Я очистил вывод переред отправкой работы на проверку, тк файлы больше 30 мб Практикум не принимает...

Описание проекта

Вы работаете в стартапе, который продаёт продукты питания. Нужно разобраться, как ведут себя пользователи вашего мобильного приложения.

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

После этого исследуйте результаты A/A/B-эксперимента. Дизайнеры захотели поменять шрифты во всём приложении, а менеджеры испугались, что пользователям будет непривычно. Договорились принять решение по результатам A/A/B-теста. Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми. Выясните, какой шрифт лучше.

Описание данных Каждая запись в логе — это действие пользователя, или событие:

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

Также сразу набросаю план работы:

- Посмотрим на данные
- Выполним необходимую предобработку
- Более подробно изучим данные
- Изучим воронку продаж
- Изучим результаты теста
- Сделаем общие выводы

Знакомство с данными

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

import warnings
warnings.filterwarnings('ignore')
```

```
t", "{:.2f}".format)
```

In [3]:
```
try:
    df = pd.read_csv('logs_exp.csv', sep='\t')
except:
    df = pd.read_csv('https://code.s3.yandex.net/datasets/logs_exp.csv', sep='\t')
```

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

Out[4]:

	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

Замените названия столбцов на удобные для вас;

In [5]:
```
df.columns = ['event_name', 'device_id_hash', 'event_timestamp', 'exp_id']
```

In [6]:
```
df.head()
```

Out[6]:

	event_name	device_id_hash	event_timestamp	exp_id
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

Проверьте пропуски и типы данных. Откорректируйте, если нужно

In [7]:
```
# Используем функцию для ознакомления с данными
def df_info(data):
    print(data.info(), '\n\n')
    print('Кол-во пропусков\n', data.isna().sum(), '\n\n')
    print('Кол-во дубликатов\n', data.duplicated().sum(), '\n\n')
    print('Стат\n', data.describe())
df_info(df)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   event_name       244126 non-null  object
 1   device_id_hash   244126 non-null  int64 
 2   event_timestamp  244126 non-null  int64 
 3   exp_id           244126 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 7.5+ MB
None


Кол-во пропусков
 event_name         0
device_id_hash     0
event_timestamp    0
exp_id             0
dtype: int64


Кол-во дубликатов
 413


Стат
               device_id_hash  event_timestamp    exp_id
count              244126.00        244126.00 244126.00
mean  4627568124590853120.00    1564913915.84    247.02
std   2642424998963707904.00        177134.32      0.82
min      6888746892508752.00    1564029816.00    246.00
25%   2372212476992240640.00    1564756580.25    246.00
50%   4623191541214045184.00    1564919395.00    247.00
75%   6932517045703054336.00    1565074511.00    248.00
max   9222603179720523776.00    1565212517.00    248.00
```

Всего 244126 записей в датасете. 4 колонки. Пропусков нет.

В данных есть 413 дубликатов. Поскольку это менее 1%, смело избавимся от дубликатов.

In [8]:
```
df = df.drop_duplicates(keep='last')
```

In [9]:
```
df.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
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```


ремени, а также отдельный столбец

дат;

In [11]:
```
#добавим колонку с датой и временем
df['event_datetime'] = pd.to_datetime(df['event_timestamp'], unit='s')
```

In [12]:
```
#добавим колонку с датой
df['event_date'] = df['event_datetime'].dt.date
```

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

Out[13]:

	event_name	device_id_hash	event_timestamp	exp_id	event_datetime	event_
0	MainScreenAppear	4575588528974610257	1564029816	246	2019-07-25 04:43:36	2019
1	MainScreenAppear	7416695313311560658	1564053102	246	2019-07-25 11:11:42	2019
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019
3	CartScreenAppear	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-07-25 11:48:42	2019

На данном этапе работы мы ознакомились с предоставленными данными. Всего в датасете 244126 строк в 4 колонках. После этого мы переименовали колонки для удобства, убрали 413 дубликата и добавили колонки с датой и временем и отдельно с датой. Пропусков в данных нет. Данные готовы для дальнейшего анализа.

Шаг 3. Изучите и проверьте данные

Сколько всего событий в логе?

In [14]:
```
print('Всего', len(df['event_name'].unique()), 'уникальных событий:', df['event_name
```

```
Всего 5 уникальных событий: ['MainScreenAppear' 'PaymentScreenSuccessful' 'CartScre
enAppear'
 'OffersScreenAppear' 'Tutorial']
```

Сколько всего пользователей в логе?

In [15]:
```
print('Уникальных пользователей:', df['device_id_hash'].nunique())
```

```
Уникальных пользователей: 7551
```

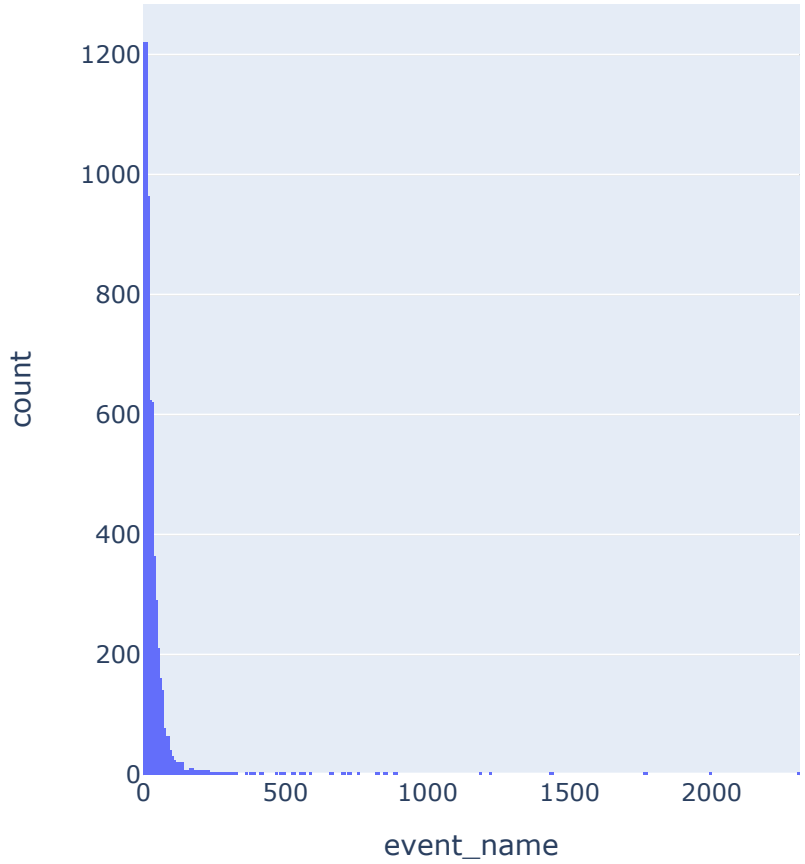
Сколько в среднем событий приходится на пользователя?

In [16]:
```
df.groupby('device_id_hash', as_index=False)['event_name'].count().mean()
```

Out[16]:
```
device_id_hash   4677318727102452736.00
event_name                       32.28
dtype: float64
```

In [17]:
```
fig = px.histogram((
    df
    .groupby('device_id_hash', as_index=False)
```

```
    .agg({'event_name': 'count'})
    .sort_values('event_name', ascending=False)
), x="event_name")
fig.show()
```



In [18]:
```
(
    df
    .groupby('device_id_hash', as_index=False)
    .agg({'event_name': 'count'})
    .sort_values('event_name', ascending=False)
    .describe()
)
```

Out[18]:

	device_id_hash	event_name
count	7551.00	7551.00
mean	4677318727102437376.00	32.28
std	2655343100552018944.00	65.15
min	6888746892508752.00	1.00
25%	2397700422051031552.00	9.00
50%	4688021588771745792.00	20.00
75%	7007352523282521088.00	37.00
max	9222603179720523776.00	2307.00

Среднее количество событий на одного пользователя - 32. Также я построил гистрограмму по количесвту событий на одного пользователя и вывел на экран

результат метода describe. По гистограмме можно заметить, что большинсто значений находится в диапозоне от 1 до 100 дейсвтий. Метод describe также указывает на то, что в выбросах есть выводы, тк есть пользователи с более чем 2000 действиями. С выбросами мы разберемся позже

В среднем на каждого пользователя приходится по 32 события

Данными за какой период вы располагаете? Найдите максимальную и минимальную дату. Постройте гистограмму по дате и времени. Можно ли быть уверенным, что у вас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать» события из прошлого — это может «перекашивать данные». Определите, с какого момента данные полные и отбросьте более старые. Данными за какой период времени вы располагаете на самом деле?

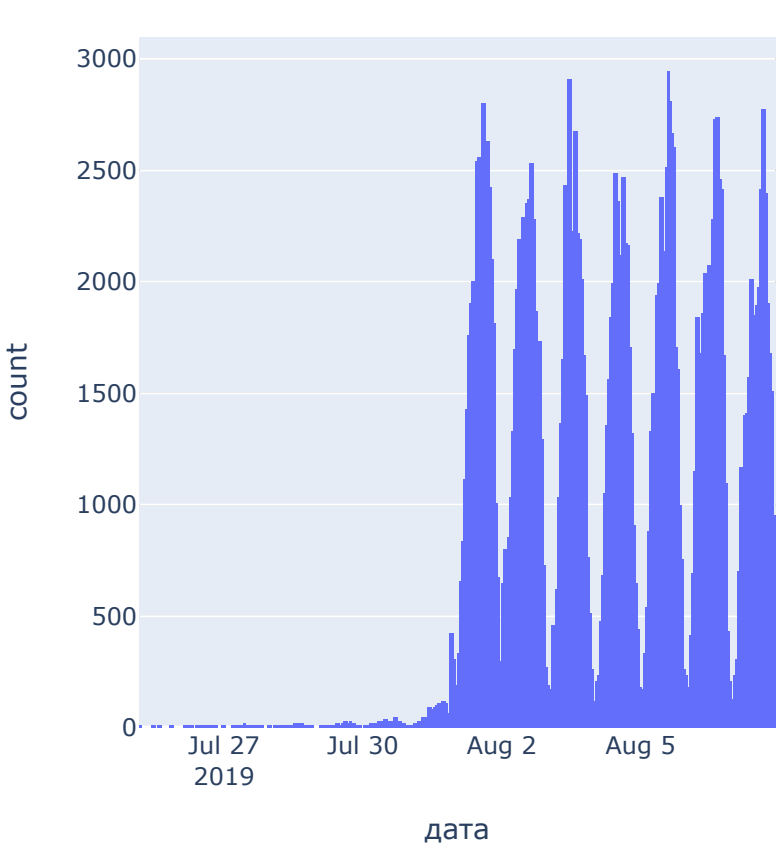
In [19]:
```
df['event_datetime'].describe(datetime_is_numeric=True)
```

Out[19]:
```
count                           243713
mean     2019-08-04 10:19:17.987665920
min                2019-07-25 04:43:36
25%                2019-08-02 14:36:45
50%                2019-08-04 11:51:00
75%                2019-08-06 06:56:24
max                2019-08-07 21:15:17
Name: event_datetime, dtype: object
```

В нашем датасете прдеставленны данные за период с 25 июля по 7 августа 2019 года.

Построим гистограмму по дате и времени и посмотрим на распределение

In [20]:
```
fig_1 = px.histogram(df,
                     x='event_datetime',
                      title='Кол-во событий по дням',
                      labels={'event_datetime':'дата'}
                     )



fig_1.show()
```



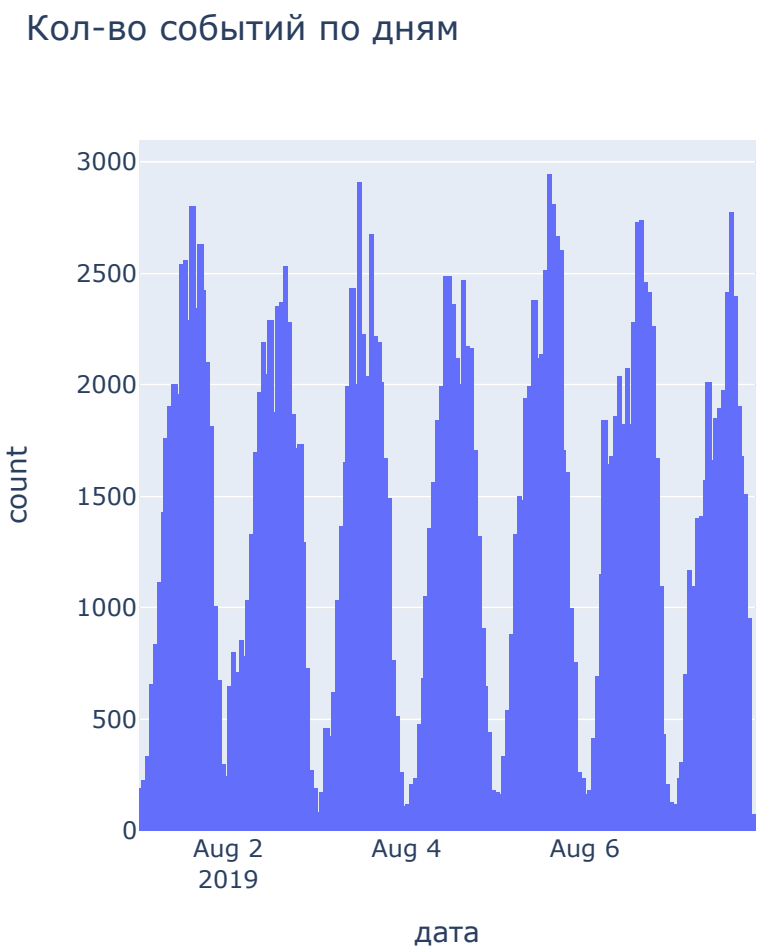
Как мы видим, в дейсвтительности полные данные доступны лишь с 1 по 7 августа включительно. Отбросим более ранние события, чтобы они не искажали наши выводы

In [21]:
```
df = df[df['event_datetime'] >= '2019-08-01']
```

In [22]:
```
df['event_datetime'].describe(datetime_is_numeric=True)
```

Out[22]:
```
count                           240887
mean     2019-08-04 11:31:47.713620736
min                2019-08-01 00:07:28
25%          2019-08-02 15:26:52.500000
50%                2019-08-04 12:28:00
75%                2019-08-06 07:21:18
max                2019-08-07 21:15:17
Name: event_datetime, dtype: object
```

In [23]:
```
fig_2 = px.histogram(df,
                     x='event_datetime',
                      title='Кол-во событий по дням',
                      labels={'event_datetime':'дата'}
                     )



fig_2.show()
```



Теперь данные полные. Всего осталось 240887 записей в датасете, то есть мы убрали около 4к записей (это менее 2%)

In [24]:
```
df['device_id_hash'].nunique()
```

Out[24]:
```
7534
```

Мы потеряли всего 17 пользователей. В общей массе это ерунда.

In [25]:
```
df['event_name'].count()
```

Out[25]:
```
240887
```

In [26]:
```
lost_users = 1 - (7534/7551)
print(f'Потеряно пользователей {lost_users:5f}')
lost_events = 1 - (240887/243713)
print(f'Потеряно событий {lost_events:5f}')
```

```
Потеряно пользователей 0.002251
Потеряно событий 0.011596
```

В данных осталось 7534 пользователей из 7551 и 240887 событий из 243713 (после удаления дубликатов). Итого мы потеряли 17 пользователей (0.2%) и 2826 событий (1.1%)

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

In [27]:
```
df.groupby('exp_id', as_index=False)['device_id_hash'].nunique()
```

Out[27]:

	exp_id	device_id_hash
0	246	2484
1	247	2513
2	248	2537

В данном шаге мы более подробно познакомились с данными: Посмотрели на то, какие события есть в логах, сколько пользователей представлено, сколько в среднем событий приходится на каждого пользователя. Также мы построили гистограмму и обнаружили явные следы выбросов с данных. После этого мы определили период, за который у нас есть именно полные данные и отбросили остальные данные. После этого мы оценили в абсолютных и относительных величиных количество потерянных пользователей и событий.
После этого мы проверили наличие в каждой из эксперементальных группах пользователей. Получилось что сть пользователи из всех групп эксперимента. Группы 246 и 247 практически одинаковы по численности. В группе 248 незначительно, но большей пользователей. С количеством будем разбираться далле, когда будем анализировать результаты теста

Шаг 4. Изучите воронку событий

Посмотрите, какие события есть в логах, как часто они встречаются. Отсортируйте события по частоте.

In [28]:
```
print('Всего', len(df['event_name'].unique()), 'уникальных событий:', df['event_
```

```
Всего 5 уникальных событий: ['Tutorial' 'MainScreenAppear' 'OffersScreenAppear'
'CartScreenAppear'
 'PaymentScreenSuccessful']
```

In [29]:
```
df['event_name'].value_counts()
```

Out[29]:
```
MainScreenAppear           117328
OffersScreenAppear          46333
CartScreenAppear            42303
PaymentScreenSuccessful     33918
Tutorial                     1005
Name: event_name, dtype: int64
```

Самое частое событие - появление главного экрана. Далее идут по уменьшению: экран фофера, корзина, успешная оплата. Tutorial(обучение) открывали лишь 1005 раз.

Посчитайте, сколько пользователей совершали каждое из этих событий. Отсортируйте события по числу пользователей. Посчитайте долю пользователей, которые хоть раз совершали событие.

In [30]:
```
(
    df.groupby('event_name', as_index=False)['device_id_hash'].nunique()
    .assign(ratio=lambda x: round(x['device_id_hash'] / df['device_id_hash'].nu
    .sort_values('device_id_hash', ascending=False)
)
```

Out[30]:

	event_name	device_id_hash	ratio
1	MainScreenAppear	7419	98.47
2	OffersScreenAppear	4593	60.96
0	CartScreenAppear	3734	49.56
3	PaymentScreenSuccessful	3539	46.97
4	Tutorial	840	11.15

Очистим данные

Посмотрим на количество операций для каждого пользователя

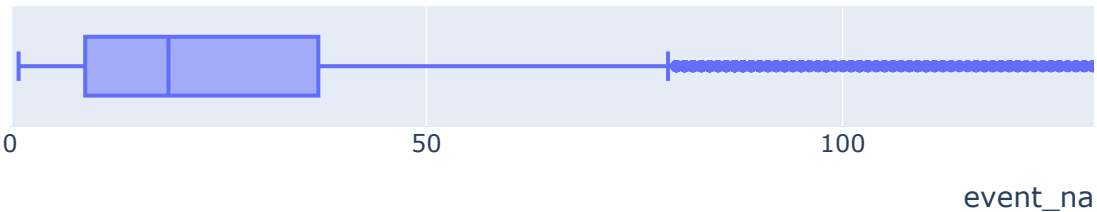
In [31]:
```
event_by_user = df.groupby('device_id_hash', as_index=False)['event_name'].cour
event_by_user
```

Out[31]:

	device_id_hash	event_name
5107	6304868067479728361	2307
146	197027893265565660	1996
3707	4623191541214045580	1768
5580	6932517045703054087	1439
1390	1754140665440434215	1221
...	...	...
3330	4182608234194850691	1
3293	4129178667624237055	1
833	1068167424061681266	1
3270	4096007885698937181	1
0	6888746892508752	1

7534 rows × 2 columns

In [32]:
```
fig_3 = px.box(event_by_user, x="event_name", width=1200, height=200)
fig_3.update_xaxes(range=[0, 250])
fig_3.show()
```



1000+ дейсвтий за неделю. Это очень много. Посмотрим на процентили

In [33]:
```
np.percentile(event_by_user['event_name'], [90, 95, 99])
```

Out[33]:
```
array([ 64.  ,  88.  , 201.01])
```

Предлагаю отсечь пользователей по 99 процентилю. (убрать пользователей, которые совершили более 201 действия)

In [34]:
```
event_by_user[event_by_user['event_name'] > 201]['device_id_hash']
```

Out[34]:
```
5107    6304868067479728361
146      197027893265565660
3707    4623191541214045580
5580    6932517045703054087
1390    1754140665440434215
               ...
5408    6707886513123955772
4182    5178884875886952056
1215    1553654098241439838
1930    2457989834692826118
5034    6220847999332178356
Name: device_id_hash, Length: 76, dtype: int64
```

In [35]:
```
df = df[~(df['device_id_hash'].isin(event_by_user[event_by_user['event_name'] >
```

От выбросов избавились. Проверим количесвто событий, которые мы потеряли в ходе очистки данных. Напомню, что до этого шага в наших логах оставалось 240887 событий

In [36]:
```
print('Потеряно событий:', (240887 - df['event_name'].count()) / 240887)
```

```
Потеряно событий: 0.14227417834918446
```

Потеряно 14% событий. В целом, это достаточно пограничный показатель, но в данном случае я все же принимаю решение избавиться от таких выбросов.

Предположите, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.

> События происходят в следующем порядке: главный экран, просмотр предложений, корзина, успешная оплата. Обучение проходят далеко не все и это событие не стоит включать в цепочку

По воронке событий посчитайте, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитайте 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.

Построим воронку продаж

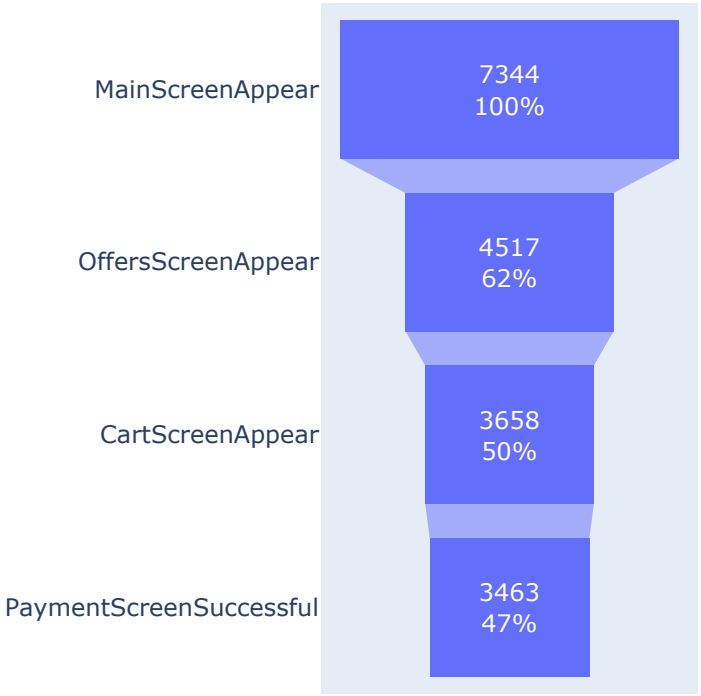
In [37]:
```
# Для начала сделаем сводник
funnel = (
    df.groupby('event_name', as_index=False)['device_id_hash'].nunique()
    .sort_values('device_id_hash', ascending=False)
#     считаем процент от предыдущего шага
    .assign(ratio=lambda x: round((x['device_id_hash'] /  x['device_id_hash']
    .reset_index(drop=True)
    .fillna(100)
#     Убираем дейсвтие Tutorial
    .loc[lambda x: x['event_name'] != 'Tutorial']
)


funnel
```

Out[37]:

	event_name	device_id_hash	ratio
0	MainScreenAppear	7344	100.00
1	OffersScreenAppear	4517	61.51
2	CartScreenAppear	3658	80.98
3	PaymentScreenSuccessful	3463	94.67

In [38]:
```
# Теперь визуализируем воронку
fig_4 = go.Figure(
    go.Funnel(
        y=funnel['event_name'],
        x=funnel['device_id_hash'],
        textposition = "inside",
        textinfo = "value+percent initial",
    )
)

fig_4.update_layout(
    title_text='Общая воронка событий'
)

fig_4.show()
```

Общая воронка событий



Воронка событий построена. По ней можно сделать некоторые короткие выводы:

- Больше половины пользователей переходят из главного экрана в экран с предложениями
- Удивительно большой процент людей переходит из экрана предложений в корзину
- Почти 95% пользователей видят экран с успешной оплатой после корзины (как минимум, можно утверждать что процесс покупки работает корректно)

На каком шаге теряете больше всего пользователей?

Большинство пользователей теряется при переходе с первого на второй шаг. Почти 40% или 2827 пользователей.

Какая доля пользователей доходит от первого события до оплаты?

47% пользователей от общего количества доходят до успешной оплаты.

Это очень высокий показатель для сервиса. Почти каждый 2й пользователь успешно совершает покупку

Шаг 5. Изучите результаты эксперимента

Сколько пользователей в каждой экспериментальной группе?

In [39]:
```
df.groupby('exp_id', as_index=False)['device_id_hash'].nunique()
```

Out[39]:

	exp_id	device_id_hash
0	246	2456
1	247	2491
2	248	2511

Общее количество пользователей в каждой эксперементальной группе примерно одинаково. Разница незначительна

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

Чтобы ответить на этот вопрос, сравним конверсии двух этих выборок. Для начала создадим сводник с разбивкой по группам

In [40]:
```
# Создадим сводник
funnel_exp = (
    df.pivot_table(index='exp_id', columns='event_name' ,values='device_id
)

# Нужный порядок колонок
funnel_exp = funnel_exp[['MainScreenAppear', 'OffersScreenAppear', 'CartSc

# Перевернем для удобства
funnel_exp = funnel_exp.transpose()

# Избавимся от мультииндекса
funnel_exp.reset_index(inplace=True)

# Добавим итоговое кол-во пользователей по каждой группе
total_users = {'event_name':'total_users',
           246:df[df['exp_id'] == 246]['device_id_hash'].nunique(),
           247:df[df['exp_id'] == 247]['device_id_hash'].nunique(),
           248:df[df['exp_id'] == 248]['device_id_hash'].nunique()}

funnel_exp = pd.concat([funnel_exp, pd.DataFrame([total_users])], ignore_i


funnel_exp
```

Out[40]:

	event_name	246	247	248
0	MainScreenAppear	2423	2454	2467
1	OffersScreenAppear	1514	1498	1505
2	CartScreenAppear	1238	1216	1204
3	PaymentScreenSuccessful	1172	1136	1155
4	total_users	2456	2491	2511

Будем сравнивать общую конверсию (от главного экрана до экрана с успешной оплатой)

Для 246й группы: Из 2423 посетивших сайт, оплатили заказ 1172

Для 247й группы: Из 2454 посетивших сайт, оплатили заказ 1136

Воспользуюсь уже готовым кусочком кода из тренажера

In [41]:
```
alpha = .05 # критический уровень статистической значимости

successes = np.array([1172, 1136])
trials = np.array([2423, 2454])

# пропорция успехов в 246й группе:
p1 = successes[0]/trials[0]

# пропорция успехов в 247й 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
```

In [42]:
```
# считаем статистику в ст.отклонениях стандартного нормального распределен
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[

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

In [43]:
```
  # считаем статистику в ст.отклонениях стандартного нормального распредел
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('p-значение: ', p_value)

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

```
p-значение:  0.14615434945299954
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли раз
ными
```

> За нулевую гипотезу брали "Между группами нет разницы", за альтернативную "Между группами есть разница"

В данном случае, конверсии - настоящие пропорции в сравниваемых генеральных совокупностях. Соответсвенно, сравнив их, можно судить о разнице или ее отсутствии между выборками 246 и 246

Дополню гипотезы

H0 - Между коверсиями выборок 246 и 247 нет статистической разницы

H1 - Между коверсиями выборок 246 и 247 есть статистическая разница

Выберите самое популярное событие. Посчитайте число пользователей, совершивших это событие в каждой из контрольных групп. Посчитайте долю пользователей, совершивших это событие. Проверьте, будет ли отличие между группами статистически достоверным. Проделайте то же самое для всех других событий (удобно обернуть проверку в отдельную функцию). Можно ли сказать, что разбиение на группы работает корректно?

In [44]:
```
(
    df.query('event_name == "MainScreenAppear"')
    .groupby('exp_id', as_index=False)['device_id_hash'].nunique()
    .rename(columns={'device_id_hash': 'users_on_main_screen'})
    .assign(total_users =
            df.groupby('exp_id', as_index=False)['device_id_hash'].nuniq
    .assign(ratio=lambda x: round((x['users_on_main_screen'] /  x['total_
)
```

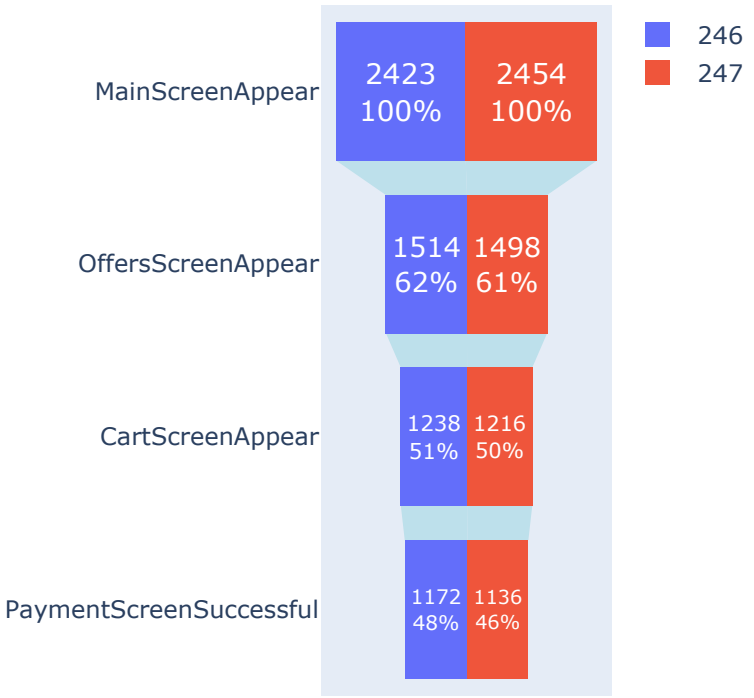
Out[44]:

	exp_id	users_on_main_screen	total_users	ratio
0	246	2423	2456	98.66
1	247	2454	2491	98.51
2	248	2467	2511	98.25

In [45]:
```
# Добавим колонку с общими данными для А-групп
funnel_exp['246+247'] = funnel_exp[246] + funnel_exp[247]

def z_value_diff(group_1, group_2, alpha, funnel):
    for i in funnel_exp.index[:-1]:
        alpha = alpha
#         Количество участников в группах
        n_user1 = funnel_exp[group_1][4]
        n_user2 = funnel_exp[group_2][4]
#         Пропорция успеха в первой группе
        p1 = funnel_exp[group_1][i] / n_user1
        # пропорция успехов во второй группе:
        p2 = funnel_exp[group_2][i] / n_user2
        # пропорция успехов в комбинированном датасете:
        p_combined = ((funnel_exp[group_1][i] + funnel_exp[group_2][i]) /
```

```
                    (n_user1 + n_user2))
        # разница пропорций в датасетах
        difference = p1 - p2
        # считаем статистику в ст.отклонениях стандартного нормального ра
        z_value = difference / mth.sqrt(p_combined * (1 - p_combined) *
                                        (1/n_user1 + 1/n_user2))
        # задаем стандартное нормальное распределение (среднее 0, ст.откл
        distr = st.norm(0, 1)
        p_value = (1 - distr.cdf(abs(z_value))) * 2
        print('{} p-значение: {}'.format(funnel_exp['event_name'][i], p_v
        if (p_value < alpha):
            print("Отвергаем нулевую гипотезу: между долями есть значима
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет основа
        print('')

# Если funnel = 1, то выводим график воронки
    if funnel == 1:
        fig = go.Figure()
        for i, group in enumerate([group_1, group_2]):
            fig.add_trace(go.Funnel(
                name = str(group),
                y = funnel_exp[funnel_exp['event_name'] != 'total_users'
                x = funnel_exp[group],
                textposition = "inside",
                textinfo = "value+percent initial",
        #         marker = {"color": colors[i+color]},
                connector = {"fillcolor": '#bde0eb'},
                insidetextfont = {'color': 'white', 'size': 14}))

        fig.update_layout(
            title_text='Воронка событий для исследуемых групп'
        )
        fig.show()

z_value_diff(246,247,0.05, 1)
```

```
MainScreenAppear p-значение: 0.6730951807208485
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли ра
зными

OffersScreenAppear p-значение: 0.27702771700769824
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли ра
зными

CartScreenAppear p-значение: 0.26299626637167783
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли ра
зными

PaymentScreenSuccessful p-значение: 0.13586258367273985
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли ра
зными
```

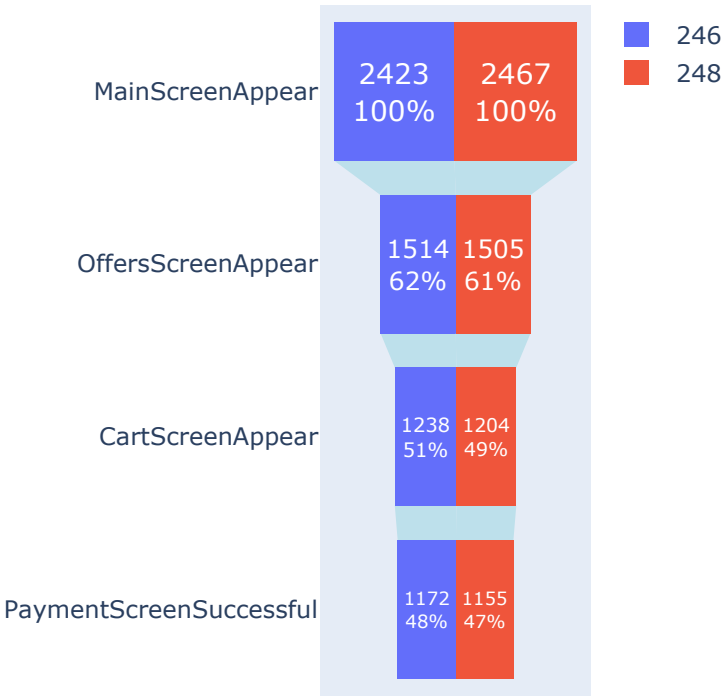


При уровне значимости 5% ни по одному действию статистических различий между группами 246 и 247 нет

Аналогично поступите с группой с изменённым шрифтом. Сравните результаты с каждой из контрольных групп в отдельности по каждому событию. Сравните результаты с объединённой контрольной группой. Какие выводы из эксперимента можно сделать?

In [46]:
```
# Сравниваем 246 и 248
z_value_diff(246,248,0.05, 1)
```

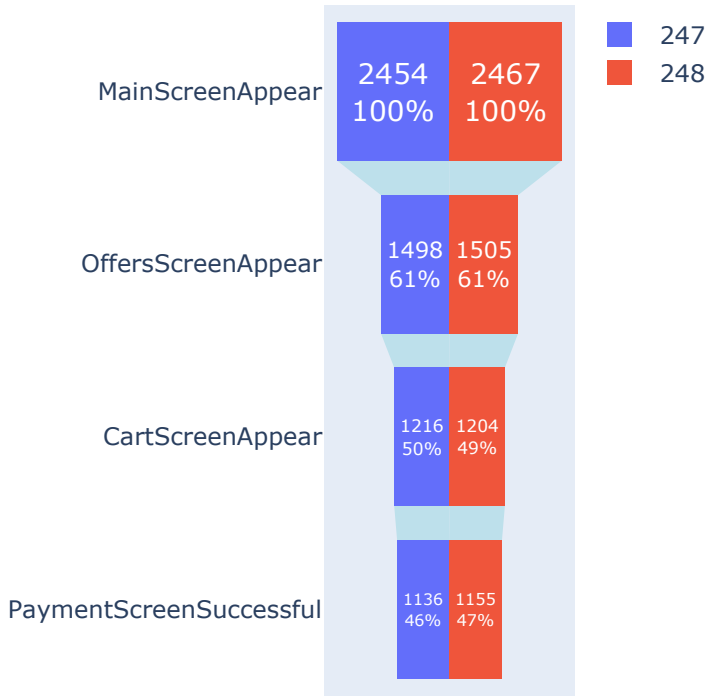
```
MainScreenAppear p-значение: 0.24380010740478975
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли р
азными

OffersScreenAppear p-значение: 0.21751867407994285
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли р
азными

CartScreenAppear p-значение: 0.08317408116828573
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли р
азными

PaymentScreenSuccessful p-значение: 0.2239322340682175
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли р
азными
```



При уровне значимости 5% ни по одному действию статистических различий между группами 246 и 248 нет

In [47]:
```
# Сравниваем 247 и 248
z_value_diff(247,248,0.05, 1)
```

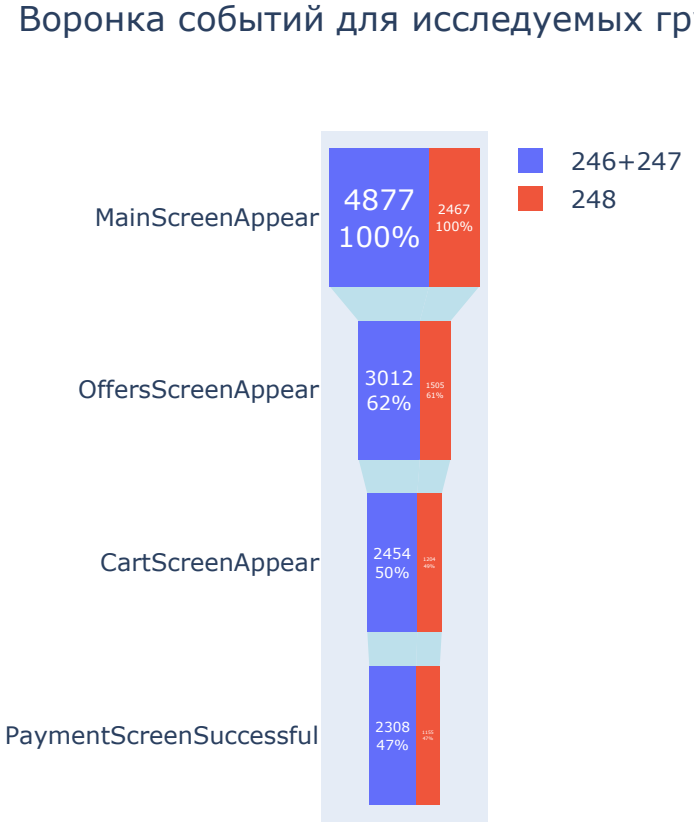
```
MainScreenAppear p-значение: 0.4545336230756303
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.8850746269495819
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.539679592997024
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.7800603242138475
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



При уровне значимости 5% ни по одному действию статистических различий между группами 247 и 248 нет

In [48]:
```
# Сравниваем 246+247 и 248
z_value_diff('246+247',248,0.05, 1)
```

```
MainScreenAppear p-значение: 0.26186402859555846
Не получилось отвергнуть нулевую гипотезу, нет оснований считать дол
и разными

OffersScreenAppear p-значение: 0.42801834712141273
Не получилось отвергнуть нулевую гипотезу, нет оснований считать дол
и разными

CartScreenAppear p-значение: 0.17619153820927536
Не получилось отвергнуть нулевую гипотезу, нет оснований считать дол
и разными

PaymentScreenSuccessful p-значение: 0.5908710902173984
Не получилось отвергнуть нулевую гипотезу, нет оснований считать дол
и разными
```



При уровне значимости 5% ни по одному действию статистических различий между группами 246+247 и 248 нет

Какой уровень значимости вы выбрали при проверке статистических гипотез выше? Посчитайте, сколько проверок статистических гипотез вы сделали. При уровне значимости 0.1 каждый десятый раз можно получать ложный результат. Какой уровень значимости стоит применить? Если вы хотите изменить его, проделайте предыдущие пункты и проверьте свои выводы.

Можно применить поправку Бонферрони, Идея метода заключается в том, чтобы снизить вероятность ошибки первого рода.

Чтобы применить поправку Бонферрони нужны допустимый уровень значимости разделить на кол-во групп и сравнений. В данном случае мы сравнивали 4 группы (и по 4 сравнения в каждой)

В нашем случае, допустимый уровень значимости будет равен 0.003
Запустим сравнение еще раз с новым уровнем значимости и посмотрим на результаты. Графики воронок тут строить не будем

```
# ...нитывая поправку Бонферрони)
```

```
z_value_diff(246,247,0.003, 0)
```

```
MainScreenAppear p-значение: 0.6730951807208485
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

OffersScreenAppear p-значение: 0.27702771700769824
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

CartScreenAppear p-значение: 0.26299626637167783
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

PaymentScreenSuccessful p-значение: 0.13586258367273985
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными
```

In [50]:
```
# Сравниваем 246 и 248 (учитывая поправку Бонферрони)
z_value_diff(246,248,0.003, 0)
```

```
MainScreenAppear p-значение: 0.24380010740478975
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

OffersScreenAppear p-значение: 0.21751867407994285
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

CartScreenAppear p-значение: 0.08317408116828573
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

PaymentScreenSuccessful p-значение: 0.2239322340682175
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными
```

In [51]:
```
# Сравниваем 247 и 248 (учитывая поправку Бонферрони)
z_value_diff(247,248,0.003, 0)
```

```
MainScreenAppear p-значение: 0.4545336230756303
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

OffersScreenAppear p-значение: 0.8850746269495819
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

CartScreenAppear p-значение: 0.539679592997024
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

PaymentScreenSuccessful p-значение: 0.7800603242138475
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными
```

In [52]:
```
# Сравниваем 246+247 и 248 (учитывая поправку Бонферрони)
z_value_diff('246+247',248,0.003, 0)
```

```
MainScreenAppear p-значение: 0.26186402859555846
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

OffersScreenAppear p-значение: 0.42801834712141273
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

CartScreenAppear p-значение: 0.17619153820927536
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными

PaymentScreenSuccessful p-значение: 0.5908710902173984
Не получилось отвергнуть нулевую гипотезу, нет оснований считать до
ли разными
```

Немного объясню про выбранный уровень значимости и про количетсво сравнений. В данном случае мы имеем 4 сравниваемые групы:

- 246 группа
- 247 групаа
- 246+247 группа
- 248 группа.

В каждой из этих груп мы сравниваем 4 конверсии (по каждому шагу воронки). Итого мы имеем 16 сравнений (4 группы по 4 сравнения). Изначально мы брали допустимый уровень значимости равный 5%. Но, как известно, при множественных сравнениях риск получить ложноположительный результат возрастает.

Есть несколько способов уменьшить риск возникновения такого результата: метод Холма, метод Бенджамини–Хозберга, поправка Бонферрони и тд.

В данном случае я использую поправку Бонферрони, это достаточно простой способ контроля над уровнем групповой ошибки. Ее цель, что и логично, заключается в уменьшении риска получения ложноположительного резульатата. Суть ее заключается в том, что необходимо разделить изначальный уровень значимости на кол-во проводимых сравнений. В нашем случае это 0.05 / 16. Результатом и будет уровень значимости, который стоит принять. Выше я заново сранил группы, но уже с уровнем значимости равным 0.003.

Результаты сравнений остались прежними и после применения поправки Бонферрони.

Выводы

Больше половины пользователей переходят из главного экрана в экран
но большой процент людей переходит из

экрана предложений в корзину Почти 95% пользователей видят экран с успешной оплатой после корзины (как минимум, можно утверждать что процесс покупки работает корректно)

Очень интересный проект. В ходе данной работы мы познакомились с данными, предобработали их, немного глубже изучили данные, изучили воронку событий и изучили резльтаты A/A/B-теста

Общий вывод можно сделать следующий:
Конверсия по воронке событий достаточно высокая. Больше половины пользователей переходят с главного экрана на экран с предложениями. Из тех, кто перешл на экран с предложениями, практически никто не отваливается, а переходят в корзину и далее на этап оплаты.

Почти каждый второй пользователь, который видит главный экран, успешно завершает покупку

Новый шрифт в нашем сервисе никак не сказался на покупательской способности пользователей. Конверсия событий практически одинакова для всех групп
