ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		В.В. Боженко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

ИССЛЕДОВАТЕЛЬСКИЙ АНАЛИЗ ДАННЫХ 2024

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217		Д.М. Никитин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. **Цель работы:** изучение связи между признаками двумерного набора данных, визуализация данных.

2. **Порядок выполнения работы:**

1. Работа выполняется в https://colab.research.google.com/

2. Осуществите обработку csv-файлов с помощью pandas по вариантам (номер варианта определяется по номеру в списке группы).

Что нужно сделать:

1. Загрузить датасет с помощью библиотеки pandas.

2. Провести предварительную обработку данных (как в 1 лр). Обратите внимание, что наборы данных могут отличаться от наборов данных 1 лр. Скачивайте актуальный набор данных.

3. Построить точечную диаграмму (матрицу диаграмм рассеяния). Выполнить анализ полученной диаграммы, отвечая на вопрос показывает ли она в среднем определенную зависимость между переменными. Изучите параметры и опишите взаимосвязи. Если параметров слишком много – может потребоваться создать несколько графиков..

4. Исследовать взаимосвязь между переменными с помощью оценки коэффициента корреляции и ковариации. Выполнить интерпретацию результатов корреляции и ковариации, отвечая на вопросы о наличии (отсутствии) линейной взаимосвязи между переменными.

5. Построить heatmap (тепловую карту корреляции).

6. Постройте графики по заданию в варианте.

7. Сделайте выводе по работе.

3. **Вариант 9:**

Набор данных visits2.csv

Данные пользовательских сессии магазина:

1. уникальный идентификатор пользователя

2. страна пользователя

3. устройство пользователя

4. идентификатор рекламного источника, из которого пришел пользователь

5. дата и время начала сессии

6. дата и время окончания сессии

7. время сессии в минутах

8. кол-во кликов пользователя

9. количество товаров в корзине

10. стоимость покупок

11. возраст пользователя

Задание 1: использовать seaborn. По группировке - region и количество клиентов, привлеченных из рекламных источников каждого типа (channel) построить диаграмму следующего вида:

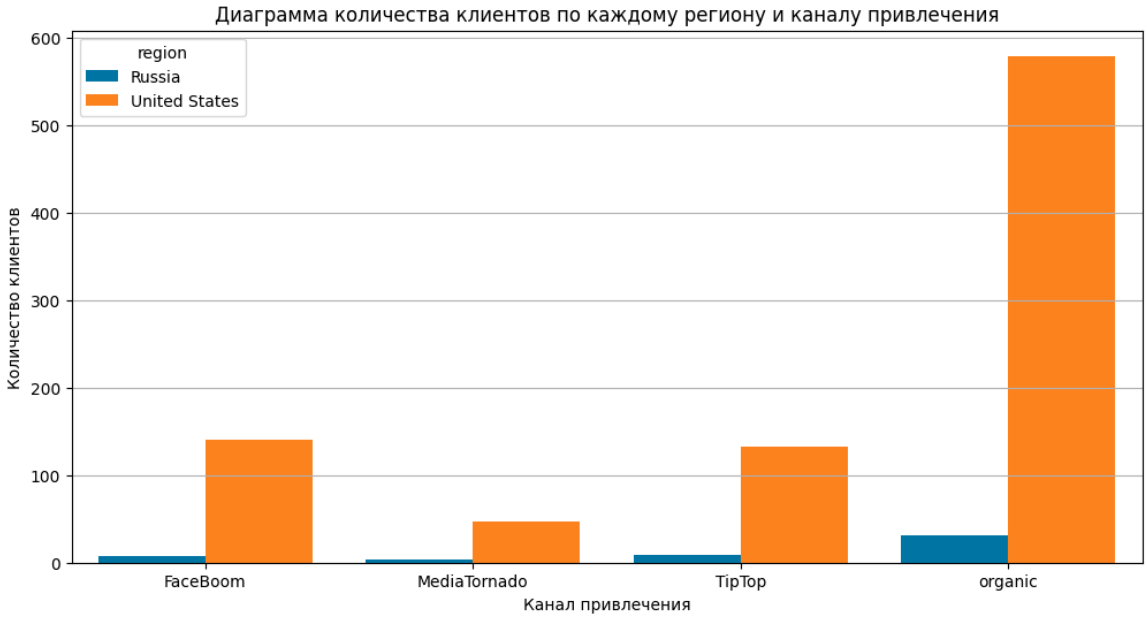


Рисунок 1 – Требуемый вид диаграммы задания 1

Задание 2: использовать pandas и plot. По сводной таблице (pivot_table) - отобразить уникальное (nunique) количество пользователей для каждого канала (channel). Оставить только маркеры в виде синего цвета размером 15.

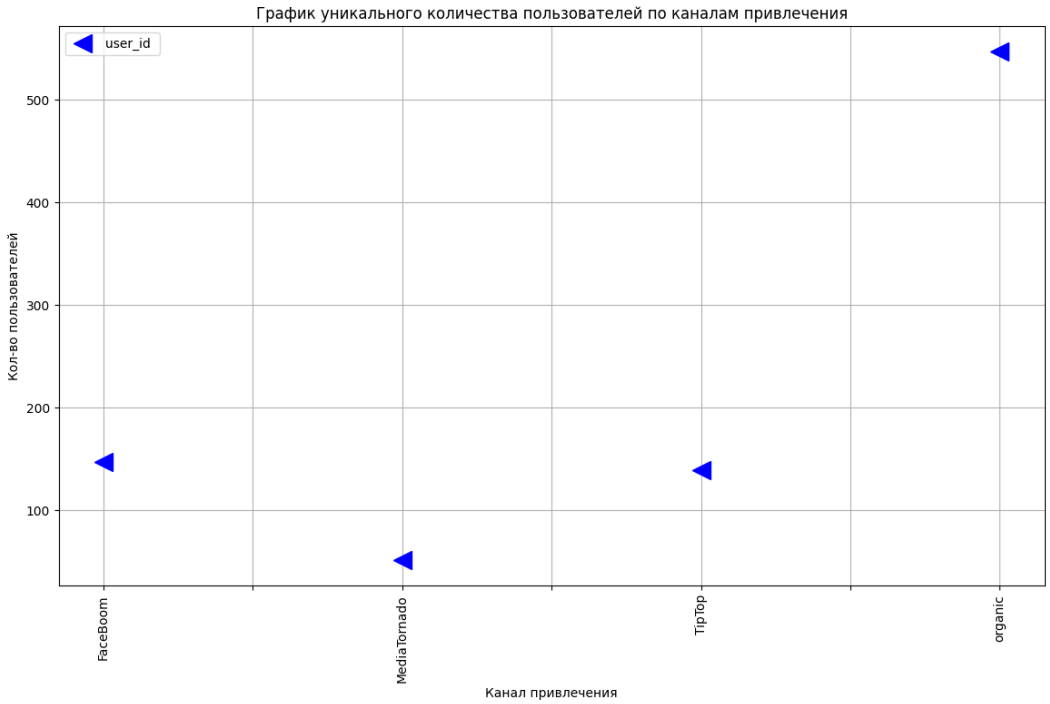


Рисунок 2 – Требуемый вид диаграммы задания 2

Задание 3: использовать matplotlib. Построить круговую диаграмму, которая отображает процент каждого устройства (device).

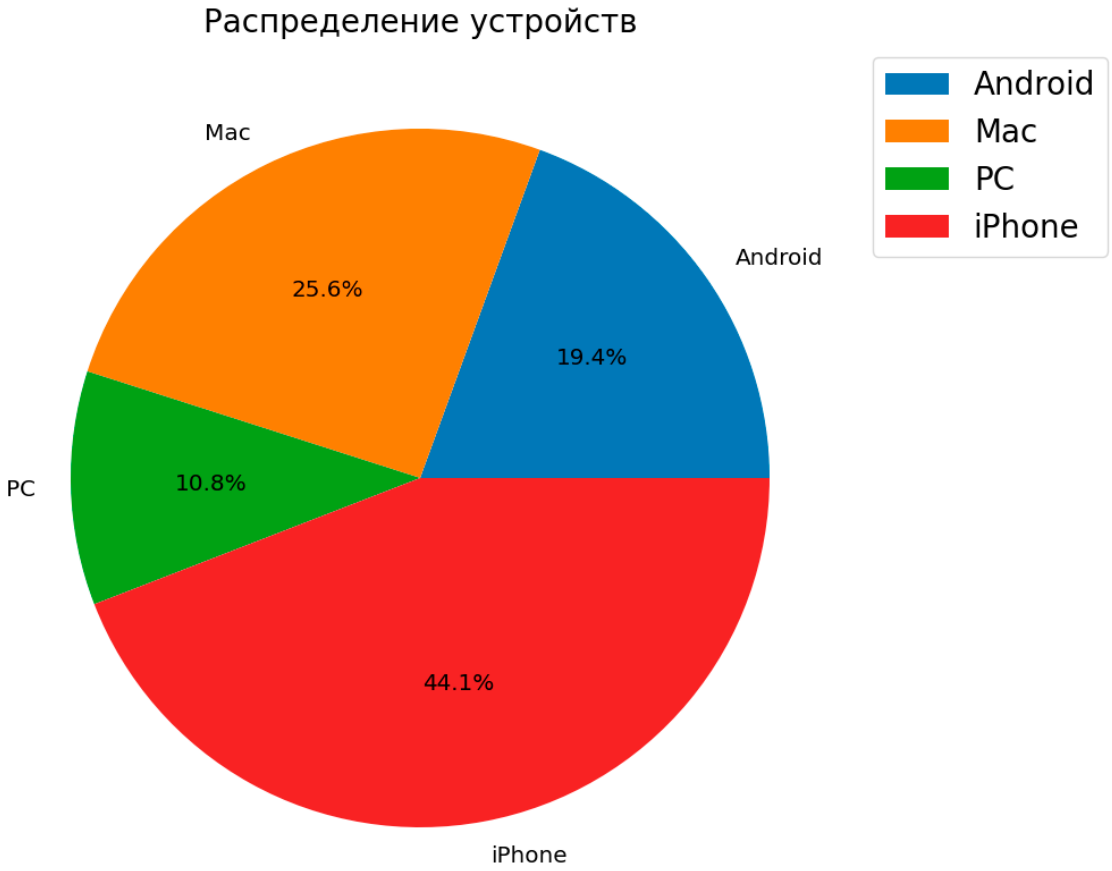


Рисунок 3 – Требуемый вид диаграммы задания 3

4. Ход работы:

Начнём работу с импорта нужных библиотек и считывания файла с данными. См. рис. 4.

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

db = pd.read_csv("visits2.csv", sep=";")
db.info()
✓ [16] 19ms
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 954 entries, 0 to 953
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   user_id        954 non-null    int64
 1   region         953 non-null    object
 2   device         953 non-null    object
 3   channel        954 non-null    object
 4   session_start  954 non-null    object
 5   session_end    954 non-null    object
 6   time_session   954 non-null    float64
 7   click_count    954 non-null    float64
 8   buy_count      954 non-null    float64
 9   price          954 non-null    float64
 10  age            954 non-null    int64
dtypes: float64(4), int64(2), object(5)
memory usage: 82.1+ KB
```

Рисунок 4 – Импорт библиотек и считывание данных

Все столбцы соответствуют варианту. Первым делом удостоверимся в виде данных. По полученной информации от функции .info() можно сделать вывод, что могут быть проблемы с типами данных "price", "time_session", "buy_count", "click_count", "session_start" и "session_end". Вероятнее всего нужно перевести их в datetime64 и int64, также заметно, что есть некоторые строки с пропусками, а также необходимо провести проверку на дубликаты. См. рис. 5.

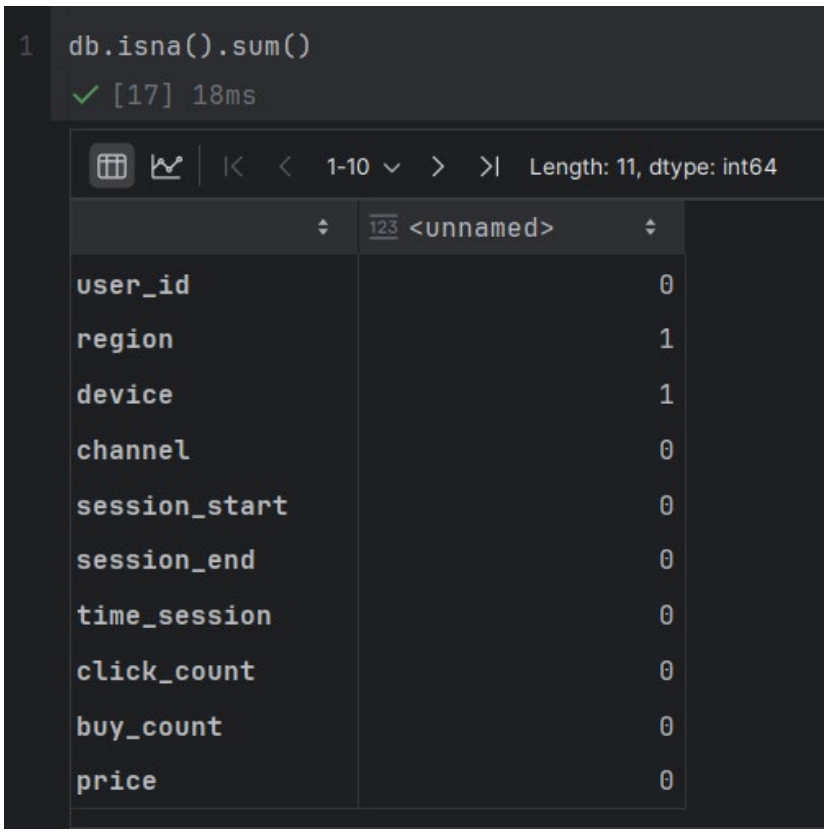


Рисунок 5 – Проверка на пропуски

Наличие пропусков подтверждается выводом этой функции. Имеются пропуски в region и device. Удалим строки с пропусками. См. рис. 6.

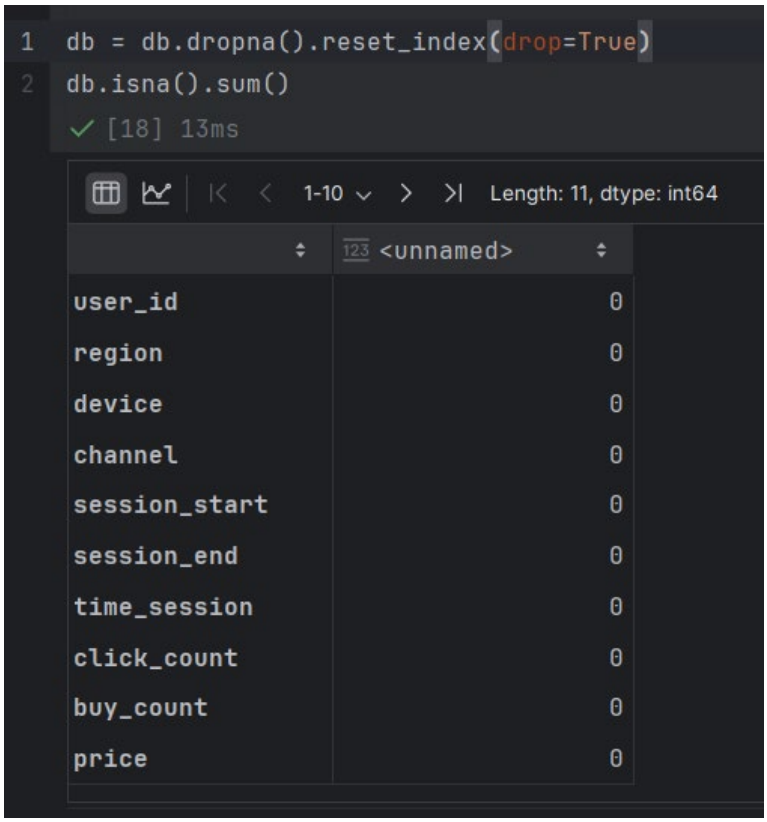


Рисунок 6 – Удаление пропусков

Таким образом пропуски были удалены, теперь оценим количество явных дубликатов. См. рис. 7.

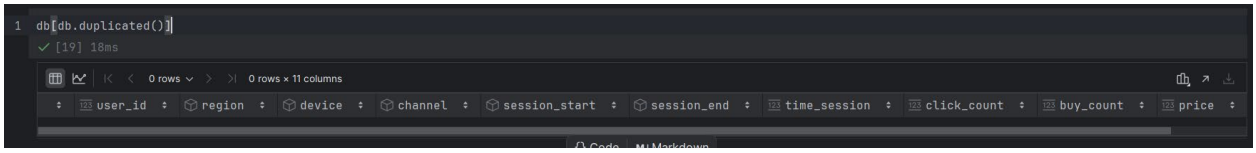


Рисунок 7 – Проверка на дубликаты

Тест на дубликаты показал отрицательный результат. Это означает, что явные дубликаты в данных отсутствуют. Теперь проверим неявные дубликаты и проверим, есть ли среди значений столбца "price", "click_count" и "buy_count" значения типа float64. См. рис. 8.

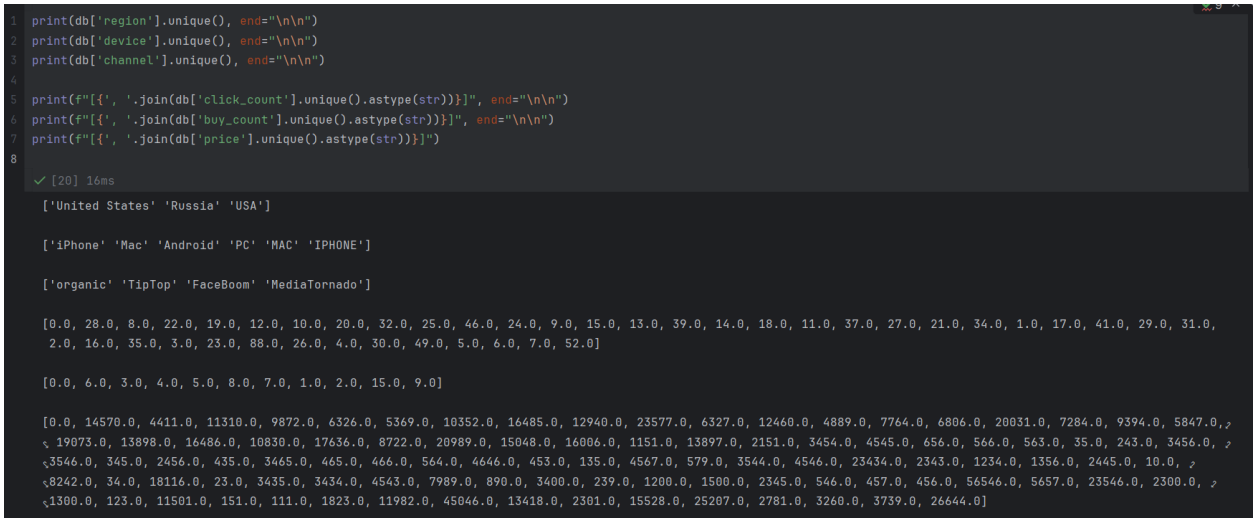


Рисунок 8 – Проверка значений в вышеупомянутых столбцах

Страны и устройства имеют дубликаты.

А количество кликов и покупок не имеет ни одного значения float64, на самом деле значения являются значениями int64. Устраним неявные дубликаты путём переименования. См. рис. 9.

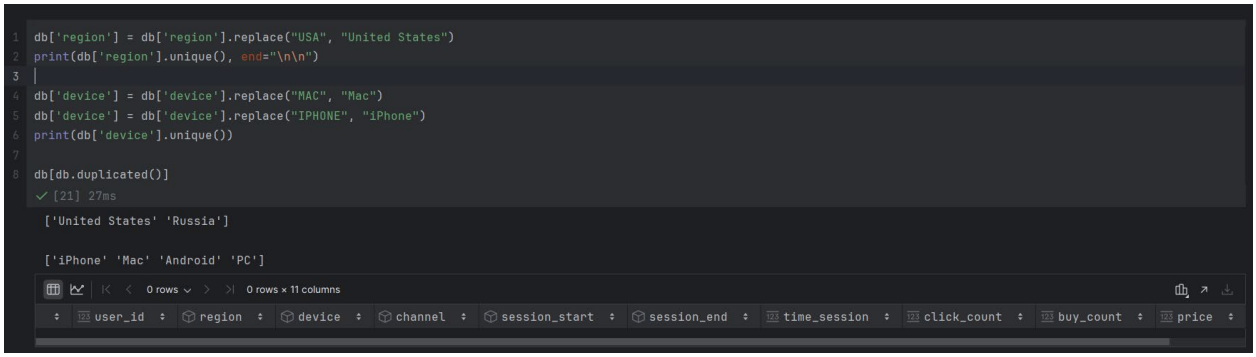
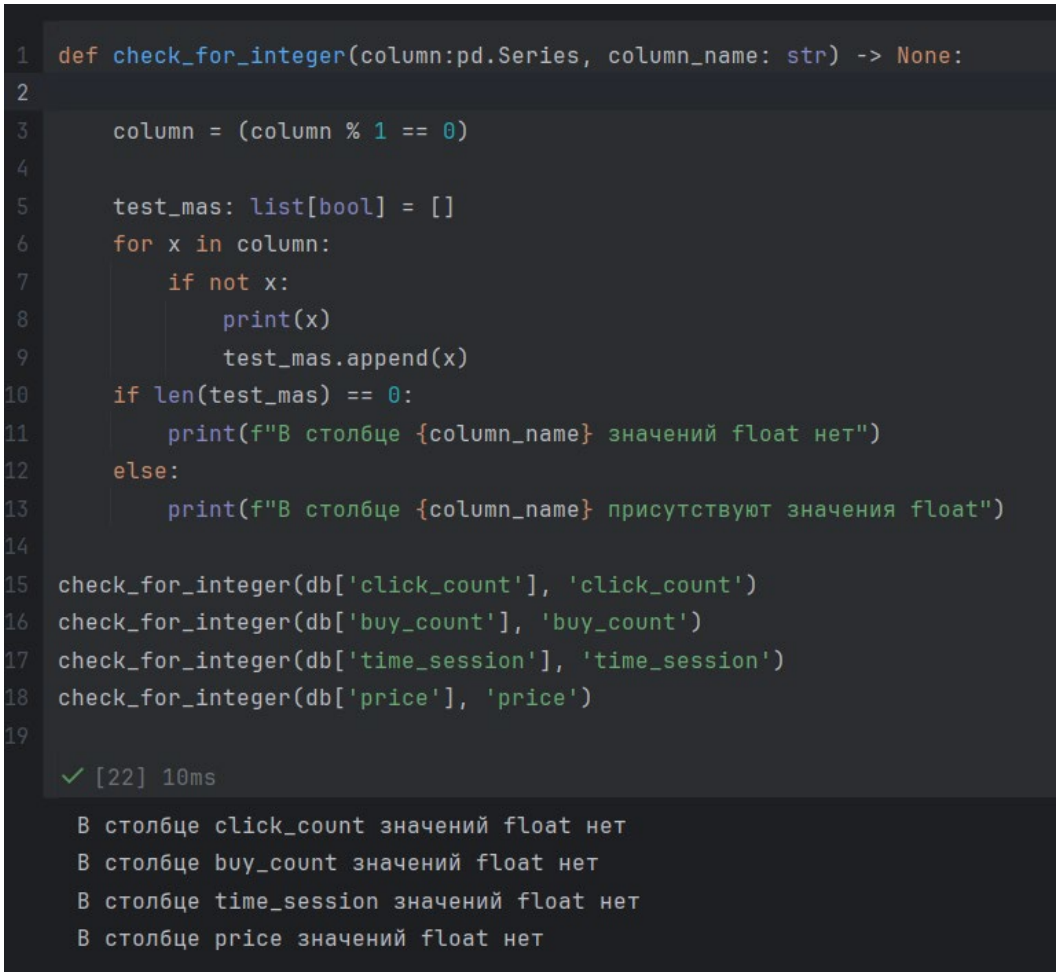


Рисунок 9 – Удаление неявных дубликатов

Неявные дубликаты также были удалены, при этом не было допущено образование новых явных дубликатов. Перед тем, как приступить к изменению типов данных в столбцах была написана функция, которая проверяет, все ли значения "price", "time_session", "buy_count" и "click_count" являются значениями int на самом деле. См. рис. 10.



```
def check_for_integer(column:pd.Series, column_name: str) -> None:

    column = (column % 1 == 0)

    test_mas: list[bool] = []
    for x in column:
        if not x:
            print(x)
            test_mas.append(x)
    if len(test_mas) == 0:
        print(f"В столбце {column_name} значений float нет")
    else:
        print(f"В столбце {column_name} присутствуют значения float")

check_for_integer(db['click_count'], 'click_count')
check_for_integer(db['buy_count'], 'buy_count')
check_for_integer(db['time_session'], 'time_session')
check_for_integer(db['price'], 'price')

✓ [22] 10ms

В столбце click_count значений float нет
В столбце buy_count значений float нет
В столбце time_session значений float нет
В столбце price значений float нет
```

Рисунок 10 – Проверка значения float64 или int64

Была произведена перепроверка типов значений в вышеупомянутых столбцах. Все значения в столбцах на самом деле являются значениями типа int. Приступим к изменению типов данных. См. рис 11.

1. session_start object -> datetime64
2. session_end object -> datetime64
3. click_count float64 -> int64
4. buy_count float64 -> int64
5. time_session float64 -> int64
6. price float64 -> int64

```
columns = ['session_start', 'session_end']
for column in columns:
    db[column] = pd.to_datetime(db[column], format='%d.%m.%Y %H:%M')

db['time_session'] = db['time_session'].astype('int64')
db['click_count'] = db['click_count'].astype('int64')
db['buy_count'] = db['buy_count'].astype('int64')
db['price'] = db['price'].astype('int64')
db.dtypes
```

✓ [23] 22ms

1-10 Length: 11, dtype: object

	<unnamed>
user_id	int64
region	object
device	object
channel	object
session_start	datetime64[ns]
session_end	datetime64[ns]
time_session	int64
click_count	int64
buy_count	int64
price	int64

Рисунок 11 – Изменение типа данных в открытом датафрейме

Все предварительные этапы подготовки данных были выполнены. Теперь приступим основному заданию. Будет произведено построение графика типа scatter или диаграммы рассеяния. См. рис 12, 13 и 14.

```
db.plot(x='click_count', y='price', kind='scatter', color='red', label='Цена от кликов')
db.plot(x='time_session', y='price', kind='scatter', color='orange', label='Цена от времени')
db.plot(x='user_id ', y='buy_count', kind='scatter', color='green', label='Цена от ID пользователя')
```

✓ [24] 672ms

<Axes: xlabel='user_id ', ylabel='buy_count'>

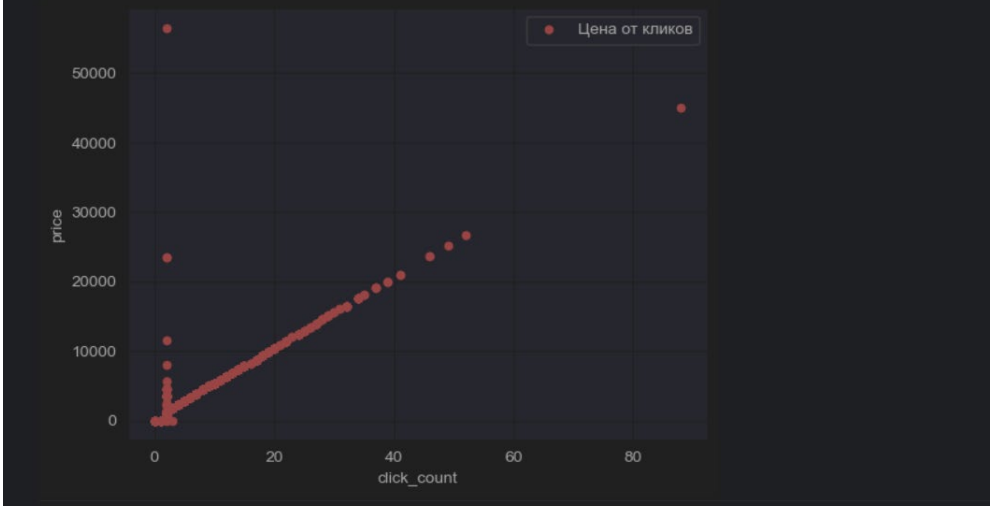


Рисунок 12 – Первый построенный график

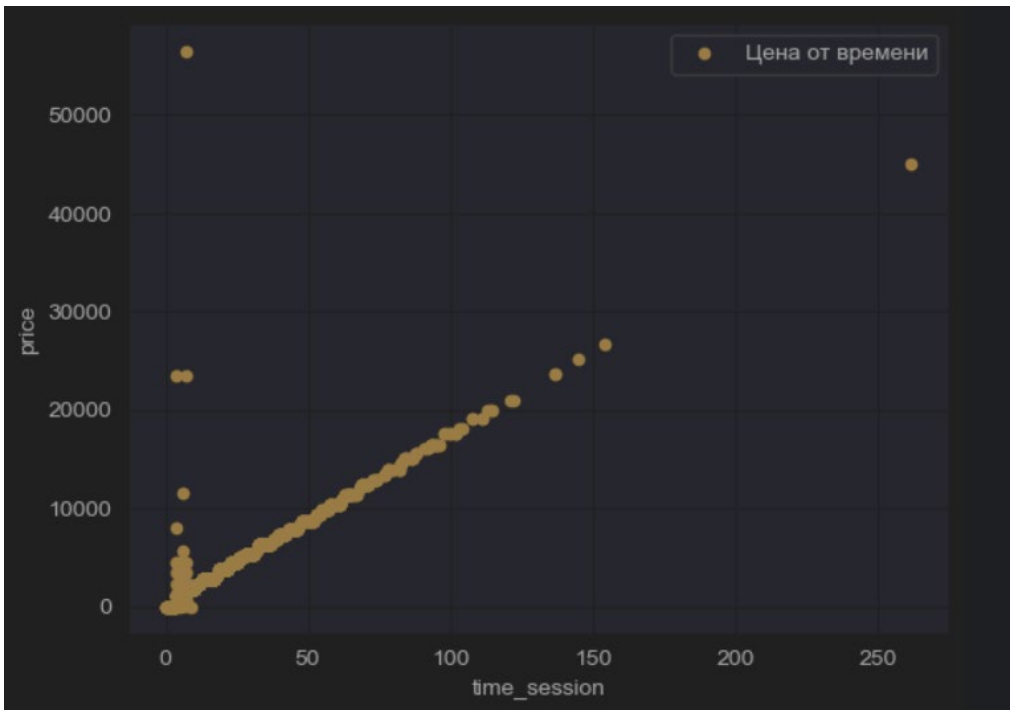


Рисунок 13 – Второй построенный график

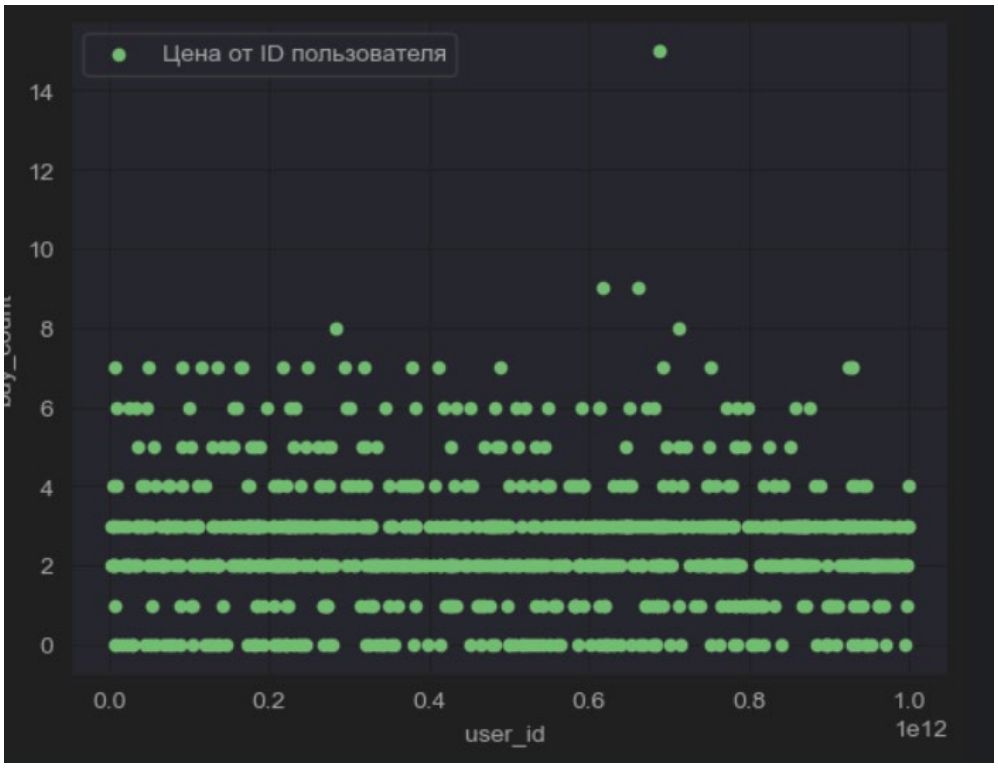


Рисунок 14 – Третий построенный график

На первой диаграмме была представлена зависимость цены купленных товаров от количества кликов пользователя. По полученной визуализации заметна выраженная зависимость: чем больше кликов сделал пользователь, тем больше цена купленных товаров. Также по третьей диаграмме можно сделать вывод, что обычно покупают 2-3 позиции, также по этому графику

можно сделать вывод, больше 9 позиций обычно не берут. Попробуем воспользоваться командой для построения всех возможных графиков. См. рис. 15 и 16.

```
1  pd.plotting.scatter_matrix(db, figsize=(7, 7))
   ✓ [34] 2s 289ms

   A:\applications\MiniConda\Lib\site-packages\pandas\plotting\_matplotlib\misc.py:121: RuntimeWarning: invalid value encountered in cast
     if np.all(locs == locs.astype(int)):
   array([[<Axes: xlabel='user_id ', ylabel='user_id '>,
           <Axes: xlabel='time_session', ylabel='user_id '>,
           <Axes: xlabel='click_count', ylabel='user_id '>,
           <Axes: xlabel='buy_count', ylabel='user_id '>,
           <Axes: xlabel='price', ylabel='user_id '>,
           <Axes: xlabel='ag
```

Рисунок 15 – Создание диаграмм

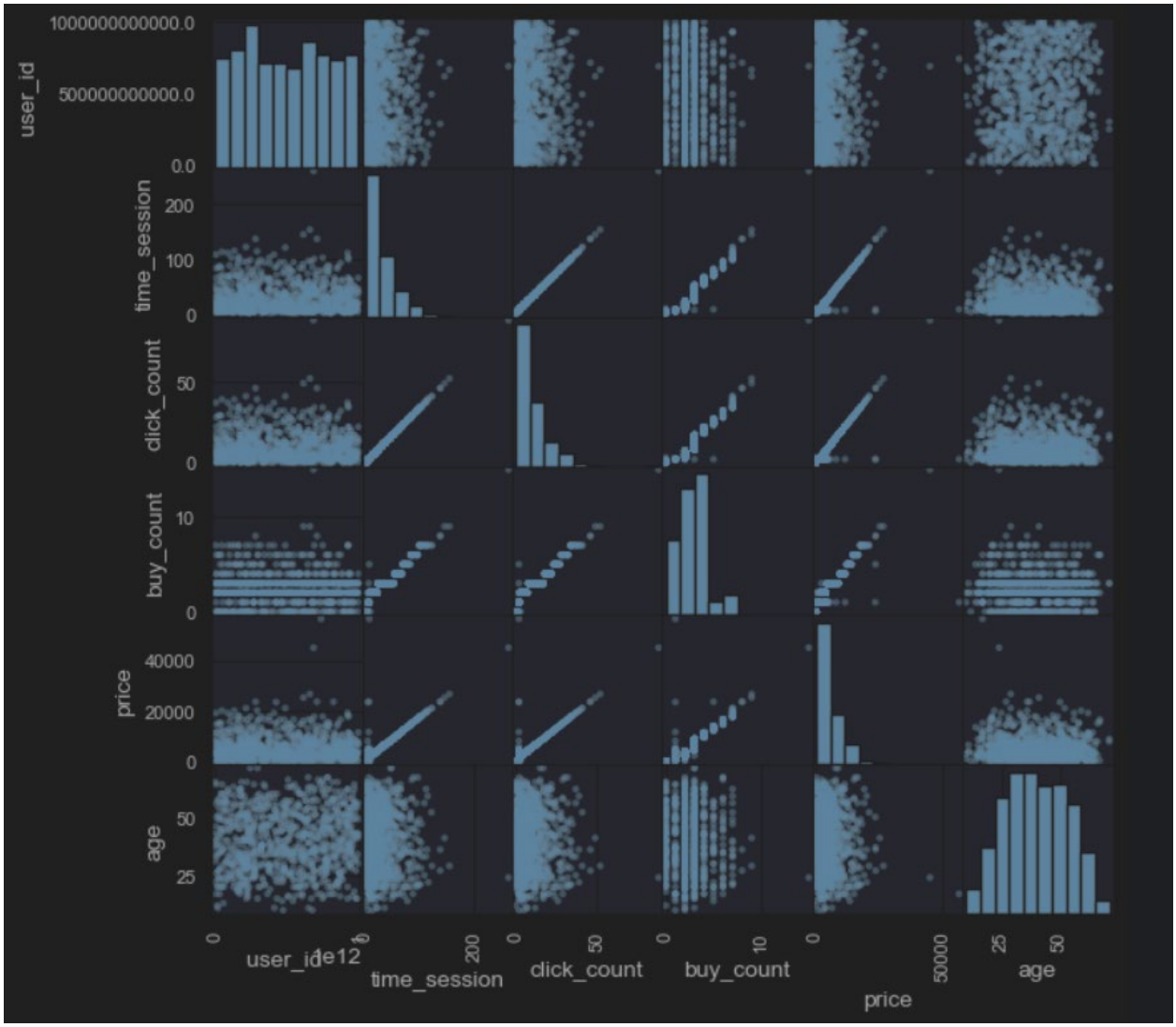


Рисунок 16 – Матрица диаграмм рассеяния

С помощью данной диаграммы можно определить также и другие зависимости, которые имеют разный характер. Например, количество кликов имеет сильную зависимость с количеством товаров, а id пользователя не зависит от его возраста и наоборот. Решотчатость графиков объясняется

малым количеством вариантов ответа в столбцах click_count и buy_count. По графикам видно, что зависимости параметров от id пользователя отсутствует, что означает, что данные адекватны. Выведем на экран таблицу корреляции. См. рис. 17.

```
print("Корреляция")
db.select_dtypes(include=[int, float, 'datetime64']).corr().round(2)
```

✓ [26] 23ms

Корреляция

8 rows × 8 columns

	user_id	session_start	session_end	time_session	click_count	buy_count	price	age
user_id	1.00	0.01	0.01	-0.06	-0.06	-0.05	-0.04	0.06
session_start	0.01	1.00	1.00	0.01	0.01	0.04	0.03	0.01
session_end	0.01	1.00	1.00	0.04	0.04	0.06	0.05	0.01
time_session	-0.06	0.01	0.04	1.00	1.00	0.95	0.90	-0.04
click_count	-0.06	0.01	0.04	1.00	1.00	0.95	0.91	-0.04
buy_count	-0.05	0.04	0.06	0.95	0.95	1.00	0.87	-0.04
price	-0.04	0.03	0.05	0.90	0.91	0.87	1.00	-0.05
age	0.06	0.01	0.01	-0.04	-0.04	-0.04	-0.05	1.00

Рисунок 17 – Коэффициент корреляции

Коэффициент корреляции показывает, насколько данные взаимосвязаны. Если он ближе к 1, то при увеличении величины А, величина Б тоже растёт, если ближе к -1, то при увеличении величина А, величина Б уменьшается, если коэффициент ближе к 0, то корреляция отсутствует, также он отображает степень зависимости величины А от величины Б. Далее создадим матрицу корреляции. См. рис. 18, 19.

```
# Вычисление матрицы корреляции
correlation_matrix = db.select_dtypes(include=[int, float, 'datetime64']).corr().round(2)

# Построение тепловой карты
plt.figure(figsize=(8, 6))
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='nearest')

# Добавление цветовой шкалы
plt.colorbar()

# Добавление меток осей
plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation=45)
plt.yticks(range(len(correlation_matrix.index)), correlation_matrix.index)

for (i, j), val in np.ndenumerate(correlation_matrix):
    plt.text(j, i, f'{val:.2f}', ha='center', va='center', color='white' if val < 0 else 'black')

# Добавление заголовка
plt.title('Корреляционная матрица')

# Показать тепловую карту
plt.tight_layout()
plt.show()

✓ [27] 327ms
```

Рисунок 18 – Создание матрицы корреляции

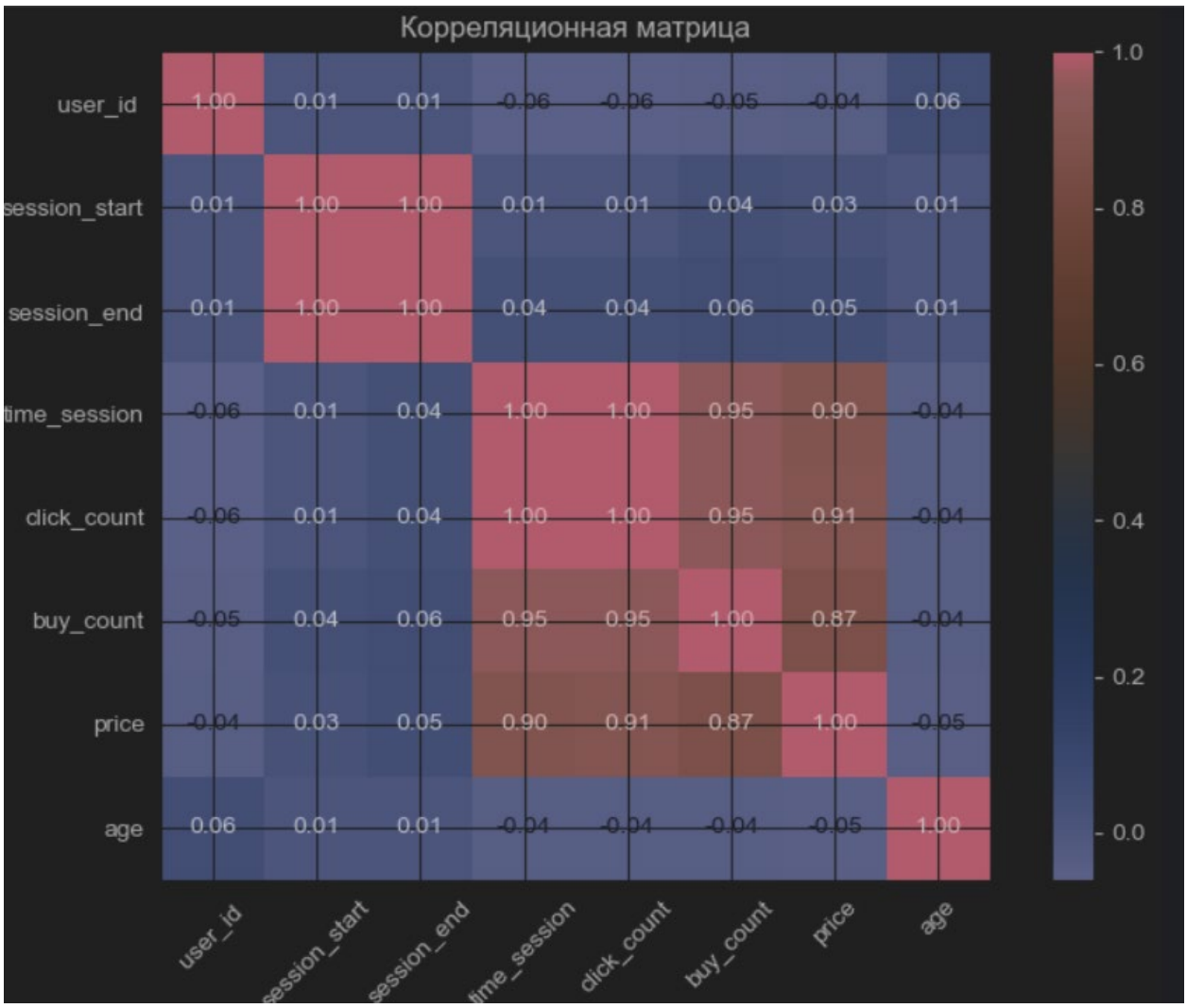


Рисунок 19 – Матрица корреляции

Данная матрица, изображённая на тепловой карте, значительно упрощает просмотр данных, представляет собой таблицу с раскрашенными ячейками. Чем краснее ячейка, тем выше корреляция, если корреляция положительная, то значения белые, иначе - чёрные. Приступим к построению графиков из варианта.

Выполним задание 1. См. рис. 20, 21.

```python
# Группировка данных по каналу и региону
group = db.groupby(['channel', 'region'], as_index=False)['user_id '].count()

plt.figure(figsize=(10, 6))
sns.barplot(data=group, x='channel', y='user_id ', hue='region')

# Добавление заголовка и меток осей
plt.title('Диаграмма количества клиентов по каждому региону и каналу привлечения')
plt.xlabel('Канал привлечения')
plt.ylabel('Количество клиентов')

# Показать диаграмму
plt.legend(title='Регион')
plt.tight_layout()
plt.show()
```

Рисунок 20 – Создание диаграммы по заданию 1

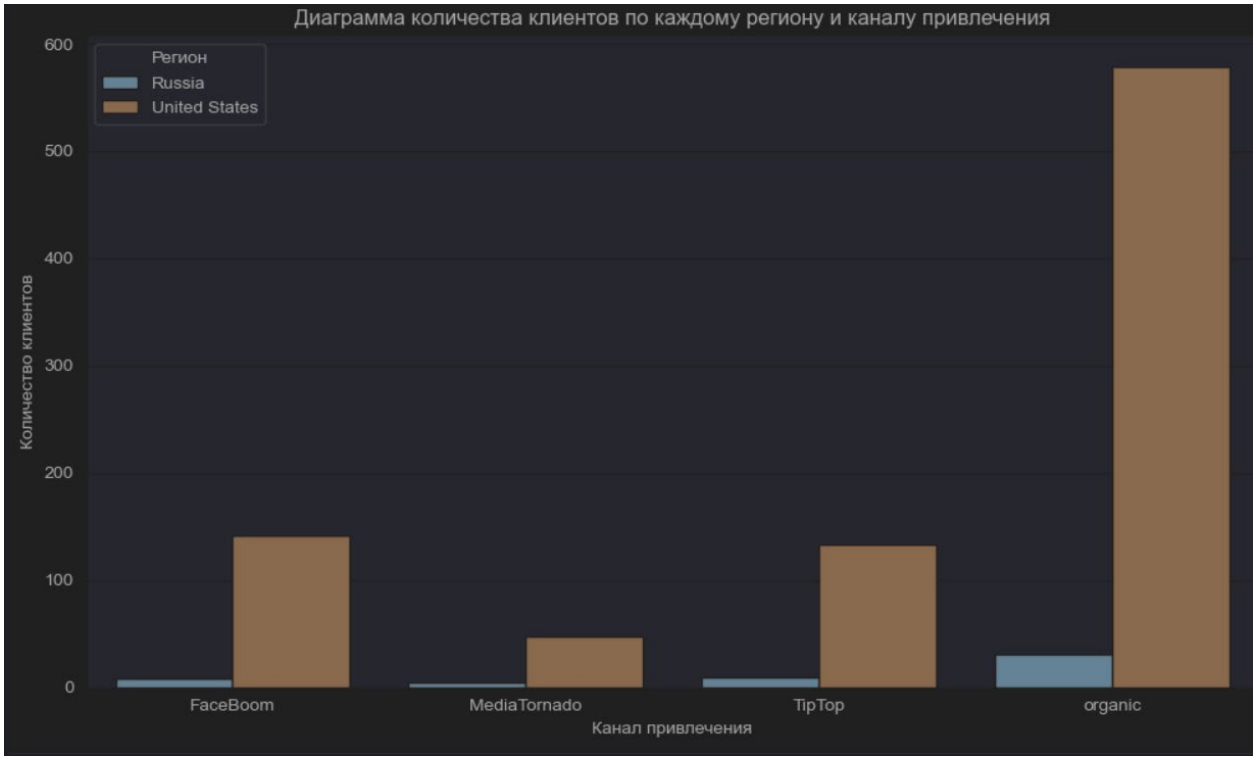


Рисунок 21 – Диаграмма по заданию 1

Данный график показывает наглядное соотношение людей, пришедших из разных источников по странам. Значительно больший трафик идёт из США, а наиболее распространённым источником как в США, так и в России является organic.

Выполним задание 2. См. рис. 22.

```
pivot_table1 = db.pivot_table(index='channel', values='user_id ', aggfunc='nunique').reset_index()
pivot_table1.plot(x='channel', y='user_id ', kind='scatter', label="График", s=15, marker='<', color='blue')
✓ [29] 269ms
<Axes: xlabel='channel', ylabel='user_id '>
```

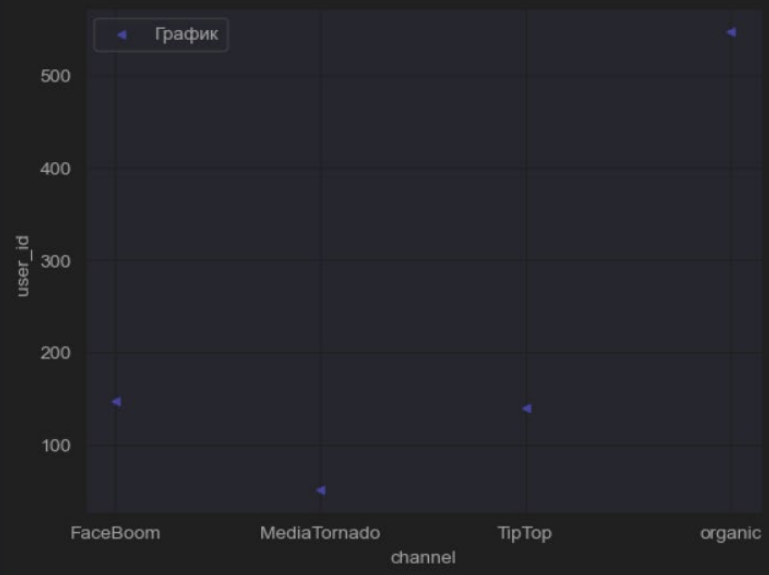


Рисунок 22 – Создание и график задания 2

Данный график показывает, насколько много пользователей в целом пришло из каждого источника. Большинство пользователей пришло из источника organic, меньшинство из MediaTornado. Из источников FaceBoom и TipTop пришло примерно равное количество пользователей.

Выполним задание 3. См. рис. 23, 24.

```
# Группировка данных и подсчет количества уникальных пользователей по каждому каналу
pivot_table2 = db.pivot_table(index='device', values='user_id ', aggfunc='nunique').reset_index()

# Вычисление процента для каждой группы
pivot_table2['percentage'] = pivot_table2['user_id '] / pivot_table2['user_id '].sum() * 100

# Округляем проценты для лучшей читаемости
pivot_table2['percentage'] = pivot_table2['percentage'].round(2)

plt.figure(figsize=(10, 6))
plt.pie(x=pivot_table2['percentage'], labels=pivot_table2['device'], autopct='%1.1f%%')
# Добавление заголовка
plt.title('Распределение устройств')

# Отображение диаграммы
plt.show()
✓ [30] 123ms
```

Рисунок 23 – Создание диаграммы задания 3

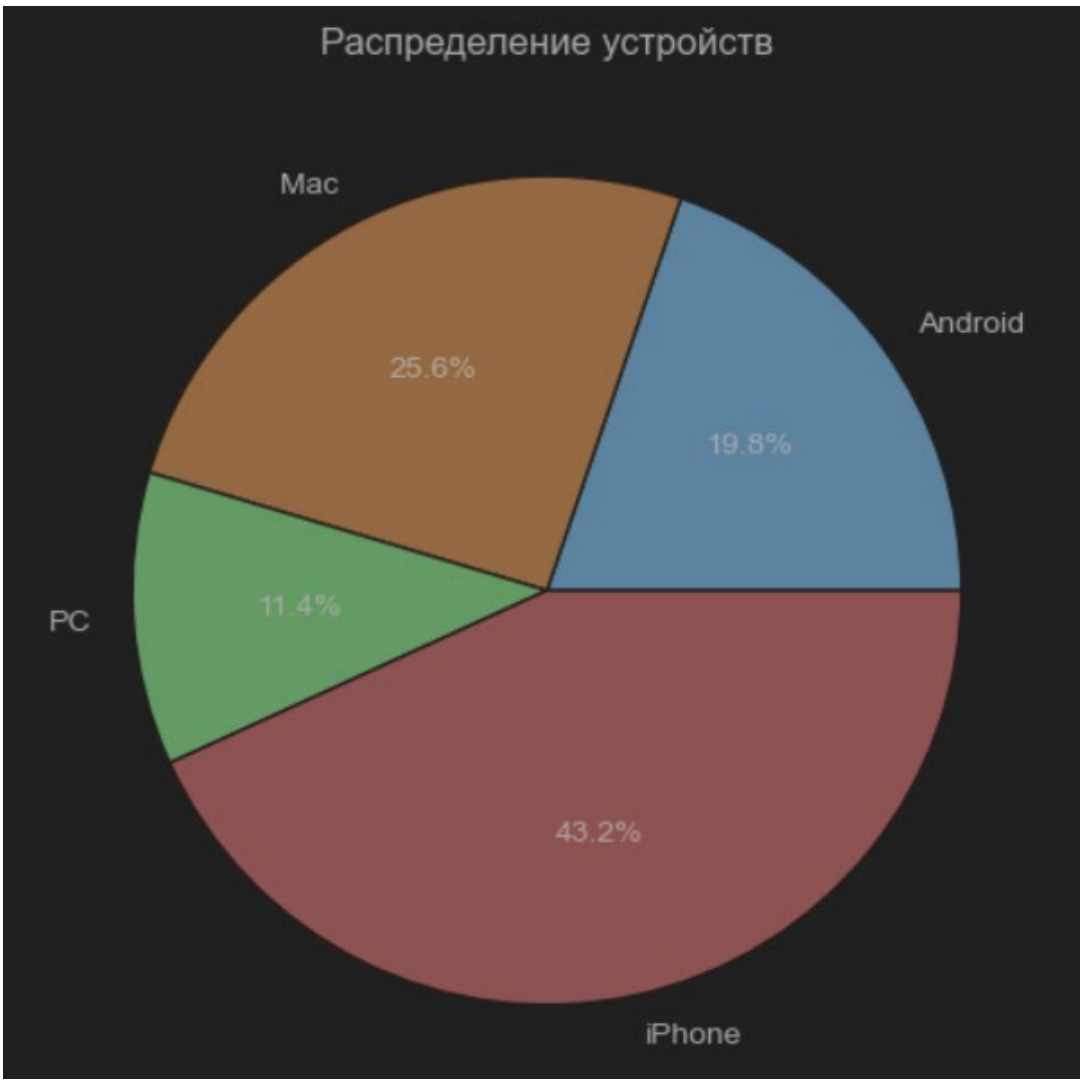


Рисунок 24 – Диаграмма задания 3

Данный график показывает процентное соотношение устройств, с которых выполняли подключение пользователи. Например, iPhone составил основу - 43.2% пользователей смогли подключиться благодаря ему.

5. Ссылка на Google Colab:

https://colab.research.google.com/drive/1CEWCUD_6xYkqBDXDt5RdYSi464Y2wbyH?usp=sharing

6. Вывод:

В ходе работы был выполнен полный цикл анализа данных, начиная с предварительной обработки и заканчивая построением различных визуализаций для анализа взаимосвязей между признаками.

1. Предварительная обработка данных:

Были обнаружены и удалены пропуски в столбцах region и device.

Проверка на явные и неявные дубликаты показала их отсутствие после обработки.

Столбцы с типом данных были приведены к корректным форматам, что позволило провести дальнейший анализ корректно.

2. Анализ зависимостей:

Построение диаграмм рассеяния выявило следующие закономерности:

Зависимость между количеством кликов и ценой покупок: чем больше кликов, тем выше сумма покупок.

Определена связь между количеством кликов и количеством товаров в корзине.

Изучение корреляции между переменными с использованием тепловой карты показало наличие значимых зависимостей между признаками, такими как количество кликов, товаров в корзине и стоимость покупок.

3. Задание по варианту:

Диаграмма распределения пользователей по регионам и каналам показала, что в США наибольшее количество пользователей приходит из источника organic, а в России также наблюдается высокая активность из этого источника.

Круговая диаграмма продемонстрировала, что наиболее часто используемым устройством для доступа к магазину является iPhone, на который приходится 43,2% всех пользователей.

Таким образом, проведенный анализ позволил выявить важные закономерности в поведении пользователей магазина. Большое значение для привлечения пользователей имеет канал organic, а наиболее часто используемое устройство — это iPhone.