Стартап по продаже продуктов питания

Описание проекта

Место работы: стартап, который продаёт продукты питания.

Глобальная задача: нужно разобраться, как ведут себя пользователи мобильного приложения.

Задачи:

1. Изучить воронку продаж;
2. Узнать, как пользователи доходят до покупки;
3. Узнать сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах;
4. Узнать шаги на которых «застревают»;
5. Исследовать результаты A/A/B-эксперимента.

Описание исследования: дизайнеры предложили поменять шрифты во всём приложении, но менеджеры считают, что это отпугнет пользователей. Принятие решения будет производится по результатам A/A/B-теста.

Описание данных
Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми.
Каждая запись в логе — это действие пользователя, или событие:

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

Оглавление

In [1]:
```
# Импорт библиотек
import pandas as pd
import scipy.stats as stats
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from plotly import graph_objects as go
import math as mth
```

In [2]:
```
# Чтение файлов
try:
    df = pd.read_csv('datasets/logs_exp.csv', sep='\t')
# если не получилось прочитать файл из локальной папки, то загружаем данные из сети
except:
    df = pd.read_csv('/datasets/logs_exp.csv', sep='\t')
```

1 Предобработка данных

In [3]:
```
df.info()
df.head()
```

```
<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
```

Out[3]:

	EventName	DeviceIDHash	EventTimestamp	ExpId
0	MainScreenAppear	4575588528974610257	1564029816	246
1	MainScreenAppear	7416695313311560658	1564053102	246
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248
3	CartScreenAppear	3518123091307005509	1564054127	248
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248

In [4]:
```
# Замена названий колонок
df.columns = ['event_name', 'user_id', 'event_time', 'group']
```

In [5]:
```
# Явные дубликаты
df.duplicated().sum()
```

Out[5]: 413

In [6]:
```
# Удаление явных дубликатов с обнулением индексов
df = df.drop_duplicates().reset_index(drop=True)
```

In [7]:
```
# Повторная проверка
df.duplicated().sum()
```

Out[7]: 0

Заметка: `event_time` записано с точностью до секунды, поэтому `unit='s'` .

In [8]:
```
# Преобразование типа данных
df['event_time'] = pd.to_datetime(df['event_time'], unit='s')

# Для укрупненного анализа по датам добавим столбец
df['date'] = df['event_time'].dt.strftime('%Y-%m-%d')
df['date'] = pd.to_datetime(df['date'])
```

In [9]:
```
# Столбец с возможными неявными дубликатами
df['event_name'].unique()
```

Out[9]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)
```

Заметка: неявных дубликатов нет.

In [10]:
```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   event_name  243713 non-null  object        
 1   user_id     243713 non-null  int64         
 2   event_time  243713 non-null  datetime64[ns]
 3   group       243713 non-null  int64         
 4   date        243713 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(2), object(1)
memory usage: 9.3+ MB
```

Out[10]:

	event_name	user_id	event_time	group	date
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	246	2019-07-25
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	246	2019-07-25
2	PaymentScreenSuccessful	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25
3	CartScreenAppear	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25
4	PaymentScreenSuccessful	6217807653094995999	2019-07-25 11:48:42	248	2019-07-25

Итоги:

Мы произвели замену названий данных на удобные, удалили дубликаты, проверили наличие пропусков, поменяли тип данных в графе с информацией о дате и времени, добавили столбец с датой.

Описание:

- event_name — название события;
- user_id — уникальный идентификатор пользователя;
- event_time — дата и время события;
- group — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная;
- date — дата события.

In [11]:
```
# Проверяем наличие пользователей, попавших в две группы и более
df.pivot_table(index='user_id', values='group', aggfunc='nunique').nunique()
```

Out[11]:
```
group    1
dtype: int64
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
event_name    0
user_id       0
event_time    0
group         0
date          0
dtype: int64
```

Заметка: пропуски не обнаружены.

2 Изучение и проверка данных

2.1 События в логе

In [13]:
```
print('Кол-во записей в логе:', df.shape[0], 'шт.')
print('Кол-во вариантов событий:', df['event_name'].nunique(), 'шт.', '\n')

print('События:')
df['event_name'].unique()
```

```
Кол-во записей в логе: 243713 шт.
Кол-во вариантов событий: 5 шт.

События:
```

Out[13]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)
```

2.2 Пользователи в логе

In [14]:
```
print('Кол-во пользователей в логе:', df['user_id'].nunique(), 'шт.')
```

```
Кол-во пользователей в логе: 7551 шт.
```

2.3 Среднее кол-во событий на пользователя

In [15]:
```
print('Среднее кол-во событий на пользователя без группировки:', round(df.shape[0]/df['user_id'].nunique()), 'шт.')
```

```
Среднее кол-во событий на пользователя без группировки: 32 шт.
```

In [16]:
```
print('Среднее кол-во событий на пользователя с группировкой по событиям:',
      int(df.groupby('user_id')['event_name'].agg('count').median()),
      'шт.'
     )
```

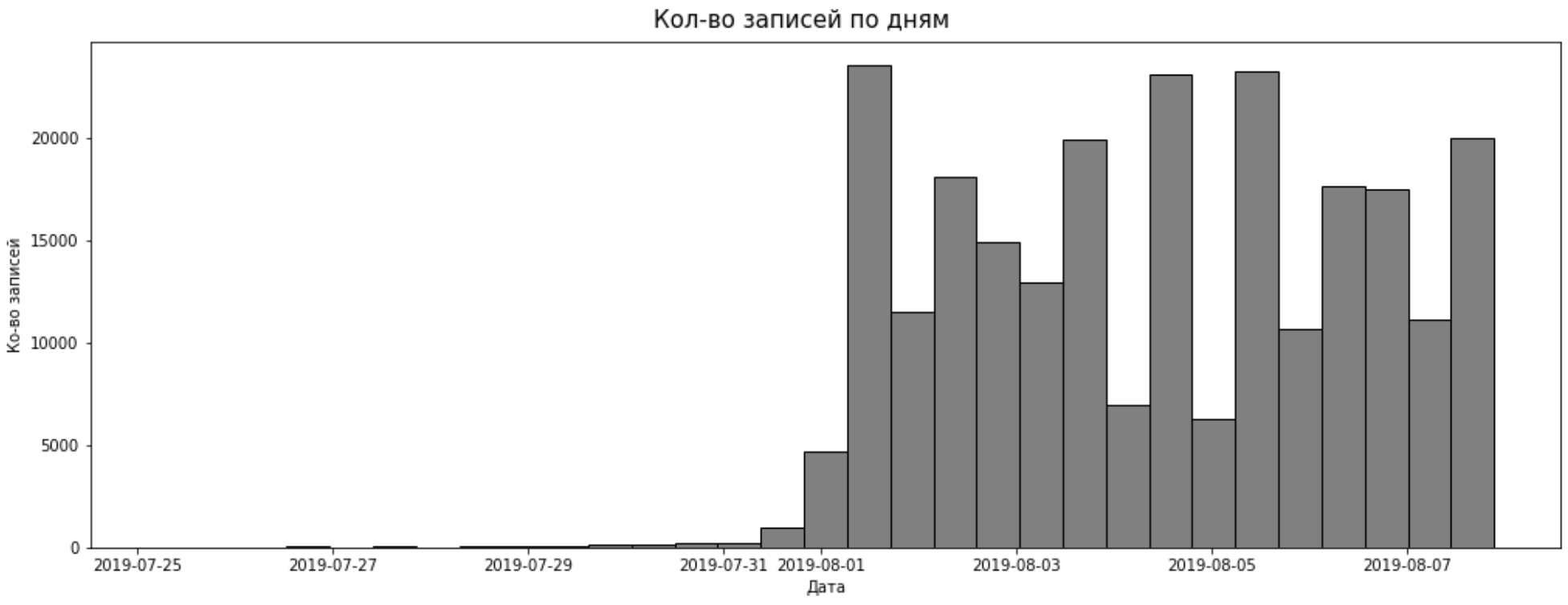
```
Среднее кол-во событий на пользователя с группировкой по событиям: 20 шт.
```

2.4 Даты

In [17]:
```
print('Начало:', df['event_time'].min())
print('Конец:', df['event_time'].max())
```

```
Начало: 2019-07-25 04:43:36
Конец: 2019-08-07 21:15:17
```

In [18]:
```
# Постром гистограмму по дате и времени

# Размер поля построения
plt.figure(figsize=(17, 6))

# График
plt.hist(df['event_time'], color = 'grey', edgecolor = 'black',
         bins = int(31))

# Подписи
plt.suptitle('Кол-во записей по дням', x=0.5, y=0.93, fontsize=15);
plt.xlabel("Дата")
plt.ylabel("Ко-во записей");
```



In [19]:
```
# Выведем таблицу
df.pivot_table(index='date', values='user_id', aggfunc='count').style.background_gradient(cmap='RdYlGn')
```

Out[19]:

date	user_id
2019-07-25 00:00:00	9
2019-07-26 00:00:00	31
2019-07-27 00:00:00	55
2019-07-28 00:00:00	105
2019-07-29 00:00:00	184
2019-07-30 00:00:00	412
2019-07-31 00:00:00	2030
2019-08-01 00:00:00	36141
2019-08-02 00:00:00	35554
2019-08-03 00:00:00	33282
2019-08-04 00:00:00	32968
2019-08-05 00:00:00	36058
2019-08-06 00:00:00	35788
2019-08-07 00:00:00	31096

Выводы:

Мы видим, что до 7-го числа кол-во данных незначительно, т.е. можно считать, что у нас в наличии данные с 8-го числа, так как кол-во записей 8-го и 7-го чисел различаются почти в 18 раз.

По причинам выше отбросим данные до 8-го августа.

In [20]:
```
# Создаем копию
df_old = df.copy()
```

In [21]:
```
#print('До:', df_old.shape[0])

# Удаляем данные
df = df.query('date>=@dt.date(2019, 8, 1)')

#print('После:', df.shape[0])
#print('Потери:', '{:.2%}'.format((df_old.shape[0] - df.shape[0]) / df_old.shape[0]))
```

In [22]:
```
print('Кол-во уникальных пользователей до очистки:', df_old['user_id'].nunique())
print('Кол-во типов событий до очистки:', df_old['event_name'].nunique())
print('Кол-во событий до очистки:', df_old['event_name'].count())

print(' ')

print('Кол-во уникальных пользователей после очистки:', df['user_id'].nunique())
print('Кол-во типов событий после очистки:', df['event_name'].nunique())
print('Кол-во событий после очистки:', df['event_name'].count())

print(' ')

print('Потери уникальных пользователей:',
      '{:.2%}'.format((df_old['user_id'].nunique() - df['user_id'].nunique()) / df_old['user_id'].nunique())
     )
print('Потери событий:',
      '{:.2%}'.format((df_old['event_name'].count() - df['event_name'].count()) / df_old['event_name'].count())
     )
```

```
Кол-во уникальных пользователей до очистки: 7551
Кол-во типов событий до очистки: 5
Кол-во событий до очистки: 243713

Кол-во уникальных пользователей после очистки: 7534
Кол-во типов событий после очистки: 5
Кол-во событий после очистки: 240887

Потери уникальных пользователей: 0.23%
Потери событий: 1.16%
```

Заметка: потери менее 5%, что является приемлемым.

In [23]:
```
# Удаляем старые индексы
df = df.reset_index(drop=True)
```

2.5 Проверка групп

In [24]:
```
df.head()
```

Out[24]:

	event_name	user_id	event_time	group	date
0	Tutorial	3737462046622621720	2019-08-01 00:07:28	246	2019-08-01
1	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	246	2019-08-01
2	MainScreenAppear	3737462046622621720	2019-08-01 00:08:55	246	2019-08-01
3	OffersScreenAppear	3737462046622621720	2019-08-01 00:08:58	246	2019-08-01
4	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	247	2019-08-01

In [25]:
```
df['group'].unique()
```

Out[25]:
```
array([246, 247, 248])
```

In [26]:
```
df1 = df.pivot_table(index='group', values='user_id', aggfunc='count')
df1['delta 246, %'] = round(df1 / df1.loc[246, 'user_id'] * 100, 2)
df1
```

Out[26]:

group	user_id	delta 246, %
246	79302	100.00
247	77022	97.12
248	84563	106.63

Заметка: есть данные о всех группах.

3 Воронка событий

3.1 События в логах, как часто они встречаются

In [27]:
```
df.head()
```

Out[27]:

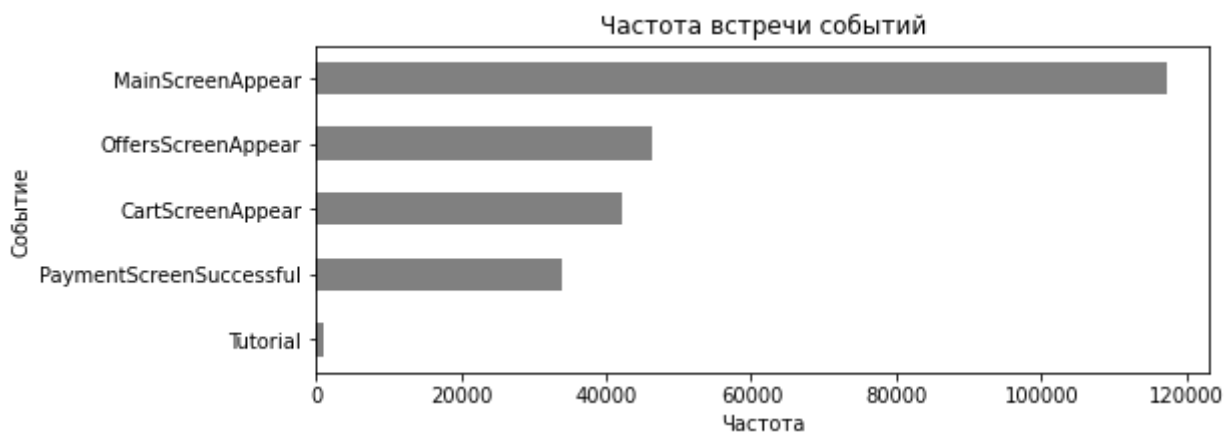
	event_name	user_id	event_time	group	date
0	Tutorial	3737462046622621720	2019-08-01 00:07:28	246	2019-08-01
1	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	246	2019-08-01
2	MainScreenAppear	3737462046622621720	2019-08-01 00:08:55	246	2019-08-01
3	OffersScreenAppear	3737462046622621720	2019-08-01 00:08:58	246	2019-08-01
4	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	247	2019-08-01

In [28]:
```
df.pivot_table(index='event_name', values='user_id', aggfunc='count')\
.sort_values(by='user_id', ascending=False)
```

Out[28]:

event_name	user_id
MainScreenAppear	117328
OffersScreenAppear	46333
CartScreenAppear	42303
PaymentScreenSuccessful	33918
Tutorial	1005

In [29]:
```
(
    df
    .pivot_table(index='event_name', values='user_id', aggfunc='count')
    .sort_values(by='user_id', ascending=True)
    .plot(kind='barh', title='Частота встречи событий', legend=False, figsize=(8, 3), color='grey')
)

# Подписи
plt.xlabel("Частота")
plt.ylabel("Событие");
```



Заметка: наиболее часто встречается событие "MainScreenAppear", что логично, ведь главный экран запускается при запуске приложения.

События:

- появление главного экрана;
- появление экрана с предложениями;
- появление экрана корзины;
- экран успешной оплаты;
- руководство.

3.2 Количество пользователей, совершавших каждое из этих событий

Отсортируем события по числу пользователей. Посчитаем долю пользователей, которые хоть раз совершали событие.

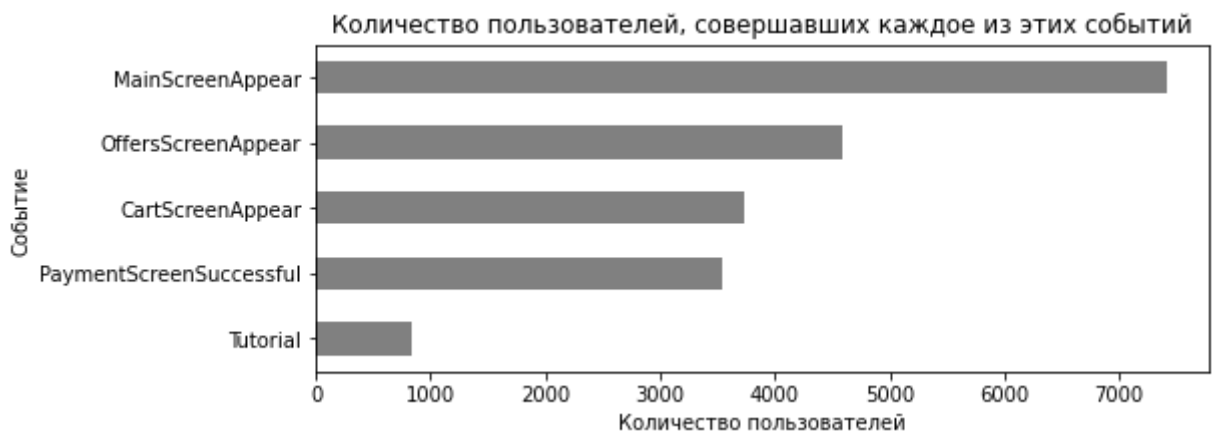
In [30]:
```
# Кол-во пользователей
(
    df
    .pivot_table(index='event_name', values='user_id', aggfunc='nunique')
    .sort_values(by='user_id', ascending=False)
    .style.set_caption('Количество пользователей, совершавших каждое из этих событий')
)

```

Out[30]:

Количество пользователей,
совершавших каждое из этих событий

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

In [31]:
```
(
    df
    .pivot_table(index='event_name', values='user_id', aggfunc='nunique')
    .sort_values(by='user_id', ascending=True)
    .plot(kind='barh', title='Количество пользователей, совершавших каждое из этих событий', legend=False, figsize=(8, 3),
)

# Подписи
plt.xlabel("Количество пользователей")
plt.ylabel("Событие");
```



In [32]:
```
# Доля пользователей, хоть раз совершавших событие
round(
    df
    .pivot_table(index='event_name', values='user_id', aggfunc='nunique')
    .sort_values(by='user_id', ascending=False)
    / df['user_id'].nunique() *100,
    2
).rename(columns={'user_id':'Доля, %'})
```

Out[32]:

event_name	Доля, %
MainScreenAppear	98.47
OffersScreenAppear	60.96
CartScreenAppear	49.56
PaymentScreenSuccessful	46.97
Tutorial	11.15

Заметки:

Выводы:

- 7419 пользователей хотя бы раз открывали главную страницу приложения (MainScreenAppear) это 98,47% всех пользователей. Возможно оставшиеся пользователи не смогли попасть на главную страницу из-за ошибок или переходили по ссылке, открывающей сразу товар, но сведений об этом нет;
- 4593 пользователя хотя бы раз видели предложение товаров (OffersScreenAppear) это 60,96% всех пользователей - пользователь может закрывать окно с предложениями и продолжать работу с приложением до шага CartScreenAppear.
- 3734 пользователя хотя бы раз открывали корзину (CartScreenAppear) это 49,56% всех пользователей.
- 3539 пользователей хотя бы раз попадали на страницу с успешной оплатой(PaymentScreenSuccessful) это 46,97% всех пользователей.
- 840 пользователей хотя бы раз открывали руководство пользователя (Tutorial) это 11,15% всех пользователей.

Окно Tutorial, вероятно, можно запустить самому, вследствие чего на него попадают немногие, или большинство пользователей уже давно пользуются приложением, а окно появляется только после регистрации.

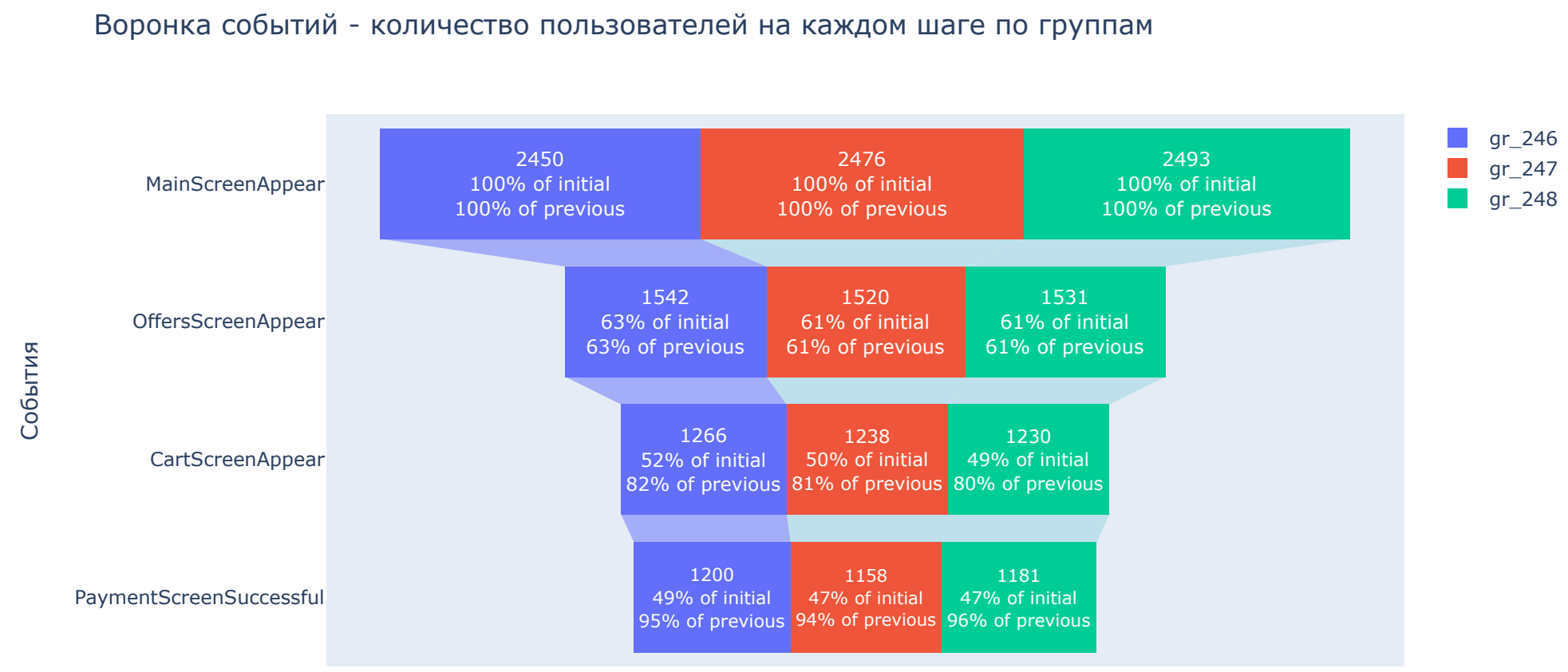
Окно Tutorial, вероятно, можно запустить самому, вследствие чего на него попадают немногие, или большинство пользователей уже давно пользуются приложением, а окно появляется только после регистрации.

Порядок действий выглядит так:

1. открытие главной страницы приложения (MainScreenAppear) →
2. появление окна предложений/листа товаров (OffersScreenAppear) →
3. кладёт их в корзину (CartScreenAppear) →
4. производит оплату (PaymentScreenSuccessful).

In [33]:
```python
# Подготовка к отображению информации
action = df[df['event_name'] != 'Tutorial']['event_name'].unique()

gr_246 = (
    df[df['event_name'] != 'Tutorial']
    .query('group == 246')
    .pivot_table(index='event_name',
                 values='user_id',
                 aggfunc='nunique')
    .sort_values(by='user_id',
                 ascending=False)
)['user_id']

gr_247 = (
    df[df['event_name'] != 'Tutorial']
    .query('group == 247')
    .pivot_table(index='event_name',
                 values='user_id',
                 aggfunc='nunique')
    .sort_values(by='user_id',
                 ascending=False)
)['user_id']

gr_248 = (
    df[df['event_name'] != 'Tutorial']
    .query('group == 248')
    .pivot_table(index='event_name',
                 values='user_id',
                 aggfunc='nunique')
    .sort_values(by='user_id',
                 ascending=False)
)['user_id']
```

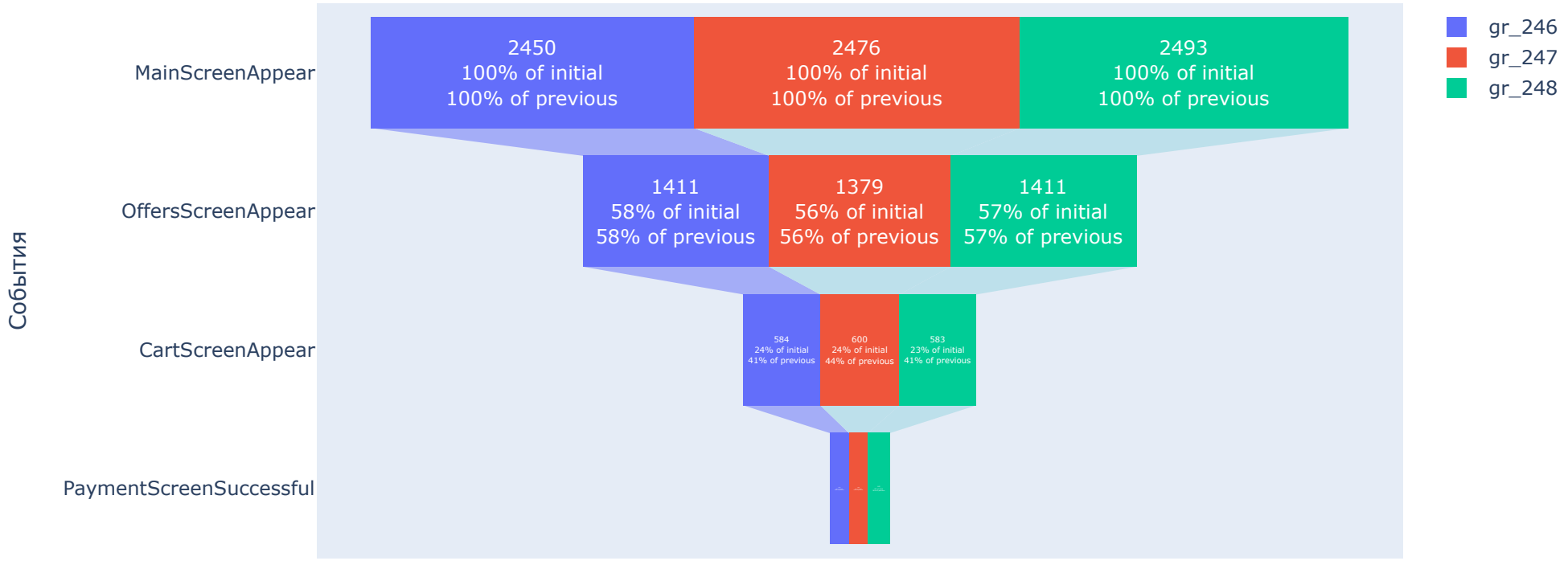
In [34]:
```python
# Построение графика↔
```



Заметка: больше всего теряется на втором шаге до 39%, а до конца доходят порядка 47 % от начальной группы.

In [35]:
```python
# Подготовка к отображению информации↔
```

In [36]:
```
# Построение графика↔
```



Заметка: полный цикл проходят порядка 6-7% пользователей.

Выводы:

Предполагаемой последовательности шагов следуют *многие* пользователи.

Для сравнения от первого события до оплаты по этому пути дошло 6% / 6% / 7% для групп 246 / 247 / 248 соответственно (в группе с изменённым шрифтом наибольшие показатели).

Возможно, кто-то минует страницу корзины, пользуясь мгновенной оплатой.

4 Результаты эксперимента

4.1 Количество пользователей в каждой экспериментальной группе

In [37]:
```
# Разобьем пользователей на группы
users_bygroup = df.groupby('group')['user_id'].nunique()
users_bygroup['246+247'] = users_bygroup[246] + users_bygroup[247]
users_bygroup
```

Out[37]:
```
group
246        2484
247        2513
248        2537
246+247    4997
Name: user_id, dtype: int64
```

Посчитаем число пользователей, совершивших события в каждой из контрольных групп. Посчитаем долю пользователей, совершивших эти события. Проверим, будет ли отличие между группами статистически достоверным.

In [38]:
```
# Создадим таблицу для тестов
event_group_test = df[df['event_name']!='Tutorial'].pivot_table(
    index='event_name',
    columns='group',
    values='user_id',
    aggfunc='nunique').sort_values(by=246, ascending=False)

event_group_test = event_group_test.reset_index()
event_group_test['246+247'] = event_group_test[246] + event_group_test[247]
event_group_test['all'] = event_group_test['246+247'] + event_group_test[248]

event_group_test['part_A1'] = (event_group_test[246] / users_bygroup[246] * 100).round(1)
event_group_test['part_A2'] = (event_group_test[247] / users_bygroup[247] * 100).round(1)
event_group_test['part_B'] = (event_group_test[248] / users_bygroup[248] * 100).round(1)
event_group_test['part_A1+A2'] = ((event_group_test[246] + event_group_test[247]) / \
                                  (users_bygroup[246] + users_bygroup[247]) * 100).round(1)

event_group_test
```

Out[38]:

group	event_name	246	247	248	246+247	all	part_A1	part_A2	part_B	part_A1+A2
0	MainScreenAppear	2450	2476	2493	4926	7419	98.6	98.5	98.3	98.6
1	OffersScreenAppear	1542	1520	1531	3062	4593	62.1	60.5	60.3	61.3
2	CartScreenAppear	1266	1238	1230	2504	3734	51.0	49.3	48.5	50.1
3	PaymentScreenSuccessful	1200	1158	1181	2358	3539	48.3	46.1	46.6	47.2

4.2 Проверка статистической разницы между выборками 246 и 247

Параметры теста:

- z-тест;
- alpha = 1% - следствие необходимости проверки равенства.

Гипотезы:

H_0: число пользователей одинаково, изменений нет.
H_1: число пользователей разное - есть изменения.

In [39]:
```
# Проверка гипотезы о равенстве долей функия "z-тест"
def z_test(group1, group2, alpha):
    for i in event_group_test.index:
        p1 = event_group_test[group1][i] / users_bygroup[group1]

        # пропорция успехов во второй группе:
        p2 = event_group_test[group2][i] / users_bygroup[group2]

        # пропорция успехов в комбинированном датасете:
        p_combined = ((event_group_test[group1][i] + event_group_test[group2][i]) /
                      (users_bygroup[group1] + users_bygroup[group2]))

        # разница пропорций в датасетах
        difference = p1 - p2

        # считаем статистику в ст.отклонениях стандартного нормального распределения
        z_value = difference / mth.sqrt(p_combined * (1 - p_combined) *
                                        (1/users_bygroup[group1] + 1/users_bygroup[group2]))

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

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

        print('{} p-значение: {}'.format(event_group_test['event_name'][i], p_value))

        if (p_value < alpha):
            print("Отвергаем нулевую гипотезу: между долями ЕСТЬ значимая разница")
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными")

        print('')
```

In [40]:
```
# Проводим тест
z_test(246, 247, 0.01)
```

```
MainScreenAppear p-значение: 0.7570597232046099
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.2480954578522181
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.22883372237997213
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.11456679313141849
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Заметки:

Степень различия ключевых метрик по группам зависит от необходимой чувствительности эксперимента. A/B-тест применяют там, где степень различия между группами ***не больше 10%***, или же подвержена колебаниям. В таких случаях точность ***A/A-теста в 1%*** — распространённый критерий.

В следствие того, что A1 и A2 (246 и 247) должны быть одинаковыми, был взят уровень статистической значимости равный 1%. По результатам A1/A2 тестирования для всех событий разница **не может считаться значимой, значит, группы считаем контрольными**.

Можно сказать, что разбиение на группы работает корректно.

Для ***A/B теста*** не нужна такая высокая точность, поэтому в дальнейших экспериментах выберем стандартный уровень значимости в 5%.

Проведем 3 группы тестов: A1/B , A2/B, A1+A2/B, что позволит уловить наличие отличий.

Общие для трех тестов гипотезы:

H_0: число пользователей одинаково, изменений нет.
H_1: число пользователей разное - есть изменения.

Тест A1/B
Параметры теста:

- z-тест;
- alpha = 5%.

In [41]:
```
# A1/B
z_test(246, 248, 0.05)
```

```
MainScreenAppear p-значение: 0.2949721933554552
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.20836205402738917
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.07842923237520116
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.2122553275697796
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Тест A2/B
Параметры теста:

- z-тест;
- alpha = 5%.

In [42]:
```
# A2/B
z_test(247, 248, 0.05)
```

```
MainScreenAppear p-значение: 0.4587053616621515
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.9197817830592261
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.5786197879539783
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.7373415053803964
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Тест A1+A2/B
Параметры теста:

- z-тест;

- alpha = 5%.

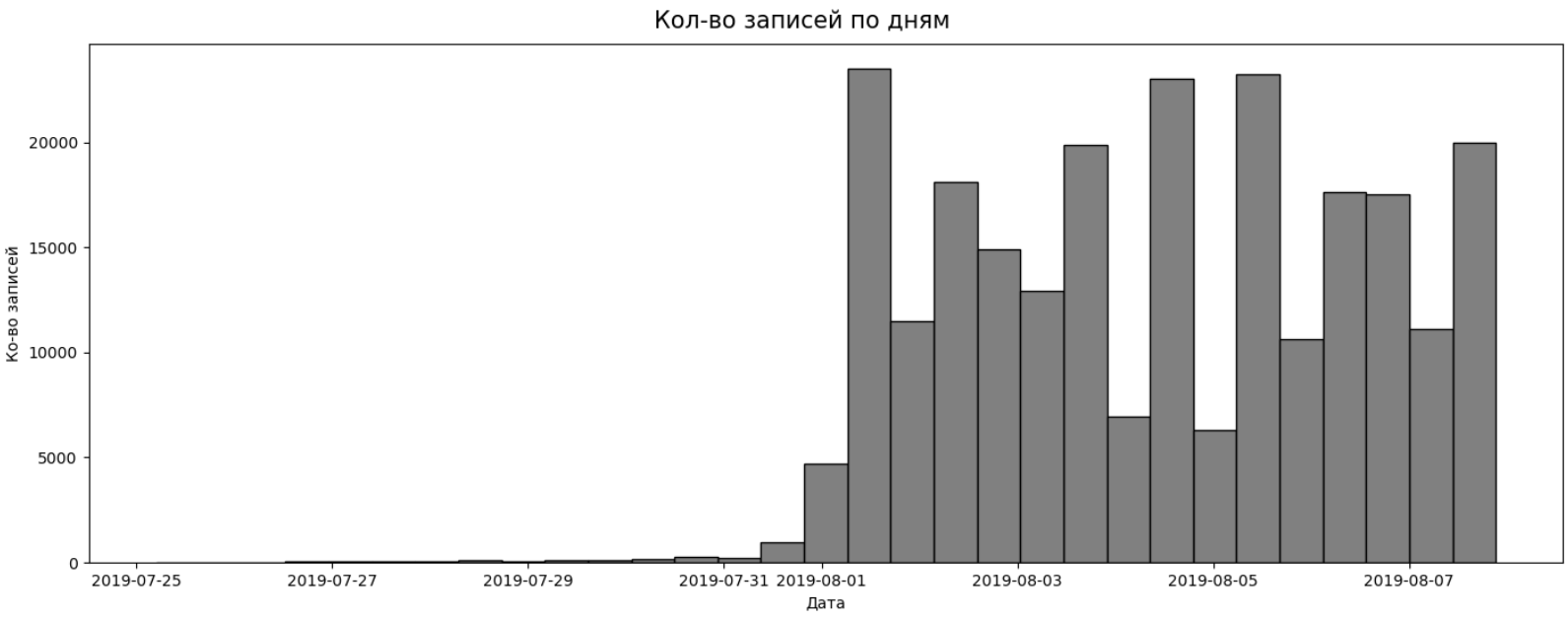
In [43]:
```
# A1+A2/B
z_test('246+247', 248, 0.05)
```

```
MainScreenAppear p-значение: 0.29424526837179577
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.43425549655188256
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.18175875284404386
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.6004294282308704
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

5 Выводы

1. Изучение данных:
В ходе изучения данных стало изветсно: до 7-го августа кол-во данных незначительно, т.е. можно считать, что у нас в наличии данные с 8-го августа числа, так как кол-во записей 8-го и 7-го чисел различаются почти в 18 раз.

По причинам выше были удалены данные до 8-го августа, потери составили 1.16% от первоначального объема, что допустимо.



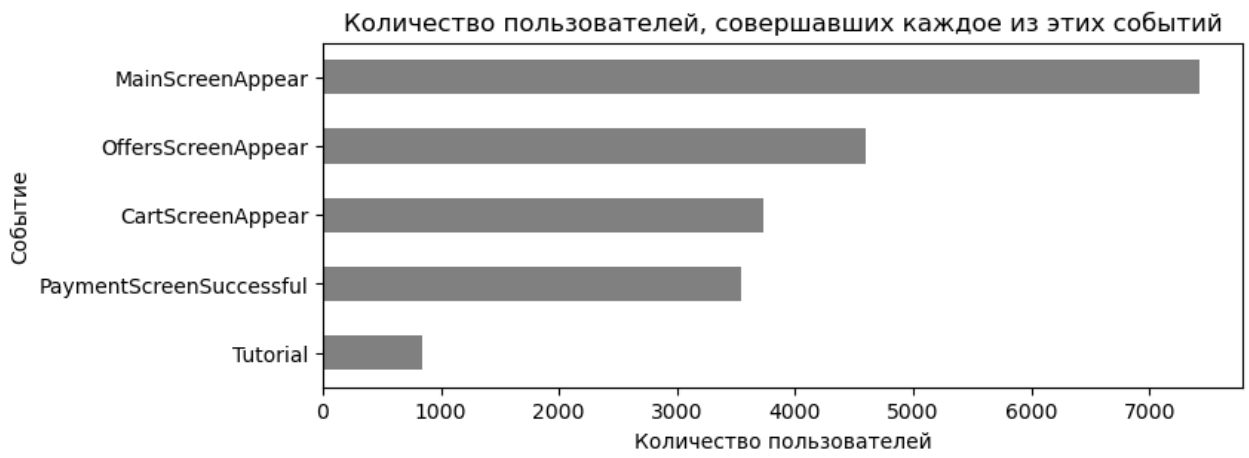
2. Воронка продаж:
Порядок действий (воронка):

1. открытие главной страницы приложения (MainScreenAppear) →
2. появление окна предложений/листа товаров (OffersScreenAppear) →
3. кладём их в корзину (CartScreenAppear) →
4. производим оплату (PaymentScreenSuccessful).

Отдельным пунктом:
0. Tutorial - руководство пользователя.

Что стоит отметить:

- 7419 пользователей хотя бы раз открывали главную страницу приложения (MainScreenAppear) это 98,47% всех пользователей. Возможно оставшиеся пользователи не смогли попасть на главную страницу из-за ошибок или переходили по ссылке, открывающей сразу товар, но сведений об этом нет;
- 4593 пользователя хотя бы раз видели предложение товаров (OffersScreenAppear) это 60,96% всех пользователей - пользователь может закрывать окно с предложениями и продолжать работу с приложением до шага CartScreenAppear.
- 3734 пользователя хотя бы раз открывали корзину (CartScreenAppear) это 49,56% всех пользователей.
- 3539 пользователей хотя бы раз попадали на страницу с успешной оплатой(PaymentScreenSuccessful) это 46,97% всех пользователей.
- 840 пользователей хотя бы раз открывали руководство пользователя (Tutorial) это 11,15% всех пользователей.

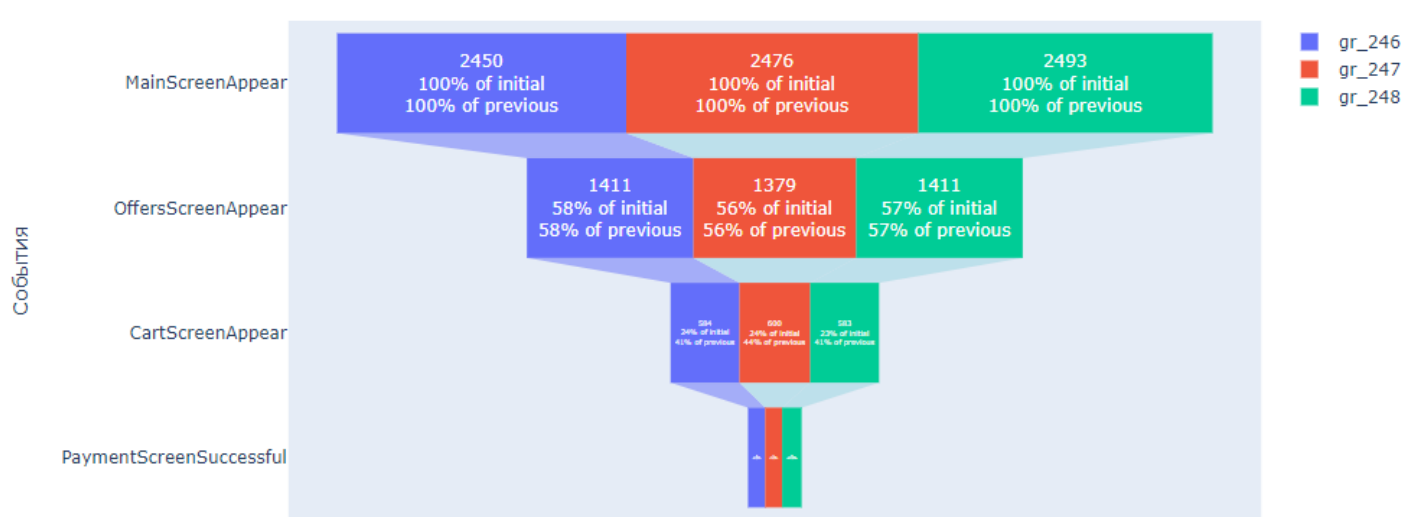


event_name	Доля, %
MainScreenAppear	98.47
OffersScreenAppear	60.96
CartScreenAppear	49.56
PaymentScreenSuccessful	46.97
Tutorial	11.15

Окно Tutorial, вероятно, можно запустить самому, вследствие чего на него попадают немногие, или большинство пользователей уже давно пользуются приложением, а окно появляется только после регистрации.

Также стоит рассмотреть `OffersScreenAppear` - на нем теряется до 39% посетителей, а до конца доходят порядка 47 % от начальной группы.

Если рассмотерть полный цикл, то к концу остаются всего 6-7% пользователей.



3. A1/A2/B - эксперименты:

В ходе тестирования были проведены 16 экспериментов:

- A1/A1 - 4 эксперемента (для каждого события) / уровень статистической значимости - 1%;
- A1/B - 4 эксперемента (для каждого события) / уровень статистической значимости - 5%;
- A2/B - 4 эксперемента (для каждого события) / уровень статистической значимости - 5%;
- A1+A2/B - 4 эксперемента (для каждого события) / уровень статистической значимости - 5%.

Замечание

Степень различия ключевых метрик по группам зависит от необходимой чувствительности эксперимента. A/B-тест применяют там, где степень различия между группами не больше 10%, или же подвержена колебаниям. В таких случаях точность A/A-теста в 1% — распространённый критерий.

В следствие того, что A1 и A2 (246 и 247) должны быть одинаковыми, был взят уровень статистической значимости равный 1%. По результатам A1/A2 тестирования для всех событий разница не может считаться значимой, значит, группы считаем контрольными.

Можно сказать, что разбиение на группы работает корректно.

Для A/B теста не нужна такая высокая точность, поэтому был выбран стандартный уровень значимости в 5%.

Уровень статистической значимости в 10% был бы слишком велик так как в тесте не ожидаем изменений более чем на 30%. При уровне значимости 0.1 каждый десятый раз можно получать ***ложный*** результат, поэтому стоит применить изначально выбранный нами уровень значимости 0.05.

В итоге A/A/B эксперемента значимой разницы между группами не было выявлено.