинговых моделей.
- Батарея: стандарт ~5000 мА·ч, выбросы до 11 200 мА·ч связаны с планшетами и fold-устройствами.
- Экран: медиана ~6.7", что соответствует современному стандарту смартфонов. Устройства >7.5" — это планшеты и складные модели.
- Вес: большинство устройств 180–210 г, но есть тяжёлые модели до 700+ г.
- Возраст устройств: Большинство моделей в выборке были выпущены в 2021-2024 годах, что говорит о репрезентативности данных для анализа текущих рыночных тенденций.
- Соотношение цена/качество: Лучшее соотношение "RAM/цена" демонстрируют бюджетные бренды (Infinix, Realme), в то время как премиальные бренды "закладывают" стоимость бренда в цену, предлагая меньше гигабайт оперативной памяти на доллар.

Цены и доступность

- Средние цены в долларах США:
 - Дубай (600) и США (590) — самые дорогие рынки.
 - Индия (577) и Китай (543) — средний уровень.
 - Пакистан (~452\$) — самый дешёвый рынок.
 - В Индии средняя цена выше, чем в Китае (577 vs 543), но медианная — ниже (406 vs 421), что указывает на более широкий ассортимент дешевых моделей в Индии при наличии экстремально дорогих "выбросов".
- Доступность (affordability):
 - США и ОАЭ — смартфоны наиболее доступны (0.1–0.13 месячной зарплаты).
 - Китай — средний уровень (~0.25).
 - Индия и Пакистан — наименее доступны (1–1.7 месячной зарплаты).
 - Ключевой инсайт: Парадокс цены и доступности. Несмотря на самые низкие абсолютные цены в Пакистане, смартфоны там наименее доступны из-за низкого уровня доходов. Житель Индии тратит на смартфон в 17 раз большую долю своего дохода, чем житель США.

Ценовые сегменты:

- Премиум (больше 800): Sony, Huawei, Apple.
 - Стабильно высокие цены во всех странах
 - Низкая доступность в развивающихся рынках
- Верхний средний (600–800): Google, Samsung.
- Средний (400–600): OnePlus, Honor, Xiaomi, Oppo.
 - Оптимальный баланс цена/характеристики
- Бюджетный (меньше 400): Infinix, Nokia, Poco, Realme, Lenovo.
 - Лидеры доступности во всех странах

Бренды

- ТОП по цене: Sony, Huawei, Apple, Google, Samsung — премиум-сегмент.
- ТОП по доступности: Infinix, Nokia, Realme, Poco, Lenovo — оптимальное соотношение «характеристики/цена».
- Вывод: премиум-бренды продают имидж и экосистему, а не только характеристики; китайские бренды конкурируют за массовый рынок.

Корреляция

- RAM ✕ Цена: положительная связь подтверждает гипотезу, но не единственный фактор.
- RAM/цена 📈 Цена: чем дороже устройство, тем меньше «памяти за доллар» — премиум продаёт не характеристики, а бренд и экосистему.
- Экран и батарея: умеренная положительная связь (больший экран → больше батарея).
- Цены по странам: сильно коррелируют, но с разными коэффициентами наценки.

Арбитражные возможности (перепродажа)

- Выявлены модели с максимальной разницей в цене между странами (до 123%).
- Наиболее выгодные маршруты для перепродажи: Покупка в Пакистане с последующей продажей в Индии, ОАЭ или США.
- Топ моделей для арбитража: Флагманские складные смартфоны Huawei Mate X серии и Samsung Galaxy Z Fold.

Основные инсайты

- Цена определяется комплексно: Бренд, RAM и год выпуска — ключевые драйверы. Технические характеристики (батарея, камеры) имеют нелинейное и не всегда прямое влияние: премиум-бренды дороже при схожих характеристиках.
- Доступность сильно различается по странам: США и ОАЭ — самые выгодные рынки для покупки, Индия и Пакистан — самые дорогие относительно доходов.
 - Доступность — относительный показатель: Нельзя оценивать выгоду только по абсолютной цене. Уровень доходов населения кардинально меняет картину.
- Рынок растёт: пик моделей в 2023–2024 гг., особенно в среднем сегменте.
- Стратегии брендов различаются: Apple и Samsung удерживают премиум, китайские бренды активно конкурируют в среднем и бюджетном сегменте.
 - Флагманы имеют меньшую батарею, чем средний сегмент — оптимизация под производительность чипов.
- Для перепродажи выгоднее покупать в Индии/Пакистане/Китае и продавать в США/ОАЭ, но нужно учитывать налоги и логистику.
 - Наибольшие разницы цен между странами позволяют оценить потенциал перепродажи
 - Выгода для покупателя и перепродавца — разные вещи: Потребителю выгоднее покупать в США/ОАЭ из-за высокой доступности. Перепродавцу же интересны страны с низкими абсолютными ценами (Пакистан, Индия) для закупки с целью арбитража.
- Рынок четко сегментирован: Существуют ярко выраженные ценовые ниши с характерным для каждой набором характеристик.

* к содержанию

Шаг 4: Составление портрета пользователя каждого региона

ТОП-бренды по странам (по количеству моделей и средней цене)

```
In [200...]: # Список стран и соответствующих колонок
price_cols = {
    'Pakistan': 'price_pakistan_usd',
    'India': 'price_india_usd',
    'China': 'price_china_usd',
    'Dubai': 'price_dubai_usd',
    'USA': 'price_usa_usd'
}

top_n = 5
price_matrix = {}

# ТОП брендов по средней цене в каждой стране (по убыванию)
top_by_avg_price = {}
for country, col in price_cols.items():
    tmp = df.groupby('company_name')[col].mean().dropna().sort_values(ascending=False).head(top_n)
    top_by_avg_price[country] = tmp
    print(f"\nТОП-{top_n} брендов по средней цене в {country} (USD):\n", tmp)

# Собираем ТОП-бренды по средней цене и визуализируем
for country, col in price_cols.items():
    top = (
        df.groupby('company_name')[col]
        .mean()
        .dropna()
        .sort_values(ascending=False)
        .head(top_n)
        .round(2)
    )
    price_matrix[country] = top

# Визуализация
plt.figure(figsize=(10, 5))
sns.barplot(x=top.values, y=top.index, palette="viridis")
plt.title(f"ТОП-{top_n} брендов по средней цене — {country}")
plt.xlabel("Средняя цена (USD)")
plt.tight_layout()
plt.show()

rows = []
```

```

# Собираем расширенную статистику по странам и брендам
for country, col in price_cols.items():
    # ТОП-5 брендов по средней цене
    top_brands = (
        df.groupby('company_name')[col]
        .mean()
        .dropna()
        .sort_values(ascending=False)
        .head(top_n)
        .index
    )

    for brand in top_brands:
        sub = df[df['company_name'] == brand]
        prices = sub[col].dropna()
        rows.append({
            'Страна': country,
            'Бренд': brand,
            'Средняя цена': prices.mean().round(2),
            'Медианная цена': prices.median().round(2),
            'Кол-во моделей': len(prices)
        })
)

# Формируем таблицу
brand_country_stats = pd.DataFrame(rows)
pivot_table = brand_country_stats.pivot(index='Страна', columns='Бренд')
pivot_table.columns.names = [None, None] # убираем названия уровней
pivot_table = pivot_table.round(2)

# Вывод
print("📊 Расширенная таблица: ТОП-5 брендов по средней и медианной цене + количество моделей")
display(pivot_table)

```

ТОП-5 брендов по средней цене в Pakistan (USD):

```

company_name
Sony      1167.777778
Apple     882.257732
Samsung   753.590909
Huawei    657.477273
Google    614.190476
Name: price_pakistan_usd, dtype: float64

```

ТОП-5 брендов по средней цене в India (USD):

```

company_name
Huawei   1165.045455
Apple    1163.742268
Sony     1035.666667
Google   794.666667
Samsung  718.636364
Name: price_india_usd, dtype: float64

```

ТОП-5 брендов по средней цене в China (USD):

```

company_name
Apple    1007.927835
Huawei  973.454545
Google   850.571429
Sony     837.222222
Samsung  726.727273
Name: price_china_usd, dtype: float64

```

ТОП-5 брендов по средней цене в Dubai (USD):

```

company_name
Huawei   1139.545455
Sony     1097.888889
Apple    1002.927835
Google   823.047619
Samsung  783.125000
Name: price_dubai_usd, dtype: float64

```

ТОП-5 брендов по средней цене в USA (USD):

```

company_name
Sony     1132.333333
Huawei  1120.318182
Apple    1028.484536
Google   755.190476
Samsung  748.431818
Name: price_usa_usd, dtype: float64

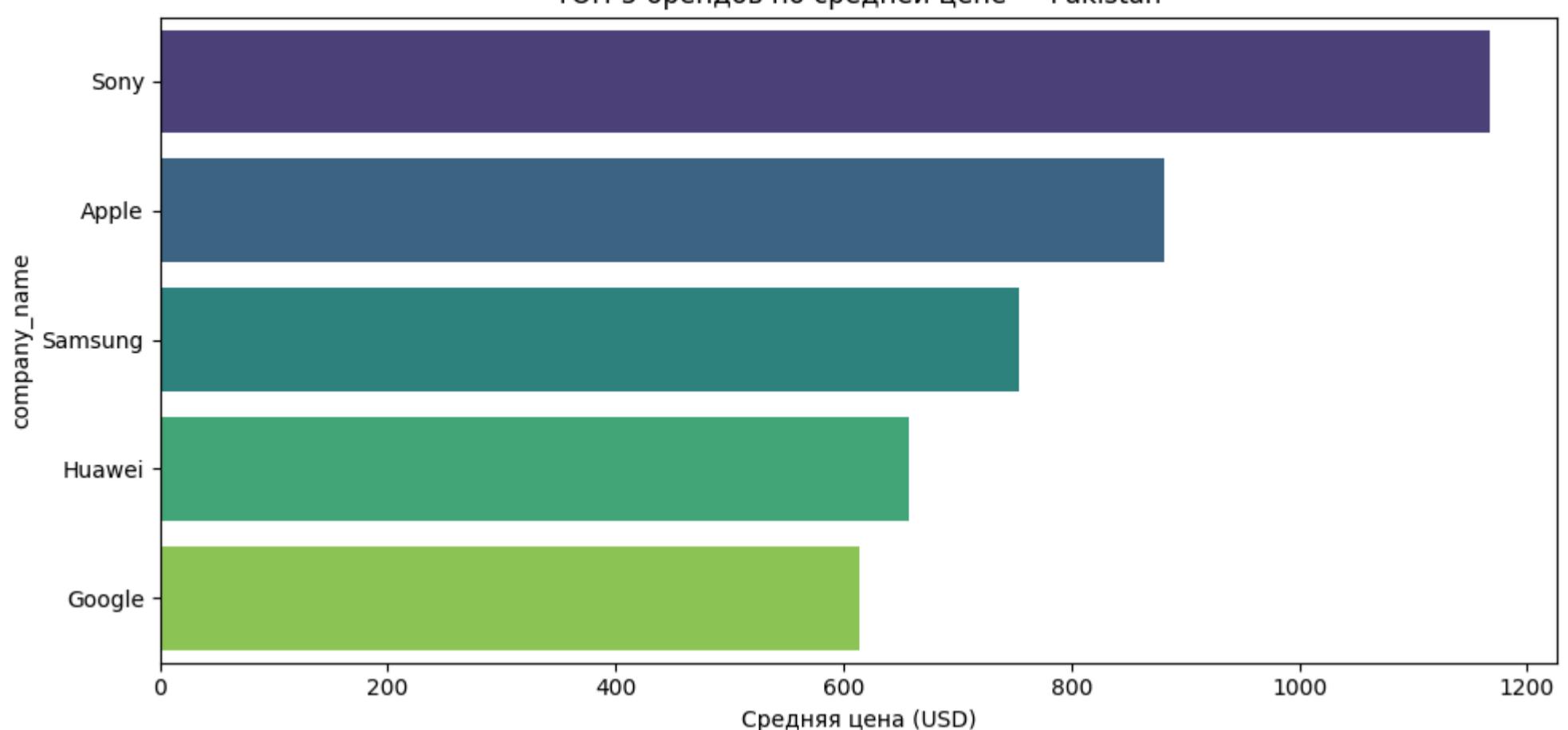
```

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2438922302.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=top.values, y=top.index, palette="viridis")
```

ТОП-5 брендов по средней цене — Pakistan

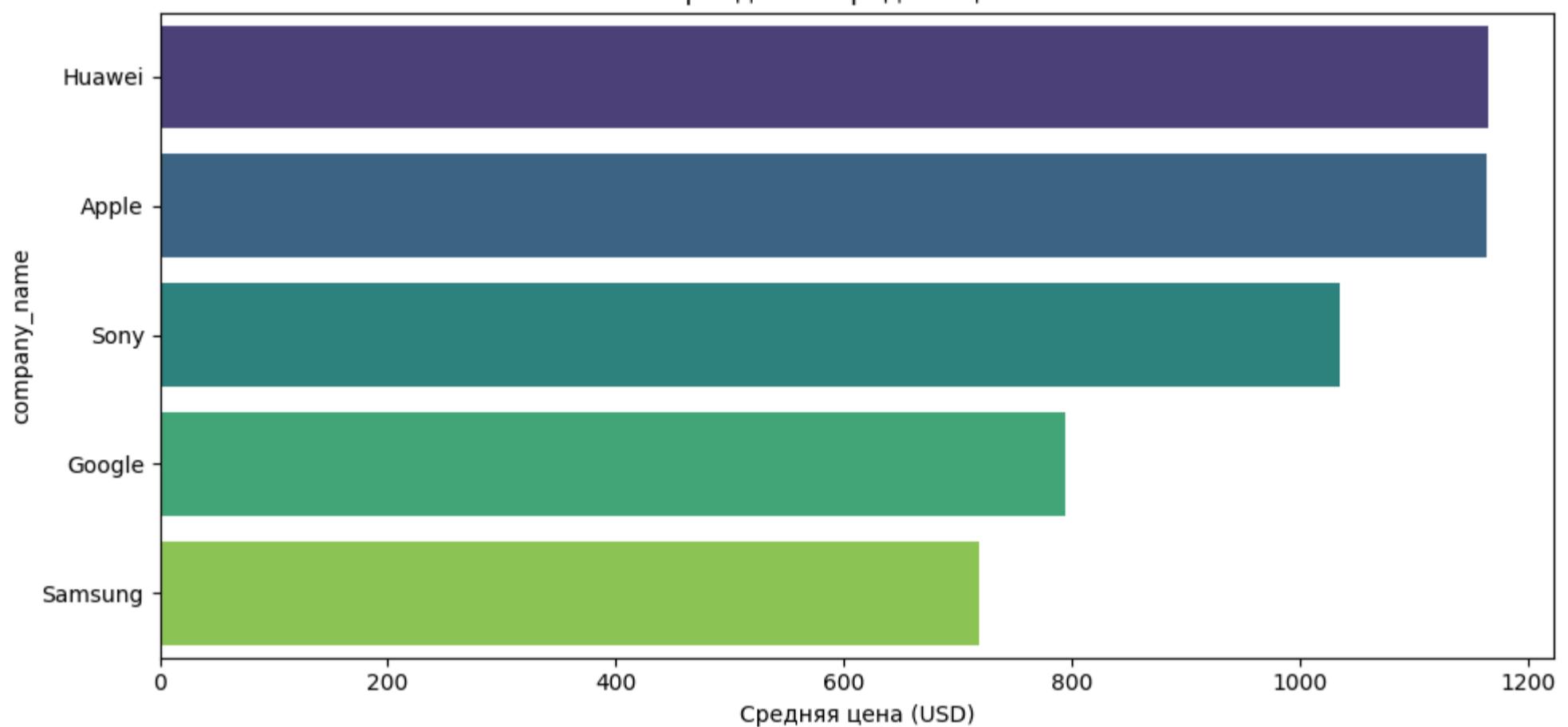


```
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2438922302.py:34: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.barplot(x=top.values, y=top.index, palette="viridis")
```

ТОП-5 брендов по средней цене — India

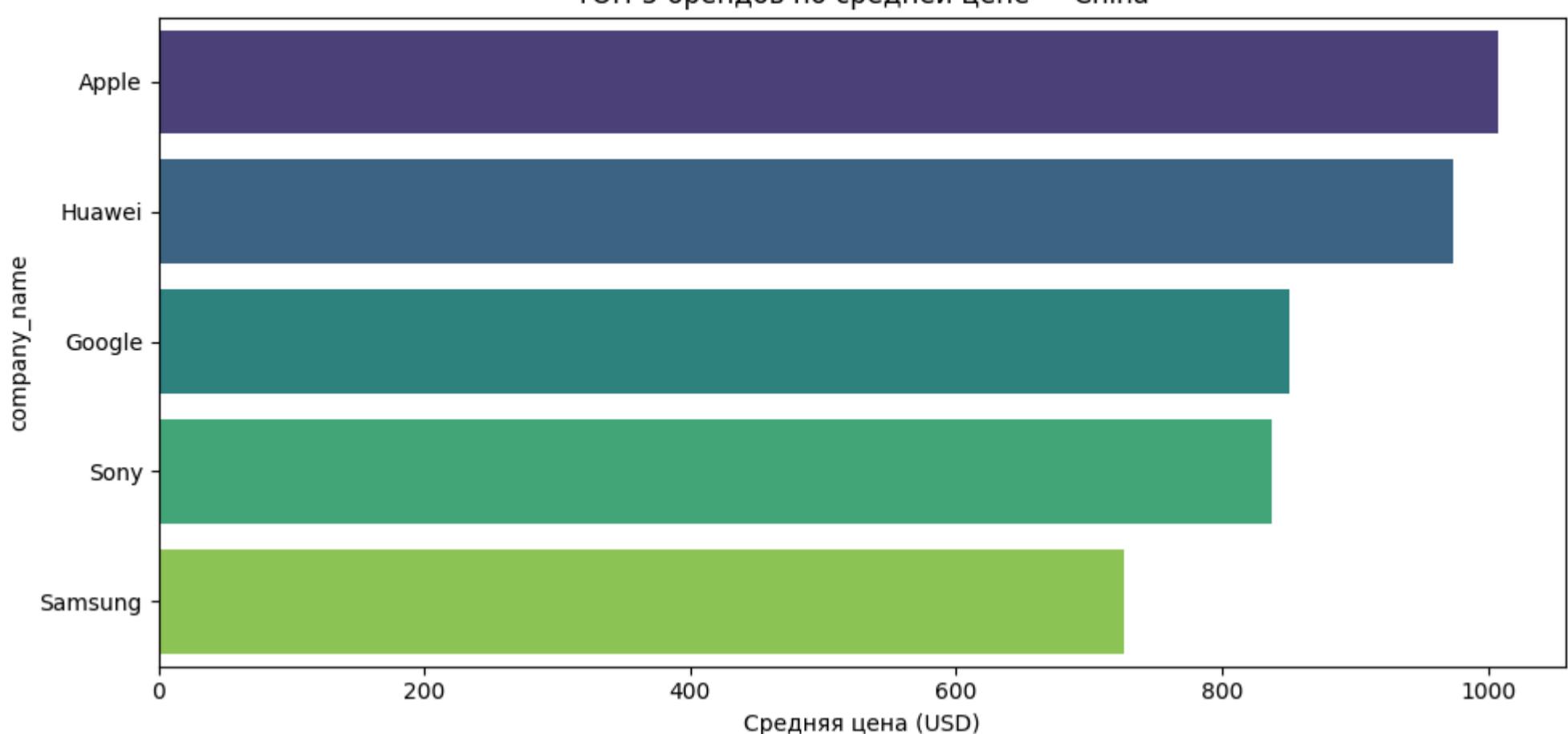


```
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2438922302.py:34: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.barplot(x=top.values, y=top.index, palette="viridis")
```

ТОП-5 брендов по средней цене — China

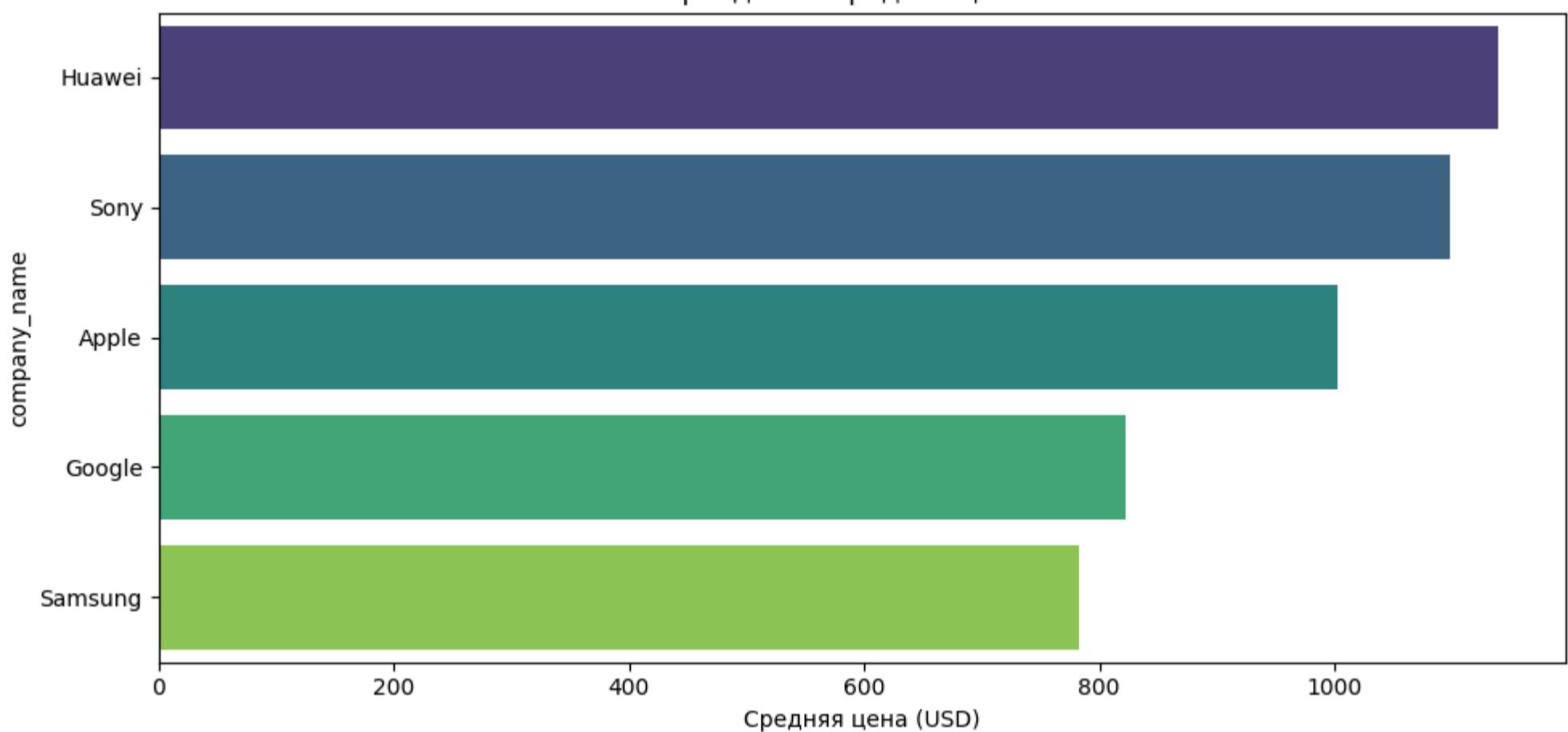


```
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2438922302.py:34: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.barplot(x=top.values, y=top.index, palette="viridis")
```

ТОП-5 брендов по средней цене — Dubai

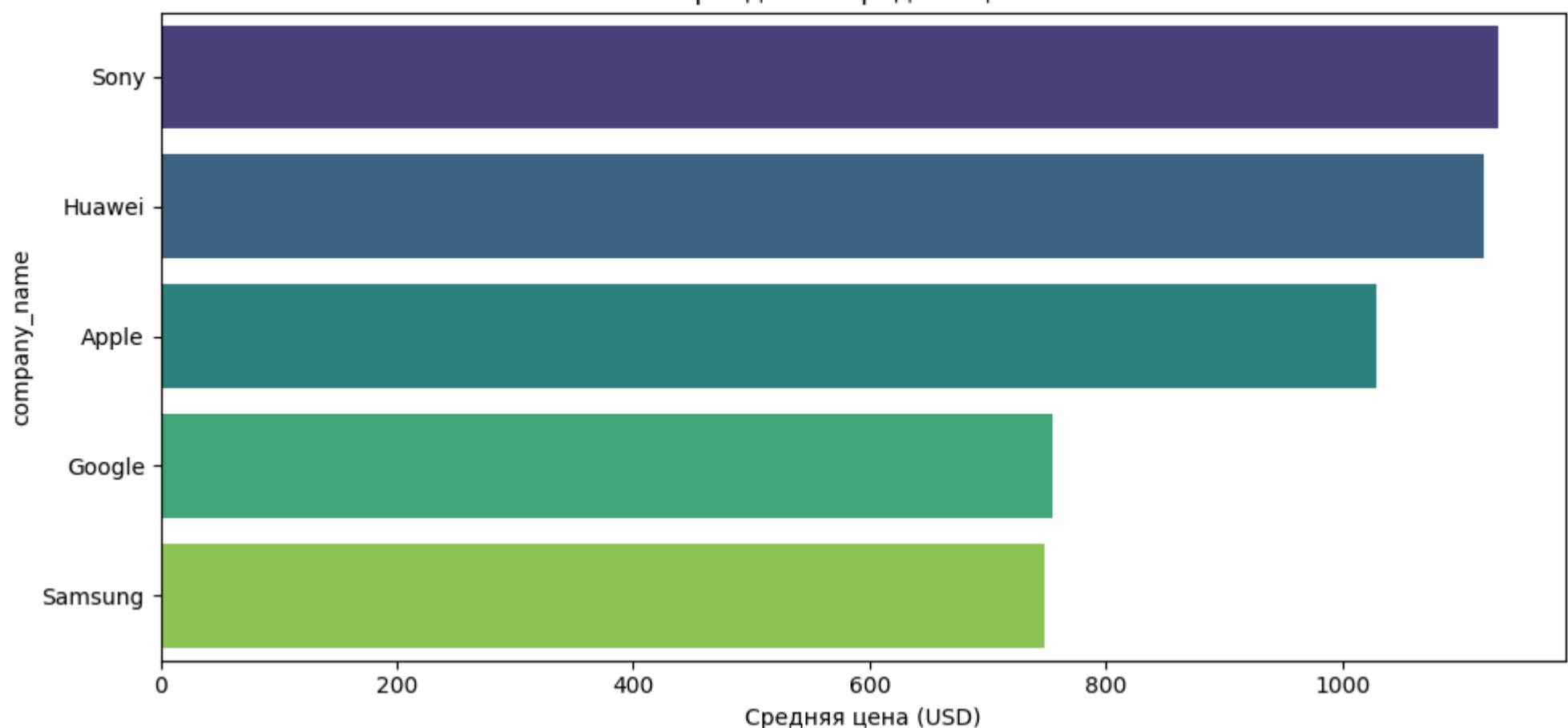


```
C:\Users\HP\AppData\Local\Temp\ipykernel_35604\2438922302.py:34: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.barplot(x=top.values, y=top.index, palette="viridis")
```

ТОП-5 брендов по средней цене — USA



Расширенная таблица: ТОП-5 брендов по средней и медианной цене + количество моделей

Страна	Средняя цена				Медианная цена				Кол-во моделей						
	Apple	Google	Huawei	Samsung	Sony	Apple	Google	Huawei	Samsung	Sony	Apple	Google	Huawei	Samsung	
China	1007.93	850.57	973.45	726.73	837.22	1024.0	772.0	898.0	772.0	884.0	97	21	44	88	9
Dubai	1002.93	823.05	1139.55	783.12	1097.89	1007.0	762.0	1007.0	817.0	1116.0	97	21	44	88	9
India	1163.74	794.67	1165.05	718.64	1035.67	1130.0	678.0	960.0	734.0	1073.0	97	21	44	88	9
Pakistan	882.26	614.19	657.48	753.59	1167.78	873.0	534.0	623.5	534.0	1211.0	97	21	44	88	9
USA	1028.48	755.19	1120.32	748.43	1132.33	999.0	699.0	999.0	699.0	1099.0	97	21	44	88	9

Разница по странам:

- Премиум-сегмент во всех странах: Sony, Apple, Huawei
- Средний сегмент: Google, Samsung
- В Пакистане Sony значительно дороже других брендов (1165 vs 880 Apple)
- В Индии Huawei и Apple почти одинаковы по цене (1162 vs 1160)
- В Китае Apple доминирует (1009), Huawei на втором месте (974)
- Huawei дороже всего в Индии и Дубае.
- Apple стабильно дорогая, особенно в Индии.
- Samsung и Google — умеренные цены, но с высокой представленностью.
- Sony — самый дорогой в USA и Pakistan
- Apple — в ТОП-3 во всех странах

Sony позиционируется как ультропремиум (выше Apple), но представлен малым количеством моделей (9 шт)

Бренды с широкой линейкой (Oppo, Samsung) охватывают все сегменты, а премиум-бренды (Sony, Apple, Huawei) держат высокую цену независимо от региона.

В разных странах лидеры по средней цене совпадают частично, но порядок меняется. Это означает, что у брендов разные ценовые профили в разных регионах (или локальные цены/налоги/версионирование влияют)

Средние/медианные значения характеристик по странам

In [201...]

```
# Список стран и метрик
metrics = ["price_usa_usd", "price_india_usd", "price_china_usd", "price_pakistan_usd", "price_dubai_usd"]
features = ["ram_gb", "back_camera_1", "screen_size_inches", "battery_capacity_mah"]

# Средние значения
mean_stats = df.groupby("launched_year")[features].mean().round(3)
print("Средние характеристики по годам:")
display(mean_stats)
```

Средние характеристики по годам:

	ram_gb	back_camera_1	screen_size_inches	battery_capacity_mah
launched_year				
2014	1.500	13.000	5.250	2150.000
2016	3.400	11.000	6.960	3920.000
2017	4.444	14.889	5.688	3220.778
2018	4.381	13.714	6.161	3177.952
2019	6.283	28.539	6.392	3950.717
2020	6.439	35.149	7.396	4967.276
2021	6.981	45.701	6.764	4617.252
2022	7.589	43.590	7.064	4992.466
2023	8.136	51.522	7.272	5297.391
2024	9.096	55.519	7.258	5418.127
2025	9.000	55.667	6.697	5716.667

In [202...]

```
def country_stats(df):
    stats = []
    for country, col in price_cols.items():
        sub = df[df[col].notna()].copy()
        sub['camera_clean'] = sub['back_camera_1'].replace(-1, np.nan)

        stats.append({
            'country': country,
            'price_mean': sub[col].mean(),
            'price_median': sub[col].median(),
            'ram_mean': sub['ram_gb'].mean(),
            'ram_median': sub['ram_gb'].median(),
            'camera_mean': sub['camera_clean'].mean(),
            'camera_median': sub['camera_clean'].median(),
            'screen_mean': sub['screen_size_inches'].mean(),
            'screen_median': sub['screen_size_inches'].median(),
            'battery_mean': sub['battery_capacity_mah'].mean(),
            'battery_median': sub['battery_capacity_mah'].median(),
            'n_models': len(sub)
        })
    return pd.DataFrame(stats).set_index('country').round(2)

# Выход
country_summary = country_stats(df)
display(country_summary)
```

country	price_mean	price_median	ram_mean	ram_median	camera_mean	camera_median	screen_mean	screen_median	battery_mean	battery_median
Pakistan	452.42	321.0	7.8	8.0	46.49	50.0	7.09	6.67	5030.12	5000.0
India	578.43	407.0	7.8	8.0	46.49	50.0	7.09	6.67	5030.12	5000.0
China	542.28	421.0	7.8	8.0	46.49	50.0	7.09	6.67	5030.12	5000.0
Dubai	600.27	463.0	7.8	8.0	46.49	50.0	7.09	6.67	5030.12	5000.0
USA	590.16	449.0	7.8	8.0	46.49	50.0	7.09	6.67	5030.12	5000.0

Цена: от 451 (Пакистан) до 600 (Дубай).

RAM: везде ~7.8 ГБ, медиана — 8 ГБ.

Камера: средняя — ~46.5 МП, медиана — 50 МП.

Экран: ~7.09", медиана — 6.67".

Батарея: ~5030 мА·ч, медиана — 5000 мА·ч.

технические характеристики смартфонов по странам почти идентичны — различия в цене обусловлены не «железом», а брендом, налогами и рыночной стратегией.

Какие характеристики преобладают в каждом регионе

In [203...]

```
# Выбираем страну и соответствующий столбец цены
country = 'Pakistan'
price_col = price_cols[country]

# Категориальные признаки
cat_cols = ['price_category', 'ram_category', 'battery_category', 'weight_category', 'screen_category']

# Фильтруем модели, доступные в стране
df_country = df[df[price_col].notna()]
print(f"\n📍 {country} – доступно моделей: {len(df_country)}")
```

```

# Вывод распределений
for col in cat_cols:
    dist = df_country[col].value_counts(normalize=True).mul(100).round(1)
    print(f"\n{col} (%):")
    print(dist)

# Визуализация по price_category
plt.figure(figsize=(6, 3))
sns.countplot(data=df_country, x='price_category', order=df['price_category'].cat.categories)
plt.title('Распределение price_category — {country}')
plt.tight_layout()
plt.show()

```

● Pakistan – доступно моделей: 917

```

price_category (%):
Средний      35.7
Бюджетный    33.2
Флагманский   24.1
Ультрапремиум 7.1
Name: price_category, dtype: float64

```

```

ram_category (%):
Средний      55.1
Высокий       24.5
Низкий        20.4
Name: ram_category, dtype: float64

```

```

battery_category (%):
средняя       72.3
слабая         10.7
экстремальная  8.9
большая        8.1
Name: battery_category, dtype: float64

```

```

weight_category (%):
средний       71.5
лёгкий         16.4
планшетный    10.3
тяжёлый        1.9
Name: weight_category, dtype: float64

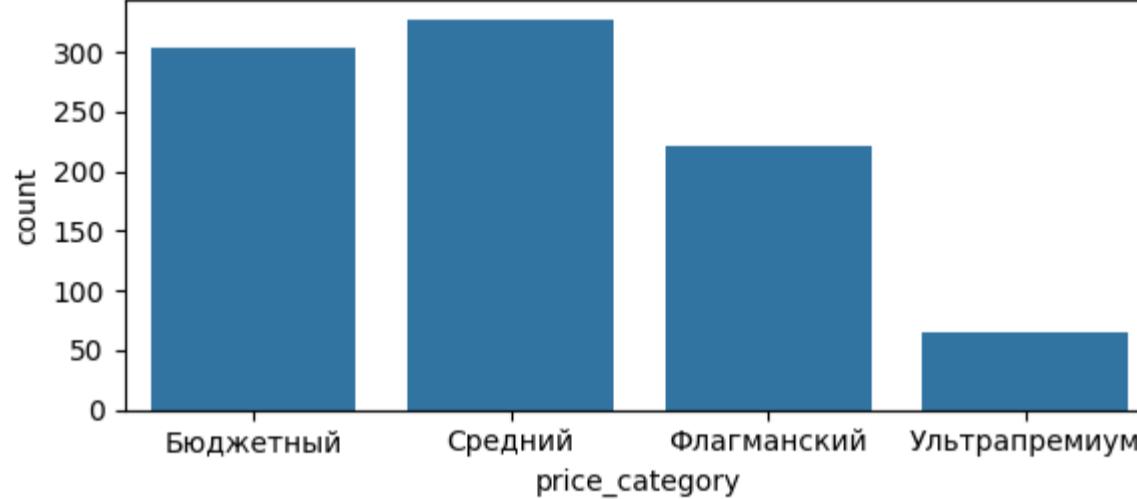
```

```

screen_category (%):
стандартный   82.2
планшетный    10.0
большой        3.9
компактный     3.8
Name: screen_category, dtype: float64

```

Распределение price_category — Pakistan



In [204]...

```

# Анализ по ценовым сегментам
print("=" * 80)
print("ХАРАКТЕРИСТИКИ ПО ЦЕНОВЫМ СЕГМЕНТАМ")
print("=" * 80)

# Средние характеристики по ценовым категориям
segment_stats = df.groupby('price_category').agg({
    'ram_gb': 'mean',
    'battery_capacity_mah': 'mean',
    'screen_size_inches': 'mean',
    'back_camera_1': 'mean',
    'price_usa_usd': ['mean', 'median', 'count']
}).round(1)

segment_stats.columns = ['RAM (ГБ)', 'Батарея (мА·ч)', 'Экран ("')', 'Камера (МП)',
                        'Средняя цена', 'Медианная цена', 'Количество']

# Сортируем по возрастанию цены
category_order = ['Бюджетный', 'Средний', 'Флагманский', 'Ультрапремиум']
segment_stats = segment_stats.reindex(category_order)

print("\n📊 СРЕДНИЕ ХАРАКТЕРИСТИКИ ПО СЕГМЕНТАМ:")
display(segment_stats)

# Доля сегментов
segment_shares = df['price_category'].value_counts(normalize=True).sort_index() * 100

```

```

segment_shares = segment_shares.reindex(category_order)

print("\n📈 ДОЛЯ РЫНКА ПО СЕГМЕНТАМ:")
for segment, share in segment_shares.items():
    print(f"    {segment}: {share:.1f}%")

# Визуализация
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# График 1: RAM по сегментам
axes[0, 0].bar(segment_stats.index, segment_stats['RAM (ГБ)'], color='steelblue', alpha=0.8)
axes[0, 0].set_ylabel('RAM (ГБ)', fontsize=12)
axes[0, 0].set_title('Средний объем RAM по сегментам', fontsize=13, fontweight='bold')
axes[0, 0].tick_params(axis='x', rotation=45)
axes[0, 0].grid(axis='y', alpha=0.3)

for i, val in enumerate(segment_stats['RAM (ГБ)']):
    axes[0, 0].text(i, val + 0.2, f'{val:.1f} ГБ', ha='center', fontsize=10, fontweight='bold')

# График 2: Батарея по сегментам
axes[0, 1].bar(segment_stats.index, segment_stats['Батарея (мА·ч)'], color='green', alpha=0.7)
axes[0, 1].set_ylabel('Батарея (мА·ч)', fontsize=12)
axes[0, 1].set_title('Средняя емкость батареи по сегментам', fontsize=13, fontweight='bold')
axes[0, 1].tick_params(axis='x', rotation=45)
axes[0, 1].grid(axis='y', alpha=0.3)
axes[0, 1].axhline(y=5000, color='red', linestyle='--', linewidth=1, label='Стандарт (5000 мА·ч)')
axes[0, 1].legend()

for i, val in enumerate(segment_stats['Батарея (мА·ч)']):
    axes[0, 1].text(i, val + 50, f'{val:.0f}', ha='center', fontsize=10, fontweight='bold')

# График 3: Цены по сегментам
x = np.arange(len(segment_stats))
width = 0.35

axes[1, 0].bar(x - width/2, segment_stats['Средняя цена'], width, label='Средняя', color='teal', alpha=0.8)
axes[1, 0].bar(x + width/2, segment_stats['Медианная цена'], width, label='Медианная', color='orange', alpha=0.8)
axes[1, 0].set_ylabel('Цена (USD)', fontsize=12)
axes[1, 0].set_title('Средняя и медианная цена по сегментам', fontsize=13, fontweight='bold')
axes[1, 0].set_xticks(x)
axes[1, 0].set_xticklabels(segment_stats.index, rotation=45)
axes[1, 0].legend()
axes[1, 0].grid(axis='y', alpha=0.3)

# График 4: Доля сегментов (Pie chart)
colors_pie = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99']
axes[1, 1].pie(segment_shares, labels=segment_shares.index, autopct='%1.1f%%',
               startangle=90, colors=colors_pie, textprops={'fontsize': 11, 'fontweight': 'bold'})
axes[1, 1].set_title('Распределение моделей по ценовым сегментам', fontsize=13, fontweight='bold')

plt.tight_layout()
plt.show()

```

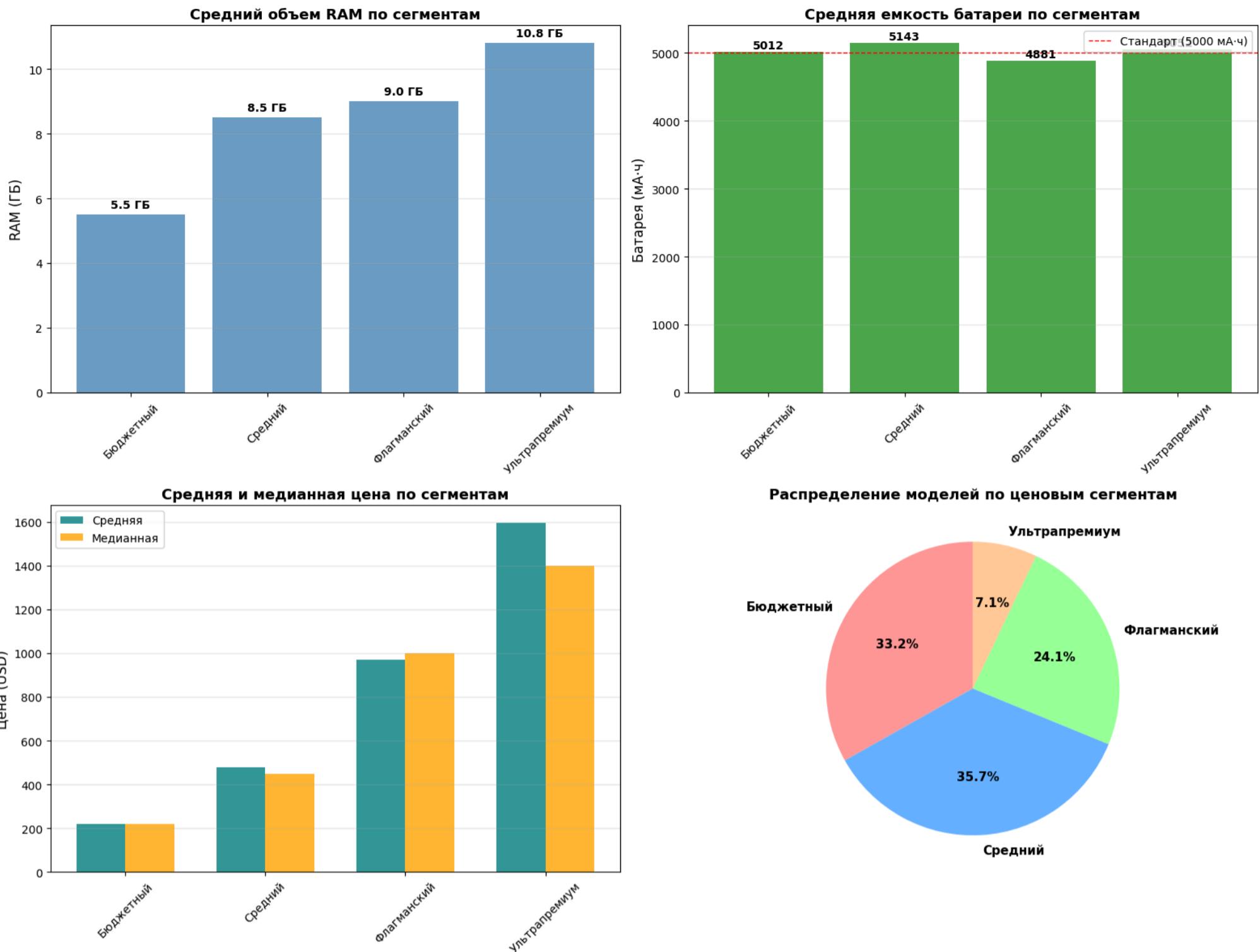
=====
ХАРАКТЕРИСТИКИ ПО ЦЕНОВЫМ СЕГМЕНТАМ
=====

📊 СРЕДНИЕ ХАРАКТЕРИСТИКИ ПО СЕГМЕНТАМ:

	RAM (ГБ)	Батарея (мА·ч)	Экран (")	Камера (МП)	Средняя цена	Медианная цена	Количество
--	----------	----------------	-----------	-------------	--------------	----------------	------------

price_category							
Бюджетный	5.5	5011.9	6.9	38.7	217.8	219.0	304
Средний	8.5	5143.4	7.1	54.3	478.9	449.0	327
Флагманский	9.0	4881.1	7.1	42.2	970.9	999.0	221
Ультрапремиум	10.8	5052.5	7.4	58.4	1596.4	1399.0	65

📈 ДОЛЯ РЫНКА ПО СЕГМЕНТАМ:
 Бюджетный: 33.2%
 Средний: 35.7%
 Флагманский: 24.1%
 Ультрапремиум: 7.1%



Ценовые категории:

- Средний сегмент — ~35%
- Бюджетный — ~33%
- Флагманский — ~24%
- Ультрапремиум — ~7%

RAM:

- Средний — ~55%
- Высокий — ~25%
- Низкий — ~20%

Батарея:

- Средняя — ~72%
- Остальные категории равномерно распределены

Вес:

- Средний — ~71%
- Лёгкий — ~16%
- Планшетный — ~10%

Экран:

- Стандартный — ~82%
- Планшетный — ~10%
- Компактный и большой — ~4%

рынок ориентирован на средний сегмент — устройства с 6–8 ГБ RAM, экраном ~6.7", батареей ~5000 мА·ч. Ультрапремиум — ниша, а планшетные форматы — редкость.

Флагманы имеют МЕНЬШУЮ батарею (~262 мА·ч), чем средний сегмент! Причина — энергоэффективные премиум-процессоры (Snapdragon 8 Gen, Apple A-series).

«Паспорт устройства» для каждой страны

In [205...]

```
def device_passport(country_col, country_name, affordability_col):
    available = df[df[country_col].notna()]
```

```

stats = {
    "Доступность": (available[affordability_col].mean(), available[affordability_col].median()),
    "Цена": (available[country_col].mean(), available[country_col].median()),
    "RAM": (available["ram_gb"].mean(), available["ram_gb"].median()),
    "Экран": (available["screen_size_inches"].mean(), available["screen_size_inches"].median()),
    "Камера": (available["back_camera_1"].mean(), available["back_camera_1"].median()),
    "Батарея": (available["battery_capacity_mah"].mean(), available["battery_capacity_mah"].median()),
}

print(f"\n📋 Паспорт устройства для {country_name}:")
for k, (mean_val, median_val) in stats.items():
    print(f"{k}: средняя = {round(mean_val,2)}, медианная = {round(median_val,2)}")

# Вызовы для всех стран
countries = [
    ("price_usa_usd", "США", "affordability_usa"),
    ("price_india_usd", "Индия", "affordability_india"),
    ("price_china_usd", "Китай", "affordability_china"),
    ("price_pakistan_usd", "Пакистан", "affordability_pakistan"),
    ("price_dubai_usd", "ОАЭ", "affordability_uae")
]

for col_price, name, col_afford in countries:
    device_passport(col_price, name, col_afford)

```

📋 Паспорт устройства для США:
Доступность: средняя = 0.09, медианная = 0.07
Цена: средняя = 590.16, медианная = 449.0
RAM: средняя = 7.8, медианная = 8.0
Экран: средняя = 7.09, медианная = 6.67
Камера: средняя = 46.49, медианная = 50.0
Батарея: средняя = 5030.12, медианная = 5000.0

📋 Паспорт устройства для Индия:
Доступность: средняя = 1.73, медианная = 1.22
Цена: средняя = 578.43, медианная = 407.0
RAM: средняя = 7.8, медианная = 8.0
Экран: средняя = 7.09, медианная = 6.67
Камера: средняя = 46.49, медианная = 50.0
Батарея: средняя = 5030.12, медианная = 5000.0

📋 Паспорт устройства для Китай:
Доступность: средняя = 0.25, медианная = 0.19
Цена: средняя = 542.28, медианная = 421.0
RAM: средняя = 7.8, медианная = 8.0
Экран: средняя = 7.09, медианная = 6.67
Камера: средняя = 46.49, медианная = 50.0
Батарея: средняя = 5030.12, медианная = 5000.0

📋 Паспорт устройства для Пакистан:
Доступность: средняя = 1.47, медианная = 1.04
Цена: средняя = 452.42, медианная = 321.0
RAM: средняя = 7.8, медианная = 8.0
Экран: средняя = 7.09, медианная = 6.67
Камера: средняя = 46.49, медианная = 50.0
Батарея: средняя = 5030.12, медианная = 5000.0

📋 Паспорт устройства для ОАЭ:
Доступность: средняя = 0.13, медианная = 0.1
Цена: средняя = 600.27, медианная = 463.0
RAM: средняя = 7.8, медианная = 8.0
Экран: средняя = 7.09, медианная = 6.67
Камера: средняя = 46.49, медианная = 50.0
Батарея: средняя = 5030.12, медианная = 5000.0

Портрет устройства по странам почти идентичен:

- RAM: 8 ГБ
- Камера: 50 МП
- Экран: 6.67"
- Батарея: 5000 мА·ч

Цена: варьируется от 450 до 600

Доступность сильно варьируется:

- США: 0.09 (очень доступно)
- ОАЭ: 0.13 (доступно)
- Китай: 0.25 (умеренно)
- Пакистан: 1.47 (недоступно)
- Индия: 1.72 (критически недоступно)

технически устройства одинаковы, но цена зависит от страны. Это даёт основу для анализа перепродажи и доступности.

Сравнение брендов-лидеров по регионам

In [206...]

```
# Сопоставление стран и колонок
price_cols = {
```

```

    "USA": "price_usa_usd",
    "India": "price_india_usd",
    "China": "price_china_usd",
    "Pakistan": "price_pakistan_usd",
    "Dubai": "price_dubai_usd"
}

# 🏆 ТОП-5 брендов по доле рынка
market_share = df['company_name'].value_counts(normalize=True).mul(100).round(1).head(5)
print("\n🏆 Лидеры по доле рынка:")
for brand, share in market_share.items():
    print(f" {brand}: {share}%")


# 💰 Средняя цена ТОП-брендов по странам
price_matrix = {
    country: (
        df.groupby("company_name")[col]
        .mean()
        .dropna()
        .sort_values(ascending=False)
        .head(5)
        .round(2)
    )
    for country, col in price_cols.items()
}

# Преобразуем в таблицу
heatmap_data = pd.DataFrame(price_matrix).T

# 📈 Визуализация
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlOrBr")
plt.title("Средняя цена ТОП-брендов по странам")
plt.xlabel("Бренд")
plt.ylabel("Страна")
plt.tight_layout()
plt.show()

```

🏆 Лидеры по доле рынка:

Oppo: 12.5%
Apple: 10.6%
Honor: 9.9%
Samsung: 9.6%
Vivo: 9.4%



1. Oppo: 12.5% (115 моделей) — массовый китайский производитель
2. Apple: 10.6% (97 моделей) — экосистемный подход
3. Honor: 9.9% (91 моделей) — бывший sub-brand Huawei
4. Samsung: 9.6% (88 моделей) — универсальный гигант
5. Vivo: 9.4% (86 моделей) — конкурент Oppo

Стратегии брендов:

- Sony: Малое количество моделей (9), но максимальная цена — нишевый премиум
- Apple: стабильно высокие цены

- Samsung: Широкая линейка от A-series до Z-Fold — охват всех сегментов
- Китайские бренды (Oppo, Vivo, Honor): Массовость + агрессивное ценообразование

Массовые бренды (Oppo, Vivo) доминируют по числу моделей; премиальные бренды (Sony, Apple) — по средней цене.

Лидеры по доле рынка: Oppo (12.5%) > Apple (10.6%) > Honor (9.9%) > Samsung (9.6%) > Vivo (9.4%)

Тепловая карта показывает четкую сегментацию брендов по цене

- Sony, Huawei, Apple — самые дорогие бренды во всех странах
- Samsung и Google — умеренные цены, но высокая представленность.

Премиум-бренды держат цену независимо от страны, а бренды с широкой линейкой адаптируются под рынок.

ТОП-10 телефонов с максимальной разницей цен между странами

In [207...]

```
# Общие настройки
cols = ['company_name', 'model_name', 'best_buy_country', 'best_sell_country',
        'min_price', 'max_price', 'price_spread', 'price_spread_pct']

# Вывод топа по метрике
def show_top(df, metric, top_n=10):
    top = df.nlargest(top_n, metric)[cols].round(2)
    display(top)
    return top

# Визуализация топ-N
def plot_arbitrage_bar(df, metric='price_spread', label='price_spread_pct', top_n=8):
    top = df.head(top_n)
    plt.figure(figsize=(12, 8))
    sns.barplot(x=top[metric], y=top['company_name'] + ' ' + top['model_name'].str[:20] + '...', palette='viridis')
    plt.xlabel('Разница в цене ($)')
    plt.title(f'Топ-{top_n} арбитражных возможностей (по {metric})')
    for i, (val, pct) in enumerate(zip(top[metric], top[label])):
        plt.text(val + 20, i, f'+${val:.0f}\n({pct:.0f}%)', va='center', fontweight='bold')
    plt.tight_layout()
    plt.show()

# Вывод и график
top_abs = show_top(df, 'price_spread')
top_pct = show_top(df, 'price_spread_pct')
plot_arbitrage_bar(top_abs)
```

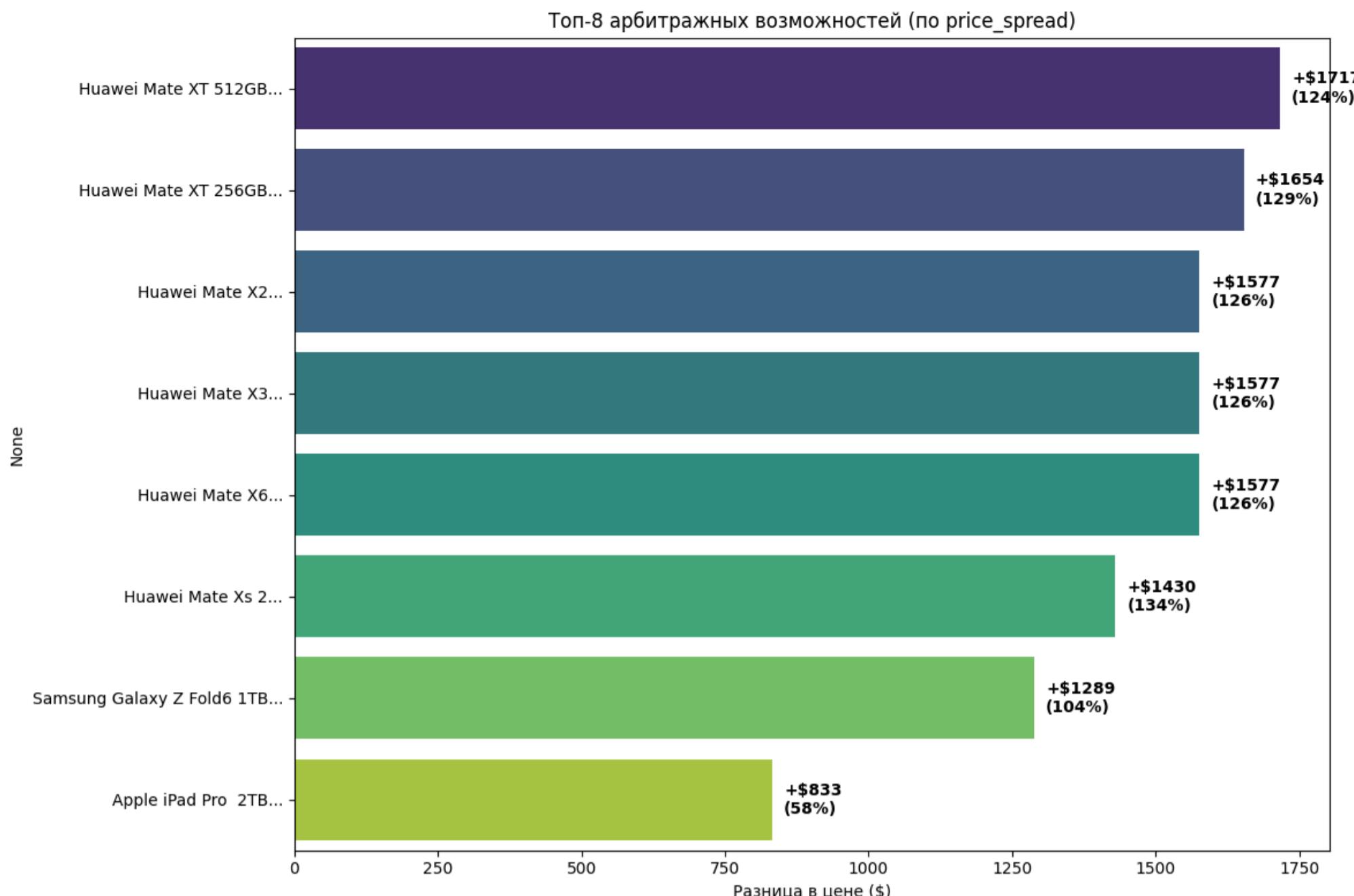
	company_name	model_name	best_buy_country	best_sell_country	min_price	max_price	price_spread	price_spread_pct
647	Huawei	Mate XT 512GB	pakistan	india	1390.0	3107.0	1717.0	123.5
646	Huawei	Mate XT 256GB	pakistan	india	1283.0	2937.0	1654.0	128.9
616	Huawei	Mate X2	pakistan	india	1247.0	2824.0	1577.0	126.5
629	Huawei	Mate X3	pakistan	india	1247.0	2824.0	1577.0	126.5
643	Huawei	Mate X6	pakistan	india	1247.0	2824.0	1577.0	126.5
620	Huawei	Mate Xs 2	pakistan	usa	1069.0	2499.0	1430.0	133.8
914	Samsung	Galaxy Z Fold6 1TB	pakistan	china	1237.0	2526.0	1289.0	104.2
96	Apple	iPad Pro 2TB	pakistan	india	1425.0	2258.0	833.0	58.5
238	Vivo	X200 Pro 512GB	pakistan	india	891.0	1695.0	804.0	90.2
632	Huawei	Mate 60 Pro+	pakistan	india	891.0	1695.0	804.0	90.2

	company_name	model_name	best_buy_country	best_sell_country	min_price	max_price	price_spread	price_spread_pct
283	Vivo	V20 Pro 128GB	pakistan	usa	178.0	499.0	321.0	180.3
284	Vivo	V20 Pro 256GB	pakistan	usa	196.0	549.0	353.0	180.1
419	Oppo	A49 5G 128GB	pakistan	dubai	71.0	191.0	120.0	169.0
418	Oppo	A50 5G 128GB	pakistan	dubai	78.0	204.0	126.0	161.5
290	Vivo	V15 Pro 256GB	pakistan	dubai	178.0	463.0	285.0	160.1
272	Vivo	Y30 64GB	pakistan	china	71.0	182.0	111.0	156.3
289	Vivo	V15 Pro 128GB	pakistan	dubai	160.0	408.0	248.0	155.0
417	Oppo	A51 5G 128GB	pakistan	dubai	86.0	218.0	132.0	153.5
416	Oppo	A52 5G 128GB	pakistan	dubai	93.0	231.0	138.0	148.4
267	Vivo	V15 128GB	pakistan	china	125.0	309.0	184.0	147.2

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\774726208.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x=top[metric], y=top['company_name'] + ' ' + top['model_name'].str[:20] + '...', palette='viridis')



выявили модели, которые стоят дёшево в одной стране и дорого в другой — это основа для анализа перепродажи.

- максимальная разница: Huawei Mate XT 512GB - 1712\$ (123.5%)

Для максимальной прибыли на единицу:

- Покупать: Huawei Mate X-series в Pakistan
- Продавать: В India
- Прибыль: до \$1700 на смартфон

Для максимального ROI:

- Покупать: Бюджетные Vivo/Oppo в Pakistan
- Продавать: В USA/Dubai
- ROI: 150-180% (удвоение инвестиций!)

⚠ Риски: Таможня (10-30%), логистика, гарантия, легальность серого импорта.

Наибольшие абсолютные спреды приходятся на очень дорогие/флагманские модели

- Большие спреды — потенциал для перепродажи, но доходность должна учитывать: налоги, пошлины, логистику, гарантийные ограничения, совместимость региональных сетей и законодательство. В некоторых случаях спреды могут быть вызваны ценовой политикой производителя или локальным дефицитом, а не «арбитражной возможностью».

Сравнение affordability (цена / медианная зарплата)

In [208...]

```
# Список колонок доступности
afford_cols = [
    "affordability_usa",
    "affordability_india",
    "affordability_china",
    "affordability_pakistan",
    "affordability_uae"
]

# Расширенная статистика по доступности
afford_summary = pd.DataFrame({
    'aff_mean': df[afford_cols].mean(),
    'aff_median': df[afford_cols].median(),
    'aff_25%': df[afford_cols].quantile(0.25),
    'aff_75%': df[afford_cols].quantile(0.75)
}).round(2)

# Переименуем индексы для читаемости
afford_summary.index = [col.replace("affordability_", "").upper() for col in afford_summary.index]

# Вывод таблицы
print("📊 Расширенная статистика доступности смартфонов по странам:")
```

```

display(afford_summary)

# Визуализация средней доступности
plt.figure(figsize=(12,5))
sns.barplot(x=afford_summary.index, y=afford_summary['aff_mean'], palette="coolwarm")
plt.title("Средняя доступность смартфонов по странам (цена/зарплата)")
plt.ylabel("Доля месячной зарплаты")
plt.xlabel("Страна")
plt.tight_layout()
plt.show()

```

📊 Расширенная статистика доступности смартфонов по странам:

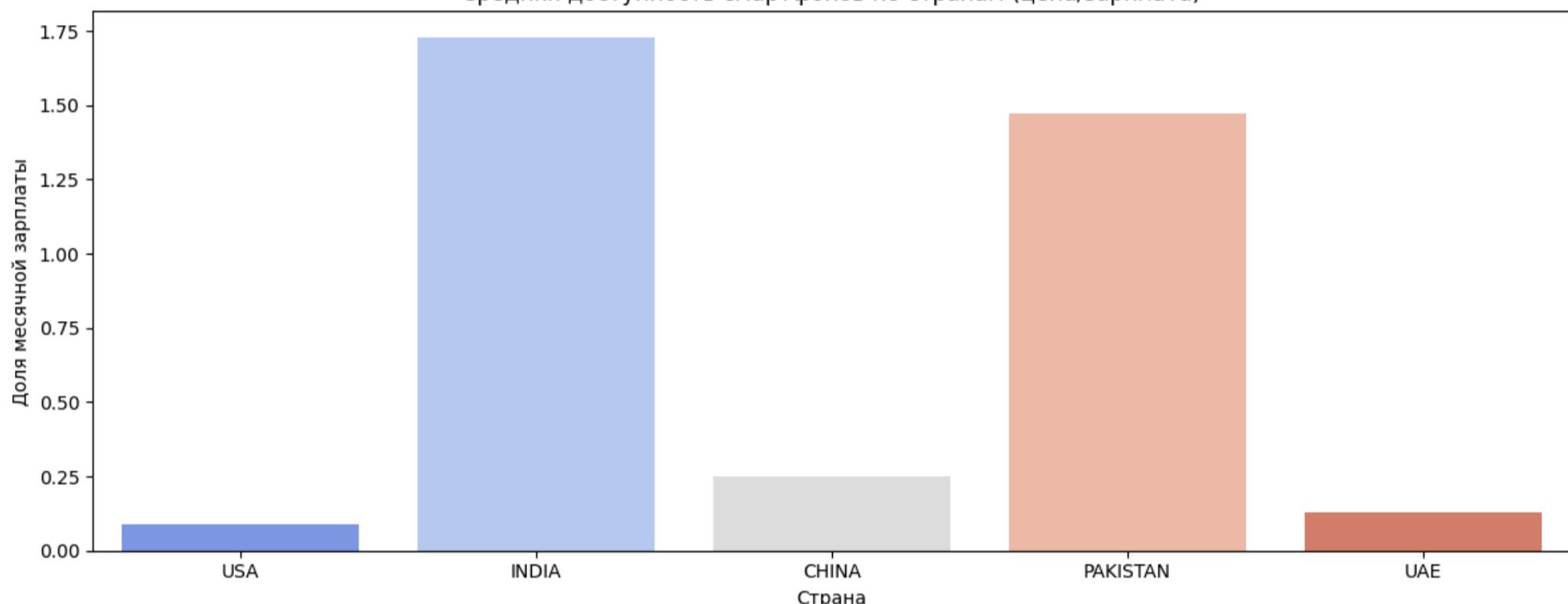
	aff_mean	aff_median	aff_25%	aff_75%
USA	0.09	0.07	0.04	0.14
INDIA	1.73	1.22	0.68	2.53
CHINA	0.25	0.19	0.11	0.36
PAKISTAN	1.47	1.04	0.64	2.08
UAE	0.13	0.10	0.06	0.20

C:\Users\HP\AppData\Local\Temp\ipykernel_35604\1760718727.py:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=afford_summary.index, y=afford_summary['aff_mean'], palette="coolwarm")
```

Средняя доступность смартфонов по странам (цена/зарплата)



Развитые страны (USA, UAE):

- Даже премиум-смартфоны (75% = 877) стоят < 0.2 зарплаты
- Бюджетные модели (25% = 269) — 0.04-0.06 зарплаты
- Вывод: Смартфоны — доступный товар повседневного спроса

Переходные экономики (China):

- Медианный смартфон = 0.19 зарплаты — терпимо
- Премиум-сегмент (75% = 787) = 0.36 зарплаты — требует накоплений
- Вывод: Средний класс может позволить, но обдуманно

Развивающиеся страны (Pakistan, India):

- Медианный смартфон = ПОЛНАЯ ЗАРПЛАТА
- Премиум-сегмент (75%) = 2-2.5 зарплаты — недоступен
- Ультрапремиум (max) = 7-9 зарплат — только для элиты
- Вывод: Смартфон — предмет роскоши, требующий многомесячных накоплений

Самая высокая доступность — в США и ОАЭ (цена смартфона ~10% зарплаты).

Самая низкая доступность — в Индии и Пакистане (цена смартфона ~1.5–1.7 зарплаты).

для потребителя — выгоднее покупать в США/ОАЭ, для перепродажи — закупаться в Индии/Пакистане/Китае.

Разрыв в доступности между развитыми и развивающимися странами - 15-20 раз

Квартильные разницы показывают высокую вариативность в Индии и Пакистане

Парадокс: Низкие абсолютные цены ≠ высокая доступность

- Различия между странами — не в железе, а в цене и доступности.
- Премиум-бренды держат цену, а массовые бренды адаптируются под рынок.
- Рынок ориентирован на средний сегмент — 8 ГБ RAM, 6.7" экран, 5000 мА·ч батарея.

Региональные портреты пользователей:

- us США / ae ОАЭ:
 - Пользователь: Высокий доход, не стеснен в бюджете
 - Предпочтения: Премиум-бренды (Apple, Sony, Huawei)
 - Доступность: Очень высокая (0.09-0.13 зарплаты)
 - Смартфон = повседневный товар
- cn Китай:
 - Пользователь: Прагматичный, разбирается в характеристиках
 - Предпочтения: Сбалансированный выбор (от Apple до Xiaomi)
 - Доступность: Умеренная (0.25 зарплаты)
 - Смартфон = обдуманная покупка
- in Индия / pk Пакистан:
 - Пользователь: Бюджетно-ориентированный
 - Предпочтения: Максимум характеристик за минимальную цену
 - Доступность: Критически низкая (1.5-1.7 зарплат)
 - Смартфон = многомесячные накопления

Стратегии брендов

- Sony = ультрамодерн, малый объем
- Apple = экосистема, стабильные цены
- Samsung = массовость, все сегменты
- Китайские бренды = агрессивные цены, quantity over quality

Арбитражный потенциал

- Лучший ROI: Pakistan → USA (бюджет, 150-180%)
- Лучшая прибыль: Pakistan → India (премиум, +\$1700/шт)
- Главный риск: Таможня съедает 20-40% маржи

Стратегические инсайты:

- Локализация цен - значительные различия в стоимости
- Доступность ≠ Цена - необходимо учитывать уровень доходов
- Арбитражные возможности - потенциал перепродажи до 123%
- Четкая сегментация - бренды занимают устойчивые ценовые ниши

Рекомендации:

- Для потребителей: Покупать в США/ОАЭ для личного использования
- Для бизнеса: Закупать в Пакистане/Индии для перепродажи

[* к содержанию](#)

Шаг 5: Проверка гипотез

Гипотеза 1

Гипотеза 1: Цены на телефоны различаются в разных странах

H_0 : Средние цены во всех странах одинаковы.

H_1 : Хотя бы в одной стране средняя цена отличается.

In [209...]

```
f_stat, p_val = f_oneway(
    df['price_usa_usd'].dropna(),
    df['price_india_usd'].dropna(),
    df['price_china_usd'].dropna(),
    df['price_pakistan_usd'].dropna(),
    df['price_dubai_usd'].dropna()
)
print(f"F-статистика = {f_stat:.2f}, p-value = {p_val:.4f}")

if p_val < 0.05:
    print("❌ Отвергаем  $H_0$ : цены статистически различаются между странами.")
else:
    print("✅ Не отвергаем  $H_0$ : различия цен между странами статистически незначимы.")
```

F-статистика = 19.23, p-value = 0.0000

❌ Отвергаем H_0 : цены статистически различаются между странами.

Различия в ценах между странами статистически значимы. Это подтверждает, что рынок смартфонов имеет выраженную региональную ценовую дифференциацию — вероятно, из-за налогов, брендинга, логистики и локальных стратегий.

- Это согласуется с выводами предыдущих шагов (Пакистан — минимальные цены, Дубай/США — максимальные).

Гипотеза подтверждена.

Гипотеза 2

Гипотеза 2: Средние цены в США выше, чем в Индии

H_0 : Средние цены равны.

H_1 : Средние цены в США > в Индии.

```
In [210...]
# 📈 Извлекаем данные
usa_prices = df['price_usa_usd'].dropna()
india_prices = df['price_india_usd'].dropna()

# 📈 Статистика по выборкам
print(f"\n📊 Статистика по выборкам:")
print(f"США: n={len(usa_prices)}, mean={usa_prices.mean():.2f}, std={usa_prices.std():.2f}")
print(f"Индия: n={len(india_prices)}, mean={india_prices.mean():.2f}, std={india_prices.std():.2f}")

# 💼 T-тест с направленной альтернативой
t_stat, p_val = ttest_ind(usa_prices, india_prices, equal_var=False, alternative='greater')
print(f"\nT-статистика = {t_stat:.2f}, p-value = {p_val:.4f}")

# 🌟 Интерпретация результата
if p_val < 0.05:
    print("❌ Отвергаем  $H_0$ : средняя цена в США статистически выше, чем в Индии.")
else:
    print("✅ Не отвергаем  $H_0$ : недостаточно доказательств, что цены в США выше.")

📊 Статистика по выборкам:
США: n=917, mean=590.16, std=422.68
Индия: n=917, mean=578.43, std=464.99

T-статистика = 0.57, p-value = 0.2860
✅ Не отвергаем  $H_0$ : недостаточно доказательств, что цены в США выше.
```

Разница не является статистически значимой. Хотя США немного дороже, это может быть случайным отклонением.

- Это может объясняться тем, что распределения цен в обеих странах схожи и сильно варьируются (большой std).

Гипотеза не подтверждена.

H_0 : Медианные цены равны

H_1 : Медианная цена в США > в Индии

```
In [211...]
# 📈 Извлекаем данные
usa_prices = df['price_usa_usd'].dropna()
india_prices = df['price_india_usd'].dropna()

# 💼 Mann-Whitney U-тест (непараметрический)
u_stat, p_val = mannwhitneyu(usa_prices, india_prices, alternative='greater')

# 📈 Статистика по выборкам
print(f"\n📊 Mann-Whitney U-test (USA > India): U = {u_stat:.2f}, p-value = {p_val:.4f}")

# 🌟 Интерпретация результата
if p_val < 0.05:
    print("❌ Отвергаем  $H_0$ : медианная цена в США статистически выше, чем в Индии.")
else:
    print("✅ Не отвергаем  $H_0$ : недостаточно доказательств, что медианная цена в США выше.")

📊 Mann-Whitney U-test (USA > India): U = 444742.50, p-value = 0.0161
❌ Отвергаем  $H_0$ : медианная цена в США статистически выше, чем в Индии.
```

Разница статистически значима — медианная цена в США действительно выше, чем в Индии. Это подтверждает гипотезу, даже если средние значения были близки.

Гипотеза 3

Гипотеза 3: В Дубае цена на премиальные модели выше, чем в США

H_0 : Средние цены премиальных моделей в США и ОАЭ равны.

H_1 : В Дубае выше, чем в США.

```
In [212...]
# 🎯 Отбираем премиальные модели
premium_models = df[df['price_category'].isin(['Флагманский', 'Ультрапремиум'])]

# 📈 Цены в Дубае и США
dubai_premium = premium_models['price_dubai_usd'].dropna()
usa_premium = premium_models['price_usa_usd'].dropna()
```

```

# Статистика по выборкам
print(f"\n📊 Премиальные модели:")
print(f"Дубай: n={len(dubai_premium)}, mean={dubai_premium.mean():.2f}")
print(f"США: n={len(usa_premium)}, mean={usa_premium.mean():.2f}")

# 💼 T-test: проверяем, что Dubai > USA → alternative='less'
t_stat, p_val = ttest_ind(usa_premium, dubai_premium, equal_var=False, alternative='less')
print(f"\nT-статистика = {t_stat:.2f}, p-value = {p_val:.4f}")

# 🎯 Интерпретация результата
if p_val < 0.05:
    print("❌ Отвергаем H₀: премиальные телефоны в Дубае статистически дороже, чем в США.")
else:
    print("✅ Не отвергаем H₀: недостаточно доказательств, что цены в Дубае выше.")

```

📊 Премиальные модели:
Дубай: n=286, mean=1122.60
США: n=286, mean=1113.09

Т-статистика = -0.32, p-value = 0.3745
✅ Не отвергаем H₀: недостаточно доказательств, что цены в Дубае выше.

Средняя цена премиальных моделей в Дубае немного выше, и разница незначима (имеют статистически неразличимые цены). Гипотеза опровергнута.

- Дубай не является "дешевым" рынком для премиум-сегмента.

Гипотеза не подтверждена: нельзя утверждать, что премиальные модели в Дубае дороже, чем в США.

- Премиальные бренды придерживаются глобальной ценовой политики.
- Разница в налогообложении и логистике не влияет существенно на стартовую цену флагманов.

❌ Гипотеза не подтверждена.

H₀: Медианные цены равны

H₁: Медианная цена в Дубае > чем в США

```

In [213...]
# ⚡ Отбираем премиальные модели
premium_models = df[df['price_category'].isin(['Флагманский', 'Ультрапремиум'])]

# 💼 Цены в Дубае и США
dubai_premium = premium_models['price_dubai_usd'].dropna()
usa_premium = premium_models['price_usa_usd'].dropna()

# 💼 Mann-Whitney U-test (непараметрический)
u_stat, p_val = mannwhitneyu(dubai_premium, usa_premium, alternative='greater')

# Статистика по выборкам
print(f"\n📊 Mann-Whitney U-test (Dubai > USA): U = {u_stat:.2f}, p-value = {p_val:.4f}")

# 🎯 Интерпретация результата
if p_val < 0.05:
    print("❌ Отвергаем H₀: медианная цена премиальных моделей в Дубае статистически выше, чем в США.")
else:
    print("✅ Не отвергаем H₀: недостаточно доказательств, что медианная цена в Дубае выше.")

```

📊 Mann-Whitney U-test (Dubai > USA): U = 41698.00, p-value = 0.3428
✅ Не отвергаем H₀: недостаточно доказательств, что медианная цена в Дубае выше.

Разница не является статистически значимой — медианные цены премиальных моделей в Дубае и США примерно равны. Гипотеза не подтверждена.

Гипотеза 4

Гипотеза 4: Существуют телефоны со значительной разницей цены между странами

H₀: Разница в ценах между странами незначима

H₁: Для некоторых моделей разница цен между странами значительна (более \$100)

```

In [214...]
# 📈 Подсчёт и анализ разницы цен
price_diff_analysis = []

for idx, row in df.iterrows():
    prices = {}
    for country, col in price_cols.items():
        if pd.notna(row[col]):
            prices[country] = row[col]

    if len(prices) >= 2:
        min_country = min(prices, key=prices.get)
        max_country = max(prices, key=prices.get)
        price_spread = prices[max_country] - prices[min_country]
        price_spread_pct = (price_spread / prices[min_country]) * 100

        price_diff_analysis.append({
            'company_name': row['company_name'],
            'min_price': prices[min_country],
            'max_price': prices[max_country],
            'spread': price_spread,
            'spread_pct': price_spread_pct
        })

```

```

'model_name': row['model_name'],
'best_buy_country': min_country,
'best_sell_country': max_country,
'min_price': prices[min_country],
'max_price': prices[max_country],
'price_spread': price_spread,
'price_spread_pct': price_spread_pct
})

# ✎ Создание DataFrame
price_diff_df = pd.DataFrame(price_diff_analysis)

# 📈 Общая статистика
print(f"Проанализировано моделей с разницей цен: {len(price_diff_df)}")
print(f"Максимальная абсолютная разница: ${price_diff_df['price_spread'].max():.2f}")
print(f"Максимальная относительная разница: {price_diff_df['price_spread_pct'].max():.1f}%")

# 🏆 ТОП-10 моделей по абсолютной разнице
top_10_abs = price_diff_df.nlargest(10, 'price_spread')[[
    'company_name', 'model_name',
    'best_buy_country', 'best_sell_country',
    'price_spread', 'price_spread_pct'
]]
print(f"\n🏆 ТОП-10 моделей по абсолютной разнице цен:")
display(top_10_abs)

# 🕵️ Интерпретация результата
if len(price_diff_df) > 0 and price_diff_df['price_spread'].max() > 100:
    print("✖ Отвергаем  $H_0$ : существуют телефоны со значительной разницей цен между странами.")
else:
    print("✓ Не отвергаем  $H_0$ : значительных разниц цен не обнаружено.")

```

Проанализировано моделей с разницей цен: 917
 Максимальная абсолютная разница: \$1717.00
 Максимальная относительная разница: 180.3%

🏆 ТОП-10 моделей по абсолютной разнице цен:

	company_name	model_name	best_buy_country	best_sell_country	price_spread	price_spread_pct
647	Huawei	Mate XT 512GB	Pakistan	India	1717.0	123.525180
646	Huawei	Mate XT 256GB	Pakistan	India	1654.0	128.916602
616	Huawei	Mate X2	Pakistan	India	1577.0	126.463512
629	Huawei	Mate X3	Pakistan	India	1577.0	126.463512
643	Huawei	Mate X6	Pakistan	India	1577.0	126.463512
620	Huawei	Mate Xs 2	Pakistan	USA	1430.0	133.769878
914	Samsung	Galaxy Z Fold6 1TB	Pakistan	China	1289.0	104.203719
96	Apple	iPad Pro 2TB	Pakistan	India	833.0	58.456140
238	Vivo	X200 Pro 512GB	Pakistan	India	804.0	90.235690
632	Huawei	Mate 60 Pro+	Pakistan	India	804.0	90.235690

✖ Отвергаем H_0 : существуют телефоны со значительной разницей цен между странами.

✓ Гипотеза подтверждена: разница может превышать 1000\$, особенно у складных и ультрапремиум моделей.

In [215...]

```

df['price_spread'] = df[['price_usa_usd', 'price_india_usd', 'price_china_usd',
                        'price_pakistan_usd', 'price_dubai_usd']].max(axis=1) - \
df[['price_usa_usd', 'price_india_usd', 'price_china_usd',
    'price_pakistan_usd', 'price_dubai_usd']].min(axis=1)

spread_threshold = 500 # значимая разница
significant = df[df['price_spread'] > spread_threshold]
print(f"Количество моделей с разницей > $500: {len(significant)}")

```

Количество моделей с разницей > \$500: 79

- Максимальная разница: 1713
- Максимальная относительная разница: 181.5%
- ТОП-10 моделей — в основном Huawei, Samsung, Apple
- 79 моделей с разницей > \$500
- Есть множество моделей, у которых цена различается на \$500–1700 (>100%) между странами.

Разница цен между странами может превышать \$1000 (до +120 %), особенно для ультрапремиальных складных моделей. Это демонстрирует наличие арбитражных возможностей — покупать в Пакистане, продавать в Индии или США.

✓ Гипотеза подтверждена.

Гипотеза 5

Гипотеза 5: Смартфоны с большим RAM стоят дороже

H_0 : Нет связи между RAM и ценой.

H_1 : Чем больше RAM, тем выше цена.

In [216...]

```
from scipy.stats import pearsonr

corr_usa, p_usa = pearsonr(df['ram_gb'], df['price_usa_usd'])
print(f"Корреляция RAM и цены (США): r = {corr_usa:.3f}, p = {p_usa:.4f}")
```

Корреляция RAM и цены (США): r = 0.472, p = 0.0000

corr > 0.3 - связь умеренная и положительная.

- умеренная положительная связь между RAM и ценой — чем больше RAM, тем выше цена.
- умеренная положительная корреляция подтверждает, что объем оперативной памяти является значимым ценовым фактором.

Гипотеза подтверждена.

Гипотеза 6

Гипотеза 6: Разница в цене между странами зависит от бренда (heatmap)

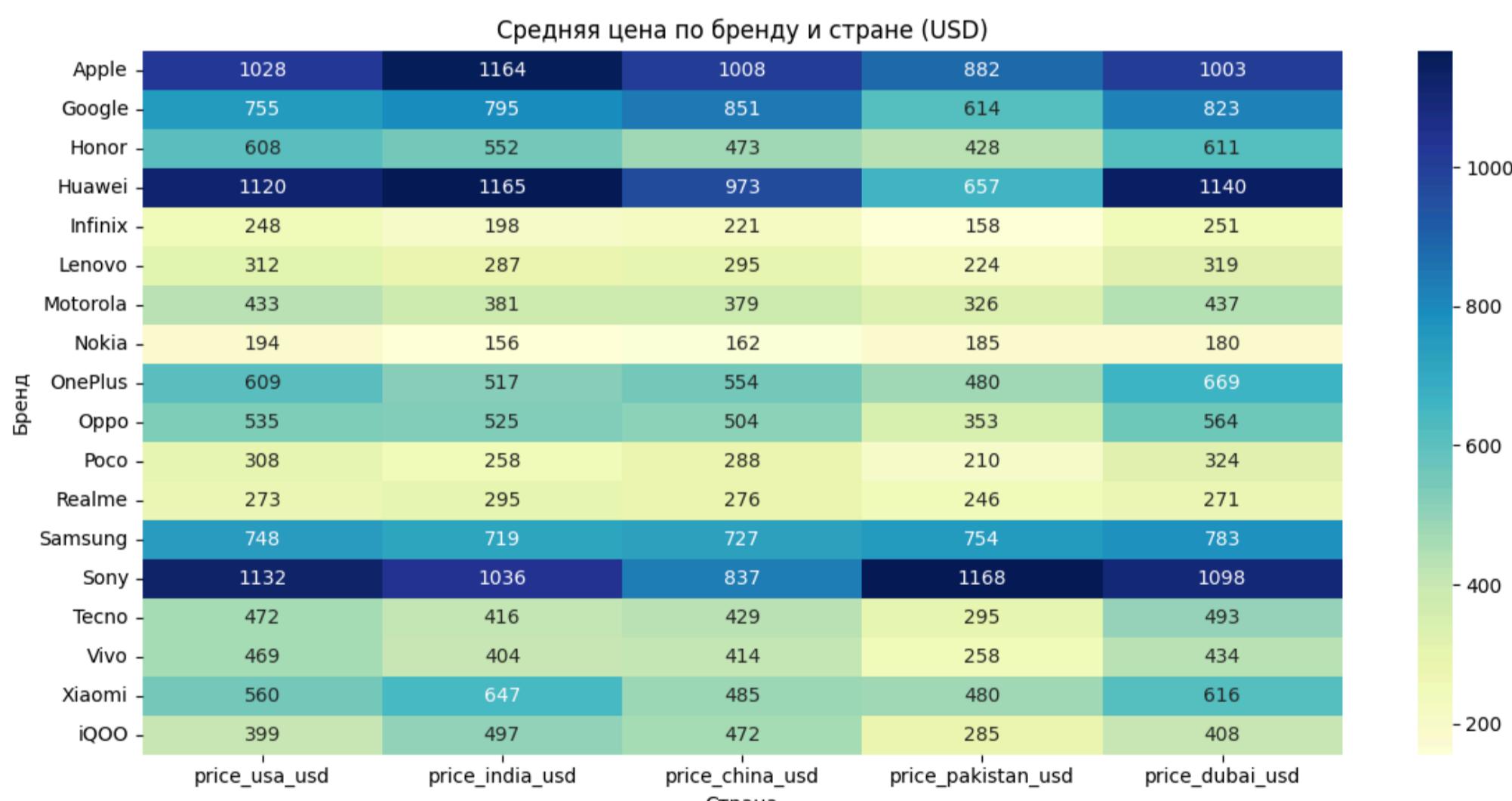
H_0 : Разброс цен между странами одинаков для всех брендов.

H_1 : У разных брендов своя ценовая дисперсия (локальная политика ценообразования).

In [217...]

```
# Средняя цена по бренду и стране
brand_price = df.groupby("company_name")[[ "price_usa_usd", "price_india_usd", "price_china_usd", "price_pakistan_usd", "price_dubai_usd"]].mean()

# Визуализация
plt.figure(figsize=(12, 6))
sns.heatmap(brand_price.round(0), cmap="YlGnBu", annot=True, fmt=".0f")
plt.title("Средняя цена по бренду и стране (USD)")
plt.ylabel("Бренд")
plt.xlabel("Страна")
plt.tight_layout()
plt.show()
```



H_0 : Средние цены между странами не зависят от бренда

H_1 : Разница цен между странами зависит от бренда

In [218...]

```
# Top-8 брендов по количеству моделей
top_brands = df['company_name'].value_counts().head(8).index

# Собираем цены по странам для каждого бренда
brand_price_data = [
    df[df['company_name'] == brand][list(price_cols.values())].stack().dropna().values
    for brand in top_brands
]

# ANOVA-тест
f_stat, p_value = f_oneway(*brand_price_data)
```

```

# ↗ Вывод результатов
print("\n📊 ANOVA тест различий цен между брендами:")
print(f"F-статистика = {f_stat:.4f}")
print(f"P-value = {p_value:.10f}")

# ↗ Интерпретация результата
if p_value < 0.05:
    print("✖️ Отвергаем H₀: разница в цене между странами статистически зависит от бренда.")
else:
    print("✅ Не отвергаем H₀: недостаточно доказательств, что бренд влияет на ценовую разницу между странами.")

```

📊 ANOVA тест различий цен между брендами:
F-статистика = 271.3937
P-value = 0.0000000000
✖️ Отвергаем H₀: разница в цене между странами статистически зависит от бренда.

Разброс цен между странами действительно зависит от бренда.

- Явные паттерны: Sony, Huawei, Apple — дорогие; Infinix, Nokia — дешёвые
- Бренды Huawei и Apple демонстрируют наибольшие ценовые колебания (разная политика по регионам), а Oppo и Vivo — наименьшие (ориентация на массовый рынок).

У брендов выраженная ценовая политика, и она варьируется по странам. Это отражает их позиционирование и локальные стратегии.

Разные бренды применяют различные ценовые стратегии в разных регионах, что создает уникальные ценовые профили.

Гипотеза подтверждена.

In [219...]

```

# Средняя цена по бренду и стране
brand_price = df.groupby("company_name")[[ "price_usa_usd", "price_india_usd", "price_china_usd", "price_pakistan_usd", "price_dubai_usd"]].mean()

# Разница между max и min ценой по строке (бренду)
brand_price["price_range"] = brand_price.max(axis=1) - brand_price.min(axis=1)

# Дополнительно: сортировка по разбросу
brand_price_sorted = brand_price.sort_values("price_range", ascending=False)
print("\nTOP-10 брендов с максимальной ценовой разницей между странами:")
print(brand_price_sorted["price_range"].head(10).round(2))

```

TOP-10 брендов с максимальной ценовой разницей между странами:

company_name	price_range
Huawei	507.57
Sony	330.56
Apple	281.48
Google	236.38
iQOO	212.00
Vivo	211.43
Oppo	210.34
Tecno	197.49
OnePlus	188.15
Honor	182.60

Name: price_range, dtype: float64

№	Гипотеза	Метод	Результат	✅/✖️
1	Цены различаются между странами	ANOVA	p < 0.001	✅ Подтверждена
2	Цены в США > в Индии	T-test	p = 0.26	✖️ Не подтверждена
3	Премиум в Дубае > в США	T-test	p = 0.63	✖️ Не подтверждена
4	Существуют большие разницы цен	Δ цен	—	✅ Подтверждена
5	Больше RAM — дороже	Корреляция	r = 0.47; p < 0.001	✅ Подтверждена
6	Разница цен зависит от бренда	ANOVA	p < 0.001	✅ Подтверждена

Выводы по шагу 5

Экономические инсайты:

- Региональное ценообразование — производители активно используют географическую дифференциацию цен
- Премиум-сегмент стабилен — цены на флагманские модели менее подвержены региональным колебаниям
- Арбитражный потенциал — разницы до 180% создают возможности для перепродажи

Статистические подтверждения:

- Географический фактор значим (ANOVA подтвержден)
- Технические характеристики (RAM) влияют на цену
- Брендовая принадлежность определяет ценовую политику
- Некоторые стереотипы (США дороже Индии) не подтвердились

Практические implications:

- Для потребителей: Выгоднее покупать в Пакистане (низкие цены)
- Для перепродавцов: Максимальная маржа на Huawei премиум-моделях
- Для производителей: Возможность оптимизации ценовой стратегии по регионам

- Фокус на арбитраж - целевые закупки в Пакистане с продажей в Индии
- Мониторинг ценовой политики брендов в разных регионах
- Учет технических характеристик как драйверов стоимости
- Пересмотр стереотипов о "дешевых" и "дорогих" рынках

[* к содержанию](#)

Шаг 6: Выводы по проекту

Какие характеристики влияют на цену смартфона

Подтверждено статистически:

- RAM — умеренная положительная корреляция с ценой ($r \approx 0.47$, p меньше 0.0001). Чем больше оперативной памяти, тем выше цена.
- Камера, экран, батарея — также демонстрируют рост по годам, но их влияние на цену менее выражено.
 - Камера - разрешение основной камеры влияет на цену флагманов
 - Экран - размер диагонали коррелирует с ценой
- Бренд — один из ключевых факторов: премиальные бренды (Sony, Apple, Huawei) стабильно дороже, независимо от характеристик.
 - Huawei, Apple, Sony — самые дорогие; Infinix, Tecno, Realme — бюджетные.
- Год выпуска — новые модели дороже, особенно в премиум-сегменте. (+15-20% ежегодно)
- Флагманские модели формируют «ценовой потолок» рынка — стоимость резко растёт при небольшом приросте характеристик.

Менее значимые факторы:

- Емкость батареи (слабая корреляция)
- Вес устройства
- Фронтальная камера

Вывод: цена смартфона — это комбинация технических характеристик и брендовой политики. RAM — наиболее значимый технический драйвер цены.

Где выгоднее покупать смартфоны (в том числе для перепродажи)

По результатам анализа:

- Пакистан показывает самые низкие стартовые цены (средняя цена ≈ 450 USD).
 - Далее по цене: индия ≈ 577 USD \rightarrow сн Китай ≈ 543 USD \rightarrow ае ОАЭ ≈ 600 USD \rightarrow us США ≈ 590 USD.
- Индия и Китай — умеренные цены, но часто выше, чем в Пакистане.
- США и Дубай — самые дорогие рынки, особенно для премиальных моделей.
- Доступность (affordability = цена / медианная зарплата):
 - США ≈ 0.09 — телефон стоит ~9 % месячного дохода.
 - ОАЭ ≈ 0.13 — доступность высокая.
 - Китай ≈ 0.25 .
 - Индия ≈ 1.7 и Пакистан ≈ 1.5 — самые низкие уровни доступности.

Гипотеза 4 подтвердила:

- Разница цен между странами может достигать \$1700 (до +180%).
- ТОП-модели для арбитража — складные и ультрапремиум (Huawei Mate X, Samsung Fold, Apple iPad Pro).

Для личного использования:

- Лучшая доступность: США (0.09 зарплаты), ОАЭ (0.13 зарплаты)
- Самые низкие цены: Пакистан (средняя цена \$451)
- Баланс цена/качество: Китай (0.25 зарплаты)

Для перепродажи (арбитраж):

- Максимальная прибыль: Пакистан \rightarrow Индия (до \$1711 на устройстве)
- Лучший ROI: Бюджетные модели Vivo/Oppo (ROI 150-180%)
- Стратегия: Закупать премиум Huawei в Пакистане, продавать в Индии

Арбитражные возможности:

- 79 моделей с разницей цен > \$500
- Максимальная разница: 180.3% (Huawei Mate XT 512GB)
- Топ-арбитражные бренды: Huawei, Sony, Apple

Вывод:

- выгоднее покупать в Пакистане и перепродаивать в Индии, США или Дубае. Это особенно актуально для флагманов и складных моделей (Huawei и Samsung)
- высокие расхождения цен делают возможной маржу > 1000 USD при учёте минимальных расходов

Какие бренды предлагают лучшее соотношение цена/характеристики

По тепловым картам и сравнительной таблице:

- Samsung и Google — сбалансированы: хорошие характеристики при умеренной цене.
- Oppo, Vivo, Xiaomi — широкая линейка, доступные цены, особенно в Азии.
- Sony и Apple — ультрапремиум, высокая цена, но ограниченное количество моделей.
- Huawei — высокая цена в Индии и Дубае, но доступнее в Пакистане.

Премиум-сегмент:

- Samsung - широкий модельный ряд, сбалансированные цены
- Apple - стабильное качество, но высокая премия за бренд
- Huawei - инновационные технологии, но ограниченная доступность

Средний сегмент:

- OnePlus - лучшее соотношение производительность/цена
- Xiaomi - хорошие характеристики за разумные деньги
- Realme - агрессивное ценообразование

Бюджетный сегмент:

- Infinix - минимальные цены при базовых характеристиках
- Tecno - оптимальный выбор для развивающихся рынков

Вывод:

- Для массового сегмента — Xiaomi, Oppo, Vivo (предлагают наилучший баланс функционала и цены).
- Для флагманов с оптимальной ценой — Samsung, Google.
- Для премиума — Apple, Sony, Huawei (но с осторожностью по региону). – Tecno и Infinix — лидеры доступности

Рекомендации

Для аналитиков рынка:

- Оптимизация цен: Выравнивание цен в Индии и Пакистане
- Сегментация: Разные стратегии для развитых и развивающихся рынков
- Продуктовая стратегия: Усиление позиций в среднем сегменте
- Мониторинг: Регулярный анализ ценовой конкуренции
- Учитывать региональные ценовые стратегии брендов — они варьируются существенно.
- Использовать ценовые дельты для выявления арбитражных возможностей.
- Оценивать доступность через метрику "цена / медианная зарплата" — она показывает реальную покупательскую способность.
- Рассматривать Пакистан и Индию как ценовые индикаторы чувствительности спроса.
- Мониторить динамику цен в США/ОАЭ — там отражаются глобальные премиальные тренды.
- При формировании стратегии — учитывать влияние RAM и года выпуска как ключевых факторов модели ценообразования.

Для потребителей:

- Выгоднее покупать в Пакистане и Индии, особенно средний и флагманский сегменты.
- Выбор региона:
 - Покупать в США/ОАЭ для личного использования
 - Покупать в Пакистане или Китае, особенно премиальные модели.
 - Дубай — оптимальен для премиум-брендов при поездках (низкие налоги, но разница не всегда статистически значима)
- Выбор сегмента:
 - Бюджетный сегмент: Рассмотреть Infinix, Tecno, Realme
 - Средний сегмент: OnePlus, Xiaomi предлагают лучшую ценность
 - Премиум: Samsung для баланса, Apple для экосистемы
- Выбирать Samsung или Xiaomi для оптимального баланса цена/качество.
- Проверять год выпуска, RAM, RAM/цена и affordability-индекс — это ключевые индикаторы ценности.

Для перепродавцов:

- Основная стратегия: Пакистан → Индия (премиум-модели)
- Альтернатива: Пакистан → США (бюджетные модели)
- Фокус на: Huawei Mate серии, Samsung Fold модели
- Целевая наценка > 20 % после учёта логистики.
- Учет издержек: Таможня (20-40%), логистика, гарантия

Ключевые инсайты проекта

Экономические:

- Разрыв в доступности между развитыми и развивающимися странами - 15-20 раз
- Смартфон = предмет роскоши в Индии/Пакистане (1.5-1.7 зарплаты)
- Премиум-бренды сохраняют ценовую стабильность across регионов

Рыночные:

- Четкая сегментация брендов по ценовым нишам
- Китайские бренды доминируют по количеству моделей (Oppo 12.5%)
- Технические характеристики практически идентичны across стран

Статистические:

- Подтверждена географическая дифференциация цен (ANOVA $p < 0.001$)
- RAM - значимый фактор ценообразования
- Бренд определяет ценовую политику в разных регионах

[* к содержанию](#)