

Описание проекта

Вы работаете в стартапе, который продаёт продукты питания. Нужно разобраться, как ведут себя пользователи вашего мобильного приложения.

Описание данных

Структура датасета logs_exp.csv:

- EventName — название события;
- DeviceIDHash — уникальный идентификатор пользователя;
- EventTimestamp — время события;
- ExpId — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

Каждая запись в логе — это действие пользователя, или событие.

Цель исследования

- изучить воронку событий
- проанализировать результаты теста и принять решение по введению новых шрифтов

План работы

Шаг 1. Открыть файл с данными и изучить общую информацию

Шаг 2. Подготовить данные

- Заменить названия столбцов на удобные для вас;
- Проверить пропуски и типы данных. Откорректировать, если нужно;
- Добавить столбец даты и времени, а также отдельный столбец дат;

Шаг 3. Изучить и проверить данные

- Сколько всего событий в логе?
- Сколько всего пользователей в логе?
- Сколько в среднем событий приходится на пользователя?
- Данными за какой период имеются? Найти максимальную и минимальную дату. Построить гистограмму по дате и времени. Можно ли быть уверенным, что у нас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать»

события из прошлого — это может «перекаивать данные». Определить, с какого момента данные полные и отбросьте более старые. Данными за какой период времени располагаем на самом деле?

- Много ли событий и пользователей потеряли, отбросив старые данные?
- Проверить, что есть пользователи из всех трёх экспериментальных групп.

Шаг 4. Изучить воронку событий

- Посмотреть, какие события есть в логах, как часто они встречаются. Отсортировать события по частоте.
- Посчитать, сколько пользователей совершали каждое из этих событий. Отсортировать события по числу пользователей. Посчитать долю пользователей, которые хоть раз совершали событие.
- Предположить, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.
- По воронке событий посчитать, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитать отношение числа пользователей с событием B к количеству пользователей с событием A, а также отношение числа пользователей с событием C к количеству пользователей с событием B.
- На каком шаге теряется больше всего пользователей?
- Какая доля пользователей доходит от первого события до оплаты?

Шаг 5. Изучить результаты эксперимента

- Сколько пользователей в каждой экспериментальной группе?
- Есть 2 контрольные группы для А/А-эксперимента, чтобы проверить корректность всех механизмов и расчётов. Проверить, находят ли статистические критерии разницу между выборками 246 и 247.
- Выбрать самое популярное событие. Посчитать число пользователей, совершивших это событие в каждой из контрольных групп. Посчитать долю пользователей, совершивших это событие. Проверить, будет ли отличие между группами статистически достоверным. Проделать то же самое для всех других событий (удобно обернуть проверку в отдельную функцию). Можно ли сказать, что разбиение на группы работает корректно?
- Аналогично поступить с группой с изменённым шрифтом. Сравнить результаты с каждой из контрольных групп в отдельности по каждому событию. Сравнить результаты с объединённой контрольной группой. Какие выводы из эксперимента можно сделать?
- Какой уровень значимости выбран при проверке статистических гипотез выше? Посчитать, сколько проверок статистических гипотез сделано. При уровне значимости 0.1 каждый десятый раз можно получать ложный результат. Какой уровень значимости стоит применить? Если хотим изменить его, проделать предыдущие пункты и проверить свои выводы.

Некоторые графики с использованием библиотеки `plotly` могут не отобразиться со старыми версиями из-за параметра `text_auto=True` (отображение значений на графике). В следующей ячейке добавила закомментированную установку более поздней версии.

```
In [1]: #!pip install plotly==5.9.0
```

1. Загрузка данных

Импортируем необходимые библиотеки и загружаем данные из файлов в датафрейм:

```
In [2]: import pandas as pd
import datetime as dt

import numpy as np
from scipy import stats as st
import math as mth

import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from plotly import graph_objects as go
import warnings

warnings.filterwarnings("ignore")
```

```
In [3]: try:
    logs = pd.read_csv('~/Desktop/project_2/data/logs_exp.csv', sep='\t')
except:
    logs = pd.read_csv('https://code.s3.yandex.net/datasets/logs_exp.csv', sep='\t')
```

Рассмотрим полученные данные из датафрейма:

```
In [4]: # Выведем 5 строк датафреймов
display(logs.head(10).style.set_caption('Логи'))
```

Логи

	EventName	DeviceIDHash	EventTimestamp	ExpId
0	MainScreenAppear	4575588528974610432	1564029816	246
1	MainScreenAppear	7416695313311560704	1564053102	246
2	PaymentScreenSuccessful	3518123091307005440	1564054127	248
3	CartScreenAppear	3518123091307005440	1564054127	248
4	PaymentScreenSuccessful	6217807653094995968	1564055322	248
5	CartScreenAppear	6217807653094995968	1564055323	248
6	OffersScreenAppear	8351860793733344256	1564066242	246
7	MainScreenAppear	5682100281902513152	1564085677	246
8	MainScreenAppear	1850981295691852800	1564086702	247
9	MainScreenAppear	5407636962369102848	1564112112	246

```
In [5]: # Выведем основные характеристики датафрейма (типы столбцов, пропущенные значения)
print('Логи')
logs.info()
```

Логи

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   EventName        244126 non-null   object 
 1   DeviceIDHash     244126 non-null   int64  
 2   EventTimestamp   244126 non-null   int64  
 3   ExpId            244126 non-null   int64  
dtypes: int64(3), object(1)
memory usage: 7.5+ MB
```

```
In [6]: # Проверим, присутствуют ли пропуски в датафреймах:  
print(f'Количество пропусков в датафрейме с логами - {logs.isna().sum()}' )
```

```
Количество пропусков в датафрейме с логами - EventName          0  
DeviceIDHash      0  
EventTimestamp    0  
ExpId            0  
dtype: int64
```

Вывод: В датафрейме `logs` пропусков нет. Всего в датафрейме 244126 строк. Отметим, что столбец даты относится к числовому типу данных. Всего столбцов четыре:

- `EventName` — название события,
- `DeviceIDHash` — уникальный идентификатор пользователя,
- `EventTimestamp` — время события,
- `ExpId` — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

2. Подготовка данных

```
In [7]: # Переименуем столбцы датафрейма Logs:  
logs = logs.rename(columns={  
    'EventName': 'event_name',  
    'DeviceIDHash': 'device_id_hash',  
    'EventTimestamp': 'event_timestamp',  
    'ExpId': 'exp_id'  
})
```

```
In [8]: # Проверим, присутствуют ли дубликаты в датафрейме:  
print(f'Всего дупликатов в датафрейме - {logs.duplicated().sum()}, '  
     f'то есть {round(100 * logs.duplicated().sum() / len(logs), 2)} процентов от всех данных')
```

Всего дупликатов в датафрейме - 413, то есть 0.17 процентов от всех данных

Удалим дубликаты:

```
In [9]: logs = logs.drop_duplicates()
```

```
In [10]: # добавим новый столбец даты, используя столбец EventTimestamp  
logs['full_date'] = pd.to_datetime(logs['event_timestamp'], unit='s')
```

```
In [11]: # добавим новый столбец даты без времени, используя ранее созданный столбец full_date  
logs['date'] = pd.to_datetime(logs['full_date'].dt.strftime('%Y-%m-%d'))
```

Вывод:

- привели наименования столбцов к нижнему регистру и переименовали в соответствии с синтаксисом
- доля удаленных данных 0,17% (удалены дубликаты)
- добавлены два столбца типа datetime - полная дата и дата без времени

3. Изучение и проверка данных

- Сколько всего событий в логе?

```
In [12]: print(f'Всего событий в логе - {logs["event_name"].count()}')
```

Всего событий в логе - 243713

Рассмотрим, какие именно события имеются в логах:

```
In [13]: print('Уникальные значения в столбце событий - ', ", ".join(map(str, logs['event_name'].unique())), sep='\n')
```

Уникальные значения в столбце событий -
MainScreenAppear, PaymentScreenSuccessful, CartScreenAppear, OffersScreenAppear, Tutorial

In [14]: # проверим, как распределены события:

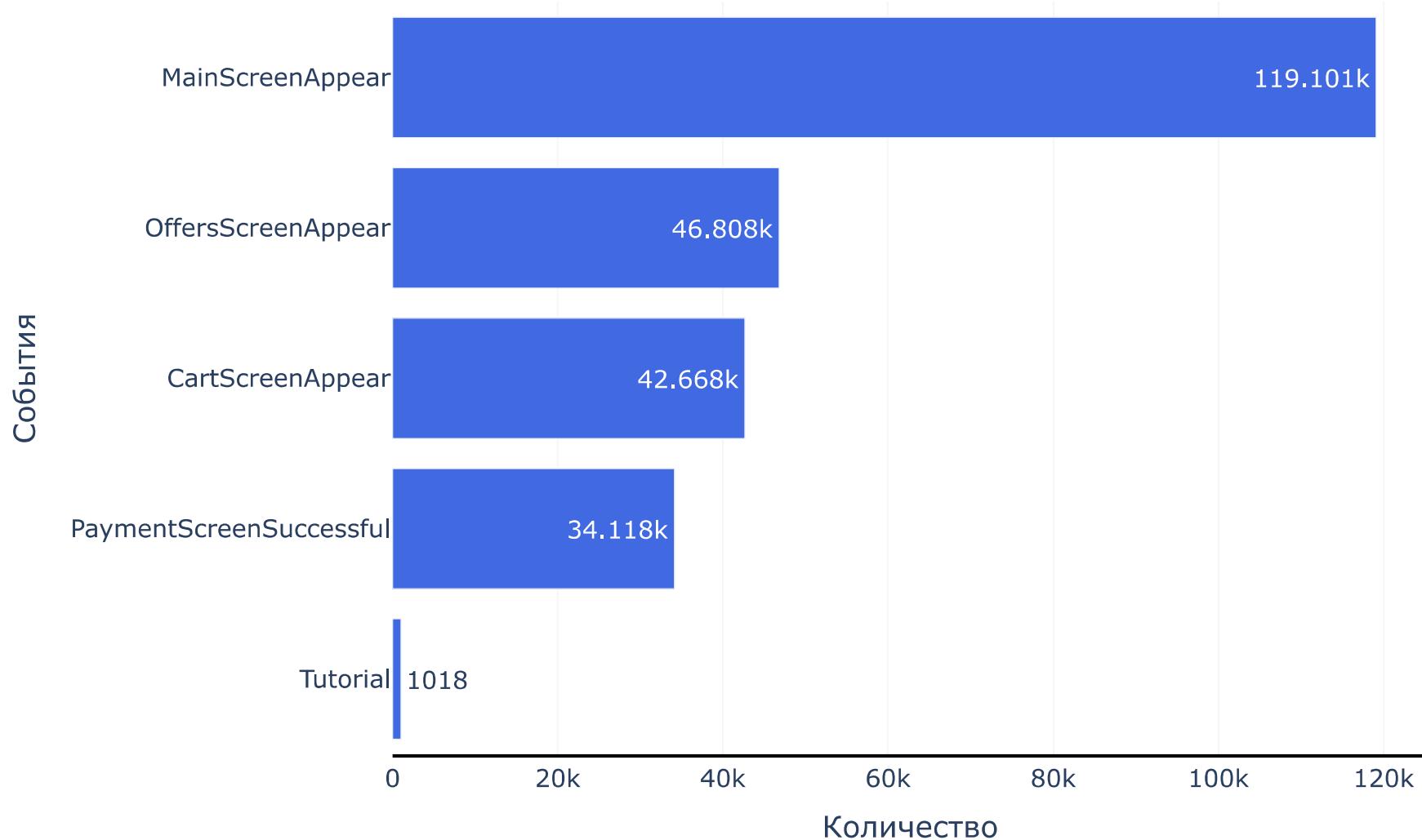
```
logs.pivot_table(index='event_name', values='event_timestamp', aggfunc='count').sort_values(by='event_timestamp')
```

Out[14]:

event_name	event_timestamp
Tutorial	1018
PaymentScreenSuccessful	34118
CartScreenAppear	42668
OffersScreenAppear	46808
MainScreenAppear	119101

```
In [15]: px.bar(logs.pivot_table(index='event_name', values='event_timestamp', aggfunc='count').sort_values(by='event_timestamp'),
    x='event_timestamp',
    color_discrete_sequence=["royalblue"],
    text_auto=True) \
.update_layout(plot_bgcolor='rgba(0,0,0,0)') \
.update_xaxes(showline=True,
    linewidth=2,
    linecolor='black',
    gridcolor='LightGrey') \
.update_layout(plot_bgcolor='rgba(0,0,0,0)',
    autosize=False,
    width=950,
    height=600,
    title = {
        'text': "Распределение событий",
        'y':0.98,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    },
    xaxis_title="Количество",
    yaxis_title="События",
    font=dict(
        size=15),
    )
```

Распределение событий



Чаще всего происходит событие `MainScreenAppear` (показ главного экрана) - 119 101, в 2,5 раза меньше происходило событие `OffersScreenAppear` (появление каталога/предложенные товары на экране) - 46 808, немногим меньше у события `CartScreenAppear` (отображение корзины) - 42 668. Событие `PaymentScreenSuccessful` - успешная оплата - произошло 34 118 раз. Меньше всего пользователи проходили `Tutorial` (обучение) - всего 1 018 раз.

- Сколько всего пользователей в логе?

```
In [16]: print(f'Всего уникальных пользователей в логе - {logs["device_id_hash"].nunique()}')
```

Всего уникальных пользователей в логе - 7551

Посмотрим, сколько уникальных пользователей находятся в каждой из групп:

Всего представлено три группы: 246 (контрольная), 247 (контрольная), 248 (экспериментальная). Найдем количество уникальных пользователей в каждой группе:

```
In [17]: print('Уникальное число пользователей в группе 246 -',
      logs.query('exp_id==246').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Уникальное число пользователей в группе 247 -',
      logs.query('exp_id==247').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Уникальное число пользователей в группе 248 -',
      logs.query('exp_id==248').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Всего уникальных пользователей - ', logs.agg({'device_id_hash':'nunique'}).device_id_hash)
```

Уникальное число пользователей в группе 246 - 2489

Уникальное число пользователей в группе 247 - 2520

Уникальное число пользователей в группе 248 - 2542

Всего уникальных пользователей - 7551

```
In [18]: print('Количество попаданий одновременно в группу 246 и 247:',
      pd.merge(logs.query('exp_id==246').device_id_hash, logs.query('exp_id==247').device_id_hash).device_id_hash.nunique())
print('Количество попаданий одновременно в группу 247 и 248:',
      pd.merge(logs.query('exp_id==247').device_id_hash, logs.query('exp_id==248').device_id_hash).device_id_hash.nunique())
print('Количество попаданий одновременно в группу 246 и 248:',
      pd.merge(logs.query('exp_id==246').device_id_hash, logs.query('exp_id==248').device_id_hash).device_id_hash.nunique())
```

Количество попаданий одновременно в группу 246 и 247: 0

Количество попаданий одновременно в группу 247 и 248: 0

Количество попаданий одновременно в группу 246 и 248: 0

Каждый пользователь находится только в своей группе.

```
In [19]: # посчитаем долю уникальных пользователей по группам тестирования
group = logs.pivot_table(index='exp_id', values='device_id_hash', aggfunc='nunique').reset_index()
group['share'] = round(100 * group['device_id_hash'] / sum(group['device_id_hash']), 2)
```

In [20]: group

Out[20]:

	exp_id	device_id_hash	share
0	246	2489	32.96
1	247	2520	33.37
2	248	2542	33.66

Доля уникальных пользователей в экспериментальной группе чуть выше остальных.

- Сколько в среднем событий приходится на пользователя?

In [21]: *# посчитаем, сколько событий приходится на каждого пользователя и выдедем основную информацию*
logs.pivot_table(index='device_id_hash', values='event_name', aggfunc='count').describe()

Out[21]:

	event_name
count	7551.000000
mean	32.275593
std	65.154219
min	1.000000
25%	9.000000
50%	20.000000
75%	37.000000
max	2307.000000

В среднем на пользователя приходится около 32 событий. Минимальное и максимальное количество событий на пользователя - 1 и 2307 соответственно. Получился сильный разброс значений. Есть пользователи, которые совершили только 1 событие (возможно, только перешли на главный экран и больше не совершали никаких действий) и пользователи, которые регулярно пользуются приложением. Так как величина среднего значения не показательна при выбросах, будем ориентироваться на значение медианы, которое составляет 20 событий .

- Данными за какой период имеются? Найти максимальную и минимальную дату. Построить гистограмму по дате и времени. Можно ли быть уверенными, что у нас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать»

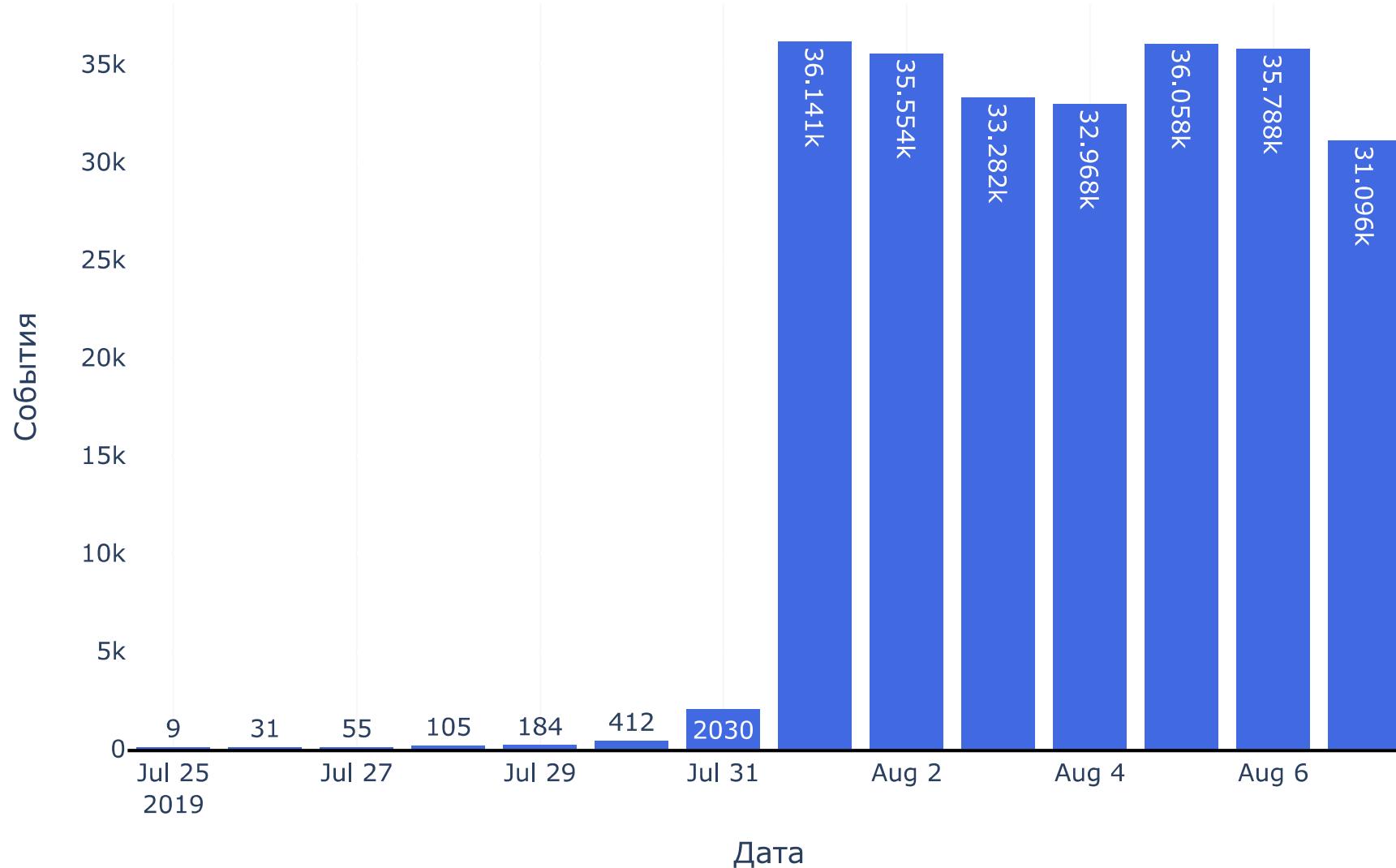
события из прошлого — это может «перекаивать данные». Определить, с какого момента данные полные и отбросьте более старые. Данными за

```
In [22]: # узнаем минимальную и максимальную даты в имеющихся данных  
print(f'Начало периода - {min(logs["date"])}), конец периода - {max(logs["date"])}')  
print(f'Всего дней - {((max(logs["date"]) - min(logs["date"])).days}')")
```

Начало периода - 2019-07-25 00:00:00, конец периода - 2019-08-07 00:00:00
Всего дней - 13

```
In [23]: fig = px.histogram(logs,
                         x="date",
                         nbins=20,
                         color_discrete_sequence=["royalblue"],
                         text_auto=True
                        )
fig.update_layout(bargap=0.2)
fig.update_layout(plot_bgcolor='rgba(0,0,0,0)',
                  autosize=False,
                  width=950,
                  height=600,
                  title = {
                      'text': "Распределение событий по датам",
                      'y':0.98,
                      'x':0.5,
                      'xanchor': 'center',
                      'yanchor': 'top'
                  },
                  xaxis_title="Дата",
                  yaxis_title="События",
                  font=dict(
                      size=15),
                  )
fig.update_xaxes(showline=True,
                  linewidth=2,
                  linecolor='black',
                  gridcolor='LightGrey')
fig.show()
```

Распределение событий по датам



Ранее выяснили, что у нас есть данные от 25 июля 2019 до 7 августа 2019. Судя по полученной гистограмме, с 25 по 31 июля количество логов увеличиваются - 25 июля было 9 логов, 31 июля - 2030 логов. Данных за июль очень мало по сравнению с данными за август. 1 августа было совершено 36 141 логов, что намного больше данных за июль. С 1 по 7 августа данные не сильно разнятся, находясь в диапазоне от 31 тысячи до 36 тысяч. Оставим наиболее полные данные с 1 августа по 7 августа.

```
In [24]: # запишем данные за июль, которые исключаем, в отдельный датафрейм  
excluded_data = logs.query('date.dt.month == 7')
```

```
In [25]: all_users = logs['device_id_hash'].nunique()  
all_users
```

Out[25]: 7551

```
In [26]: # оставим в датафрейме Logs только данные за август  
logs = logs.query('date.dt.month == 8')
```

- Много ли событий и пользователей потеряли, отбросив старые данные?

```
In [27]: print(f'Удалено событий - {len(excluded_data)}, относительные потери данных составили '  
f'{round(100 * len(excluded_data) / (len(logs) + len(excluded_data)), 2)} процентов')
```

Удалено событий - 2826, относительные потери данных составили 1.16 процентов

```
In [28]: print(f'Количество уникальных пользователей до удаления - {all_users}, \n'  
f'количество уникальных пользователей после удаления - {logs["device_id_hash"].nunique()}, \n'  
f'Количество удаленных пользователей - {all_users - logs["device_id_hash"].nunique()}, \n'  
f'Относительные потери удаленных уникальных пользователей - '  
f'{round(100 * (all_users - logs["device_id_hash"].nunique()) / all_users, 2)} процентов')
```

Количество уникальных пользователей до удаления - 7551,
количество уникальных пользователей после удаления - 7534,
Количество удаленных пользователей - 17,
Относительные потери удаленных уникальных пользователей - 0.23 процентов

Удалена незначительная часть данных и уникальных пользователей. Проверим, как изменились основные характеристики событий на каждого пользователя:

```
In [29]: logs.pivot_table(index='device_id_hash', values='event_name', aggfunc='count').describe()
```

Out[29]:

	event_name
count	7534.000000
mean	31.973321
std	65.090307
min	1.000000
25%	9.000000
50%	19.000000
75%	37.000000
max	2307.000000

В среднем на пользователя приходится почти 32 события. Минимальное и максимальное количество событий на пользователя - 1 и 2307 соответственно. Получился сильный разброс значений. Значение медианы составляет 19 событий. В целом, характеристика сильно не изменились, сохранился разброс данных, значение медианы изменилось с 20 до 19 событий.

- Проверим, что есть пользователи из всех трёх экспериментальных групп.

```
In [30]: print('Уникальное число пользователей в группе 246 - ',  
        logs.query('exp_id==246').agg({'device_id_hash':'nunique'}).device_id_hash)  
print('Уникальное число пользователей в группе 247 - ',  
        logs.query('exp_id==247').agg({'device_id_hash':'nunique'}).device_id_hash)  
print('Уникальное число пользователей в группе 248 - ',  
        logs.query('exp_id==248').agg({'device_id_hash':'nunique'}).device_id_hash)  
print('Всего уникальных пользователей - ', logs.agg({'device_id_hash':'nunique'}).device_id_hash)
```

Уникальное число пользователей в группе 246 - 2484

Уникальное число пользователей в группе 247 - 2513

Уникальное число пользователей в группе 248 - 2537

Всего уникальных пользователей - 7534

```
In [31]: group = logs.pivot_table(index='exp_id', values='device_id_hash', aggfunc='nunique').reset_index()
group['share'] = round(100 * group['device_id_hash'] / sum(group['device_id_hash']), 2)
group
```

Out[31]:

	exp_id	device_id_hash	share
0	246	2484	32.97
1	247	2513	33.36
2	248	2537	33.67

Доля уникальных пользователей практически не изменилась после удаления части данных.

Вывод:

До удаления части данных :

- количество логов составляло 243713.
- чаще всего происходит событие MainScreenAppear (показ главного экрана) - 119 101, в 2,5 раза меньше происходило событие OffersScreenAppear (появление предложения на экране) - 46 808, немногим меньше у события CartScreenAppeare (отображение корзины) 42 668. Событие PaymentScreenSuccessful - успешная оплата - случилось 34 118 раз. Меньше всего пользователи проходили Tutorial (обучение) - всего 1 018 раз.
- все пользователи относятся только к одной группе, то есть попаданий одновременно в несколько групп отсутствуют.
- выяснили, что у нас есть данные от 25 июля 2019 до 7 августа 2019. С 25 по 31 июля количество логов увеличиваются - 25 июля было 9 логов, 31 июля - 2030 логов. Данных за июль очень мало по сравнению с данными за август. 1 августа было совершено 36 141 логов, что намного больше данных за июль. С 1 по 7 августа данные не сильно разнятся, находясь в диапазоне от 31 тысячи до 36 тысяч. Оставили наиболее полные данные с 1 августа по 7 августа.

После удаления :

- Удалено событий - 2826, относительные потери данных составили 1.16 процентов, количество логов - 240887.
- Количество уникальных пользователей до удаления - 7551, количество уникальных пользователей после удаления - 7534, количество удаленных пользователей - 17, относительные потери удаленных пользователей - 0.23 процентов
- В среднем на пользователя приходится почти 32 события. Минимальное и максимальное количество событий на пользователя - 1 и 2307 соответственно. Получился сильный разброс значений. То есть есть пользователи, которые совершили только 1 событие (возможно, только перешли на главный экран и больше не совершали никаких действий) и пользователи, которые регулярно пользуются приложением. Так как величина среднего значения не показательна при выбросах, будем ориентироваться на значение медианы, которое составляет 19 событий.
- Уникальное число пользователей в группе 246 - 2484, в группе 247 - 2513, в группе 248 - 2537. Всего уникальных пользователей - 7534

Шаг 4. Изучение воронки событий

- Посмотреть, какие события есть в логах, как часто они встречаются. Отсортировать события по частоте.

In [32]: # проверим, как распределены события:

```
logs.pivot_table(index='event_name', values='event_timestamp', aggfunc='count').sort_values(by='event_timestamp').reset_index()
```

Out[32]:

	event_name	event_timestamp
0	Tutorial	1005
1	PaymentScreenSuccessful	33918
2	CartScreenAppear	42303
3	OffersScreenAppear	46333
4	MainScreenAppear	117328

- Чаще всего происходит событие MainScreenAppear (показ главного экрана) - 117 328. Скорее всего, эта страница открывается первой при входе в мобильное приложение.
 - В 2,5 раза меньше происходило событие OffersScreenAppear (появление каталога на экране) - 46 333, немногим меньше у события CartScreenAppear (отображение корзины) 42 303.
 - Событие PaymentScreenSuccessful - успешная оплата - случилось 33 918 раз.
 - Меньше всего пользователи проходили Tutorial (обучение) - всего 1 018 раз.
-
- Посчитать, сколько пользователей совершили каждое из этих событий. Отсортировать события по числу пользователей. Посчитать долю пользователей, которые хоть раз совершали событие.

```
In [33]: # посчитаем, сколько пользователей совершили каждое из событий  
logs.pivot_table(index='device_id_hash', values='event_name', aggfunc='nunique').head()
```

Out[33]:

device_id_hash	event_name
6888746892508752	1
6909561520679493	4
6922444491712477	4
7435777799948366	1
7702139951469979	4

Теперь по каждому пользователю видим, сколько уникальных событий он совершил. Отберем тех пользователей, у которых таких событий больше 4:

```
In [34]: logs.pivot_table(index='device_id_hash', values='event_name', aggfunc='nunique').query('event_name > 4').count()
```

Out[34]: event_name 466
dtype: int64

Всего пользователей, для которых имеются все события, 466. Посмотрим, сколько пользователей совершило все события, кроме Tutorial (обучение):

```
In [35]: logs.query('event_name != "Tutorial"').pivot_table(index='device_id_hash', values='event_name', aggfunc='nunique') \  
.query('event_name > 3').count()
```

Out[35]: event_name 3429
dtype: int64

Таких пользователей гораздо больше.

```
In [36]: # Отсортируем события по числу пользователей.  
logs.pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique').sort_values(by='device_id_hash')
```

Out[36]:

device_id_hash

event_name	device_id_hash
Tutorial	840
PaymentScreenSuccessful	3539
CartScreenAppear	3734
OffersScreenAppear	4593
MainScreenAppear	7419

Итак, наибольшее количество уникальных пользователей попадает на главную страницу - 7419. Вспомним, что после "фильтрации данных" у нас стало 7534 уникальных пользователей. Возможно, что 115 пользователей прошли сначала обучение сразу после скачивания приложения, а после перешли на главный экран, либо это техническая ошибка.

```
In [37]: # Посчитаем долю пользователей, которые хоть раз совершали событие.  
logs.pivot_table(index='device_id_hash', values='event_name', aggfunc='nunique').query('event_name != 0').count()
```

Out[37]: event_name 7534
dtype: int64

Получается, что каждый пользователь хоть раз совершал событие, так как всего уникальных пользователей тоже 7534.

In [38]: # найдем долю каждого события от общего числа

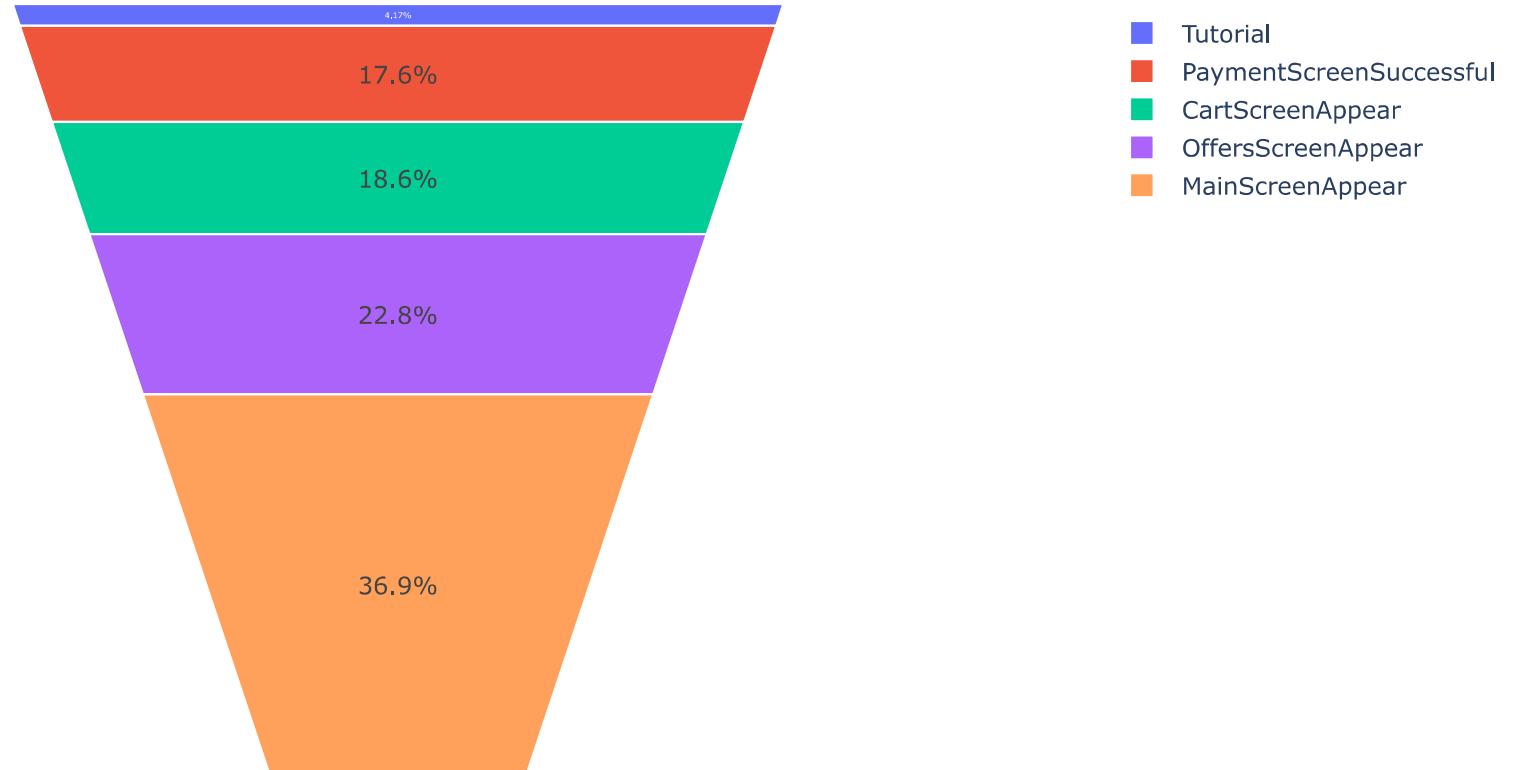
```
events = logs.pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique') \
    .sort_values(by='device_id_hash').reset_index()
events['share'] = round(100 * events['device_id_hash'] / logs['device_id_hash'].nunique(), 2)
events
```

Out[38]:

	event_name	device_id_hash	share
0	Tutorial	840	11.15
1	PaymentScreenSuccessful	3539	46.97
2	CartScreenAppear	3734	49.56
3	OffersScreenAppear	4593	60.96
4	MainScreenAppear	7419	98.47

In [39]: # отобразим воронку событий

```
fig = px.funnel_area(  
    names = events.event_name,  
    values = events.share)  
  
fig.show()
```



Показательно, что намечается последовательность событий: просмотр основного экрана-каталог/предложения товаров-корзина-оплата

- Предположить, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при

Предположим, что сразу после скачивания приложения пользователю предлагается пройти обучение, на этом этапе можно выбрать его пройти и затем попасть на главную страницу, либо можно пропустить обучение и попасть сразу на главную страницу (помним, что у нас было 115 пользователей, для которых главная страница не являлась первой). Скорее всего, обучение можно пройти на любом этапе.

Пока что последовательность выстраивается следующая: главная страница -> каталог/предложения товаров -> страница корзины -> успешная оплата.

- По воронке событий посчитать, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитать отношение числа пользователей с событием B к количеству пользователей с событием A, а также отношение числа пользователей с событием C к количеству пользователей с событием B.

Зная количество людей на каждом этапе, можно посчитать долю дошедших до определённого шага. А ещё — долю перешедших с каждого шага на следующий.

```
In [40]: # построим таблицу, аналогичную events, добавив столбец с конверсией и исключив событие tutorial
events = logs.query('event_name != "Tutorial"').pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique') \
    .sort_values(by='device_id_hash')
events['share'] = round(100 * events['device_id_hash'] / logs.query('event_name != "Tutorial"')['device_id_hash'].nunique(), 2)
events = events.sort_values(by='share', ascending=False).reset_index()
events['conversion'] = round(100 * events['device_id_hash'] / max(events['device_id_hash']), 2)
events
```

Out[40]:

	event_name	device_id_hash	share	conversion
0	MainScreenAppear	7419	98.53	100.00
1	OffersScreenAppear	4593	61.00	61.91
2	CartScreenAppear	3734	49.59	50.33
3	PaymentScreenSuccessful	3539	47.00	47.70

```
In [41]: list = []

step = 1
for i in range(4):
    if i == 0:
        list.append(100 * events['device_id_hash'][i] / events['device_id_hash'][i])
    else:
        list.append(100 * events['device_id_hash'][step] / events['device_id_hash'][step - 1])
    step += 1
events['conv_step'] = list
events['conv_step'] = round(events['conv_step'], 2)
events
```

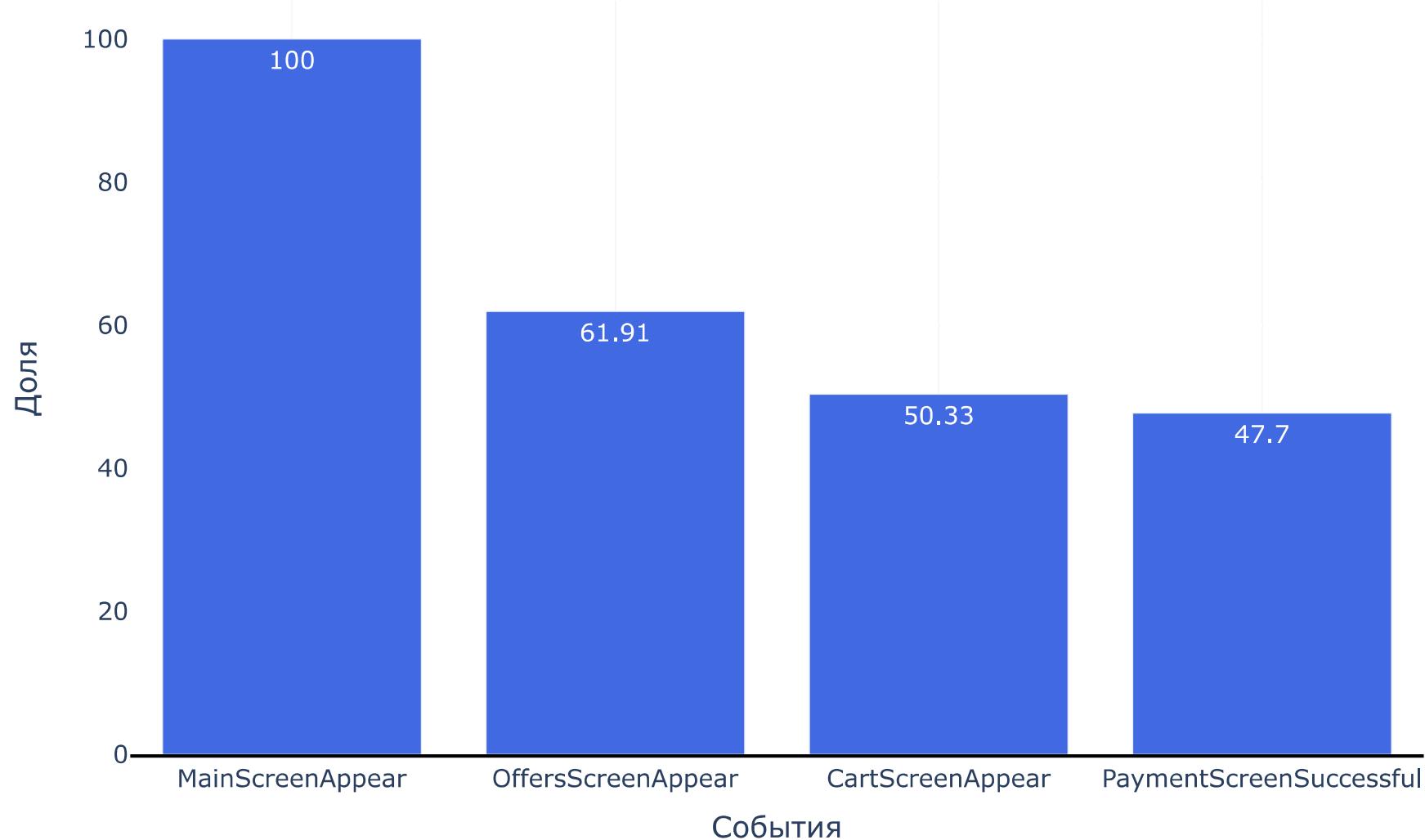
Out[41]:

	event_name	device_id_hash	share	conversion	conv_step
0	MainScreenAppear	7419	98.53	100.00	100.00
1	OffersScreenAppear	4593	61.00	61.91	61.91
2	CartScreenAppear	3734	49.59	50.33	81.30
3	PaymentScreenSuccessful	3539	47.00	47.70	94.78

In [42]: # Визуализируем полученные конверсии:

```
px.bar(events,
       y='conversion',
       x='event_name',
       color_discrete_sequence=["royalblue"],
       text_auto=True) \
.update_layout(plot_bgcolor='rgba(0,0,0,0)') \
.update_xaxes(showline=True,
              linewidth=2,
              linecolor='black',
              gridcolor='LightGrey') \
.update_layout(plot_bgcolor='rgba(0,0,0,0)',
              autosize=False,
              width=950,
              height=600,
              title = {
                  'text': "Доля пользователей к количеству пользователей на первом шаге",
                  'y':0.98,
                  'x':0.5,
                  'xanchor': 'center',
                  'yanchor': 'top'
              },
              xaxis_title="События",
              yaxis_title="Доля",
              font=dict(
                  size=15),
              )
```

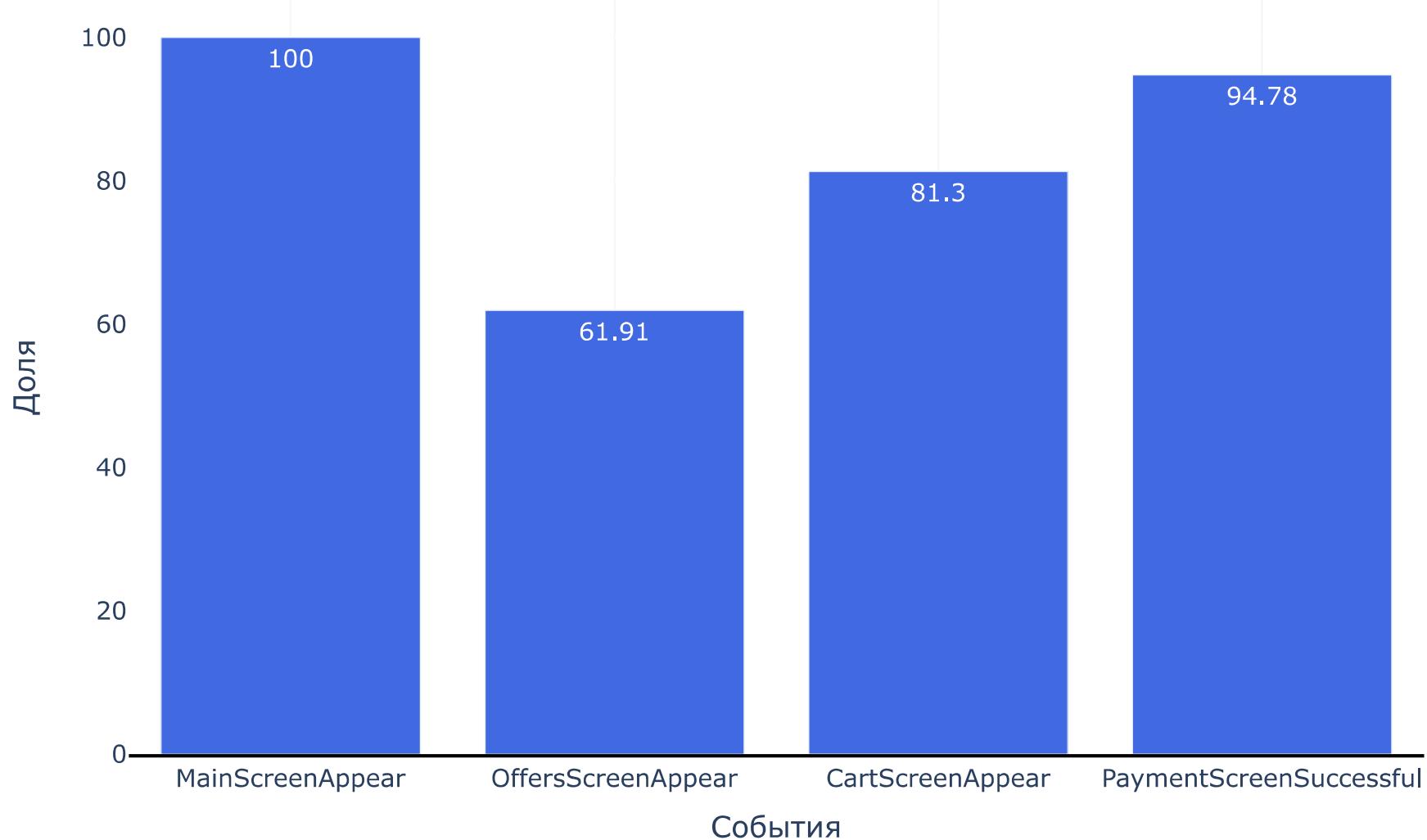
Доля пользователей к количеству пользователей на первом шаге



Можно отметить, что до последнего шага доходят 47,7% пользователей.

```
In [43]: px.bar(events,
    y='conv_step',
    x='event_name',
    color_discrete_sequence=["royalblue"],
    text_auto=True) \
.update_layout(plot_bgcolor='rgba(0,0,0,0)') \
.update_xaxes(showline=True,
    linewidth=2,
    linecolor='black',
    gridcolor='LightGrey') \
.update_layout(plot_bgcolor='rgba(0,0,0,0)',
    autosize=False,
    width=950,
    height=600,
    title = {
        'text': "Доля пользователей к количеству пользователей на предыдущем шаге",
        'y':0.98,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    },
    xaxis_title="События",
    yaxis_title="Доля",
    font=dict(
        size=15),
    )
```

Доля пользователей к количеству пользователей на предыдущем шаге



Низкий показатель конверсии при переходе от главного экрана к каталогу/предложению товаров и услуг. Это может говорить как о системных, так и о технических проблемах. Например, сайт может быть свёрстан не самым удобным для пользователя образом, либо страница долго загружается. Конверсия на шагах страницы корзины и оплаты довольно высокая.

- На каком шаге теряется больше всего пользователей?

In [44]: # Дополним имеющуюся таблицу events

```
list = []

step = 1
for i in range(4):
    if i == 0:
        list.append(events['device_id_hash'][i] - events['device_id_hash'][i])
    else:
        list.append(events['device_id_hash'][step - 1] - events['device_id_hash'][step])
    step += 1
events['lost_users'] = list
events['lost_users'] = round(events['lost_users'], 2)
events['lost_users_share'] = round(100 * events['lost_users'] / events['device_id_hash'], 2)
events
```

Out[44]:

	event_name	device_id_hash	share	conversion	conv_step	lost_users	lost_users_share
0	MainScreenAppear	7419	98.53	100.00	100.00	0	0.00
1	OffersScreenAppear	4593	61.00	61.91	61.91	2826	61.53
2	CartScreenAppear	3734	49.59	50.33	81.30	859	23.00
3	PaymentScreenSuccessful	3539	47.00	47.70	94.78	195	5.51

Больше половины пользователей уходят на шаге перехода от главного экрана. Меньше всего уходят на переходе из корзины к странице оплаты. Значит, сервис работает и проблем у пользователей не вызывает. Пятая часть пользователей "отваливается" на шаге от каталога к корзине, возможно, что проблема с ассортиментом или пользователям неудобно выбирать товары.

Вывод:

- Чаще всего происходит событие MainScreenAppear (показ главного экрана) - 117 328. Скорее всего, эта страница открывается первой при входе в мобильное приложение. В 2,5 раза меньше происходило событие OffersScreenAppear (появление каталога на экране) - 46 333, немногим меньше у события CartScreenAppear (отображение корзины) 42 303. Событие PaymentScreenSuccessful - успешная оплата - случилось 33 918 раз. Меньше всего пользователи проходили Tutorial (обучение) - всего 1 018 раз.
- Сразу после скачивания приложения пользователю предлагается пройти обучение, на этом этапе можно выбрать его пройти и затем попасть на главную страницу, либо можно пропустить обучение и попасть сразу на главную страницу (помним, что у нас было 115 пользователей, для которых главная страница не являлась первой). Скорее всего, обучение можно пройти на любом этапе. Последовательность событий выстраивается следующая: главная страница -> каталог/предложения товаров -> страница корзины -> успешная оплата.
- До последнего шага (оплата) доходят 47,7% пользователей.
- Низкий показатель конверсии при переходе от главного экрана к каталогу/предложению товаров и услуг - 61,9%. Это может говорить как о системных, так и о технических проблемах. Например, сайт может быть свёрстан не самым удобным для пользователя образом, либо страница долго загружается. Конверсия на шагах страницы корзины и оплаты довольно высокая - 81,3% и 94,78% соответственно.

- Больше половины пользователей уходят на шаге перехода от главного экрана - 61,53%. Меньше всего уходят на переходе из корзины к странице оплаты - 5,51%. Значит, сервис работает и проблем у пользователей не вызывает. Пятая часть пользователей "отваливается" на шаге от каталога к корзине, возможно, что проблема с ассортиментом или пользователям неудобно выбирать товары.

Шаг 5. Изучение результатов эксперимента

- Сколько пользователей в каждой экспериментальной группе?

```
In [45]: group = logs.pivot_table(index='exp_id', values='device_id_hash', aggfunc='nunique').reset_index()
group['share'] = round(100 * group['device_id_hash'] / sum(group['device_id_hash']), 2)
group
```

Out[45]:

	exp_id	device_id_hash	share
0	246	2484	32.97
1	247	2513	33.36
2	248	2537	33.67

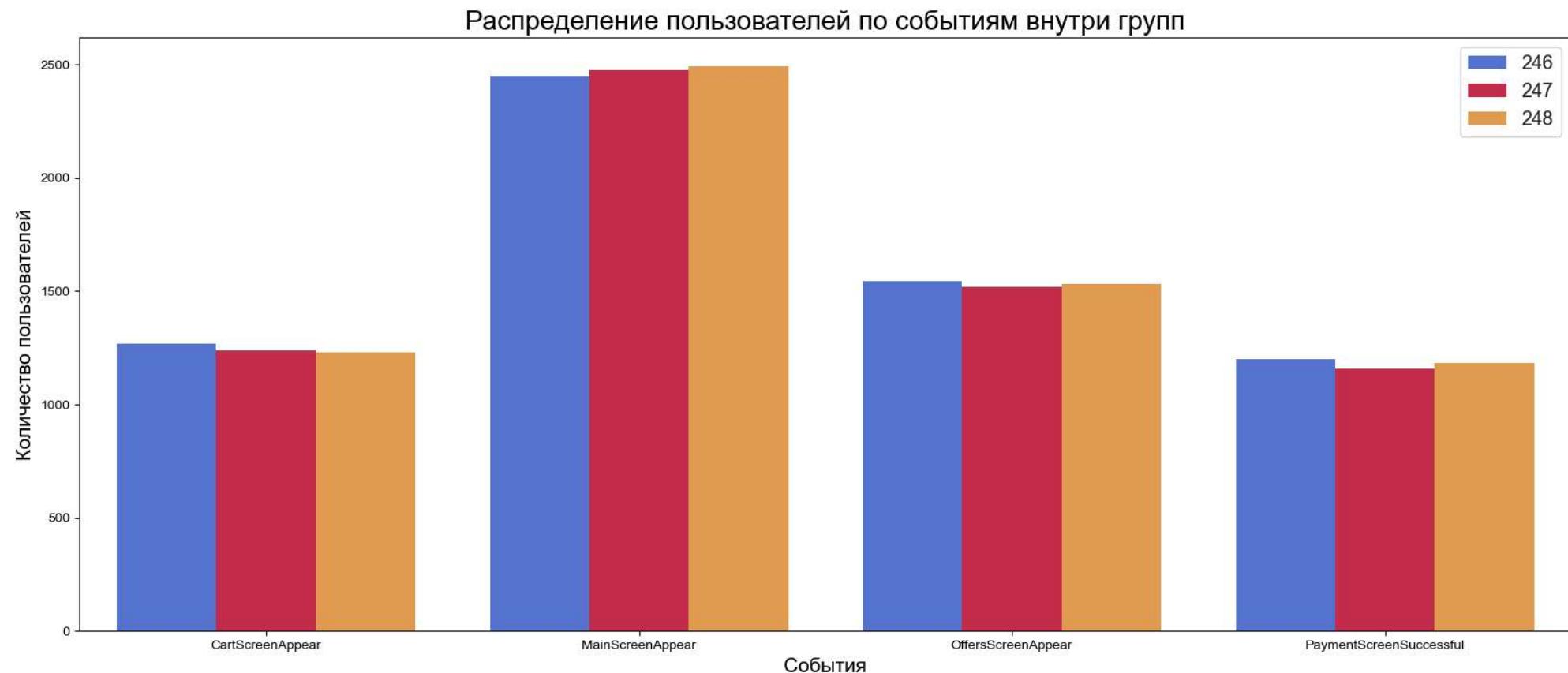
```
In [46]: print('Уникальное число пользователей в группе 246 -',
          logs.query('exp_id==246').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Уникальное число пользователей в группе 247 -',
          logs.query('exp_id==247').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Уникальное число пользователей в группе 248 -',
          logs.query('exp_id==248').agg({'device_id_hash':'nunique'}).device_id_hash)
print('Всего уникальных пользователей - ', logs.agg({'device_id_hash':'nunique'}).device_id_hash)
```

Уникальное число пользователей в группе 246 - 2484
Уникальное число пользователей в группе 247 - 2513
Уникальное число пользователей в группе 248 - 2537
Всего уникальных пользователей - 7534

Доля уникальных пользователей в экспериментальной группе чуть выше контрольных.

Проверим, как распределяются пользователи по событиям:

```
In [47]: plt.figure(figsize=(20, 8))
sns.barplot(data=logs.query('event_name != "Tutorial"').pivot_table(index=['event_name', 'exp_id'],
                                                               values='device_id_hash',
                                                               aggfunc='nunique').reset_index(),
             x="event_name",
             y="device_id_hash",
             hue="exp_id",
             #color=['blue', 'Limegreen', 'darkgreen']
             palette=['royalblue', 'crimson', '#fb9f3a'])
sns.set_style("whitegrid")
plt.xlabel('События', fontsize=15)
plt.ylabel("Количество пользователей", fontsize=15)
plt.title('Распределение пользователей по событиям внутри групп', fontsize=20)
plt.legend(fontsize='x-large')
plt.show()
```



В целом, события по группам распределены практически равномерно. Уникальных пользователей в группах тестирования 246, 247 и 248 составляет 2484, 2513 и 2537 соответственно.

- Есть 2 контрольные группы для A/A-эксперимента, чтобы проверить корректность всех механизмов и расчётов. Проверить, находят ли статистические критерии разницу между выборками 246 и 247.

In [48]: # выведем распределение уникальных пользователей каждой группы по событиям

```
groups = logs.query('event_name != "Tutorial"') \
    .pivot_table(columns='exp_id', index='event_name', values='device_id_hash', aggfunc='nunique') \
    .sort_values(by=246, ascending=False) \
    .reset_index()
groups['control_group'] = groups[246] + groups[247]
groups
```

Out[48]:

exp_id	event_name	246	247	248	control_group
0	MainScreenAppear	2450	2476	2493	4926
1	OffersScreenAppear	1542	1520	1531	3062
2	CartScreenAppear	1266	1238	1230	2504
3	PaymentScreenSuccessful	1200	1158	1181	2358

Применим z-тест для проверки гипотезы о равенстве долей и сформулируем гипотезы, а также сравним группу 246 и группу 247 (экспериментальную)

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

In [49]: # обернем код в функцию

```
def stat_test(all_data, group_data, first_group, second_group, event, col, text):

    print(text)

    alpha = 0.05 # критический уровень статистической значимости

    test_group_first = all_data.query('exp_id == @first_group')
    test_group_second = all_data.query('exp_id == @second_group')

    positive = np.array([group_data.query('event_name == @event')[first_group],
                         group_data.query('event_name == @event')[second_group]])

    prob = np.array([all_data.query('exp_id == @first_group')[col].nunique(),
                    all_data.query('exp_id == @second_group')[col].nunique()])

    # пропорция в первой группе:
    p1 = positive[0]/prob[0]

    # пропорция во второй группе:
    p2 = positive[1]/prob[1]

    # пропорция в комбинированном датасете:
    p_combined = (positive[0] + positive[1]) / (prob[0] + prob[1])

    # разница пропорций в датасетах
    difference = p1 - p2

    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/prob[0] + 1/prob[1]))

    # задаем стандартное нормальное распределение (среднее 0, std.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

    if p_value < alpha:# ваш код
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
```

```
In [50]: stat_test(logs, groups, 246, 247, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.75705972]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.24809546]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.22883372]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.11456679]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Для каждого события не получилось отвергнуть нулевую гипотезу, значит, будем считать, что не оснований считать доли в группах тестирования разными.

- Выбрать самое популярное событие. Посчитать число пользователей, совершивших это событие в каждой из контрольных групп.

```
In [51]: group = logs.query('event_name != "Tutorial"') \
            .pivot_table(index=['event_name', 'exp_id'], values='device_id_hash', aggfunc='nunique')
group
```

Out[51]:

		device_id_hash
	event_name	exp_id
CartScreenAppear	246	1266
	247	1238
	248	1230
MainScreenAppear	246	2450
	247	2476
	248	2493
OffersScreenAppear	246	1542
	247	1520
	248	1531
PaymentScreenSuccessful	246	1200
	247	1158
	248	1181

Самым популярным событием является страница главного экрана. В группах 246, 247 и 248 такое событие совершили 2450, 2476 и 2493 пользователей соответственно.

```
In [52]: group = logs.query('event_name == "MainScreenAppear"') \
            .pivot_table(index=['event_name', 'exp_id'], values='device_id_hash', aggfunc='nunique')
group['share'] = round(100 * group['device_id_hash'] / sum(group['device_id_hash']), 2)
group
```

Out[52]:

		device_id_hash	share
	event_name	exp_id	
MainScreenAppear	246	2450	33.02
	247	2476	33.37
	248	2493	33.60

- Используя функцию, аналогично поступим с группой с изменённым шрифтом. Сравним результаты с каждой из контрольных групп в отдельности по каждому событию. Сравним результаты с объединённой контрольной группой.

Сравним группу 246 (контрольную) и группу 248 (экспериментальную)

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [53]: stat_test(logs, groups, 246, 248, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.29497219]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.20836205]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.07842923]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.21225533]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Для каждого события не получилось отвергнуть нулевую гипотезу, значит, будет считать, что не оснований считать доли в группах тестирования разными.

Теперь сравним группу 247 (контрольную) и группу 248 (экспериментальную)

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [54]: stat_test(logs, groups, 247, 248, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.45870536]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.91978178]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.57861979]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.73734151]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Как и в предыдущих тестах, для каждого события не получилось отвергнуть нулевую гипотезу, значит, будет считать, что не оснований считать доли в группах тестирования разными.

Сравним группу 246 и 247 (контрольные) с группой 248 (экспериментальную)

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

In [55]: # обернем код в функцию

```
def stat_test_control(all_data, group_data, second_group, event, col, text, col_two):

    print(text)

    alpha = 0.05 # критический уровень статистической значимости

    test_group_first = all_data.query('exp_id in [246, 247]')
    test_group_second = all_data.query('exp_id == @second_group')
    positive = np.array([group_data.query('event_name == @event')[col_two],
                         group_data.query('event_name == @event')[second_group]])

    prob = np.array([all_data.query('exp_id in [246, 247]')[col].nunique(),
                    all_data.query('exp_id == @second_group')[col].nunique()])

    # пропорция в первой группе:
    p1 = positive[0]/prob[0]

    # пропорция во второй группе:
    p2 = positive[1]/prob[1]

    # пропорция в комбинированном датасете:
    p_combined = (positive[0] + positive[1]) / (prob[0] + prob[1])

    # разница пропорций в датасетах
    difference = p1 - p2

    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/prob[0] + 1/prob[1]))

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

    if p_value < alpha:# Ваш код
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
```

```
In [56]: stat_test_control(logs, groups, 248, 'MainScreenAppear', 'device_id_hash',
    'Рассматриваем событие MainScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'OffersScreenAppear', 'device_id_hash',
    'Рассматриваем событие OffersScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'CartScreenAppear', 'device_id_hash',
    'Рассматриваем событие CartScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'PaymentScreenSuccessful', 'device_id_hash',
    'Рассматриваем событие PaymentScreenSuccessful', 'control_group')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.29424527]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.4342555]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.18175875]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.60042943]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

В данном случае для каждого события не получилось отвергнуть нулевую гипотезу, значит, будем считать, что не оснований считать доли в группах тестирования разными.

- Какой уровень значимости выбран при проверке статистических гипотез выше? Посчитать, сколько проверок статистических гипотез сделано. При уровне значимости 0.1 каждый десятый раз можно получать ложный результат. Какой уровень значимости стоит применить?

Всего произведено 16 проверок статистических гипотез. Уровень значимости выбран при проверке статистических гипотез - 0,05.

Можно попробовать применить поправку Бонферони: число сравнений невелико (16), поэтому примем уровень значимости в 16 раз меньше того, что мы использовали ранее и проведем тесты еще раз с новым уровнем значимости - 0,003125.

Запишем заново функции для проведения теста:


```
In [57]: def stat_test(all_data, group_data, first_group, second_group, event, col, text):

    print(text)

    alpha = 0.05 / 16 # критический уровень статистической значимости

    test_group_first = all_data.query('exp_id == @first_group')
    test_group_second = all_data.query('exp_id == @second_group')

    positive = np.array([group_data.query('event_name == @event')[first_group],
                         group_data.query('event_name == @event')[second_group]])

    prob = np.array([all_data.query('exp_id == @first_group')[col].nunique(),
                    all_data.query('exp_id == @second_group')[col].nunique()])

    # пропорция в первой группе:
    p1 = positive[0]/prob[0]

    # пропорция во второй группе:
    p2 = positive[1]/prob[1]

    # пропорция в комбинированном датасете:
    p_combined = (positive[0] + positive[1]) / (prob[0] + prob[1])

    # разница пропорций в датасетах
    difference = p1 - p2

    z_value = difference / math.sqrt(p_combined * (1 - p_combined) * (1/prob[0] + 1/prob[1]))

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

    if p_value < alpha:# Ваш код
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
        )
```

```

def stat_test_control(all_data, group_data, second_group, event, col, text, col_two):

    print(text)

    alpha = 0.05 / 16 # критический уровень статистической значимости

    test_group_first = all_data.query('exp_id in [246, 247]')
    test_group_second = all_data.query('exp_id == @second_group')
    positive = np.array([group_data.query('event_name == @event')[col_two],
                         group_data.query('event_name == @event')[second_group]])

    prob = np.array([all_data.query('exp_id in [246, 247]')[col].nunique(),
                    all_data.query('exp_id == @second_group')[col].nunique()])

    # пропорция в первой группе:
    p1 = positive[0]/prob[0]

    # пропорция во второй группе:
    p2 = positive[1]/prob[1]

    # пропорция в комбинированном датасете:
    p_combined = (positive[0] + positive[1]) / (prob[0] + prob[1])

    # разница пропорций в датасетах
    difference = p1 - p2

    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/prob[0] + 1/prob[1]))

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

    if p_value < alpha:# ваш код
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
        )

```

Применим z-тест для проверки гипотезы о равенстве долей и сформулируем гипотезы, а также сравним группу 246 и группу 247

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [58]: stat_test(logs, groups, 246, 247, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 246, 247, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.75705972]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.24809546]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.22883372]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.11456679]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравним группу 246 и группу 248 (экспериментальную), сформулируем гипотезы:

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [59]: stat_test(logs, groups, 246, 248, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 246, 248, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.29497219]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.20836205]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.07842923]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.21225533]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравним группу 247 и группу 248 (экспериментальную), сформулируем гипотезы:

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [60]: stat_test(logs, groups, 247, 248, 'MainScreenAppear', 'device_id_hash', 'Рассматриваем событие MainScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'OffersScreenAppear', 'device_id_hash', 'Рассматриваем событие OffersScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'CartScreenAppear', 'device_id_hash', 'Рассматриваем событие CartScreenAppear')
print()
stat_test(logs, groups, 247, 248, 'PaymentScreenSuccessful', 'device_id_hash', 'Рассматриваем событие PaymentScreenSuccessful')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.45870536]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.91978178]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.57861979]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.73734151]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравним группу 246 и 247 с группой 248 (экспериментальную), сформулируем гипотезы:

- Нулевая гипотеза: статистически значимой разницы между долями нет
- Альтернативная гипотеза: статистически значимая разница между долями есть

```
In [61]: stat_test_control(logs, groups, 248, 'MainScreenAppear', 'device_id_hash',
    'Рассматриваем событие MainScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'OffersScreenAppear', 'device_id_hash',
    'Рассматриваем событие OffersScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'CartScreenAppear', 'device_id_hash',
    'Рассматриваем событие CartScreenAppear', 'control_group')
print()
stat_test_control(logs, groups, 248, 'PaymentScreenSuccessful', 'device_id_hash',
    'Рассматриваем событие PaymentScreenSuccessful', 'control_group')
```

Рассматриваем событие MainScreenAppear

р-значение: [0.29424527]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие OffersScreenAppear

р-значение: [0.4342555]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие CartScreenAppear

р-значение: [0.18175875]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Рассматриваем событие PaymentScreenSuccessful

р-значение: [0.60042943]

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Вывод:

- События по группам тестирования распределены практически равномерно. Уникальных пользователей в группах тестирования 246, 247 и 248 составляет 2484, 2513 и 2537 соответственно.
- Самым популярным событием является страница главного экрана. В группах 246, 247 и 248 такое событие совершили 2450, 2476 и 2493 пользователей соответственно.
- Всего проведено 2 раза по 16 тестов при различных уровнях значимости. При уровне значимости 0,05 и сравнении групп 246 и 247, 246 и 248, 247 и 248, 246/247 и 248 выявлено, что статистически значимых различий между долями нет. При уровне значимости 0,05/16 ($\approx 0,0031$) и сравнении групп 246 и 247, 246 и 248, 247 и 248, 246/247 и 248 выявлено, что статистически значимых различий между долями также нет.

6. Общий вывод

Было проведено исследование, которое включало в себя изучение воронки событий, а также анализ результатов проведения A/A/B теста.

В ходе предобработки данных:

- привели наименования столбцов к нижнему регистру и переименовали в соответствии с синтаксисом
- доля удаленных данных 0,17% (удалены дубликаты)
- добавлены два столбца типа datetime - полная дата и дата без времени

До фильтрации данных:

- количество логов составляло 243713.
- чаще всего происходит событие MainScreenAppear (показ главного экрана) - 119 101, в 2,5 раза меньше происходило событие OffersScreenAppear (появление предложения на экране) - 46 808, немногим меньше у события CartScreenAppear (отображение корзины) 42 668. Событие PaymentScreenSuccessful - успешная оплата - случилось 34 118 раз. Меньше всего пользователи проходили Tutorial (обучение) - всего 1 018 раз.
- все пользователи относятся только к одной группе, то есть попаданий одновременно в несколько групп отсутствуют.
- выяснили, что у нас есть данные от 25 июля 2019 до 7 августа 2019. С 25 по 31 июля количество логов увеличиваются - 25 июля было 9 логов, 31 июля - 2030 логов. Данных за июль очень мало по сравнению с данными за август. 1 августа было совершено 36 141 логов, что намного больше данных за июль. С 1 по 7 августа данные не сильно разнятся, находясь в диапазоне от 31 тысячи до 36 тысяч. Оставили наиболее полные данные с 1 августа по 7 августа.

После удаления данных:

- Удалено событий - 2826, относительные потери данных составили 1.16 процентов, количество логов стало - 240887.
- Количество уникальных пользователей до удаления - 7551, количество уникальных пользователей после удаления - 7534, количество удаленных пользователей - 17, относительные потери удаленных пользователей - 0.23 процентов
- В среднем на пользователя приходится почти 32 события. Минимальное и максимальное количество событий на пользователя - 1 и 2307 соответственно. Получился сильный разброс значений. То есть есть пользователи, которые совершили только 1 событие (возможно, только перешли на главный экран и больше не совершали никаких действий) и пользователи, которые регулярно пользуются приложением. Так как величина среднего значения не показательна при выбросах, будем ориентироваться на значение медианы, которое составляет 19 событий.
- Уникальное число пользователей в группе 246 - 2484, в группе 247 - 2513, в группе 248 - 2537. Всего уникальных пользователей - 7534

При анализе воронки событий выявлено:

- Чаще всего происходит событие MainScreenAppear (показ главного экрана) - 117 328. Скорее всего, эта страница открывается первой при входе в мобильное приложение. В 2,5 раза меньше происходило событие OffersScreenAppear (появление каталога на экране) - 46 333, немногим меньше у события CartScreenAppear (отображение корзины) 42 303. Событие PaymentScreenSuccessful - успешная оплата - случилось 33 918 раз. Меньше всего пользователи проходили Tutorial (обучение) - всего 1 018 раз.
- Сразу после скачивания приложения пользователю предлагается пройти обучение, на этом этапе можно выбрать его пройти и затем попасть на главную страницу, либо можно пропустить обучение и попасть сразу на главную страницу (помните, что у нас было 115 пользователей, для которых главная страница не являлась первой). Скорее всего, обучение можно пройти на любом этапе. Последовательность событий выстраивается следующая: главная страница -> каталог/предложения товаров -> страница корзины -> успешная оплата.
- До последнего шага (оплата) доходят 47,7% пользователей.

- Низкий показатель конверсии при переходе от главного экрана к каталогу/предложению товаров и услуг - 61,9%. Это может говорить как о системных, так и о технических проблемах. Например, сайт может быть свёрстан не самым удобным для пользователя образом, либо страница долго загружается. Конверсия на шагах страницы корзины и оплаты довольно высокая - 81,3% и 94,78% соответственно.
- Больше половины пользователей уходят на шаге перехода от главного экрана - 61,53%. Меньше всего уходят на переходе из корзины к странице оплаты - 5,51%. Значит, сервис работает и проблем у пользователей не вызывает. Пятая часть пользователей "отваливается" на шаге от каталога к корзине, возможно, что проблема с ассортиментом или пользователям неудобно выбирать товары.

В ходе анализа результатов теста:

- События по группам тестирования распределены практически равномерно. Уникальных пользователей в группах тестирования 246, 247 и 248 составляет 2484, 2513 и 2537 соответственно.
- Самым популярным событием является страница главного экрана. В группах 246, 247 и 248 такое событие совершили 2450, 2476 и 2493 пользователей соответственно.

Были поставлено несколько гипотез:

- Гипотеза 1: статистической значимости различий между долями при выбранном уровне значимости в 0,05 нет
- Гипотеза 2: статистической значимости различий между долями при выбранном уровне значимости в 0,0032 нет

В результате получено следующее:

Гипотеза 1 подтверждена: Всего проведено 2 раза по 16 тестов при различных уровнях значимости. При уровне значимости 0,05 и сравнении групп 246 и 247, 246 и 248, 247 и 248, 246/247 и 248 выявлено, что статистически значимых различий между долями нет.

Гипотеза 2 подтверждена: При уровне значимости 0,05/16 ($\approx 0,0031$) и сравнении групп 246 и 247, 246 и 248, 247 и 248, 246/247 и 248 выявлено, что статистически значимых различий между долями также нет.

Рекомендации:

- добавление новых шрифтов сильно не влияет на основные метрики и поведение пользователей.
- проверить функциональность главной страницы, в дальнейшем провести исследование, зависит ли низкая конверсия от конкретной модели используемого гаджета, от времени события, от количества багов, дизайна и т.д.