# Сборный проект 2

## Описание проекта

Нужно разобраться, как ведут себя пользователи мобильного приложения в стартапе по продаже продуктов питания. Изучить воронку продаж.Исследовать результаты A/A/B-эксперимента.

## Загрузка данных и подготовка их к анализу.

### Загрузка данных

Загрузим данные. Убедимся, что тип данных в каждой колонке — правильный, а также отсутствуют пропущенные значения и дубликаты. При необходимости обработаем их.Путь к файлу:/datasets/logs_exp.csv

In [1]:
```
#загружаем необходимые библиотеки
import pandas as pd
import matplotlib.pyplot as plt
from plotly import graph_objects as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"
import numpy as np
import seaborn as sns
sns.set(style="whitegrid")
colors =["#ef476f","#ffd166","#06d6a0","#118ab2","#073b4c"]
sns.set_palette(sns.color_palette(colors))
import re
from scipy import stats as st
import math as mth
```

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

In [3]:
```
#получаем информацию
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 |

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
```

- Имеем столбцы:
  - EventName - название события
  - DeviceIDHash - уникальный идентификатор пользователя
  - EventTimestamp - время события (в формате unix timestamp)
  - ExpId - номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

## Предобработка данных

In [5]:
```
df.columns# проверка результатов - перечень названий столбцов
```

Out[5]:
```
Index(['EventName', 'DeviceIDHash', 'EventTimestamp', 'ExpId'], dtype='object')
```

Названия столбцов и названия событий указаны некорректно.Переведем их в нижний регистр и переименуем. Так же добавим столбец даты и времени и отдельный столбец дат в формат datetime.

In [6]:
```
# есть проблемы с регистром названий - исправляем их
df = df.rename(columns={'DeviceIDHash':'device_hash_id'})
df = df.rename(columns={'ExpId':'exp_id'})
df = df.rename(columns={'EventName':'event_name'})
df = df.rename(columns={'EventTimestamp':'event_timestamp'})
df.columns.str.lower()
```

Out[6]:
```
Index(['event_name', 'device_hash_id', 'event_timestamp', 'exp_id'], dtype='object')
```

In [7]:
```
# название событий переведем в нижний регистр
df['event_name'] = df['event_name'].str.lower()
```

In [8]:
```
# время события переведем в формат datetime
df['dt'] = pd.to_datetime(df['event_timestamp'], unit='s')
df['date'] =df['dt'].dt.date
```

In [9]:
```
# проверим
df.head()# проверка результатов - перечень названий столбцов
```

Out[9]:

| | event_name | device_hash_id | event_timestamp | exp_id | dt | 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 |

In [10]:
```
# проверим на дубликаты
df.duplicated(subset=['event_name', 'device_hash_id', 'event_timestamp', 'exp_id']).sum()
```

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

In [11]:
```
# удаляем дубликаты
df.drop_duplicates(subset=['event_name', 'device_hash_id', 'event_timestamp', 'exp_id'], inplac
```

In [12]:
```
# проверим на дубликаты
df.duplicated(subset=['event_name', 'device_hash_id', 'event_timestamp', 'exp_id']).sum()
```

Out[12]:
```
0
```

In [13]:
```
# проверим на пропуски
df.isna().sum()
```

Out[13]:
```
event_name         0
device_hash_id     0
event_timestamp    0
exp_id             0
dt                 0
date               0
dtype: int64
```

пропусков не обнаружено.

In [14]:
```
# Удалим столбец.
df.drop(["event_timestamp"], axis=1, inplace=True)
```

In [15]:
```
df.columns = ['event_name', 'device_hash_id', 'exp_id', 'dt','date']
```

In [16]:
```
df.head()
```

Out[16]:

| | event_name | device_hash_id | exp_id | dt | date |
|---|---|---|---|---|---|
| **0** | mainscreenappear | 4575588528974610257 | 246 | 2019-07-25 04:43:36 | 2019-07-25 |
| **1** | mainscreenappear | 7416695313311560658 | 246 | 2019-07-25 11:11:42 | 2019-07-25 |
| **2** | paymentscreensuccessful | 3518123091307005509 | 248 | 2019-07-25 11:28:47 | 2019-07-25 |
| **3** | cartscreenappear | 3518123091307005509 | 248 | 2019-07-25 11:28:47 | 2019-07-25 |
| **4** | paymentscreensuccessful | 6217807653094995999 | 248 | 2019-07-25 11:48:42 | 2019-07-25 |

### Вывод.

- В процессе предобработки:
  - привели к нижнему регистру текстовые данные в столбцах
  - переименовали столбцы
  - проверили на дубли и отсутствующие значения
  - добавили столбец даты и времени и отдельный столбец дат в формат datetime.
- Тип данных в каждой колонке — правильный Дубликаты устранены. Пропущенных значений нет.

## Изучим и проверим данные.

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243713 entries, 0 to 244125
Data columns (total 5 columns):
 #   Column          Non-Null Count   Dtype         
---  ------          --------------   -----         
 0   event_name      243713 non-null  object        
 1   device_hash_id  243713 non-null  int64         
 2   exp_id          243713 non-null  int64         
 3   dt              243713 non-null  datetime64[ns]
```

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

In [18]:
```
# найдем,сколько всего уникальных событий в логе
df['event_name'].unique()
```

Out[18]:
```
array(['mainscreenappear', 'paymentscreensuccessful', 'cartscreenappear',
       'offersscreenappear', 'tutorial'], dtype=object)
```

- Имеем действия:
  - mainscreenappear - открыл главный экран
  - paymentscreensuccessful - оплатил
  - cartscreenappear - добавил в корзину
  - offersscreenappear - зашел в каталог
  - tutorial - обратился к справочнику

Уникальных событий в логе 5

In [19]:
```
# найдем,сколько всего пользователей в логе
users = df['device_hash_id'].nunique()
users
```

Out[19]:
```
7551
```

In [20]:
```
# найдем,сколько всего событий в логе
event = df['event_name'].count()
event
```

Out[20]:
```
243713
```

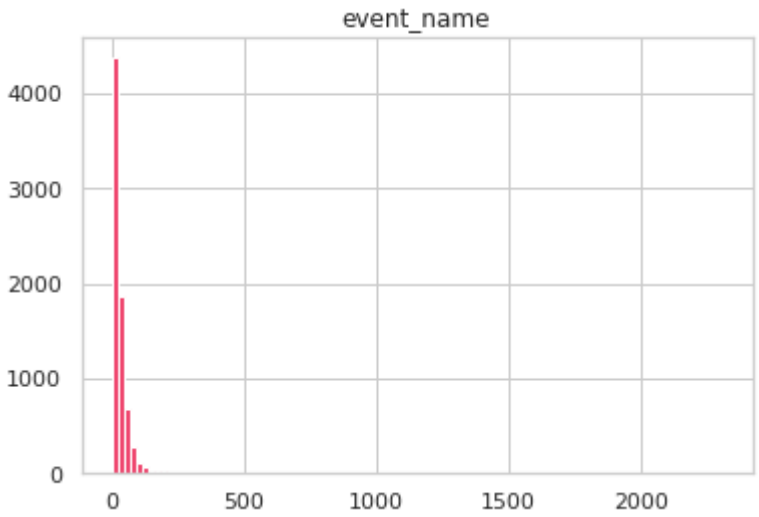
In [21]:
```
# найдем,сколько в среднем событий приходится на пользователя
int(df.groupby('device_hash_id')['event_name'].agg('count').median())
```

Out[21]:
```
20
```

In [22]:
```
# выведем статистические данные через метод describe()
df.groupby('device_hash_id')['event_name'].agg('count').describe()
```

Out[22]:
```
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_name, dtype: float64
```

In [23]:
```
# построим гистограмму
df.groupby('device_hash_id').agg({'event_name':'count'}).hist(bins=100)
df.groupby('device_hash_id').agg({'event_name':'count'}).median()
plt.show()
```




In [24]:
```
 # минимальная дата привлечения пользователей.
df['dt'].min()
```

Out[24]:
```
Timestamp('2019-07-25 04:43:36')
```

In [25]:
```
# максимальная дата привлечения пользователей.
df['dt'].max()
```

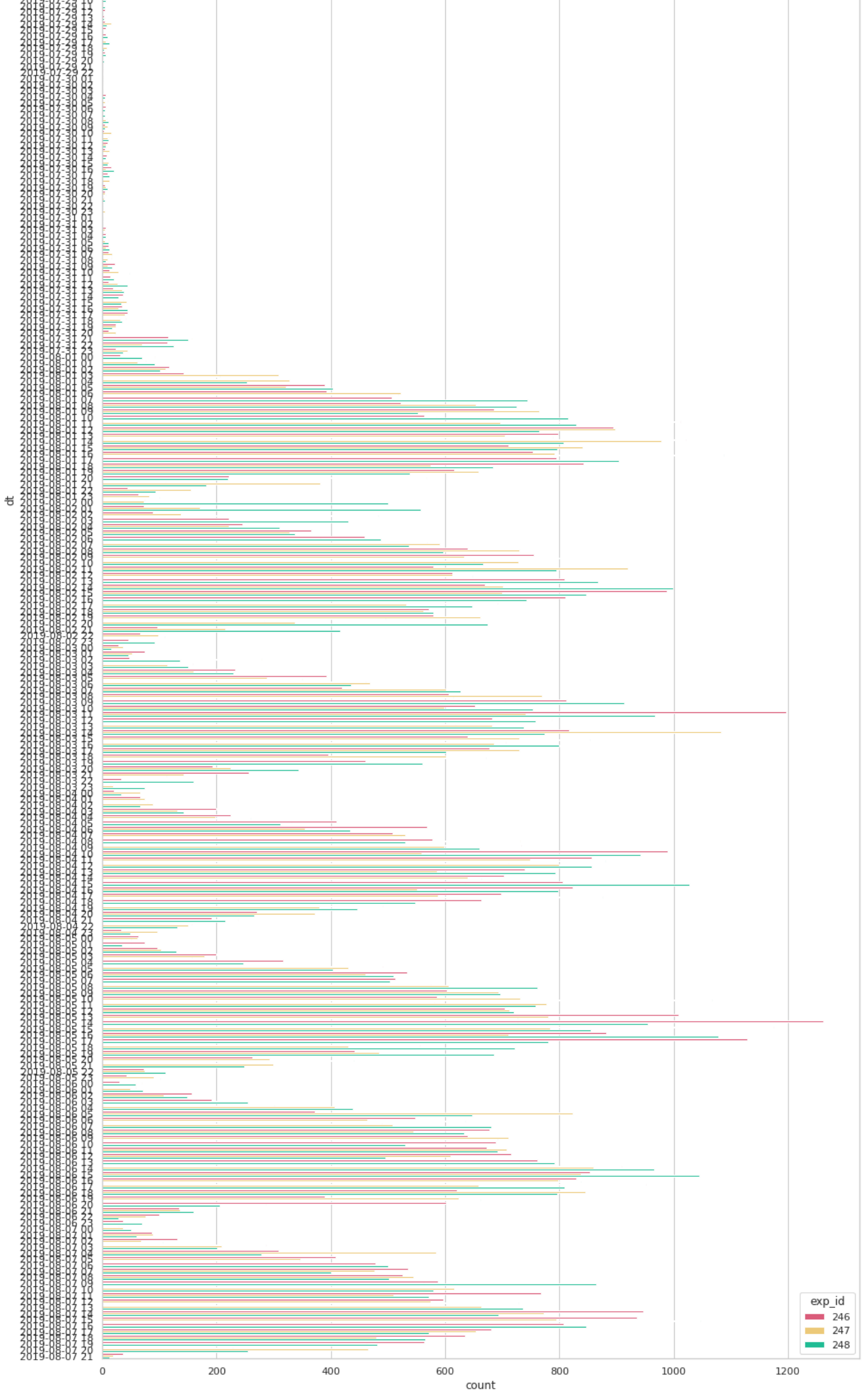
Out[25]:
```
Timestamp('2019-08-07 21:15:17')
```

In [26]:
```
# период
df['dt'].max() - df['dt'].min()
```

Out[26]:
```
Timedelta('13 days 16:31:41')
```

In [27]:
```
print('Всего событий в логе за актуальный период {}.'.format(df['device_hash_id'].count()))
print('')
print('Всего пользователей в логе за актуальный период {}.'.format(df['device_hash_id'].nunique
print('')
plt.figure(figsize=(15, 35))
ax = sns.countplot(y=df['dt'].dt.strftime('%Y-%m-%d %H'),  data=df, hue='exp_id')
ax.set_title('Количество событий для каждой из групп')
plt.show()
```

```
Всего событий в логе за актуальный период 243713.

Всего пользователей в логе за актуальный период 7551.
```


dt
count
0
200
400
600
800
1000
1200
exp_id
246
247
248

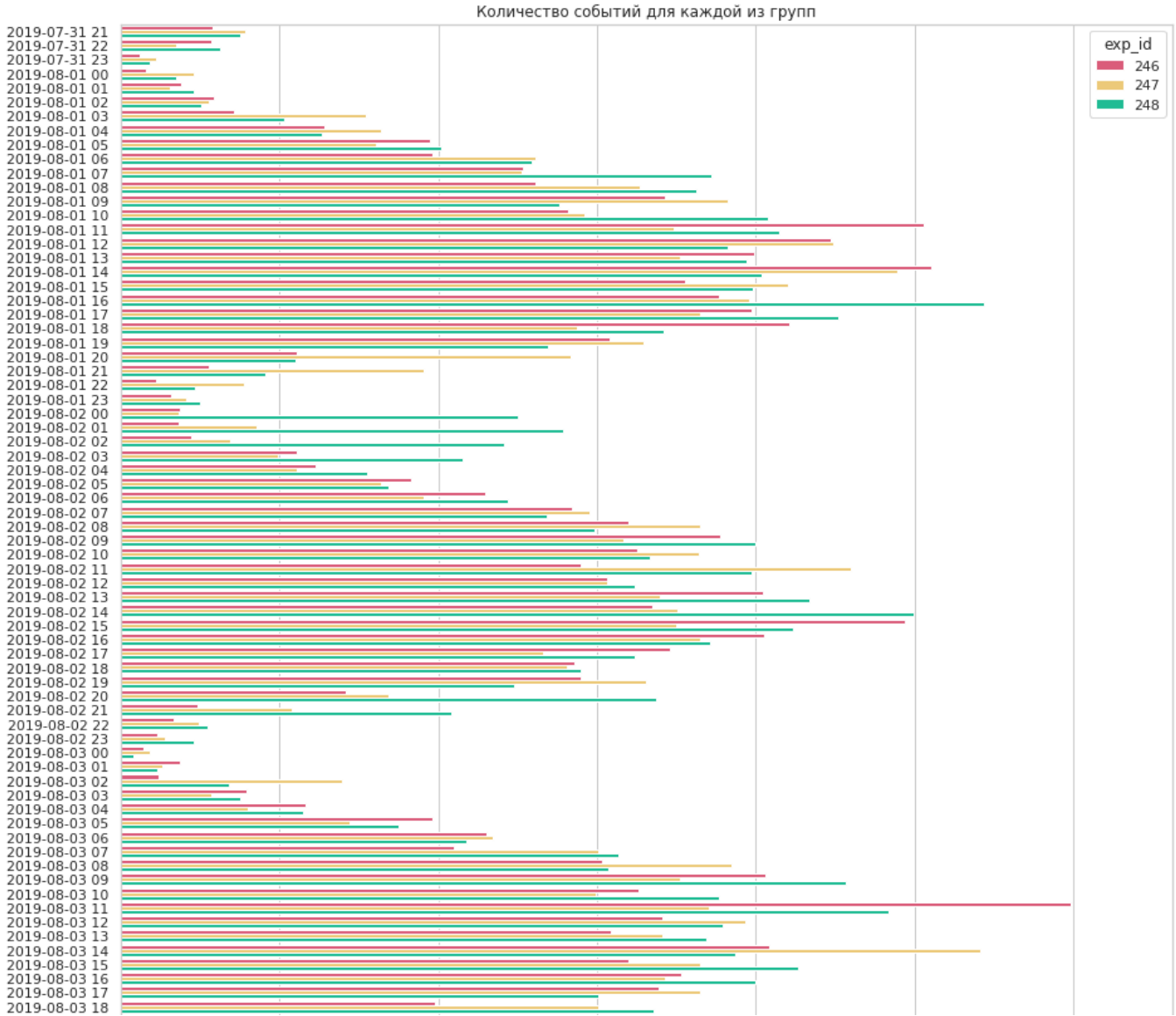
In [28]:
```
# Проверим, есть ли пользователи из всех трёх экспериментальных групп.
df.groupby('device_hash_id')['exp_id'].nunique().count() > 1
```

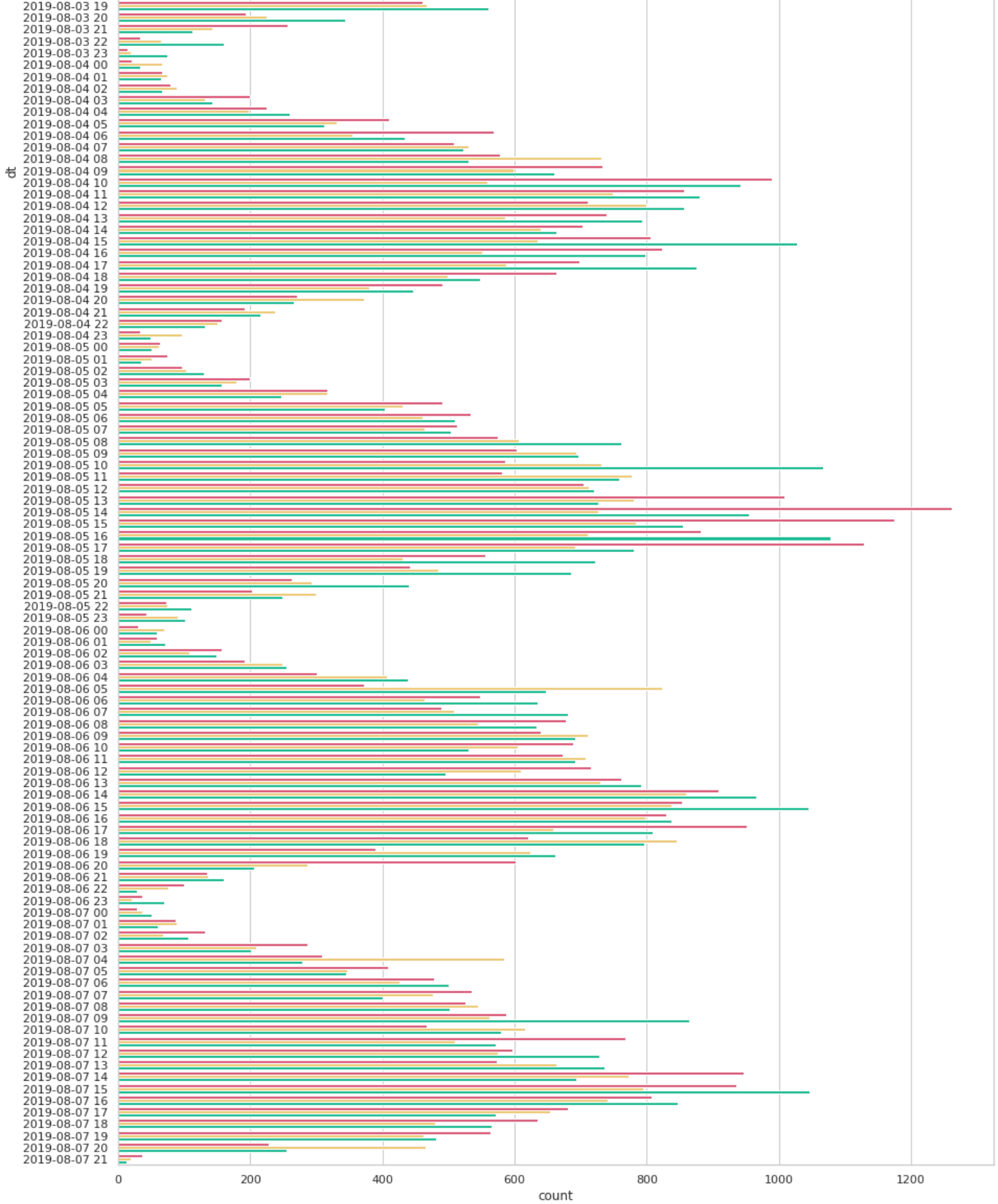
Out[28]: True

2019-08-01 видим резкий скачок. С этого момента данные полные,поэтому отбросим более старые.
Теперь мы располагаем данными с 2019-07-31 21:00:57 по 2019-08-07 21:15:17 .

In [29]:
```
df = df['2019-07-31 21:00' <= df['dt']]
print('Дальнейший анализ будет проводиться для данных в период с {} по {}.'.format(df['dt'].mir
print('')
print('Всего событий в логе за актуальный период {}.'.format(df['device_hash_id'].count()))
print('')
print('Всего пользователей в логе за актуальный период {}.'.format(df['device_hash_id'].nunique
print('')
print('В среднем на пользователя приходится {} событий.'
      .format(int(df.groupby('device_hash_id')['event_name'].agg('count').median())))
plt.figure(figsize=(15, 35))
ax = sns.countplot(y=df['dt'].dt.strftime('%Y-%m-%d %H'),  data=df, hue='exp_id')
ax.set_title('Количество событий для каждой из групп')
plt.show()
```

```
Дальнейший анализ будет проводиться для данных в период с 2019-07-31 21:00:57 по 2019-08-07 21:
15:17.

Всего событий в логе за актуальный период 241724.

Всего пользователей в логе за актуальный период 7538.

В среднем на пользователя приходится 19 событий.
```






In [30]:
```
# число событий после удаления
event_new = df['event_name'].count()
event_new
```

Out[30]: 241724

In [31]:
```
# кол-во потерянных событий
event - event_new
```

Out[31]: 1989

1989 Событий мы потеряли после удаления.

In [32]:
```
# процент потерянных событий
(event - event_new)/ event
```

Out[32]:
```
0.008161238834202524
```

In [33]:
```
# Уникальные пользователи после удаления
users_new = df['device_hash_id'].nunique()
users_new
```

Out[33]:
```
7538
```

In [34]:
```
# кол-во потерянных пользователей
users - users_new
```

Out[34]:
```
13
```

13 Пользователей мы потеряли после удаления.

In [35]:
```
# процент потерянных пользователей
(users - users_new)/ users
```

Out[35]:
```
0.0017216262746656073
```

In [36]:
```
# найдем,сколько в среднем событий приходится на пользователя
int(df.groupby('device_hash_id')['event_name'].agg('count').median())
```

Out[36]:
```
19
```

In [37]:
```
# выведем статистические данные через метод describe()
df.groupby('device_hash_id')['event_name'].agg('count').describe()
```

Out[37]:
```
count    7538.000000
mean       32.067392
std        65.161568
min         1.000000
25%         9.000000
50%        19.000000
75%        37.000000
max      2307.000000
Name: event_name, dtype: float64
```

Отбросив неполноценные данные мы потеряли менее 1% событий и менее 1% пользователей.В среднем на пользователя приходится 19 событий.На гистограмме видно, что у нас присутствуют данные всех групп.

# Изучим воронку событий.

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

In [38]:
```
# Отсортируем события в логах по частоте.
event_name_ = df.groupby('event_name')['device_hash_id'].nunique().sort_values()
event_name_
```

Out[38]:
```
event_name
tutorial                   843
paymentscreensuccessful   3540
cartscreenappear          3736
offersscreenappear        4597
```

```
mainscreenappear           7423
Name: device_hash_id, dtype: int64
```

In [39]:
```
events = df.groupby('event_name')['device_hash_id'].agg(['count', 'nunique']).reset_index()
events
```

Out[39]:

| | event_name | count | nunique |
|---|---|---|---|
| **0** | cartscreenappear | 42343 | 3736 |
| **1** | mainscreenappear | 117889 | 7423 |
| **2** | offersscreenappear | 46531 | 4597 |
| **3** | paymentscreensuccessful | 33951 | 3540 |
| **4** | tutorial | 1010 | 843 |

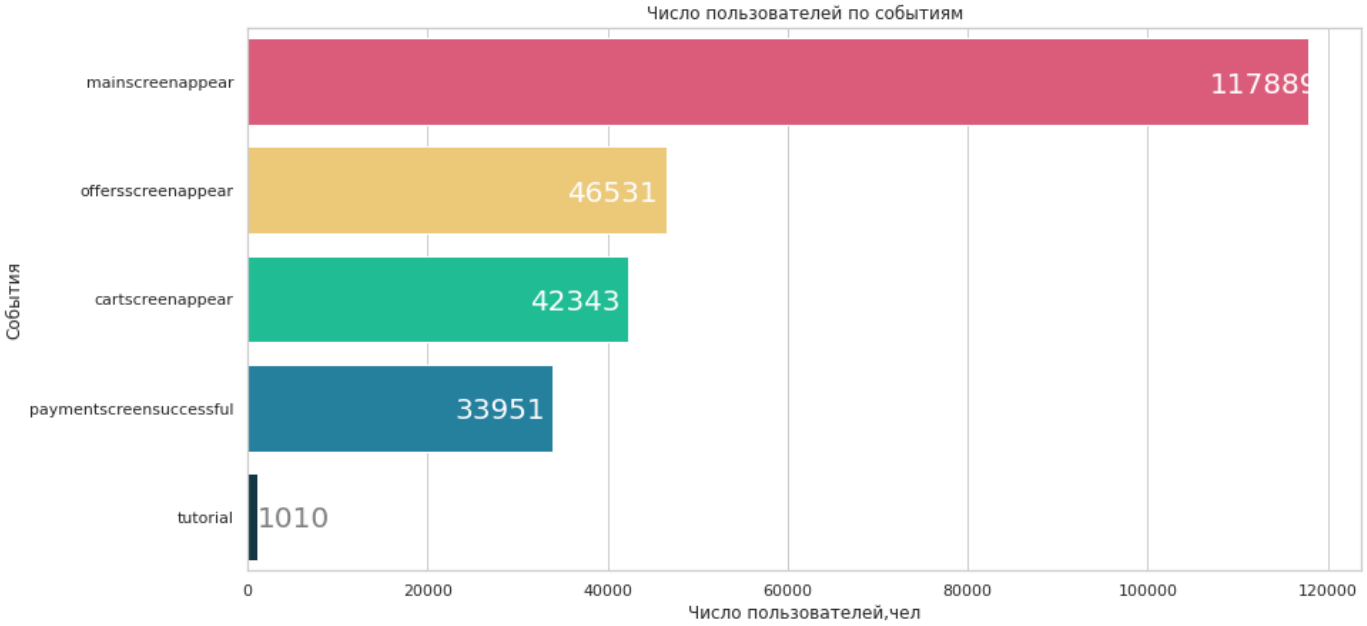
In [40]:
```
n_users = {'all': df['device_hash_id'].nunique(),
           246: df[df['exp_id']==246]['device_hash_id'].nunique(),
           247: df[df['exp_id']==247]['device_hash_id'].nunique(),
           248: df[df['exp_id']==248]['device_hash_id'].nunique(),
           '246+247': df[(df['exp_id']==246) | (df['exp_id']==247)]['device_hash_id'].nunique()
print('Кол-во участников в 246 группе:',df[df['exp_id']==246]['device_hash_id'].nunique())
print('Кол-во участников в 247 группе:',df[df['exp_id']==247]['device_hash_id'].nunique())
print('Кол-во участников в 248 группе:',df[df['exp_id']==248]['device_hash_id'].nunique())
print('Кол-во участников в объединенной группе:',df[(df['exp_id']==246) | (df['exp_id']==247)][
```

```
Кол-во участников в 246 группе: 2484
Кол-во участников в 247 группе: 2517
Кол-во участников в 248 группе: 2537
Кол-во участников в объединенной группе: 5001
```

Группы примерно равны.

**Число пользователей по событиям**

In [41]:
```
events.columns =['event_name', 'n_events', 'n_users']
plt.figure(figsize=(14, 7))
order = events.sort_values('n_events', ascending=False).reset_index(drop=True)['event_name']
ax = sns.barplot(y='event_name', x='n_events', order = order,  data=events)
ax.set_title('Число пользователей по событиям')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События')
for i in ax.patches:
    if i.get_width() > 20000:
        ax.text(i.get_width()-11000, i.get_y()+0.5,
                str(int(i.get_width())), fontsize=20, color='white')
    else:
        ax.text(i.get_width()+30, i.get_y()+0.5,
                str(int(i.get_width())), fontsize=20, color='grey')
plt.show()
```




**Число пользователей по событиям в группах**

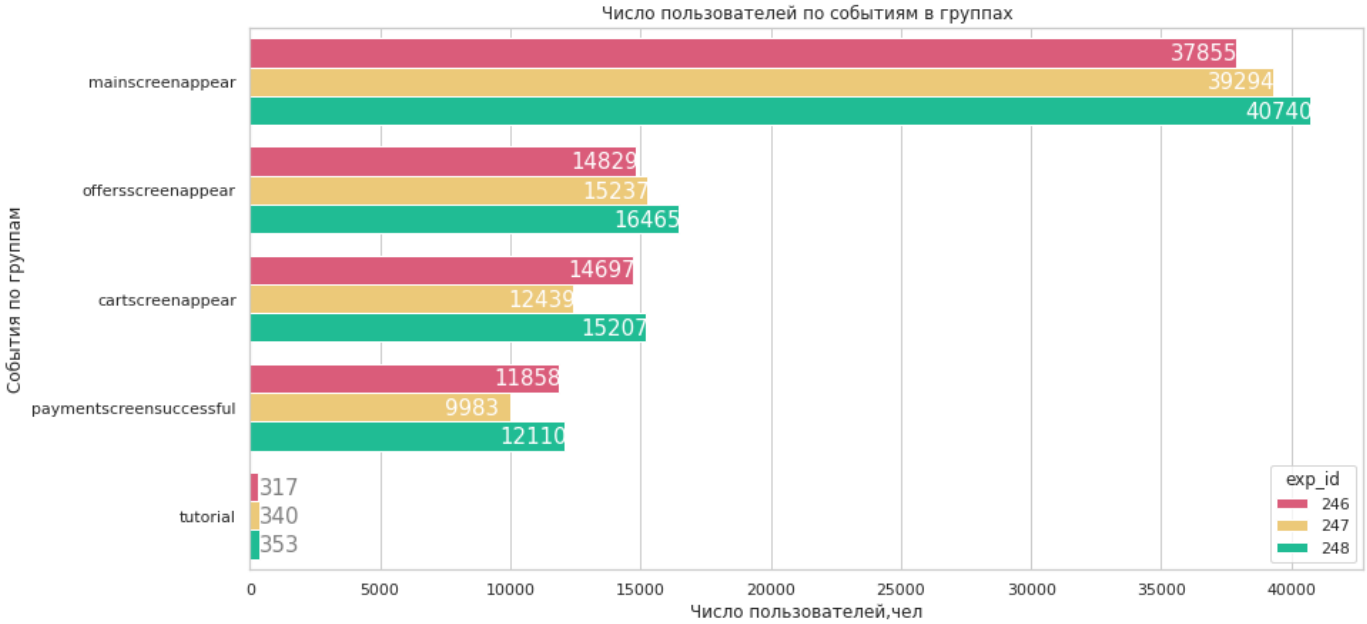
In [42]:
```
event_pivot=df.pivot_table(index=['event_name','exp_id'],values='device_hash_id', aggfunc=['cou
event_pivot.columns= ['event_name','exp_id', 'n_events', 'n_users']
event_pivot
```

Out[42]:

| | event_name | exp_id | n_events | n_users |
|---|---|---|---|---|
| **0** | cartscreenappear | 246 | 14697 | 1266 |
| **1** | cartscreenappear | 247 | 12439 | 1239 |
| **2** | cartscreenappear | 248 | 15207 | 1231 |
| **3** | mainscreenappear | 246 | 37855 | 2450 |
| **4** | mainscreenappear | 247 | 39294 | 2479 |
| **5** | mainscreenappear | 248 | 40740 | 2494 |
| **6** | offersscreenappear | 246 | 14829 | 1542 |
| **7** | offersscreenappear | 247 | 15237 | 1524 |
| **8** | offersscreenappear | 248 | 16465 | 1531 |
| **9** | paymentscreensuccessful | 246 | 11858 | 1200 |
| **10** | paymentscreensuccessful | 247 | 9983 | 1158 |
| **11** | paymentscreensuccessful | 248 | 12110 | 1182 |
| **12** | tutorial | 246 | 317 | 278 |
| **13** | tutorial | 247 | 340 | 284 |
| **14** | tutorial | 248 | 353 | 281 |

In [43]:
```
plt.figure(figsize=(14, 7))
ax = sns.barplot(y='event_name', x='n_events', order = order, hue='exp_id', data=event_pivot)
ax.set_title('Число пользователей по событиям в группах')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События по группам')
for i in ax.patches:
    if i.get_width() > 9000:
        ax.text(i.get_width()-2500, i.get_y()+0.2,
                str(int(i.get_width())), fontsize=15, color='white')
    else:
        ax.text(i.get_width()+30, i.get_y()+0.2,
```

```
                str(int(i.get_width()))), fontsize=15, color='grey')
plt.show()
```




- Самое частое событие это появление основного экрана(mainscreenappear) случалось 117889 раз.
- Второе по популярности событие это появление экрана с каталогом продуктов(offersscreenappear) случалось 46531 раз.
- Третье по популярности событие это появление экрана с корзиной(cartscreenappear) случалось 42343 раза.
- Четвёртое по популярности событие это появление экрана с успешной оплатой(paymentscreensuccessful) случалось 33951 раз.
- Пятое по популярности событие это открытие руководства пользователя(tutorial) случалось 1010 раз.
- Кол-во участников в 246 группе: 2484
- Кол-во участников в 247 группе: 2517
- Кол-во участников в 248 группе: 2537
- Кол-во участников в объединенной группе: 5001

## Посчитаем, сколько пользователей совершали каждое из этих событий и долю пользователей, которые хоть раз совершали событие.

In [44]:
```
events.sort_values(by = 'n_events')
```

Out[44]:

| | event_name | n_events | n_users |
|---|---|---|---|
| **4** | tutorial | 1010 | 843 |
| **3** | paymentscreensuccessful | 33951 | 3540 |
| **0** | cartscreenappear | 42343 | 3736 |
| **2** | offersscreenappear | 46531 | 4597 |
| **1** | mainscreenappear | 117889 | 7423 |

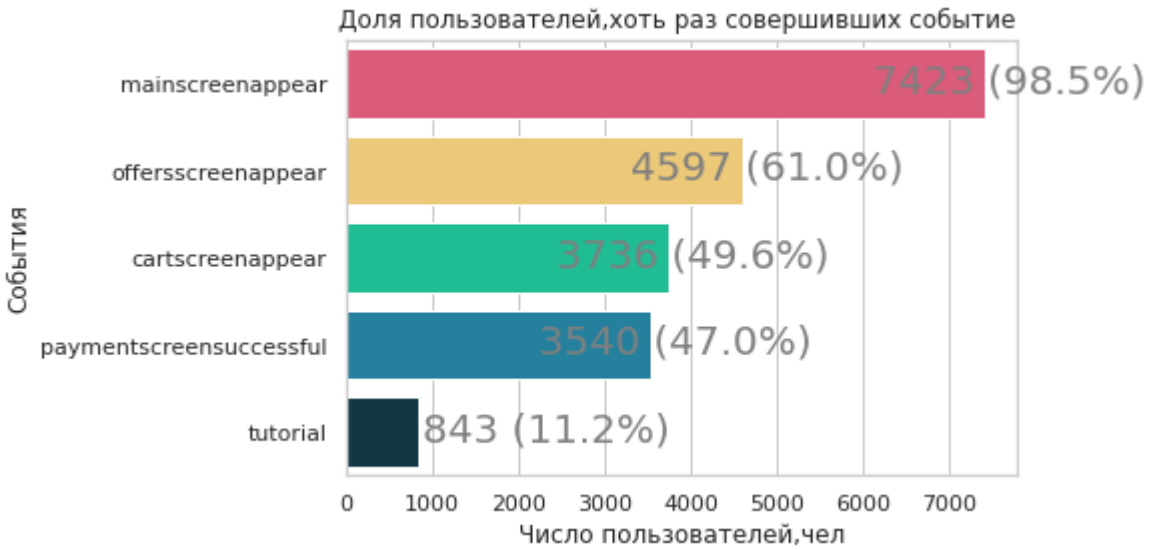
In [45]:
```
events['users_part'] = round(events['n_users']*100/len(df['device_hash_id'].unique()), 2)
events['users_part']
```

Out[45]:
```
0    49.56
1    98.47
```

```
2    60.98
3    46.96
4    11.18
Name: users_part, dtype: float64
```

In [46]:

```
ax = sns.barplot(y='event_name', x='n_users', order = order,  data=events)
ax.set_title('Доля пользователей,хоть раз совершивших событие ')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События')
for i in ax.patches:
    if i.get_width() > 3000:
        ax.text(i.get_width()-1300, i.get_y()+0.5,
            str(int(i.get_width()))+' ({:.1%})'.format(i.get_width() / n_users['all']), fontsiz
    else:
        ax.text(i.get_width()+30, i.get_y()+0.5,
            str(int(i.get_width()))+' ({:.1%})'.format(i.get_width() / n_users['all']), fontsiz
plt.figure(figsize=(14, 7))
plt.show()
```



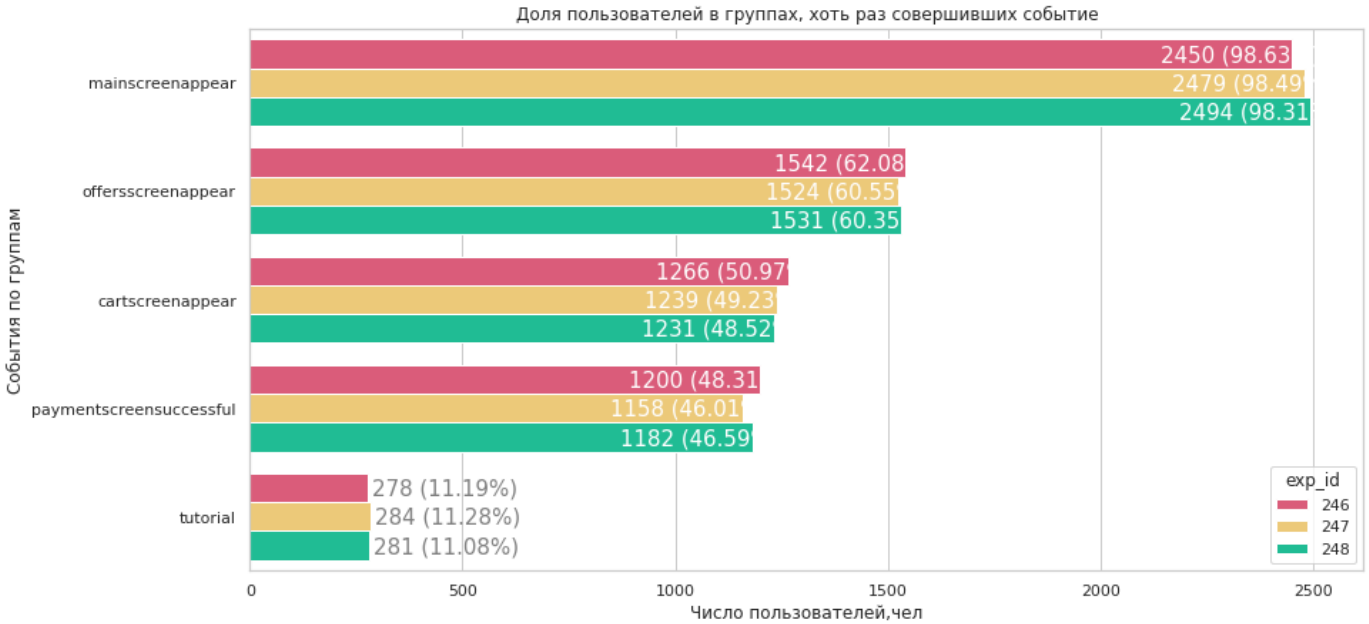

```
<Figure size 1008x504 with 0 Axes>
```

**Доля пользователей в группах, хоть раз совершивших событие**

In [47]:

```
plt.figure(figsize=(14, 7))
ax = sns.barplot(y='event_name', x='n_users', order = order, hue='exp_id', data=event_pivot)
ax.set_title('Доля пользователей в группах, хоть раз совершивших событие')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События по группам')

for i,v in enumerate(ax.patches):
    if i < 5:
        if v.get_width() > 1000:
            ax.text(v.get_width()-310, v.get_y()+0.2,
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[246]),
                    fontsize=15, color='white')
        else:
            ax.text(v.get_width()+10, v.get_y()+0.2,
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[246]),
                    fontsize=15, color='grey')
    if 5 <= i < 10:
        if v.get_width() > 1000:
            ax.text(v.get_width()-310, v.get_y()+0.2,
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[247]),
                    fontsize=15, color='white')
        else:
            ax.text(v.get_width()+10, v.get_y()+0.2,
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[247]),
                    fontsize=15, color='grey')
    if i >= 10:
        if v.get_width() > 1000:
            ax.text(v.get_width()-310, v.get_y()+0.2,
```

```
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[248]),
                    fontsize=15, color='white')
        else:
            ax.text(v.get_width()+10, v.get_y()+0.2,
                str(int(v.get_width()))+' ({:.2%})'.format(v.get_width() / n_users[248]),
                    fontsize=15, color='grey')

plt.show()
```




Даже на первом шаге воронки мы не имеем конверсии 1.Это говорит о том, что не все пользователи даже открывают приложение. Проход по всем этапам воронки не является обязательным.

In [48]:
```
# Первое событие
id_mainscreenappear = event_name_['mainscreenappear']/df['device_hash_id'].nunique()* 100
round(id_mainscreenappear,2)
```

Out[48]: 98.47

In [49]:
```
# Второе событие
id_offersscreenappear = event_name_['offersscreenappear']/df['device_hash_id'].nunique()* 100
round(id_offersscreenappear,2)
```

Out[49]: 60.98

In [50]:
```
# Третье событие
id_cartscreenappear = event_name_['cartscreenappear']/df['device_hash_id'].nunique()* 100
round(id_cartscreenappear,2)
```

Out[50]: 49.56

In [51]:
```
# Четвертое событие
id_paymentscreensuccessful = event_name_['paymentscreensuccessful']/df['device_hash_id'].nuniqu
round(id_paymentscreensuccessful,2)
```

Out[51]: 46.96

In [52]:
```
# Пятое событие
id_tutorial = event_name_['tutorial']/df['device_hash_id'].nunique()* 100
round(id_tutorial,2)
```

11.18

Out[52]:

In [53]:
```
#среднее число событий на пользователя
int(df.groupby('device_hash_id')['event_name'].agg('count').median())
```

Out[53]: 19

In [54]:
```
events.sort_values(by = 'users_part')
```

Out[54]:

| | event_name | n_events | n_users | users_part |
|---|---|---|---|---|
| **4** | tutorial | 1010 | 843 | 11.18 |
| **3** | paymentscreensuccessful | 33951 | 3540 | 46.96 |
| **0** | cartscreenappear | 42343 | 3736 | 49.56 |
| **2** | offersscreenappear | 46531 | 4597 | 60.98 |
| **1** | mainscreenappear | 117889 | 7423 | 98.47 |

- 7423 пользователей хотя бы раз открывали главную страницу приложения(mainscreenappear), это 98.47% всех пользователей.Возможно, оставшаяся часть не смогла открыть главную страницу(ошибки реализации, часть эвентов может теряться.)
- 4597 пользователей хотя бы раз открывали страницу с каталогом товаров(offersscreenappear),это 60.96% всех пользователей.
- Почти 39% пользователей не открыли каталог товаров.Возможно, приложение не на всех устройствах работает корректно. Следует проверить.
- 3736 пользователей хотя бы раз открывали корзину(cartscreenappear),это 49.56% всех пользователей.
- 3540 пользователей хотя бы раз попадали на страницу с успешной оплатой(paymentscreensuccessful), это 46.97% всех пользователей.
- 843 пользователя хотя бы раз открывали руководство пользователя(tutorial), это 11.15% всех пользователей.

## Предположим, в каком порядке происходят события

Предполагаем,что события порядке происходят в таком порядке:

- сначала открывает главную страницу приложения(mainscreenappear)
- некоторые пользователи обратились к справочнику(tutorial)
- в каталоге выбирает товары(offersscreenappear)
- кладёт их в корзину(cartscreenappear)
- оплачивает(paymentscreensuccessful)
- Видим, что не все они выстраиваются в последовательную цепочку. Шаг tutorial совершают не все пользователи -около 11%.Этот шаг является обучением и не относится к последовательности событий.

## По воронке событий посчитаем, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем)

In [55]:
```
# От первого ко второму
event_name_['%'] = event_name_['offersscreenappear']/event_name_['mainscreenappear'] * 100
```

```
print(round(event_name_['%'],2))
```

```
61.93
```

In [56]:

```
# От второго к третьему
event_name_['%'] = event_name_['cartscreenappear']/event_name_['offersscreenappear'] * 100
print(round(event_name_['%'],2))
```

```
81.27
```

In [57]:

```
# От третьего к четвертому
event_name_['%'] = event_name_['paymentscreensuccessful']/event_name_['cartscreenappear'] * 100
print(round(event_name_['%'],2))
```

```
94.75
```

## Определим,на каком шаге теряется больше всего пользователей.

Больше всего пользователей теряется на первом шаге - только около 62% переходят к выбору предложения.

## Определим,какая доля пользователей доходит от первого события до оплаты

In [58]:

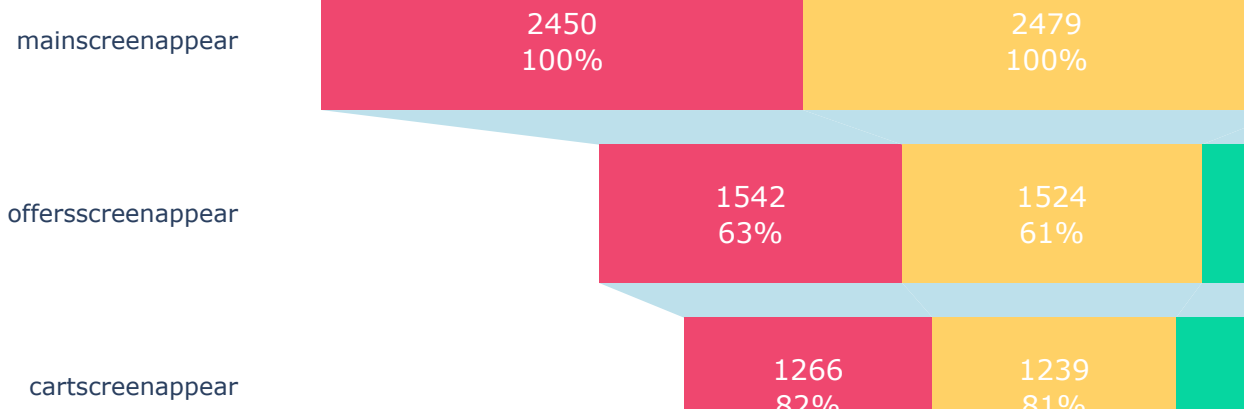
```
# Доля дошедших от первого шага до оплаты
event_name_['%']= event_name_['paymentscreensuccessful']/event_name_['mainscreenappear'] * 100
print(round(event_name_['%'],2))
```

```
47.69
```

До оплаты дохотит около 48% пользователей.

In [59]:

```
order = (events[events['event_name'] != 'tutorial']
         .sort_values('n_events', ascending=False)['event_name']
         .reset_index(drop=True))
groups = [246, 247, 248]
simple_funnel = {}
for group in groups:
    simple_funnel[group] =[]
    for event in order:
        simple_funnel[group].append(df[(df['exp_id'] == group) & (df['event_name'] == event)]['
fig = go.Figure()
for i, group in enumerate(groups):
    fig.add_trace(go.Funnel(
        name = str(group),
        y = (event_pivot[(event_pivot['exp_id'] == group) & (event_pivot['event_name'] != 'tutc
             .sort_values('n_users', ascending=False)['event_name']),
        x = (event_pivot[(event_pivot['exp_id'] == group) & (event_pivot['event_name'] != 'tutc
             .sort_values('n_users', ascending=False)['n_users']),
        textposition = "inside",
        textinfo = "value+percent previous",
        marker = {"color": colors[i]},
        connector = {"fillcolor": '#bde0eb'},
        insidetextfont = {'color': 'white', 'size': 14}))

fig.show()
```




Глядя на эту воронку мы видим, что больше всего пользователей (37% для группы 246 и 39% для групп 247 и 248) уходят без перехода к каталогу товаров(OffersScreenAppear). От первого события до оплаты доходит меньше половины пользователей(49%/46,7%/47,4% для групп 246/247/248 соответственно.)

# Изучим результаты эксперимента.

## Проверим,сколько пользователей в каждой экспериментальной группе.

In [60]:
```
print('Для A/A/B-теста пользователей разбили на 3 группы: '
      '2 контрольные(246 и 247) со старыми шрифтами и одну экспериментальную(248) — с новыми.')
print('')
for group in groups:
    n_users
    print('В {} группе {} пользователей'.format(group, n_users[group]))
    print('')
```

```
Для A/A/B-теста пользователей разбили на 3 группы: 2 контрольные(246 и 247) со старыми шрифтами
и одну экспериментальную(248) — с новыми.

В 246 группе 2484 пользователей

В 247 группе 2517 пользователей

В 248 группе 2537 пользователей
```

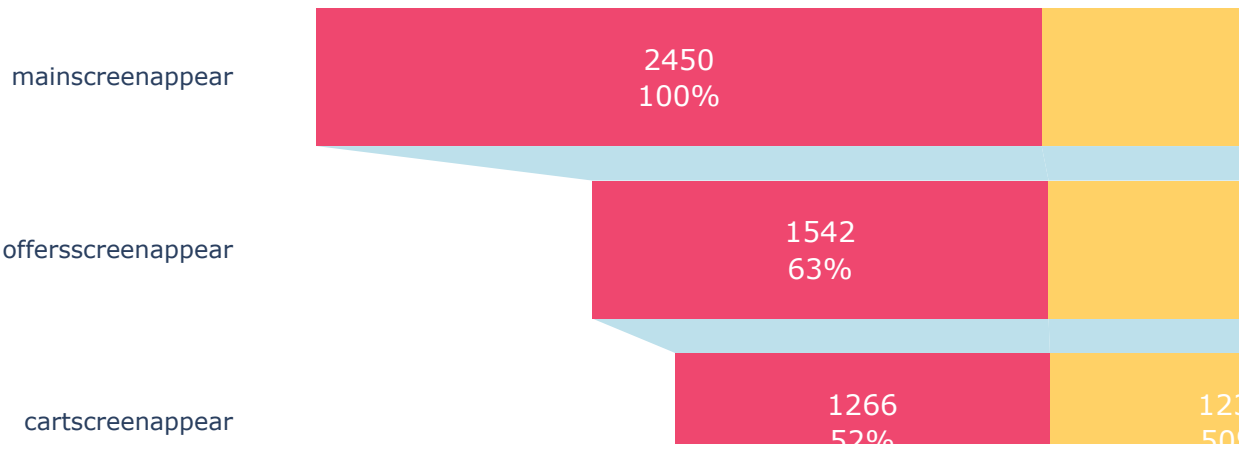
## Проверим, находят ли статистические критерии разницу между выборками 246 и 247

Ранее мы выяснили, что проход по всем этапам воронки не является обязательным, так что работать нужно именно относительно общего числа клиентов.

In [65]:
```
simple_funnel = pd.DataFrame(simple_funnel)
simple_funnel['246+247'] = simple_funnel[246] + simple_funnel[247]
simple_funnel['event_name'] = order
def z_value_diff(first_group, second_group, alpha, color):
    for i in simple_funnel.index:
        alpha = alpha
```

```
        # пропорция успехов в первой группе:
        p1 = simple_funnel[first_group][i] / n_users[first_group]
        # пропорция успехов во второй группе:
        p2 = simple_funnel[second_group][i] / n_users[second_group]
        # пропорция успехов в комбинированном датасете:
        p_combined = ((simple_funnel[first_group][i] + simple_funnel[second_group][i]) /
                      (n_users[first_group] + n_users[second_group]))
        # 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/n_users[first_group] + 1/n_users[second_group]))
        # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
        distr = st.norm(0, 1)
        p_value = (1 - distr.cdf(abs(z_value))) * 2
        print('{} p-значение: {}'.format(simple_funnel['event_name'][i], p_value))
        if (p_value < alpha):
            print("Отвергаем нулевую гипотезу: между долями есть значимая разница")
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разным
        print('')

    fig = go.Figure()
    for i, group in enumerate([first_group, second_group]):
        fig.add_trace(go.Funnel(
            name = str(group),
            y = order,
            x = simple_funnel[group],
            textposition = "inside",
            textinfo = "value+percent initial",
            marker = {"color": colors[i+color]},
            connector = {"fillcolor": '#bde0eb'},
            insidetextfont = {'color': 'white', 'size': 14}))

    fig.show()
z_value_diff(246,247,0.05,0)
```

```
mainscreenappear p-значение: 0.6756217702005545
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

offersscreenappear p-значение: 0.26698769175859516
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

cartscreenappear p-значение: 0.2182812140633792
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

paymentscreensuccessful p-значение: 0.10298394982948822
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



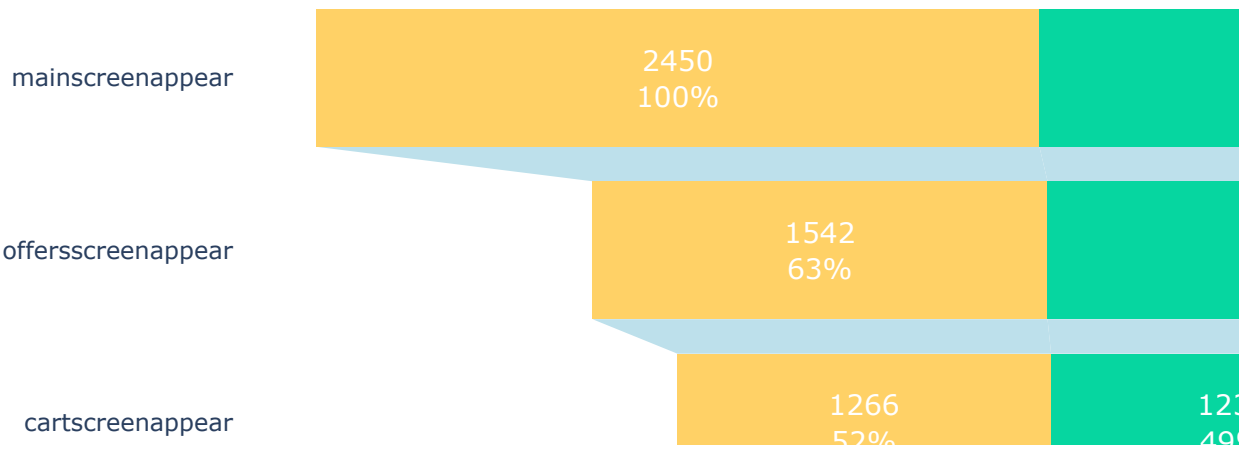
Ни для одного из событий разница не оказалось значимой обе эти группы можно считать контрольными. Именно такой результат мы и ожидали.Тест проведен корректно

## Проверим то же самое с группой с изменённым шрифтом

Группы 246 и 248:

In [62]:
```
z_value_diff(246,248,0.05,1)
```

```
mainscreenappear p-значение: 0.34705881021236484
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: 0.08328412977507749
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

paymentscreensuccessful p-значение: 0.22269358994682742
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

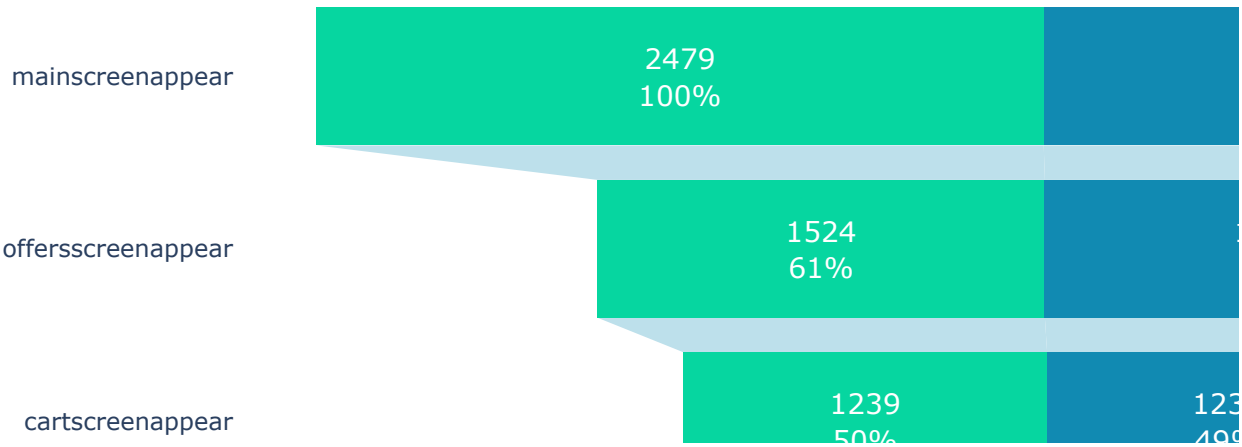



Значимой разницы между контрольной группой 246 и экспериментальной группой не выявлено

Группы 247 и 248:

In [63]:
```
z_value_diff(247,248,0.05,2)
```

```
mainscreenappear p-значение: 0.6001661582453706
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

offersscreenappear p-значение: 0.8835956656016957
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

cartscreenappear p-значение: 0.6169517476996997
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

paymentscreensuccessful p-значение: 0.6775413642906454
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

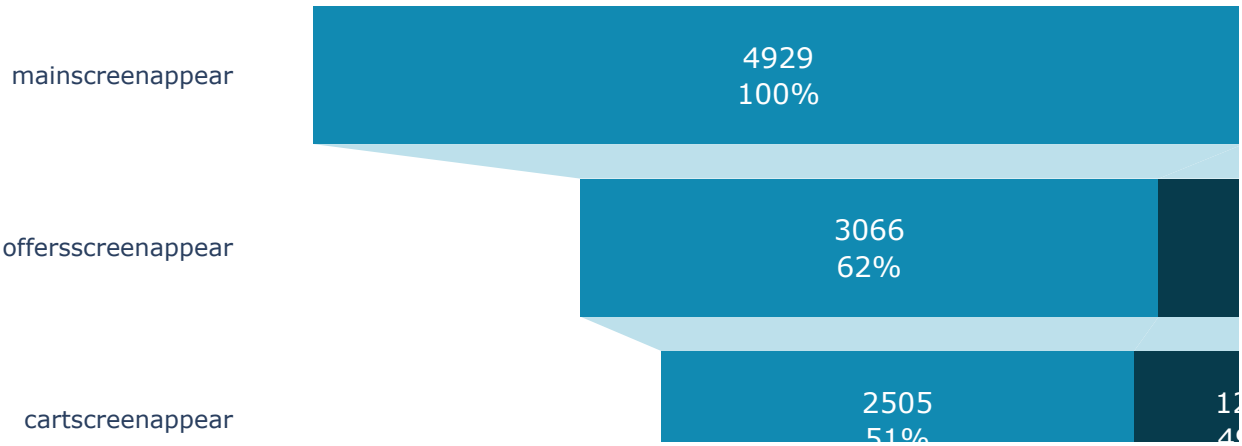



Значимой разницы между контрольной группой 247 и экспериментальной группой не выявлено.

Теперь сравним контрольные 246 и 247, объединенные в одну группу, с экспериментальной 248-ой:

In [64]:
```
z_value_diff('246+247',248,0.05,3)
```

```
mainscreenappear p-значение: 0.39298914928006035
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

offersscreenappear p-значение: 0.418998284007599
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

cartscreenappear p-значение: 0.19819340844527744
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

paymentscreensuccessful p-значение: 0.6452057673098244
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```




Сравнение результатов с объединенной контрольной группой также не показало значимой разницы. Можно использовать уровень значимости 0.1 При этом уровне мы получим те же результаты.

## Вывод.

В процессе предобработки:

- привели к нижнему регистру текстовые данные в столбцах
- переименовали столбцы
- проверили на дубли и отсутствующие значения
- добавили столбец даты и времени, а также отдельный столбец дат
- Тип данных в каждой колонке — правильный. Дубликаты устранены. Пропущенных значений нет.

Изучили и проверили данные.

- всего событий в логе за актуальный период 243713.
- всего пользователей в логе за актуальный период 7551
- нашли максимальную и минимальную дату : 2019-07-25 и 2019-08-07.
- в среднем на пользователя приходится 19 событий.
- построили гистограмму по дате и времени.

Анализируя гистограмму по дате и времени, приняли решение отбросить неполные данные и оставить только период с 2019-07-31 21 часа. Отбросив неполноценные данные потеряли менее 1% событий и менее 1% пользователей. На гистограмме видно, что у нас присутствуют данные всех групп.

Изучили воронку событий

- Самое частое событие это появление основного экрана(mainscreenappear) случалось 117328 раз.

- Второе по популярности событие это появление экрана с каталогом продуктов(offersscreenappear) случалось 46333 раз.
- Третье по популярности событие это появление экрана с корзиной(cartscreenappear) случалось 42303 раза.
- Четвёртое по популярности событие это появление экрана с успешной оплатой(paymentscreensuccessful) случалось 33918 раз.
- Пятое по популярности событие это открытие руководства пользователя(tutorial) случалось 1005 раз.
- 7423 пользователей хотя бы раз открывали главную страницу приложения(mainscreenappear), это 98.47% всех пользователей.Возможно, оставшаяся часть не смогла открыть главную страницу(ошибки реализации, часть эвентов может теряться.)
- 4597 пользователей хотя бы раз открывали страницу с каталогом товаров(offersscreenappear),это 60.96% всех пользователей.
- Почти 39% пользователей не открыли каталог товаров.Возможно, приложение не на всех устройствах работает корректно. Следует проверить.
- 3736 пользователей хотя бы раз открывали корзину(cartscreenappear),это 49.56% всех пользователей.
- 3540 пользователей хотя бы раз попадали на страницу с успешной оплатой(paymentscreensuccessful), это 46.97% всех пользователей.
- 843 пользователя хотя бы раз открывали руководство пользователя(tutorial), это 11.15% всех пользователей.
- События происходят в следующем порядке:
  - сначала открывают главную страницу приложения(mainscreenappear)
  - некоторые пользователи обратились к справочнику(tutorial)
  - в каталоге выбирают товары(offersscreenappear)
  - кладут их в корзину(cartscreenappear)
  - оплачивают(paymentscreensuccessful)
- Видим, что не все они выстраиваются в последовательную цепочку. Шаг tutorial совершают не все пользователи -около 11%.
- Больше всего пользователей теряется на первом шаге - только около 62% переходят к выбору предложения.
- До оплаты дохотит около 48% пользователей.
- Если смотреть по группам:

  - больше всего пользователей (37% для группы 246 и 39% для групп 247 и 248) уходят без перехода к каталогу товаров(OffersScreenAppear).
  - От первого события до оплаты доходит меньше половины пользователей(49%/46,7%/47,4% для групп 246/247/248 соответственно.)
- Результаты эксперимента:

- Для A/A/B-теста пользователей разбили на 3 группы: 2 контрольные(246 и 247) со старыми шрифтами и одну экспериментальную(248) — с новыми:

```
- В 246 группе 2484 пользователей.
- В 247 группе 2517 пользователей.
- В 248 группе 2537 пользователей.
```

- Проверили контрольные группы - ни для оного из событий разница не оказалось значимой. Обе эти группы можно считать контрольными.
- Провели 16 проверок статистических гипотез с уровнем значимости 0.05 (12 из них проверял разницу между контрольными группами и группой с изменённым шрифтом) и ни одна из них не выявила значимой разницы.

- При уровне значимости 0.1 только одна из проверок покажет значимую разницу, между контрольной группой 246 и экспериментальной в доле перехода пользователей в корзину(CartScreenAppear), но эта разница будет не в пользу нашей экспериментальной группы.
- Но при уровне значимости 0.1 каждый десятый раз можно получать ложный результат, поэтому стоит применить изначально выбранный нами уровень значимости 0.05
- Исходя из результатов данного A/A/B-эксперимента, мы можем судить, что на поведение пользователей изменение шрифта значимого эффекта не оказало. Что можно считать успехом, т.к. целью было узнать не отпугнут ли изменения пользователей. В то же время учитывая результаты эксперимента, если изменение шрифта не продиктовано проблемами в работе приложения, его можно не менять.
- Изменение шрифтов можно провести, однако это не увеличит показатели.