Сборный проект 2: *Событийный анализ воронки продаж. Анализ результатов A/A/B-теста*

Заказчик:

стартап, продающий продукты питания в мобильном приложении.

Цель исследования:

- изучить поведение пользователей мобильного приложения;
- узнать сколько пользователей доходит до покупки;
- узнать на каких этапах "застревают" пользователи, несовершившие покупку;
- оценить целесообразность замены шрифтов во всём приложении.

Этапы исследования:

- изучение воронки продаж (событий);
- анализ результатов A/A/B-теста.

Источники данных:

сервер мобильного приложения.

Данные:

logs_exp.csv - лог-файл с действиями пользователей (событиями).

Структура файла (названия столбцов):

Структура ***logs_exp.csv***:

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

Импорт библиотек

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

Подготовительный этап анализа

Загрузка данных

к списку таблиц проекта

In [2]:
```
# загрузка лог-файла logs_exp.csv
try:
    df0 = pd.read_csv('/datasets/logs_exp.csv', sep='\t')
except:
    df0 = pd.read_csv('https://code.s3.yandex.net/datasets/logs_exp.csv', sep='\t')
```

Проверка качества данных в датасете

In [3]:
```
# Вывод первых 5 строк датасета:
df0.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

df1_info

In [4]:
```
# Вывод общей информации по датасету:
df0.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]:
```
# Количество полных дубликатов по датасету:

df0.duplicated().sum()
```

Out[5]:
```
413
```

In [6]:
```
# уникальные названия событий
print('Уникальные названия событий в приложении:', *df0['EventName'].unique())
print('Количество уникальных названий событий в приложении:', df0['EventName'].nunique())
```

```
Уникальные названия событий в приложении: MainScreenAppear PaymentScreenSuccessful CartScreenAppear OffersScreenAppea
r Tutorial
Количество уникальных названий событий в приложении: 5
```

In [7]:
```
# уникальные номера групп эксперимента
print('Уникальные номера групп эксперимента:', *df0['ExpId'].unique())
print('Количество уникальных номеров групп эксперимента:', df0['ExpId'].nunique())
```

```
Уникальные номера групп эксперимента: 246 248 247
Количество уникальных номеров групп эксперимента: 3
```

Выводы:

- названия столбцов необходимо привести к стандартному виду;
- тип данных в столбце EventTimestamp необходимо заменить на datetime;
- пропусков в датасете нет;
- полных дубликатов в датасете 413, их следует удалить;
- всего 4 столбца, 244126 строк;
- 5 уникальных названий события в приложении:

```
- MainScreenAppear - главный экран;
- PaymentScreenSuccessful - экран оплата прошела успешно;
- CartScreenAppear - экран корзины;
- OffersScreenAppear - экран предложений (экран с каталогом продукции);
- Tutorial - руководство пользователя.
```

- 3 уникальных номера групп, участвующих в эксперименте: 246 248 247

Данные соотвтетсвуют ранее заявленным, т.е. в лог-файле содержится следующая информация:

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

Предварительная обработка данных

```
# Приведение названий столбцов к стандартному виду
df0.columns = ['event_name', 'user_id', 'event_dt', 'group']
df0.columns
```

In [8]:
```
df0.rename(columns = {'EventName':'event_name', 'DeviceIDHash':'user_id', 'EventTimestamp':'event_dt', 'ExpId':'group'
df0.columns
```

Out[8]:
```
Index(['event_name', 'user_id', 'event_dt', 'group'], dtype='object')
```

In [9]:
```
# изменение типа данных в столбце event_time_stamp
df0['event_dt'] = pd.to_datetime(df0['event_dt'], unit='s')
df0.dtypes
```

Out[9]:
```
event_name            object
user_id                int64
event_dt      datetime64[ns]
group                  int64
dtype: object
```

In [10]:
```
# визуальная проверка замены тип данных
df0.head(2)
```

Out[10]:

	event_name	user_id	event_dt	group
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	246
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	246

К списку таблиц проекта

In [11]:
```
# создание очищенного от дубликатов датафрейма df
df = df0.copy(deep=True)
df = df.drop_duplicates().reset_index(drop=True)
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
# добавим в df столбец дата
#import warnings
#warnings.filterwarnings("ignore")
#pd.options.mode.chained_assignment = None
df['date'] = df['event_dt'].dt.date
df.head(2)
```

Out[12]:

	event_name	user_id	event_dt	group	date
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	246	2019-07-25
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	246	2019-07-25

df1_info

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   event_name  243713 non-null  object        
 1   user_id     243713 non-null  int64         
 2   event_dt    243713 non-null  datetime64[ns]
 3   group       243713 non-null  int64         
 4   date        243713 non-null  object        
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 9.3+ MB
```

Итак:

- столбцы переименованы;
- дубликаты удалены;
- столбец с датой приведен к типу данных datetime;
- создан столбец date.

Наш новый датафрейм df имеет 5 столбцов и 243713 строки, против исходного df0, состоящего из 4 столбцов и 244126 строк.

In [14]:
```
# Проверка на попадание пользователей в разные группы
test = df.groupby('user_id').agg({'group':'nunique'}).sort_values(by='group', ascending=False)
test['group'].value_counts()
```

Out[14]:
```
1    7551
Name: group, dtype: int64
```

Вывод:

Пользователей, одновременно попавших в разные группы, нет.

Предварительный анализ данных

In [15]:
```
# узнаем сколько всего событий в логе
print('Всего в логе', df['user_id'].count(), 'событий')
```

```
Всего в логе 243713 событий
```

In [16]:
```
# узнаем сколько всего пользователей в логе
print('Всего в логе', df['user_id'].nunique(), 'пользователей')
```

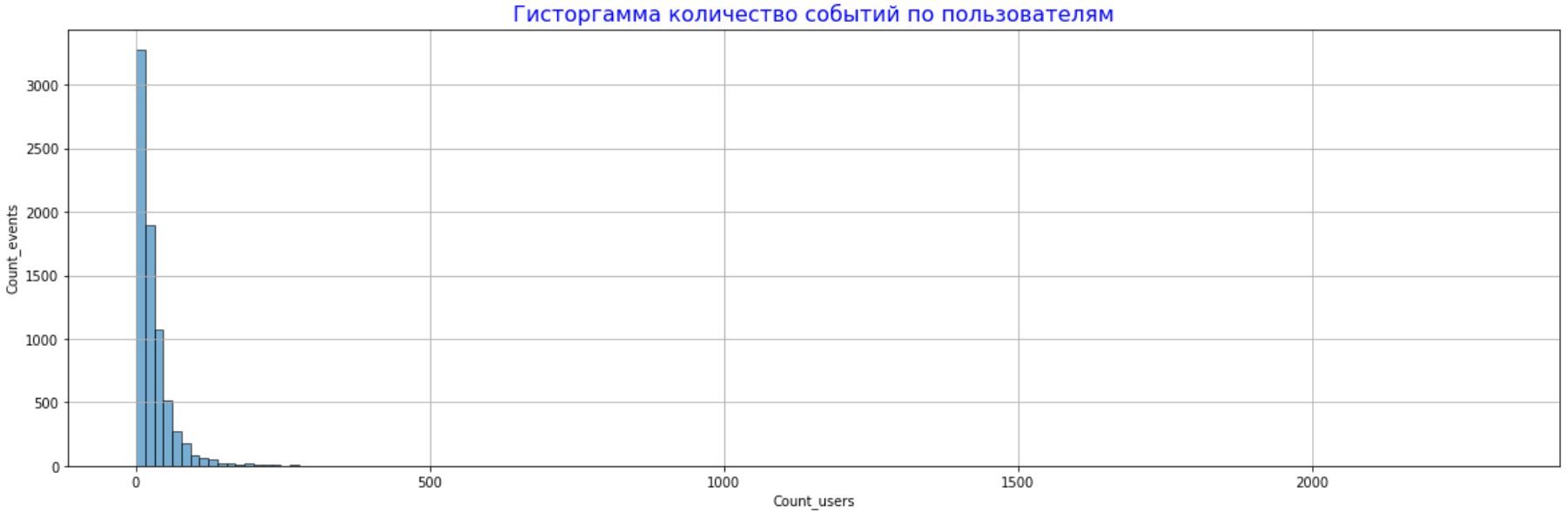
```
Всего в логе 7551 пользователей
```

In [17]:
```
# узнаем сколько в среднем событий приходится на пользователя
print('В среднем на одного пользователя приходится', round(df['user_id'].count()/df['user_id'].nunique()), 'события')
```

```
В среднем на одного пользователя приходится 32 события
```

In [18]:
```
#  Посмотреть на распределение количества событий по пользователям
user_events = df.groupby('user_id').agg(count=('event_name', 'count'))
print(user_events.head(2), end = '\n\n')
print(user_events.describe())

# Построим гистограмму с распределение количества событий по пользователям
user_events.hist(bins = 150, figsize = [20, 6], alpha = 0.6, edgecolor = 'black');
plt.xlabel('Count_users')
plt.ylabel('Count_events')
plt.title('Гисторгамма количество событий по пользователям', size=16, color='blue')
plt.show()
```

```
                 count
user_id
6888746892508752     1
6909561520679493     5

              count
count  7551.000000
mean     32.275593
std      65.154219
min       1.000000
25%       9.000000
50%      20.000000
75%      37.000000
max    2307.000000
```



Распределение количества событий по пользователям имеет сильное смещение вправо, 75-й квартиль равен 37 событиям на пользователя, а максимальное количество - 2307. В качестве описательной статистики лучше выбрать медиану - 20 событий на пользователя.

In [19]:
```
# узнаем за какой период представлены данные в логе
print('Начало периода:', df['event_dt'].min())
print('Окончание периода:', df['event_dt'].max())
print('Продолжительность периода:', df['event_dt'].max()-df['event_dt'].min())
```

```
Начало периода: 2019-07-25 04:43:36
Окончание периода: 2019-08-07 21:15:17
Продолжительность периода: 13 days 16:31:41
```

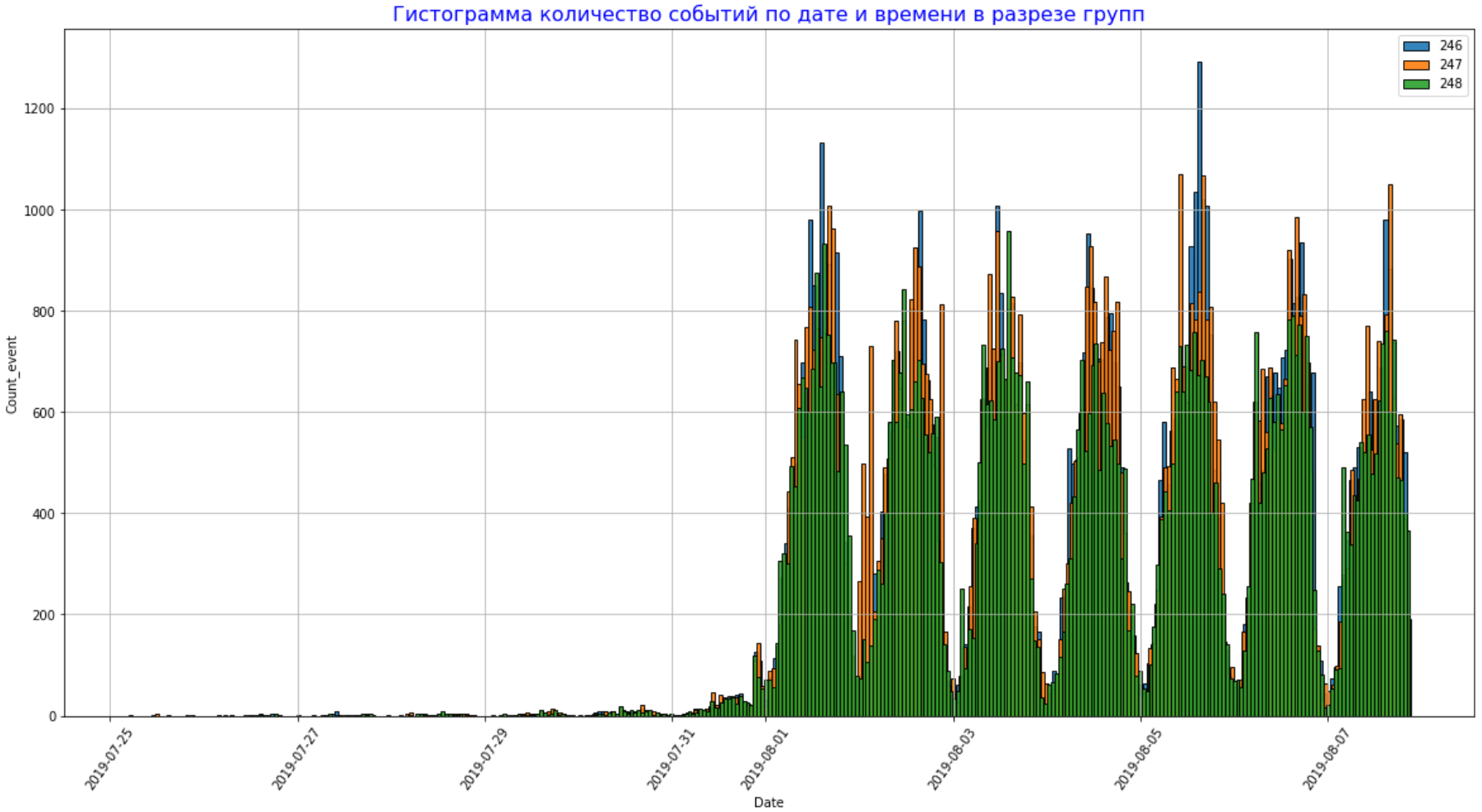
In [20]:
```
# построим гистограмму по дате и времени
df['event_dt'].hist(figsize=[20, 6], bins=336,  alpha=0.6, edgecolor = 'black')
plt.xlabel('Date')
plt.ylabel('Count_event')
plt.title('Гисторгамма количество событий по дате и времени', size=16, color='blue')
plt.show()
```

Данные становятся полными с 2019-08-01, более ранние данные следует отбросить.

Прослеживается четкая зависимость количества событий от времени суток.

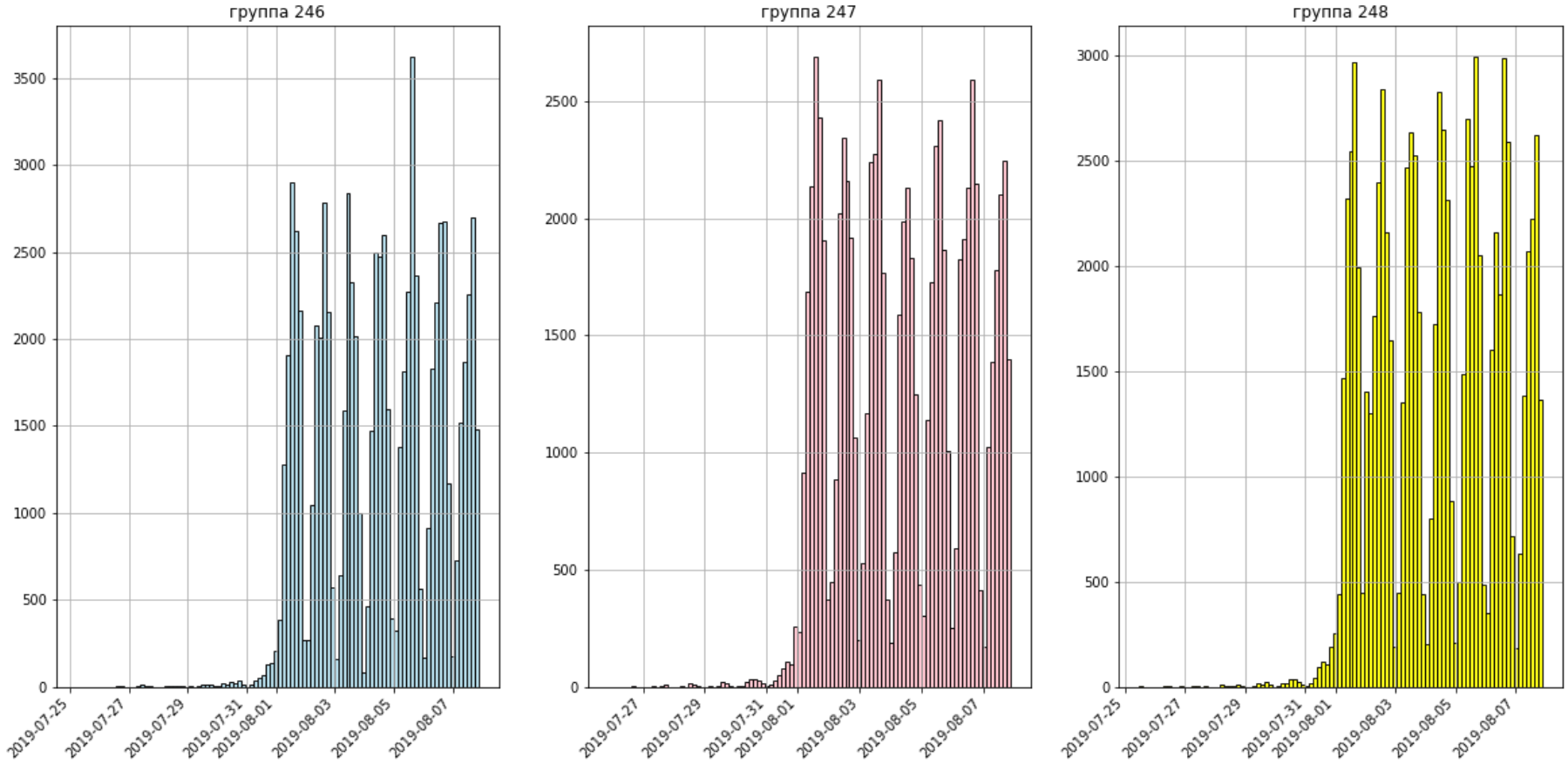
Перед удалением неполных данных интересно посмотреть гистограмму в разрезе групп.

In [21]:
```
# построим гистограмму по дате и времени в разрезе групп
for i in df['group'].unique():
    df[df['group']==i]['event_dt'].hist(figsize=[20, 10], bins=336, alpha=0.9, edgecolor = 'black')
plt.xlabel('Date')
plt.ylabel('Count_event')
plt.xticks(rotation=55)
plt.legend([246, 247, 248])
plt.title('Гистограмма количество событий по дате и времени в разрезе групп', size=16, color='blue')
plt.show;
```



In [22]:
```
fig, ax = plt.subplots(1, 3, figsize=(10,6))
j = 0
a = ['lightblue', 'pink', 'yellow']
for i in sorted(df['group'].unique()):
    plt.title(f'группа {i}')
    plt.xticks(rotation=45)
    df[df['group']==i]['event_dt'].hist(figsize=[20, 10], bins=100, alpha=0.9, edgecolor = 'black', ax = ax[j], color=
    ax[j].set_title(f'группа {i}')
    j += 1
fig.autofmt_xdate(rotation=45)
fig.suptitle(f'Гистограмма количество событий по дате и времени в разрезе групп', size=16, color='blue')
plt.show();
```

Гистограмма количество событий по дате и времени в разрезе групп




```
# еще один вариант графика мне на память
plt.figure(figsize=(15, 35))
ax = sns.countplot(y=df['event_dt'].dt.strftime('%Y-%m-%d %H'), hue='group', data=df, dodge=False, alpha=0.8)
ax.set_title('График количества событий по дате и времени')
plt.xlabel('Count_event')
plt.ylabel('Date')
plt.show()
```

Внутри групп пользователи ведут себя примерно одинаково.

Полные данные у нас за 7 дней с 01 по 07 августа 2019 г.

Во всех трех группах есть зависимость количества событий от времени суток.

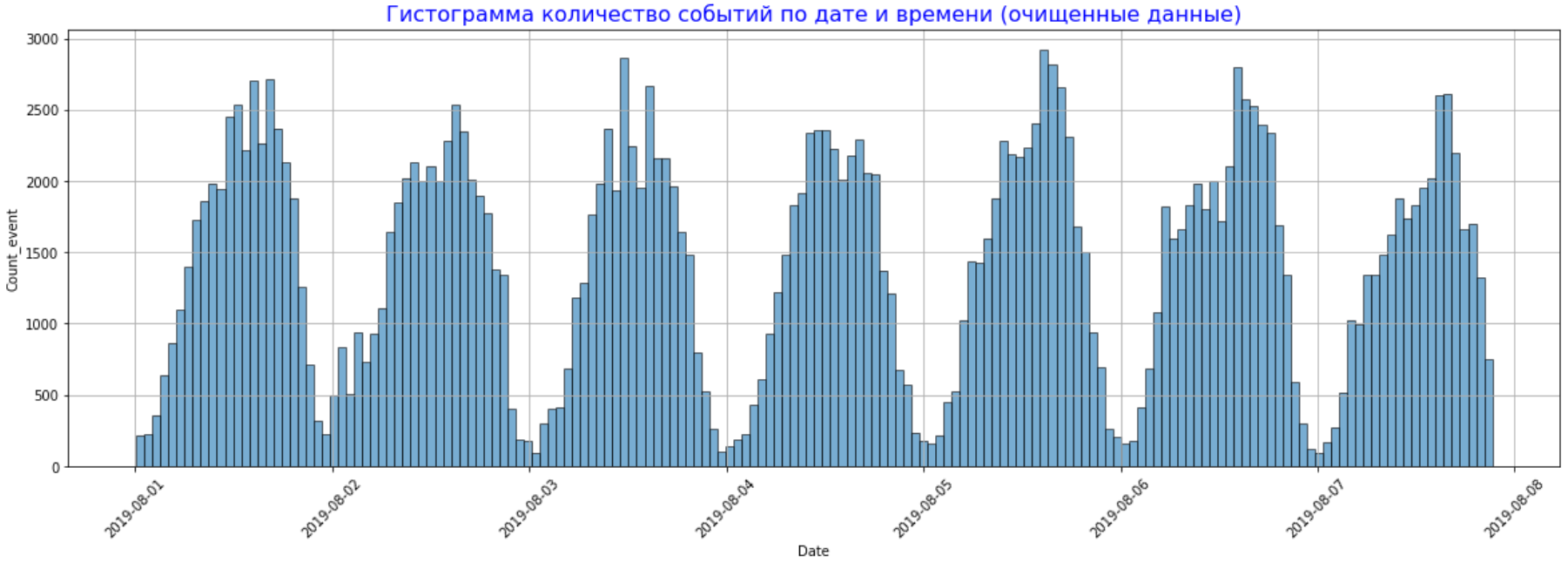
К списку таблиц проекта

In [23]:
```
# создадим новый датафрейм df1, очищенный от неполных данных и включающий период с 01 по 07 августа 2019 г. включитель
df1 = df.copy(deep=True)
df1 = df1.loc[df1['event_dt']>='2019-08-01']
df1.head(2)
```

Out[23]:

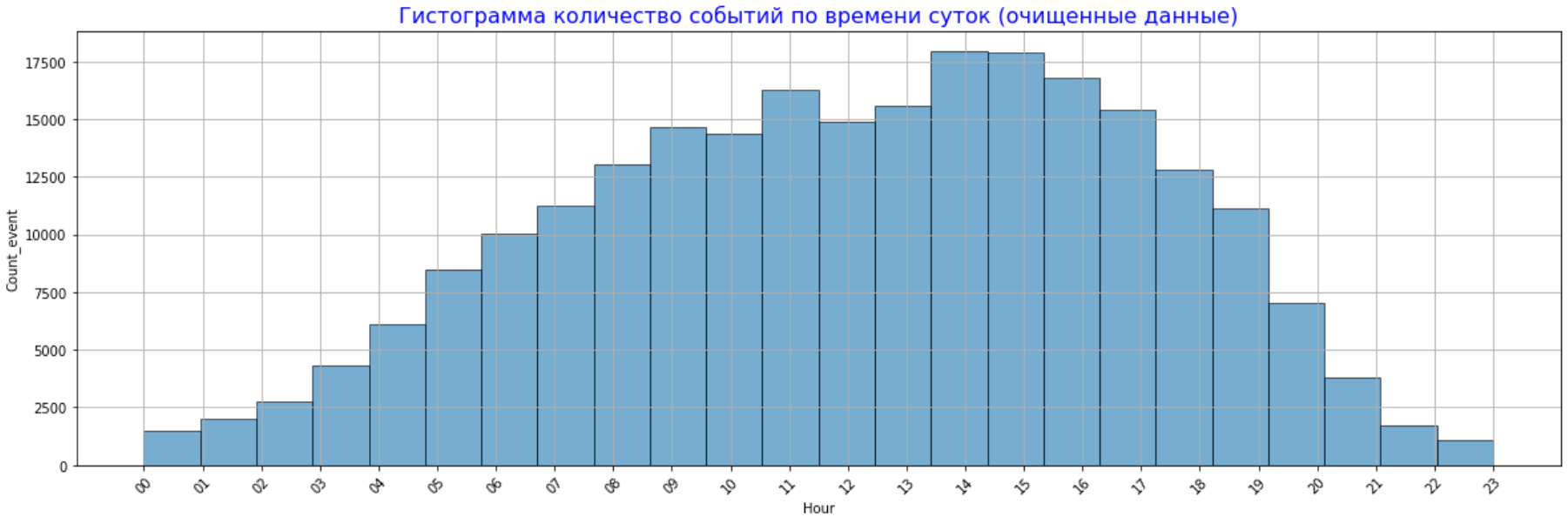
	event_name	user_id	event_dt	group	date
2826	Tutorial	3737462046622621720	2019-08-01 00:07:28	246	2019-08-01
2827	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	246	2019-08-01

In [24]:
```
# построим гистограмму по дате и времени по датафрейму с очищенными данными
df1['event_dt'].hist(figsize=[20, 6], bins=168,  alpha=0.6, edgecolor = 'black')
plt.xlabel('Date')
plt.ylabel('Count_event')
plt.xticks(rotation=45)
plt.title('Гистограмма количество событий по дате и времени (очищенные данные)', size=16, color='blue')
plt.show()
```



В глаза бросается внутридневная сезонность. Визуализируем ее.(выводы делать не будем, просто посмотрим)

In [25]:
```
# построим гистограмму  распределения событий по времени суток
df1['event_dt'].dt.strftime('%H').hist(figsize=[20, 6], bins=24,  alpha=0.6, edgecolor = 'black')
plt.xlabel('Hour')
plt.ylabel('Count_event')
plt.xticks(rotation=45)
plt.title('Гистограмма количество событий по времени суток (очищенные данные)', size=16, color='blue')
plt.show()
```



In [26]:
```
# проанализируем изменение датафрейма
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 240887 entries, 2826 to 243712
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   event_name  240887 non-null  object        
 1   user_id     240887 non-null  int64         
 2   event_dt    240887 non-null  datetime64[ns]
 3   group       240887 non-null  int64         
 4   date        240887 non-null  object        
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 11.0+ MB
```

df_info

df0_info

In [27]:
```
# сравним количество событий до и после очистки данных
#print('Количество событий в исходном файле (справочно):', df0['event_name'].count())
print('Количество событий до очистки данных:', df['event_name'].count())
print('Количество событий после очистки данных:', df1['event_name'].count())
print('Количество удаленных событий (количество событий, произошедших в период неполных данных):', df['event_name'].co
print('Процент удаленных событий от общего числа событий до очистки данных:', ((df['event_name'].count() - df1['event_
```

```
Количество событий до очистки данных: 243713
Количество событий после очистки данных: 240887
Количество удаленных событий (количество событий, произошедших в период неполных данных): 2826
Процент удаленных событий от общего числа событий до очистки данных: 1.16
```

In [28]:
```
# сравним количество пользователей до и после очистки данных
print('Количество пользователей после очистки данных:', df1['user_id'].nunique())
print('Количество пользователей до очистки данных:', df['user_id'].nunique())
print('Абсолютное изменение количества пользователей после очистки данных:', df1['user_id'].nunique() - df['user_id'].
print('Относительное изменение количества пользователей после очистки данных:', (df1['user_id'].nunique() - df['user_i
print('Медиана количества событий на одного пользователя до очистки данных', user_events.median())
print('Медиана количества событий на одного пользователя после очистки данных', df1.groupby('user_id').agg(count=('eve
```

```
Количество пользователей после очистки данных: 7534
Количество пользователей до очистки данных: 7551
Абсолютное изменение количества пользователей после очистки данных: -17
Относительное изменение количества пользователей после очистки данных: -0.0022513574361011784
Медиана количества событий на одного пользователя до очистки данных count    20.0
dtype: float64
Медиана количества событий на одного пользователя после очистки данных count    19.0
dtype: float64
```

Вывод:

Изменения, полученные в результате удаления из датафрейма неполных данных за период с 25 по 31 июля 2019 г. включительно, можно считать незначительными: удалено 1.15% событий, совершенных 17 пользователями.

В очищенном датафрейме осталось 240887 строк(событий) и 7534 пользователей.

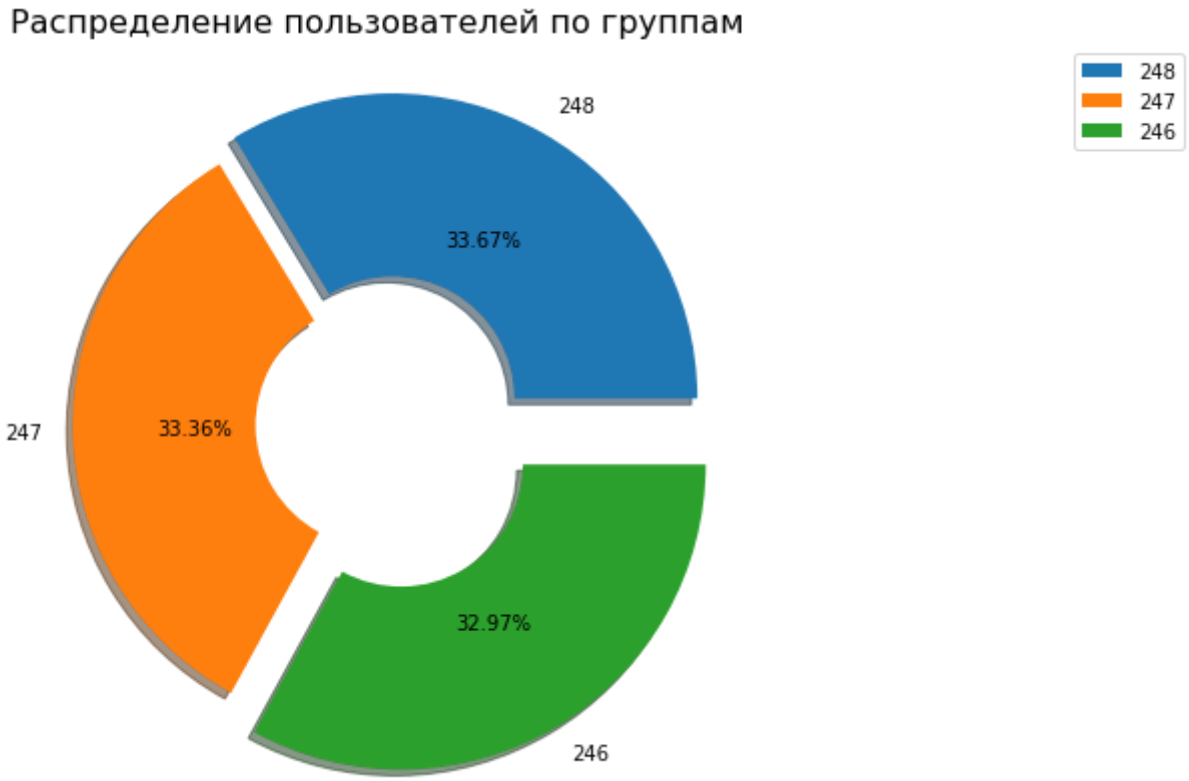
К списку таблиц проекта

In [29]:
```
# узнаем количество пользователей в каждой группе
users_group_count = df1.groupby('group').agg({'user_id':'nunique'})
users_group_count.columns = ['count_users']
users_group_count['count_users_%'] = round(users_group_count['count_users']/df1['user_id'].nunique()*100, 2)
users_group_count = users_group_count.reset_index().sort_values(by='count_users_%', ascending=False)
(
    users_group_count
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['count_users_%'])
)
```

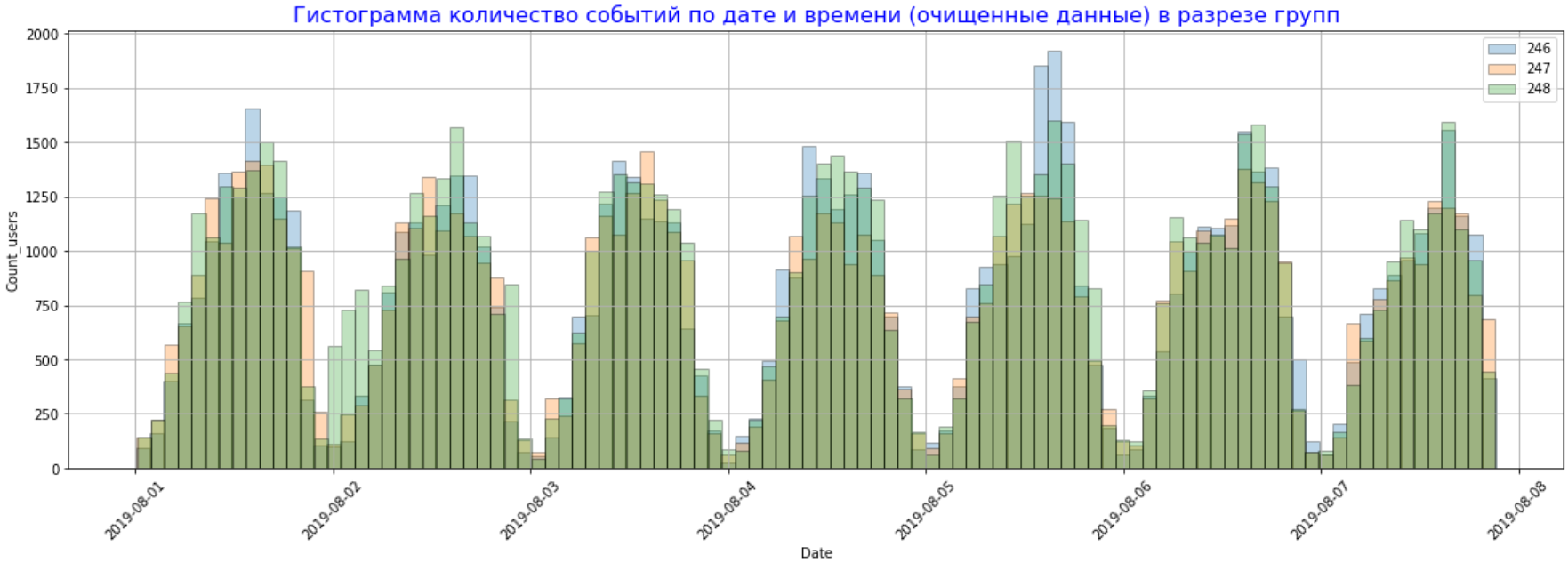
Out[29]:

	group	count_users	count_users_%
2	248	2537	33.670000
1	247	2513	33.360000
0	246	2484	32.970000

In [30]:
```
# Построим круговую диаграмму - количество пользователей в каждой группе
fig = plt.figure(figsize =(7, 7))
plt.title('Распределение пользователей по группам', size=16)
explode = (0.1, 0, 0.15)
plt.pie(list(users_group_count['count_users']), labels = list(users_group_count['group']), wedgeprops=dict(width=0.6),

plt.legend(loc='upper left', bbox_to_anchor=(1.4,1.0))
plt.show()
```



In [31]:
```
# построим гистограмму по дате и времени по датафрейму с очищенными данными в разрезе групп
for i in df1['group'].unique():
    df1[df1['group']==i]['event_dt'].hist(figsize=[20, 6], bins=100,  alpha=0.3, edgecolor = 'black')
plt.xlabel('Date')
plt.ylabel('Count_users')
plt.legend([246, 247, 248])
plt.xticks(rotation=45)
plt.title('Гистограмма количество событий по дате и времени (очищенные данные) в разрезе групп', size=16, color='blue'
plt.show()
```



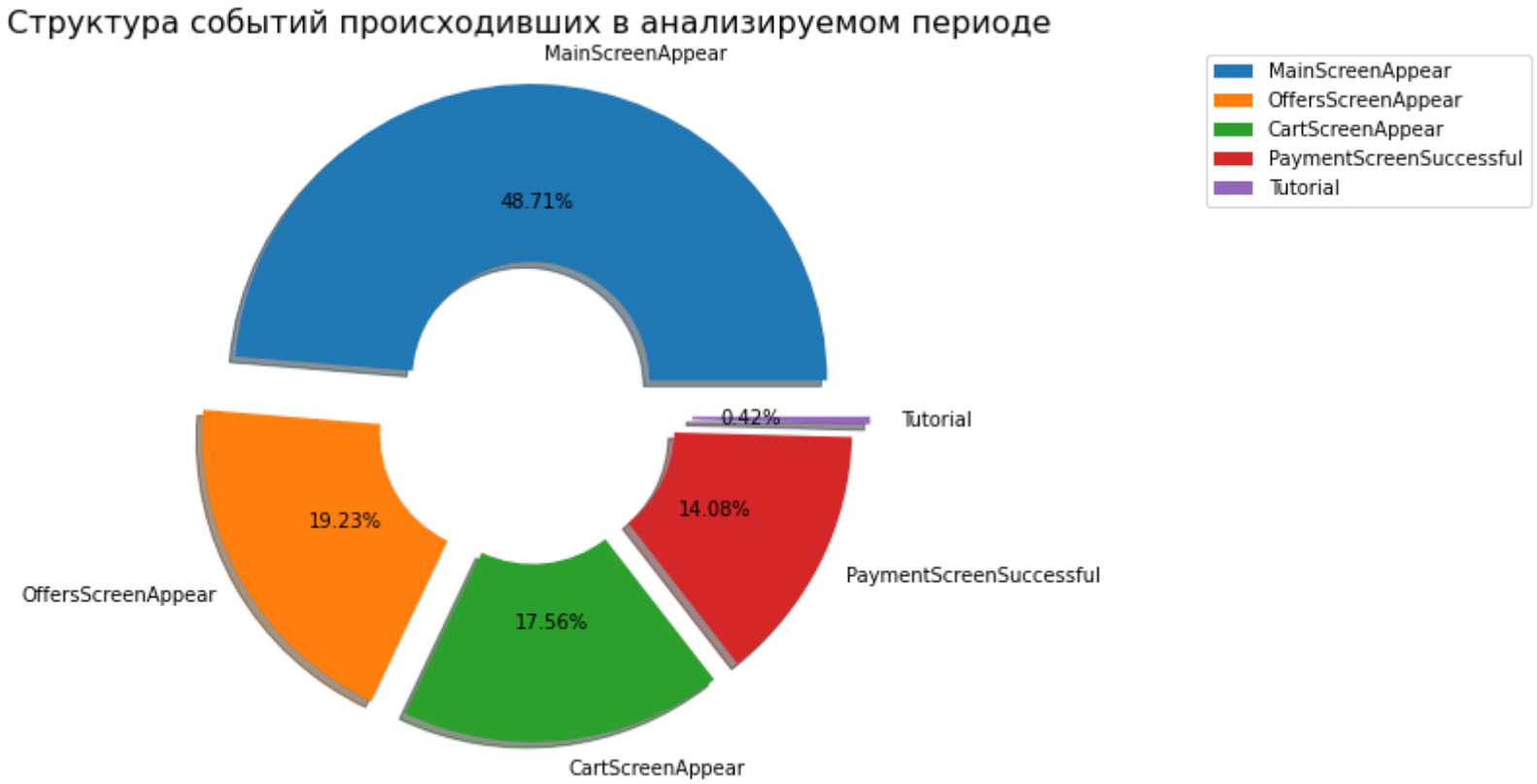
К списку таблиц проекта

In [32]:
```
# узнаем количество событий, произошедших в анализируемом периоде
event_count = df1.groupby('event_name').agg({'event_name':'count'})
event_count.columns = ['count_event']
event_count['count_event_%'] = round(event_count['count_event']/df1['event_name'].count()*100, 2)
event_count = event_count.reset_index().sort_values(by='count_event_%', ascending=False)
(
    event_count
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['count_event_%'])
)
```

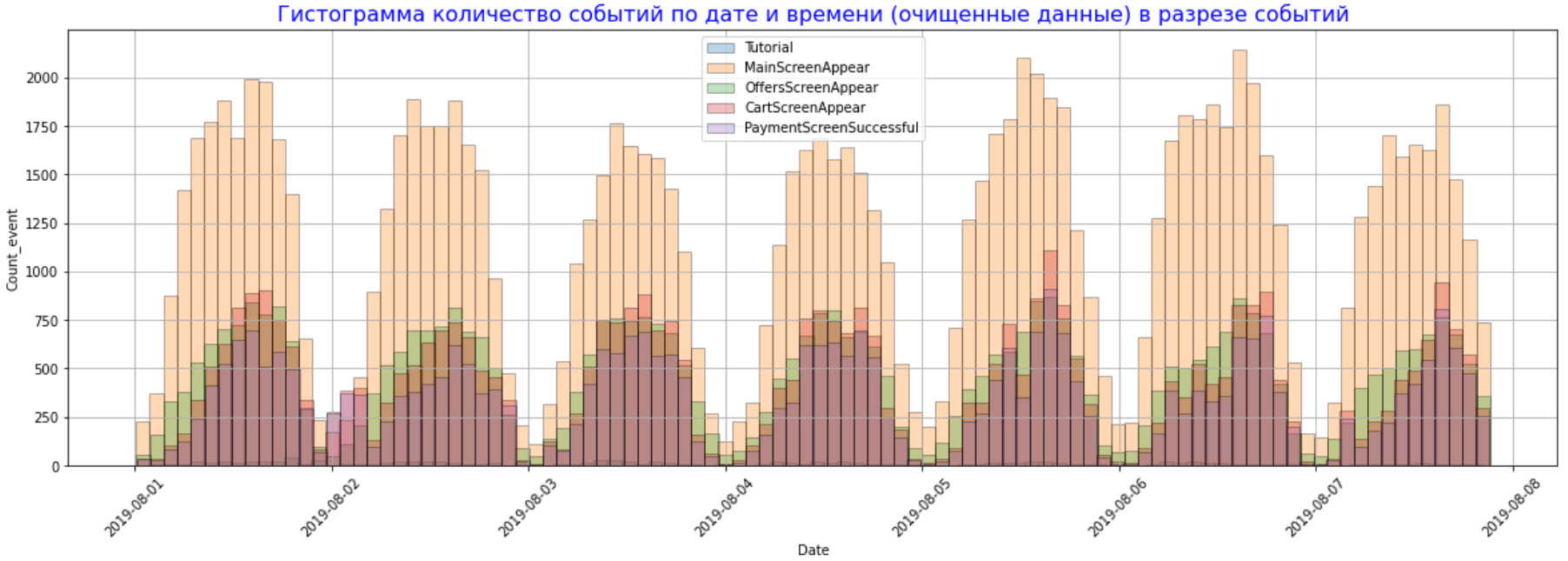
Out[32]:

	event_name	count_event	count_event_%
1	MainScreenAppear	117328	48.710000
2	OffersScreenAppear	46333	19.230000
0	CartScreenAppear	42303	17.560000
3	PaymentScreenSuccessful	33918	14.080000
4	Tutorial	1005	0.420000

In [33]:
```
# Построим круговую диаграмму - количество событий в анализируемом периоде
fig = plt.figure(figsize =(7, 7))
explode = (0.12, 0.12, 0.1, 0.1, 0.15)
plt.title('Структура событий происходивших в анализируемом периоде', size=16)
plt.pie(list(event_count['count_event']), labels = list(event_count['event_name']), wedgeprops=dict(width=0.6), shadow

plt.legend(loc='upper left', bbox_to_anchor=(1.4,1.0))
plt.show();
```



In [34]:
```
# построим гистограмму по дате и времени по датафрейму с очищенными данными в разрезе событий
for i in df1['event_name'].unique():
    df1[df1['event_name']==i]['event_dt'].hist(figsize=[20, 6], bins=100,  alpha=0.3, edgecolor = 'black')
plt.xlabel('Date')
plt.ylabel('Count_event')
plt.xticks(rotation=45)
plt.legend(df1['event_name'].unique())
plt.title('Гистограмма количество событий по дате и времени (очищенные данные) в разрезе событий', size=16, color='blu
plt.show()
```



Выводы:

- пользователи равномерно распределены по группам;
- для всех групп характерно наличие зависимости между количеством событий и временем суток;
- количество событий равномерно распределено по группам;
- наибольшее количество событий приходится на просмотр главной страницы сайта, наименьшее - изучение руководства пользователя;
- распределение событий по дням равномерное.

Таким образом, данные выглядят убедительно, после удаления периода неполных данных, у меня нет сомнений в пригодности базы для дальнейшего анализа.

Событийный анализ воронки продаж

Анализ событий по количеству случаев за анализируемый период

In [35]:
```
# Итак мы знаем какие события есть влогах и как часто они встречаются
event_count = df1.groupby('event_name').agg({'event_name':'count'})
event_count.columns = ['count_event']
event_count['count_event_%'] = round(event_count['count_event']/df1['event_name'].count()*100, 2)
event_count = event_count.reset_index().sort_values(by='count_event_%', ascending=False)
(
    event_count
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['count_event_%'])
)
```

Out[35]:

	event_name	count_event	count_event_%
1	MainScreenAppear	117328	48.710000
2	OffersScreenAppear	46333	19.230000
0	CartScreenAppear	42303	17.560000
3	PaymentScreenSuccessful	33918	14.080000
4	Tutorial	1005	0.420000

In [36]:
```
# еще один дублирующий график
fig = plt.figure(figsize =(18, 5))
p = sns.barplot(y='event_name', x='count_event', data=event_count, alpha = 0.6)
fig.suptitle(f'Распределение событий по количеству случаев в анализируемом периоде', size=16, color='blue')
for i in p.patches:
    p.text(i.get_width()+30, i.get_y()+0.5, str(int(i.get_width())), fontsize=12, color='darkblue')
plt.show;
```

Данный раздел не трубует разделения пользователей на группы. Ради практики я потренеровалась, но код, не имеющий отношения к решению поставленной задачи отключила.

К списку таблиц проекта

Вывод:

- самое частое событие - посещение главной страницы, 117328 случаев;
- рехе всего пользователи заходят в руководство пользователя, 1005 случаев;
- каталог посетили 46333 раз, что примерно 40% от количества просмотров главной страницы;
- пользователи посетили свою корзину 42303 раз, что примерно 91% от количества просмотров каталога;
- страница оплаты была открыта 33918 раз, что примерно 80% от количества просмотров карзины.

Анализ событий по числу совершивших их пользователей

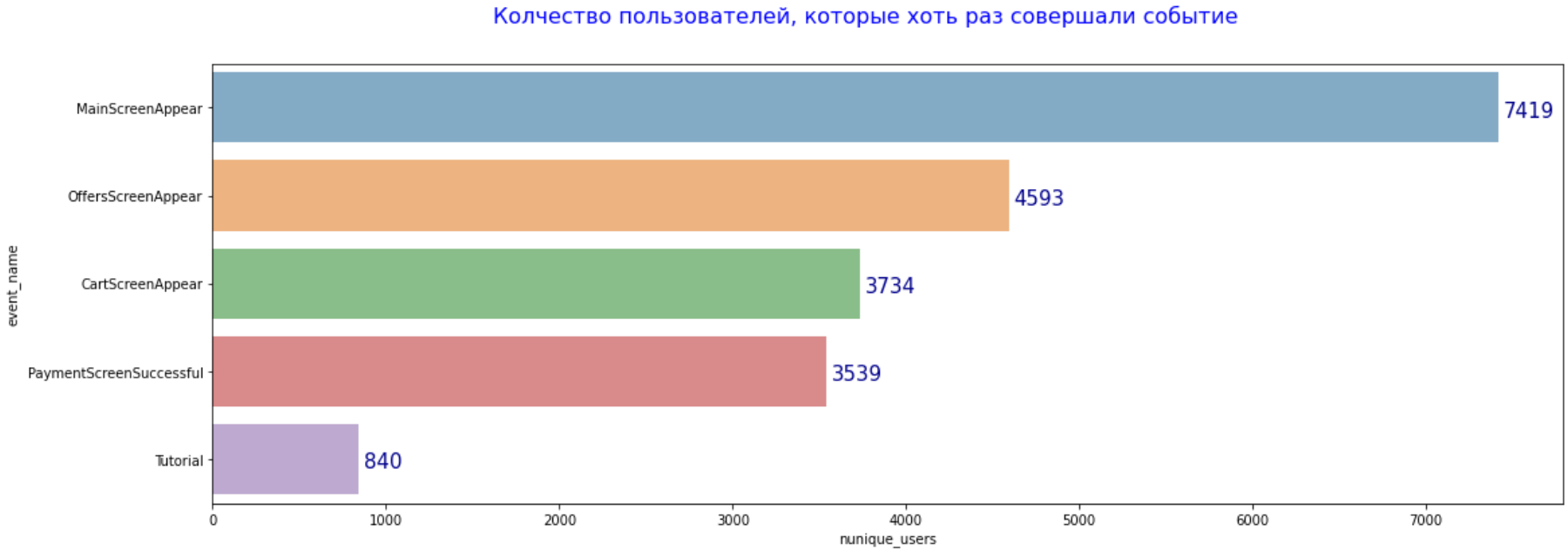
К списку таблиц проекта

In [37]:
```
# Определим количество пользователей, совершивших то или иное событие за анализируемый период
event_count_users = df1.pivot_table(index=['event_name'], values = 'user_id', aggfunc=['nunique']).reset_index()
event_count_users.columns = ['event_name', 'nunique_users']
event_count_users = event_count_users.sort_values(by='nunique_users', ascending = False)

# Посчитаем долю пользователей, которые хоть раз совершали событие
event_count_users['event_count_users_%'] = (event_count_users['nunique_users']/df1['user_id'].nunique()*100).round(2)

(
    event_count_users
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[37]:

	event_name	nunique_users	event_count_users_%
1	MainScreenAppear	7419	98.470000
2	OffersScreenAppear	4593	60.960000
0	CartScreenAppear	3734	49.560000
3	PaymentScreenSuccessful	3539	46.970000
4	Tutorial	840	11.150000

In [38]:
```
# построим график - колчество пользователей, которые хоть раз совершали событие
fig = plt.figure(figsize =(18, 6))
p = sns.barplot(y='event_name', x='nunique_users', data=event_count_users, alpha = 0.6)
fig.suptitle(f'Колчество пользователей, которые хоть раз совершали событие', size=16, color='blue')
for i in p.patches:
    p.text(i.get_width()+30, i.get_y()+0.5, str(int(i.get_width())), fontsize=15, color='darkblue')
plt.show;
```



Вывод:

- на главной странице побывали 98.47% пользователей. Не понятно почему не 100%, возможно, они по ссылке сразу попадали в каталог;
- каталог посетили 61% пользователей. Интересно понять почему 39% пользователей не побывали в каталоге, это технический сбой или причина в чем-то другом?
- 49.56% пользователей хотя бы раз открывали корзину, думаю, это хороший показатель;
- 46.97% пользователей хотя бы раз открывали страницу с успешной оплатой. Интересно, что 94.78% пользователей, побывавших на странице корзины, открывали страницу успешной оплаты. Предположу, что с оплатой проблем нет.

- руководство пользователя открывали 11.15% пользователей, думаю, чем меньше данный процент, тем понятнее пользователю интерфейс нашего приложения.

Предположительный порядок событий, совершаемых пользователей в приложении

Порядок действий пользователя в приложении:

1. Открытие главной страницы - начало работы в приложении;
2. Открытие страницы с каталогом товаров - выбор товаров и перемещение выбранных товаров в корзину;
3. Открытие страницы корзины - проверка деталей заказа и его оформление;
4. Открытие страницы оплаты - оплата и подствержднение успешности платежа.

Событие, связанное с открытием руководства пользователя, в эту цепочку не входит и не будет учитываться при построении воронки событий.

К списку таблиц проекта

In [39]:
```
# Для построения воронки событий создадим новый датафрей, в котором не будет события Tutorial
df2 = event_count_users.copy(deep=True)
df2 = df2.loc[df2['event_name']!='Tutorial']
df2
```

Out[39]:

	event_name	nunique_users	event_count_users_%
1	MainScreenAppear	7419	98.47
2	OffersScreenAppear	4593	60.96
0	CartScreenAppear	3734	49.56
3	PaymentScreenSuccessful	3539	46.97

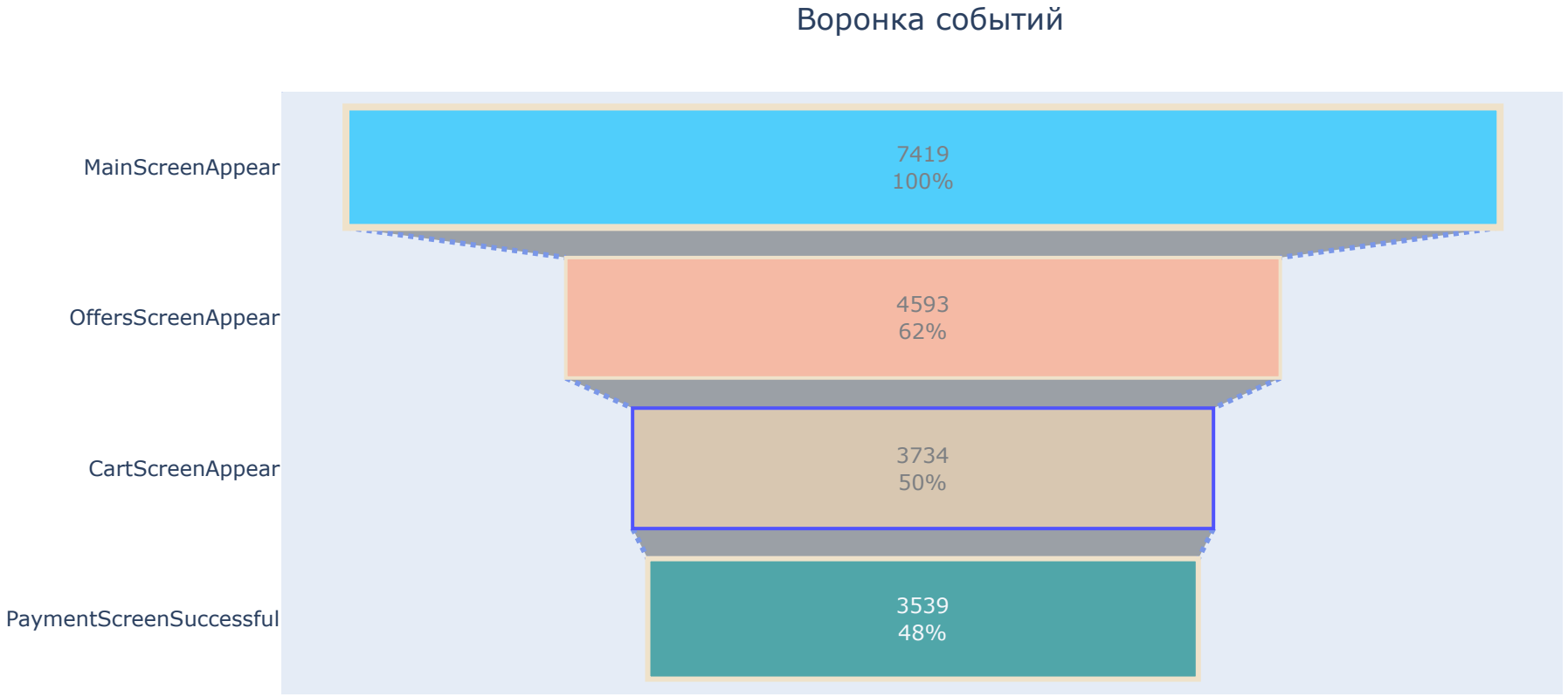
Построение воронки событий

Комментарий:

На курсе мы не проходили построение графика воронки продаж/событий, поэтому воспользуюсь помощью Хабр
https://habr.com/ru/companies/otus/articles/588190/ (https://habr.com/ru/companies/otus/articles/588190/)

Вариант 1: предположим, что все клиенты оплатившие товар, двигались по нашей воронке

In [40]:
```
# построим воронку событий
fig2 = go.Figure(go.Funnel(
    y = df2['event_name'],
    x = df2['nunique_users'],
    textposition = "inside",
    textinfo = "value+percent initial",
    opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
    )

fig2.update_layout(title_text='Воронка событий',title_y=0.9, title_x= 0.55)
fig2.show()
```


Воронка событий
MainScreenAppear
OffersScreenAppear
CartScreenAppear
PaymentScreenSuccessful
7419
100%
4593
62%
3734
50%
3539
48%

In [41]:
```
# Добавим столбец с конверсией - сколько %-ов пользователей перешли на следующую страницу
df2['step_conversion'] = ((df2['nunique_users']/df2['nunique_users'].shift(1))*100).round(2)

# Добавим столбец с конверсией по сравнению с первым событием
df2['total_conversion'] = (df2['nunique_users']/df2['nunique_users'].max()*100).round(2)
df2['total_conversion_delta'] = df2['total_conversion'].shift(1) - df2['total_conversion']
df2 = df2.fillna('-')

(
    df2
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[41]:

	event_name	nunique_users	event_count_users_%	step_conversion	total_conversion	total_conversion_delta
1	MainScreenAppear	7419	98.470000	-	100.000000	-
2	OffersScreenAppear	4593	60.960000	61.910000	61.910000	38.090000
0	CartScreenAppear	3734	49.560000	81.300000	50.330000	11.580000
3	PaymentScreenSuccessful	3539	46.970000	94.780000	47.700000	2.630000

In [42]:
```
d1 = df2['nunique_users'].max()
d2 = df2['nunique_users'].min()

print(f'От первого события до оплаты доходит {d1-d2} пользователей, т.е. {100 - round((d1-d2)/d1*100, 2)}% от числа по
```

```
От первого события до оплаты доходит 3880 пользователей, т.е. 47.7% от числа пользователей посетивших главный экран
```

Вывод:

- с главного экрана (MainScreenAppear) в каталог (OffersScreenAppear) перешло 61.91% пользователей, побывавших на главном экране;
- из каталога (OffersScreenAppear) в корзину (CartScreenAppear) перешло 81.3% пользователей, посетивших каталог;
- из корзины (CartScreenAppear) к оплате (PaymentScreenSuccessful) перешло 94.78% пользователей, побывавших в корзине;

- больше всего пользователей мобильное приложение теряет на переходе с главного экрана (MainScreenAppear) в каталог (OffersScreenAppear) - 38.09%;
- от первого события до оплаты доходит 47.7 % пользователей.

На этом этапе можно было бы остановиться, но что если не все пользователи шли к покупке по нашей воронке, могли быть и прямые ссылки...Проверим.

Вариант 2: на каждом этапе воронки возьмем в расчет только клиентов, прошедших все предыдущие этапы воронки

В исходном датафрейме есть информация об id пользователей, совершавших действия в приложении. Следовательно, если id пользователя, посетившего страницу каталога, нет в списке id пользователей, посетивших главную страницу, их можно мсключить из нашей воронки, так как они пришли в приложение по какой-то ссылке. Эту же логику можно применить к другим этапам нашей воронки.

Работать будем с df3 - копией очищенного от неполных данных датафрейма df1.

К списку таблиц проекта

In [43]:
```
# создадим df3
df3 = df1.copy(deep=True)
```

In [44]:
```
# создадим список событий, состовляющих нашу воронку
event = df2['event_name'].tolist()
event
```

Out[44]:
```
['MainScreenAppear',
 'OffersScreenAppear',
 'CartScreenAppear',
 'PaymentScreenSuccessful']
```

In [45]:
```
# создадим список с количеством клиентов на каждом этапе нашей воронки при условии, что эти клиенты прошли все предыду
funnel = []
last_group = set()
for i in range(len(event)):
    group = set(df3.loc[df3['event_name']==event[i], 'user_id'].unique())
    print(f'Первоначальное количество id пользователей, посетивших страницу {event[i]} равно: {len(group)}')
    if i==0:
        last_group = group
    group = group.intersection(last_group)
    print(f'Количество id пользователей, посетивших и страницу {event[i]}, и все предыдущие страницы равно: {len(group
    print()
    funnel.append(len(group))
    last_group = group
funnel
```

```
Первоначальное количество id пользователей, посетивших страницу MainScreenAppear равно: 7419
Количество id пользователей, посетивших и страницу MainScreenAppear, и все предыдущие страницы равно: 7419

Первоначальное количество id пользователей, посетивших страницу OffersScreenAppear равно: 4593
Количество id пользователей, посетивших и страницу OffersScreenAppear, и все предыдущие страницы равно: 4482

Первоначальное количество id пользователей, посетивших страницу CartScreenAppear равно: 3734
Количество id пользователей, посетивших и страницу CartScreenAppear, и все предыдущие страницы равно: 3580

Первоначальное количество id пользователей, посетивших страницу PaymentScreenSuccessful равно: 3539
Количество id пользователей, посетивших и страницу PaymentScreenSuccessful, и все предыдущие страницы равно: 3429
```

Out[45]:
```
[7419, 4482, 3580, 3429]
```

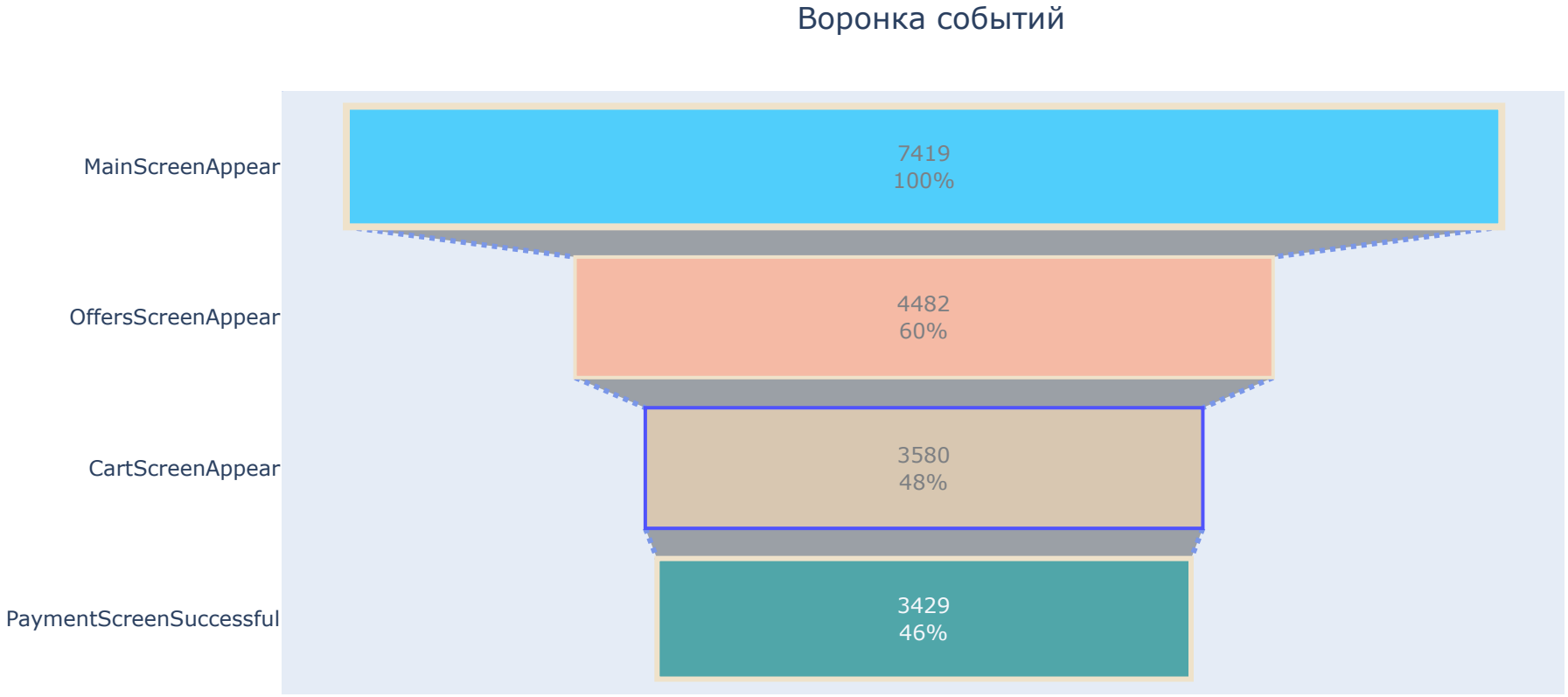
К списку таблиц проекта

In [46]:
```
df4 = df2[['event_name', 'nunique_users']].copy(deep=True)
df4['nunique_users'] = funnel
df4['event_count_users_%'] = (df4['nunique_users']/df1['user_id'].nunique()*100).round(2)
(
    df4
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[46]:

	event_name	nunique_users	event_count_users_%
1	MainScreenAppear	7419	98.470000
2	OffersScreenAppear	4482	59.490000
0	CartScreenAppear	3580	47.520000
3	PaymentScreenSuccessful	3429	45.510000

In [47]:
```
# построим воронку событий
fig3 = go.Figure(go.Funnel(
    y = df4['event_name'],
    x = df4['nunique_users'],
    textposition = "inside",
    textinfo = "value+percent initial",
    opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
    )

fig3.update_layout(title_text='Воронка событий',title_y=0.9, title_x= 0.55)
fig3.show()
```


Воронка событий
MainScreenAppear
OffersScreenAppear
CartScreenAppear
PaymentScreenSuccessful
7419
100%
4482
60%
3580
48%
3429
46%

In [48]:
```
# Добавим столбец с конверсией - сколько %-ов пользователей перешли на следующую страницу
df4['step_conversion'] = ((df4['nunique_users']/df4['nunique_users'].shift(1))*100).round(2)

# Добавим столбец с конверсией по сравнению с первым событием
df4['total_conversion'] = (df4['nunique_users']/df4['nunique_users'].max()*100).round(2)
df4['total_conversion_delta'] = df4['total_conversion'].shift(1) - df4['total_conversion']
df4 = df4.fillna('-')

(
    df4
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[48]:

	event_name	nunique_users	event_count_users_%	step_conversion	total_conversion	total_conversion_delta
1	MainScreenAppear	7419	98.470000	-	100.000000	-
2	OffersScreenAppear	4482	59.490000	60.410000	60.410000	39.590000
0	CartScreenAppear	3580	47.520000	79.880000	48.250000	12.160000
3	PaymentScreenSuccessful	3429	45.510000	95.780000	46.220000	2.030000

Вывод:

По сравнению с предыдущим шагом исследования результат изменился не существенно:

- наибольшее количество пользователей отсеивается на этапе перехода с главного экрана в каталог;
- на втором месте по количеству отсеившихся клиентов стоит переход из каталога в корзину;
- меньше всего пользователей теряется при переходе из корзины на страницу оплаты;
- все 4 этапа воронки прошли 46% пользователей.

Можно продолжить "игры разума" и предположить, что клиенты, подходящие для целей построения нашей воронки должны сначала попасть на главный экран, затем в каталог, затем в корзину, затем на страницу оплаты. То есть в идиале нужно учесть время, когда эти события происходят.

Вариант 3: на каждом этапе воронки возьмем в расчет только клиентов, прошедших последовательно во времени все предыдущие этапы воронки

К списку таблиц проекта

In [49]:
```
# Построим таблицу с последней датой прохождения каждого события в разрезе пользователей
df5 = df3[df3['event_name'] != 'Tutorial'].pivot_table(
        index='user_id',
        columns='event_name',
        values='event_dt',
        aggfunc='min').reset_index()
print('Всего пользователей: ', df5['user_id'].count())
df5.head(2)
```

```
Всего пользователей:  7530
```

Out[49]:

event_name	user_id	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful
0	6888746892508752	NaT	2019-08-06 14:06:34	NaT	NaT
1	6909561520679493	2019-08-06 18:52:58	2019-08-06 18:52:54	2019-08-06 18:53:04	2019-08-06 18:52:58

In [50]:
```
# Удалим строки с пустым значением в столбце MainScreenAppear
df5 = df5.dropna(subset=['MainScreenAppear'])
print('Всего пользователей: ', df5['user_id'].count())
df5.head(2)
```

```
Всего пользователей:  7419
```

Out[50]:

event_name	user_id	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful
0	6888746892508752	NaT	2019-08-06 14:06:34	NaT	NaT
1	6909561520679493	2019-08-06 18:52:58	2019-08-06 18:52:54	2019-08-06 18:53:04	2019-08-06 18:52:58

In [51]:
```
# создадим список с количеством клиентов на каждом этапе нашей воронки при условии, что эти клиенты последовательно во
funnel2 =[]

step_1 = df5['MainScreenAppear'].count()
step_2 = df5.loc[df5['OffersScreenAppear'] > df5['MainScreenAppear'], 'OffersScreenAppear'].count()
step_3 = df5.loc[(df5['OffersScreenAppear'] > df5['MainScreenAppear']) & (df5['CartScreenAppear'] > df5['OffersScreenA
step_4 = (
    df5.loc[(df5['OffersScreenAppear'] > df5['MainScreenAppear'])
            & (df5['CartScreenAppear'] > df5['OffersScreenAppear'])
            & (df5['PaymentScreenSuccessful'] > df5['CartScreenAppear'])
            , 'PaymentScreenSuccessful'].count()
)
funnel2.append(step_1)
funnel2.append(step_2)
funnel2.append(step_3)
funnel2.append(step_4)
funnel2
```

Out[51]:
```
[7419, 4201, 1767, 454]
```

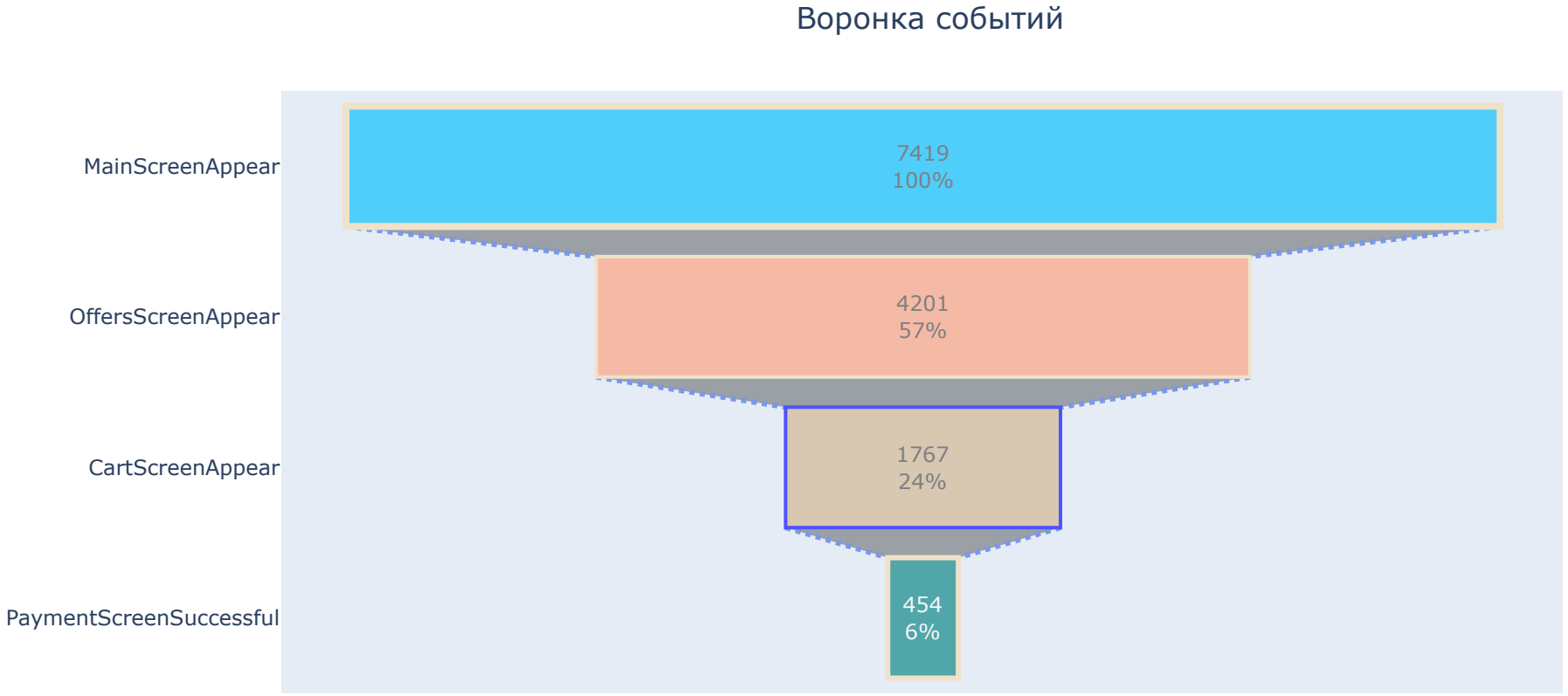
К списку таблиц проекта

In [52]:
```
# Создадим промежуточную таблицу для построения графика воронки
df6 = df2[['event_name', 'nunique_users']].copy(deep=True)
df6['nunique_users'] = funnel2
df6['event_count_users_%'] = (df6['nunique_users']/df1['user_id'].nunique()*100).round(2)
(
    df6
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[52]:

	event_name	nunique_users	event_count_users_%
1	MainScreenAppear	7419	98.470000
2	OffersScreenAppear	4201	55.760000
0	CartScreenAppear	1767	23.450000
3	PaymentScreenSuccessful	454	6.030000

In [53]:
```
# построим воронку событий
fig4 = go.Figure(go.Funnel(
    y = df6['event_name'],
    x = df6['nunique_users'],
    textposition = "inside",
    textinfo = "value+percent initial",
    opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
    )

fig4.update_layout(title_text='Воронка событий',title_y=0.9, title_x= 0.55)
fig4.show()
```


Воронка событий
MainScreenAppear
OffersScreenAppear
CartScreenAppear
PaymentScreenSuccessful
7419
100%
4201
57%
1767
24%
454
6%

In [54]:
```
# Добавим столбец с конверсией - сколько %-ов пользователей перешли на следующую страницу
df6['step_conversion'] = ((df6['nunique_users']/df6['nunique_users'].shift(1))*100).round(2)

# Добавим столбец с конверсией по сравнению с первым событием
df6['total_conversion'] = (df6['nunique_users']/df6['nunique_users'].max()*100).round(2)
df6['total_conversion_delta'] = df6['total_conversion'].shift(1) - df6['total_conversion']
df6 = df6.fillna('-')

(
    df6
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=['nunique_users'])
    .background_gradient(cmap='mako_r', axis=0, subset=['event_count_users_%'])
)
```

Out[54]:

	event_name	nunique_users	event_count_users_%	step_conversion	total_conversion	total_conversion_delta
1	MainScreenAppear	7419	98.470000	-	100.000000	-
2	OffersScreenAppear	4201	55.760000	56.620000	56.620000	43.380000
0	CartScreenAppear	1767	23.450000	42.060000	23.820000	32.800000
3	PaymentScreenSuccessful	454	6.030000	25.690000	6.120000	17.700000

Общий вывод:

- у мобильного приложения хорошая конверсия, 48% пользователей хотя бы раз были на странице успешной оплаты;
- основная масса пользователей отсеивается на этапе перехода с главной страницы в каталог;
- далеко не все пользователи, хотя бы раз были побывавшие на странице успешной оплаты, проходят полные 4 этапа построеной нами воронки;
- при наличии большей информации, можно было бы построить и проанализировать несколько воронок событий.

Анализ результатов A/A/B-теста

Рассчитаем количество пользователей в каждой эксперементальной группе

In [55]:
```
# расчет количества пользователей в группах был проведен мною на этапе первичного знакомства с данными
users_group_count
```

Out[55]:

	group	count_users	count_users_%
2	248	2537	33.67
1	247	2513	33.36
0	246	2484	32.97

Согласно условию проекта, для проведения A/A/B теста пользователей разделили на 3 группы:

- 246 — контрольная группа (2484 пользователей);
- 247 — контрольная группа (2513 пользователей);
- 248 — экспериментальная группа (2537 пользователей).

In [56]:
```
# добавим столбец со средним количеством пользователей
users_group_count['mean_count_users'] = users_group_count['count_users'].mean().round()
# добавим столбец с отклонением количества пользователей в группе от  среднего
users_group_count['delta_%'] = ((users_group_count['count_users'] - users_group_count['mean_count_users'])/users_group
users_group_count
```

Out[56]:

	group	count_users	count_users_%	mean_count_users	delta_%
2	248	2537	33.67	2511.0	1.0354
1	247	2513	33.36	2511.0	0.0796
0	246	2484	32.97	2511.0	-1.0753

К списку таблиц проекта

In [57]:
```
# Построим временную таблицу с количеством каждого событя в разрезе групп
events_group_count = df1.pivot_table(index='group', columns = 'event_name', values = 'user_id', aggfunc = 'count')
events_group_count.loc[:,'Total'] = events_group_count.sum(axis=1)
events_group_count.columns = ['CartScreenAppear', 'MainScreenAppear', 'OffersScreenAppear',
       'PaymentScreenSuccessful', 'Tutorial', 'Total']
#events_group_count.loc['Total',:] = events_group_count.sum(axis=0)
events_group_count['%_total'] = (events_group_count['Total']/events_group_count['Total'].sum()*100).round(2)
events_group_count
```

Out[57]:

group	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial	Total	%_total
246	14690	37676	14767	11852	317	79302	32.92
247	12434	39090	15179	9981	338	77022	31.97
248	15179	40562	16387	12085	350	84563	35.10

Вывод:

Количество пользователей в группах различается на 1-2%, что можно считать приемлемым отклонением.

Однако, количество не может ответить на вопрос: есть или нет статистически значимые отличия в контрольных группах 246 и 247, работающих со старыми шрифтами. Для этих групп необходимо провести проверку гипотезы о равенстве долей.

Проверим разницу между выборками 246 и 247 на статистическую значимость (2 контрольные группы для A/A-эксперимента)

Сформулируем гипотезы:

- H0: статистически значимой разницы между контрольными группами 246 и 247 нет (доли уникальных посетителей, побывавших на этапе воронки, не имеют статистически значимой разницы);
- H1: статистически значимая разница между контрольными группами 246 и 247 есть (доли уникальных посетителей, побывавших на этапе воронки, имеют статистически значимую разницу).

In [58]:
```
# создам список со значениями p-value

list_p_values = []
```

В нашем случае статистически значимую разницу между группами нужно искать по пяти событиям, следовательно удобнее для этих целей написать отдельную функцию.

In [59]:

```python
# Напишим функцию для проверки гипотезы о равенстве доле
def z_test(t_1, t_2, s_1, s_2, alpha):

    successes = np.array([s_1, s_2]
)
    trials = np.array([t_1, t_2])

# пропорция успехов в первой группе:
    p1 = successes[0]/trials[0]

# пропорция успехов во второй группе:
    p2 = successes[1]/trials[1]

# пропорция успехов в комбинированном датасете:
    p_combined = (successes[0] + successes[1]) / (trials[0] + trials[1])

# 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/trials[0] + 1/trials[1]))

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

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

    print(event)
    print('p-значение: ', p_value)

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

    return list_p_values.append(p_value[0])
```

In [60]:
```
# Напишим функцию для построения графика условной воронки

def show_funnel(a, b):
    fig = go.Figure()

    for i in [a, b]:
        fig.add_trace(go.Funnel(
            name = i,
            y = event_group_user_nunique['event_name'],
            x = event_group_user_nunique[i],
            textposition = "inside",
            textinfo = "value+percent initial",
            opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"],
            "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
            connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
)
    fig.update_layout(title_text = f'Условная воронка событий (группы: {a} и {b})',title_y=0.9, title_x= 0.55)
    fig.show()
```

In [61]:
```
# Напишим функцию для запуска теста

def start_test(a, b, alpha):
    print(f'Результаты проверки гипотезы равества долей (группы: {a} и {b})', end = '\n\n')
    for event in events:
        t_1 = group_user_nunique[a]
        t_2 = group_user_nunique[b]
        s_1 = event_group_user_nunique[event_group_user_nunique['event_name']==event][a]
        s_2 = event_group_user_nunique[event_group_user_nunique['event_name']==event][b]
        z_test(t_1, t_2, s_1, s_2, alpha)
    show_funnel(a, b)
```

In [62]:
```
# Список событий
events = df1['event_name'].unique().tolist()
events
```

Out[62]:
```
['Tutorial',
 'MainScreenAppear',
 'OffersScreenAppear',
 'CartScreenAppear',
 'PaymentScreenSuccessful']
```

In [63]:
```
# критический уровень статистической значимости
alpha = 0.05
```

К списку таблиц проекта

In [64]:
```
# На основе таблицы df1 создадим промежуточную таблицу - распределение пользователей по группам
group_user_nunique = df1.pivot_table(columns = 'group', values = 'user_id', aggfunc='nunique').reset_index(drop=True)
group_user_nunique['246+247'] = group_user_nunique[246] + group_user_nunique[247]
group_user_nunique.columns.name = ''
group_user_nunique
```

Out[64]:

	246	247	248	246+247
0	2484	2513	2537	4997

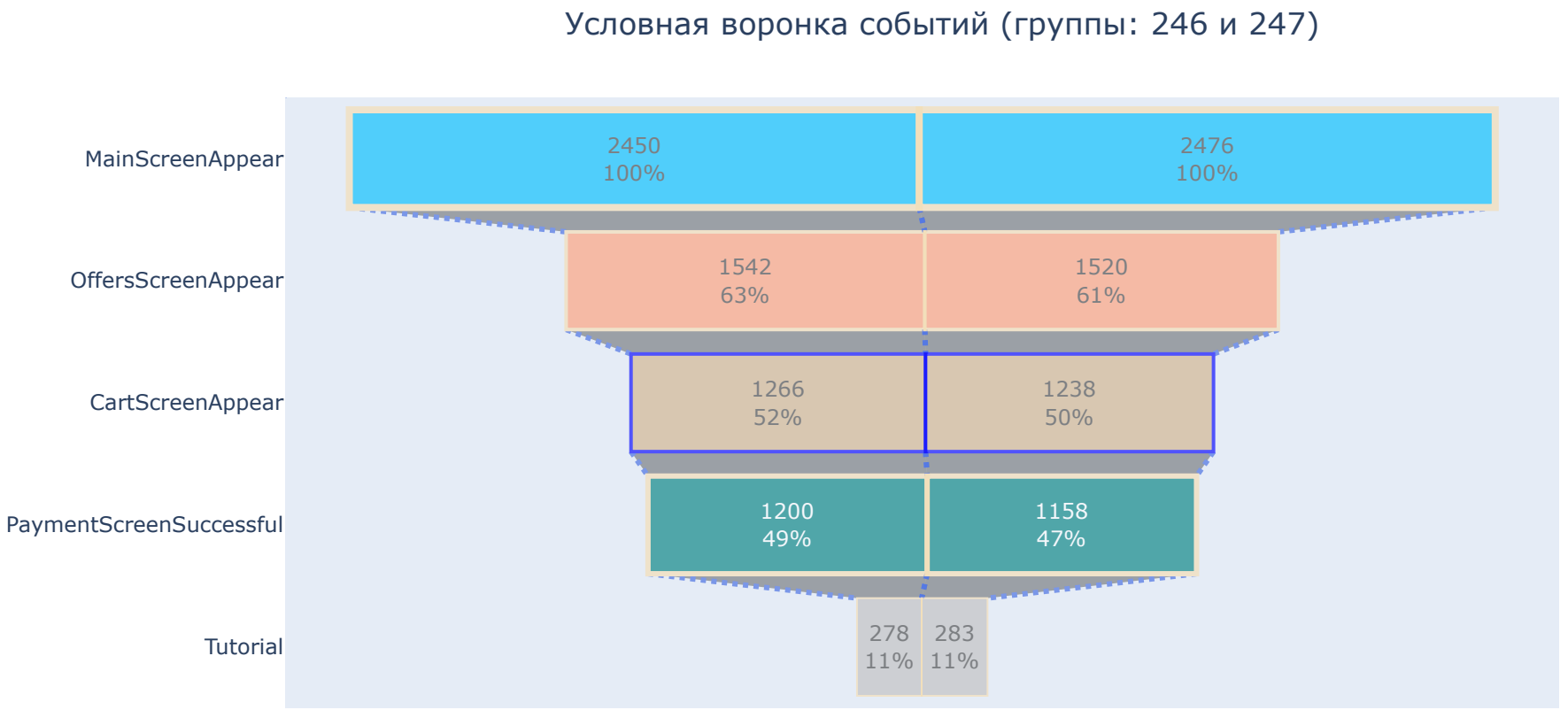
In [65]:
```
# На основе таблицы df1 создадим промежуточную таблицу - распределение пользователей по группам в разрезе событий
event_group_user_nunique = df1.pivot_table(index='event_name', columns='group', values = 'user_id', aggfunc='nunique')
event_group_user_nunique['246+247'] = event_group_user_nunique[246] + event_group_user_nunique[247]
event_group_user_nunique['246_%'] = (event_group_user_nunique[246]/event_group_user_nunique[246].sum()*100).round(2)
event_group_user_nunique['247_%'] = (event_group_user_nunique[247]/event_group_user_nunique[247].sum()*100).round(2)
event_group_user_nunique['248_%'] = (event_group_user_nunique[248]/event_group_user_nunique[248].sum()*100).round(2)
event_group_user_nunique = event_group_user_nunique.reset_index(drop=True)
event_group_user_nunique.columns.name = ''
(
    event_group_user_nunique
    .style
    .background_gradient(cmap='mako_r', axis=0, subset=event_group_user_nunique.columns[1:])
)
```

Out[65]:

	event_name	246	247	248	246+247	246_%	247_%	248_%
0	MainScreenAppear	2450	2476	2493	4926	36.370000	37.090000	37.130000
1	OffersScreenAppear	1542	1520	1531	3062	22.890000	22.770000	22.800000
2	CartScreenAppear	1266	1238	1230	2504	18.790000	18.550000	18.320000
3	PaymentScreenSuccessful	1200	1158	1181	2358	17.810000	17.350000	17.590000
4	Tutorial	278	283	279	561	4.130000	4.240000	4.160000

In [66]:
```
# Проверим гипотезу равенства долей к каждому событию (группы 246 и 247)
start_test(246, 247, alpha)
```

```
Результаты проверки гипотезы равества долей (группы: 246 и 247)

['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.93769962]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.75705972]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.24809546]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.22883372]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.11456679]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



Вывод:

- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольными группами 246 и 247 нет (доли равны);
- группы 246 и 247 могут быть использованы в качестве контрольных групп в A/A/B тесте.

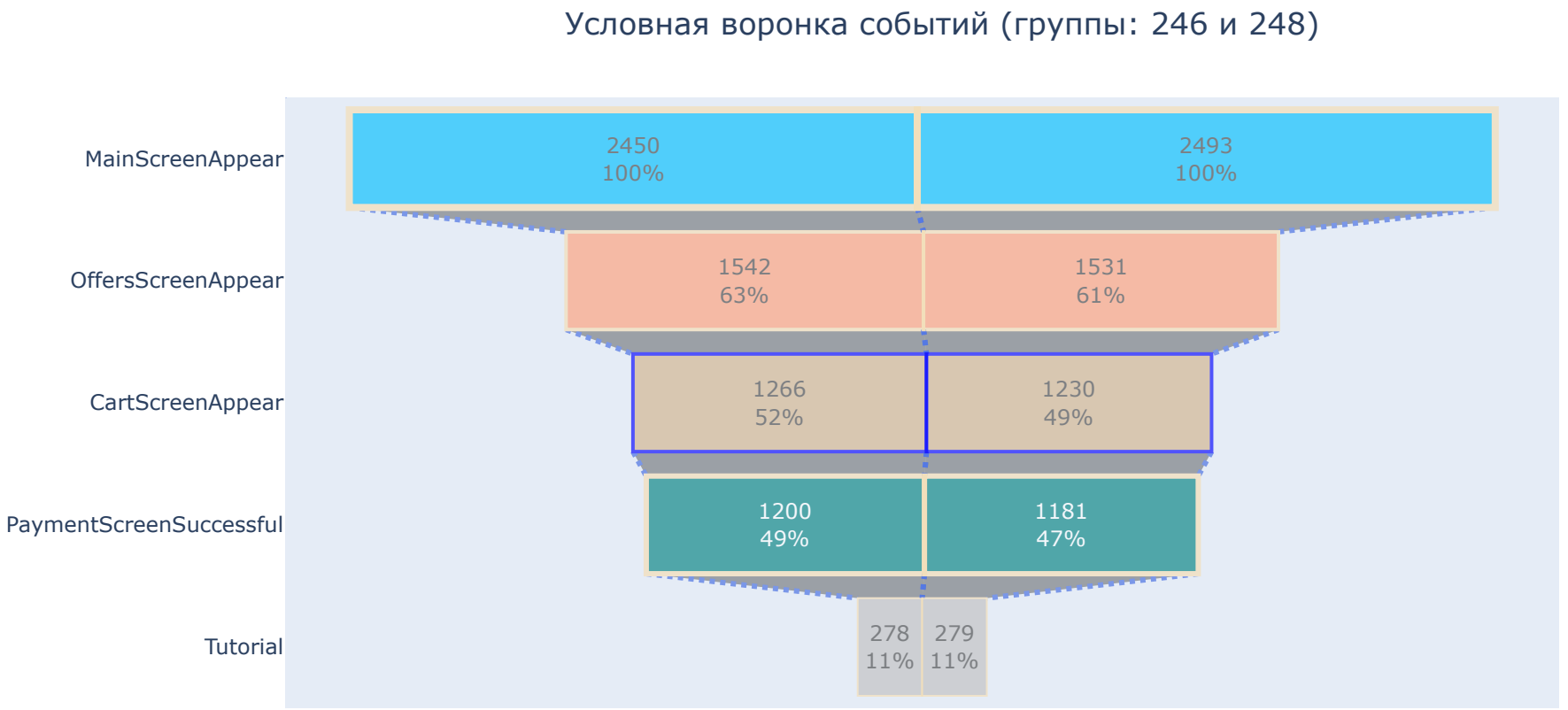
Проверим разницу между выборками 246 и 248 на статистическую значимость (246 - контрольная группа, 248 - экспериментальная группа, A/B-эксперимент)

Сформулируем гипотезы:

- H0: статистически значимой разницы между контрольной группой 246 и эксперементальной группой 248 нет (доли уникальных посетителей, побывавших на этапе воронки, не имеют статистически значимой разницы);
- H1: статистически значимая разница между контрольной группой 246 и эксперементальной группой 248 есть (доли уникальных посетителей, побывавших на этапе воронки, имеют статистически значимую разницу).

In [67]:
```
# Проверим гипотезу равенства долей к каждому событию (группы 246 и 248)
start_test(246, 248, alpha)
```

```
Результаты проверки гипотезы равества долей (группы: 246 и 248)

['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.8264294]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.29497219]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.20836205]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.07842923]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.21225533]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



Вывод:

- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольной группой 246 и экспериментальной группой 248 нет (доли равны).

Проверим разницу между выборками 247 и 248 на статистическую значимость (247 - контрольная

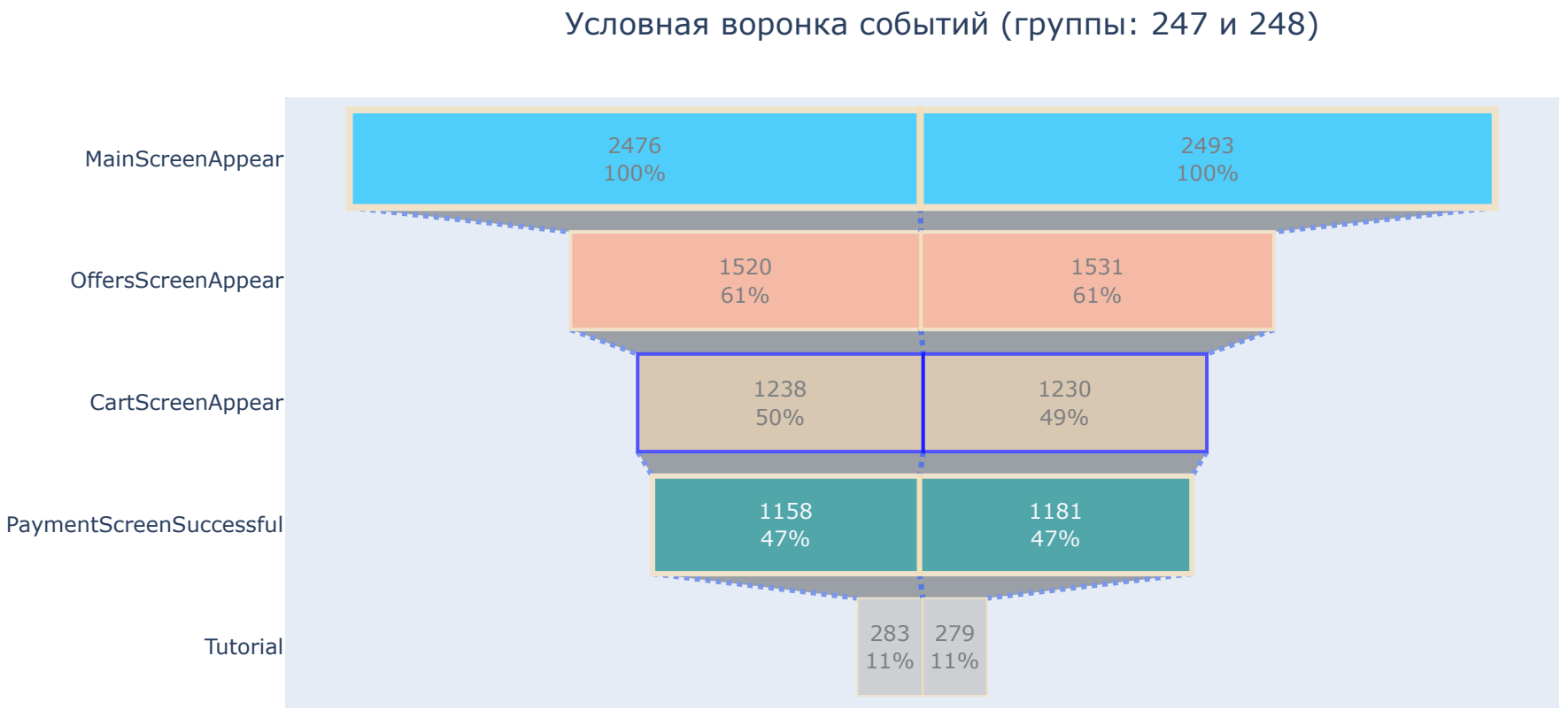
группа, 248 - экспериментальная группа, A/B-эксперимент)

Сформулируем гипотезы:

- H0: статистически значимой разницы между контрольной группой 246 и эксперементальной группой 248 нет (доли уникальных посетителей, побывавших на этапе воронки, не имеют статистически значимой разницы);
- H1: статистически значимая разница между контрольной группой 246 и эксперементальной группой 248 есть (доли уникальных посетителей, побывавших на этапе воронки, имеют статистически значимую разницу).

In [68]:
```
# Проверим гипотезу равенства долей к каждому событию (группы 247 и 248)
start_test(247, 248, alpha)
```

```
Результаты проверки гипотезы равества долей (группы: 247 и 248)

['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.76532392]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.45870536]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.91978178]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.57861979]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.73734151]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



Вывод:

- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольной группой 246 и экспериментальной группой 248 нет (доли равны).

Проверим разницу между объединённой контрольной группой (246 + 247) и 248 на статистическую

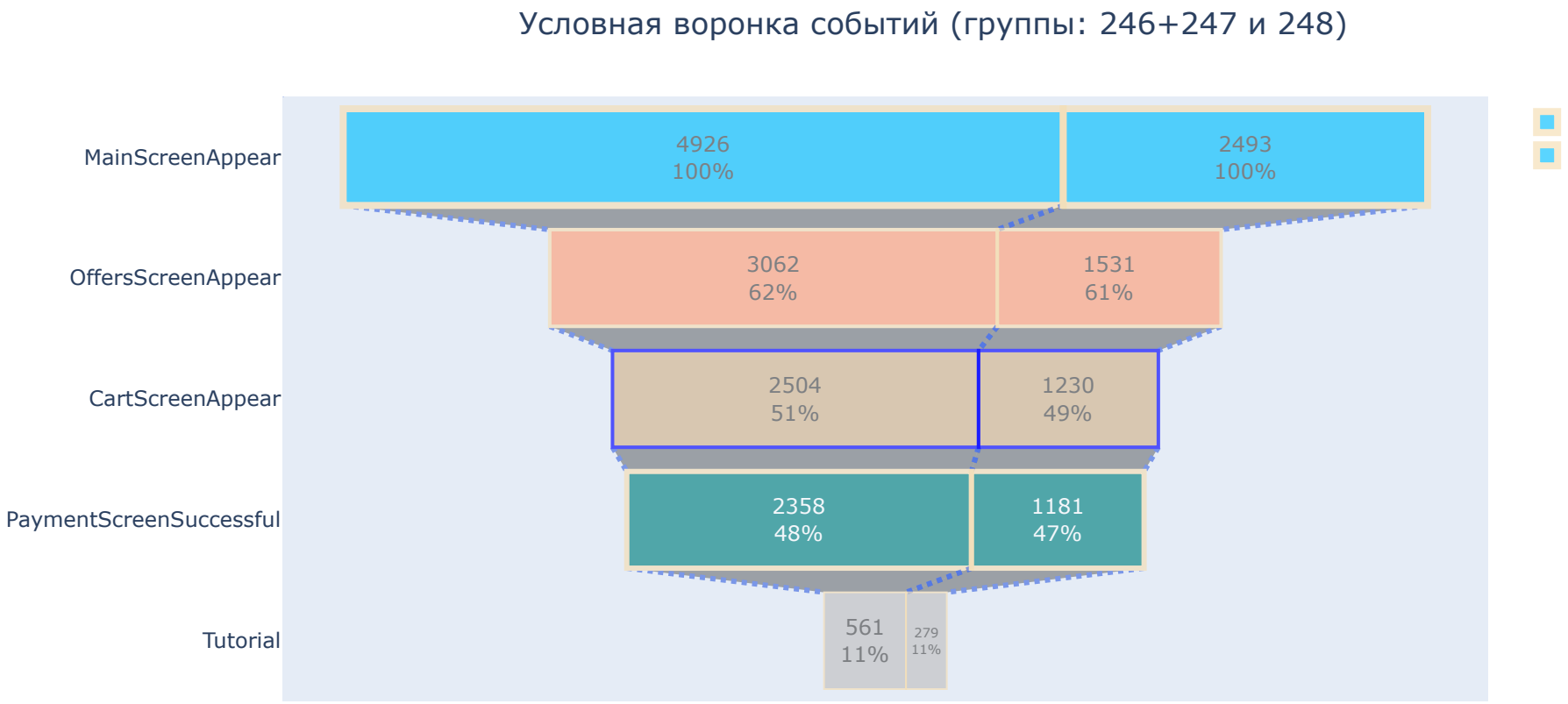
значимость (246, 247 - контрольные группы, 248 - экспериментальная группа, A/B-эксперимент)

Сформулируем гипотезы:

- H0: статистически значимой разницы между объединённой контрольной группой (246 + 247) и эксперементальной группой 248 нет (доли уникальных посетителей, побывавших на этапе воронки, не имеют статистически значимой разницы);
- H1: статистически значимая разница между объединённой контрольной группой (246 + 247) и эксперементальной группой 248 есть (доли уникальных посетителей, побывавших на этапе воронки, имеют статистически значимую разницу).

In [69]:
```
# Проверим гипотезу равенства долей к каждому событию (группы 247 и 248)
start_test('246+247', 248, alpha)
```

```
Результаты проверки гипотезы равества долей (группы: 246+247 и 248)

['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.76486247]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.29424527]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.4342555]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.18175875]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful']
p-значение:  [0.60042943]
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



Вывод:

- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между объединенной контрольной группой (246 + 247) и экспериментальной группой 248 нет (доли равны).

Проанализируем какой уровень значимости стоит применить в множественном сравнении

In [70]:
```
# Выведем на печать список с накопленными результатами тестов
list_p_values
```

Out[70]:
```
[0.9376996189257114,
 0.7570597232046099,
 0.2480954578522181,
 0.22883372237997213,
 0.11456679313141849,
 0.8264294010087645,
 0.2949721933554552,
 0.20836205402738917,
 0.07842923237520116,
 0.2122553275697796,
 0.765323922474501,
 0.4587053616621515,
 0.9197817830592261,
 0.5786197879539783,
 0.7373415053803964,
 0.764862472531507,
 0.29424526837179577,
 0.43425549655188256,
 0.18175875284404386,
 0.6004294282308704]
```

In [71]:
```
# Напишем функцию для расчета уровня значимости для m сравнений:
def multi_alpha(alpha_0, m):
    print(f'Вероятность хотя бы одного ложнопозитивный результат в {m} тестах равна: {round(1-(1 - alpha_0)**m, 2)}')
    print(f'Уровень значимости, расчитанный Методом Бонферрони для {m} тестов, равен: {round(alpha_0/m, 4)}')
    print(f'Уровень значимости, расчитанный Методом Шидака для {m} тестов, равен: {round(1 - (1 - alpha_0)**(1/m), 4)}
```

In [72]:
```
# Применим функцию к нашим данным
multi_alpha(alpha, len(list_p_values))
```

```
Вероятность хотя бы одного ложнопозитивный результат в 20 тестах равна: 0.64
Уровень значимости, расчитанный Методом Бонферрони для 20 тестов, равен: 0.0025
Уровень значимости, расчитанный Методом Шидака для 20 тестов, равен: 0.0026
```

In [73]:
```
# посчитаем сколько раз в проведенных тестах p_value<alpha (ключевое слово - посчитаем)
len([i for i in list_p_values if i<alpha])
```

Out[73]:
```
0
```

In [74]:
```
# Применим функцию к alpha = 0.01
multi_alpha(0.01, len(list_p_values))
```

```
Вероятность хотя бы одного ложнопозитивный результат в 20 тестах равна: 0.18
Уровень значимости, расчитанный Методом Бонферрони для 20 тестов, равен: 0.0005
Уровень значимости, расчитанный Методом Шидака для 20 тестов, равен: 0.0005
```

In [75]:
```
# Применим функцию к alpha = 0.1
multi_alpha(0.1, len(list_p_values))
```

```
Вероятность хотя бы одного ложнопозитивный результат в 20 тестах равна: 0.88
Уровень значимости, расчитанный Методом Бонферрони для 20 тестов, равен: 0.005
Уровень значимости, расчитанный Методом Шидака для 20 тестов, равен: 0.0053
```

Вывод:

В проведенных нами тестах p_value стабильно выше alpha, причем существенно выше. В 20 тестах p_value только 1 раз приблизилось достаточно близко к alpha. Пользователи почти не чувствительны к исследуемому показателю, поэтому уровень значимости равный 0.05 считаю оптимальным.

Итоговый вывод по результатам анализа A/A/B теста:

- выбран уровень значимости 0.05;
- проведено 20 тестов на наличие статистически значимой разницы между группами (5 тестов между контрольными группами и 15 тестов между контрольными и экспериментальной группой);
- ни один тест не выявил статистически значимой разницы между группами.

Таким образом, A/A/B теста показал, что изменение шрифтов в приложении не повлияло на поведение пользователей. Судить о целесообразности замены шрифтов в приложении нужно по каким-то другим критериям.

Выбранный уровень значимости 0.05 считаю оптимальным.

Общий вывод

Выводы по предварительному анализу данных

1. На входе был получен файл с данными logs_exp.csv:

- 4 столбца, 244126 строк;
- пропусков в нет;
- полных дубликатов 413;
- содержание столбцов:
  - EventName — название события;
  - DeviceIDHash — уникальный идентификатор пользователя;
  - EventTimestamp — время события;
  - ExpId — номер группы эксперимента.

2. В ходе первичной обработки данных были выполнены следующие действия:

- названия столбцов изменены и приведены к стандартному виду;
- тип данных в столбце EventTimestamp заменен на datetime;
- удалены полные дубликаты - 413 строк.

3. В ходе предварительного анализа данных было установлено:

- в данных содержится информация о 5 событиях - посещениях одного из экранов:

```
- MainScreenAppear - главный экран;
- PaymentScreenSuccessful - экран оплата прошела успешно;
- CartScreenAppear - экран корзины;
- OffersScreenAppear - экран предложений (экран с каталогом продукции);
- Tutorial - экран руководство пользователя;
```

- в данных содержится информация о 3 группах, участвующих в эксперименте: 246 и 247 — контрольные группы, а 248 — экспериментальная;
- данные становятся полными с 2019-08-01, более ранние данные были удалены: удалено 1.15% событий, совершенных 17 пользователями;
- в данных прослеживается четкая зависимость количества событий от времени суток: дневная активность существенно выше ночной;

В очищенном датафрейме осталось 240887 строк(событий) и 7534 пользователей.

Выводы по анализу воронки событий

Задача проекта:

- изучите воронку событий;
- узнать как пользователи доходят до покупки;
- узнать сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах;
- узнать на каких именно шагах «застревают» клиенты.

Вывод:

- самое частое событие - посещение главной страницы, 117328 случаев, на главной странице побывали 98.47% пользователей;
- реже всего пользователи заходят в руководство пользователя, 1005 случаев, хотя бы раз его открывали 11.15% пользователей. Думаю, чем меньше данный процент, тем понятнее пользователю интерфейс нашего приложения;
- каталог таваров посетили 46333 раз, хотя бы раз его открывали 61% пользователей;
- корзину посетили 42303 раз, хотя бы раз ее открывали 49.56% пользователей;
- страница оплаты была открыта 33918 раз, хотя бы раз ее открывали 46.97% пользователей. Предположу, что с оплатой проблем нет, так как 95.78% пользователей, побывавших в корзине, успешно оплатили свой заказ;
- с главного экрана (MainScreenAppear) в каталог (OffersScreenAppear) перешло 61.91% пользователей, побывавших на главном экране;
- из каталога (OffersScreenAppear) в корзину (CartScreenAppear) перешло 81.3% пользователей, посетивших каталог;
- из корзины (CartScreenAppear) к оплате (PaymentScreenSuccessful) перешло 94.78% пользователей, побывавших в корзине;
- больше всего пользователей мобильное приложение теряет на переходе с главного экрана (MainScreenAppear) в каталог (OffersScreenAppear) - 38.09%;
- на втором месте по количеству отсеившихся клиентов стоит переход из каталога в корзину;
- от первого события до оплаты доходит 47.7 % пользователей.

В ходе исследования было сформулировано предположение, что клиенты проходят в процессе покупки следующий путь:

1. Открытие главной страницы - начало работы в приложении;
2. Открытие страницы с каталогом товаров - выбор товаров и перемещение выбранных товаров в корзину;
3. Открытие страницы корзины - проверка деталей заказа и его оформление;
4. Открытие страницы оплаты - оплата и подствержднение успешности платежа.

Событие, связанное с открытием руководства пользователя, в эту процесс покупки не входит.

Однако, последующий анализ показал, что далеко не все пользователи, хотя бы раз были побывавшие на странице успешной оплаты, проходят полные 4 этапа построеной нами воронки.

Выводы по анализу результатов A/A/B теста

Задача проекта:

- исследовать 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;

- проверить на статистическую значимость различия между группами 246 и 247;
- проверить на статистическую значимость различия между группами 246 и 248;
- проверить на статистическую значимость различия между группами 247 и 248;
- проверить на статистическую значимость различия между объединенной группой (246+247) и 248;
- выяснить, какой шрифт лучше.

Результаты:

- количество пользователей в группах различается на 1-2%, что можно считать приемлемым отклонением;
- выбран уровень значимости 0.05;
- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольными группами 246 и 247 нет;
- группы 246 и 247 могут быть использованы в качестве контрольных групп в A/A/B тесте;

- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольной группой 246 и экспериментальной группой 248 нет;
- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между контрольной группой 247 и экспериментальной группой 248 нет;
- ни по одному событию нет оснований отвергнуть нулевую гепотезу о том, что статистически значимой разницы между объединенной контрольной группой (246+247) и экспериментальной группой 248 нет;

Выводы:

- проведено 20 тестов на наличие статистически значимой разницы между группами (5 тестов между контрольными группами и 15 тестов между контрольными и экспериментальной группой);
- ни один тест не выявил статистически значимой разницы между группами;
- P-значения в тестах достаточно высокие, чтобы изменить результаты теста уровнь значимости нужно было бы поднять до уровня не менее 0.12, что существенно увеличит вероятность ошибки первого рода. Выбранный уровень значимости 0.05 считаю верным.

Таким образом, A/A/B теста показал, что **изменение шрифтов в приложении не повлияло на поведение пользователей**. Судить о целесообразности замены шрифтов в приложении нужно по каким-то другим критериям.

Рекомендации:

- выяснить причину, почему только 98.47% пользователей побывали на главной странице, возможно 1.53% столкнулись с каким-то техническим сбоем;
- выяснить причину, почему самый большой отсев пользователей на переходе с главного экрана в каталог - 38%, возможно пользователи также столкнулись с каким-то техническим сбоем. Маловероятно, что зайдя на главную страницу приложения, клиент не проявил никакого интереса к каталогу товаров;
- проанализировать на какие вопросы клиенты чаще всего ищут ответ в руководстве пользователя, 11.15% пользователей обращались к руководству, наверное, это большой показатель, требующий изучения;
- также следует проанализировать степень удовлетворенности клиентов от обращения к руководству пользователя и узнать, удалось ли пользователям найти ответ на их вопрос и решить возникшую проблему.

Чек-лист готовности проекта

- ☑ открыт файл;
- ☑ файл изучен (выведены первые строки, метод info(), проверка на дубликаты, анализ содержания столбцов);
- ☑ названия столбцов переименованы и приведины к стандартному виду;

- ☑ пропуски и сомнительные данные не обнаружены;
- ☑ обнаружены и удалены 413 дубликатов;
- ☑ тип данных в столбце с датой преобразован в datetime;
- ☑ создан дополнительный столбец date с датой события(без времени);
- ☑ проведен предварительный анализ данных, в ходе которого выявлен и удален период неполных данных;
- ☑ проведен анализ распределения пользователей по группам;
- ☑ проведен анализ распределения пользователей и событий по дням и времени;
- ☑ изучены события в логах и их частота в отчетном периоде;
- ☑ расчитано количество пользователей, совершавших каждое из событий. Расчитана доля пользователей, которые хоть раз совершали событие.
- ☑ определен предположительный порядок событий, который проходит пользователь в приложении;
- ☑ построена воронка событий;
- ☑ расчитана доля пользователей, которые проходят на следующий шаг воронки (от числа пользователей на предыдущем);
- ☑ выявлен шаг, на котором теряется больше всего пользователей;
- ☑ определена доля пользователей, которые доходят от первого события до оплаты;
- ☑ расчитано количество пользователей в каждой экспериментальной группе A/A/B теста;
- ☑ проведена проверка на предмет находят ли статистические критерии разницу между выборками 246 и 247;
- ☑ выберано самое популярное событие. Расчитано число и доля пользователей, совершивших это событие в каждой из контрольных групп. Проверена статистическая достоверность отличий между группами. То же самое проделано с оставшимися событиями (для проверки создана отдельная функция);
- ☑ аналогично с предыдущим пунктом проведены расчеты с группой с изменённым шрифтом;
- ☑ проведено сравнение результатов группы с изменённым шрифтом с результатами каждой из контрольных групп в отдельности по каждому событию;
- ☑ проведено сравнение результатов с объединённой контрольной группой;
- ☑ по результатам анализа A/A/B теста сделаны выводы.

Список таблиц проекта:

1. df0 - исходная таблица, созданная на базе файла logs_exp.csv;
2. df - очищенная от дубликатов таблица;
3. df1 - очищенная от неполных данных таблица df (содержит данные за период с 01 по 07 августа 2019 г. включительно);

4. users_group_count - промежуточная таблица - количество пользователей в разрезе групп, создана на базе df1;
5. event_count - промежуточная таблица - название событий и их количество, создана на базе df1;
6. event_count_group - промежуточная таблица - название событий и их количество в разре групп A/A/B теста, создана на базе df1;
7. event_count_users - промежуточная таблица - количество уникальных пользователей, совершивших хотя бы раз то или иное событие, создана на базе df1;
8. df2 - промежуточная таблица - количество уникальных пользователей, совершивших хотя бы раз то или иное событие, кроме события Tutorial, создана на базе event_count_users (удалено событие Tutorial);
9. df3 - рабочая копия таблицы df1(для дальнейшей модификации);
10. df4 - рабочая копия таблицы двух столбцов df2[['event_name', 'nunique_users']](для дальнейшей модификации);
11. df5 - промежуточная таблица с последней датой прохождения каждого события в разрезе пользователей, создана на базе df3;
12. df6 - промежуточная таблица для построения графика воронки, создана на базе df2;
13. group_user_nunique - промежуточная таблица распределение пользователей по группам, создана на базе df1.
14. event_group_user_nunique - промежуточная таблица распределение пользователей по группам в разрезе событий, создана на базе df1.
15. events_group_count - промежуточная таблица с количеством каждого события в разрезе групп, создана на базе df1.