Учебный проект №10 (сборный №2)

Анализ событий в мобильных приложениях

студента когорты DA61
Алиева Рустама

Цель проекта:

- изучение воронки продаж (событий)
- изучение результатов проведения A/B-теста для интернет-магазина на основе данных лог-файла (оценка корректности формирования групп пользователей - 2-ух контрольных и 1-й экспериментальной, поиск статистически значимых отличий в пропорциях групп, как аргумент для принятия решения о положительном/отрицательном результате теста).

Описание данных: Для работы предлагается использовать лог-файл, где каждая запись — 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.

In [1]:
```
# блок загрузки библиотек
import pandas as pd
from matplotlib import pyplot as plt
from scipy import stats as st
import datetime as dt
import math as mth
import plotly.express as px
import warnings

# скрываем предупреждения
warnings.filterwarnings('ignore')
```

Загрузка файла с данными и изучение общей информации

In [2]:
```
# загружаем данные локально или с сервера
try:
    df = pd.read_csv('logs_exp.csv', sep='\t')
except:
    df = pd.read_csv('network path hidden', sep='\t')
```

In [3]:
```
display(df.head(5))
df.info()
```

	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

```
<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
```

Исходно имеем дф с 4 столбцами, пропущенных данных нет, явно требуется сменить тип данных для столбца с датой/временем, также вероятно, имеет смысл сменить тип на str для столцба ExpId, т.к. это идентификатор, в первую очередь.

Подготовка данных

Столбец с датами содержит информацию в Linux-стиле, также сразу переименуем столбцы в привычный регистр, столбец с ид группы преобразуем в строковый формат.

In [4]:
```
# преобразуем даты
df['EventTimestamp'] = pd.to_datetime(df['EventTimestamp'], unit='s')

df['ExpId'] = df['ExpId'].astype('str')

# переименуем столбцы в snake_case
df = df.rename(columns = (
    {'EventName':'event_name'
    ,'DeviceIDHash':'device_id'
```

```
    ,'EventTimestamp':'event_timestamp'
    ,'ExpId':'exp_id'}))
```

Пропусков в таблице, судя по info нет, проверим дубликаты

In [5]:
```
print('Количество дубликатов:', df.duplicated().sum())
print('Доля дубликтов в дф составляет {:.2f} % от общего числа записей'
      .format(100 * df.duplicated().sum() / df.shape[0]))
```

```
Количество дубликатов: 413
Доля дубликтов в дф составляет 0.17 % от общего числа записей
```

Полные дубли могут искажать информацию, поэтому удаляем, к тому же их доля не являеся существенной.

In [6]:
```
df = df.drop_duplicates(ignore_index=True)
```

Добавим столбец с датой (без указания времени)

In [7]:
```
df['event_dt'] = df['event_timestamp'].dt.date
```

Изучение и проверка данных

Основные характеристики дф

Рассмотрим основные характеристики имеющихся данных

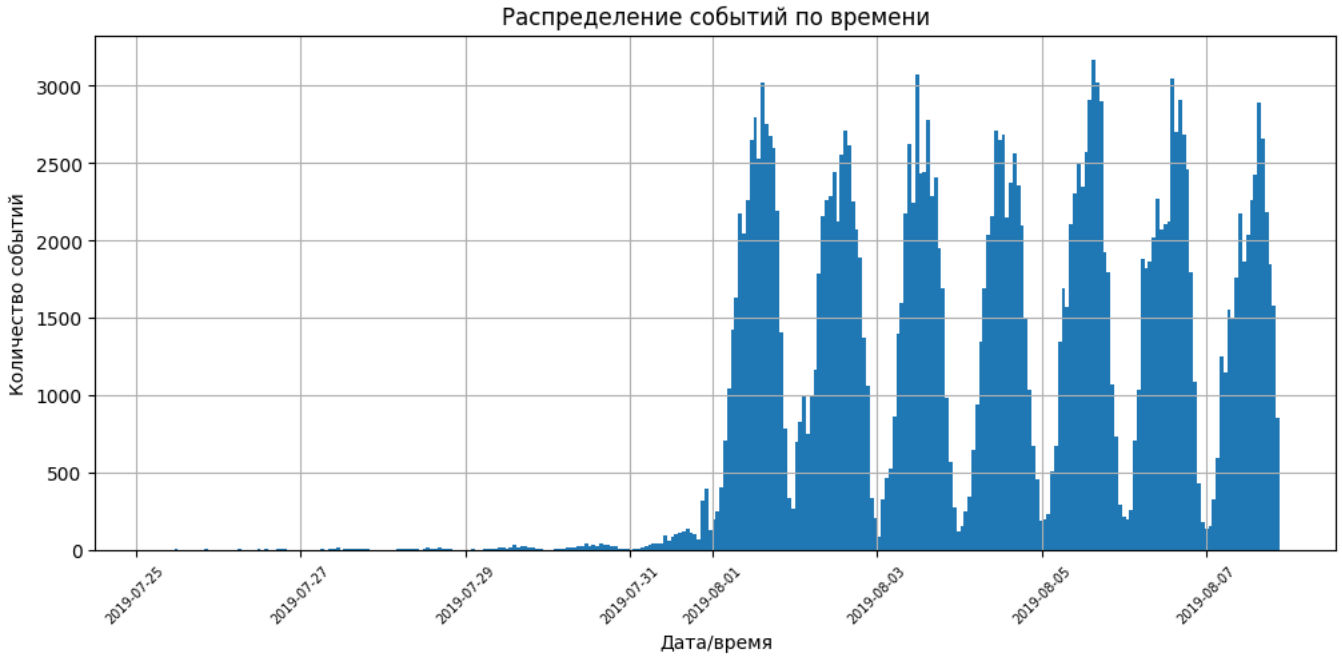
In [8]:
```
print('Общее число событий в логе: {}'
      .format(dfev:=df.shape[0]))
print('Из них уникальных событий, по которым строится воронка, в логе: {}'
      .format(dfunev:=df['event_name'].nunique()))
print('Общее число устройств пользователей в логе: {}'
      .format(dfdevid:=df['device_id'].nunique()))
print('В среднем на устройство приходится {0:.2f} события'
      .format(dfavev:=df.shape[0] / df['device_id'].nunique()))
print('Медианное значение количества событий на устройство: {0:.0f}'
      .format(dfmedev:=df.groupby('device_id')
              .agg({'event_name':'count'})['event_name'].median()))
print('Минимальная (самая ранняя) дата события в логе: {}'
      .format(df['event_timestamp'].min()))
print('Максимальная (самая поздняя) дата события в логе: {}'
      .format(df['event_timestamp'].max()))
```

```
Общее число событий в логе: 243713
Из них уникальных событий, по которым строится воронка, в логе: 5
Общее число устройств пользователей в логе: 7551
В среднем на устройство приходится 32.28 события
Медианное значение количества событий на устройство: 20
Минимальная (самая ранняя) дата события в логе: 2019-07-25 04:43:36
Максимальная (самая поздняя) дата события в логе: 2019-08-07 21:15:17
```

Проверка полноты/достаточности данных

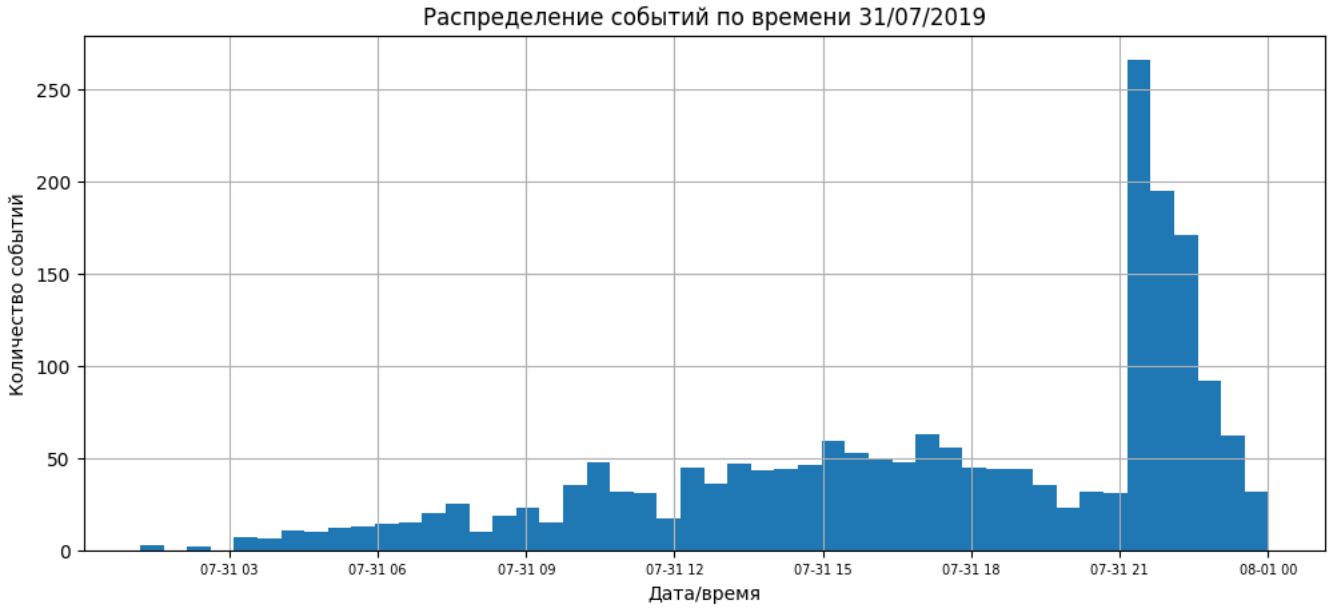
Для оценки полноты данных по времени (датам) построим гистрограмму

In [9]:
```
plt.figure(figsize=(12,5))
plt.hist(df['event_timestamp'],300)
plt.title(label='Распределение событий по времени')
plt.ylabel('Количество событий')
plt.xlabel('Дата/время')
plt.xticks(rotation=45, fontsize=7)
plt.grid(True)
plt.show()
```



Отдельно рассмотрим 31/07 более крупным масштабом

In [10]:
```
plt.figure(figsize=(12,5))
plt.hist(df[df['event_dt'] == dt.date(2019,7,31)]['event_timestamp'], 48)
plt.title(label='Распределение событий по времени 31/07/2019')
plt.ylabel('Количество событий')
plt.xlabel('Дата/время')
plt.xticks(rotation=0, fontsize=7)
plt.grid(True)
plt.show()
```



Судя по гистограммме, имеем информацию в полном объёме с 21:00 31/07/2019 по максимальную отметку времени в логе 21:15 07/08/2019, более старую информацию до 21:00 31/07/2019 удалим.

In [11]:
```
df = df[df['event_timestamp'] >= dt.datetime(2019,7,31,21,0,0)].reset_index()
```

Оценим, сколько информации мы удалили

In [12]:
```
print('Изменение общего числа событий в логе: {0:.2f}% или {1} событий'
      .format(100 * (df.shape[0] / dfev - 1)
              ,df.shape[0] - dfev))
print('Изменение общего числа устройств пользователей в логе: {:.2f}% или {} ид устройств'
      .format(100 * (df['device_id'].nunique() / dfdevid - 1)
              ,df['device_id'].nunique() - dfdevid))
print('Изменение количества событий в среднем на устройство {0:.2f}%'
      .format(100 * (df.shape[0] / df['device_id'].nunique() / dfavev -1)))
```

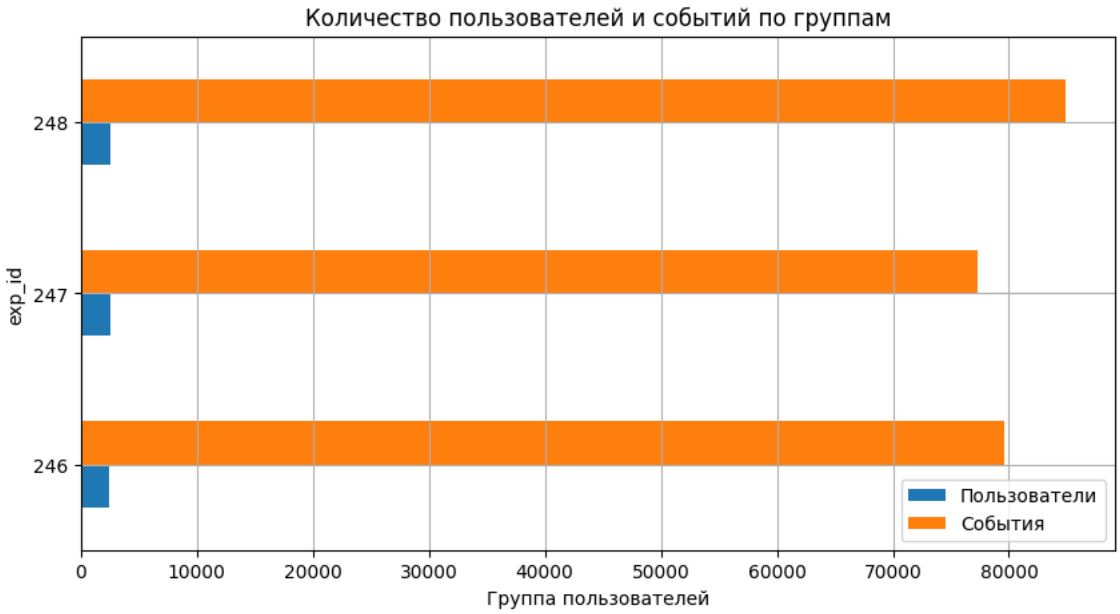
```
Изменение общего числа событий в логе: -0.82% или -1989 событий
Изменение общего числа устройств пользователей в логе: -0.17% или -13 ид устройств
Изменение количества событий в среднем на устройство -0.65%
```

Можно признать потери в исходных данных незначительными

Проверка состава групп

Теперь изучим состав исследуемых групп по количеству устрйств (пользователей). Важно, чтобы каждая из групп имела достаточно большой объём, это позволит использовать методы стат.анализа с большей степенью достоверности результата.

In [13]:
```
[
    df.groupby('exp_id')
    .agg({'device_id':'nunique','event_name':'count'})
    .rename(columns = (
    {'device_id':'Пользователи'
    ,'event_name':'События'}))
    .plot(
        kind='barh'
        ,figsize=(10,5)
        ,xlabel='Группа пользователей'
        ,title='Количество пользователей и событий по группам'
        ,grid=True
    )
];
```



Наблюдаем, что количество уникальных пользователей в каждой группе сопоставимо (около 2,5тыс), также и количество событий - около 80 тыс. в для каждой группы. Данных для исследования достаточно.

Изучение воронки событий

События в логе

Изучим, какие события представлены в предложенном лог-файле и с какой частотой они встречаются:

In [14]:
```
#строим воронку по событиям
funnel_events = (df.groupby('event_name')
                 .agg({'device_id':'count'})
                 .sort_values(by='device_id', ascending=False))
funnel_events
```

Out[14]:

event_name	device_id
MainScreenAppear	117889
OffersScreenAppear	46531
CartScreenAppear	42343
PaymentScreenSuccessful	33951
Tutorial	1010

В логе встречаются 5 видов событий:

- **MainScreenAppear** - это, вероятно, событие "отображение на устройстве главной страницы приложения", лендинг - **первый шаг в цепочке** к целевому событию
- **OffersScreenAppear** - "отображение экрана с предложениями", вероятно, каталог товаров. Это **второй шаг** в цепочке
- **CartScreenAppear** - "отображение страницы заказа", это, по всей видимости, сформированная корзина покупок. **Третий шаг цепочки**
- **PaymentScreenSuccessful** - "отображение экрана с информацией об успешной оплате", свидетельство того, что посетитель совершил целевое действие - оплатил покупку. **Четвёртый шаг**, финальный.
- **Tutorial** - по всей видимости, экран с инструкцией. Действие, как таковое, не относится к шагам, ведущим к целевому действию. Ну, или, по краней мере, не является обязательным этапом.

События в разрезе пользователей

С учётом того, что один пользователь может совершать одни и те же действия неоднократно, имеет смысл пересчитать воронку событий "по пользователям", т.е. учитывать только уникальных пользователей.

In [15]:
```
funnel_devices = (
    df.groupby('event_name', as_index=False)
    .agg({'device_id':'nunique'})
    .sort_values(by='device_id', ascending=False)
)
funnel_devices['share'] = (round(100
                                 * funnel_devices['device_id']
                                 / df['device_id'].nunique(), 2))
funnel_devices
```

Out[15]:

	event_name	device_id	share
1	MainScreenAppear	7423	98.47
2	OffersScreenAppear	4597	60.98
0	CartScreenAppear	3736	49.56
3	PaymentScreenSuccessful	3540	46.96
4	Tutorial	843	11.18

In [16]:
```
# с учётом того, что событие Tutorial не входит в цепочку обязательных событий, исключим его из воронки
funnel_devices = funnel_devices[funnel_devices['event_name'] != 'Tutorial'].reset_index(drop=True)
```

Таким образом, почти половина пользователей, которые хоть раз открывали начальную страницу приложения, доходили до успешной оплаты! А главную страницу приложения открыли не все пользователи... Как они это делают? ;)

Анализ воронки событий

Посчитаем долю пользователей, переходящих на каждый следующий шаг цепочки, а также долю пользователей, дошедших от шага "главный экран" до каждого следующего шага

In [17]:
```
# считаем конверсию от шага к шагу
funnel_devices['convers'] = (round(100
                            * funnel_devices['device_id']
                            / funnel_devices['device_id'].shift(1), 2).fillna(100))

# считаем конверсию от шага "главный экран" к каждому шагу
funnel_devices['convers_from_main_screen'] = (round(100
                                        * funnel_devices['device_id']
                                        / funnel_devices['device_id'].max(), 2))
funnel_devices
```

Out[17]:

	event_name	device_id	share	convers	convers_from_main_screen
0	MainScreenAppear	7423	98.47	100.00	100.00
1	OffersScreenAppear	4597	60.98	61.93	61.93
2	CartScreenAppear	3736	49.56	81.27	50.33
3	PaymentScreenSuccessful	3540	46.96	94.75	47.69

Расчёт показывает, что наибольшие потери пользователей происходят на этапе перехода от главного экрана к экрану с предложениями (mainscreen -> offersscreen). На этом этапе теряется больше 38% пользователей.
Наименьшие потери на последнем шаге, посетители, сформировавшие свой заказ, оплачивают его почти в 95 случаев из 100.

От главного экрана до успешной оплаты доходят 47,69% пользователей!

Визуализируем воронку событий

In [18]:
```
fig = (px.funnel(funnel_devices[['event_name', 'device_id']]
                 ,x='device_id'
                 ,y='event_name'
                 ,title='Воронка событий по уникальным пользователям')
      )
fig.show()
```

Анализ включенных в лог-файл событий показал, что в воронку событий (продаж) логично включить только 4 события из 5, т.к. 5-ое - не является обязательным в цепочке действий к целевому действию (покупке - оплате заказа). С каждым шагом воронки количество пользователей сокращается - признак

верного построения цепочки событий.
1-ый шаг воронки - самый популярный, на него попадают 98,5 из всех пользователей.
Наибольшие потери фиксируются на втором шаге.
В целом, доля пользователей, которые прошли весь путь от первого шага до финала достаточно высока - больше 47%.

Изучение результатов эксперимента

Оценка количества пользователей по группам

Данные представлены для устройств, разбитых на 3 группы. Необходимо убедиться, что ид не "задваиваются", т.е. не попали сразу в две/три группы одновременно.

In [19]:
```
print('Количество уникальных пользователей в таблице: {}'
      .format(df['device_id'].nunique()))
```

```
Количество уникальных пользователей в таблице: 7538
```

In [20]:
```
print('Сумма уникальных пользователей из каждой группы: {}'
      .format(df[df['exp_id'] == '246']['device_id'].nunique()
              + df[df['exp_id'] == '247']['device_id'].nunique()
              + df[df['exp_id'] == '248']['device_id'].nunique()
             )
     )
```

```
Сумма уникальных пользователей из каждой группы: 7538
```

Общая сумма уникальных ид устройств совпадает с суммой ид устройств по группам, значит "задвоений" нет.

Посчитаем количество уникальных устройств (пользователей) в каждой группе и сохраним в отдельных переменных

In [21]:
```
devices_246 = df.query('exp_id == "246"')['device_id'].nunique()
devices_247 = df.query('exp_id == "247"')['device_id'].nunique()
devices_248 = df.query('exp_id == "248"')['device_id'].nunique()
print("""Количество пользователей в группе 246 - {0},
      в группе 247 - {1}, в группе 248 - {2}."""
      .format(devices_246, devices_247, devices_248))
print('Соотношение количества пользователей групп A и A1 {0:.3f}%'
      .format((1 - devices_246 / devices_247) *100))
```

```
"Количество пользователей в группе 246 - 2484,
     в группе 247 - 2517, в группе 248 - 2537.
Соотношение количества пользователей групп A и A1 1.311%
```

Количество устройств (пользователей) в трёх группах сопоставимо, соотношение количества пользователей в контрольных группах (246 и 247) в районе 1%.
Пользователи не пересекаются - можно считать формирование групп для эксперимента корректным. Кроме этого критерия, необходимо определить наличие статически значимых различий междй группами.

Статистически значимые различия между группами A/A-теста

Подготовка данных

Для нахождения статистически значимых различий между контрольными группами A/A-теста необходимо сравнивать использовать z-критерий для проверки гипотез о равенстве долей в каждой из групп.
Подготовим агрегированные данные о каждой группе в одном дф

In [22]:
```
# соберём все данные о результатах эксперимента в одну сводную таблицу
# сортируем по популярности - так воронка наглядней
results_users = (df.query('event_name != "Tutorial"')
          .pivot_table(index='event_name'
                       ,columns='exp_id'
                       ,values='device_id',
                       aggfunc='nunique')
          .sort_values(by='246', ascending=False)
                )
results_users
```

Out[22]:

exp_id	246	247	248
event_name			
MainScreenAppear	2450	2479	2494
OffersScreenAppear	1542	1524	1531
CartScreenAppear	1266	1239	1231
PaymentScreenSuccessful	1200	1158	1182

In [23]:
```
## соединяем таблицу и строку с суммарными значениями
results = (pd.concat([results_users
                      , pd.DataFrame({'246':devices_246
                                      ,'247':devices_247
                                      ,'248':devices_248}
                                      ,index=['Total'])]
                     , axis=0)
           .reset_index()
           .rename(columns={'index':'event_name'})
          )

# добавляем сумму контрольных групп
results['246247'] = results[['246','247']].sum(axis = 1)
```

```
# добавляем общую сумму
results['sum_all_users'] = results[['246','247','248']].sum(axis = 1)

# считаем "популярность"
#какой процент от общего числа уникальных ид выполнил событие
results['ratio_246'] = round(results['246'] / devices_246, 3)
results['ratio_247'] = round(results['247'] / devices_247, 3)
results['ratio_248'] = round(results['248'] / devices_248, 3)
results
```

Out[23]:

	event_name	246	247	248	246247	sum_all_users	ratio_246	ratio_247	ratio_248
0	MainScreenAppear	2450	2479	2494	4929	7423	0.986	0.985	0.983
1	OffersScreenAppear	1542	1524	1531	3066	4597	0.621	0.605	0.603
2	CartScreenAppear	1266	1239	1231	2505	3736	0.510	0.492	0.485
3	PaymentScreenSuccessful	1200	1158	1182	2358	3540	0.483	0.460	0.466
4	Total	2484	2517	2537	5001	7538	1.000	1.000	1.000

Самым популярным событием является открытие главной страницы приложения (MainScreenAppear): больше 98% пользователей (устройств) совершили это событие во всех трёх группах.

Функция для сравнения выборок

Для нахождения статистически значимых отличий между контрольными группами зададим функцию, которая последовательно сравнит все события.
С учётом того, что в данном случае мы работаем с множественными сравнению (гипотезами), необходимо корректировать уровень значимости (alpha) в зависимости от количества сравнений, чтобы уменьшить вероятность допустить групповую ошибку первого рода.

In [24]:
```
#z-test
def z_test(successes_1
           ,successes_2
           ,trials_1
           ,trials_2, multi):

    #задаём aplha с учётом множественности исследований - используем метод Шидака
    # начальный уровень значимости принимаем за 0,05
    alpha = 0.05
    alpha_shidak = 1 - (1 - alpha) ** (1 / multi)

    # пропорции успехов в двух группах
    p_1 = successes_1 / trials_1
    p_2 = successes_2 / trials_2

    # пропорция успеха в комбинированной группе
    p = (successes_1 + successes_2) / (trials_1 + trials_2)

    # разница между пропорциями
    difference = p_1 - p_2

    # статистика в стандартных отклонениях стандартного нормального распределения
    z_value = (difference
               / mth.sqrt(p * (1 - p)
               * (1 / trials_1 + 1 / trials_2)))

    # задаём нормальное распределение
    distr = st.norm(0,1)

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

    print('p-value: ', round(p_value, 5))
    print('alpha: ', round(alpha, 3))
    print('alpha corrected: ', round(alpha_shidak, 4))

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

# вторая функция для вызова z-теста для выбранных групп
def run_ztest(group1, group2, multi = len(results['event_name']) - 1):
    g1 = str(group1)
    g2 = str(group2)
    for step in range(0, len(results['event_name']) - 1):
        print('Шаг воронки: {0}, группы сравнения: {1} vs {2}'
              .format(results.loc[step, 'event_name'], g1, g2))
        z_test(
            results.loc[step, g1]
            , results.loc[step, g2]
            , results.loc[4, g1]
            , results.loc[4, g2]
            , multi)
        print()
```

In [25]:
```
results.loc[0, '246'], results.loc[0, '247'], results.loc[4, '246'], results.loc[4, '247']
```

Out[25]:
```
(2450, 2479, 2484, 2517)
```

Поиск статистически значимых различий между группами A/A-теста

Сформулируем гипотезы:

- H0 - доли пользователей на разных шагах воронки одинаковы, статистически значимой разницы нет
- H1 - доли пользователей на разных шагах воронки отличны, есть значимая разница

In [26]:
```
run_ztest(246, 247)
```

```
Шаг воронки: MainScreenAppear, группы сравнения: 246 vs 247
p-value:  0.67562
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: OffersScreenAppear, группы сравнения: 246 vs 247
p-value:  0.26699
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: CartScreenAppear, группы сравнения: 246 vs 247
p-value:  0.21828
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: PaymentScreenSuccessful, группы сравнения: 246 vs 247
p-value:  0.10298
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Таким образом, ни по одному шагу воронки (событию) нет статитически значимых различий между двумя контрольными группами A/A-теста. Это также является подтверждением корректности проведённого A/A-теста.

Статистически значимые различия между группами A/B-теста

Поиск статистически значимых различий между группами A/B-теста (246/248)

Попробуем определить статистически значимые различия между контрольной группой A (246) и экспериментальной группой с изменённым шрифтом (248). Сформулируем гипотезы (они, по сути, прежние):

- H0 - доли пользователей на разных шагах воронки одинаковы, статистически значимой разницы нет
- H1 - доли пользователей на разных шагах воронки отличны, есть значимая разница

In [27]:
```
run_ztest(246, 248)
```

```
Шаг воронки: MainScreenAppear, группы сравнения: 246 vs 248
p-value:  0.34706
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: OffersScreenAppear, группы сравнения: 246 vs 248
p-value:  0.20836
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: CartScreenAppear, группы сравнения: 246 vs 248
p-value:  0.08328
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: PaymentScreenSuccessful, группы сравнения: 246 vs 248
p-value:  0.22269
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

С выбранным уровнем статистической значимости и его корректировкой по методу Шидака не удалось найти различия между пропорциями наблюдаемых выборок (групп): контрольной A и экспериментальной B (246 и 248). Другими словами, изменение шрифта не отразилось на поведение пользователей.

Поиск статистически значимых различий между группами A1/B-теста (247/248)

Сформулируем гипотезы:

- H0 - доли пользователей на разных шагах воронки одинаковы, статистически значимой разницы нет
- H1 - доли пользователей на разных шагах воронки отличны, есть значимая разница

In [28]:
```
run_ztest(247, 248)
```

```
Шаг воронки: MainScreenAppear, группы сравнения: 247 vs 248
p-value:  0.60017
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: OffersScreenAppear, группы сравнения: 247 vs 248
p-value:  0.8836
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: CartScreenAppear, группы сравнения: 247 vs 248
p-value:  0.61695
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: PaymentScreenSuccessful, группы сравнения: 247 vs 248
p-value:  0.67754
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Результат аналогичен прошлому: не удалось найти различия между пропорциями наблюдаемых выборок (групп): контрольной A и экспериментальной B (246 и 248). Изменение шрифта не отразилось на поведении пользователей группы 248 по сравнению с контрольной группой 247.

Поиск статистически значимых различий между группами A-A1/B-теста (246+247/248)

Сформулируем гипотезы (они продолжают быть неизменными):

- H0 - доли пользователей на разных шагах воронки одинаковы, статистически значимой разницы нет
- H1 - доли пользователей на разных шагах воронки отличны, есть значимая разница

In [29]:
```
run_ztest(246247, 248)
```

```
Шаг воронки: MainScreenAppear, группы сравнения: 246247 vs 248
p-value:  0.39299
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: OffersScreenAppear, группы сравнения: 246247 vs 248
p-value:  0.419
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: CartScreenAppear, группы сравнения: 246247 vs 248
p-value:  0.19819
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Шаг воронки: PaymentScreenSuccessful, группы сравнения: 246247 vs 248
p-value:  0.64521
alpha:  0.05
alpha corrected:  0.0127
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Вновь мы наблюдаем p-value значительно превосходящий как изначально выбранный уровень стат.значимости, так и корректированный.

Вывод: статистическими критериями не удалось найти значимую разницу между контрольными группами (и их комбинированным вариантом) с экспериментальной. Тест можно считать завершённым, а результаты отрицательными: изменение шрифта никак на влияет на принятие решения пользователем о выполнении действия (регистрируемого события).

Оценка выбранного уровня статистической значимости и его влияния на результаты эксперимента

При проведении статистических исследований мы проводили групповые проверки гипотез, в связи с этим, начальный уровень статистической значимости (alpha = 0.05) корректировался в сторону уменьшения методом Шидака в соответствии с количеством групп (выборок) при сравнении. Скорректированный уровень значимости alpha по методу Шидака составил 0,0127.
Однако, для формулировки финального вывода необходимо учитывать рост вероятнсти ошибки первого рода дополнительно с каждым новым исследованием! Таким образом, по методу Шидака необходимо считать, что у нас не 4 повторяющихся сравнения (по количеству шагов воронки), а 4 * количество проведённых сравнений (246vs247, 246vs248, 247vs248, 246247vs248) - ещё 4, т.е. 16!

Таким образом, получается, что поиске статистически значимых различий необходимо использовать alpha =

In [30]:
```
alpha = 0.05
multi = 16
print('Скорректированная alpha = ' , round(1 - (1 - alpha) ** (1 / multi), 5))
```

```
Скорректированная alpha =  0.0032
```

Имело бы смысл пересчитать с новым уровнем стат.значимости все исследования, но с учётом того, что во всех исследованиях p-value значительно превышает alpha и без корректировки (как результат, мы не имели оснований отвергать нулевые гипотезы о равенстве пропорций выборок), ещё большее сокращение alpha не изменит результат.
Позволю себе остановиться на простой констатации факта, о том, что alpha необходимо сократить ещё больше.

Вывод

Главная задача настоящего исследования: анализ проведённого A/B теста на основе предоставленного лог-файла.
В ходе проекта были выполнены работы:

- построение воронки событий с вычленением основной цепочки событий и отбросом несущественных событий: было определено 4 события и 1 отброшено;
- очистка лог-файла от неполных данных на осное построенного гистограммы (для исключения искажения данных): данные до 21:00 31/07/19 были отфильтрованы;
- проведён анализ состава групп (2 контрольные и 1 экспериментальная). По результатам анализа удалось установить верность корректность работы механизма распределения пользователей по группам, отсутствия "задвоений", идентичности пропорций ;
- в финальной части был осуществлён поиск статически значимых различий между контрольными группами и экспериментальной, а также между комбинацией контрольных групп и экспериментальной. Основной вывод: выборки не имеют статистически значимых отличий.
- был проведён анализ выбранного уровня стат. значимости, в результате чего были дополнительно подтверждены полученные выше результаты за счёт необходимости дополнительного снижения уровня стат. значимости.

Вывод Проведение эксперимента было выполнено корректно, группы сформированы правильно. Результат экспермента - отрицательный: не удалось установить отличия в поведении экспериментальной группы по сравнению с контрольными методами статического анализа.