Анализ поведения пользователей мобильного приложения

Оглавление

Общая информация

Вернуться к оглавлению

Описание проекта
Вы работаете в стартапе, который продаёт продукты питания. Нужно разобраться, как ведут себя пользователи вашего мобильного приложения. Изучите воронку продаж. Узнайте, как пользователи доходят до покупки. Сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах? На каких именно?

Входные данные
Файл `/datasets/logs_exp.csv` с действиями пользователей

Цель: провести анализ поведения пользователей по результатам AAB-эксперимента по замене шрифтов в интерфейсе мобильного приложения для заказа продуктов питания.

Задачи
Шаг 1. Загрузите данные и подготовьте их к анализу
Шаг 2. Подготовьте данные

- Замените названия столбцов на удобные для вас;
- Проверьте пропуски и типы данных. Откорректируйте, если нужно;
- Добавьте столбец даты и времени, а также отдельный столбец дат;

Шаг 3. Изучите и проверьте данные

- Сколько всего событий в логе?
- Сколько всего пользователей в логе?
- Сколько в среднем событий приходится на пользователя?
- Данными за какой период вы располагаете? Найдите максимальную и минимальную дату. Постройте гистограмму по дате и времени. Можно ли быть уверенным, что у вас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать» события из прошлого — это может «перекашивать данные». Определите, с какого момента данные полные и отбросьте более старые. Данными за какой период времени вы располагаете на самом деле?
- Много ли событий и пользователей вы потеряли, отбросив старые данные?
- Проверьте, что у вас есть пользователи из всех трёх экспериментальных групп.

Шаг 4. Изучите воронку событий

- Посмотрите, какие события есть в логах, как часто они встречаются. Отсортируйте события по частоте.
- Посчитайте, сколько пользователей совершали каждое из этих событий. Отсортируйте события по числу пользователей. Посчитайте долю пользователей, которые хоть раз совершали событие.
- Предположите, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.
- По воронке событий посчитайте, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитайте 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 теряете больше всего пользователей?
- Какая доля пользователей доходит от первого события до оплаты?

Шаг 5. Изучите результаты эксперимента

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

Описание данных

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.

Изучение данных

Вернуться к оглавлению

In [1]:
```
import pandas as pd
import numpy as np
import datetime as dt
from scipy import stats as st
import math as mth
import plotly.express as px
from plotly import graph_objects as go
import warnings
warnings.filterwarnings("ignore", 'Boolean Series key will be reindexed to match DataFra
```

In [2]:
```
# чтение данных
df = pd.read_csv('logs_exp.csv', sep='\t')
```

In [3]:
```
# информация по датафрейму
df.info()
```

```
<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 [4]:
```
df.head()
```

Out[4]:

	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

Подготовка данных к анализу

Вернуться к оглавлению

In [5]:
```
# переименуем столбцы
```

```
df.columns = ['event', 'user_id', 'event_time', 'group']
```

In [6]:
```
# добавим столбцы дата-время и даты
df['date_time'] = pd.to_datetime(df['event_time'], unit='s')
df['date'] = df['date_time'].astype('datetime64[D]')
```

In [7]:
```
# проверка уникальных событий
df['event'].unique()
```

Out[7]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)
```

In [8]:
```
# описание числовых данных
df.describe()
```

Out[8]:

	user_id	event_time	group
count	2.441260e+05	2.441260e+05	244126.000000
mean	4.627568e+18	1.564914e+09	247.022296
std	2.642425e+18	1.771343e+05	0.824434
min	6.888747e+15	1.564030e+09	246.000000
25%	2.372212e+18	1.564757e+09	246.000000
50%	4.623192e+18	1.564919e+09	247.000000
75%	6.932517e+18	1.565075e+09	248.000000
max	9.222603e+18	1.565213e+09	248.000000

In [9]:
```
# проверка уникальных групп
df['group'].unique()
```

Out[9]:
```
array([246, 248, 247], dtype=int64)
```

In [10]:
```
# проверка дубликатов строк
df.duplicated().sum()
```

Out[10]:
```
413
```

In [11]:
```
# удаление дубликатов
df = df.drop_duplicates().reset_index(drop=True)
```

In [12]:
```
# минимальная дата
print(df['date'].dt.date.min())
```

```
2019-07-25
```

In [13]:
```
# максимальная дата
print(df['date'].dt.date.max())
```

```
2019-08-07
```

In [14]:
```
# информация по датафрейму
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   event       243713 non-null  object
```

```
 1   user_id     243713 non-null  int64
 2   event_time  243713 non-null  int64
 3   group       243713 non-null  int64
 4   date_time   243713 non-null  datetime64[ns]
 5   date        243713 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(3), object(1)
memory usage: 11.2+ MB
```

Выводы

Всего записей после очистки 243713.
Пропусков нет.
Явные дубликаты удалены.
Столбцы переименованы в соответствии со стилистическими нормами.
Добавлены столбцы с датой и временем.

Изучение и проверка данных

Вернуться к оглавлению

Расчитаем количество событий в логе.

In [15]:
```
print('Всего событий в логе:', len(df))
print('Уникальных событий:', df['event'].nunique())
```

```
Всего событий в логе: 243713
Уникальных событий: 5
```

Расчитаем уникальных пользователей в логе.

In [16]:
```
print('Всего уникальных пользователей:', df['user_id'].nunique())
```

```
Всего уникальных пользователей: 7551
```

Расчитаем среднее количество событий на пользователя.

In [17]:
```
# првоерим данные по количеству событий напользователя
user_group = df.groupby('user_id')['event'].agg('count')
user_group.describe()
```

Out[17]:
```
count    7551.000000
mean       32.275593
std        65.154219
min         1.000000
25%         9.000000
50%        20.000000
75%        37.000000
max      2307.000000
Name: event, dtype: float64
```

В логе есть пользователи с аномальным количеством событий. Посчитаем персентили для отсечки аномальных пользователей.

In [18]:
```
# посчитаем 95-й и 99-й перцентили количества событий на пользователя
np.percentile(user_group, [95, 99])
```

Out[18]:
```
array([ 89. , 200.5])
```

Отбросим пользователей, которые участвовали более чем в 200 событиях.

In [19]:
```
# отбросим пользователей с более чем 200 событиями
q99 =  user_group[user_group > np.percentile(user_group, 99)].index
df_sort = df.query('user_id not in @q99').reset_index(drop=True)
```

In [20]:
```
# построим гистограмму
fig = go.Figure(go.Histogram(x=df_sort.groupby('user_id')['event'].agg('count')))
fig.update_layout(title={'text':'Гистограмма количества событий на каждого пользователя'
                  xaxis_title='количество событий',
                  yaxis_title='частота')
fig.show()
```

In [21]:
```
# Сколько в среднем событий приходится на пользователя?
print('На пользователя в среднем приходится', int(df_sort.groupby('user_id')['event'].ag
```

```
На пользователя в среднем приходится 19 событий
```

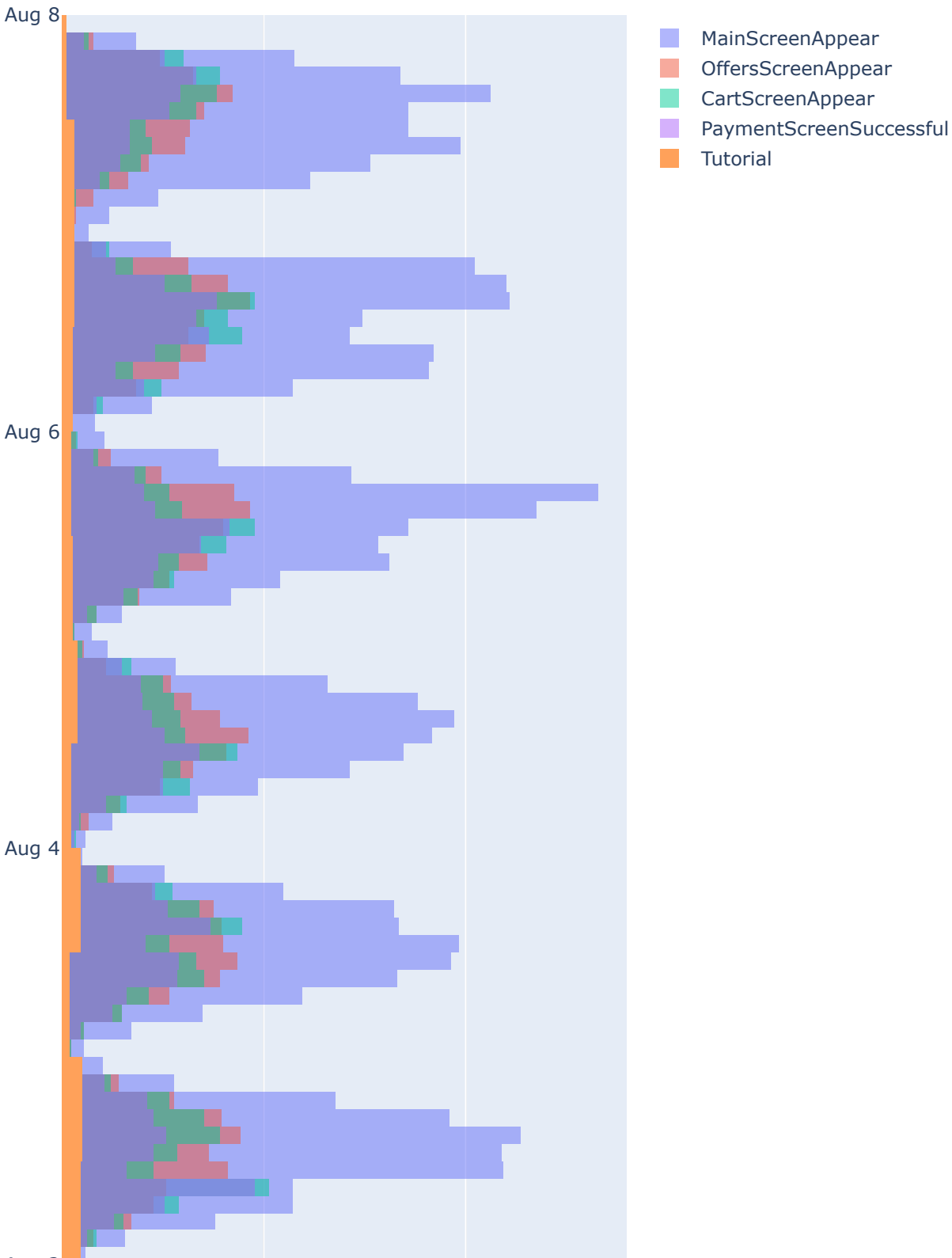
Найдем максимальную и минимальную дату. Построим гистограмму по дате и времени.

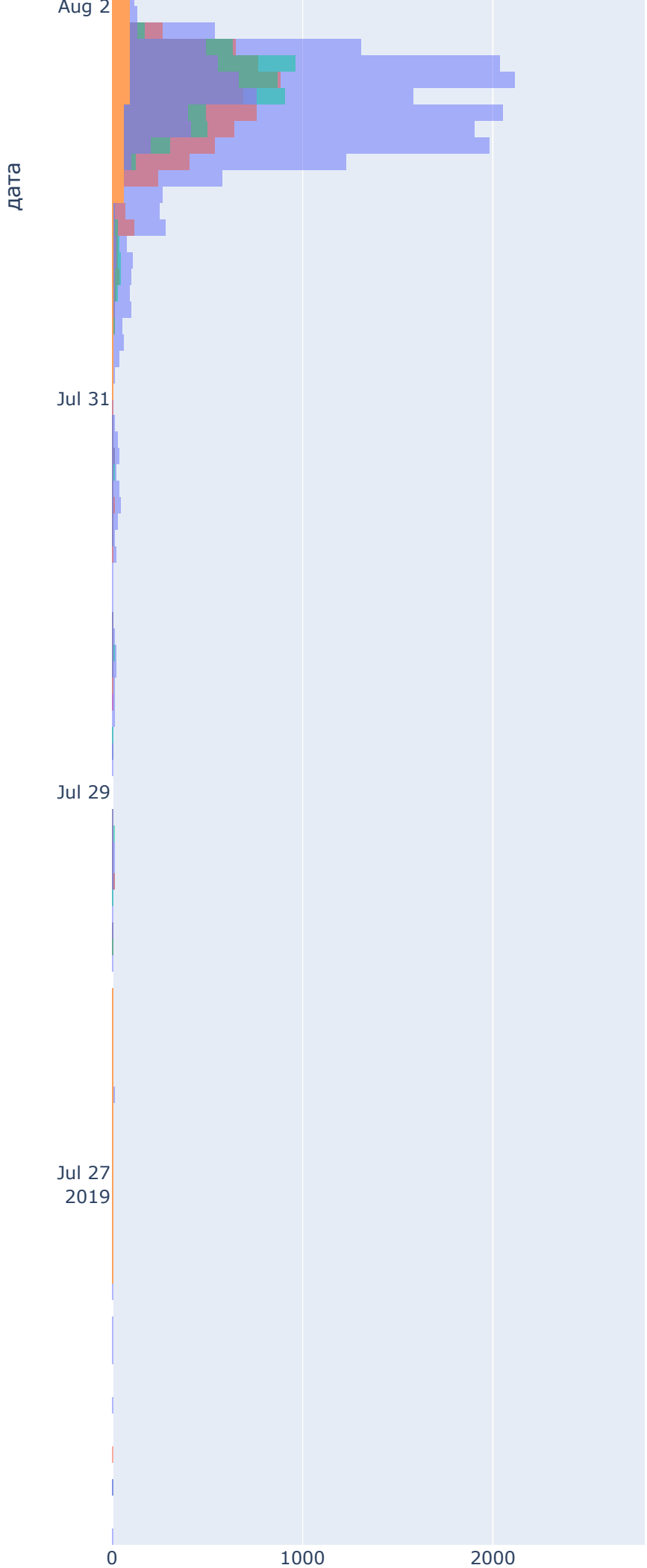
In [22]:
```
# Найдите максимальную и минимальную дату.
print('Дата начала сбора данных:', df_sort['date'].dt.date.min())
print('Дата окончания сбора данных:', df_sort['date'].dt.date.max())
print('Период сбора данных:', (df_sort['date'].max() - df_sort['date'].min() + dt.timede
```

```
Дата начала сбора данных: 2019-07-25
Дата окончания сбора данных: 2019-08-07
Период сбора данных: 14 дней
```

In [23]:
```
# построим гистограмму по дате и времени
fig = go.Figure()
fig.add_trace(go.Histogram(y=df_sort[df['event']=='MainScreenAppear']['date_time'], opac
```

```
fig.add_trace(go.Histogram(y=df_sort[df['event']=='OffersScreenAppear']['date_time'], op
fig.add_trace(go.Histogram(y=df_sort[df['event']=='CartScreenAppear']['date_time'], opac
fig.add_trace(go.Histogram(y=df_sort[df['event']=='PaymentScreenSuccessful']['date_time'
fig.add_trace(go.Histogram(y=df_sort[df['event']=='Tutorial']['date_time'], opacity=1, n
fig.update_layout(title={'text':'Гистограмма активности пользователей по дате и времени'
                  xaxis_title='количество событий',
                  yaxis_title = 'дата',
                  height=2000,
                  barmode='overlay')
fig.show()
```




Aug 2
Jul 31
Jul 29
Jul 27
2019
дата
0
1000
2000

количество событий

По гистограмме видно, что до первого августа активность пользователей была низкой. Наиболее полные данные наблюдаются после первого августа и видна активность пользователей по всем пяти событиям.

Отсечем "хвост" данных до первого августа.

In [24]:
```
df_sort = df_sort[df_sort['date'] > '2019-07-31'].reset_index(drop=True)
```

In [25]:
```
print('Из исходных данных было удалено', str(round((1-len(df_sort)/len(df))*100,1))+'%',
print('Дата начала сбора данных:', df_sort['date'].dt.date.min())
print('Дата окончания сбора данных:', df_sort['date'].dt.date.max())
print('Период сбора данных:', (df_sort['date'].max() - df_sort['date'].min() + dt.timede
print('Всего событий в логе:', len(df_sort), '(было', len(df), ')')
print('Уникальных событий:', df_sort['event'].nunique())
print('Всего уникальных пользователей:', df_sort['user_id'].nunique(), '(было', df['user
print('На пользователя в среднем приходится', int(df_sort.groupby('user_id')['event'].ag
print('Количество пользователей в первой контрольной группе равно:', df_sort[df_sort['gr
print('Количество пользователей во второй контрольной группе равно:', df_sort[df_sort['g
print('Количество пользователей в экспериментальной группе равно:', df_sort[df_sort['gro
```

```
Из исходных данных было удалено 15.2% данных
Дата начала сбора данных: 2019-08-01
Дата окончания сбора данных: 2019-08-07
Период сбора данных: 7 дней
Всего событий в логе: 206615 (было 243713 )
Уникальных событий: 5
Всего уникальных пользователей: 7458 (было 7551 )
На пользователя в среднем приходится 19 событий
Количество пользователей в первой контрольной группе равно: 2456
Количество пользователей во второй контрольной группе равно: 2491
Количество пользователей в экспериментальной группе равно: 2511
```

Выводы

Из исходного датафрейма были удалены события с неполными данными и низкой активностью пользователей. В окончательную таблицу попали данные в период с 2019-08-01 по 2019-08-07. Также были удалены пользователи с аномально высокой активностью: пользователи, которые совершили более двухсот событий за весь период исследования. Из исходной таблицы было удалено 15,2% данных.
Всего записей 206615, уникальных событий: 5.
Всего уникальных пользователей 7458. В среденем на каждого пользователя приходится 19 событий.
Пользователи разделены на три группы: две экспериментальных и одну контрольную. В каждой группе находится 2456, 2491 и 2511 пользователей соответственно.

Изучение воронки событий

Вернуться к оглавлению

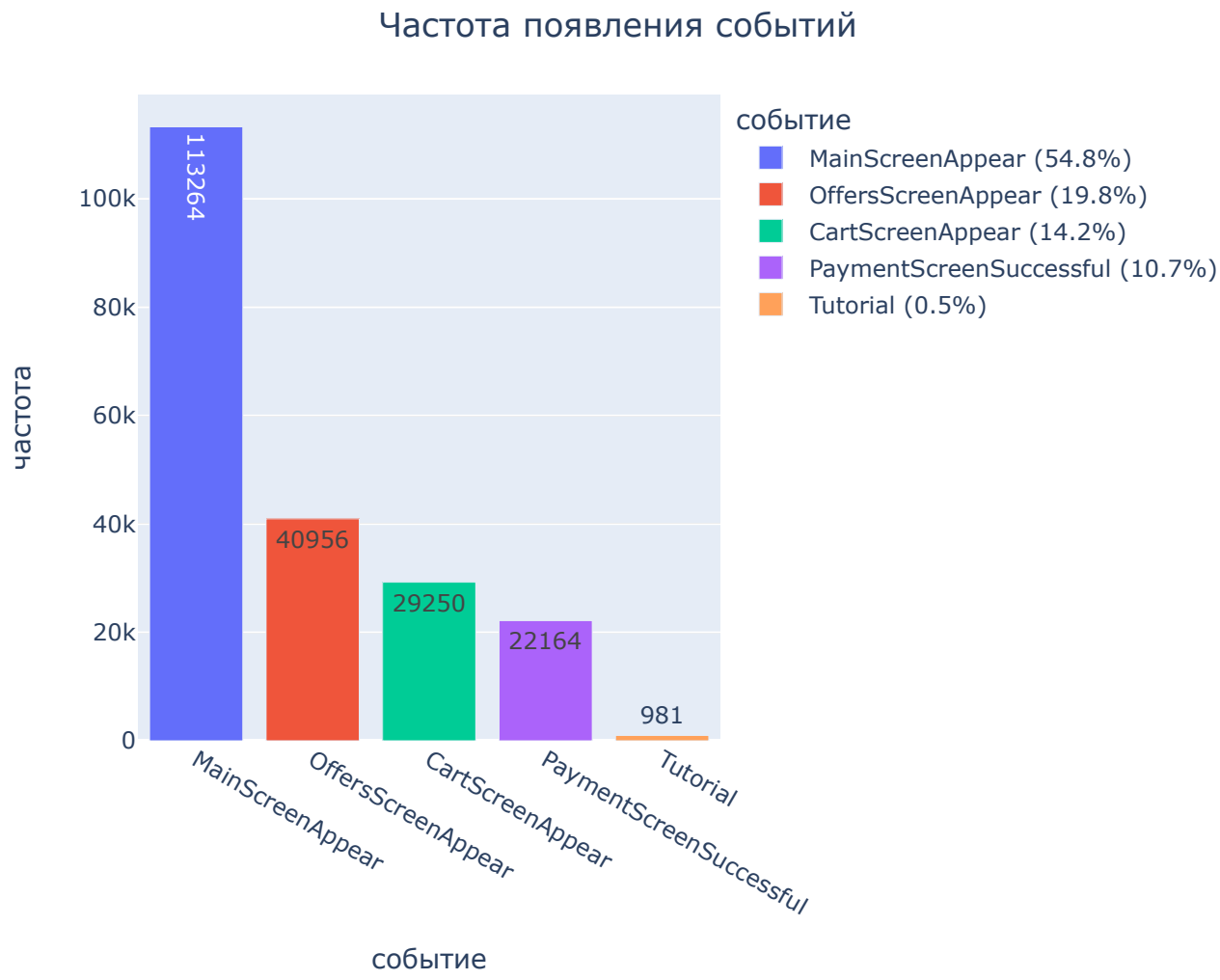
Анализ частоты появления событий

Вернуться к оглавлению

Посмотрим, какие события есть в логах, как часто они встречаются. Отсортируем события по частоте.

In [26]:
```
# создадим таблицу событий
events = df_sort.groupby('event')['user_id'].agg(['count', 'nunique']).reset_index().sor
events.columns = ['event', 'number_events', 'number_users']
```

In [27]:
```
# построим столбчатые диаграммы по частоте событий
fig = px.bar(
    events,
    x='event',
    y='number_events',
    color='event',
    labels=dict(event='событие', number_events='частота'),
    text = 'number_events'
)
fig.update_layout(
    title={'text':'Частота появления событий', 'x':0.5}
)
for trace, percent in zip(fig.data, (events['number_events']/len(df_sort)*100).round(1).
    trace.name = trace.name + ' (' + percent+ ')'
fig.show()
```



Из графика следует, что чаще всего происходило событие загрузки главной страницы приложения MainScreenAppear(54.8%) - 113264 раз.
40956 (19.8%) - OffersScreenAppear.
29250 (14.2%) - CartScreenAppear.

22164 (10.7%) - PaymentScreenSuccessful.
Реже всего клиенты пользовались Tutorial (0.5%) - 981.

Анализ активности пользователей

Вернуться к оглавлению

Посчитаем, сколько пользователей совершали каждое из этих событий. Отсортируем события по числу пользователей. Посчитаем долю пользователей, которые хоть раз совершали событие.

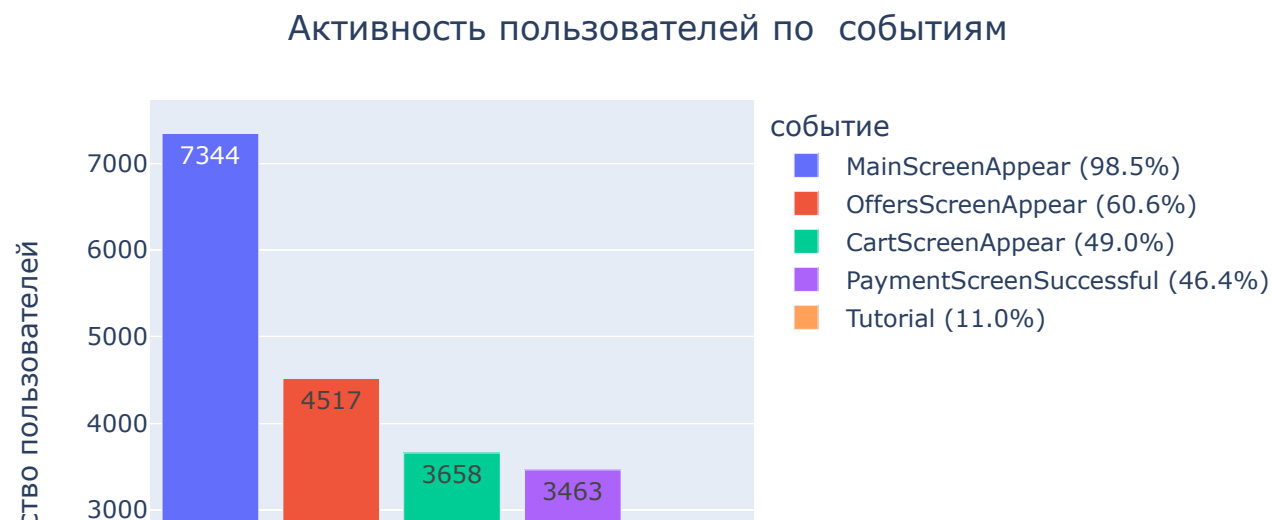
In [28]:
```
# отсортируем таблицу событий по пользователям и посчитаем долю пользователей
events = events.sort_values(by='number_users', ascending=False).reset_index(drop=True)
events['users_rate'] = round(events['number_users']/df_sort['user_id'].nunique()*100, 1)
```

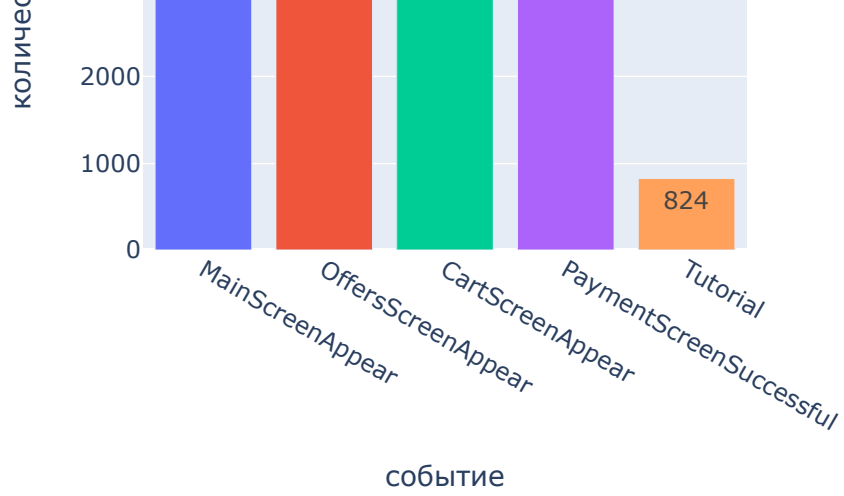
In [29]:
```
events
```

Out[29]:

	event	number_events	number_users	users_rate
0	MainScreenAppear	113264	7344	98.5
1	OffersScreenAppear	40956	4517	60.6
2	CartScreenAppear	29250	3658	49.0
3	PaymentScreenSuccessful	22164	3463	46.4
4	Tutorial	981	824	11.0

In [30]:
```
# построим столбчатые диаграммы по действиям пользователей
fig = px.bar(
    events,
    x='event',
    y='number_users',
    color='event',
    labels=dict(event='событие', number_users='количество пользователей'),
    text = 'number_users'
)
fig.update_layout(
    title={'text':'Активность пользователей по  событиям', 'x':0.5}
)
for trace, percent in zip(fig.data, events['users_rate'].astype('str') + '%' ) :
    trace.name = trace.name + ' (' + percent+ ')'
fig.show()
```





Из графика следует, что 7344 пользователей хотя бы раз запускали главный экран приложения. Это составляет 98,5% всех пользователей. Активность остальных 1,5% пользователей не была зарегестрирована, что, возможно, связано с ошибками подключения или регистрации пользователей в логах.
4517 пользователей (60,6%) октрывали каталог товаров.
3658 пользователей (49%) переходили в корзину.
3463 пользователя (46,6%) упешно оплатили заказ.
824 пользователя (11%) хотя бы раз открывали руководство пользователя.

Анализ воронки событий

Вернуться к оглавлению

Предположим, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.

Предположим, что пользователь последовательно:

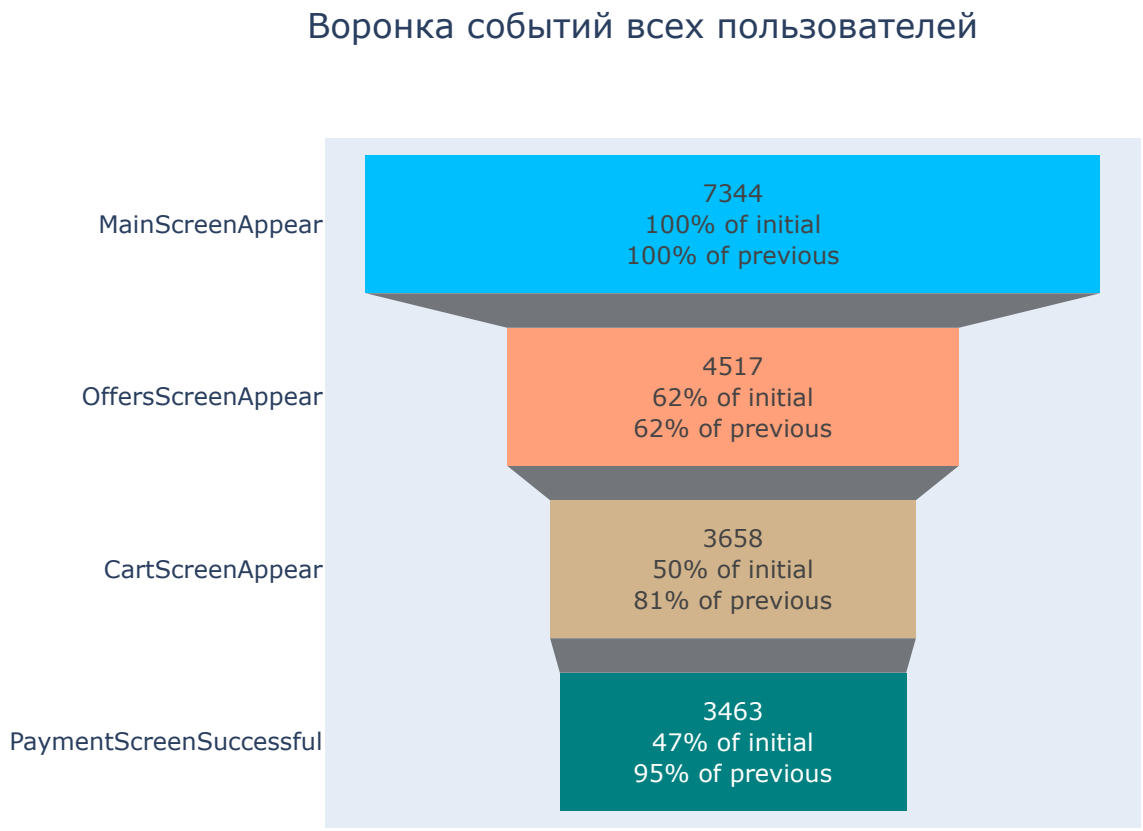
- открывает главный экран приложения;
- изучает каталог товаров, добавляет их в корзину;
- переходит в корзину для оформления и оплаты заказа;
- переходит на экран успешной оплаты заказа.

Руководство пользователя не будем учитывать при расчете воронки, т.к. это необязательное событие.

По воронке событий посчитаем, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитаем 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.

In [31]:
```
# построим воронку
fig = go.Figure(
    go.Funnel(
        y=events.loc[:3]['event'],
        x=events.loc[:3]['number_users'],
        textinfo = "value+percent previous+percent initial",
        marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"]}
    )
```

```
)
fig.update_layout(
    title={'text':'Воронка событий всех пользователей', 'x':0.5}
)
fig.show()
```



In [32]:
```
# функция расчета воронки
def funnel_calc(row):
    row['funnel_step'] = row['number_users'].shift(1)
    row['funnel_step'] = row['funnel_step'].fillna(row.loc[0]['number_users'])
    row['funnel_step'] = round(row['number_users'] / row['funnel_step'] * 100, 1)
    row['funnel_first'] = round(row['number_users'] / row.loc[0]['number_users'] * 100,
    return row

funnel_calc(events);
```

In [33]:
```
# результат расчета воронки
events[0:4]
```

Out[33]:

	event	number_events	number_users	users_rate	funnel_step	funnel_first
0	MainScreenAppear	113264	7344	98.5	100.0	100.0
1	OffersScreenAppear	40956	4517	60.6	61.5	61.5
2	CartScreenAppear	29250	3658	49.0	81.0	49.8
3	PaymentScreenSuccessful	22164	3463	46.4	94.7	47.2

На первом шаге теряется 38% пользователей. Пользователи не переходят с главного экрана в каталог. Возможно, это связано с проблемами интерфейса стартовой страницы.

81% пользователей, изучивших каталог, переходят к оформлению товара.
95% пользователей, оформиших заказ, успешно оплачивает заказ.
От запуска стратовой страницы до оформления заказа доходит 47% пользователей.

Выводы

Вернуться к оглавлению

Чаще всего происходило событие загрузки главной страницы приложения MainScreenAppear(54.8%) - 113264 раз.
40956 (19.8%) - OffersScreenAppear.
29250 (14.2%) - CartScreenAppear.
22164 (10.7%) - PaymentScreenSuccessful.
Реже всего клиенты пользовались Tutorial (0.5%) - 981.

7344 пользователей хотя бы раз запускали главный экран приложения. Это составляет 98,5% всех пользователей. Активность остальных 1,5% пользователей не была зарегестрирована, что, возможно, связано с ошибками подключения или регистрации пользователей в логах.
4517 пользователей (60,6%) открывали каталог товаров.
3658 пользователей (49%) переходили в корзину.
3463 пользователя (46,6%) упешно оплатили заказ.
824 пользователя (11%) хотя бы раз открывали руководство пользователя.

Возможная последовательность действий пользователей для оформления заказа:

- открывает главный экран приложения;
- изучает каталог товаров, добавляет их в корзину;
- переходит в корзину для оформления и оплаты заказа;
- переходит на экран успешной оплаты заказа.

Руководство пользователя не будем учитывать при расчете воронки.

Анализ воронки показал следующие результаты:

- На первом шаге теряется 38% пользователей. Пользователи не переходят с главного экрана в каталог. Возможно, это связано с проблемами интерфейса стартовой страницы.
- 81% пользователей, изучивших каталог, переходят к оформлению товара.
- 95% пользователей успешно оплачивает заказ.
- От запуска стратовой страницы до оформления заказа доходит 47% пользователей.

Изучение результатов эксперимента

Вернуться к оглавлению

Анализ экспериментальных групп

Вернуться к оглавлению

Сколько пользователей в каждой экспериментальной группе?

In [34]:
```
# создадим группы пользователей
gr246 = df_sort[(df_sort['group'] == 246) &  (df_sort['event'] != 'Tutorial')]
gr247 = df_sort[(df_sort['group'] == 247) &  (df_sort['event'] != 'Tutorial')]
gr248 = df_sort[(df_sort['group'] == 248) &  (df_sort['event'] != 'Tutorial')]
```

In [35]:
```
print('Количество пользователей в контрольной группе №246 равно:', gr246['user_id'].nuni
print('Количество пользователей в контрольной группе №247 равно:', gr247['user_id'].nuni
print('Количество пользователей в экспериментальной группе №248 равно:', gr248['user_id'
```

```
Количество пользователей в контрольной группе №246 равно: 2455
Количество пользователей в контрольной группе №247 равно: 2490
Количество пользователей в экспериментальной группе №248 равно: 2509
```

In [36]:
```
# проверка того, что пользователи не входят в другие группы
df_sort.groupby('user_id', as_index=False)['group'].nunique().query('group > 1').count()
```

Out[36]:
```
user_id    0
group      0
dtype: int64
```

Без учета события Tutorial в каждой группе количество пользователей равно:

- в группе 246: 2455
- в группе 247: 2490
- в группе 248: 2509

Каждый пользователь находится в одной группе и не входит в другие группы.

Анализ результатов A/A-эксперимента

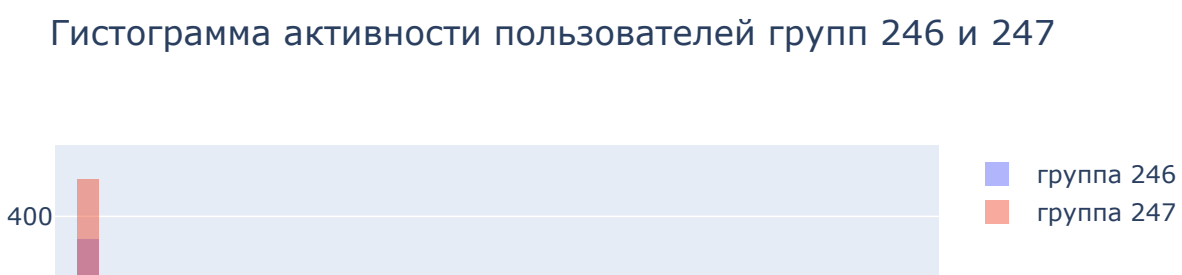
Вернуться к оглавлению

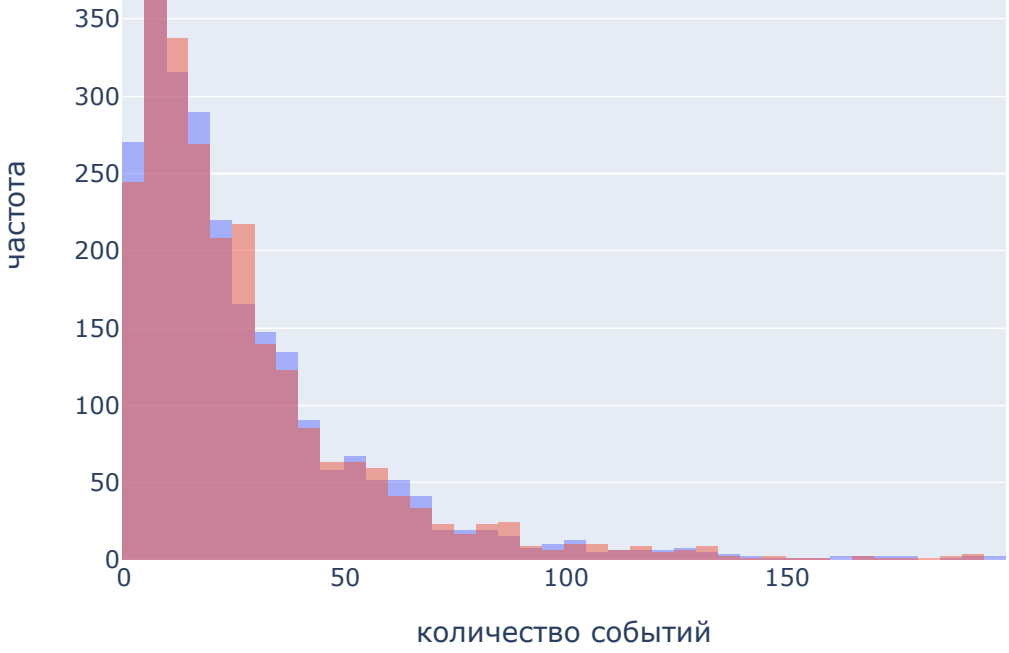
Есть 2 контрольные группы для A/A-эксперимента, чтобы проверить корректность всех механизмов и расчётов. Проверим, находят ли статистические критерии разницу между выборками 246 и 247.

In [37]:
```
# проверим соотношение пользователей в гр. 246 и 247
print('Соотношение пользователей группы 246 к группе 247 равно:',
      round(gr246['user_id'].nunique()/gr247['user_id'].nunique(), 3))
```

```
Соотношение пользователей группы 246 к группе 247 равно: 0.986
```

In [38]:
```
# построим гистограммы событий по каждой группе
fig = go.Figure()
fig.add_trace(go.Histogram(x=gr246.groupby('user_id')['event'].agg('count'), opacity=0.5
fig.add_trace(go.Histogram(x=gr247.groupby('user_id')['event'].agg('count'), opacity=0.5
fig.update_layout(title={'text':'Гистограмма активности пользователей групп 246 и 247',
                  xaxis_title='количество событий',
                  yaxis_title = 'частота',
                  barmode='overlay')
fig.show()
```





Чтобы проверить коорректность A/A-эксперимента посчитаем статистическую значимость различий между событиями в группах 246 и 247.
Введем нулевую и альтернативные гипотезы:

- H0: Различий в среднем количестве событий между группами нет
- H1: Различия в среднем количестве событий между группами есть.

In [39]:
```
# проверим статичстическую значимость событий групп 246 и 247
alpha = .05 # критический уровень статистической значимости

results = st.ttest_ind(
    gr246.groupby('user_id')['event'].agg('count'),
    gr247.groupby('user_id')['event'].agg('count'),
    equal_var = False
)

print('p-значение:', results.pvalue.round(5))

if results.pvalue < alpha:
    print("Отвергаем нулевую гипотезу, статистически значимые различия есть")
else:
    print("Не получилось отвергнуть нулевую гипотезу, статистически значимых различий не
```

```
p-значение: 0.95934
Не получилось отвергнуть нулевую гипотезу, статистически значимых различий нет
```

Выберем самое популярное событие. Посчитаем число пользователей, совершивших это событие в каждой из контрольных групп.

In [40]:
```
# построим воронки для групп 246 и 247
funnel_246 = gr246.groupby('event', as_index=False)['user_id'].agg(['count', 'nunique'])
funnel_246.columns = ['event', 'number_events', 'number_users']
funnel_247 = gr247.groupby('event', as_index=False)['user_id'].agg(['count', 'nunique'])
funnel_247.columns = ['event', 'number_events', 'number_users']
```

In [41]:
```
funnel_calc(funnel_246)
```

Out[41]:

	event	number_events	number_users	funnel_step	funnel_first

0	MainScreenAppear	36082	2423	100.0	100.0
1	OffersScreenAppear	13267	1514	62.5	62.5
2	CartScreenAppear	10082	1238	81.8	51.1
3	PaymentScreenSuccessful	7676	1172	94.7	48.4

In [42]:
```
funnel_calc(funnel_247)
```

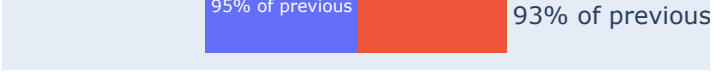
Out[42]:

	event	number_events	number_users	funnel_step	funnel_first
0	MainScreenAppear	37835	2454	100.0	100.0
1	OffersScreenAppear	13516	1498	61.0	61.0
2	CartScreenAppear	9415	1216	81.2	49.6
3	PaymentScreenSuccessful	7199	1136	93.4	46.3

In [43]:
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'группа 246',
    y = funnel_246['event'],
    x = funnel_246['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))

fig.add_trace(go.Funnel(
    name = 'группа 247',
    y = funnel_247['event'],
    x = funnel_247['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))
fig.update_layout(
    title={'text':'Воронка событий контрольных групп 246 и 247', 'x':0.5}
)
fig.show()
```



Самое популярное событие - MainScreenAppear.
2423 пользователя из группы 246 совершили событие MainScreenAppear.
2454 пользователя из группы 247 совершили событие MainScreenAppear.

Посчитаем долю пользователей, совершивших это событие. Проверим, будет ли отличие между группами статистически достоверным. Проделаем то же самое для всех других событий. Можно ли сказать, что разбиение на группы работает корректно?

In [44]:
```
# проверим соотношение пользователей на каждом этапе воронки
print('Соотношение пользователей контрольных групп 247 и 246:')
round(funnel_247['number_users']/funnel_246['number_users'] *100, 1)
```

Out[44]:
```
Соотношение пользователей контрольных групп 247 и 246:
0    101.3
1     98.9
2     98.2
3     96.9
Name: number_users, dtype: float64
```

In [45]:
```
# посчитаем долю пользователей на каждом этапе
funnel_246['users_rate'] = funnel_246['number_users'] / gr246['user_id'].nunique()
funnel_247['users_rate'] = funnel_247['number_users'] / gr247['user_id'].nunique()
```

In [46]:
```
funnel_246
```

Out[46]:

	event	number_events	number_users	funnel_step	funnel_first	users_rate
0	MainScreenAppear	36082	2423	100.0	100.0	0.986965
1	OffersScreenAppear	13267	1514	62.5	62.5	0.616701
2	CartScreenAppear	10082	1238	81.8	51.1	0.504277
3	PaymentScreenSuccessful	7676	1172	94.7	48.4	0.477393

In [47]:
```
funnel_247
```

Out[47]:

	event	number_events	number_users	funnel_step	funnel_first	users_rate
0	MainScreenAppear	37835	2454	100.0	100.0	0.985542
1	OffersScreenAppear	13516	1498	61.0	61.0	0.601606
2	CartScreenAppear	9415	1216	81.2	49.6	0.488353
3	PaymentScreenSuccessful	7199	1136	93.4	46.3	0.456225

In [48]:
```
# проверка гипотез о равенстве долей
def z_test(group1, group2, i, alpha):
    # alpha = .05# критический уровень статистической значимости

    successes1 = group1.loc[i]['number_users'] # количество пользователей на шаге в груп
    successes2 = group2.loc[i]['number_users'] # количество пользователей на шаге в груп
    trials1 = group1.loc[i]['number_users'] / group1.loc[i]['users_rate'] # количество п
```

```
    trials2 = group2.loc[i]['number_users'] / group2.loc[i]['users_rate'] # количество п

    # пропорция успехов в первой группе:
    p1 = successes1/trials1

    # пропорция успехов во второй группе:
    p2 = successes2/trials2

    # пропорция успехов в комбинированном датасете:
    p_combined = (successes1 + successes2) / (trials1 + trials2)

    # 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 / trials1 + 1 / trials2)
    )

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

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

    print('Событие', group1.loc[i]['event']+',', 'p-значение: ', p_value.round(5))

    if p_value < alpha:
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')
    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разны
        )
    print()
```

In [49]:

```
print('Проверка гипотез о равенстве долей групп 246 и 247')
print()
print('Введем нулевую и альтернативную гипотезу:')
print('H0: пропорции пользователей в группах равны')
print('H1: пропорции пользователей в группах различны')
print()

for i in funnel_246.index:
    z_test(funnel_246, funnel_247, i, 0.05)
```

```
Проверка гипотез о равенстве долей групп 246 и 247

Введем нулевую и альтернативную гипотезу:
H0: пропорции пользователей в группах равны
H1: пропорции пользователей в группах различны

Событие MainScreenAppear, p-значение:  0.66743
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие OffersScreenAppear, p-значение:  0.27677
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие CartScreenAppear, p-значение:  0.26282
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие PaymentScreenSuccessful, p-значение:  0.13574
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Проверка критериев A/A-теста показала следующее:

- Количество пользователей в различных группах достигает 3,1%;
- Для всех групп фиксируют и отправляют в системы аналитики данные об одном и том же;
- Различие частоты событий по группам не превышает 5% и не имеет статистической значимости;
- Попавший в одну из групп посетитель остаётся в этой группе до конца теста.

Ни для одного из событий разница не оказалось значимой и обе группы 246 и 247 можно считать контрольными. A/A эксперимент прошел успешно.

Анализ результатов A/B-эксперимента

Вернуться к оглавлению

Аналогично посчитаем долю пользователей с группой с изменённым шрифтом и проверим, будет ли отличие между группами статистически достоверным. Сравним результаты с каждой из контрольных групп в отдельности по каждому событию. Сравним результаты с объединённой контрольной группой. Какие выводы из эксперимента можно сделать?

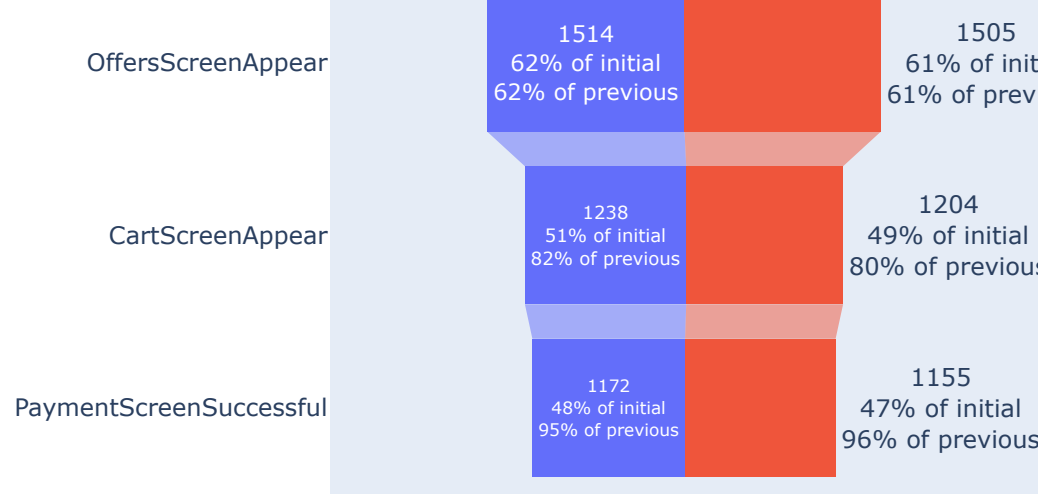
In [50]:
```
# посчитаем воронку для группы 248
funnel_248 = gr248.groupby('event', as_index=False)['user_id'].agg(['count', 'nunique'])
funnel_248.columns = ['event', 'number_events', 'number_users']
funnel_calc(funnel_248)
funnel_248['users_rate'] = funnel_248['number_users'] / gr248['user_id'].nunique()
```

Построим воронку групп 246 и 248.
И проверим гипотезы о равенстве долей групп 246 и 248.

In [51]:
```
# построим воронку
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'группа 246',
    y = funnel_246['event'],
    x = funnel_246['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))

fig.add_trace(go.Funnel(
    name = 'группа 248',
    y = funnel_248['event'],
    x = funnel_248['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))
fig.update_layout(
    title={'text':'Воронка событий групп 246 и 248', 'x':0.5}
)
fig.show()
```



In [52]:
```
# проверим соотношение пользователей на каждом этапе воронки
print('Соотношение пользователей групп 248 и 246:')
round(funnel_248['number_users']/funnel_246['number_users'] *100, 1)
```

Out[52]:
```
Соотношение пользователей групп 248 и 246:
0    101.8
1     99.4
2     97.3
3     98.5
Name: number_users, dtype: float64
```

In [53]:
```
print('Проверка гипотез о равенстве долей групп 246 и 248')
print()
print('Введем нулевую и альтернативную гипотезу:')
print('H0: пропорции пользователей в группах равны')
print('H1: пропорции пользователей в группах различны')
print()

for i in funnel_246.index:
    z_test(funnel_246, funnel_248, i, 0.05)
```

```
Проверка гипотез о равенстве долей групп 246 и 248

Введем нулевую и альтернативную гипотезу:
H0: пропорции пользователей в группах равны
H1: пропорции пользователей в группах различны

Событие MainScreenAppear, p-значение:  0.28147
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие OffersScreenAppear, p-значение:  0.22374
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие CartScreenAppear, p-значение:  0.08551
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие PaymentScreenSuccessful, p-значение:  0.22876
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Значимой разницы между контрольной группой 246 и экспериментальной группой 248 не выявлено.

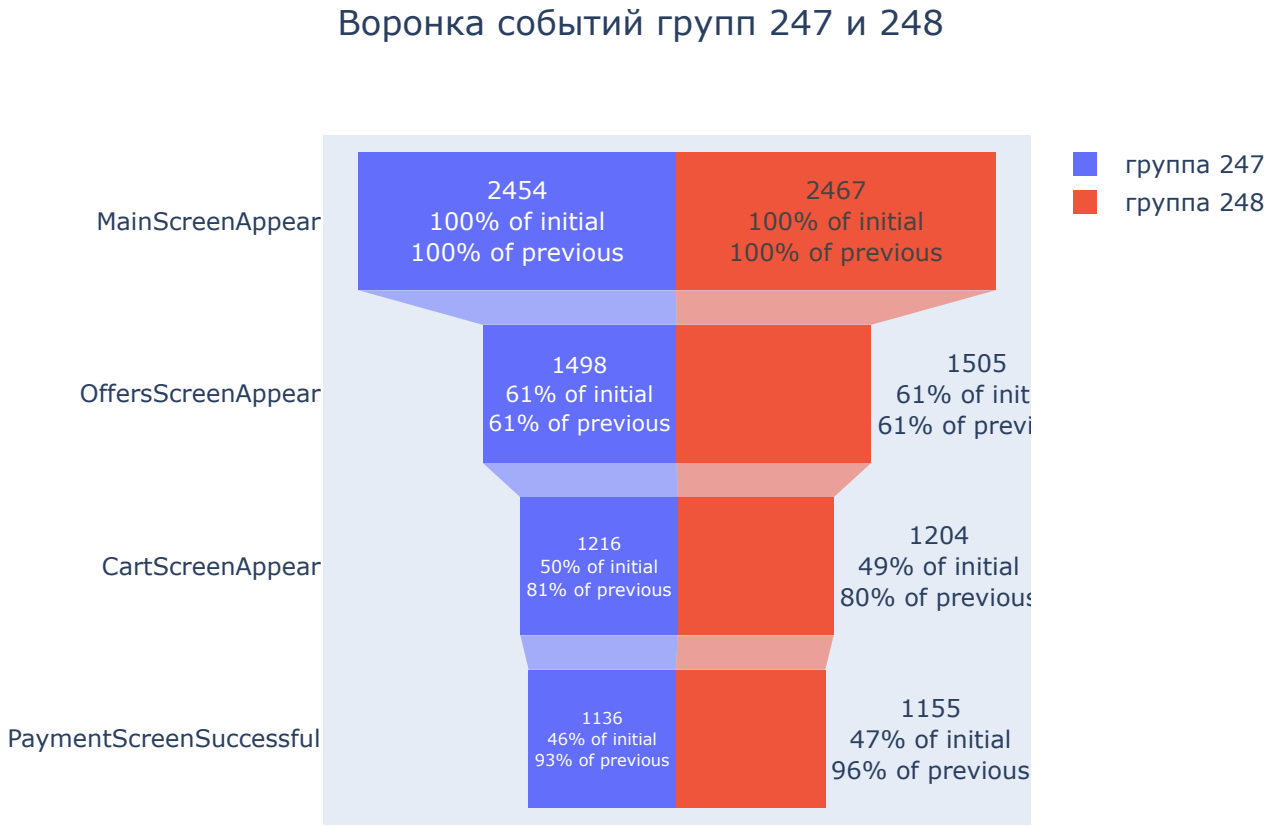
Построим воронку групп 247 и 248.

И проверим гипотезы о равенстве долей групп 247 и 248.

In [54]:
```
#  построим воронку
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'группа 247',
    y = funnel_247['event'],
    x = funnel_247['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))

fig.add_trace(go.Funnel(
    name = 'группа 248',
    y = funnel_248['event'],
    x = funnel_248['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))
fig.update_layout(
    title={'text':'Воронка событий групп 247 и 248', 'x':0.5}
)
fig.show()
```



In [55]:
```
# проверим соотношение пользователей на каждом этапе воронки
print('Соотношение пользователей групп 248 и 247:')
round(funnel_248['number_users']/funnel_247['number_users'] *100, 1)
```

Out[55]:
```
Соотношение пользователей групп 248 и 247:
0    100.5
1    100.5
2     99.0
```

```
3    101.7
Name: number_users, dtype: float64
```

In [56]:
```
print('Проверка гипотез о равенстве долей групп 247 и 248')
print()
print('Введем нулевую и альтернативную гипотезу:')
print('H0: пропорции пользователей в группах равны')
print('H1: пропорции пользователей в группах различны')
print()

for i in funnel_247.index:
    z_test(funnel_247, funnel_248, i, 0.05)
```

```
Проверка гипотез о равенстве долей групп 247 и 248

Введем нулевую и альтернативную гипотезу:
H0: пропорции пользователей в группах равны
H1: пропорции пользователей в группах различны

Событие MainScreenAppear, p-значение:  0.51511
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие OffersScreenAppear, p-значение:  0.89857
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие CartScreenAppear, p-значение:  0.54855
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие PaymentScreenSuccessful, p-значение:  0.77016
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Значимой разницы между контрольной группой 247 и экспериментальной группой 248 не выявлено.

Построим объединенную контрольную группу

In [57]:
```
funnel_union = funnel_246
funnel_union['number_events'] = funnel_union['number_events'] + funnel_247['number_event
funnel_union['number_users'] = funnel_union['number_users'] + funnel_247['number_users']
funnel_calc(funnel_union)
funnel_248['users_rate'] = funnel_union['number_users'] / (gr246['user_id'].nunique() +
```

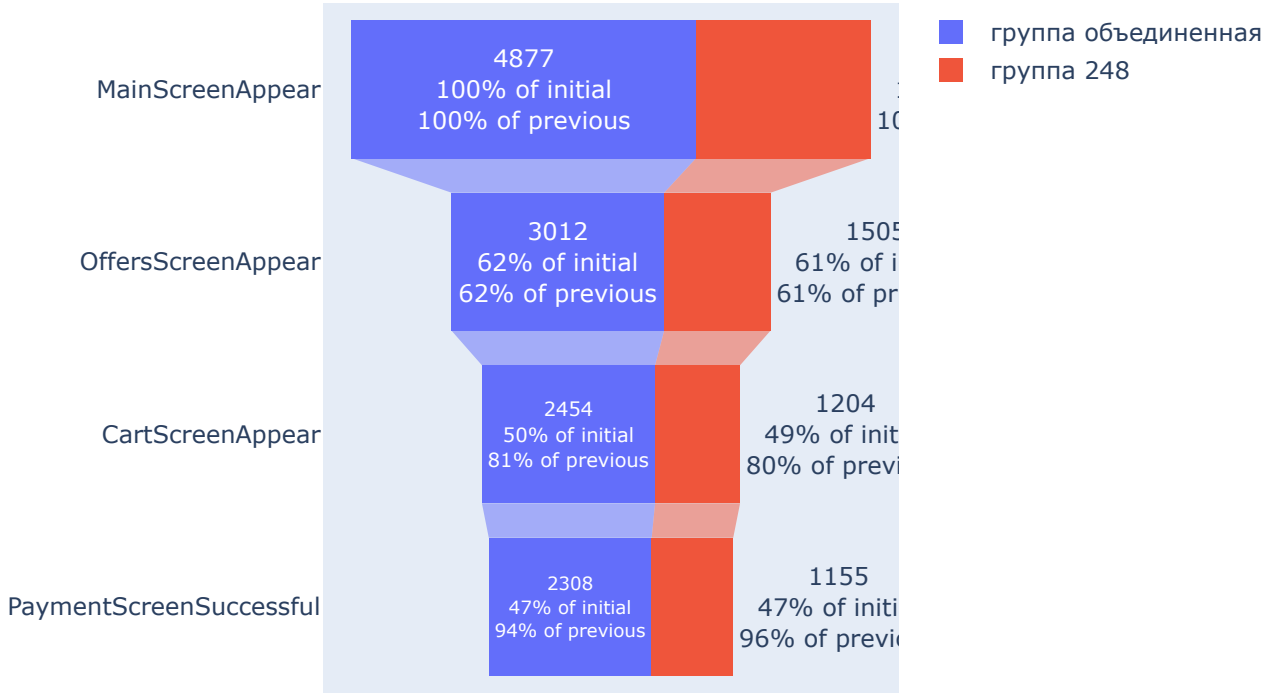
Построим воронку объединенной группы и 248.

И проверим гипотезы о равенстве долей объединенной группы и 248.

In [58]:
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'группа объединенная',
    y = funnel_union['event'],
    x = funnel_union['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))

fig.add_trace(go.Funnel(
    name = 'группа 248',
    y = funnel_248['event'],
    x = funnel_248['number_users'],
    textinfo = "value+percent previous+percent initial"
    ))
fig.update_layout(
    title={'text':'Воронка событий объединенной группы и 248', 'x':0.5}
)
fig.show()
```



In [59]:
```
# проверим соотношение пользователей на каждом этапе воронки
print('Соотношение пользователей групп 248 и объединенной группы:')
round(funnel_248['number_users']/funnel_union['number_users'] *100, 1)
```

Out[59]:
```
Соотношение пользователей групп 248 и объединенной группы:
0    50.6
1    50.0
2    49.1
3    50.0
Name: number_users, dtype: float64
```

In [60]:
```
print('Проверка гипотез о равенстве долей объединенной группы и 248')
print()
print('Введем нулевую и альтернативную гипотезу:')
print('H0: пропорции пользователей в группах равны')
print('H1: пропорции пользователей в группах различны')
print()

for i in funnel_union.index:
    z_test(funnel_union, funnel_248, i, 0.05)
```

```
Проверка гипотез о равенстве долей объединенной группы и 248

Введем нулевую и альтернативную гипотезу:
H0: пропорции пользователей в группах равны
H1: пропорции пользователей в группах различны

Событие MainScreenAppear, p-значение:  0.79859
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие OffersScreenAppear, p-значение:  0.5271
```

```
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие CartScreenAppear, p-значение:  0.51876
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие PaymentScreenSuccessful, p-значение:  0.38777
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Значимой разницы между объединенной контрольной группой и экспериментальной группой не выявлено.

Выводы

Без учета события Tutorial в каждой группе количество пользователей равно:

- в группе 246: 2455
- в группе 247: 2490
- в группе 248: 2509

Каждый пользователь находится в одной группе и не входит в другие группы.

Проверка критериев A/A-теста показала следующее:

- Количество пользователей в различных группах достигает 3,1%;
- Для всех групп фиксируют и отправляют в системы аналитики данные об одном и том же;
- Различие ключевых метрик по группам не превышает 5% и не имеет статистической значимости;
- Попавший в одну из групп посетитель остаётся в этой группе до конца теста.

Ни для одного из событий разница не оказалось значимой и обе группы 246 и 247 можно считать контрольными. A/A эксперимент прошел успешно.

Сравнение результатов с каждой из контрольных групп в отдельности и результатов с объединённой контрольной группой по каждому событию не показало статистически занчимых различий. Из чего следует, что введение новых шрифтов в интерфейс программы не повлияло на конверсию.

При проверке статистических гипотез был принят уровень значимости 0,05. Всего было сделано 17 проверок:

- одна проверка событий A/A эксперимента
- четыре проверки долей пользователей по событиям контрольных групп
- 12 проверок долей пользователей по событиям контрольных, кобинированных и экспериментальных групп.

Минимальное p-значение при проверке гипотез было 0,086 > 0,05, поэтому оставим уровень значимости равен 0,05, т.к. внедрение поправок на множественную проверку не приведет к увеличению значения уровня статистической значимости.

Выводы

С целью анализа поведения пользователей по результатам AAB-эксперимента по замене шрифтов в интерфейсе мобильного приложения для заказа продуктов питания были сделаны следующие задачи:

- Подготовлены исходные данные: заменены названия столбцов, проверены пропуски и типы данных, добавлены столбцы даты и времени
- Изучены и проверены данные: посчитано количество событий в логе, посчитано количество пользователей в логе, посчитано среднее количество событий на пользователя. Построены гистограммы событий по дате и времени. Определено, с какого момента данные полные и отброшены старые данные. Оценен период анализа, найдены максимальная и минимальная дата эксперимента. Посчитано, сколько событий и пользователей потеряли, отбросив старые данные. Проверено, что пользователи есть во всех трёх экспериментальных группах.
- Изучена воронка событий: изучены события в логах и посчитано как часто они встречаются. Посчитано, сколько пользователей совершали каждое из этих событий. Посчитана доля пользователей, которые хоть раз совершали событие. Предположено, в каком порядке происходят события. По воронке событий посчитано, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). Посчитано на каком шаге теряется больше всего пользователей. Посчитано какая доля пользователей доходит от первого события до оплаты.
- Изучены результаты эксперимента: посчитано количество пользователей в каждой экспериментальной группе. Проверены статистические критерии между выборками 246 и 247. Посчитаны доли пользователей, совершивших каждое событие. Проверено, будет ли отличие между группами статистически достоверным.
- Проверены статистические критерии с группой с изменённым шрифтом. Проведено сравните результатов с каждой из контрольных групп в отдельности по каждому событию. Проведено сравнение результатов с объединённой контрольной группой. Выбран уровень значимости при проверке статистических гипотез. Посчитано количество проверок статистических гипотез.
- Сделаны выводы.

Из исходного датафрейма были удалены события с неполными данными и низкой активностью пользователей. В окончательную таблицу попали данные в период с 2019-08-01 по 2019-08-07. Также были удалены пользователи с аномально высокой активностью: пользователи, которые совершили более двухсот событий за весь период исследования. Из исходной таблицы было удалено 15,2% данных. Всего записей 206615, уникальных событий: 5. Всего уникальных пользователей 7458. В среденем на каждого пользователя приходится 19 событий. Пользователи разделены на три группы: две экспериментальных и одну контрольную. В каждой группе находится 2456, 2491 и 2511 пользователей соответственно.

Чаще всего происходило событие загрузки главной страницы приложения MainScreenAppear(54.8%) - 113264 раз. 40956 (19.8%) - OffersScreenAppear. 29250 (14.2%) - CartScreenAppear. 22164 (10.7%) - PaymentScreenSuccessful. Реже всего клиенты пользовались Tutorial (0.5%) - 981.

7344 пользователей хотя бы раз запускали главный экран приложения. Это составляет 98,5% всех пользователей. Активность остальных 1,5% пользователей не была зарегестрирована, что, возможно, связано с ошибками подключения или регистрации пользователей в логах. 4517 пользователей (60,6%) открывали каталог товаров. 3658 пользователей (49%) переходили в корзину. 3463 пользователя (46,6%) упешно оплатили заказ. 824 пользователя (11%) хотя бы раз открывали руководство пользователя.

Возможная последовательность действий пользователей для оформления заказа:

- открывает главный экран приложения;
- изучает каталог товаров, добавляет их в корзину;

- переходит в корзину для оформления и оплаты заказа;
- переходит на экран успешной оплаты заказа.
  Руководство пользователя не учитывается при расчете воронки.

Анализ воронки показал следующие результаты:

На первом шаге теряется 38% пользователей. Пользователи не переходят с главного экрана в каталог. Возможно, это связано с проблемами интерфейса стартовой страницы. 81% пользователей, изучивших каталог, переходят к оформлению товара. 95% пользователей успешно оплачивает заказ. От запуска стратовой страницы до офрмления заказа доходит 47% пользователей.

Без учета события Tutorial в каждой группе количество пользователей равно:

- в группе 246: 2455
- в группе 247: 2490
- в группе 248: 2509
  Каждый пользователь находится в одной группе и не входит в другие группы.

Проверка критериев A/A-теста показала следующее:

Количество пользователей в различных группах достигает 3,1%; Для всех групп фиксируют и отправляют в системы аналитики данные об одном и том же; Различие ключевых метрик по группам не превышает 5% и не имеет статистической значимости; Попавший в одну из групп посетитель остаётся в этой группе до конца теста. Ни для одного из событий разница не оказалось значимой и обе группы 246 и 247 можно считать контрольными. A/A эксперимент прошел успешно.

В рамках анализа A/B-теста сравнение результатов с каждой из контрольных групп в отдельности и результатов с объединённой контрольной группой по каждому событию не показало статистически занчимых различий. Из чего следует, что введение новых шрифтов в интерфейс программы не повлияло на конверсию.

При проверке статистических гипотез был принят уровень значимости 0,05. Всего было сделано 17 проверок:

- одна проверка событий A/A эксперимента
- четыре проверки долей пользователей по событиям контрольных групп
- 12 проверок долей пользователей по событиям контрольных, кобинированных и экспериментальных групп.

Минимальное p-значение при проверке гипотез было 0,086 > 0,05, поэтому оставим уровень значимости равен 0,05, т.к. внедрение поправок на множественную проверку не приведет к увеличению значения уровня статистической значимости.

Из результатов анализа A/A/B-эксперимента следует, что изменение шрифта в приложении не повлияло на поведение пользователей.