ТЗ:

```
Цель данного проекта - выявить, влияет ли изменения шрифта в
приложении на поведение пользователей.
Имеем датасет с действиями пользователей, которые разбиты на 3 группы:
2 контрольные и 1 экспериментальная, пользователи в этой группе
пользовались приложением с новым шрифтом.
```

План:

```
1. Загрузить данные. Обработать для расчетов и анализа.
2. Изучить данные. Определить количество событий и пользователей,
период времени, за который представлены данные.
2. Расчитать и изучить воронку событий.
3. Сформулировать гипотезы, провести статистические тесты для их
проверки.
4. Вывод.
```

Проект: Событийная аналитика

Цель проекта - разобраться, как ведут себя пользователи мобильного приложения.

Описание данных

В нашем распоряжении датасет: logs_exp.csv

Загрузка данных и подготовка их к анализу

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

In [2]:
```
data = pd.read_csv('/datasets/logs_exp.csv', sep = '\t') #сохраняем в датафрейм
```

In [3]:
```
data.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]:
```
data.columns = data.columns.str.lower() # приводим названия заголовков к строчным бу
```

In [5]:
```
data.rename(
            columns={'eventname': 'event_name', 'deviceidhash': 'device_id_hash',\
                     'eventtimestamp':'event_timestamp','expid':'exp_id' },\
            inplace=True
)
```

In [6]:
```
data.duplicated().sum()# проверим датайфрейм на дубликаты
```

Out[6]:
```
413
```

В датафрейме 413 дублирующих записей

In [7]:
```
duplicate = data[data.duplicated()]
```

In [8]:
```
duplicate.head()
```

Out[8]:

	event_name	device_id_hash	event_timestamp	exp_id
453	MainScreenAppear	5613408041324010552	1564474784	248
2350	CartScreenAppear	1694940645335807244	1564609899	248
3573	MainScreenAppear	434103746454591587	1564628377	248
4076	MainScreenAppear	3761373764179762633	1564631266	247
4803	MainScreenAppear	2835328739789306622	1564634641	248

In [9]:
```
data.drop_duplicates(inplace=True)# удалим дубликаты
```

In [10]:
```
data.isna().sum()# проверим датайфрейм на наличие пропусков
```

Out[10]:
```
event_name         0
device_id_hash     0
event_timestamp    0
exp_id             0
dtype: int64
```

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243713 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   event_name       243713 non-null  object
 1   device_id_hash   243713 non-null  int64 
 2   event_timestamp  243713 non-null  int64 
 3   exp_id           243713 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 9.3+ MB
```

Добавим столбец даты и времени, а также отдельный столбец дат:

In [12]:
```
data['event_data'] = pd.to_datetime(data['event_timestamp'], unit='s').dt.date
```

In [13]:
```
data['event_data'] = pd.to_datetime(data['event_data'] )
```

In [14]:
```
data['event_timestamp'] = pd.to_datetime(data['event_timestamp'], unit='s')
```

In [15]:
```
data.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_id_hash   243713 non-null  int64         
 2   event_timestamp  243713 non-null  datetime64[ns]
 3   exp_id           243713 non-null  int64         
 4   event_data       243713 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(2), object(1)
memory usage: 11.2+ MB
```

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

Out[16]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
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
2	PaymentScreenSuccessful	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25
3	CartScreenAppear	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25
4	PaymentScreenSuccessful	6217807653094995999	2019-07-25 11:48:42	248	2019-07-25

Подведем итог обзора данных:

- мы привели названия столбцов к общепринятому виду;
- изменили типы данных в соответсвии с их содержанием;
- проверили данные на наличие дубликатов: явные дубликаты удалены;
- проверили данные на наличие пропусков. Пропусков не обнаружено.

Изучим и проверим данные

Найдем количество событий в логе

In [17]:
```
event_count = data.shape[0]
```

In [18]:
```
print(f'Количество событий в логе: {event_count}' )
```

```
Количество событий в логе: 243713
```

In [19]:
```
event = data['event_name'].unique().tolist()
```

In [20]:
```
print(f'Количество уникальных событий в логе: {len(event)}. События: {", ".join (eve
```

```
Количество уникальных событий в логе: 5. События: MainScreenAppear, PaymentScreenSuc
cessful, CartScreenAppear, OffersScreenAppear, Tutorial
```

Найдем сколько всего пользователей в логе

In [21]:
```
id_count = data['device_id_hash'].nunique()
print(f'Количество уникальных пользователей в логе: {id_count}')
```

```
Количество уникальных пользователей в логе: 7551
```

Посмотрим, есть ли пользователи, которые находятся в нескольких группах:

In [22]:
```
m = data.pivot_table(index='device_id_hash', values='exp_id', aggfunc='nunique').res
m[m['exp_id']>1].count()
```

Out[22]:
```
device_id_hash    0
exp_id            0
dtype: int64
```

Таких пользователей нет.

Найдем сколько в среднем событий приходится на пользователя

In [23]:
```
# сгруппируем данные по пользователям
y = data.pivot_table(index='device_id_hash', values='event_name', aggfunc='count').r
y.head()
```

Out[23]:

	device_id_hash	event_name
0	6888746892508752	1
1	6909561520679493	5
2	6922444491712477	47
3	7435777799948366	6
4	7702139951469979	137

In [24]:
```
#так как виден большой разброс по количеству событий на пользователя, найдем медианн
print('В среднем на пользователя приходится событий:', y['event_name'].median())
```

```
В среднем на пользователя приходится событий: 20.0
```

In [25]:
```
y['event_name'].max()
```

Out[25]:
```
2307
```

In [26]:
```
y['event_name'].min()
```

Out[26]: 1

In [27]:
```
y2=y[y['event_name']>=300]
```

In [28]:
```
y['event_name'].describe()
```

Out[28]:
```
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 [29]:
```
warning = y2['device_id_hash'].tolist()# сохраним в список пользователей, которые со
```

Посмотрим, какие события совершали пользователи в списке warning

In [30]:
```
data_warning = data.query('device_id_hash in @warning')
data_warning.pivot_table(
    index='device_id_hash', columns = 'event_name',
    aggfunc={'event_name':'count'}
).reset_index().head()
```

Out[30]:

	device_id_hash				
event_name		**CartScreenAppear**	**MainScreenAppear**	**OffersScreenAppear**	**Paym**
0	197027893265565660	932.0	93.0	107.0	
1	624637828747274142	16.0	137.0	150.0	
2	674541835027541643	232.0	21.0	28.0	
3	1055544042902443290	379.0	52.0	58.0	
4	1100007125648169445	312.0	14.0	41.0	

In [31]:
```
data_warning.head()
```

Out[31]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
721	OffersScreenAppear	1989685320445148348	2019-07-30 18:09:41	247	2019-07-30
1161	CartScreenAppear	1100007125648169445	2019-07-31 12:02:11	248	2019-07-31
1298	CartScreenAppear	4257848154605915902	2019-07-31 13:36:50	247	2019-07-31
1299	PaymentScreenSuccessful	4257848154605915902	2019-07-31 13:36:50	247	2019-07-31
1662	CartScreenAppear	1872978591788881482	2019-07-31 17:07:15	247	2019-07-31

Определим за какой период времени данные.

Определим минимальную и максимальную даты в логе:

In [32]:
```
print('Минимальная дата в логе:', data['event_data'].min())
```

```
Минимальная дата в логе: 2019-07-25 00:00:00
```

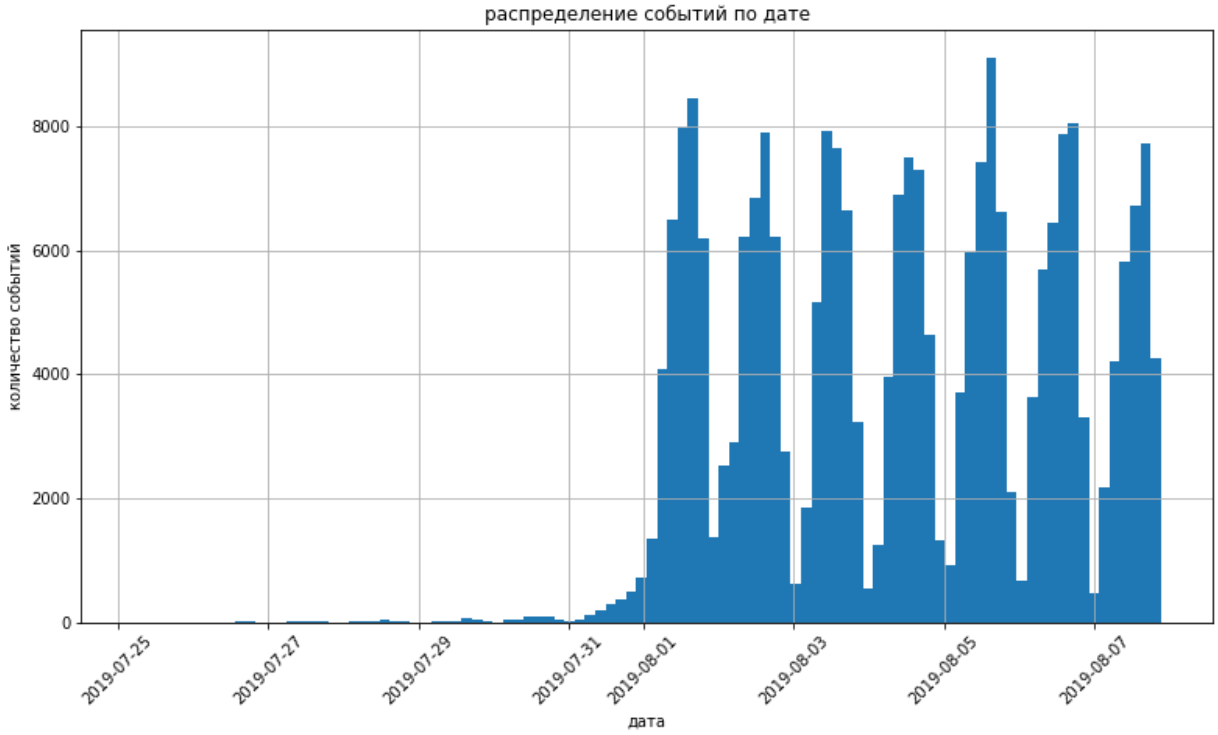
In [33]:
```
print('Максимальная дата в логе:', data['event_data'].max())
```

```
Максимальная дата в логе: 2019-08-07 00:00:00
```

Построим гистограмму по дате и времени

In [34]:
```
plt.figure(figsize=(13, 7))
#plt.set_xticklabels(df.index,rotation=90)
plt.xticks(rotation=45)
data['event_timestamp'].hist(bins=100)
plt.xlabel('дата', fontsize=10)
plt.ylabel('количество событий',fontsize=10)
plt.title('распределение событий по дате', fontsize=12)
```

Out[34]:
```
Text(0.5, 1.0, 'распределение событий по дате')
```



Отбросим данные, раньше 01 августа 2019 года: по гистограме мы видим, что данные за период с 25 июля 2019 года до 01 августа 2019 года не полные. В итоге мы имеем данные за неделю: с 01 по 07 августа 2019 года.

Данные за 31.07 к концу дня в логе записывались полные, но не будем учитывать эти несколько часов. При расчете метрик, нам нужны полные данные за весь день. Так же видим, что активность пользователей в приложении в течени суток меняется, в середине дня-самая высокая, так же засисит от дня недели- будни, либо выходной (события смещаются к вечеру).

In [35]:
```
data.groupby('event_data', as_index=False).agg({'event_name':'count'})
```

Out[35]:

	event_data	event_name
0	2019-07-25	9
1	2019-07-26	31
2	2019-07-27	55
3	2019-07-28	105
4	2019-07-29	184
5	2019-07-30	412
6	2019-07-31	2030
7	2019-08-01	36141
8	2019-08-02	35554
9	2019-08-03	33282
10	2019-08-04	32968
11	2019-08-05	36058
12	2019-08-06	35788
13	2019-08-07	31096

In [36]:
```
data_new = data.query('event_data > "2019-07-31"')# исключим события ранее 01.08.201
```

In [37]:
```
# посмотрим, попали ли аномальные пользователи в срез
data_warning = data_new.query('device_id_hash in @warning')
data_warning.pivot_table(index='exp_id',  values='device_id_hash', aggfunc='nunique'
```

Out[37]:

	exp_id	device_id_hash
0	246	10
1	247	7
2	248	15

In [38]:
```
data_warning.query('event_name =="CartScreenAppear" and device_id_hash == 6304868067
```

Out[38]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
32947	CartScreenAppear	6304868067479728361	2019-08-01 18:01:43	248	2019-08-01
33019	CartScreenAppear	6304868067479728361	2019-08-01 18:04:19	248	2019-08-01
33036	CartScreenAppear	6304868067479728361	2019-08-01 18:04:53	248	2019-08-01
33068	CartScreenAppear	6304868067479728361	2019-08-01 18:05:43	248	2019-08-01
33120	CartScreenAppear	6304868067479728361	2019-08-01 18:06:46	248	2019-08-01

In [39]:
```
data_warning.query('exp_id ==246').pivot_table(
    index='device_id_hash',  columns = 'event_name',
    aggfunc={'event_name':'count'}
).reset_index()
```

Out[39]:

event_name	device_id_hash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	Paym
0	197027893265565660	931.0	93.0	107.0	
1	624637828747274142	16.0	129.0	142.0	
2	1055544042902443290	379.0	52.0	57.0	
3	3337471580007169353	527.0	57.0	77.0	
4	4705890939205361551	188.0	8.0	6.0	
5	6613527411922640441	217.0	21.0	25.0	
6	7738058666231999878	413.0	38.0	68.0	
7	8118046800480174342	321.0	94.0	46.0	
8	8173190940950873857	252.0	33.0	21.0	
9	8309980273750324949	159.0	20.0	34.0	

In [40]:

```
data_warning.query('exp_id ==247').pivot_table(
    index='device_id_hash',  columns = 'event_name',
    aggfunc={'event_name':'count'}
).reset_index()
```

Out[40]:

event_name	device_id_hash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	Paym
0	1754140665440434215	596	10	45	
1	1872978591788881482	144	4	40	
2	1989685320445148348	349	11	21	
3	2768580714092136945	22	191	291	
4	4257848154605915902	148	4	89	
5	4623191541214045580	778	91	137	
6	8192783670889455257	138	10	24	

In [41]:

```
data_warning.pivot_table(index='device_id_hash', values='event_name', aggfunc='count
```

Out[41]:

device_id_hash	event_name
0 197027893265565660	1996
1 624637828747274142	297
2 674541835027541643	496
3 1055544042902443290	853
4 1100007125648169445	660
5 1754140665440434215	1221
6 1872978591788881482	320

	device_id_hash	event_name
7	1989685320445148348	728
8	2305766456715991733	658
9	2768580714092136945	523
10	2910761286178294850	489
11	3198863056321683492	411
12	3337471580007169353	1186
13	3521696259961091617	702
14	3610536745613892312	661
15	3940194724501792579	569
16	4148267947677649217	418
17	4257848154605915902	378
18	4623191541214045580	1768
19	4705890939205361551	385
20	4713748699910253089	824
21	5444091050002420401	319
22	5996739817823449506	585
23	6304868067479728361	2307
24	6613527411922640441	468
25	6932517045703054087	1439
26	7738058666231999878	888
27	8118046800480174342	755
28	8173190940950873857	553
29	8192783670889455257	302
30	8309980273750324949	363
31	8876255867200307343	359

In [42]:
```
data_warning.query('exp_id ==248').pivot_table(
    index='device_id_hash',  columns = 'event_name',
    aggfunc={'event_name':'count'}
).reset_index()
```

Out[42]:

	device_id_hash				
event_name		CartScreenAppear	MainScreenAppear	OffersScreenAppear	Paym
0	674541835027541643	232.0	21.0	28.0	
1	1100007125648169445	311.0	14.0	41.0	
2	2305766456715991733	319.0	41.0	65.0	
3	2910761286178294850	214.0	5.0	64.0	
4	3198863056321683492	182.0	64.0	40.0	

	device_id_hash				
event_name		CartScreenAppear	MainScreenAppear	OffersScreenAppear	Paym
5	3521696259961091617	252.0	121.0	106.0	
6	3610536745613892312	293.0	45.0	94.0	
7	3940194724501792579	256.0	46.0	55.0	
8	4148267947677649217	146.0	41.0	109.0	
9	4713748699910253089	376.0	23.0	59.0	
10	5444091050002420401	110.0	43.0	72.0	
11	5996739817823449506	289.0	1.0	20.0	
12	6304868067479728361	1100.0	46.0	76.0	
13	6932517045703054087	417.0	112.0	509.0	
14	8876255867200307343	22.0	101.0	226.0	

В три экспериментальные группы попали аномальные пользователи, в 246 группу 10 пользователей, в 247 - 7, в 248 - 15. Так же мы видим что, что бы дойти до событий CartScreenAppear и PaymentScreenSuccessful, не обязательно совершать события MainScreenAppear, OffersScreenAppear и Tutorial.

Много ли событий и пользователей вы потеряли, отбросив старые данные?

Найдем количество событий, которые потеряли, отбросив старые данные:

In [43]:
```
event_count_new = data_new.shape[0]
print(event_count_new,event_count)
```

```
240887 243713
```

In [44]:
```
a = event_count-event_count_new
print(
    f'Количество событий в логе стало: {event_count_new}.\
    Абсолютные потери событий: {a}, относительные: {a/event_count:.2%}'
)
```

```
Количество событий в логе стало: 240887.    Абсолютные потери событий: 2826, относит
ельные: 1.16%
```

Найдем количество пользователей, которых потеряли, отбросив старые данные:

In [45]:
```
id_count_new = data_new['device_id_hash'].nunique()
d = id_count - id_count_new
print(
    f'Количество уникальных пользователей в логе стало: {id_count_new}.\
    Абсолютные потери: {d}, относительные: {d/id_count:.2%}'
)
```

```
Количество уникальных пользователей в логе стало: 7534.    Абсолютные потери: 17, от
носительные: 0.23%
```

In [46]:
```
data_new['event_name'].unique()
```

Out[46]:
```
array(['Tutorial', 'MainScreenAppear', 'OffersScreenAppear',
       'CartScreenAppear', 'PaymentScreenSuccessful'], dtype=object)
```

Уникальных событий попрежнему - 5.

Проверьте, что у вас есть пользователи из всех трёх экспериментальных групп.

In [47]:
```
v = data_new.pivot_table(index='exp_id', values='device_id_hash', aggfunc='nunique')
v['perc, %'] =(round(v['device_id_hash'] / v['device_id_hash'].sum(),3))*100
v
```

Out[47]:

	exp_id	device_id_hash	perc, %
0	246	2484	33.0
1	247	2513	33.4
2	248	2537	33.7

В каждой из трех групп есть пользователи.

Подведем итоги:

- Количество событий в логе: 243713, количество уникальных 5: MainScreenAppear, PaymentScreenSuccessful, CartScreenAppear, OffersScreenAppear, Tutorial
- Количество уникальных пользователей в логе: 7551. Пользователей, которые находятся в нескольких группах - нет. Пользователи равномерно распределены по группам(разница 0.4%-0.7%).
- В среднем на пользователя приходится событий: 19.0
- Мы отбросили данные, раньше 01 августа 2019 года: данные за период с 25 июля 2019 года до 01 августа 2019 года не полные. В итоге мы имеем данные за неделю: с 01 по 07 августа 2019 года.

После удаления неполных данных:

- Количество событий в логе стало: 240887. Абсолютные потери событий составили: 2826, относительные: 1.16%
- Количество уникальных пользователей в логе стало: 7534. Абсолютные потери: 17, относительные: 0.23%

Так же стоит изучить поведение аномальных пользователей, может быть у них некорректно работает приложение и они не могут завершить действие, либо запись данных об их действиях некорректна, либо это результат работы вредоносных программ(?).

Изучим воронку событий

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

In [48]:
```
ev = data_new['event_name'].value_counts()
ev=pd.DataFrame(ev)
```

In [49]:
```
ev['perc, %'] = round(ev['event_name']/ev['event_name'].sum(),3)*100
ev
```

Out[49]:

	event_name	perc, %
MainScreenAppear	117328	48.7
OffersScreenAppear	46333	19.2
CartScreenAppear	42303	17.6
PaymentScreenSuccessful	33918	14.1
Tutorial	1005	0.4

Чаще всего встречается событие: MainScreenAppear (48.7% от всех событий), реже всего PaymentScreenSuccessful (14.1%) и Tutorial (всего 0.4% от всех событий).

Посчитаем, сколько пользователей совершали каждое из этих событий.

In [50]:
```
w = data_new.pivot_table(index='event_name', values='device_id_hash', aggfunc='nuniq
w['perc, %'] = round(w['device_id_hash']/id_count_new, 3)*100
sort_w= w.sort_values('device_id_hash',ascending=False)
sort_w
```

Out[50]:

	event_name	device_id_hash	perc, %
1	MainScreenAppear	7419	98.5
2	OffersScreenAppear	4593	61.0
0	CartScreenAppear	3734	49.6
3	PaymentScreenSuccessful	3539	47.0
4	Tutorial	840	11.1

Событие MainScreenAppear совершили хотя бы раз 98.5 % от числа пользователей, наименьшее число пользователей совершили хотя бы раз события PaymentScreenSuccessful (47%) и Tutorial(11.1%)

Предположим, в каком порядке происходят события.

In [51]:
```
sort_w
```

Out[51]:

	event_name	device_id_hash	perc, %
1	MainScreenAppear	7419	98.5
2	OffersScreenAppear	4593	61.0
0	CartScreenAppear	3734	49.6
3	PaymentScreenSuccessful	3539	47.0
4	Tutorial	840	11.1

Из результатов предыдущего пункта можно сделать вывод, что в основном события выстаиваются в следующую цепочку:

MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful

Можно предположить, что:

- MainScreenAppear - это просмотр главной страницы
- OffersScreenAppear - переход на страницу товаров
- CartScreenAppear - переход на страницу «Корзина»
- PaymentScreenSuccessful - переход на страницу успешной оплаты товара
- Tutorial - Посмотрел страницу "Обучение"

Эти события не всегда происходят в таком порядке. Поскольку, например, только 98.5 % заходят на главную страницу, можно предположить, что они попадают сразу на страницу товара из поисковой системы либо страница с товарами/корзиной у них в закладках, но это всего 1.5%. Событие Tutorial не может быть включено в последовательную цепочку, поскольку оно может произойти в независимоти от других событий, предыдущих и последующих.

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

То есть для последовательности событий A → B → C посчитаем отношение числа пользователей с событием B к количеству пользователей с событием A, а также отношение числа пользователей с событием C к количеству пользователей с событием B.

In [52]:
```
data_without_Tut = data_new.query('event_name !="Tutorial"')
```

In [53]:
```
q = data_without_Tut.pivot_table(
    index='event_data', columns = 'event_name',
    values='device_id_hash', aggfunc='nunique'
).reset_index()
q
```

Out[53]:

	event_name	event_data	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenS
0		2019-08-01	1510	3545	2050	
1		2019-08-02	1525	3485	2070	
2		2019-08-03	1505	3301	2072	
3		2019-08-04	1517	3354	2085	
4		2019-08-05	1547	3608	2151	
5		2019-08-06	1528	3766	2167	
6		2019-08-	1434	3482	2062	

event_name	event_data	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenS
	07				

In [ ]:

Расчитаем доли пользователей, переходящих на следующий шаг.

In [54]:
```
conv= pd.DataFrame()
#conv['event_data']=q['event_data']
conv['Offers_Main, %'] = round(q['OffersScreenAppear']/q['MainScreenAppear'], 3)*100
conv['Cart_Offers, %'] = round(q['CartScreenAppear']/q['OffersScreenAppear'], 3)*100
conv['Payment_Cart, %'] = round(q['PaymentScreenSuccessful']/q['CartScreenAppear'],
conv
```

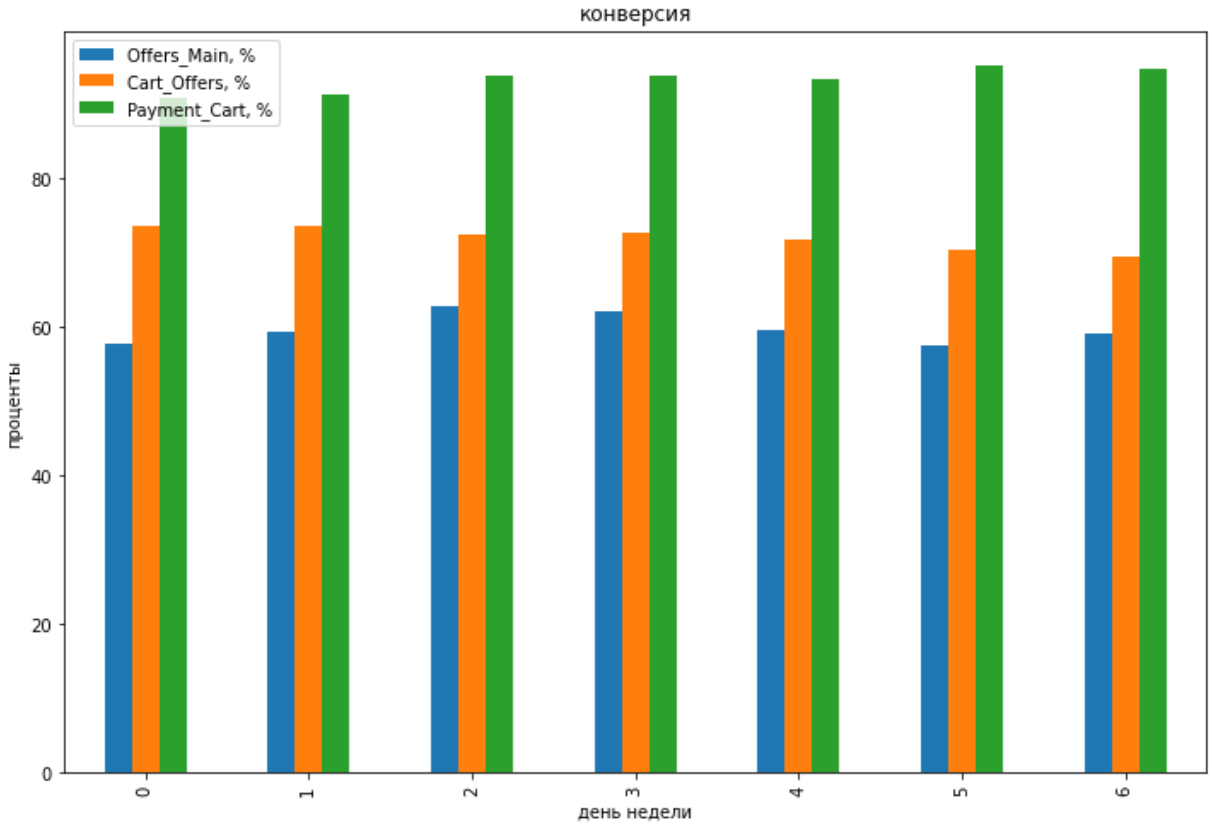
Out[54]:

	Offers_Main, %	Cart_Offers, %	Payment_Cart, %
0	57.8	73.7	90.9
1	59.4	73.7	91.4
2	62.8	72.6	93.9
3	62.2	72.8	93.9
4	59.6	71.9	93.5
5	57.5	70.5	95.2
6	59.2	69.5	94.9

- Offers_Main - переход с главной страницы на страницу товаров
- Cart_Offers - переход со страницы товаров на страницу "Корзина"
- Payment_Cart - переход со страницы "Корзина" к оплате

In [55]:
```
conv.plot(kind='bar', fontsize=10, figsize=(12,8))
plt.xlabel('день недели', fontsize=10)
plt.ylabel('проценты',fontsize=10)
plt.title('конверсия', fontsize=12)
```

Out[55]:
```
Text(0.5, 1.0, 'конверсия')
```



In [56]:
```
row = data_without_Tut.pivot_table(index='event_name', values='device_id_hash', aggf
sort_values('device_id_hash',ascending=False).reset_index()
row
```

Out[56]:

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

In [57]:
```
#количество уникальных пользователей
id_count_new
```

Out[57]: 7534

In [58]:
```
row['next'] = row['device_id_hash'].shift(periods=1)
row.loc[0,'next'] = id_count_new
row['conv, %'] = round(row['device_id_hash']/row['next'], 3)*100
```

In [59]:
```
row
```

Out[59]:

	event_name	device_id_hash	next	conv, %
0	MainScreenAppear	7419	7534.0	98.5
1	OffersScreenAppear	4593	7419.0	61.9
2	CartScreenAppear	3734	4593.0	81.3

	event_name	device_id_hash	next	conv, %
3	PaymentScreenSuccessful	3539	3734.0	94.8

In [60]:
```
#построим базовую диаграмму воронки
data = dict(
    #number=[100, 61.9, 81.3, 94.8],
    number=row['conv, %'],
    stage=['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScr
fig = px.funnel(data, x='number', y='stage', title = 'Воронка событий')
fig.show()
```

Посмотрим на воронку событий по группам

In [61]:
```
v# распределение уникальных пользователей по группам
```

Out[61]:

	exp_id	device_id_hash	perc, %
0	246	2484	33.0
1	247	2513	33.4
2	248	2537	33.7

In [62]:
```
#Группа контрольная 246
row_246 = data_without_Tut.query('exp_id==246').\
```

```
pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique').\
sort_values('device_id_hash',ascending=False).reset_index()
row_246
```

Out[62]:

	event_name	device_id_hash
0	MainScreenAppear	2450
1	OffersScreenAppear	1542
2	CartScreenAppear	1266
3	PaymentScreenSuccessful	1200

In [63]:

```
row_246['next'] = row_246['device_id_hash'].shift(periods=1)
row_246.loc[0,'next'] = v.loc[0,'device_id_hash']
row_246['conv, %'] = round(row_246['device_id_hash']/row_246['next'], 3)*100
row_246
```

Out[63]:

	event_name	device_id_hash	next	conv, %
0	MainScreenAppear	2450	2484.0	98.6
1	OffersScreenAppear	1542	2450.0	62.9
2	CartScreenAppear	1266	1542.0	82.1
3	PaymentScreenSuccessful	1200	1266.0	94.8

In [64]:

```
#Группа контрольная 247
row_247 = data_without_Tut.query('exp_id==247').\
pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique').\
sort_values('device_id_hash',ascending=False).reset_index()
row_247
```

Out[64]:

	event_name	device_id_hash
0	MainScreenAppear	2476
1	OffersScreenAppear	1520
2	CartScreenAppear	1238
3	PaymentScreenSuccessful	1158

In [65]:

```
row_247['next'] = row_247['device_id_hash'].shift(periods=1)
row_247.loc[0,'next'] = v.loc[1,'device_id_hash']
row_247['conv, %'] = round(row_247['device_id_hash']/row_247['next'], 3)*100
row_247
```

Out[65]:

	event_name	device_id_hash	next	conv, %
0	MainScreenAppear	2476	2513.0	98.5
1	OffersScreenAppear	1520	2476.0	61.4
2	CartScreenAppear	1238	1520.0	81.4
3	PaymentScreenSuccessful	1158	1238.0	93.5

In [66]:
```
#Группа экспериментальная 248
row_248 = data_without_Tut.query('exp_id==248').\
pivot_table(index='event_name', values='device_id_hash', aggfunc='nunique').\
sort_values('device_id_hash',ascending=False).reset_index()
row_248
```

Out[66]:

	event_name	device_id_hash
0	MainScreenAppear	2493
1	OffersScreenAppear	1531
2	CartScreenAppear	1230
3	PaymentScreenSuccessful	1181

In [67]:
```
row_248['next'] = row_248['device_id_hash'].shift(periods=1)
row_248.loc[0,'next'] = v.loc[2,'device_id_hash']
row_248['conv_248, %'] = round(row_248['device_id_hash']/row_248['next'], 3)*100
row_248
```

Out[67]:

	event_name	device_id_hash	next	conv_248, %
0	MainScreenAppear	2493	2537.0	98.3
1	OffersScreenAppear	1531	2493.0	61.4
2	CartScreenAppear	1230	1531.0	80.3
3	PaymentScreenSuccessful	1181	1230.0	96.0

In [68]:
```
row_248['conv_247, %'] = row_247['conv, %']
row_248['conv_246, %'] = row_246['conv, %']
row_248
```

Out[68]:

	event_name	device_id_hash	next	conv_248, %	conv_247, %	conv_246, %
0	MainScreenAppear	2493	2537.0	98.3	98.5	98.6
1	OffersScreenAppear	1531	2493.0	61.4	61.4	62.9
2	CartScreenAppear	1230	1531.0	80.3	81.4	82.1
3	PaymentScreenSuccessful	1181	1230.0	96.0	93.5	94.8

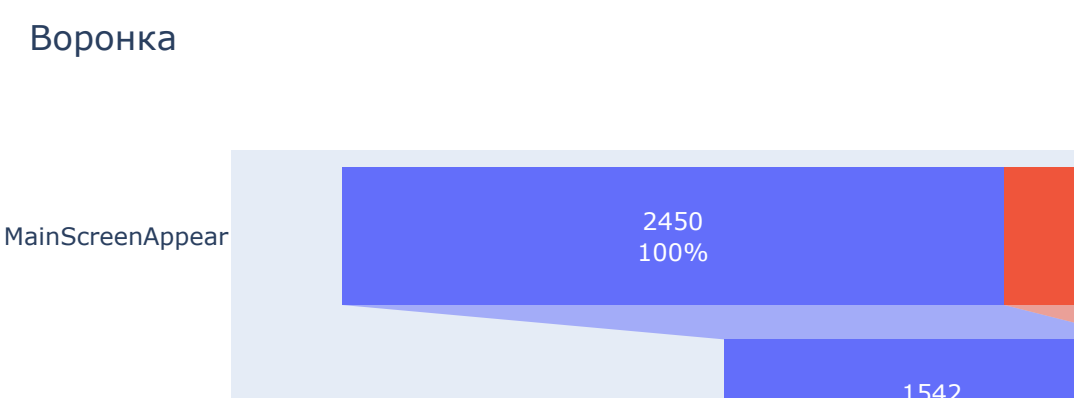
In [69]:
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = '246',
    y = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScree
    x = row_246['device_id_hash'],
    textinfo = "value+percent previous"))

fig.add_trace(go.Funnel(
    name = '247',
    orientation = "h",
    y = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScree
    x = row_247['device_id_hash'],
    textposition = "inside",
    textinfo = "value+percent previous"))
```

```
fig.add_trace(go.Funnel(
    name = '248',
    orientation = "h",
    y = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScree
    x = row_248['device_id_hash'],

    textposition = "outside",
    textinfo = "value+percent previous",))
fig.update_layout(title='Воронка')
fig.show()
```



Конверсия перехода Корзина->Оплата выше у экспериментальной группы на 2.5% и 1.2%, чем в контрольных, а вот конверсия перехода с Главной страницы и Каталога товаров на следующий шаг, равна либо ниже, чем в контрольных группах. Например переход из каталога товаров в корзину в контрольных группах 81.4% и 82.1%, а в экспериментальной 80.3%.

Выясним, на каком шаге теряем больше всего пользователей

In [70]:
```
conv
```

Out[70]:

	Offers_Main, %	Cart_Offers, %	Payment_Cart, %
0	57.8	73.7	90.9
1	59.4	73.7	91.4

	Offers_Main, %	Cart_Offers, %	Payment_Cart, %
2	62.8	72.6	93.9
3	62.2	72.8	93.9
4	59.6	71.9	93.5
5	57.5	70.5	95.2
6	59.2	69.5	94.9

In [71]:
```
row_248
```

Out[71]:

	event_name	device_id_hash	next	conv_248, %	conv_247, %	conv_246, %
0	MainScreenAppear	2493	2537.0	98.3	98.5	98.6
1	OffersScreenAppear	1531	2493.0	61.4	61.4	62.9
2	CartScreenAppear	1230	1531.0	80.3	81.4	82.1
3	PaymentScreenSuccessful	1181	1230.0	96.0	93.5	94.8

Из полученных данные можно сделать вывод, что больше всего пользователей теряем на шаге: переход с главной страницы на страницу товаров. Потеряли за неделю от 42.5% до 37.2% пользователей(в выходные дни потеря пользователей ниже на 3-5%). Либо если смотреть на показатели в целом, без разбивки по дням, переходят только 61.4%-62.9%, т.е. потеря 37.1% - 38.6% пользователей.

Выясним какая доля пользователей доходит от первого события до оплаты

In [72]:
```
q = data_without_Tut.pivot_table(
    index='event_data', columns = 'event_name',
    values='device_id_hash', aggfunc='nunique'
).reset_index()
q
```

Out[72]:

event_name	event_data	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenS
0	2019-08-01	1510	3545	2050	
1	2019-08-02	1525	3485	2070	
2	2019-08-03	1505	3301	2072	
3	2019-08-04	1517	3354	2085	
4	2019-08-05	1547	3608	2151	
5	2019-08-06	1528	3766	2167	

	event_name	event_data	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenS
	6	2019-08-07	1434	3482	2062	

In [73]:
```
conv_pay = round(q['PaymentScreenSuccessful']/q['MainScreenAppear'], 3)*100
conv_pay
```

Out[73]:
```
0    38.7
1    40.0
2    42.8
3    42.5
4    40.1
5    38.6
6    39.1
dtype: float64
```

In [74]:
```
row
```

Out[74]:

	event_name	device_id_hash	next	conv, %
0	MainScreenAppear	7419	7534.0	98.5
1	OffersScreenAppear	4593	7419.0	61.9
2	CartScreenAppear	3734	4593.0	81.3
3	PaymentScreenSuccessful	3539	3734.0	94.8

In [75]:
```
row.loc[3,'device_id_hash']/row.loc[0,'device_id_hash']
```

Out[75]:
```
0.47701846610055265
```

In [76]:
```
conv_pay_all = round(row.loc[3,'device_id_hash']/row.loc[0,'device_id_hash'], 2)*100
conv_pay_all
```

Out[76]:
```
48.0
```

In [77]:
```
data_without_Tut_248 = data_without_Tut.query('exp_id==248')
q248 = data_without_Tut_248.pivot_table(
    index='event_data', columns = 'event_name',
    values='device_id_hash', aggfunc='nunique'
).reset_index()

conv_pay248 = round(q248['PaymentScreenSuccessful']/q248['MainScreenAppear'], 3)*100
conv_pay248
```

Out[77]:
```
0    39.4
1    40.0
2    42.9
3    41.5
4    39.6
5    39.0
6    37.4
dtype: float64
```

In [78]:
```
data_without_Tut.head()
```

Out[78]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
2829	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	246	2019-08-01
2830	MainScreenAppear	3737462046622621720	2019-08-01 00:08:55	246	2019-08-01
2831	OffersScreenAppear	3737462046622621720	2019-08-01 00:08:58	246	2019-08-01
2832	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	247	2019-08-01
2833	MainScreenAppear	4899590676214355127	2019-08-01 00:10:15	247	2019-08-01

In [79]:

```
print(
    f'От первого события до оплаты товара доходит от {conv_pay248.min()}% до {conv_p
)
```

```
От первого события до оплаты товара доходит от 37.4% до 42.9% пользователей
```

Так же видим, что есть рост количество оплат в выходные дни. Если смотреть без разбивки на дни, процент равен 48.0. Такая разница из-за того, что есть много постоянных пользователей, которые оплачивают товар, не переходя на главную страницу, возможно в закладках у них каталог товаров, либо корзина сразу.

Подведем итоги:

Чаще всего в логе встречается событие: MainScreenAppear (48.7% от всех событий), реже всего PaymentScreenSuccessful (14.1%) и Tutorial (всего 0.4% от всех событий). Событие MainScreenAppear совершили хотя бы раз 98.5 % от числа пользователей, наименьшее число пользователей совершили хотя бы раз события PaymentScreenSuccessful (47%) и Tutorial(11.1%) В основном события выстаиваются в следующую цепочку:

MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful.

Можно предположить, что:

- MainScreenAppear - это просмотр главной страницы
- OffersScreenAppear - переход на страницу товаров
- CartScreenAppear - переход на страницу «Корзина»
- PaymentScreenSuccessful - переход на страницу успешной оплаты товара
- Tutorial - Посмотрел страницу "Обучение" (событие не входит в последовательную цепочку)

Конверсия перехода Корзина->Оплата выше у экспериментальной группы на 2.5% и 1.2%, чем в контрольных, а вот конверсия перехода с Главной страницы и Каталога товаров на следующий шаг, равна либо ниже, чем в контрольных группах. Например переход из каталога товаров в корзину в контрольных группах 81.4% и 82.1%, а в экспериментальной 80.3%.

Так же мы выяснили, что больше всего пользователей теряем на шаге: переход с главной страницы на страницу товаров. Потеряли за неделю от 42.5% до 37.2% пользователей. Либо если смотреть на показатели в целом, без разбивки по дням, переходят только 61.4%-62.9%, т.е. потеря 37.1% - 38.6% пользователей.

От первого события до оплаты товара доходит от 37.4% до 42.9% пользователей, в зависимости от дня недели. В общем за период - 48.0%.

Изучите результаты эксперимента

Сколько пользователей в каждой экспериментальной группе

Воспользуемся расчетом, который сделали ранее. Выведем сразу результат.

In [80]:
```
v
```

Out[80]:

	exp_id	device_id_hash	perc, %
0	246	2484	33.0
1	247	2513	33.4
2	248	2537	33.7

В контрольной группе 246 - 2484 уникальных пользователя, в контрольной группе 247 - 2513 уникальных пользователей, в экспериментальной группе - 2537, что на 0.7% и на 0.3% больше, чем в группах 246 и 247 соответственно. Контрольные группы различаются в 0.4% - это не критическая разница. Для успешного A/A теста, различия в количестве не должна превышеть 1%.

У нас 2 контрольные группы для A/A-эксперимента, чтобы проверить корректность всех механизмов и расчётов. Проверим, находят ли статистические критерии разницу между выборками 246 и 247.

Проверим, есть ли разница между выборками 246 и 247. Применим proportions_ztest - эта функция используется для проверки пропорций в выборках.

In [81]:
```
data_ztest = data_without_Tut.query('exp_id!=248')
h = data_ztest.pivot_table(index=['exp_id', 'event_name'], values='device_id_hash',
sort_values(by=['device_id_hash', 'exp_id'],ascending=False).reset_index()
h
```

Out[81]:

	exp_id	event_name	device_id_hash
0	247	MainScreenAppear	2476
1	246	MainScreenAppear	2450
2	246	OffersScreenAppear	1542
3	247	OffersScreenAppear	1520
4	246	CartScreenAppear	1266
5	247	CartScreenAppear	1238
6	246	PaymentScreenSuccessful	1200
7	247	PaymentScreenSuccessful	1158

In [82]:
```
#сохраним в список количество удач, т.е. совершения покупок в каждой группе
count_list = h.loc[6:7,'device_id_hash']
count_list
```

Out[82]:
```
6    1200
7    1158
Name: device_id_hash, dtype: int64
```

In [83]:
```
h2 = data_ztest.pivot_table(index='exp_id', values='device_id_hash', aggfunc='nuniqu
sort_values('exp_id',ascending=True).reset_index()
h2
```

Out[83]:

	exp_id	device_id_hash
0	246	2483
1	247	2512

In [84]:
```
#сохраним в список
nobs_list = h2['device_id_hash']
nobs_list
```

Out[84]:
```
0    2483
1    2512
Name: device_id_hash, dtype: int64
```

- Нулевая гипотеза: различий между выборками нет
- Альтернативная: разница есть

In [85]:
```
# В count передадим количество совершения события оплаты в двух группах,
#в nobs число пользователей в группах
import numpy as np
from statsmodels.stats.proportion import proportions_ztest


stat, p_value = proportions_ztest(count_list, nobs_list, alternative='two-sided')
print('{0:0.3f}'.format(p_value))

if p_value < 0.05:
        print('Отвергаем нулевую гипотезу: различий между выборками нет')
else:
        print(        'Не получилось отвергнуть нулевую гипотезу, нет оснований счит
        print()
```

```
0.114
Не получилось отвергнуть нулевую гипотезу, нет оснований считать выборки разными
```

Тут все в порядке

Выберем самое популярное событие. Посчитаем число пользователей, совершивших это событие в каждой из контрольных групп. Посчитаем долю пользователей, совершивших это событие.

Воспользуемся расчетами из предыдущих пунктов:

In [86]:
```
# Распределение событий:
ev
```

Out[86]:

	event_name	perc, %
MainScreenAppear	117328	48.7
OffersScreenAppear	46333	19.2
CartScreenAppear	42303	17.6
PaymentScreenSuccessful	33918	14.1
Tutorial	1005	0.4

Самое популярное событие MainScreenAppear (просмотр главной страницы), на его долю приходится 48.7% от всего числа событий в логе

In [87]:
```
# количество пользователей, совершавших событие MainScreenAppear
r = data_new.pivot_table(index='exp_id', columns = 'event_name', values='device_id_h
reset_index()
r[['exp_id','MainScreenAppear']]
```

Out[87]:

event_name	exp_id	MainScreenAppear
0	246	2450
1	247	2476
2	248	2493

In [88]:
```
#распределение уникальных пользователей по группам
v
```

Out[88]:

	exp_id	device_id_hash	perc, %
0	246	2484	33.0
1	247	2513	33.4
2	248	2537	33.7

In [89]:
```
row_248 # воронка событий
```

Out[89]:

	event_name	device_id_hash	next	conv_248, %	conv_247, %	conv_246, %
0	MainScreenAppear	2493	2537.0	98.3	98.5	98.6
1	OffersScreenAppear	1531	2493.0	61.4	61.4	62.9
2	CartScreenAppear	1230	1531.0	80.3	81.4	82.1
3	PaymentScreenSuccessful	1181	1230.0	96.0	93.5	94.8

Доля пользователей совершивших событие MainScreenAppear в группе 248 - 98.3%, в группе 247 - 98.5%, в 246 - 98.6 %

Проверим, будет ли отличие между группами статистически достоверным для самого популярного события и так же для остальных событий.

Для того,что проверить есть ли статистически значимая разница в выборках, воспользуемся Z-тестом. Z-тест решает задачу сравнения пропорций двух генеральных совокупностей по выборкам из них, может использоваться для больших выборок.

В нашем случае пропоции - это конверсия, выборки - экспериментальные группы. Так же необходимо учесть, что у нас 4 события, т.е. тест будет проведен 4 раза, для каждой пары групп. Для каждого события считаем конверсию в каждой группе. По мере увеличения количества статистических тестов вероятность совершения ошибки первого рода хотя бы в одном из тестов быстро возрастает, поэтому воспользуемся поправкой Бонферрони(хотя, забегая вперед, поправка не нужна, вероятность мы получаем большую во всем тестах), которая корректирует уровень альфа (α) таким образом, чтобы можно было контролировать вероятность совершения ошибки I рода. Уровень значимости при проверке статистических гипотез выберем 0.05, поскольку при большом объеме выборки случайные отклонения компенсируют друг друга, и получить значимое различие в выборках при их однородности менее вероятно.

In [90]:
```
# выполнение z-теста выведем в функцию, чтобы не считать для каждого события отдельн
def ztest(data_1, data_2, event_name, alpha, n): # alpha уровень статистической знач
                                                 #экспериментов n, data_1, data_2 -
                                #
    user_id = np.array([data_1['device_id_hash'].nunique(),data_2['device_id_hash'].

    # число пользователей, совершивших событие event_name
    successful_event = np.array([data_1[data_1['event_name'] == event_name]['device_
                                 data_2[data_2['event_name'] == event_name]['device_
                               ])

    # пропорция успехов в первой группе:
    p1 = successful_event[0]/user_id[0]
    # пропорция успехов во второй группе:
    p2 = successful_event[1]/user_id[1]

    # пропорция совершения события в комбинированном датасете:
    p_combined = (successful_event[0] + successful_event[1]) / (user_id[0] + user_id

    # разница пропорций в датасетах
    difference = p1 - p2
    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference /  np.sqrt(p_combined * (1 - p_combined) * (1/user_id[0] +
    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)
    n_alpha = alpha/n
    p_value = (1 - distr.cdf(abs(z_value))) * 2
    print('Действие пользователя:', event_name)
    print('p-value: ', '{0:.2}'.format(p_value))
    if p_value < n_alpha:
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')
    else:
        print(        'Не получилось отвергнуть нулевую гипотезу, нет оснований счит
        print()
```

- Нулевая гипотеза: различий в пропорциях двух выборок 246 и 247 нет

- Альтернативная: различия есть

In [91]:
```
# делаем поправку alpha/16
data_without_Tut = data_new.query('event_name !="Tutorial"')
for event_name in data_without_Tut['event_name'].unique():
    ztest(data_without_Tut[data_without_Tut['exp_id']==246], \
          data_without_Tut[data_without_Tut['exp_id']==247], \
          event_name, 0.05, 16
         )
```

```
Действие пользователя: MainScreenAppear
p-value:  0.75
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: OffersScreenAppear
p-value:  0.25
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: CartScreenAppear
p-value:  0.23
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: PaymentScreenSuccessful
p-value:  0.11
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Можем сделать вывод, что контрольные группы подобраны корректно. Различий в конверсии от события к событию нет.

Аналогично поступим с группой с изменённым шрифтом.

Сравним результаты с каждой из контрольных групп в отдельности по каждому событию.

In [92]:
```
# делаем поправку alpha/16
for event_name in data_without_Tut['event_name'].unique():
    ztest(data_without_Tut[data_without_Tut['exp_id']==246], \
          data_without_Tut[data_without_Tut['exp_id']==248], \
          event_name, 0.05, 16
         )
```

```
Действие пользователя: MainScreenAppear
p-value:  0.34
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: OffersScreenAppear
p-value:  0.21
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: CartScreenAppear
p-value:  0.081
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: PaymentScreenSuccessful
p-value:  0.22
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

In [93]:
```
# делаем поправку alpha/16
for event_name in data_without_Tut['event_name'].unique():
    ztest(data_without_Tut[data_without_Tut['exp_id']==247], \
          data_without_Tut[data_without_Tut['exp_id']==248], \
```

```
            event_name, 0.05, 16
        )
```

```
Действие пользователя: MainScreenAppear
p-value:  0.52
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: OffersScreenAppear
p-value:  0.93
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: CartScreenAppear
p-value:  0.59
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: PaymentScreenSuccessful
p-value:  0.73
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

In [94]:
```
data_without_Tut[data_without_Tut['exp_id']==247].append(data_without_Tut[data_witho
```

Out[94]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
2832	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	247	2019-08-01
2833	MainScreenAppear	4899590676214355127	2019-08-01 00:10:15	247	2019-08-01
2838	MainScreenAppear	4899590676214355127	2019-08-01 00:11:28	247	2019-08-01
2839	OffersScreenAppear	4899590676214355127	2019-08-01 00:11:30	247	2019-08-01
2843	OffersScreenAppear	4899590676214355127	2019-08-01 00:12:36	247	2019-08-01

Сравним результаты с объединённой контрольной группой.

In [95]:
```
# объединим два датафрейма
data_246_247 = data_without_Tut[data_without_Tut['exp_id']==247].\
append(data_without_Tut[data_without_Tut['exp_id']==246])
data_246_247.head()
```

Out[95]:

	event_name	device_id_hash	event_timestamp	exp_id	event_data
2832	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	247	2019-08-01
2833	MainScreenAppear	4899590676214355127	2019-08-01 00:10:15	247	2019-08-01
2838	MainScreenAppear	4899590676214355127	2019-08-01 00:11:28	247	2019-08-01
2839	OffersScreenAppear	4899590676214355127	2019-08-01 00:11:30	247	2019-08-01
2843	OffersScreenAppear	4899590676214355127	2019-08-01 00:12:36	247	2019-08-01

In [96]:
```
# делаем поправку alpha/16
data_without_Tut = data_new.query('event_name !="Tutorial"')
for event_name in data_without_Tut['event_name'].unique():
    ztest(data_246_247, data_without_Tut[data_without_Tut['exp_id']==248], \
          event_name, 0.05, 16
         )
```

```
Действие пользователя: MainScreenAppear
p-value:  0.35
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

```
Действие пользователя: OffersScreenAppear
p-value:  0.45
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: CartScreenAppear
p-value:  0.19
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Действие пользователя: PaymentScreenSuccessful
p-value:  0.61
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Ни в одном из тестов у нас не получилось отвергнуть гипотезу о том, что разницы между группами нет. Все различия между группами не имеют статистичекой значимости.

Вывод:

В нашем распоряжении был лог с действиями пользователей (события). Нам нужно было разобраться, как ведут себя пользователи мобильного приложения.

Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми. Нужно было выяснить, какой шрифт лучше.

После изучения данных, выяснили, что количество событий в логе: 243713, количество уникальных событий 5: MainScreenAppear, PaymentScreenSuccessful, CartScreenAppear, OffersScreenAppear, Tutorial Количество уникальных пользователей в логе: 7551. Пользователей, которые находятся в нескольких группах - нет. Их распределение равномерно по группам. В среднем на пользователя приходится событий: 20.

Так же мы отбросили данные, раньше 01 августа 2019 года, поскольку данные за период с 25 июля 2019 года до 01 августа 2019 года были не полные. В итоге мы распологали данными за неделю: с 01 по 07 августа 2019 года. При это мы потеряли абсолютные потери событий составили: 2826, относительные: 1.16%. Абсолютные потери количество пользователй: 17, относительные: 0.23%. Так же выявили пользователей с аномальным поведением - у 32 пользователей за неделю более 300 событий. Стоит изучить причины такого поведения, поскольку данные пользователи могут искажать результаты анализа. Может быть у них некорректно работает приложение и они не могут завершить действие, либо запись данных об их действиях некорректна, либо это результат работы вредоносных программ(?).

Чаще всего в логе встречается событие: MainScreenAppear - 117328 раз (составляет 48.7% от всех событий), реже всего PaymentScreenSuccessful - 33918 раз (14.1%) и Tutorial - 1005 раз (всего 0.4% от всех событий). Событие MainScreenAppear совершили хотя бы раз 98.5 % от числа пользователей, наименьшее число пользователей совершили хотя бы раз события PaymentScreenSuccessful (47%) и Tutorial(11.1%). Из этого можем сделать вывод, что в основном события выстаиваются в следующую цепочку:

MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful.

Можно предположить, что:

- MainScreenAppear - это просмотр главной страницы

- OffersScreenAppear - переход на страницу товаров
- CartScreenAppear - переход на страницу «Корзина»
- PaymentScreenSuccessful - переход на страницу успешной оплаты товара
- Tutorial - Посмотрел страницу "Обучение" Событие Tutorial не входит в последовательную цепочку выше, поскольку его наступление не зависит от других событий. Однако При детальном изучении, выяснили, что событие MainScreenAppear и OffersScreenAppear не является обязательным для оплаты товара.

При изучении воронки событий, выяснили, что конверсия перехода Корзина->Оплата выше у экспериментальной группы на 2.5% и 1.2%, чем в контрольных, а вот конверсия перехода с Главной страницы и Каталога товаров на следующий шаг, равна либо ниже, чем в контрольных группах. Например переход из каталога товаров в корзину в контрольных группах 81.4% и 82.1%, а в экспериментальной 80.3%.

Так же мы выяснили, что больше всего пользователей теряем на шаге: переход с главной страницы на страницу товаров. Потеряли за неделю от 42.5% до 37.2% пользователей. Либо если смотреть на показатели в целом, без разбивки по дням, переходят только 61.4%-62.9%, т.е. потеря 37.1% - 38.6% пользователей. От первого события до оплаты товара доходит от 37.4% до 42.9% пользователей, в зависимости от дня недели. В общем за период - 48.0%. Меньше всего теряем пользователей на шаге Корзина-> оплата 5.2%, а при переходе с Каталога товаров в Корзину теряем 28.7%.

Результаты эксперимента: У нас 3 группы: 246 и 247 - контрольные(для A/A теста) и 248 - эксперименталья, которая видела приложение с новым шрифтом. Пользователи по группам распределены равномерно. В контрольной группе 246 - 2484 уникальных пользователя, в контрольной группе 247 - 2513 уникальных пользователей, в экспериментальной группе - 2537, что на 0.7% и на 0.3% больше, чем в группах 246 и 247 соответственно. Контрольные группы различаются в 0.4% - это не критическая разница. Для успешного A/A теста, различия в количестве не должна превышеть 1%. При проведении A/A теста, выяснили, что контрольные группы подобраны корректно. Далее провели тесты на выявление статистически значемых различий между группами при наступлении каждого из событий. Различия между контрольными группами, между каждой контрольной и экспериментальной и между объединенной контрольной и экспериментальной. Ни один из тесто не обнаружил статистически значимой разницы между группами.
Таким образом, по расчету воронки событий и после проведения 16 тестов на выявление статистически значемых различий между группами можем сделать вывод, что изменения шрифта в приложение практически не отразилось на поведении пользователей. Если данное изменение не требует материальных затрат, его можно внести. Если же требует, то введение нового шрифта не целесообразно.