Анализ пользовательского поведения в мобильном приложении

Описание проекта

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

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

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

Создание двух групп A вместо одной имеет определённые преимущества. Если две контрольные группы окажутся равны, мы можем быть уверены в точности проведенного тестирования. Если же между значениями A и A будут существенные различия, это поможет обнаружить факторы, которые привели к искажению результатов. Сравнение контрольных групп также помогает понять, сколько времени и данных потребуется для дальнейших тестов.

В случае общей аналитики и A/A/B-эксперимента работаем с одними и теми же данными. В реальных проектах всегда идут эксперименты. Аналитики исследуют качество работы приложения по общим данным, не учитывая принадлежность пользователей к экспериментам.

Оглавление

Описание данных и задачи

[к Оглавлению](#0.0)

Таблица **logs_exp**:

- **EventName** — название события
- **DeviceIDHash** — уникальный индфикатор пользователя
- **EventTimestamp** — время события
- **ExpId** — номер эксперимента: 246 и 247 - контрольные группы, а 248 - экспериментаьлная

Задачи

Шаг 1. Загрузить данные и подготовить их к анализу

- Замените названия столбцов на удобные для нас;
- Проверить пропуски и типы данных. Откорректировать, если нужно;
- Добавить столбец даты и времени, а также отдельный столбец дат;

Шаг 2. Проверка данных

- Сколько всего событий в логе?
- Сколько всего пользователей в логе?
- Сколько в среднем событий приходится на пользователя?
- Данными за какой период мы располагаем? Найдем максимальную и минимальную дату. Построем гистограмму по дате и времени. Можно ли быть уверенным, что у нас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать» события из прошлого — это может «перекашивать данные». Определим, с какого момента данные полные и отбросим более старые. Данными за какой период времени мы располагаем на самом деле?
- Много ли событий и пользователей мы потеряли, отбросив старые данные?
- Проверим, что у нас есть пользователи из всех трёх экспериментальных групп.

Шаг 3. Воронка событий

- Посмотрим, какие события есть в логах, как часто они встречаются. Отсортируем события по частоте.
- Посчитаем, сколько пользователей совершали каждое из этих событий. Отсортируем события по числу пользователей. Посчитаем долю пользователей, которые хоть раз

совершали событие.
- Предположим, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.
- 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.
- На каком шаге теряем больше всего пользователей?
- Какая доля пользователей доходит от первого события до оплаты?

Шаг 4. Результаты эксперимента

- Сколько пользователей в каждой экспериментальной группе?
- Есть 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.
- Выберем самое популярное событие. Посчитаем число пользователей, совершивших это событие в каждой из контрольных групп. Посчитаем долю пользователей, совершивших это событие. Проверим, будет ли отличие между группами статистически достоверным. Проделаем то же самое для всех других событий (удобно обернуть проверку в отдельную функцию). Можно ли сказать, что разбиение на группы работает корректно?
- Аналогично поступим с группой с изменённым шрифтом. Сравним результаты с каждой из контрольных групп в отдельности по каждому событию. Сравним результаты с объединённой контрольной группой. Какие выводы из эксперимента можно сделать?
- Какой уровень значимости мы выбрали при проверке статистических гипотез выше? Посчитаем, сколько проверок статистических гипотез мы сделали. При уровне значимости 0.1 каждый десятый раз можно получать ложный результат. Какой уровень значимости стоит применить? Если мы хотим изменить его, проделаем предыдущие пункты и проверим свои выводы.

Шаг 5. Вывод

1. Загрузка данных и подготовка их к анализу

[к Оглавлению](#0.0)

In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from IPython.display import display
import numpy as np
import math as mth
from scipy import stats as st

import plotly.express as px
from plotly import graph_objects as go
```

In [2]:

```
#!pip install plotly==4.10.0
```

In [3]:

```
df = pd.read_csv('logs_exp.csv', delimiter='\t')
```

Изучим общую информацию о данных. Посмотрим таблицу. Изучим описание данных для числовых колонок.
Убедимся, что тип данных в каждой колонке — правильный, а также отсутствуют пропущенные значения и дубликаты. При необходимости обработем их.

In [4]:

```
display(df.head(), df.info(), df.describe())
```

```
<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
```

	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

```
None
```

	DeviceIDHash	EventTimestamp	ExpId
count	2.441260e+05	2.441260e+05	244126.000000
mean	4.627568e+18	1.564914e+09	247.022296
std	2.642425e+18	1.771343e+05	0.824434
min	6.888747e+15	1.564030e+09	246.000000
25%	2.372212e+18	1.564757e+09	246.000000
50%	4.623192e+18	1.564919e+09	247.000000
75%	6.932517e+18	1.565075e+09	248.000000
max	9.222603e+18	1.565213e+09	248.000000

Посмотрим есть ли пропущенные значения в наших данных

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
EventName          0
DeviceIDHash       0
EventTimestamp     0
ExpId              0
dtype: int64
```

Посмотри наличие дубликатов

In [6]:

```
df.duplicated().sum()
```

Out[6]:

```
413
```

In [7]:

```
df[df.duplicated()].head()
```

Out[7]:

	EventName	DeviceIDHash	EventTimestamp	ExpId
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

Удалим дубликаты

In [8]:

```
df = df.drop_duplicates().reset_index(drop=True)
```

Изменим названия столбцов

In [9]:

```
df.columns = ['event_name', 'user_id', 'event_timestamp', 'exp_id']
```

Изучим названия событий и посчитаем количество событий

In [10]:

```
df['event_name'].value_counts()
```

Out[10]:

```
MainScreenAppear           119101
OffersScreenAppear          46808
CartScreenAppear            42668
PaymentScreenSuccessful     34118
Tutorial                     1018
Name: event_name, dtype: int64
```

MainScreenAppear

In [11]:

```
df_event = df.groupby('event_name').agg({'event_name': ['count']}).reset_i
ndex()
df_event.columns = ('event_name', "count")
df_event = df_event.sort_values(by='count', ascending=False)
```

In [12]:

```
fig = go.Figure(go.Funnel(
    y = df_event['event_name'],
    x = df_event['count']
    ))
fig.update_layout(title_text='Воронка событий', yaxis_title="События")
fig.show()
```

Изменим тип столбца `event_timestamp` на дату и время и добавим отдельный столбец даты

In [13]:

```
df['date_time'] = pd.to_datetime(df['event_timestamp'], unit='s')
df['date'] = df['date_time'].astype('datetime64[D]')
df.head()
```

Out[13]:

	event_name	user_id	event_timestamp	exp_id	date_ti
0	MainScreenAppear	4575588528974610257	1564029816	246	2019-04:43
1	MainScreenAppear	7416695313311560658	1564053102	246	2019-11:11
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248	2019-11:28
3	CartScreenAppear	3518123091307005509	1564054127	248	2019-11:28
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-11:48

Вывод:

- Мы получили данные состоящие из 244тыс строк. В данных есть дубликаты и не правильные типы данных.
- Проверили данные на пропуски, их нет.
- Проверили данные на наличие дубликатов и удалили их.
- Изменили название столбцов на более удобно читаемые и записываемые.
- Изучили события и посчитали их количество
- Поменяли тип данных столбца `event_timestamp` на тип `pd.to_datetime` и добавили два столбца с датой и датой и временем события.

2. Проверка данных

2.1. Количество событий и пользователей в логе

[к Оглавлению](#0.0)

In [14]:

```
print('Всего событий %d, типов событий %d' % (df.shape[0], df['event_name'
].nunique()))
```

```
Всего событий 243713, типов событий 5
```

In [15]:

```
print('Всего пользователей в логе %d' % (df['user_id'].nunique()))
```

```
Всего пользователей в логе 7551
```

2.2. Количество событий на пользователя в среднем

[к Оглавлению](#0.0)

In [16]:

```
print('В среднем на пользователя приходится %d события' % (df['user_id'].v
alue_counts().mean()))
print('По медиане на пользователя приходится %d событий' % (df['user_id'].
value_counts().median()))
print('По моде на пользователя приходится %d событий' % (df['user_id'].val
ue_counts().mode()))
```

```
В среднем на пользователя приходится 32 события
По медиане на пользователя приходится 20 событий
По моде на пользователя приходится 5 событий
```

В данных много выбросов, это показывает разница в среднем и по медиане. Чаще всего встречаются пользователи с 5 событиями.

2.3. Временные данные

[к Оглавлению](#0.0)

Изучим столбец с датой. Найдем минимальную и максимальную дату.

In [17]:

```
df['date'].value_counts()
```

Out[17]:

```
2019-08-01    36141
2019-08-05    36058
2019-08-06    35788
2019-08-02    35554
2019-08-03    33282
2019-08-04    32968
2019-08-07    31096
```

```
2019-07-31     2030
2019-07-30      412
2019-07-29      184
2019-07-28      105
2019-07-27       55
2019-07-26       31
2019-07-25        9
Name: date, dtype: int64
```

Первая дата 2019-07-25. Видимо первый пробный запуск ресурса. Основная дата начала сбора информации 2019-08-01. Последняя дата через неделю 2019-08-07

Сгруппируем данные и построим график по дате и количестве событий в эти даты.

In [18]:

```
df_date = df.groupby('date').agg({'date': ['count']}).reset_index()
df_date.columns = ('date', "count")
df_date = df_date.sort_values(by='date', ascending=False)
fig = go.Figure(data=[go.Bar(x=df_date['date'], y=df_date['count'], text=d
f_date['count'],
                             textposition='auto')])
fig.update_layout(barmode='group', title_text='Гистограмма по дате',
                  xaxis_title="День", yaxis_title="Количество событий",)
fig.show()
```

Найдем данные о дате и времени и построим график

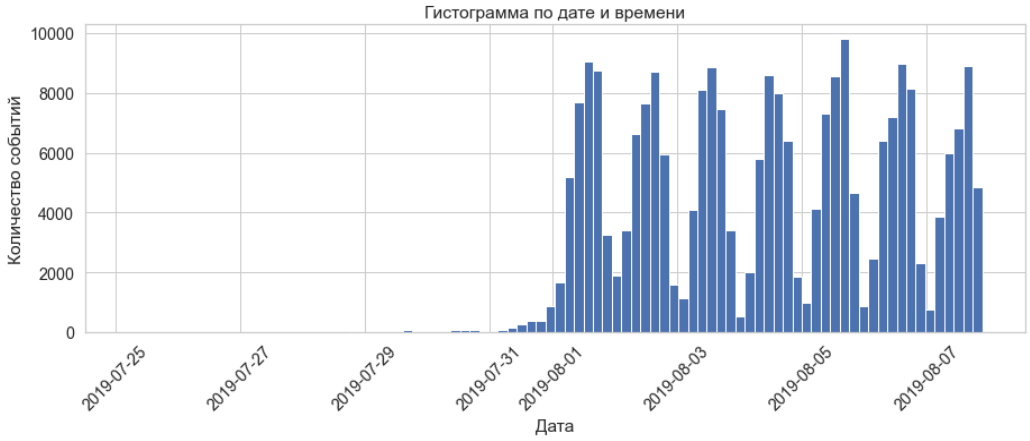
In [19]:

```
df_date_time = df.groupby('date_time').agg({'user_id': ['count']}).reset_i
ndex()
df_date_time.columns = ('date_time', "count")
```

In [20]:

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

sns.set(font_scale=1.3)
sns.set_style("whitegrid")
plt.figure(figsize=(15,5))
ax = df['date_time'].hist(bins=90)
plt.title('Гистограмма по дате и времени')
plt.ylabel("Количество событий")
plt.xlabel("Дата")
plt.xticks(rotation=45)
plt.show()
```



Мы видим, что наибольшее количество событий происходит в будний день. Выходные имеют спад. 2019-08-07 последний день сбора информации и самое малое количество событий в этот день произошло. Возможно это связано с тем, что данные не успели записаться за этот день т.к. сбор их прекратился. У нас может нехватать 10-15% данных.
Мы имеем данные только за 7 дней.

2.4. Чистка данных

[к Оглавлению](#0.0)

Видно, что основные события начинаются с 2019-08-01 и происходят вплоть до 2019-08-07. Сделаем срез по этим датам и изучим их

In [21]:

```
df_iul = df.query('date < "2019-08-01"')
df_avg = df.query('date > "2019-07-31"')
```

In [22]:

```
print('Всего событий в июле %d, типов событий %d, экспериментов %d.' % (df
_iul.shape[0], df_iul['event_name'].nunique(), df_iul['exp_id'].nunique
()))
print('Всего событий в августе %d, типов событий %d, экспериментов %d.' %
(df_avg.shape[0], df_avg['event_name'].nunique(), df_avg['exp_id'].nunique
()))
print('Мы отбросили {} пользователей или {:,.1%} от общего количества поль
зователей.'
      .format(df['user_id'].nunique() - df_avg['user_id'].nunique(),
              ((df['user_id'].nunique() - df_avg['user_id'].nunique())/df_
avg['user_id'].nunique())))
```

```
Всего событий в июле 2826, типов событий 5, экспериментов 3.
Всего событий в августе 240887, типов событий 5, эксперименто
в 3.
Мы отбросили 17 пользователей или 0.2% от общего количества п
ользователей.
```

Мы отбросили 2826 событий или 1,2% от общих данных и 17 пользователй мы оставили в июле.

2.5. Подсчет пользователей из всех трех групп

[к Оглавлению](#0.0)

Посмотрим сколько уникальных пользователей было в июле и какое соотношение их по группам

In [23]:

```
iul_user = df_iul.groupby('exp_id')['user_id'].agg(['nunique']).reset_inde
x()
iul_user['procent'] = ((iul_user['nunique']/iul_user['nunique'].sum())*100
).round(1)
iul_user
```

Out[23]:

	exp_id	nunique	procent
0	246	459	31.6
1	247	484	33.4
2	248	508	35.0

Посмотрим сколько уникальных пользователей было в авусте и какое соотношение их по группам

In [24]:

```
avg_user = df_avg.groupby('exp_id')['user_id'].agg(['nunique']).reset_inde
```

```
x()
avg_user['procent'] = ((avg_user['nunique']/avg_user['nunique'].sum())*100
).round(1)
avg_user
```

Out[24]:

	exp_id	nunique	procent
0	246	2484	33.0
1	247	2513	33.4
2	248	2537	33.7

Посмотрим сколько уникальных пользователей было 07 августа и какое соотношение их по группам

In [25]:

```
avg_user_day7 = df_avg.query('date == "2019-08-07"').groupby('exp_id')['us
er_id'].agg(['nunique']).reset_index()
avg_user_day7['procent'] = ((avg_user_day7['nunique']/avg_user_day7['nuniq
ue'].sum())*100).round(1)
avg_user_day7
```

Out[25]:

	exp_id	nunique	procent
0	246	1198	32.8
1	247	1212	33.2
2	248	1241	34.0

Пользователи из всех трех экспериментальных групп есть. Соотношение нормальное, одинаковое.

Вывод:

- Мы обнаружили, что в логе 243713 событий и 5 типов событий.
- Уникальных пользователей 7551
- В среднем на пользователя приходится 302 события. По медиане 20. По моде 5 событий.
- Мы располагаем данными за 14 дней с 2019-07-25 по 2019-08-07. Данные не одинаково полные. В первую неделю произошло только 1,2% всех событий.
- Чистые данные за одну неделю с 2019-08-01 по 2019-08-07.
- Отбросив более старые данные мы потеряли 2826 событий или 1,2% и 17 пользователей или 0,2%.
- Все пользвати, из трех экспериментальных групп, у нас есть.

3. Воронка событий

3.1. Частота и вид событий в логах

[к Оглавлению](#0.0)

Посмотрим какие события есть в логах и их частоту появления.

In [26]:

```
df_event = df_avg.groupby('event_name').agg({'event_name': ['count']}).res
et_index()
df_event.columns = ('event_name', "count")
df_event = df_event.sort_values(by='count', ascending=False)
```

In [27]:

```
fig = go.Figure(go.Funnel(
    y = df_event['event_name'],
    x = df_event['count'],
    textinfo = "value+percent initial"))
fig.update_layout(title_text='Воронка событий', yaxis_title="События")
fig.show()
```

3.2. Количество пользователей совершивших событие

[к Оглавлению](#0.0)

Посчитаем сколько пльзователей совершаликаждое событие и посчитаем долю пользователей, которые хоть раз совершали событие

In [28]:

```
user_event = df_avg.groupby('event_name')['user_id'].nunique().sort_values
(ascending=False).to_frame().reset_index()\
        .rename(columns={'user_id': 'total_users'})
user_event['percent'] = (user_event['total_users'] / df_avg['user_id'].nun
ique()*100).round(1)
user_event
```

Out[28]:

	event_name	total_users	percent
0	MainScreenAppear	7419	98.5
1	OffersScreenAppear	4593	61.0
2	CartScreenAppear	3734	49.6
3	PaymentScreenSuccessful	3539	47.0
4	Tutorial	840	11.1

3.3. Порядок происхождения событий

[к Оглавлению](#0.0)

События происходят в следующем порядке:

- **MainScreenAppear** - появление главного экрана;
- **Tutorial** - прохождение обучения по использоваю приложения (по желанию);
- **OffersScreenAppear** - предложение о товаре (экран с товаром);
- **CartScreenAppear** - переход в корзину;
- **PaymentScreenSuccessful** - экран успешной оплаты заказа.

Этап Tutorial тут лишний, так как обязательным не является и не влияет на то, дойдет ли пользователь до финальной покупки или нет. Да и событий этого типа слишком мало для анализа (по сравнению с другими).

In [29]:

```
df_avg = df_avg[df_avg['event_name'] != 'Tutorial']
```

3.4. Воронка событий по пользователям

[к Оглавлению](#0.0)

Соберем новую табличку по уникальным пользователям и посмотрим распределение пользователей по событиям

In [30]:

```
user_event = df_avg.groupby('event_name')['user_id'].nunique().sort_values
(ascending=False).to_frame().reset_index()\
        .rename(columns={'user_id': 'total_users'})
user_event['percent'] = (user_event['total_users'].apply(lambda x: round(x
/7419*100, 2))).round(1)
user_event['percent-'] = (user_event['total_users'] / user_event['total_us
ers'].sum() * 100).round(1)
user_event
```

Out[30]:

	event_name	total_users	percent	percent-
0	MainScreenAppear	7419	100.0	38.5
1	OffersScreenAppear	4593	61.9	23.8
2	CartScreenAppear	3734	50.3	19.4
3	PaymentScreenSuccessful	3539	47.7	18.4

In [31]:

```
fig = go.Figure(go.Funnel(
    y = user_event['event_name'],
    x = user_event['total_users'],
    textinfo = "value+percent initial"))
fig.update_layout(title_text='Воронка событий по пользователям', yaxis_tit
le="События")
fig.show()
```

- При переходе с первого этапа на второй теряется 38.1% пользователей. Переходит 61.9% пользователей.
- При переходе со второго этапа на третий теряется 18.7% пользователей. Переходит 81.3% пользователей.
- При переходе с третьего на четвертый этап теряется 5.2% пользователей. Дошло до оплаты 47.7% пользователей.

3.5. Потеря наибольшего количества пользователей

[к Оглавлению](#0.0)

Больше всего пользователей теряется на преходе с главного экрана на предложение о товаре (экран с товаром). Это 38.1 %

3.6. Доля пользователей прошедших все события

[к Оглавлению](#0.0)

Доля пользователей, которая доходит от первого события до оплаты 47.7% или 3539 пользователей

Вывод:

- Изучили события в логах и посчитали их частоту.
- 117 тыс. событий появления главного экрана, 46 тыс. событий с предложением о товаре, 42 тыс. событий в корзине и 34 тыс. событий успешных оплат.
- Предположили наиболее вероятный порядок событий и обнаружили, что некоторые пользователи проходят обучение по использованию приложения сразу после главного экрана. 840 пользователей. Этот этап не влияет на покупку и мы его не учитываем при расчетах.
- Доля пользователей на певом шаге составила 61,9%, на втором шаге 50,3, на третьем шаге 47,7
- Наибольшее количестов пользователей теряется на первом шаге, 38,1%.
- От первого события и до оплаты доходит 47,7% пользователей.

4. Результаты эксперимента

4.1. Количество пользователей по группам

[к Оглавлению](#0.0)

In [32]:

```
user_exp = df_avg.groupby('exp_id')['user_id'].nunique().sort_values(ascending=False).to_frame().reset_index()\
        .rename(columns={'user_id': 'total_users'})
user_exp['percent'] = user_exp['total_users'] / user_exp['total_users'].sum() * 100
user_exp
```

Out[32]:

	exp_id	total_users	percent
0	248	2535	33.665339
1	247	2512	33.359894
2	246	2483	32.974768

В эксперементальной группе 248 - 2535 пользователей

4.2. Статистические критерии между выборками 246 и 247

[к Оглавлению](#0.0)

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

Критерии успешного A/A-теста:

- Количество пользователей в различных группах различается не более, чем на 1% или 0.5%
- Для всех групп фиксируют и отправляют в системы аналитики данные об одном и том же
- Различие ключевых метрик по группам не превышает 1% и не имеет статистической значимости
- Попавший в одну из групп посетитель остаётся в этой группе до конца теста. Если пользователь видит разные версии исследуемой страницы в ходе одного исследования, неизвестно, какая именно повлияла на его решения. Значит, и результаты такого теста нельзя интерпретировать однозначно.

In [33]:

```
exp_246 = df_avg.query('exp_id == 246')
exp_247 = df_avg.query('exp_id == 246')
exp_248 = df_avg.query('exp_id == 246')
```

In [34]:

```
funnel_246 = (exp_246.groupby('event_name')['user_id'].nunique().sort_valu
es(ascending=False).to_frame().reset_index()
              .rename(columns={'user_id': 'total_users'}))
funnel_247 = (exp_247.groupby('event_name')['user_id'].nunique().sort_valu
es(ascending=False).to_frame().reset_index()
              .rename(columns={'user_id': 'total_users'}))
funnel_248 = (exp_248.groupby('event_name')['user_id'].nunique().sort_valu
es(ascending=False).to_frame().reset_index()
              .rename(columns={'user_id': 'total_users'}))
funnel_246;
```

In [35]:

```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'exp 246',
    y = funnel_246['event_name'],
    x = funnel_246['total_users'],
    textinfo = "value+percent initial"))

fig.add_trace(go.Funnel(
    name = 'exp 247',
    y = funnel_247['event_name'],
    x = funnel_247['total_users'],
    textinfo = "value+percent initial"))

fig.add_trace(go.Funnel(
    name = 'exp 248',
    y = funnel_248['event_name'],
    x = funnel_248['total_users'],
    textinfo = "value+percent initial"))

fig.show()
```

4.3. Самое популярное событие

[к Оглавлению](#0.0)

Самое популярное событие - главный экран `MainScreenAppear`
Добавим в таблицу данные по группе 246_247, которая будет объединять две А-группы AAB-теста.
Посчитаем долю пользователей, совершивших самое популярное событие

In [36]:

```
users_events = (df_avg.pivot_table(index = 'event_name', columns = 'exp_i
d', values = 'user_id', aggfunc = 'nunique')
                .reset_index().sort_values(246,ascending=False))

users_events['246+247'] = users_events[246] + users_events[247]
users_events['246_percent'] = users_events[246] / users_events[246].sum()
* 100
users_events['247_percent'] = users_events[247] / users_events[247].sum()
* 100
users_events['248_percent'] = users_events[248] / users_events[248].sum()
* 100
users_events = users_events.set_index(users_events.columns[0])
users_events
```

Out[36]:

exp_id	246	247	248	246+247	246_percent	247_percent	2
event_name							
MainScreenAppear	2450	2476	2493	4926	37.937442	38.735920	
OffersScreenAppear	1542	1520	1531	3062	23.877361	23.779725	
CartScreenAppear	1266	1238	1230	2504	19.603592	19.367960	
PaymentScreenSuccessful	1200	1158	1181	2358	18.581604	18.116395	

Добавим количестов пользователей в таблицу

In [37]:

```
user_exp.loc[3] = ['246+247', 4997, 0]
user_exp
```

Out[37]:

	exp_id	total_users	percent
0	248	2535	33.665339
1	247	2512	33.359894
2	246	2483	32.974768
3	246+247	4997	0.000000

Уберем индекс у таблицы

In [38]:

```
user_exp = user_exp.set_index(user_exp.columns[0])
user_exp
```

Out[38]:

exp_id	total_users	percent
248	2535	33.665339
247	2512	33.359894
246	2483	32.974768
246+247	4997	0.000000

Проверим гипотезу о равенстве долей при помощи Z-критерия. Для этого напишем функцию.

Проведем сравнение двух А-групп и убедимся, что тест был проведен корректно и между ними нет стат. значимых различий.
Нулевая гипотеза - между долями нет стат. значимой разницы.
Альтернативная гипотеза - между долями есть стат. значимая разница.

In [39]:

```
# Воспользуемся Z-критерием
def z_test(exp1, exp2, event, alpha):
    p1_ev = users_events.loc[event, exp1]
    p2_ev = users_events.loc[event, exp2]
    p1_us = user_exp.loc[exp1, 'total_users']
    p2_us = user_exp.loc[exp2, 'total_users']
    p1 = p1_ev / p1_us
    p2 = p2_ev / p2_us
    difference = p1 - p2
    p_combined = (p1_ev + p2_ev) / (p1_us + p2_us)
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1 / p
1_us + 1 / p2_us))
    distr = st.norm(0, 1)
```

```
    p_value = (1 - distr.cdf(abs(z_value))) * 2
    print('Проверка для  {} и {}, событие: {}, p-значение: {p_value:.2f}'.
format(exp1, exp2, event, p_value=p_value))
    if (p_value < alpha):
        print("Принимаем альтернативную гипотезу: между долями есть значим
ая разница")
    else:
        print("Оставляем нулевую гипотезу, нет оснований считать доли разн
ыми")
```

In [40]:

```
for event in users_events.index:
    z_test(246, 247, event, 0.05)
    print()
```

```
Проверка для  246 и 247, событие: MainScreenAppear, p-значени
е: 0.75
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246 и 247, событие: OffersScreenAppear, p-значе
ние: 0.25
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246 и 247, событие: CartScreenAppear, p-значени
е: 0.23
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246 и 247, событие: PaymentScreenSuccessful, p-
значение: 0.11
Оставляем нулевую гипотезу, нет оснований считать доли разным
и
```

Тест проведен корректно - стат. значимых различий между двумя АА-группами не выявлено.

4.4. Сравнение результатов с объединенной контрольной группой

[к Оглавлению](#0.0)

Аналогично поступим с группой с измененным шрифтом. Сравним результаты с каждой из контрольных групп в отдельности по каждому событию.
Первый АВ-тест: 246 и 248
Нулевая гипотеза - между долями нет стат. значимой разницы.
Альтернативная гипотеза - между долями есть стат. значимая разница.

In [41]:

```
for event in users_events.index:
    z_test(246, 248, event, 0.05)
```

```
    print()
```

```
Проверка для  246 и 248, событие: MainScreenAppear, p-значени
е: 0.34
Оставляем нулевую гипотезу, нет оснований считать доли разным
и


Проверка для  246 и 248, событие: OffersScreenAppear, p-значе
ние: 0.21
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246 и 248, событие: CartScreenAppear, p-значени
е: 0.08
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246 и 248, событие: PaymentScreenSuccessful, p-
значение: 0.22
Оставляем нулевую гипотезу, нет оснований считать доли разным
и
```

Разницы между объединенной группой 246 и 248 не обнаружено

Второй АВ-тест: 247 и 248
Нулевая гипотеза - между долями нет стат. значимой разницы.
Альтернативная гипотеза - между долями есть стат. значимая разница.

In [42]:

```
for event in users_events.index:
    z_test(247, 248, event, 0.05)
    print()
```

```
Проверка для  247 и 248, событие: MainScreenAppear, p-значени
е: 0.52
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  247 и 248, событие: OffersScreenAppear, p-значе
ние: 0.93
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  247 и 248, событие: CartScreenAppear, p-значени
е: 0.59
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  247 и 248, событие: PaymentScreenSuccessful, p-
значение: 0.73
Оставляем нулевую гипотезу, нет оснований считать доли разным
и
```

Разницы между объединенной группой 247 и 248 не обнаружено

Сравним результаты с объединенной контрольной группой.
Третий AB-тест: 246+247 и 248
Нулевая гипотеза - между долями нет стат. значимой разницы.
Альтернативная гипотеза - между долями есть стат. значимая разница.

In [43]:

```
for event in users_events.index:
    z_test('246+247', 248, event, 0.05)
    print()
```

```
Проверка для  246+247 и 248, событие: MainScreenAppear, p-зна
чение: 0.43
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246+247 и 248, событие: OffersScreenAppear, p-з
начение: 0.46
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246+247 и 248, событие: CartScreenAppear, p-зна
чение: 0.19
Оставляем нулевую гипотезу, нет оснований считать доли разным
и

Проверка для  246+247 и 248, событие: PaymentScreenSuccessfu
l, p-значение: 0.62
Оставляем нулевую гипотезу, нет оснований считать доли разным
и
```

Разницы между объединенной группой 246+247 и 248 не обнаружено

4.5. Итоги проверок статистических гипотез

[к Оглавлению](#0.0)

В рамках множественного сравнения групп мы использовали критический уровень значимости 0.05.
Мы провели 4 проверки статистических гипотез AA-теста и 12 проверок AB-теста

5. Вывод

[к Оглавлению](#0.0)

На этапе подготовки данных мы выявили несущественное количество дублей строк и удалили их.

- Всего событий в логе: 243713.
- Типов событий 5.
- Всего пользователей: 7551.
- В среднем на пользователя приходится 32 события.
- Временные рамки: начало - 2019-07-25, конец - 2019-08-07

- В группе 246 в августе было 2484 уникальных пользователя
- В группе 247 в августе было 2513 уникальных пользователей
- В группе 248 в августе было 2537 уникальных пользователей

Данные в логе были за период в 2 недели, но мы выяснили, что полноценные данные были лишь за 7 дней, поэтому неполноценные данные мы отсекли и взяли период с 1 по 7 августа. Количество данных уменьшилось несущественно, мы отбросили 2826 событий или 1,2% от общих данных.

Больше всего пользователей теряем на первом шаге воронки MainScreenAppear (появление главного экрана) - OffersScreenAppear (предложение о товаре (экран с товаром)) - 38.1%. До оплаты же доходит 47.7% пользователей.

Критический уровень значимости перед проведением тестирований мы взяли 0.05. Перед A/B тестированием мы провели A/A тестирование групп и выяснилось, что всё впорядке и распределение на группы было корректным.

Мы провели A/B тестирование и не нашли статистически значимых отличий между контрольными группами и экспериментальной. Это значит, что изменение шрифта не дало никакого эффекта.

Проводя множественное сравнение при A/A и A/B тестированиях, мы не использовали никакие поправки критического уровня значимости, так как в этом не было никакого смысла - при уменьшении критического уровня значимости с помощью имеющихся поправок результаты тестирования не изменились бы, это видно из полученных p-value (минимальное p-value - 0.08).