Тема - Анализ клиентского поведения в мобильном приложении

Содержание

Описание проекта

Мы работаем в **стартапе, который продаёт продукты питания**. Нам необходимо разобраться, как ведут себя пользователи нашего мобильного приложения.

- Изучим воронку продаж. Узнаем, как пользователи доходят до покупки. Сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах? На каких именно?

- После этого исследуем результаты A/A/B-эксперимента. Наши 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.

Часть 1. Подготовка данных

Для анализа нам предоставлен файл - `logs_exp.csv` . Откроем и изучим общую информацию из него.

Загрузим данные и подключим необходимые библиотеки

In [1]:
```
import pandas as pd
import seaborn as sns
from datetime import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import math as mth
from pandas.plotting import register_matplotlib_converters
import scipy.stats as st
import plotly.express as px

import warnings
# конвертеры, которые позволяют использовать типы pandas в matplotlib
register_matplotlib_converters()
```

In [2]:
```
try:
    df = pd.read_csv('/datasets/logs_exp.csv', sep ='\t')
except:
    df = pd.read_csv('logs_exp.csv', sep='\t')
```

Познакомимся с данными, хранящимися в файле с логами

Посмотрим общую информацию о датафрейме

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

Out[3]:

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

В датафрейме мы имеем 4 столбца -

- **EventName** - наименование события
- **DeviceIDHash** - уникальный идентификатор пользователя (устройства)
- **EventTimestamp** - время события
- **ExpId** - номер эксперимента (для A/A/B - теста)

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

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

In [5]:
```
df.shape
```

Out[5]:
```
(244126, 4)
```

Как мы видим, в нашем датафрейме **244 126 строк** и **4 столбца**.

Переименуем столбцы для удобного обращения к датафрейму

In [6]:
```
df =(
    df.rename(columns = {'EventName':'event_name','DeviceIDHash':'user_id','EventTimestamp
)
```

Изучим какие уникальные значения есть в столбцах с категориями

In [7]:
```
df['event_name'].unique()
```

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

В нашем датасете мы имеем 5 различных типов событий -

- **MainScreenAppear** - появление главного меню (основной страницы)

- **PaymentScreenSuccessful** - появление страницы об успешной оплате
- **CartScreenAppear** - появление страницы с корзиной
- **OffersScreenAppear** - появление страницы с предложениями
- **Tutorial** - обучение (возможно страница с обучением)

In [8]:
```
df['group'].unique()
```

Out[8]:
```
array([246, 248, 247], dtype=int64)
```

В столбце `group` (ранее - `ExpId`) - у нас представлены 3 группы для проведения A/A/B теста

- **246** и **247** — контрольные группы - `A / A`
- **248** — экспериментальная - `B`

Проверим данные на пропуски

In [9]:
```
print(df.isna().mean())
```

```
event_name          0.0
user_id             0.0
event_unix_time     0.0
group               0.0
dtype: float64
```

Пропусков в данных - нет

Проверим данные на явные дубликаты

In [10]:
```
print(f"Количество дубликатов в датафрейме - {df.duplicated().sum()}")
```

```
Количество дубликатов в датафрейме - 413
```

Удалим обнаруженные дубликаты -

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

Посмотрим на количество дубликатов после удаления

In [12]:
```
print(f"Количество дубликатов в датафрейме - {df.duplicated().sum()}")
```

```
Количество дубликатов в датафрейме - 0
```

Добавим столбец даты и времени и отдельный столбец дат

In [13]:
```
# Переведем время unix time в формтат datetime
df['event_datetime'] = pd.to_datetime(df['event_unix_time'],unit='s')
```

In [14]:
```
#Создадим столбце только с датой события (без времени)
df['event_date'] = pd.to_datetime(df['event_datetime']).dt.date
```

Посмотрим на получившийся датафрейм -

In [15]:
```
df
```

Out[15]:

	event_name	user_id	event_unix_time	group	event_datetime	event_date

	event_name	user_id	event_unix_time	group	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
...	...	...	...	...	...	...
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

243713 rows × 6 columns

Проверим, не попали ли у нас оди и те же пользователи в разные группы

In [16]:

```
#создадим переменную с пользователями в группе 246

a_users = df[df['group']==246]['user_id'].unique()
print('Количество уникальных пользователей группы A -',len(a_users))
print()
#создадим переменную с пользователями в группе 247
b_users = df[df['group']==247]['user_id'].unique()
print('Количество уникальных пользователей группы B -',len(b_users))
print()

#создадим переменную с пользователями в группе 248
c_users = df[df['group']==248]['user_id'].unique()
print('Количество уникальных пользователей группы B -',len(b_users))
print()
#обозначим переменную для подсчета одинаковых пользователей

def same_check(A,B):
    same_users = 0
    same_AB = []
    for user in A:
        if user in B:
            same_users+=1
            same_AB.append(user)
    print('Количество повторяющихся пользователей в группе A и B -', same_users)
    print()
    print('ID пользователей, которые присутсвуют в группах A и B')
    print(same_AB)
    print('-----------------------------------------------')

print('Проверка для группы 246 и 247:')
same_check(a_users,b_users)
print('Проверка для группы 247 и 248:')
same_check(b_users,c_users)
print('Проверка для группы 246 и 248:')
same_check(a_users,c_users)
```

```
Количество уникальных пользователей группы A - 2489
```

```
Количество уникальных пользователей группы B - 2520

Количество уникальных пользователей группы B - 2520

Проверка для группы 246 и 247:
Количество повторяющихся пользователей в группе A и B - 0

ID пользователей, которые присутсвуют в группах A и B
[]
--------------------------------------------
Проверка для группы 247 и 248:
Количество повторяющихся пользователей в группе A и B - 0

ID пользователей, которые присутсвуют в группах A и B
[]
--------------------------------------------
Проверка для группы 246 и 248:
Количество повторяющихся пользователей в группе A и B - 0

ID пользователей, которые присутсвуют в группах A и B
[]
--------------------------------------------
```

Как мы видим, во всех группах присутствуют уникальные пользователи

Вывод по первой части анализа

Мы **изучили** исходный файл `logs_exp.csv` и получили следующую информацию -

- В нашем датафрейме присутствует информация о **5 (пяти) типах событий** -
  1. Появление главного экрана
  2. Экран корзины
  3. Экран успешной покупки
  4. Экран предложения
  5. Экран обучения

- В нашем датафрейме присутствует информация о 3 тестовых группах для A/A/B теста - 246, 247, 248

и **провели предобработку датафрейма** -

- Переименовали столбцы для удобства использования датафрейма
- Проверили датафрейм на пропуски - пропусков не обнаружено
- Проверили датафрейм на явные дубликаты - Обнаружили 413 дубликатов - **удалили их**
- Создали столбцы с датой и временем и датой события
- Проверили, что во всех группах присутствуют уникальные пользователи

Часть 2. Изучение и проверка данных

Проведем анализ предобработанных данных - узнаем количество событий

In [17]:
```
df.info()
```

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

```
 0   event_name       243713 non-null  object
 1   user_id          243713 non-null  int64
 2   event_unix_time  243713 non-null  int64
 3   group            243713 non-null  int64
 4   event_datetime   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
```

In [18]:
```
df.shape
```

Out[18]:
```
(243713, 6)
```

У нас есть данные по **243 713** событиям (поделенным на 5 типов)

Посчитаем количество уникальных пользователей

In [19]:
```
print(df['user_id'].nunique())
```

```
7551
```

Количество уникальных пользователей в нашем датасете - 7 551 человек

Посчитаем среднее количество событий на пользователя

In [20]:
```
print(round(df['user_id'].count()/df['user_id'].nunique(),2))
```

```
32.28
```

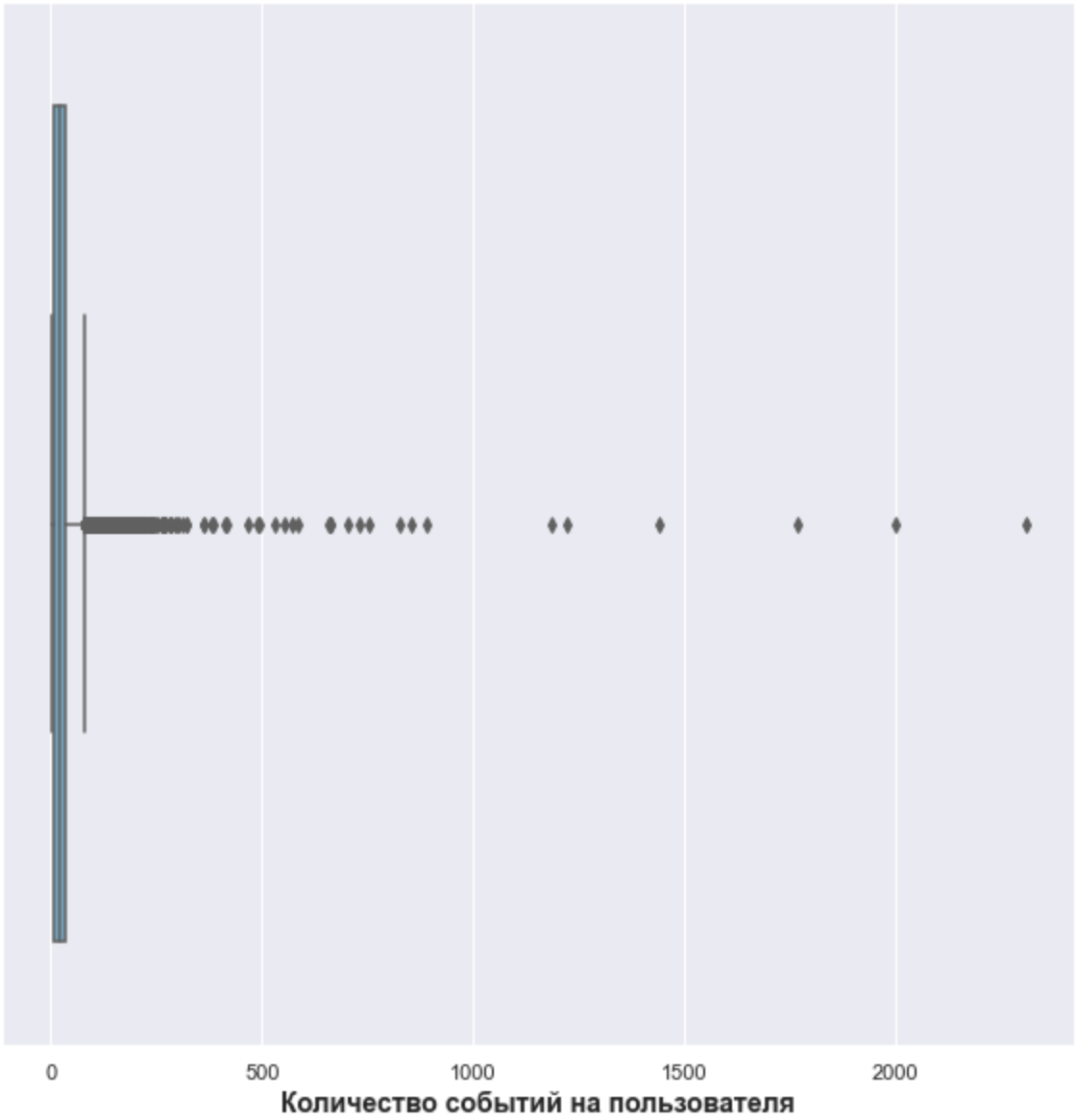
Среднее количество - 32 события на пользователя

Посмотрим на медианные значения (среднее может сильно отличаться из-за выбросов)

In [21]:
```
events = df.groupby('user_id', as_index = False)['event_unix_time'].count()
events.columns = (['user_id', 'events_count'])
print(events.sort_values('events_count', ascending = False))
```

```
                user_id  events_count
5116  6304868067479728361          2307
147    197027893265565660          1998
3714  4623191541214045580          1768
5590   693251704570305408 7          1439
1391  1754140665440434215          1221
...                    ...           ...
6013   7399061063341528729             1
2356   2968164493349205501             1
6575   8071397669512236988             1
311     425817683219936619             1
0         6888746892508752             1

[7551 rows x 2 columns]
```

In [22]:
```
sns.set(rc={"figure.figsize":(10, 10)})
ax = sns.boxplot(x=events['events_count'], palette='Blues')
ax = ax.set_xlabel('Количество событий на пользователя', fontsize= 14, fontweight='bold')
```



Как мы видим на графике `boxplot` - много выбросов - количество событий на пользователя больше 1 000 (тысячи)

In [23]:
```
events.describe()
```

Out[23]:

	user_id	events_count
count	7.551000e+03	7551.000000
mean	4.677319e+18	32.275593
std	2.655343e+18	65.154219
min	6.888747e+15	1.000000
25%	2.397700e+18	9.000000
50%	4.688022e+18	20.000000
75%	7.007353e+18	37.000000
max	9.222603e+18	2307.000000

Медианное значение - 20 событий на пользователя.

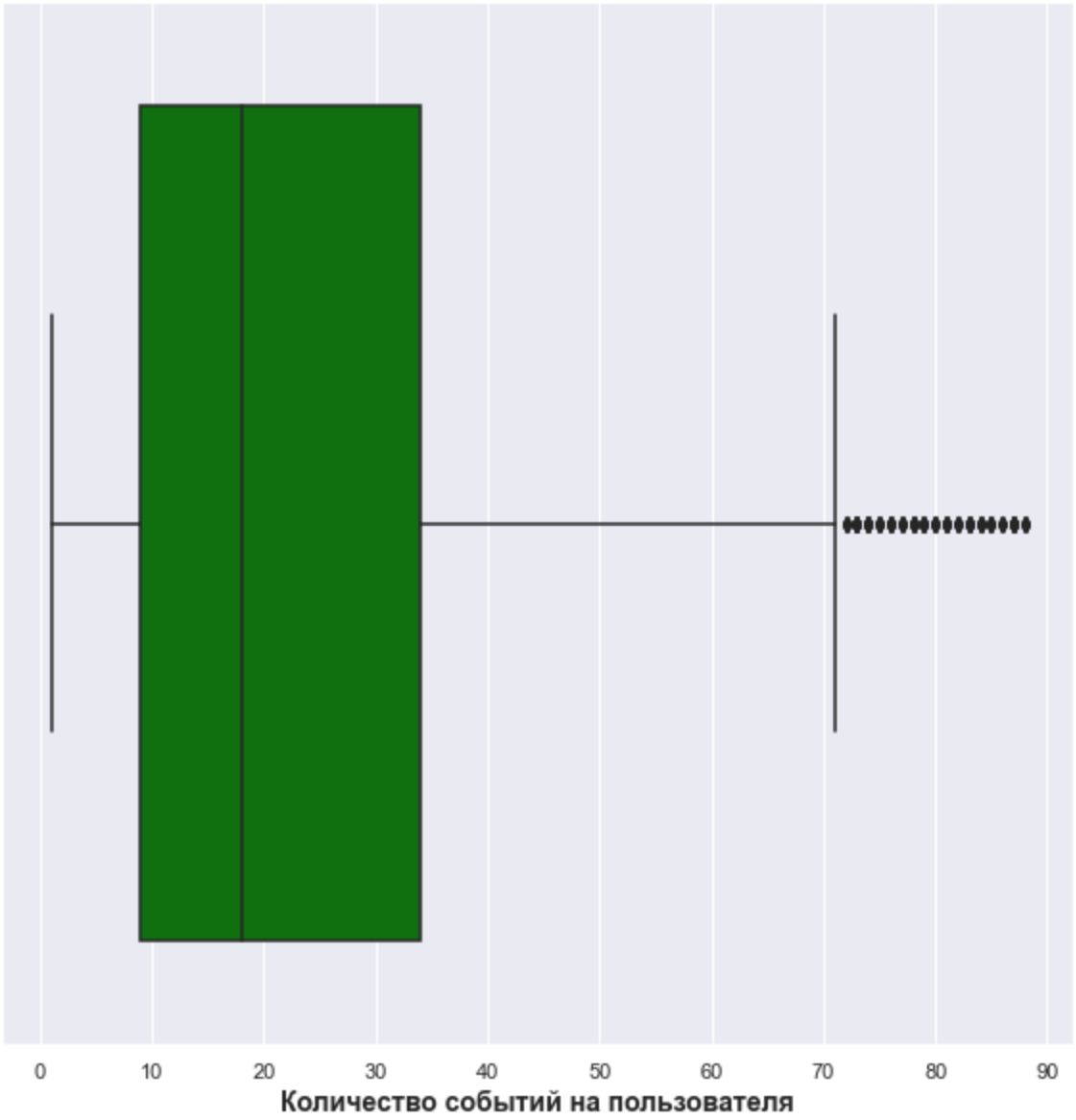
Посмотрим на 95й перцентиль датасета, куда попадает 95% пользователей

In [24]:
```
np.percentile(events.events_count,95)
```

Out[24]:
```
89.0
```

Построим **BoxPlot с учетом 95 % перцентиля**, приняв его за верхнюю границу

In [25]:
```
sns.set(rc={"figure.figsize":(10, 10)})

ax = sns.boxplot(x=events[events['events_count'] < np.percentile(events.events_count,95)]
ax = ax.set_xlabel('Количество событий на пользователя', fontsize= 14, fontweight='bold')
increments = 10
ax = plt.gca()
ax = ax.xaxis.set_ticks(np.arange(0, 100, increments))
```



Как мы видим, **у 75% пользователей в датасете до 37 событий на человека**

медианное значение - 19 событий на человека

Посмотрим как распределяются события внутри каждой из тестовой групп (посмотрим "воронку")

In [26]:
```
funnel = (
    pd.pivot_table(df,values='event_unix_time',index=['group','event_name'],aggfunc='count
    .sort_values(['group','event_unix_time'],ascending =[True,False])
```

```
)
funnel.columns = (['Number of Events'])
funnel
```

Out[26]:

group	event_name	Number of Events
246	MainScreenAppear	38249
	OffersScreenAppear	14904
	CartScreenAppear	14798
	PaymentScreenSuccessful	11912
	Tutorial	318
247	MainScreenAppear	39677
	OffersScreenAppear	15341
	CartScreenAppear	12548
	PaymentScreenSuccessful	10039
	Tutorial	345
248	MainScreenAppear	41175
	OffersScreenAppear	16563
	CartScreenAppear	15322
	PaymentScreenSuccessful	12167
	Tutorial	355

Как мы видим, **во всех тестовых группах одинаковый порядок событий (по убыванию)**

1. Появление основного экрана
2. Появление экрана с преложениями
3. Появление экрана с корзиной
4. Появление экрана с успешной покупкой
5. Появление экрана с обучением

Посмотрим какие даты есть в нашем датафрейме по тестовым группам

In [27]:

```
group = df['group'].unique()
for i in group:
    print(f"Название группы - {i}")
    print(f"Минимальная дата - {df[df['group']==i]['event_date'].min()}")
    print(f"Максимальная дата - {df[df['group']==i]['event_date'].max()}")
    print('-----------------------------------------\n')
```

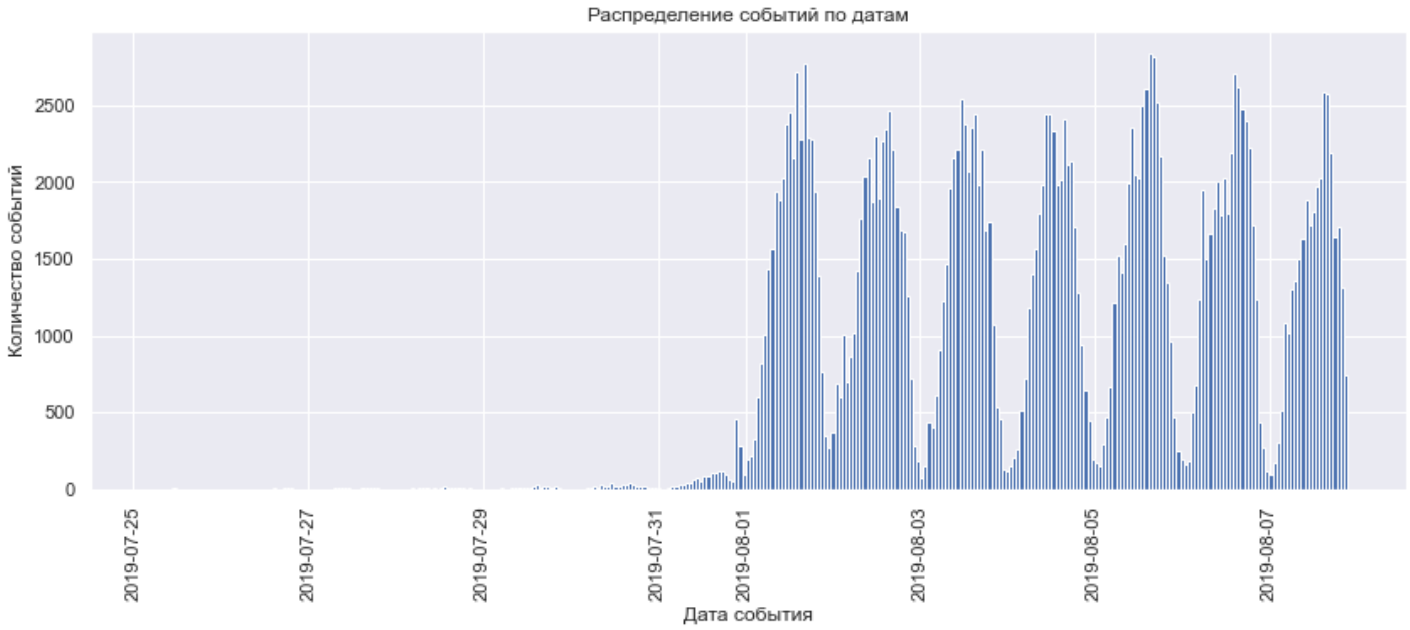
```
Название группы - 246
Минимальная дата - 2019-07-25
Максимальная дата - 2019-08-07
-----------------------------------------

Название группы - 248
Минимальная дата - 2019-07-25
Максимальная дата - 2019-08-07
-----------------------------------------
```

```
Название группы - 247
Минимальная дата - 2019-07-25
Максимальная дата - 2019-08-07
-----------------------------------------
```

Как мы видим, **по всем 3м группам мы имеем данные за перид с 25 июля 2019 по 08 августа 2019**

Построим гистограмму по дате и времени

In [28]:
```
ax = df['event_datetime'].hist(figsize=(14,5),bins=14*24, xrot=90)
ax.set_title('Распределение событий по датам')
ax.set_ylabel("Количество событий")
ax = ax.set_xlabel("Дата события")
```



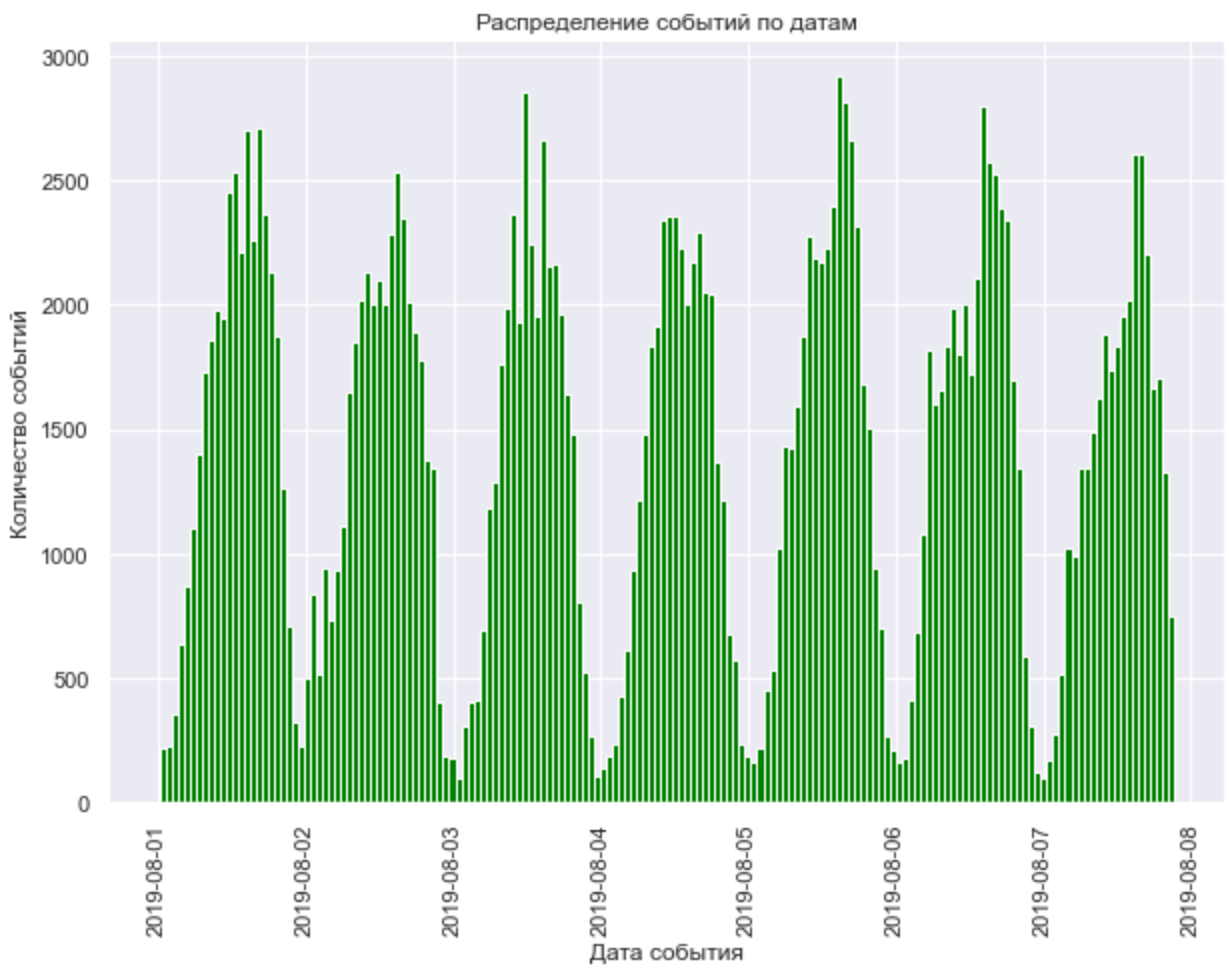
Как мы видим, **основное количество событий приходится на период с 01 по 07 августа 2019 года**.

Отбросим более старые данные -

In [29]:
```
df_filtered = df[df['event_datetime'] > '2019-08-01']
```

Построим гистограмму по "отфильрованному" датасету -

In [30]:
```
ax = df_filtered['event_datetime'].hist(figsize=(10,7),bins=7*24, xrot=90, color = 'green'
ax.set_title('Распределение событий по датам')
ax.set_ylabel("Количество событий")
ax = ax.set_xlabel("Дата события")
```



Посчитаем количество отброшенных событий

In [31]:
```
print('Количество отброшенных событий - ',
      df[df['event_datetime'] < '2019-08-01']['event_name'].count())

print('Процент отброшенных событий -',
      round(
          df[df['event_datetime'] < '2019-08-01']['event_name'].count()*100/df['event_name
      '%'
     )
```

```
Количество отброшенных событий -  2826
Процент отброшенных событий - 1.16 %
```

Посчитаем количество отброшенных Пользователей

In [32]:
```
all_users = df['user_id'].nunique()
filtered_users = df_filtered['user_id'].nunique()
lost_users = all_users - filtered_users
print(f'Количество отброшенных пользователей - {lost_users}')
print(f'Процент отброшенных пользователей - {round(lost_users*100/all_users,2)}%')
```

```
Количество отброшенных пользователей - 17
Процент отброшенных пользователей - 0.23%
```

Проверим, что мы сохранили данные по всем 3-м экспериментальным группам

In [33]:
```
funnel_filtered = (
    pd.pivot_table(df_filtered,values='event_unix_time',index=['group','event_name'],aggfu
```

```
    .sort_values(['group','event_unix_time'],ascending =[True,False])
)
funnel_filtered.columns = (['Number of Events'])
funnel_filtered
```

Out[33]:

		Number of Events
group	**event_name**	
246	**MainScreenAppear**	37676
	OffersScreenAppear	14767
	CartScreenAppear	14690
	PaymentScreenSuccessful	11852
	Tutorial	317
247	**MainScreenAppear**	39090
	OffersScreenAppear	15179
	CartScreenAppear	12434
	PaymentScreenSuccessful	9981
	Tutorial	338
248	**MainScreenAppear**	40562
	OffersScreenAppear	16387
	CartScreenAppear	15179
	PaymentScreenSuccessful	12085
	Tutorial	350

Как мы видим, данные сохранились по всем 3 группам,

посчитаем их Общее количество

In [34]:

```
funnel_filtered = (
    pd.pivot_table(df_filtered,values='event_unix_time',index=['group'],aggfunc='count')
    .sort_values(['group','event_unix_time'],ascending =[True,False])
)
funnel_filtered.columns = (['Number of Events'])
funnel_filtered.reset_index()
```

Out[34]:

	group	Number of Events
0	246	79302
1	247	77022
2	248	84563

Вывод по 2-й части анализа

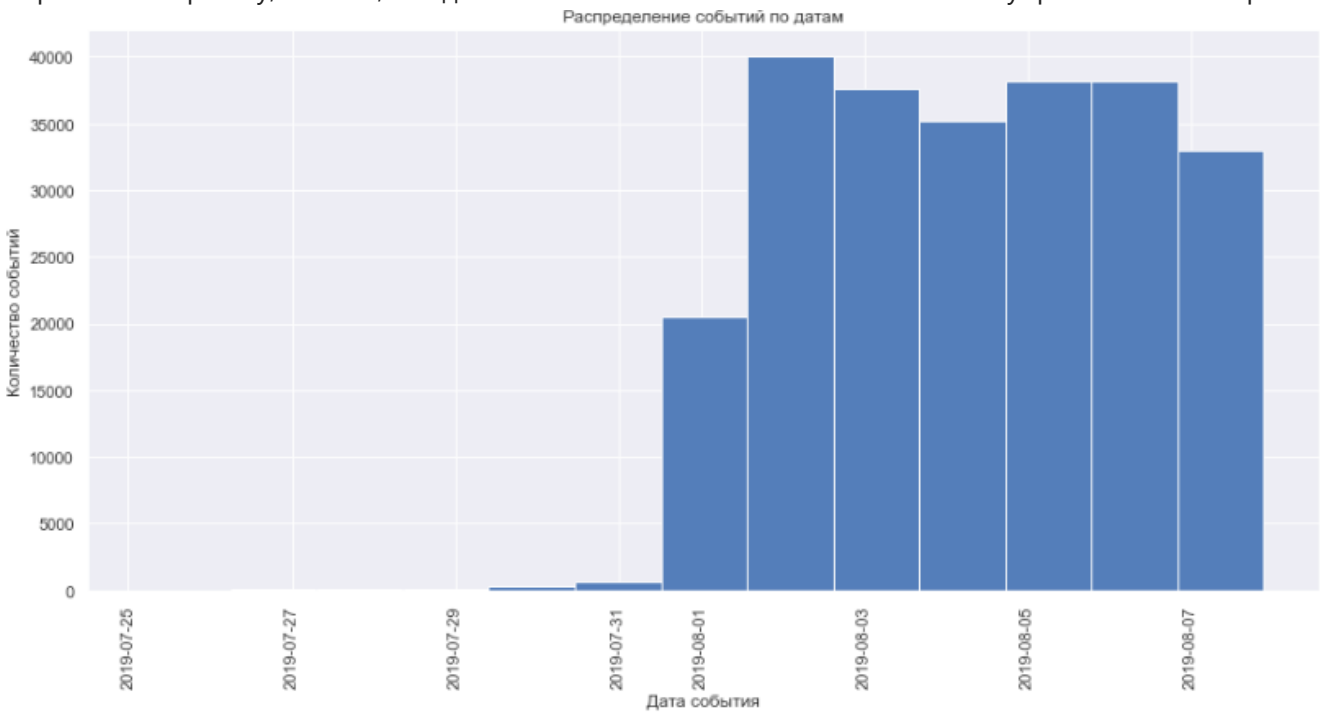
Мы подробно изучили датафрейм и выяснили -

- В датафрейме информация об 7551 уникальных пользователях
- Среднее количество событий на пользователя - 32 шт.

- Медианное значение событий на пользователя - 20 шт.
- В каждой группе есть 5 (пять) типов событий - Основной экран, Экран с предложениями, экран с корзиной, экран с успешной покупкой, экран обучения

Изначально мы имели данные за период с 25 июля 2019 по 08 августа 2019.

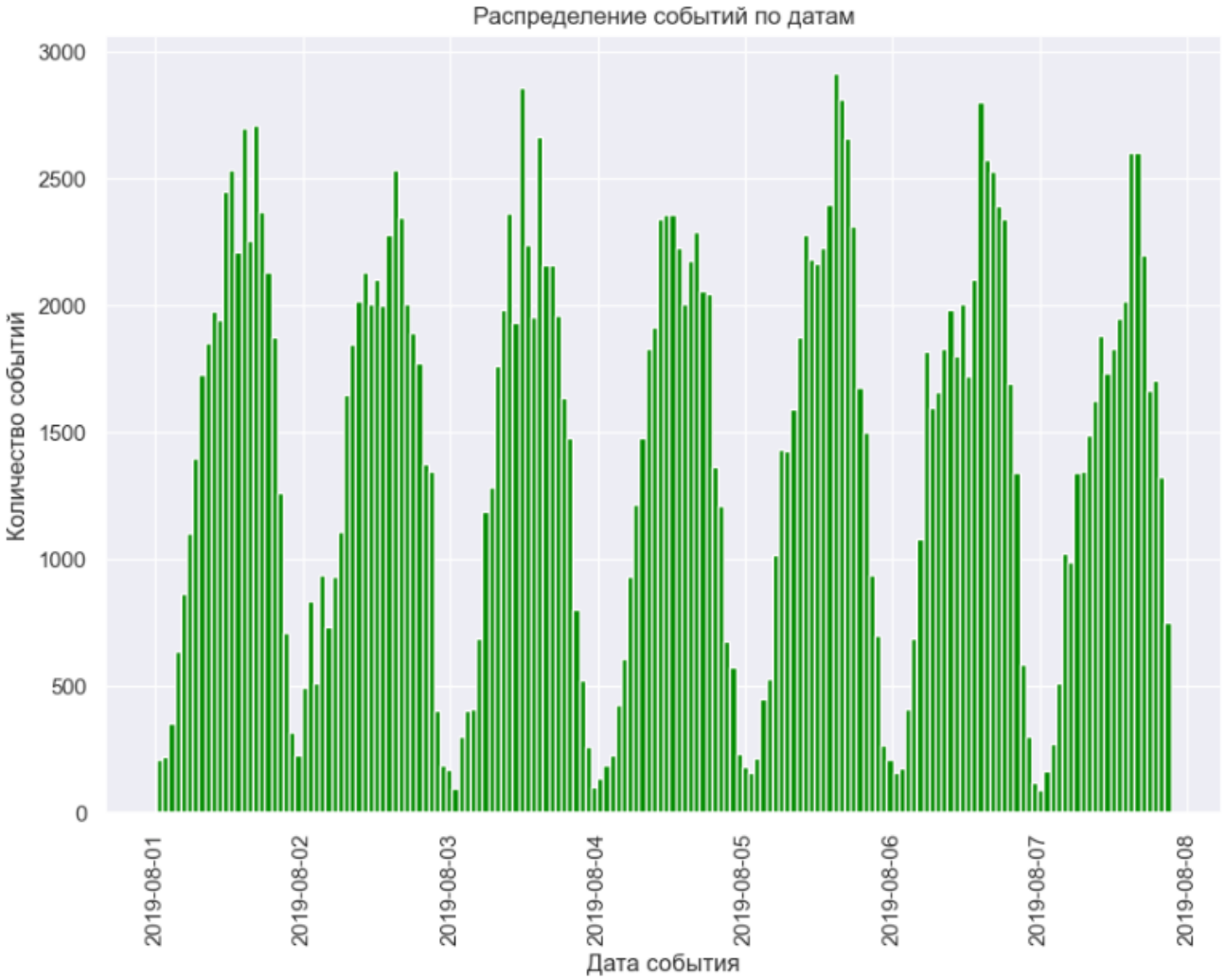
Но построив гистограмму, поняли, что данных с 25 по 31 июля 2019 не хватает - и убрали их из выборки



В итоге, мы отбросили следующие данные -

- Количество отброшенных пользователей - 17
- Процент отброшенных пользователей - 0.23%

Итоговая гистограма с 01 по 08 августа 2019



Часть 3. Изучим воронку событий

Посмотрим, какие события есть в логах и как часто они встречаются

In [35]:
```
events = (
    df_filtered.groupby('event_name', as_index=False)['user_id']
    .count()
    .sort_values('user_id', ascending = False)
)
events.columns = (['event_name', 'num_of_events'])
events
```

Out[35]:

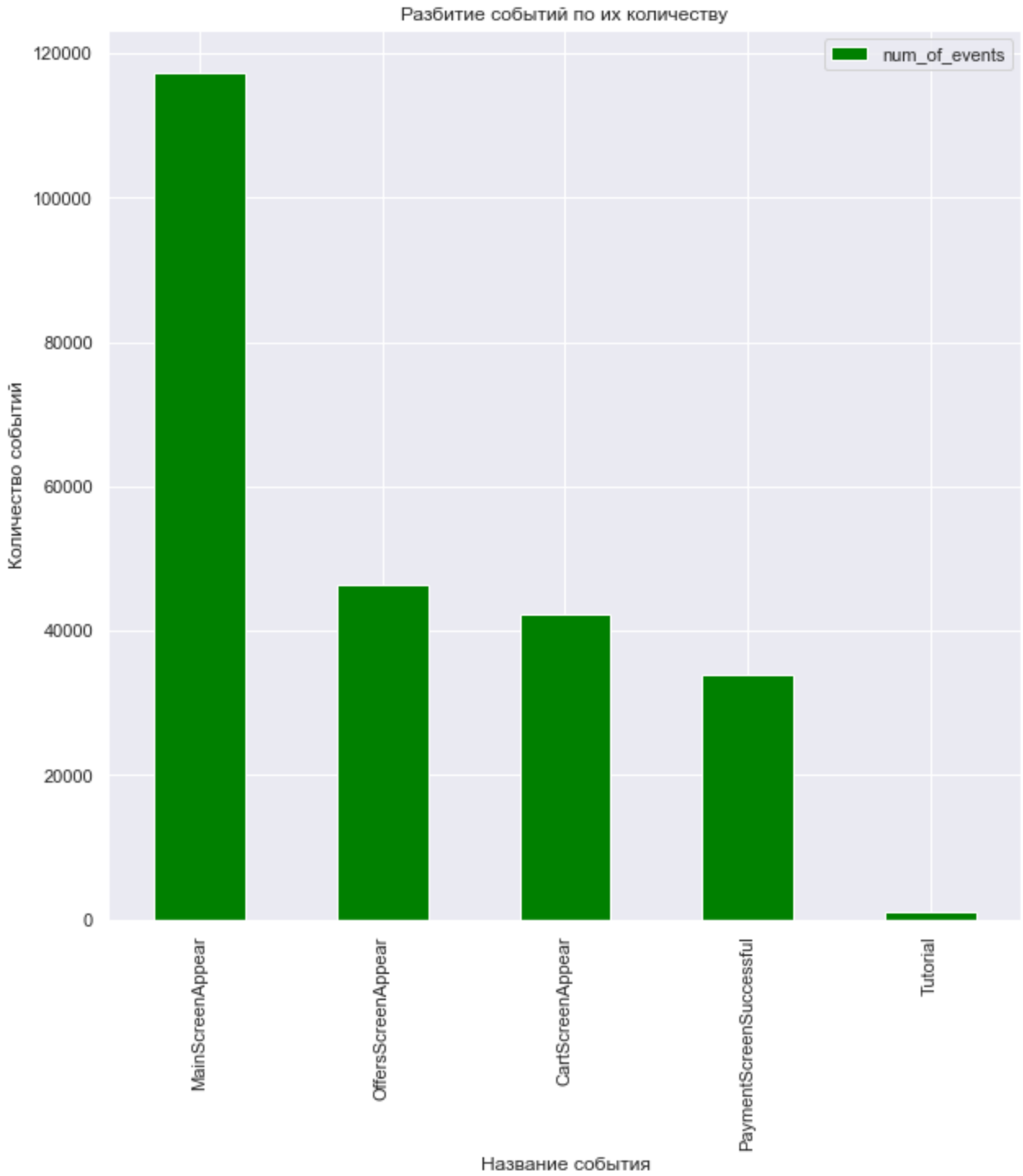
	event_name	num_of_events
1	MainScreenAppear	117328
2	OffersScreenAppear	46333
0	CartScreenAppear	42303
3	PaymentScreenSuccessful	33918
4	Tutorial	1005

В нашем датасете мы имеем следующие события (отсортированы по убыванию):

- Появление основного экрана

- Появление экрана с предложениями
- Появление экрана с корзиной
- Появление экрана с успешной покупкой
- Появление экрана с обучением

In [36]:
```
ax = events.plot.bar(x='event_name',y='num_of_events', color = 'green')
ax.set_title('Разбитие событий по их количеству')
ax.set_ylabel("Количество событий")
ax = ax.set_xlabel("Название события")
```



Посмотрим, сколько пользователей совершали каждое из этих событий, посчитаем их долю

In [37]:
```
users = (
    df_filtered.groupby('event_name', as_index=False)['user_id']
```

```
    .nunique()
    .sort_values('user_id', ascending = False)
)
users.columns = (['event_name', 'unique_users'])
users['share'] = round(users['unique_users']/df_filtered['user_id'].nunique(),2)
users
```

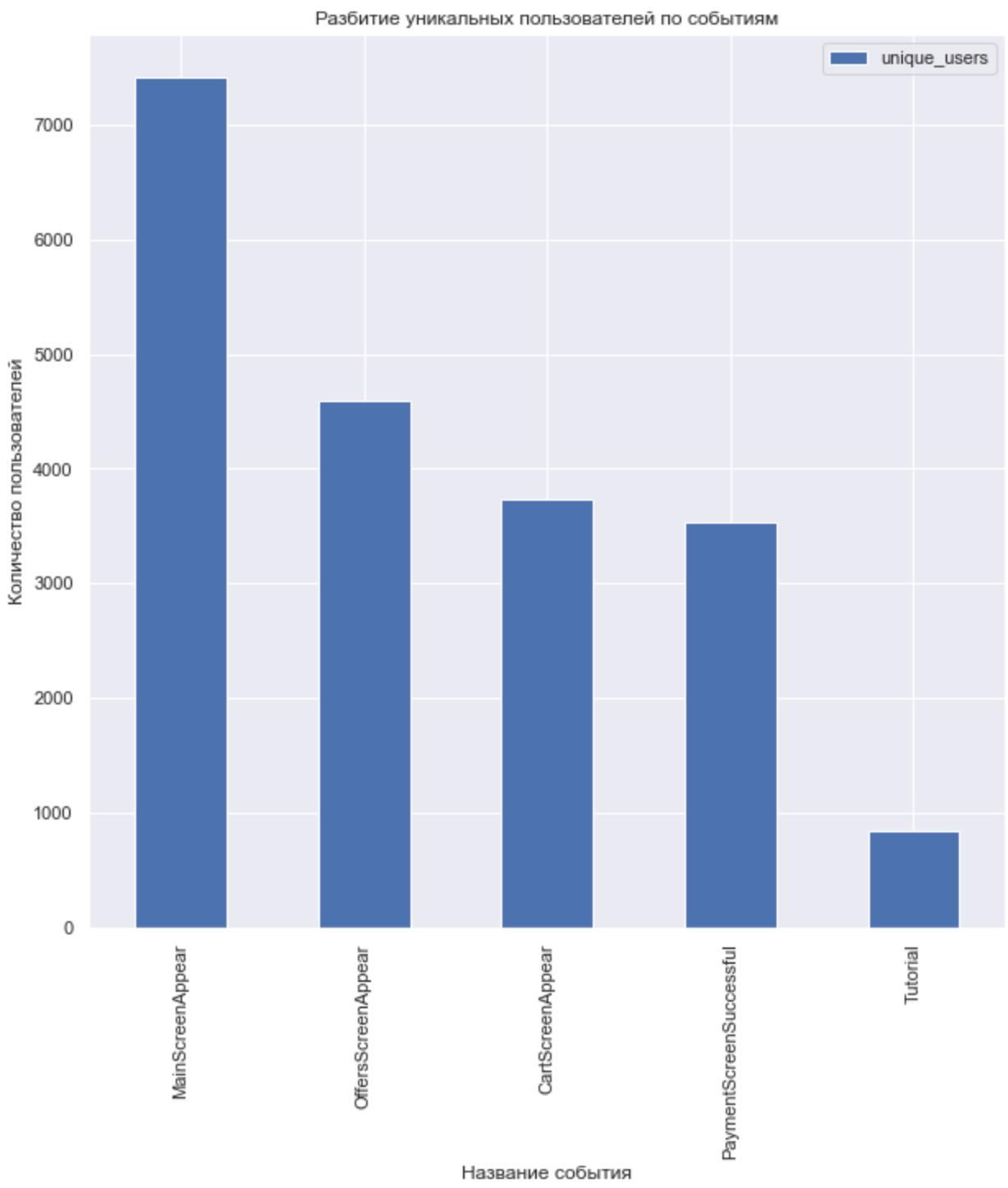
Out[37]:

	event_name	unique_users	share
1	MainScreenAppear	7419	0.98
2	OffersScreenAppear	4593	0.61
0	CartScreenAppear	3734	0.50
3	PaymentScreenSuccessful	3539	0.47
4	Tutorial	840	0.11

Как мы видим, разбитие событий по уникальным пользователям имеет тот же порядок (по убыванию), что и по количеству событий.

In [38]:

```
ax = users.plot.bar(x='event_name',y='unique_users')
ax.set_title('Разбитие уникальных пользователей по событиям')
ax.set_ylabel("Количество пользователей")
ax = ax.set_xlabel("Название события")
```



Предположим в каком порядке происходят события

Судя по полученным данным, воронка получается следующая -

1. Пользователь заходит в приложение и видит главный экран - событие `MainScreenAppear`
2. Пользователь переходит на экран предложений - событие `OffersScreenAppear` и выбирает товары
3. Пользователь переходит в корзину - событие `CartScreenAppear`
4. Пользователь оплачивает товар и появляется экран успешной оплаты - событие `PaymentScreenSuccessfull`

Отдельное независимое событие - **экран обучения** - событие `tutorial` - пользователь может пройти обучение по пользованию приложения, а может и не проходить.

Не будем учитывать событие `education` в воронке продаж.

Посчитаем, какая доля пользователей проходит на следующий шаг

воронки (от числа пользователей на предыдущем).

In [39]:
```
users = (
    pd.pivot_table(df_filtered,values='user_id',index ='event_name',aggfunc='nunique')

).sort_values('user_id',ascending = False).reset_index()
users.columns = (['event_name','user_count'])
users = users.drop(labels = [4],axis = 0)

users
```

Out[39]:

	event_name	user_count
0	MainScreenAppear	7419
1	OffersScreenAppear	4593
2	CartScreenAppear	3734
3	PaymentScreenSuccessful	3539

In [40]:
```
funnel_users = users
funnel_users['prev'] = funnel_users['user_count'].shift(1)
funnel_users['funnel'] = round(funnel_users['user_count']/funnel_users['prev'],2)
funnel_users = funnel_users.fillna('0')
funnel_users
```

Out[40]:

	event_name	user_count	prev	funnel
0	MainScreenAppear	7419	0	0
1	OffersScreenAppear	4593	7419.0	0.62
2	CartScreenAppear	3734	4593.0	0.81
3	PaymentScreenSuccessful	3539	3734.0	0.95

In [41]:
```
funnel_users.iloc[:,[0,3]]
```

Out[41]:

	event_name	funnel
0	MainScreenAppear	0
1	OffersScreenAppear	0.62
2	CartScreenAppear	0.81
3	PaymentScreenSuccessful	0.95

Посчитаем долю пользователей, дошедших с первого шага (главное меню) до успешной покупки (событие - `PaymentScreenSuccessful`)

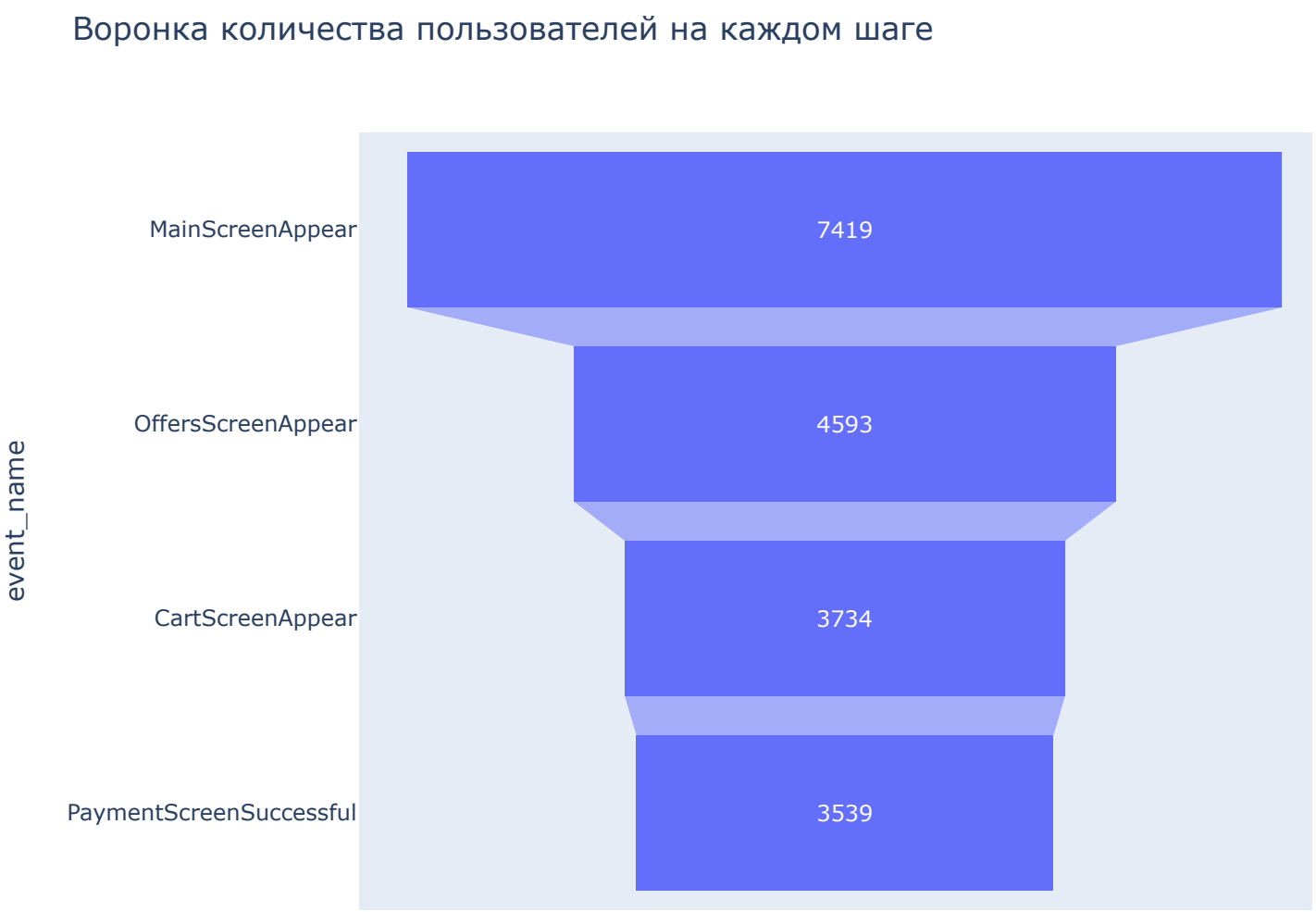
In [42]:
```
print(f'main_to_payment = {round(funnel_users.iloc[3,1]*100/funnel_users.iloc[0,1],2)}%')
```

```
main_to_payment = 47.7%
```

Визуализируем воронку действий пользователя -

In [43]:
```
fig = px.funnel(funnel_users, y='event_name', x='user_count',title="Воронка количества пол
                width=800, height=600)
```

```
fig.show()
```



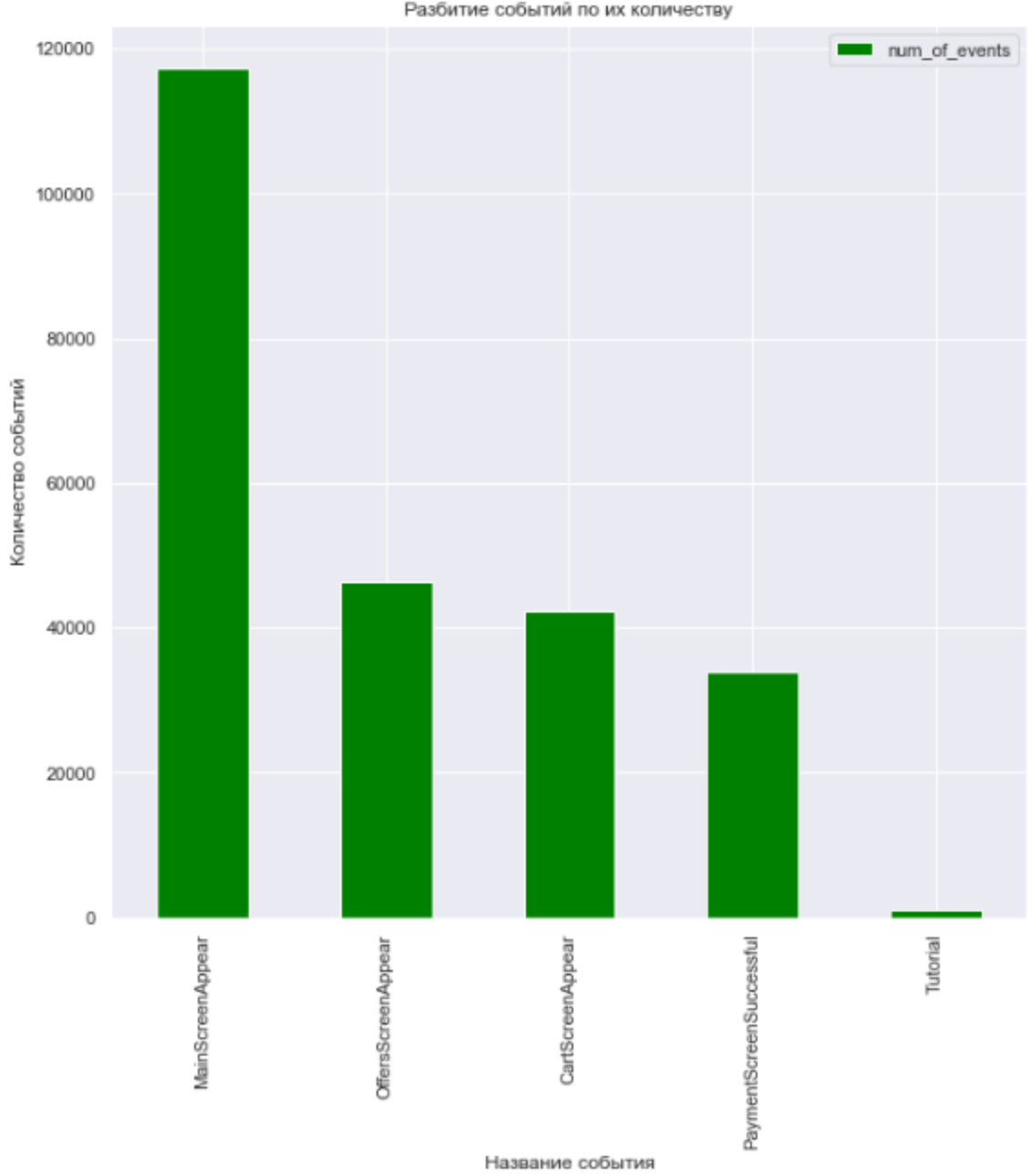
Больше всего пользователей мы "теряем" на первом шаге - переход с главной страницы, на страницу с предложениями ~ 38 % всех пользователей</u>

Количество пользователей, дошедших с первого шага (экран главного меню) до последнего (экран успешной оплаты) - 47.7 %

Вывод по 3-й части исследования

В нашем датасете мы имеем следующие события (отсортированы по убыванию):

- Появление основного экрана
- Появление экрана с предложениями
- Появление экрана с корзиной
- Появление экрана с успешной покупкой
- Появление экрана с обучением



Затем мы посчитали долю уникальных пользователей, которые совершают события -

	event_name	unique_users	share
1	MainScreenAppear	7419	0.98
2	OffersScreenAppear	4593	0.61
0	CartScreenAppear	3734	0.50
3	PaymentScreenSuccessful	3539	0.47
4	Tutorial	840	0.11

Затем посчитали - какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем)

	event_name	funnel
0	MainScreenAppear	0
1	OffersScreenAppear	0.62
2	CartScreenAppear	0.81
3	PaymentScreenSuccessful	0.95

Как мы видим, **больше всего пользователей мы "теряем" на первом шаге - переход с главной страницы, на страницу с предложениями** ~ 38 % всех пользователей

Количество пользователей, дошедших с первого шага (экран главного меню) до последнего (экран успешной оплаты) - 47.7 %

Часть 4. Изучим результаты эксперимента

Посчитаем количество пользователей в каждой экспериментальной группе

In [44]:
```
funnel_filtered = (
    pd.pivot_table(df_filtered,values='user_id',index='group',aggfunc='nunique')
    .sort_values('group')
)
funnel_filtered.columns = (['Number of Unique Users'])
funnel_filtered.reset_index()
```

Out[44]:

	group	Number of Unique Users
0	246	2484
1	247	2513
2	248	2537

Построим сводную таблицу по группам - 246,247, 248

In [45]:
```
aab_table = pd.pivot_table(
```

```
    df_filtered,
    values = 'user_id',
    index = 'group',
    columns = 'event_name',
    aggfunc = 'nunique'
).reset_index()

aab_table = aab_table[['group','MainScreenAppear','OffersScreenAppear','CartScreenAppear',
                       'PaymentScreenSuccessful']]
aab_table
```

Out[45]:

event_name	group	MainScreenAppear	OffersScreenAppear	CartScreenAppear	PaymentScreenSuccessful
0	246	2450	1542	1266	1200
1	247	2476	1520	1238	1158
2	248	2493	1531	1230	1181

Добавим в таблицу общее число уникальных пользователей в группе

In [46]:

```
total = pd.pivot_table(
    df_filtered,
    values = 'user_id',
    index = 'group',
    aggfunc = 'nunique'
).reset_index()
total.columns = (['group', 'total'])

aab_table = aab_table.merge(total,left_on='group',right_on='group')

aab_table
```

Out[46]:

	group	MainScreenAppear	OffersScreenAppear	CartScreenAppear	PaymentScreenSuccessful	total
0	246	2450	1542	1266	1200	2484
1	247	2476	1520	1238	1158	2513
2	248	2493	1531	1230	1181	2537

In [47]:

```
total_246 = total.iloc[0,1]
total_247 = total.iloc[1,1]
total_248 = total.iloc[2,1]
```

Для каждой группы и для каждого шага воронки посчитаем - **какой процент от общего числа пользователей в группе доходит до каждого шага**

In [48]:

```
col = ['MainScreenAppear','OffersScreenAppear','CartScreenAppear','PaymentScreenSuccessful
aab_table_share=pd.DataFrame({'group': [246, 247,248]})
for element in col:
    aab_table_share[element+', share'] = aab_table[element]/aab_table['total']

aab_table_share
```

Out[48]:

	group	MainScreenAppear, share	OffersScreenAppear, share	CartScreenAppear, share	PaymentScreenSuccessful, share
0	246	0.986312	0.620773	0.509662	0.483092
1	247	0.985277	0.604855	0.492638	0.460804

	group	MainScreenAppear, share	OffersScreenAppear, share	CartScreenAppear, share	PaymentScreenSuccessful, share
2	248	0.982657	0.603469	0.484825	0.465510

Обернем расчет z_test'a для пропорций в функцию -

In [49]:
```
def z_calc (p1,p2,total_a,total_b, alpha):
    # зададим критический уровень статистической значимости
    p_combined = (p1*total_a+p2*total_b)/(total_a+total_b)
    #Посчитаем статистику z

    z_value = (p1-p2)/(mth.sqrt(p_combined*(1-p_combined)*(1/total_a+1/total_b)))
    print(f'z-value - {round(z_value,4)}')

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

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

    print(f'p-значение: {round(p_value,4)}')

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

Проверим корректность данных и расчетов по A/A группам - 246, 247

Применим z-тест для пропорций для проверки гипотезы о равенстве долей

Тест будем проверять для каждого события -

- H0 - В группах 246 и 247 доли пользователей на каждом шаге воронки равны
- H1 - В группах 246 и 247 доли пользователей на каждом шаге воронки различны
- alpha = 0.05 - уровень значимости

Расчитаем p_value для каждого события в контрольных группах 246 и 247 -

In [50]:
```
for i in range(1,5):
    print('Событие:')
    print(aab_table_share.columns[i])
    print()
    z_calc(aab_table_share.iloc[0,i],aab_table_share.iloc[1,i],total_246,total_247,0.05)
    print('\n---------------\n')
```

```
Событие:
MainScreenAppear, share

z-value - 0.3093
p-значение: 0.7571
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
OffersScreenAppear, share

z-value - 1.155
p-значение: 0.2481
```

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

---------------

Событие:
CartScreenAppear, share

z-value = 1.2034
p-значение: 0.2288
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
PaymentScreenSuccessful, share

z-value = 1.578
p-значение: 0.1146
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------
```

Как мы видим, в результате z-test'a для пропорций - **мы подтвердили равенство долей для групп 246 и 247**

Проверим, есть ли статистически значимое различие между контрольными группами (246, 247) и тестовой (248)

Применим z-тест для пропорций для проверки гипотезы о различии долей **между контрольной группой - 246 и тестовой 248**

Тест будем проверять для каждого события -

- H0 - В группах 246 и 248 доли пользователей на каждом шаге воронки равны
- H1 - В группах 246 и 248 доли пользователей на каждом шаге воронки различны
- alpha = 0.05 - уровень значимости

In [51]:
```
for i in range(1,5):
    print('Событие:')
    print(aab_table_share.columns[i])
    print()
    z_calc(aab_table_share.iloc[0,i],aab_table_share.iloc[2,i],total_246,total_248,0.05)
    print('\n---------------\n')
```

```
Событие:
MainScreenAppear, share

z-value = 1.0473
p-значение: 0.295
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
OffersScreenAppear, share

z-value = 1.2581
p-значение: 0.2084
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------
```

```
Событие:
CartScreenAppear, share

z-value - 1.7599
p-значение: 0.0784
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
PaymentScreenSuccessful, share

z-value - 1.2474
p-значение: 0.2123
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------
```

По результатам z-test'a о равенстве долей - мы не можем отвергнуть нулевую гипотезу и признать изменения между группами 246 и 248 статистически значимыми

Применим z-тест для пропорций для проверки гипотезы о различии долей между контрольной группой - 247 и тестовой 248

Тест будем проверять для каждого события -

- H0 - В группах 247 и 248 доли пользователей на каждом шаге воронки равны
- H1 - В группах 247 и 248 доли пользователей на каждом шаге воронки различны
- alpha = 0.05 - уровень значимости

In [52]:
```
for i in range(1,5):
    print('Событие:')
    print(aab_table_share.columns[i])
    print()
    z_calc(aab_table_share.iloc[1,i],aab_table_share.iloc[2,i],total_247,total_248,0.05)
    print('\n---------------\n')
```

```
Событие:
MainScreenAppear, share

z-value - 0.741
p-значение: 0.4587
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
OffersScreenAppear, share

z-value - 0.1007
p-значение: 0.9198
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
CartScreenAppear, share

z-value - 0.5554
p-значение: 0.5786
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

```
---------------

Событие:
PaymentScreenSuccessful, share

z-value = -0.3354
p-значение: 0.7373
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------
```

По результатам z-test'a о равенстве долей - мы не можем отвергнуть нулевую гипотезу и признать изменения между группами 247 и 248 статистически значимыми

Создадим датафрейм объединенных контрольных групп (246+247) и тестовой группа (248)

In [53]:
```
# переименуем группу 247 в 246 для объединения
aab_table.loc[1,'group'] = 246
#Объединим по группам
ab_table = aab_table.groupby('group', as_index = False).sum()
# Назовем общую группу `246+247` значением `500`
ab_table.loc[0,'group'] = 500
ab_table
```

Out[53]:

	group	MainScreenAppear	OffersScreenAppear	CartScreenAppear	PaymentScreenSuccessful	total
0	500	4926	3062	2504	2358	4997
1	248	2493	1531	1230	1181	2537

Для каждой группы и для каждого шага воронки посчитаем - **какой процент от общего числа пользователей в группе доходит до каждого шага**

In [54]:
```
col = ['MainScreenAppear','OffersScreenAppear','CartScreenAppear','PaymentScreenSuccessful
ab_table_share=pd.DataFrame({'group': [500, 248]})
for element in col:
    ab_table_share[element+', share'] = ab_table[element]/ab_table['total']

____
ab_table_share
```

Out[54]:

	group	MainScreenAppear, share	OffersScreenAppear, share	CartScreenAppear, share	PaymentScreenSuccessful, share
0	500	0.985791	0.612768	0.501101	0.471883
1	248	0.982657	0.603469	0.484825	0.465510

Применим z-тест для пропорций для проверки гипотезы о различии долей **между объединенной контрольной группой - 500 и тестовой 248**

In [55]:
```
for i in range(1,5):
    print('Событие:')
    print(ab_table_share.columns[i])
    print()
    z_calc(ab_table_share.iloc[0,i],ab_table_share.iloc[1,i],total_247+total_246,total_248
    print('\n---------------\n')
```

```
Событие:
MainScreenAppear, share

z-value - 1.0489
p-значение: 0.2942
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
OffersScreenAppear, share

z-value - 0.7819
p-значение: 0.4343
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
CartScreenAppear, share

z-value - 1.3354
p-значение: 0.1818
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------

Событие:
PaymentScreenSuccessful, share

z-value - 0.5238
p-значение: 0.6004
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

---------------
```

По результатам z-test'а о равенстве долей - мы не можем отвергнуть нулевую гипотезу и признать изменения между объединенной контрольной группой (246+247) и 248 статистически значимыми

Подведем итог

По результатам проведенных `z-test'ов о равенстве доли` (мы провели 12 штук) **мы не можем отвергнуть нулевую гипотезу и признать изменения статистически значимыми**, как между группами (246 и 248) , (247 и 248) , так и между объединенной контрольной группой (246 + 247) и тестовой - 248 .

В тестах мы использовали уровень значимости - 0,05.

В связи с тем, что у нас идет множественная проверка гипотез по 3 м группам, у нас увеличивается риск ложноположительного результата при сравнении долей, в связи с этим - нам необходимо скорректировать уровень **alpha**.

Для этого мы можем использовать поправку бонферрони , для этого надо разделить показатель alpha (0.05) на количество сравниваемых групп -

Рекомендуемый уровень α = 0.05/16 = 0.003125

В связи с тем, что в наших проведенных z-test'ах на равенство долей - нет ложноположительных результатов с `alpha=0.05` , то уменьшив `alpha` до уровня 0.003125 - мы так же не получим результатов, на основании которых мы сможем отклонить нулевую гипотезу о равенстве долей.

В дальнейших исследованиях и тестах - рекомендуется использовать значение alpha с поправкой на множественное сравнение.

Итоговый результат иследования - Изменение шрифта в тестовой группе статистически не повлияло на доли воронки конверсии.

Заключение

Мы провели исследование, как ведут себя пользователи мобильного приложения, который продает продукты питания.

Для начала мы изучили воронку продаж и получили следующие наблюдения -

- Доля пользователей, которые совершают события

	event_name	unique_users	share
1	MainScreenAppear	7419	0.98
2	OffersScreenAppear	4593	0.61
0	CartScreenAppear	3734	0.50
3	PaymentScreenSuccessful	3539	0.47
4	Tutorial	840	0.11

- Доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем)

	event_name	funnel
0	MainScreenAppear	0
1	OffersScreenAppear	0.62
2	CartScreenAppear	0.81
3	PaymentScreenSuccessful	0.95

Больше всего пользователей мы "теряем" на первом шаге - переход с главной страницы, на страницу с предложениями ~ 38 % всех пользователей

Количество пользователей, дошедших с первого шага (экран главного меню) до последнего (экран успешной оплаты) - 47.7 %

Далее мы изучили результаты A/A/B - теста -

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.

В результате исследования мы должны были понять, какой шрифт лучше.

Для начала мы посмотрели на количество уникальных пользователей в каждой из групп

	group	Number of Unique Users
0	246	2484
1	247	2513
2	248	2537

Для каждой группы и каждого шага воронки - посчитали - какой процент от общего числа пользователей в группе доходит до каждого шага -

	group	MainScreenAppear, share	OffersScreenAppear, share	CartScreenAppear, share	PaymentScreenSuccessful, share
0	246	0.986312	0.620773	0.509662	0.483092
1	247	0.985277	0.604855	0.492638	0.460804
2	248	0.982657	0.603469	0.484825	0.465510

Провели A/A - тест и удостоверились, что между контрольными группами 246 и 247 нет статистически значимых изменений в долях пользователей.

Далее мы провели A/B - тесты - используя `z-тест для пропорций`, для сравнения следующих групп -

- Контрольная 246 и тестовая 248
- Контрольная 247 и тестовая 248
- Объединенная контрольная (246+247) и тестовая 248

По результатам, `z-тестов` (с уровнем значимости **alpha = 0.05**) - мы не смогли отклонить нулевую гипотезу о равенстве долей и признать изменения в доле пользователей статистически значимыми.

Изменение шрифта в тестовой группе статистически не повлияло на доли воронки конверсии.