Анализ поведения пользователей мобильного приложения

Описание проекта

Стартап занимается продажей продуктов питания.

Цель проекта: Небходимо разобраться, как ведут себя пользователи мобильного приложения.

Для этого решим следующие задачи:

1. Проанализируем имеющиеся данные (количество пользователей, событий, период тестирования и т.д. и их корректность)
2. Изучим воронку продаж. Узнаем, как пользователи доходят до покупки.
3. Выясним, сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах? На каких именно?
4. Исследуем результаты A/A/B-эксперимента: дизайнеры захотели поменять шрифты во всём приложении, а менеджеры испугались, что пользователям будет непривычно. Договорились принять решение по результатам A/A/B-теста. Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми. Выясним, какой шрифт лучше.

Для анализа A/A/B теста проанализируем:

- статистическую значимость разницы долей пользователей для каждого события для контрольных групп 246 И 247;
- статистическую значимость разницы долей между контрольными группами и эксперементальной группой 248;
- статистическую значимость разницы долей между объединенной контрольной группой (246+247) и эксперементальной группой 248;
- проанализируем результаты.

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.

Загрузка и обработка данных

In [1]:

```
#Загрузка бибилиотек
import pandas as pd
import datetime as dt
import scipy.stats as st
import numpy as np
import math as mth
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
#загружаем данные
data = pd.read_csv('/datasets/logs_exp.csv', sep='\t')
data.head()
```

Out[2]:

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

In [3]:
```
#переименуем столбцы
data.columns=['event_name','user_id','event_time_stamp','group']
```

In [4]:
```
data.info()
```

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

In [5]:
```
data.isna().sum()
```

Out[5]:
```
event_name          0
user_id             0
event_time_stamp    0
group               0
dtype: int64
```

In [6]:
```
print(f'Данные содержат {data.duplicated().sum()} дубликатов, что составляет {(data.duplicated(
```

```
Данные содержат 413 дубликатов, что составляет 0.17% от всех данных
```

In [7]:
```
#удаляем повтряющиеся строки
data = data.drop_duplicates().reset_index(drop=True)
```

In [8]:
```
#добавим столбец даты и времени, а также отдельный столбец дат
data['event_time'] = pd.to_datetime(data['event_time_stamp'], unit='s')
data['event_date'] = data['event_time'].dt.strftime('%Y-%m-%d')
data.head()
```

Out[8]:

	event_name	user_id	event_time_stamp	group	event_time	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

	event_name	user_id	event_time_stamp	group	event_time	event_date
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-07-25 11:48:42	2019-07-25

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   event_name        243713 non-null  object        
 1   user_id           243713 non-null  int64         
 2   event_time_stamp  243713 non-null  int64         
 3   group             243713 non-null  int64         
 4   event_time        243713 non-null  datetime64[ns]
 5   event_date        243713 non-null  object        
dtypes: datetime64[ns](1), int64(3), object(2)
memory usage: 11.2+ MB
```

Итак, первоначально данные содержали 244 126 строк с наименованием события, идентификатором пользователя, временем события, номером группы.

Данные не содержали пропущенных значений, однако в них присутствовали 413 дубликатов, что составило 0,17% всех данных. Повторяющиеся строки были удалены.

Добавили столбец с датой и временем и отдельно с датой события. В результате обработки данных имеем 243713 строк.

Исследовательский анализ данных

In [10]:
```
#посмотрим сколько событий представлены в данных
print('Данные хранят информацию о', data['event_name'].nunique(), 'событиях')
```

```
Данные хранят информацию о 5 событиях
```

In [11]:
```
data['event_name'].unique()
```

Out[11]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)
```

Неявных дубликатов в названиях событий не обнаружено.

In [12]:
```
#найдем количество пользователей
print('Количество пользователей составляет', data['user_id'].nunique(), 'человек')
```

```
Количество пользователей составляет 7551 человек
```

In [13]:
```
#количество событий, приходящихся на одного пользователя
event_per_user = data.groupby('user_id')['event_name'].agg('count').reset_index()
event_per_user.columns = ['user_id','n_events']
event_per_user
```

Out[13]:

	user_id	n_events
0	6888746892508752	1
1	6909561520679493	5
2	6922444491712477	47
3	7435777799948366	6
4	7702139951469979	137
...	...	...

	user_id	n_events
7546	9217594193087726423	9
7547	9219463515465815368	17
7548	9220879493065341500	7
7549	9221926045299980007	7
7550	9222603179720523844	59

7551 rows × 2 columns

In [14]:
```
event_per_user['n_events'].describe()
```

Out[14]:
```
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: n_events, dtype: float64
```

В среднем на одного пользователея приходится 32 события. Однако медианное значение составляет 20 событий. Максимальное значение событий на 1 пользователя составляет 2307 событий.

In [15]:
```
data['event_date'].min()
```

Out[15]:
```
'2019-07-25'
```

In [16]:
```
data['event_date'].max()
```

Out[16]:
```
'2019-08-07'
```

Данные представлены за период с 25.07.2019 по 07.08.2019.

Изучим, как меняется количество событий в зависимости от времени в разрезе групп.

In [17]:
```
data.pivot_table(index='event_date', columns='group', values='event_name', aggfunc='count').plo
plt.title('Количество событий в зависимости от времени по группам')
plt.xlabel('Дата')
plt.ylabel('Количество событий');
```

Исходя из построенного графика можно сделать вывод, что за период с 25.07.2019 по 31.07.2019 данных недостаточно. Этот период не стоит рассматривать в анализе. Найдем количество пользователей и событий, которые хотим отбросить.

In [18]:
```
#Количество пользователей
data['user_id'].nunique()-data[data['event_date']>='2019-08-01']['user_id'].nunique()
```

Out[18]: 17

In [19]:
```
#количество событий
data[data['event_date']<'2019-08-01']['event_name'].count()
```

Out[19]: 2826

In [20]:
```
#доля потерянных пользователей
round((data['user_id'].nunique()-data[data['event_date']>='2019-08-01']['user_id'].nunique())*1
```

Out[20]: 0.23

In [21]:
```
#доля потерянных событий
round(data[data['event_date']<'2019-08-01']['event_name'].count()*100/data['event_name'].count(
```

Out[21]: 1.16

Отбросив данные за указанный период мы потеряем информацию о 2826 событиях, что составит 1,16% от всех имеющихся данных. При этом потеря пользователей составит 17 человек или 0,23%.

In [22]:
```
#отбросим неактуальные данные
actual_data = data.query('event_date>="2019-08-01"')
actual_data
```

Out[22]:

	event_name	user_id	event_time_stamp	group	event_time	event_date
2826	Tutorial	3737462046622621720	1564618048	246	2019-08-01 00:07:28	2019-08-01
2827	MainScreenAppear	3737462046622621720	1564618080	246	2019-08-01 00:08:00	2019-08-01
2828	MainScreenAppear	3737462046622621720	1564618135	246	2019-08-01 00:08:55	2019-08-01

	event_name	user_id	event_time_stamp	group	event_time	event_date
2829	OffersScreenAppear	3737462046622621720	1564618138	246	2019-08-01 00:08:58	2019-08-01
2830	MainScreenAppear	1433840883824088890	1564618139	247	2019-08-01 00:08:59	2019-08-01
...	...	...	...	...	...	...
243708	MainScreenAppear	4599628364049201812	1565212345	247	2019-08-07 21:12:25	2019-08-07
243709	MainScreenAppear	5849806612437486590	1565212439	246	2019-08-07 21:13:59	2019-08-07
243710	MainScreenAppear	5746969938801999050	1565212483	246	2019-08-07 21:14:43	2019-08-07
243711	MainScreenAppear	5746969938801999050	1565212498	246	2019-08-07 21:14:58	2019-08-07
243712	OffersScreenAppear	5746969938801999050	1565212517	246	2019-08-07 21:15:17	2019-08-07

240887 rows × 6 columns

In [23]:
```
#проверим, что у нас остались пользователи во всех группах
users_per_group = actual_data.groupby('group').agg({'user_id':'nunique'})
users_per_group
```

Out[23]:

group	user_id
246	2484
247	2513
248	2537

В результате проведенного анализа было обнаружено, что за период с 25.07.2019 по 31.07.2019 данные неполные. Этот период исключен из данных для дальнейшего анализав результате чего мы потеряем информацию о 2826 событиях, что составляет 1,16% от всех имеющихся данных. Потеря пользователей составила 17 человек или 0,23% от обещго количества. Для анализа результатов A/A/B теста будут использованы данные за период с 01.08.2019 по 07.08.2019, содержащие информацию о 240887 событиях, совершенных 7534 пользователями.

Изучение воронки событий

In [24]:
```
#посчитаем частоту событий
events = actual_data['event_name'].value_counts()
```

In [25]:
```
#посчитаем количество пользователей на каждом этапе
events = (
    actual_data.groupby('event_name')
    .agg({'user_id': ['count', 'nunique']})
    .sort_values(by=('user_id','count'), ascending=False).reset_index()
)
events.columns = ['event_name','events', 'users']
events
```

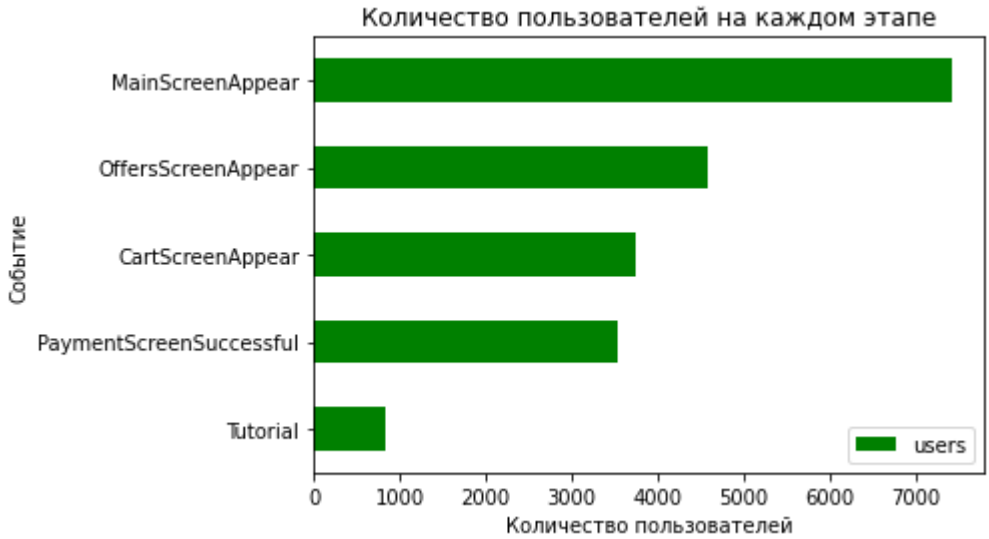
Out[25]:

	event_name	events	users
0	MainScreenAppear	117328	7419
1	OffersScreenAppear	46333	4593
2	CartScreenAppear	42303	3734
3	PaymentScreenSuccessful	33918	3539

	event_name	events	users
4	Tutorial	1005	840

In [26]:
```
plt.figure(figsize=(12, 6))
events.sort_values(by='users').plot(kind='barh', x='event_name', y='users', color='green')
plt.xlabel('Количество пользователей')
plt.ylabel('Событие')
plt.title('Количество пользователей на каждом этапе');
```

```
<Figure size 864x432 with 0 Axes>
```



Всего имеем 5 событий, 4 из которых проходят в следующей последовательности: MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful

Событие Tutorial ('Обращение к руководству пользователя') может появиться на любом шаге

In [27]:
```
#находим долю пользователей, которые хоть раз совершали событие
events['ratio'] = round(events['users']/(actual_data['user_id'].nunique())*100,2)
events
```

Out[27]:

	event_name	events	users	ratio
0	MainScreenAppear	117328	7419	98.47
1	OffersScreenAppear	46333	4593	60.96
2	CartScreenAppear	42303	3734	49.56
3	PaymentScreenSuccessful	33918	3539	46.97
4	Tutorial	1005	840	11.15

Главный экран отобразился у 98,5% пользователей. 61% пользователей видят экран с предложением, 49,6% доходят до оплаты, однако успешная оплата проходит только 47%. 2,4% пользователей не могут оплатить. К руководству пользователя обаращается лишь 11,2%. Данная операция может быть осуществлена на любом шаге и скорее всего будет связана с проблема осуществления той или иной операции. Это событие мы не будем учитывать при расчете воронки.

In [28]:
```
events = events.loc[events['event_name']!="Tutorial"]
events['conversion'] = round((events['users'].pct_change()+1)*100,2)
events
```

```
/tmp/ipykernel_234/2010021621.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  events['conversion'] = round((events['users'].pct_change()+1)*100,2)
```

Out[28]:

	event_name	events	users	ratio	conversion
0	MainScreenAppear	117328	7419	98.47	NaN
1	OffersScreenAppear	46333	4593	60.96	61.91
2	CartScreenAppear	42303	3734	49.56	81.30
3	PaymentScreenSuccessful	33918	3539	46.97	94.78

38,1% не проходят дальше главного экрана. Еще 18,7% от перешедших не доходяд до оплаты. 5,22% теряются в момент оплаты. До успешной оплаты доходят только 47,7% пользователей.

In [29]:

```
fig = px.funnel(events, x='users', y='event_name', title = 'Воронка событий')
fig.show()
```

Анализ результатов исследования

In [30]:

```
#посчитаем количество пользователей по группам
users_per_group = actual_data.groupby('group').agg({'user_id':'nunique'})
users_per_group
```

Out[30]:

	user_id
group	
246	2484

group	user_id
247	2513
248	2537

In [32]:
```
# Проверим пользователей, которые могли участвовать в двух или нескольких группах одновременно:
actual_data.groupby('user_id').agg({'group':'nunique'}).query('group > 1')
```

Out[32]:

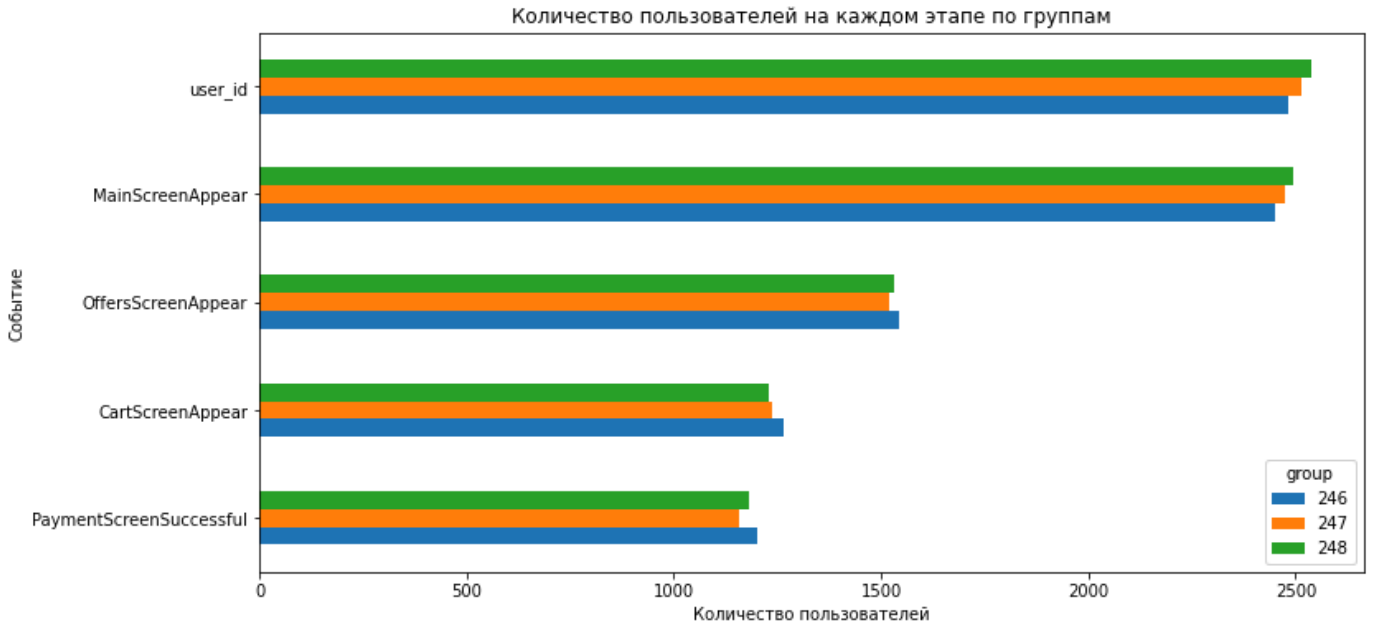
user_id	group

In [33]:
```
#найдем количество пользователей для каждого события с разбивкой по группам
grouped_data = (
    actual_data.query('event_name!="Tutorial"')
    .pivot_table(index='group',columns='event_name', values='user_id', aggfunc='nunique')
)

#добавляем в таблицу информацию об общем количестве пользователей в каждой группе
grouped_data = users_per_group.merge(grouped_data, on='group', how='left').T.sort_values(by=246

#считаем аналогичные показатели суммарно для групп A/A теста
grouped_data['246_247'] = grouped_data[246]+grouped_data[247]

#найдем долю пользователей, совершивших событие в каждой группе
grouped_data['conv_246'] = round(grouped_data[246]/grouped_data[246][0],2)
grouped_data['conv_247'] = round(grouped_data[247]/grouped_data[247][0],2)
grouped_data['conv_248'] = round(grouped_data[248]/grouped_data[248][0],2)
grouped_data['conv_246_247'] = round(grouped_data['246_247']/grouped_data['246_247'][0],2)
grouped_data
```

Out[33]:

group	246	247	248	246_247	conv_246	conv_247	conv_248	conv_246_247
user_id	2484	2513	2537	4997	1.00	1.00	1.00	1.00
MainScreenAppear	2450	2476	2493	4926	0.99	0.99	0.98	0.99
OffersScreenAppear	1542	1520	1531	3062	0.62	0.60	0.60	0.61
CartScreenAppear	1266	1238	1230	2504	0.51	0.49	0.48	0.50
PaymentScreenSuccessful	1200	1158	1181	2358	0.48	0.46	0.47	0.47

In [34]:
```
grouped_data.sort_values(by=246).plot(kind='barh', y=[246,247,248], figsize=(12,6))
plt.xlabel('Количество пользователей')
plt.ylabel('Событие')
plt.title('Количество пользователей на каждом этапе по группам');
```



In [35]:

```
#напишем функцию для проверки гипотезы о равенстве долей

def test_diff(first_group, second_group, events, alpha, n):

    for i in range(1,len(first_group)):

    # пропорция успехов в первой группе:
        p1 = first_group[i]/first_group[0]

    # пропорция успехов во второй группе:
        p2 = second_group[i]/second_group[0]

    # пропорция успехов в комбинированном датасете:
        p_combined = (first_group[i] + second_group[i]) / (first_group[0] + second_group[0])

    # 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/first_group[0] + 1/s

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
        distr = st.norm(0, 1)
        p_value = (1 - distr.cdf(abs(z_value))) * 2

        alpha_bonf = alpha/n  #поправка на множественность гипотез
        print('{} p-значение: {}'.format(events[i], p_value))

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

A/A тест

Проверим, находят ли статистические критерии разницу между выборками 246 и 247.

Для каждого события будем проверять гипотезу:
H0: доли пользователей в группах 246 и 247, совершивших событие, равны;
H1: доли пользователей в группах 246 и 247, совершивших событие, не равны.

Уровень значимости alpha=0.05. Т.к. мы проверяем гипотезу для каждого из 4 событий для 4 комбинаций групп, необходимо сделать поправку на множественность гипотез (n=16).

In [36]:
```
test_diff(grouped_data[246], grouped_data[247], grouped_data.index, 0.05, 16)
```

```
MainScreenAppear p-значение: 0.7570597232046099
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
OffersScreenAppear p-значение: 0.2480954578522181
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
CartScreenAppear p-значение: 0.22883372237997213
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
PaymentScreenSuccessful p-значение: 0.11456679313141849
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

В группах 246 и 247 для всех событий нет оснований считать доли пользователей разными.

Анализ результатов A/A/B теста

Проверим для каждого события, находят ли статистические критерии разницу между выборкой 248 и каждой из выборок 246, 247, 246+247.

Для каждого события будем проверять гипотезу:
H0: доли пользователей, совершивших событие, равны;
H1: доли пользователей, совершивших событие, не равны.

Уровень значимости alpha=0.05.
Т.к. мы проверяем гипотезу для каждого из 4 событий в 4 выборках, необходимо сделать поправку на множественность гипотез (n=16).

In [37]:
```
test_diff(grouped_data[246], grouped_data[248],grouped_data.index, 0.05, 16)
```

```
MainScreenAppear p-значение: 0.2949721933554552
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
OffersScreenAppear p-значение: 0.20836205402738917
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
CartScreenAppear p-значение: 0.07842923237520116
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
PaymentScreenSuccessful p-значение: 0.2122553275697796
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

In [38]:
```
test_diff(grouped_data[247], grouped_data[248],grouped_data.index, 0.05, 16)
```

```
MainScreenAppear p-значение: 0.4587053616621515
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
OffersScreenAppear p-значение: 0.9197817830592261
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
CartScreenAppear p-значение: 0.5786197879539783
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
PaymentScreenSuccessful p-значение: 0.7373415053803964
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

In [39]:
```
test_diff(grouped_data['246_247'], grouped_data[248],grouped_data.index, 0.05,16)
```

```
MainScreenAppear p-значение: 0.29424526837179577
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
OffersScreenAppear p-значение: 0.43425549655188256
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
CartScreenAppear p-значение: 0.18175875284404386
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
PaymentScreenSuccessful p-значение: 0.6004294282308704
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Все 12 тестов показали отсутствие статистически значимых различий между долями в группах.

Вывод

В результате анализа представленных данных можно сказать следующее:

1. Первоначальные данные содержали 244 126 строк с наименованием события, идентификатором пользователя, временем события, номером группы за период с 25.07.2019 по 07.08.2019.
2. На этапе предварительной обработки данных были выявлены и удалены 413 дублирующихся строк, что составило 0,17% всех данных.
3. Исследование данных показало, что за период с 25.07.2019 по 31.07.2019 данные являются неполными. Этот период исключен из данных для дальнейшего анализа, в результате чего мы потеряли информацию о 2826 событиях, что составляет 1,16% от всех имеющихся данных. Потеря пользователей составила 17 человек или 0,23% от обещго количества. В результате обработки данных для анализа результатов A/A/B теста использованы данные за период с 01.08.2019 по 07.08.2019, содержащие информацию о 240887 событиях, совершенных 7534 пользователями.
4. Анализ воронки событий показал: Всего имеем 5 событий, 4 из которых проходят в следующей последовательности:
   MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful
   Событие Tutorial ('Обращение к руководству пользователя') может появиться на любом шаге. Это событие мы не учитываем при анализе воронки.
   На главный экран (MainScreenAppear) зашло 98,7% пользователей. Возможно у 1,3% появились проблемы со входом в приложение. Больше всего пользователей теряется на шаге перехода на страницу товаров (OffersScreenAppear). Сюда доходит лишь 62% пользователей с предыдущего шага (61% от всех пользователей), из них 81,3% (49,6% от общего числа пользователей) доходят до формы оплаты, однако успешная оплата проходит не у всех, а только у 94,8%. Часть пользователей не проходит успешно оплату.
5. Для проведение A/A/B теста было задействовано 2 контрольных группы (246 и 247) и одна эксперементальная (группа 248). Общее количество пользователей составило 7534 человека. В том числе: 246 - 2484 чел., 247 - 2513 чел., 248 - 2537 чел. Пересечений пользователей между группами обнаружено не было.
   В результате анализа было проведено 16 тестов для проверки гипотезы о равенстве долей пользователей в группах для каждого события (кроме "Tutorial"):
   Для пар групп 246 и 247, 246 и 248, 247 и 248, 246+247 и 248 проверена гипотеза:
   H0: доли пользователей в группах, совершивших событие, равны
   H1: доли пользователей в группах, совершивших событие, не равны.
   Уровень статистической значимости (alpha=0,05) учитывался с поправкой на множественность гипотез (alpha/16).
   Увеличение значимости до alpha=0,1 в нашем случае не изменит результатов теста (alpha_bonf=0,1/16=0,00625). Увеличение уровня значимости приводит к увеличению случаев, когда можно ошибочно отклонить нулевую гипотезу при условии, что она верна. Поэтому мы оставим выбранный изначально уровень 5%.
   Итак, по результатм теста можно сказать, что изменение шрифта не оказало статистичеки значимого влияния на поведение пользователей.

In [ ]: