Исследование поведения пользователей мобильного приложения по продаже продуктов питания

Содержание

Команда стартапа, который продаёт продукты питания с помощью мобильного приложения, заинтересована в исследовании поведения пользователей приложения:

1. Требуется изучить воронку продаж, узнать, как пользователи доходят до покупки, сколько их доходит до покупки, а сколько — «застревает» на предыдущих шагах (на каких именно)?
2. Кроме того, нужно исследовать результаты A/A/B-эксперимента: дизайнеры захотели поменять шрифты во всём приложении, будет ли это непривычно пользователям? Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми. Необходимо выяснить, какой шрифт лучше.

Описание данных.

Данные для исследования представляют собой лог действий пользователя, или событий:

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

Данные представлены в файле: `logs_exp.csv` .

В ходе исследования предполагается:

- загрузить и подготовить данные;
- изучить и проверить данные:
  - определить количество событий в логе;
  - определить количество пользователей в логе;
  - определить среднее количество событий на пользователя;
  - определить период проведения исследования (максимальную, минимальную дату, гистограмму по дате и времени), одинаково ли полные данные за весь период;
  - определить, с какого момента данные полные и отбросить более старые;
- изучить воронку событий:
  - какие события есть в логах, как часто они встречаются;
  - определить количество пользователей, совершавших каждое из этих событий;
  - определить долю пользователей, которые хоть раз совершали событие;
  - предположить порядке происходят события, все ли они выстраиваются в последовательную цепочку;
  - по воронке событий посчитать, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем);
  - определить, на каком шаге теряется больше всего пользователей;
  - определить, какая доля пользователей доходит от первого события до оплаты;
- проанализировать результаты проведённого A/A/B-эксперимента:
  - определить количество пользователей в каждой экспериментальной группе;
  - проверить, находят ли статистические критерии разницу между выборками контрольных групп 246 и 247;
  - для самого популярного события определить количество пользователей в каждой из контрольных групп, совершивших это событие, а также долю таких пользователей;
  - оценить корректность разбиения на группы (для контрольных групп);
  - сравнить результаты экспериментальной группы с каждой из контрольных групп в отдельности по каждому событию, а также с объединённой контрольной группой
  - оценить уровень статистической значимости для эксперимента.

1 Импорт библиотек и определение функций проекта

1.1 Импорт библиотек

In [1]:
```
import os
import pandas as pd
import numpy as np
from plotly import graph_objects as go
import plotly.express as px
from scipy import stats as st
import math as mth
```

1.2 Функции загрузки и обзора данных

In [2]:
```
# определение кроссплатформенной функции загрузки данных
# ======================================================
# на вход подаётся:
#     file_name - имя файла
# на выходе - датафрейм с загруженными данными
# ======================================================
def open_file(file_name, sep=','):↔
```

In [3]:
```
# определение функции обзора данных
# ================================================
# на вход подаётся датафрейм df
# на выходе:
#     - 10 случайных строк df
#     - информация df.info()
#     - количество явных дубликатов в строках df
#     - процент пропусков данных в столбцах df
# ================================================
def data_observe(df):↔
```

1.3 Функции визуализации данных

In [4]:
```
# определение функции построения гистограммы plotly
# =================================================
def build_px_hist(df, col, hist_title, x_title, y_title, bins=200):↔
```

1.4 Функция статистического критерия на равенство долей

In [5]:
```
# определение функции выполнения статистического критерия на разность долей
# ==========================================================================
def prop_difference_criteria(
    df,           # датафрейм с данными пропорций
    part_col,     # числитель пропорции
    full_col,     # знаменатель пропорции
    alpha=.05     # критический уровень статистической значимости

):↔
```

---

2 Обзор данных

Откроем и изучим файл `logs_exp.csv` :

In [6]:
```
logs = open_file('logs_exp.csv', sep='\t')
if not logs.empty:
    data_observe(logs)
```

```
Произвольные строки таблицы:
============================
```

	EventName	DeviceIDHash	EventTimestamp	ExpId
82740	MainScreenAppear	5375697691143223385	1564823137	247
16473	PaymentScreenSuccessful	7738058666231999878	1564658954	246
180108	OffersScreenAppear	4445909378868832446	1565068556	248
17736	MainScreenAppear	3158207671983836184	1564660722	247
95852	MainScreenAppear	8415324821149653065	1564843268	247

```
Информация о таблице:
====================
<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

Количество явных дубликатов в таблице:
======================================
413

Процент пропусков в столбцах:
============================
```

	NaNs, %
EventName	0.00
DeviceIDHash	0.00
EventTimestamp	0.00
ExpId	0.00

Таблица `logs` состоит из 244126 строк, 4 столбцов. Типы данных в столбцах - object, int64.

Согласно описанию данных, столбцы хранят следующую информацию:

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

Столбцы поименованы в смешаном стиле.

Пользователей в логе различают по уникальному идентификатору устройства.

Время события представлено целым числом (вероятно, в Unix-формате), целесообразно преобразовать эту метку к формату даты и времени.

В таблице, предварительно, отсутствуют пропуски данных, но присутствуют явные дубликаты.

2.1 Вывод

Данные, на первый взгляд, достаточно качественные.

На этапе предварительной обработки целесообразно:

- привести для удобства названия столбцов к хорошему стилю;
- удалить явные дубликаты;
- привести метки времени к формату `datetime` (с созданием новых столбцов для удобства);
- изучить состав значений столбца `ExpId` ;
- изучить состав значений столбца `DeviceIDHash` ;
- изучить состав значений столбца `EventName` на предмет наличия неявных дубликатов;
- изучить столбцы `ExpId` и `DeviceIDHash` на предмет возможности сокращения размерности чисел до int32 в целях экономии памяти.

---

3 Предварительная обработка данных

Приведём для удобства названия столбцов к хорошему стилю:

In [7]:
```
# logs.columns = ['event_name', 'user_id', 'event_timestamp', 'experiment_id']
logs = logs.rename(columns={
    'EventName':'event_name',
    'DeviceIDHash':'user_id',
    'EventTimestamp':'event_timestamp',
    'ExpId':'experiment_id'
})
logs.columns
```

Out[7]:
```
Index(['event_name', 'user_id', 'event_timestamp', 'experiment_id'], dtype='object')
```

Столбцы поименованы в стиле snake_case.

Удалим явные дубликаты:

In [8]:
```
logs.drop_duplicates(inplace=True)
len(logs)
```

Out[8]:
```
243713
```

Из 244126 строк в таблице осталось 243713 - явные дубликаты удалены.

Преобазуем метку времени события к формату `datetime` и сохраним в столбце `event_datetime` :

In [9]:
```
logs['event_datetime'] = pd.to_datetime(logs['event_timestamp'], unit='s')
logs.head()
```

Out[9]:

	event_name	user_id	event_timestamp	experiment_id	event_datetime
0	MainScreenAppear	4575588528974610257	1564029816	246	2019-07-25 04:43:36
1	MainScreenAppear	7416695313311560658	1564053102	246	2019-07-25 11:11:42
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248	2019-07-25 11:28:47
3	CartScreenAppear	3518123091307005509	1564054127	248	2019-07-25 11:28:47
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-07-25 11:48:42

Опытным путём мы установили, что метка времени указана в секундах в формате `unix` .

Для удобства добавим также столбец `'event_date'` , содержащий только дату события:

In [10]:
```
logs['event_date'] = logs['event_datetime'].dt.date
logs.head()
```

Out[10]:

	event_name	user_id	event_timestamp	experiment_id	event_datetime	event_date
0	MainScreenAppear	4575588528974610257	1564029816	246	2019-07-25 04:43:36	2019-07-25
1	MainScreenAppear	7416695313311560658	1564053102	246	2019-07-25 11:11:42	2019-07-25
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019-07-25
3	CartScreenAppear	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019-07-25
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-07-25 11:48:42	2019-07-25

Изучим состав уникальных значений столбца `'experiment_id'` :

In [11]:
```
logs.experiment_id.sort_values().unique()
```

Out[11]:
```
array([246, 247, 248])
```

Действительно в таблице приведены данные эксперимента для трёх групп - 246, 247, 248.

Для таких данных тип int64 явно избыточен. Приведём столбец `'experiment_id'` к типу int32 для экономии памяти:

In [12]:
```
logs.experiment_id = logs.experiment_id.astype('int32')
logs.experiment_id.dtype
```

Out[12]:
```
dtype('int32')
```

Изучим состав уникальных значений столбца `'user_id'` :

In [13]:
```
logs.user_id.sort_values().unique()
```

Out[13]:
```
array([   6888746892508752,    6909561520679493,    6922444491712477, ...,
       9220879493065341500, 9221926045299980007, 9222603179720523844])
```

In [14]:
```
logs.user_id.nunique()
```

Out[14]:
```
7551
```

Итак, мы имеем 7551 уникальный идентификатор пользователя, выраженный целым числом длиной от 16 до 19 десятичных цифр. ***Изменение типа для экономии памяти не целесообразно***.

Изучим состав значений столбца `'event_name'` на предмет наличия неявных дубликатов

In [15]:
```
logs.event_name.sort_values().unique()
```

Out[15]:
```
array(['CartScreenAppear', 'MainScreenAppear', 'OffersScreenAppear',
       'PaymentScreenSuccessful', 'Tutorial'], dtype=object)
```

Столбец `'event_name'` содержит всего 5 возможных значений. Неявные дубликаты, очевидно, отсутствуют.

3.1 Вывод

В ходе предварительной обработки мы улучшили данные следующим образом:

- поименовали столбцы в стиле snake_case, при этом заменив для удобства понятие `DeviceIDHash` , по которому в эксперименте различались пользователи, понятием `user_id` - уникальный идентификатор пользователя (в контексте данных эта замена правомочна);
- удалили 413 явных дубликатов из таблицы (эти дубликаты, вероятно,появились на этапе сбора данных из разных источников и в перспективе могли оказать влияние на интерпретацию результатов теста);
- установили, что в таблице отсутствуют неявные дубликаты, количество групп эксперимента равно заявленному, количество исследуемых различных событий равно 5;
- преобразовали метку времени в объект `datetime` , а также выделили дату в отдельный столбец для удобства.

---

4 Исследование данных

4.1 Исследование пользователей и событий

Исследовательский анализ данных мы начали уже на этапе предварительной обработки. Мы установили, что:

1. В логе хранятся данные о 5 событиях:
   - 'CartScreenAppear' - пользователь увидел экран корзины,
   - 'MainScreenAppear' - пользователь увидел главный экран приложения,
   - 'OffersScreenAppear' - пользователь увидел специальные предложения,
   - 'PaymentScreenSuccessful' - пользователь увидел экран успешной оплаты,
   - 'Tutorial' - пользователь ознакомился с обучением пользованию приложением.
2. В логе зафиксирована деятельность 7551 уникального пользователя.

Определим общее количество зафиксированных событий:

In [16]:
```
len(logs)
```

Out[16]: 243713

В логе зафиксированы данные о 243713 событиях.

Определим, сколько в среднем событий приходится на 1 пользователя. Для этого посчитаем общее количество зафиксированных событий и разделим его на количество уникальных пользователей:

In [17]:
```
round(len(logs) / len(logs.user_id.unique()))
```

Out[17]: 32

Итак, в среднем 32 события приходится на 1 пользователя. Но прежде, чем полагаться на эту оценку, сгруппируем лог по `'user_id'` , посчитаем количество событий для каждого пользователя (одновременно получим диапазон дат, за которые эти события произошли) и охарактеризуем полученное распределение, построим гистограмму:

In [18]:
```
# сгруппируем данные
logs_by_users = (
    logs
    .groupby(by='user_id')
    .agg({
        'event_name':'count',
        'event_date':['min', 'max']
    })
)
logs_by_users.columns = ['event_count', 'min_event_date', 'max_event_date']
# результат группировки
display(logs_by_users.sample(5))
# характеризуем распределение
display(logs_by_users.event_count.describe())
# строим гистограмму
build_px_hist(
    df=logs_by_users,
    col='event_count',
    hist_title="Гистограмма количества событий для пользователя",
    x_title="Количество событий",
    y_title="Частота встречаемости",
    bins=200
)
```

user_id	event_count	min_event_date	max_event_date
3158207671983836184	68	2019-07-31	2019-08-06
33176906322804559	54	2019-07-31	2019-08-06
1321126011609488133	5	2019-08-01	2019-08-03
6040191502175430751	28	2019-07-30	2019-08-06
2673581778942917840	35	2019-08-01	2019-08-07

```
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_count, dtype: float64
```

Мы получили большой разброс значений - от 1 до 2307 событий на пользователя. При этом для 75% пользователей количество событий не превышает 37.

На гистограмме более 200-250 событий встречаются в единичных случаях.

Вероятно, для небольшого количества пользователей зафиксировано аномально большое количество событий (выбросы). Поэтому в качестве первичной оценки среднего количества событий на пользователя целесообразно рассматривать медиану:

In [19]:
```
logs_by_users.event_count.median()
```

Out[19]:
```
20.0
```

Итак, в среднем на пользователя в логе приходится 20 событий.

Для того, чтобы отобрать аномальных пользователей, оценим 95 и 99 перцентили распределения:

In [20]:
```
np.percentile(logs_by_users.event_count, [95, 99])
```

Out[20]:
```
array([ 89. , 200.5])
```

Действительно, 99% пользователей имеют не более 200 событий. Поэтому большее количество событий можно считать аномальными выбросами. ***Не будем пока отбрасывать этих пользователей, но будем учитывать их при оценке результатов теста.***

In [21]:
```
# выделение аномальных пользователей в список
abnormal_users = list(
    logs_by_users
    .query('event_count > 200')
    .index
)
len(abnormal_users)
```

Out[21]: 76

Итак, 76 пользователей признаны аномальными по количеству событий.

4.2 Исследование временных диапазонов

Установим период времени, за который имеются данные эксперимента:

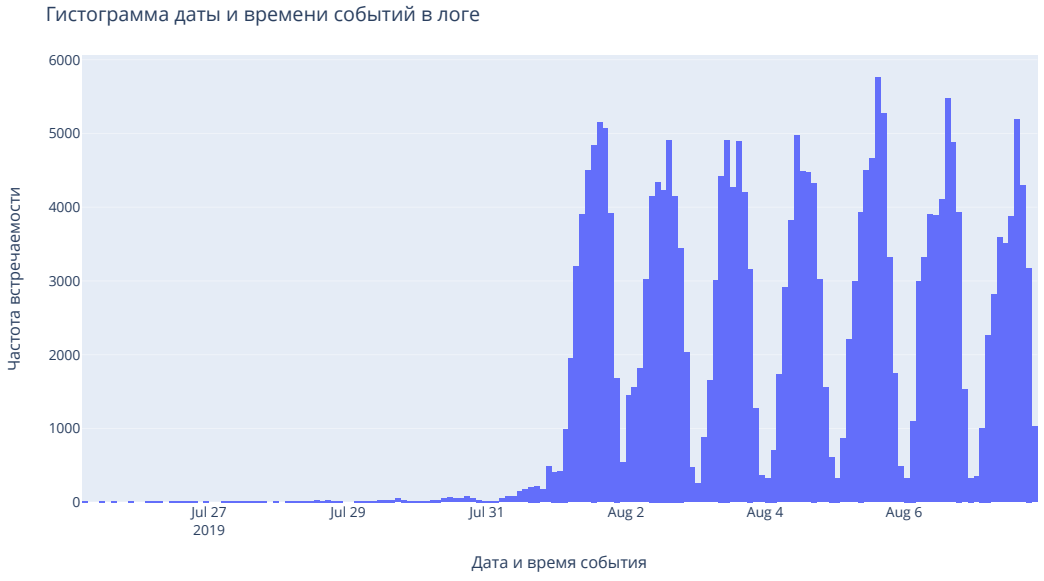
In [22]:
```
print(logs.event_datetime.min())
print(logs.event_datetime.max())
```

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

Наш эксперимент продолжался с 25 июля по 7 августа 2019 года.

Для оценки полноты данных по времени эксперимента построим гистограмму распределения по дате и времени события:

In [23]:
```
# строим гистограмму
build_px_hist(
    df=logs,
    col='event_datetime',
    hist_title="Гистограмма даты и времени событий в логе",
    x_title="Дата и время события",
    y_title="Частота встречаемости",
    bins=200
)
```



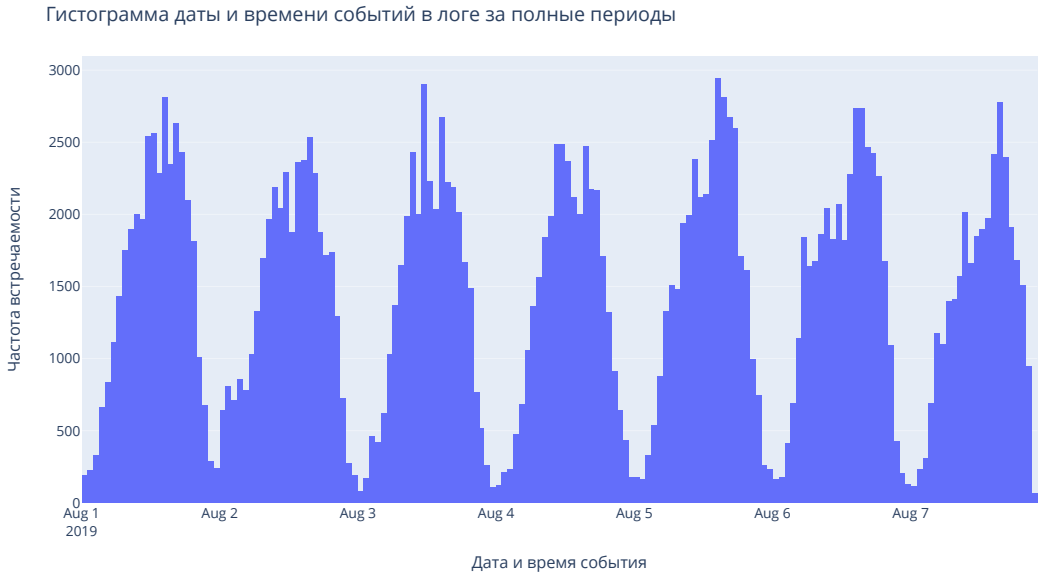
Из гистограммы следует, что данные за 25-31 июля не могут считаться полными. Целесообразно оставить для дальнейшего анализа только полные дни теста - с 1 по 7 августа:

In [24]:
```
good_logs = logs.query("'2019-08-01' <= event_datetime").copy()
good_logs.event_datetime.min()
```

Out[24]: Timestamp('2019-08-01 00:07:28')

Теперь мы оперируем только полными данными:

In [25]:
```
# строим гистограмму
build_px_hist(
    df=good_logs,
    col='event_datetime',
    hist_title="Гистограмма даты и времени событий в логе за полные периоды",
    x_title="Дата и время события",
    y_title="Частота встречаемости",
    bins=200
)
```



Определим, какую долю событий и пользователей мы потеряли, отбросив неполные данные:

In [26]:
```
print(
    'При отбрасывании неполных данных потеряно',
    round((1 - len(good_logs) / len(logs)) * 100, 2),
    '% событий.'
)
print(
    'При отбрасывании неполных данных потеряно',
    round((1 - good_logs.user_id.nunique() / logs.user_id.nunique()) * 100, 2),
    '% пользователей.'
)
```

```
При отбрасывании неполных данных потеряно 1.16 % событий.
При отбрасывании неполных данных потеряно 0.23 % пользователей.
```

Оставив только полные данные, мы потеряли 1.16 % событий и менее 1 % пользователей. Для целей исследования это не критично.

Проверим, есть ли у нас пользователи всех трёх групп после удаления неполных дат:

In [27]:
```
# посчитаем количество уникальных пользователей до и после редукции
group_counts = (
    logs
    .groupby(by='experiment_id')
    .agg({'user_id':'nunique'})
    .rename(columns={'user_id': 'user_count_before'})
    .join(
        (
            good_logs
            .groupby(by='experiment_id')
            .agg({'user_id':'nunique'})
            .rename(columns={'user_id': 'user_count_after'})
        ),
        how='left'
    )
)
# посчитаем %, на который снизилось количество уникальных пользователей
# в каждой группе
group_counts['reduce_percent'] = round(
    (1 - group_counts['user_count_after'] / group_counts['user_count_before'])
    * 100,
    2
)

group_counts
```

Out[27]:

experiment_id	user_count_before	user_count_after	reduce_percent
246	2489	2484	0.20
247	2520	2513	0.28
248	2542	2537	0.20

После удаления неполных периодов из данных мы потеряли не более 0.3 % уникальных пользователей из каждой группы тестирования.

4.3 Вывод

В ходе исследовательского анализа мы изучили:

- пользователей и события;
- полноту данных в различные временные диапазоны.

1. В логе хранятся данные о 5 событиях:
   - 'CartScreenAppear' - пользователь увидел экран корзины,
   - 'MainScreenAppear' - пользователь увидел главный экран приложения,
   - 'OffersScreenAppear' - пользователь увидел специальные предложения,
   - 'PaymentScreenSuccessful' - пользователь увидел экран успешной оплаты,
   - 'Tutorial' - пользователь ознакомился с обучением пользованию приложением.
2. В логе зафиксированы данные о 243713 событиях, совершённых 7551 уникальным пользователем.
3. В среднем 32 события приходится на 1 пользователя, однако эта оценка является смещённой в силу наличия пользователей с аномальным количеством событий. Медианное количество событий на пользователя равно 20.
4. Аномальные пользователи перечислены (не более 1 % от общего количества пользователей с более чем 200 событиями) и оставлены в датасете. Будем учитывать их при оценке результатов теста.
5. Эксперимент продолжался с 25 июля по 7 августа 2019 года.
5. Данные за весь период времени признаны неполными. Полные данные наблюдаются за период 1-7 августа 2019 года.
6. Отбросив неполные данные до 1 августа, мы потеряли 1.16 % событий и менее 1 % пользователей. При этом, в каждой группе теста потеряно не более 0.3 % уникальных пользователей.

---

5 Построение и анализ воронки событий

Как мы уже отмечали в разделах 3 и 4, в логе хранятся данные о 5 событиях:

- 'CartScreenAppear' - пользователь увидел экран корзины,
- 'MainScreenAppear' - пользователь увидел главный экран приложения,
- 'OffersScreenAppear' - пользователь увидел специальные предложения,
- 'PaymentScreenSuccessful' - пользователь увидел экран успешной оплаты,
- 'Tutorial' - пользователь ознакомился с обучением пользованию приложением.

Оценим частоту каждого события? количество пользователей, совершивших эти события, на периоде полных данных. Кроме того, определим долю пользователей, хоть раз совершивших событие:

In [28]:
```
# построение воронки
good_logs_funnel = (
    good_logs
    .groupby(by='event_name')
    .agg({
        'event_datetime':'count',
        'user_id':'nunique'
    })
    .rename(columns={
        'event_datetime':'event_count',
        'user_id':'user_count'
    })
    .sort_values(by='event_count', ascending=False)
)
# вычисление доли от общего количества пользователей
good_logs_funnel['user_percent'] = (
    round(good_logs_funnel['user_count'] / good_logs.user_id.nunique() * 100,
          2)
)
good_logs_funnel
```

Out[28]:

event_name	event_count	user_count	user_percent
MainScreenAppear	117328	7419	98.47
OffersScreenAppear	46333	4593	60.96
CartScreenAppear	42303	3734	49.56
PaymentScreenSuccessful	33918	3539	46.97
Tutorial	1005	840	11.15

In [29]:
```
good_logs.user_id.nunique()
```

Out[29]: 7534

Замечания:

1. Процент пользователей для самого распространённого события не равен 100 % потому, что мы отрезали периоды неполных данных, в которые оставшиеся 1.53 % пользователей могли совершить событие *MainScreenAppear*.
2. С другой стороны, нам не до конца известна логика работы приложения. Если оно допускает попадание пользователей на карточки предложений, например, по рекламным ссылкам, то наблюдаемая ситуация - в порядке вещей.

В дальнейших рассуждениях будем считать, что **на любой экран приложения можно попасть по внешней ссылке** (как правило, кроме успешной оплаты, на который не логично попадать, минуя корзину), а не только через главный экран. В этой ситуации **удалять пользователей, не увидевших главный экран, не следует**.

В полученной таблице мы наблюдаем воронку событий, отсортированную по убыванию количества событий. Легко видеть, что количество пользователей, совершивших каждое событие, равно как и их доля от общего количества, также убывает:

In [30]:
```
# визуализируем убывание
fig = go.Figure(data=[
    go.Bar(
        name='event_count',
        x=good_logs_funnel.index,
        y=good_logs_funnel.event_count,
        text=good_logs_funnel.event_count
    ),
    go.Bar(
        name='user_count',
        x=good_logs_funnel.index,
        y=good_logs_funnel.user_count,
        text=good_logs_funnel.user_count
    )
])
# настроим график
fig.update_layout(
    barmode='group',
    title='Количество событий и пользователей для каждого вида событий',
    xaxis_title='Виды событий',
    yaxis_title='Количество'
)
fig.show()
```

Согласно данным, с высокой долей вероятности события в приложении происходят в следующем порядке:

1. MainScreenAppear
2. OffersScreenAppear
3. CartScreenAppear
4. PaymentScreenSuccessful

Это согласуется с общей логикой приложения для продажи чего-либо.

Особняком стоит событие *Tutorial*: немногие пользователи открывают экран обучения/справки в приложении. Кроме того, данный экран можно вызвать в любой момент. Поэтому данное событие следует исключить из воронки.

Визуализируем полученную воронку событий по числу пользователей, их совершивших:

In [31]:
```
# визуализируем воронку
fig = go.Figure(
    go.Funnel(
        x=(
            good_logs_funnel.query('index != "Tutorial"')
            .user_count
        ),
        y=(
            good_logs_funnel.query('index != "Tutorial"')
            .index
        ),
   )
)
# настроим график
fig.update_layout(
    title='Воронка событий по числу пользователей, их совершивших',
)
fig.show()
```

Согласно воронке событий, больше всего пользователей приложение теряет при переходе от главного экрана на экран предложений.

Посчитаем в процентах, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем), а также какая доля пользователей доходит от первого события до оплаты:

In [32]:
```
# отбросим ненужное событие
good_logs_funnel = good_logs_funnel.query('index != "Tutorial"').copy()
# создадим вспомогательный столбец с числом пользователей на предыдущем шаге
good_logs_funnel['prev_user_count'] = (
    good_logs_funnel['user_count'].shift(
        1,
        fill_value=good_logs_funnel['user_count'].max()
    )
)
# посчитаем процент от предыдущего шага
good_logs_funnel['prev_user_percent'] = (
    round(
        good_logs_funnel['user_count'] / good_logs_funnel['prev_user_count'] * 100,
        2
    )
)
good_logs_funnel
```

Out[32]:

event_name	event_count	user_count	user_percent	prev_user_count	prev_user_percent
MainScreenAppear	117328	7419	98.47	7419	100.00
OffersScreenAppear	46333	4593	60.96	7419	61.91
CartScreenAppear	42303	3734	49.56	4593	81.30
PaymentScreenSuccessful	33918	3539	46.97	3734	94.78

Из данных следует, что до успешной оплаты добираются только 46.97 % пользователей. Больше всего пользователей теряется на шаге *OffersScreenAppear* (38.09 %).

5.1 Вывод

Мы построили и проанализировали воронку событий для приложения по продаже продуктов питания.

1. Воронку формируют события
   A. MainScreenAppear
   B. OffersScreenAppear
   C. CartScreenAppear
   D. PaymentScreenSuccessful
2. Событие *Tutorial* не попадает в воронку, поскольку не укладывается в логику приложения и не является обязательным шагом на пути к покупке.
3. Больше всего пользователей теряется на шаге *OffersScreenAppear* (38.09 % от предыдущего шага и от общего количества).

4. До успешной оплаты добираются только 46.97 % пользователей - меньше половины!

---

6 Анализ результатов A/A/B-теста

A/A/B-тест заключался в демонстрации тестовой группе (248) экранов приложения с изменённым шрифтом, а двум контрольным (246 и 247) - с прежним.

Анализ результатов A/A/B-теста будем проводить в 2 этапа:

1. проверка результатов A/A-теста;
2. проверка результатов A/B-теста в сравнении с каждой из контрольных групп и с объединенной контрольной группой.

После удаления временных периодов с неполными данными у нас осталось следующее количество пользователей в каждой из групп теста:

In [33]:
```
(
    good_logs
    .groupby(by='experiment_id')
    .agg({'user_id':'nunique'})
    .rename(columns={'user_id': 'user_count'})
)
```

Out[33]:

experiment_id	user_count
246	2484
247	2513
248	2537

Группы примерно равномощны. Проверим, попали ли одни и те же пользователи в разные группы:

In [34]:
```
def list_intersection(list_1, list_2):
    intersection = []
    for item in list_1:
        if item in list_2:
            intersection.append(item)
    return intersection
```

In [35]:
```
users_246 = list(
    good_logs
    .query('experiment_id == 246')
    .user_id.unique()
)
users_247 = list(
    good_logs
    .query('experiment_id == 247')
    .user_id.unique()
)
users_248 = list(
    good_logs
    .query('experiment_id == 248')
    .user_id.unique()
)
# найдём пересечение групп A1 и A2
users_246_247 = list_intersection(users_246, users_247)
print(f'В группах 246 и 247: {len(users_246_247)} общих пользователей')
# найдём пересечение групп A1 и B
users_246_248 = list_intersection(users_246, users_248)
print(f'В группах 246 и 248: {len(users_246_248)} общих пользователей')
# найдём пересечение групп A2 и B
users_247_248 = list_intersection(users_247, users_248)
print(f'В группах 247 и 248: {len(users_247_248)} общих пользователей')
```

```
В группах 246 и 247: 0 общих пользователей
В группах 246 и 248: 0 общих пользователей
В группах 247 и 248: 0 общих пользователей
```

Пользователи в группах не пересекаются.

6.1 Проверка результатов A/A-теста

Проверим, находят ли статистические критерии разницу между выборками 246 и 247. Для этого построим воронки по каждой группе и выполним тест на равенство конверсии пользователей в покупателей для контрольных групп.

In [36]:
```
# построение воронки
groups_funnel = (
    good_logs.query('event_name != "Tutorial"')
    .groupby(by=['experiment_id', 'event_name'], as_index=False)
    .agg({
        'user_id':'nunique'
    })
    .rename(columns={
        'user_id':'user_count'
    })
    .sort_values(by='user_count', ascending=False)
)
groups_funnel
```

Out[36]:

	experiment_id	event_name	user_count
9	248	MainScreenAppear	2493
5	247	MainScreenAppear	2476
1	246	MainScreenAppear	2450
2	246	OffersScreenAppear	1542
10	248	OffersScreenAppear	1531
6	247	OffersScreenAppear	1520
0	246	CartScreenAppear	1266
4	247	CartScreenAppear	1238
8	248	CartScreenAppear	1230
3	246	PaymentScreenSuccessful	1200
11	248	PaymentScreenSuccessful	1181
7	247	PaymentScreenSuccessful	1158

In [37]:
```
groups_funnel = (
    groups_funnel
    .pivot(index='experiment_id',
           columns='event_name',
           values='user_count')
    .join(
        (
            good_logs
            .groupby(by='experiment_id')
            .agg({'user_id':'nunique'})
            .rename(columns={'user_id': 'user_count'})
        ),
        how='left'
    )
)
groups_funnel
```

Out[37]:

experiment_id	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	user_count
246	1266	2450	1542	1200	2484
247	1238	2476	1520	1158	2513
248	1230	2493	1531	1181	2537

Мы собрали данные об общем количестве пользователей, а также о количестве пользователей на каждом этапе воронки для каждой из групп. Используем критерий на равенство долей для определения статистической разницы между конверсией пользователей (user_count) в покупателей (PaymentScreenSuccessful).

Критерий на равенство долей основан на вычислении статистики, представляющей разницу между пропорциями (можно доказать, что она распределена нормально):

$$Z \approx \frac{(p_1 - p_2) - (\pi_1 - \pi_2)}{\sqrt{p(1-p)(\frac{1}{n_1} - \frac{1}{n_2})}} \sim N(0,1),$$

где:

- n_1 и n_2 - размеры двух сравниваемых выборок;
- p_1 и p_2 - пропорции, наблюдаемые в выборках;
- p - пропорция в выборке, скомбинированной из двух наблюдаемых;
- π_1 и π_2 - настоящие пропорции в сравниваемых генеральных совокупностях.

При проверке гипотезы о равенстве π_1 и π_2 рассчитываемая статистика $Z \approx \frac{(p_1-p_2)}{\sqrt{p(1-p)(\frac{1}{n_1} - \frac{1}{n_2})}}$ зависит только от выборочных данных.

Подсчёт статистики Z и вычисление вероятности получить заданное или большее отклонение от 0 при заданном уровне статистической значимости помещены в функцию `prop_difference_criteria` .

Сформулируем статистические гипотезы:

- **H0** - конверсия (доля) пользователей в покупателей *одинакова* для групп 246 и 247
- **H1** - конверсия пользователей в покупателей для групп 246 и 247 различна

Выделим требуемый датафрейм и выполним на нём критерий на равенство долей:

In [38]:
```
test_data = groups_funnel.loc[[246,247], ['PaymentScreenSuccessful',
                                          'user_count']]
test_data
```

Out[38]:

experiment_id	PaymentScreenSuccessful	user_count
246	1200	2484
247	1158	2513

In [39]:
```
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'PaymentScreenSuccessful', 'user_count')
```

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии контрольных групп различными.

Предварительно можно утверждать, что:

- на результаты теста, проведённого на двух контрольных группах, не влияют аномалии и выбросы, обнаруженные нами ранее;
- данные отправляются в системы аналитики корректно.

Перейдём к проверке для контрольных групп гипотез о равенстве пошаговых конверсий воронки. По-прежнему, для каждого шага будем следующим образом формулировать статистические гипотезы:

- **H0** - конверсия (доля) пользователей на текущем шаге воронки *одинакова* для групп 246 и 247
- **H1** - конверсия пользователей на текущем шаге воронки для групп 246 и 247 различна

Самым популярным событием является *MainScreenAppear*. Проверим гипотезу о равенстве долей пользователей контрольных групп, совершивших данное событие:

In [40]:
```
# выделим данные
test_data = groups_funnel.loc[[246,247], ['MainScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'MainScreenAppear', 'user_count')
```

experiment_id	MainScreenAppear	user_count
246	2450	2484
247	2476	2513

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать доли пользователей контрольных групп на шаге MainScreenAppear различными.

Следующим по популярности событием является *OffersScreenAppear*. Проверим гипотезу о равенстве долей пользователей контрольных групп, совершивших данное событие:

In [41]:
```
# выделим данные
test_data = groups_funnel.loc[[246,247], ['OffersScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'OffersScreenAppear', 'user_count')
```

experiment_id	OffersScreenAppear	user_count
246	1542	2484
247	1520	2513

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать доли пользователей контрольных групп на шаге OffersScreenAppear различными.

Следующим по популярности событием является *CartScreenAppear*. Проверим гипотезу о равенстве долей пользователей контрольных групп, совершивших данное событие:

In [42]:
```
# выделим данные
test_data = groups_funnel.loc[[246,247], ['CartScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'CartScreenAppear', 'user_count')
```

experiment_id	CartScreenAppear	user_count
246	1266	2484
247	1238	2513

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать доли пользователей контрольных групп на шаге CartScreenAppear различными.

Завершает воронку событие *PaymentScreenSuccessful*, равенство долей пользователей для которого мы проверяли в самом начале, и также не нашли статистически значимых различий.

Вывод: для контрольных групп 246 и 247 при заданном критическом уровне статистической значимости 0.05:

- нет оснований считать конверсии пользователей в покупателей различными;
- нет оснований считать конверсии с каждого предыдущего на каждый последующий шаг воронки различными.

6.2 Проверка результатов A/B-теста

Проверять результаты A/B-теста будем в три подэтапа:

1. сравнение конверсий групп 246 и 248;
2. сравнение конверсий групп 247 и 248;
3. сравнение конверсий объединённой контрольной группы (246 и 247) и тестовой группы 248.

6.2.1 Сравнение конверсий групп 246 и 248

Сформулируем статистические гипотезы:

- **H0** - конверсия (доля) пользователей в покупателей *одинакова* для групп 246 и 248
- **H1** - конверсия пользователей в покупателей для групп 246 и 248 различна

In [43]:
```
# выделим данные
test_data = groups_funnel.loc[[246,248], ['PaymentScreenSuccessful',
                                          'user_count']]
# проверим гипотезу о равенстве конверсий
prop_difference_criteria(test_data, 'PaymentScreenSuccessful', 'user_count')
```

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии пользователей в покупателей для групп 246 и 248 различными.

Перейдём к проверке для выбранных групп гипотез о равенстве пошаговых конверсий воронки. По-прежнему, для каждого шага будем следующим образом формулировать статистические гипотезы:

- **H0** - конверсия (доля) пользователей на текущем шаге воронки *одинакова* для групп 246 и 248
- **H1** - конверсия пользователей на текущем шаге воронки для групп 246 и 248 различна

Для события *MainScreenAppear*:

In [44]:
```
# выделим данные
test_data = groups_funnel.loc[[246,248], ['MainScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'MainScreenAppear', 'user_count')
```

experiment_id	MainScreenAppear	user_count
246	2450	2484
248	2493	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 246 и 248 на шаге MainScreenAppear различными.

Для события *OffersScreenAppear*:

In [45]:
```
# выделим данные
test_data = groups_funnel.loc[[246,248], ['OffersScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'OffersScreenAppear', 'user_count')
```

experiment_id	OffersScreenAppear	user_count
246	1542	2484
248	1531	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 246 и 248 на шаге OffersScreenAppear различными.

Для шага *CartScreenAppear*:

In [46]:
```
# выделим данные
test_data = groups_funnel.loc[[246,248], ['CartScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'CartScreenAppear', 'user_count')
```

experiment_id	CartScreenAppear	user_count
246	1266	2484
248	1230	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 246 и 248 на шаге CartScreenAppear различными.

Для шага *PaymentScreenSuccessful* равенство долей пользователей мы проверили в самом начале, и не нашли статистически значимых различий.

6.2.2 Сравнение конверсий групп 247 и 248

Сформулируем статистические гипотезы:

- **H0** - конверсия (доля) пользователей в покупателей *одинакова* для групп 247 и 248
- **H1** - конверсия пользователей в покупателей для групп 247 и 248 различна

In [47]:
```
# выделим данные
test_data = groups_funnel.loc[[247,248], ['PaymentScreenSuccessful',
                                          'user_count']]
# проверим гипотезу о равенстве конверсий
prop_difference_criteria(test_data, 'PaymentScreenSuccessful', 'user_count')
```

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии пользователей в покупателей для групп 247 и 248 различными.

Перейдём к проверке для выбранных групп гипотез о равенстве пошаговых конверсий воронки. По-прежнему, для каждого шага будем следующим образом формулировать статистические гипотезы:

- **H0** - конверсия (доля) пользователей на текущем шаге воронки *одинакова* для групп 247 и 248
- **H1** - конверсия пользователей на текущем шаге воронки для групп 247 и 248 различна

Для шага *MainScreenAppear*:

In [48]:
```
# выделим данные
test_data = groups_funnel.loc[[247,248], ['MainScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'MainScreenAppear', 'user_count')
```

experiment_id	MainScreenAppear	user_count
247	2476	2513
248	2493	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 247 и 248 на шаге MainScreenAppear различными.

Для шага *OffersScreenAppear*:

In [49]:
```
# выделим данные
test_data = groups_funnel.loc[[247,248], ['OffersScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'OffersScreenAppear', 'user_count')
```

experiment_id	OffersScreenAppear	user_count
247	1520	2513
248	1531	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 247 и 248 на шаге OffersScreenAppear различными.

Для шага *CartScreenAppear*:

In [50]:
```
# выделим данные
test_data = groups_funnel.loc[[247,248], ['CartScreenAppear',
                                          'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'CartScreenAppear', 'user_count')
```

experiment_id	CartScreenAppear	user_count
247	1238	2513
248	1230	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 247 и 248 на шаге CartScreenAppear различными.

Для шага *PaymentScreenSuccessful* равенство долей пользователей мы проверили в самом начале, и не нашли статистически значимых различий.

6.2.3 Сравнение конверсий объединённой контрольной группы (246 и 247) и тестовой группы 248

Соберём объединённую контрольную группу, обозначим её 245:

In [51]:
```
t_groups_funnel = groups_funnel.T
t_groups_funnel[245] = t_groups_funnel[246] + t_groups_funnel[247]
t_groups_funnel.T
```

Out[51]:

experiment_id	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	user_count
246	1266	2450	1542	1200	2484
247	1238	2476	1520	1158	2513
248	1230	2493	1531	1181	2537
245	2504	4926	3062	2358	4997

Сформулируем статистические гипотезы:

- **H0** - конверсия (доля) пользователей в покупателей *одинакова* для групп 245 и 248
- **H1** - конверсия пользователей в покупателей для групп 245 и 248 различна

In [52]:
```
# выделим данные
test_data = t_groups_funnel.T.loc[[245,248], ['PaymentScreenSuccessful',
                                              'user_count']]
# проверим гипотезу о равенстве конверсий
prop_difference_criteria(test_data, 'PaymentScreenSuccessful', 'user_count')
```

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии пользователей в покупателей для групп 245 и 248 различными.

Перейдём к проверке для выбранных групп гипотез о равенстве пошаговых конверсий воронки. По-прежнему, для каждого шага будем следующим образом формулировать статистические гипотезы:

- **H0** - конверсия (доля) пользователей на текущем шаге воронки *одинакова* для групп 245 и 248
- **H1** - конверсия пользователей на текущем шаге воронки для групп 245 и 248 различна

Для шага *MainScreenAppear*:

In [53]:
```
# выделим данные
test_data = t_groups_funnel.T.loc[[245,248], ['MainScreenAppear',
                                              'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'MainScreenAppear', 'user_count')
```

experiment_id	MainScreenAppear	user_count
245	4926	4997
248	2493	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 245 и 248 на шаге MainScreenAppear различными.

Для шага *OffersScreenAppear*:

In [54]:
```
# выделим данные
test_data = t_groups_funnel.T.loc[[245,248], ['OffersScreenAppear',
                                              'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'OffersScreenAppear', 'user_count')
```

experiment_id	OffersScreenAppear	user_count
245	3062	4997
248	1531	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 245 и 248 на шаге OffersScreenAppear различными.

Для шага *CartScreenAppear*:

In [55]:
```
# выделим данные
test_data = t_groups_funnel.T.loc[[245,248], ['CartScreenAppear',
                                              'user_count']]
display(test_data)
# проверим гипотезу о равенстве конверсий контрольных групп
prop_difference_criteria(test_data, 'CartScreenAppear', 'user_count')
```

experiment_id	CartScreenAppear	user_count
245	2504	4997
248	1230	2537

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

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии групп 245 и 248 на шаге CartScreenAppear различными.

Для шага *PaymentScreenSuccessful* равенство долей пользователей мы проверили в самом начале, и не нашли статистически значимых различий.

Предварительные выводы:

1. Проведённые тесты **статистических гипотез относительно равенства долей для конверсий из пользователей в покупателей** показали, что при сравнении тестовой группы 248 со всеми тремя контрольными группами (246, 247 и 246+247) при заданном критическом уровне статистической значимости 0.05 нет оснований различать данные конверсии для всех пар тестовой и контрольной групп. Следовательно, опираясь на результаты статистических тестов, **можно утверждать, что изменение шрифта не привело к статистически значимому изменению конверсии**.
2. Аналогичные результаты показали попытки статистически различить доли пользователей, совершивших отдельные события (*MainScreenAppear*, *OffersScreenAppear* и *CartScreenAppear*) при сравнении тестовой группы 248 со всеми тремя контрольными группами (246, 247 и 246+247).

При проверке статистических гипотез мы выбрали уровень значимости 0.05. Это значит, что каждый 20-й раз можно получить ложный результат.

Всего мы провели 12 статистических тестов для проверки результатов A/B-теста и 4 статистических теста для проверки результатов A/A-теста. Всего 16 статистических тестов.

In [56]:
```
# вероятность ошибиться в тесте
p_err = 1/20
# вероятность не ошибиться в тесте
p_corr = 1 - p_err
# вероятность ни разу не ошибиться за 16 тестов
p_corr_16 = p_corr ** 16

print('Вероятность ошибиться хотя бы 1 раз за 16 тестов равна', 1 - p_corr_16)
```

```
Вероятность ошибиться хотя бы 1 раз за 16 тестов равна 0.5598733313482347
```

При уровне статистической значимости 0.05 вероятность ошибиться хотя бы единожды довольно высока - примерно 56%. Оценим вероятность ошибиться хотя бы 1 раз за 16 испытаний при уровне статистической значимости 0.01:

In [57]:
```
# вероятность ошибиться в тесте
p_err = 1/100
# вероятность не ошибиться в тесте
p_corr = 1 - p_err
# вероятность ни разу не ошибиться за 16 тестов
p_corr_16 = p_corr ** 16

print('Вероятность ошибиться хотя бы 1 раз за 16 тестов равна', 1 - p_corr_16)
```

```
Вероятность ошибиться хотя бы 1 раз за 16 тестов равна 0.14854222890512447
```

При уровне статистической значимости 0.01 вероятность ошибиться хотя бы единожды значительно ниже - около 15%. Проверим наши гипотезы при уровне статистической значимости 0.01.

6.2.4 Проверка статистических гипотез при уровне значимости 0.01

Для группированной проверки гипотез напишем функцию:

In [58]:

```
def get_z_stat(group_1, group_2, alpha=.05):
    print('Проверка гипотезы о равенстве конверсий контрольных групп')
    print('============================================================')
    test_data = t_groups_funnel.T.loc[[group_1,group_2],
                                      ['PaymentScreenSuccessful',
                                        'user_count']]
    display(test_data)
    # проверим гипотезу о равенстве конверсий контрольных групп
    prop_difference_criteria(test_data,
                             'PaymentScreenSuccessful',
                             'user_count',
                             alpha=alpha)

    print('\nПроверка гипотезы о равенстве конверсий контрольных групп')
    print('на шаге MainScreenAppear')
    print('============================================================')
    # выделим данные
    test_data = t_groups_funnel.T.loc[[group_1,group_2],
                                      ['MainScreenAppear',
                                        'user_count']]
    display(test_data)
    # проверим гипотезу о равенстве конверсий контрольных групп
    prop_difference_criteria(test_data,
                             'MainScreenAppear',
                             'user_count',
                             alpha=alpha)

    print('\nПроверка гипотезы о равенстве конверсий контрольных групп')
    print('на шаге OffersScreenAppear')
    print('============================================================')
    # выделим данные
    test_data = t_groups_funnel.T.loc[[group_1,group_2],
                                      ['OffersScreenAppear',
                                        'user_count']]
    display(test_data)
    # проверим гипотезу о равенстве конверсий контрольных групп
    prop_difference_criteria(test_data,
                             'OffersScreenAppear',
                             'user_count',
                             alpha=alpha)

    print('\nПроверка гипотезы о равенстве конверсий контрольных групп')
    print('на шаге CartScreenAppear')
    print('============================================================')
    # выделим данные
    test_data = t_groups_funnel.T.loc[[group_1,group_2],
                                      ['CartScreenAppear',
                                        'user_count']]
    display(test_data)
    # проверим гипотезу о равенстве конверсий контрольных групп
    prop_difference_criteria(test_data,
                             'CartScreenAppear',
                             'user_count',
                             alpha=alpha)
```

Проверим обозначенные выше гипотезы для результатов A/A-теста при уровне значимости 0.01:

In [59]:
```
1 get_z_stat(246, 247, alpha=.01)
```

```
Проверка гипотезы о равенстве конверсий контрольных групп
============================================================
```

experiment_id	PaymentScreenSuccessful	user_count
246	1200	2484
247	1158	2513

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге MainScreenAppear
============================================================
```

experiment_id	MainScreenAppear	user_count
246	2450	2484
247	2476	2513

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге OffersScreenAppear
============================================================
```

experiment_id	OffersScreenAppear	user_count
246	1542	2484
247	1520	2513

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге CartScreenAppear
============================================================
```

experiment_id	CartScreenAppear	user_count
246	1266	2484
247	1238	2513

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

Мы получили схожие с предыдущими результаты - для контрольных групп 246 и 247 при заданном критическом уровне статистической значимости 0.01:

- нет оснований считать конверсии пользователей в покупателей различными;
- нет оснований считать конверсии с каждого предыдущего на каждый последующий шаг воронки различными.

Проверим обозначенные выше гипотезы для результатов A/B-теста (группы 246 и 248) при уровне значимости 0.01:

In [60]:
```
1 get_z_stat(246, 248, alpha=.01)
```

```
Проверка гипотезы о равенстве конверсий контрольных групп
============================================================
```

experiment_id	PaymentScreenSuccessful	user_count
246	1200	2484
248	1181	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге MainScreenAppear
============================================================
```

experiment_id	MainScreenAppear	user_count
246	2450	2484
248	2493	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге OffersScreenAppear
============================================================
```

experiment_id	OffersScreenAppear	user_count
246	1542	2484
248	1531	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге CartScreenAppear
============================================================
```

experiment_id	CartScreenAppear	user_count
246	1266	2484
248	1230	2537

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

Проверим обозначенные выше гипотезы для результатов A/B-теста (группы 247 и 248) при уровне значимости 0.01:

In [61]:
```
1 get_z_stat(247, 248, alpha=.01)
```

```
Проверка гипотезы о равенстве конверсий контрольных групп
============================================================
```

experiment_id	PaymentScreenSuccessful	user_count
247	1158	2513
248	1181	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге MainScreenAppear
============================================================
```

experiment_id	MainScreenAppear	user_count
247	2476	2513
248	2493	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге OffersScreenAppear
============================================================
```

experiment_id	OffersScreenAppear	user_count
247	1520	2513
248	1531	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге CartScreenAppear
============================================================
```

experiment_id	CartScreenAppear	user_count
247	1238	2513
248	1230	2537

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

Проверим обозначенные выше гипотезы для результатов A/B-теста с объединённой контрольной группой (группы 245 и 248) при уровне значимости 0.01:

In [62]:
```
get_z_stat(245, 248, alpha=.01)
```

```
Проверка гипотезы о равенстве конверсий контрольных групп
============================================================
```

experiment_id	PaymentScreenSuccessful	user_count
245	2358	4997
248	1181	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге MainScreenAppear
============================================================
```

experiment_id	MainScreenAppear	user_count
245	4926	4997
248	2493	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге OffersScreenAppear
============================================================
```

experiment_id	OffersScreenAppear	user_count
245	3062	4997
248	1531	2537

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

Проверка гипотезы о равенстве конверсий контрольных групп
на шаге CartScreenAppear
============================================================
```

experiment_id	CartScreenAppear	user_count
245	2504	4997
248	1230	2537

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

При уровне статистической значимости 0.01 картина не изменилась - у нас по-прежнему нет оснований считать разными конвенсии на каждом шаге для объединённой контрольной и тестовой групп. И вероятность ошибиться на всём наборе статистических тестов ниже.

6.3 Вывод

1. Проведённые тесты **статистических гипотез относительно равенства долей для конверсий из пользователей в покупателей** показали, что при сравнении тестовой группы 248 со всеми тремя контрольными группами (246, 247 и 246+247) при заданном критическом уровне статистической значимости 0.01 нет оснований различать данные конверсии для всех пар тестовой и контрольной групп.
2. Аналогичные результаты показали попытки статистически различить пошаговые конверсии для всех событий воронки при сравнении тестовой группы 248 со всеми тремя контрольными группами (246, 247 и 246+247).
3. Опираясь на результаты статистических тестов, **можно утверждать, что изменение шрифта не привело к статистически значимому изменению конверсии**.

---

7 Общий вывод

В рамках проведённого исследования мы:

- загрузить и подготовить данные;
- изучить и проверить данные;
- изучить воронку событий;
- проанализировать результаты проведённого A/A/B-эксперимента.

Выводы по каждому этапу подробно описаны в соответствующих разделах.

Сформулируем общие итоги исследования:

1. Эксперимент продолжался с 25 июля по 7 августа 2019 года. Данные за весь период времени признаны неполными. Полные данные наблюдаются за период 1-7 августа 2019 года. Отбросив неполные данные до 1 августа, мы потеряли 1.16 % событий и менее 1 % пользователей. При этом, в каждой группе теста потеряно не более 0.3 % уникальных пользователей.
2. В логе в среднем 32 события приходится на 1 пользователя, однако эта оценка является смещённой в силу наличия пользователей с аномальным количеством событий. Медианное количество событий на пользователя равно 20. Аномальные пользователи перечислены (не более 1 % от общего количества пользователей с более чем 200 событиями) и оставлены в датасете.
3. Воронку формируют события:

A. MainScreenAppear
B. OffersScreenAppear
C. CartScreenAppear
D. PaymentScreenSuccessful

4. Больше всего пользователей теряется на шаге *OffersScreenAppear* (38.09 % от от общего количества). До успешной оплаты добираются только 46.97 % пользователей - меньше половины!

5. При проверке результатов A/A-теста для контрольных групп 246 и 247 при заданном критическом уровне статистической значимости 0.01:

- нет оснований считать конверсии пользователей в покупателей различными;
- нет оснований считать конверсии для каждого шага воронки различными.

6. По результатам A/A-теста можно утверждать, что:

- на результаты теста, проведённого на двух контрольных группах, не влияют аномалии и выбросы, обнаруженные нами ранее;
- данные отправляются в системы аналитики корректно.

7. По результатам A/B-теста при заданном критическом уровне статистической значимости 0.01 **можно утверждать, что изменение шрифта не привело к статистически значимому изменению конверсии**.

Рекомендации: в зависимости от целей изменения шрифта можно рекомендовать оставить новый шрифт в приложении, поскольку по результатам A/B-теста он не ухудшает пользовательский опыт.