Сборный проект №2

Описание проекта

Стартап продаёт продукты питания. Нужно проанализировать поведение пользователей мобильного приложения. Изучить воронку продаж:

- как пользователи доходят до покупки
- сколько пользователей доходит до покупки
- сколько — «застревает» на предыдущих шагах. На каких именно

Провести сследование результатов 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. Нужно выяснить, какой шрифт лучше.

Создание двух групп A вместо одной имеет определённые преимущества. Если две контрольные группы окажутся равны, можно быть уверенным 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.

План работ:

Загрузка данных и подготовка их к анализу

Загрузим данные о scbdfbdb. Убедимся, что тип данных в каждой колонке — правильный, а также отсутствуют пропущенные значения и дубликаты. При необходимости обработаем их.

Откроем файл с данными и изучим общую информацию

In [1]:
```
#Импорт библиотек
import pandas as pd
import numpy as np
import math as mth
import matplotlib.pyplot as plt


from collections import Counter
from scipy import stats as st

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

In [2]:
```
#Сформируем датасет
df = pd.read_csv('logs_exp.csv', sep='\t')
df
```

Out[2]:

	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
...	...	...	...	...
244121	MainScreenAppear	4599628364049201812	1565212345	247
244122	MainScreenAppear	5849806612437486590	1565212439	246
244123	MainScreenAppear	5746969938801999050	1565212483	246
244124	MainScreenAppear	5746969938801999050	1565212498	246
244125	OffersScreenAppear	5746969938801999050	1565212517	246

244126 rows × 4 columns

In [3]:
```
#Чтобы часть столбцов в дальнейшем не скрывалась, настроим принудительное отображени
pd.set_option('display.max_columns', None)
```

In [4]:
```
#Установим ширину ячеек
pd.set_option("max_colwidth", 999)
```

In [5]:
```
#Посмотрим общую информацию
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   EventName       244126 non-null  object
 1   DeviceIDHash    244126 non-null  int64 
 2   EventTimestamp  244126 non-null  int64 
 3   ExpId           244126 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 7.5+ MB
```

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

Out[6]:

	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 [7]:
```
df.tail()
```

Out[7]:

	EventName	DeviceIDHash	EventTimestamp	ExpId
244121	MainScreenAppear	4599628364049201812	1565212345	247
244122	MainScreenAppear	5849806612437486590	1565212439	246
244123	MainScreenAppear	5746969938801999050	1565212483	246
244124	MainScreenAppear	5746969938801999050	1565212498	246
244125	OffersScreenAppear	5746969938801999050	1565212517	246

In [8]:
```
df['ExpId'].value_counts()
```

Out[8]:
```
248    85747
246    80304
247    78075
Name: ExpId, dtype: int64
```

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.

Подготовка данных

Заменим названия столбцов

In [9]:
```
df.columns = ['event_name', 'user_id', 'event_time', 'group']
```

Проверим пропуски и типы данных. Откорректируем, если нужно

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

Out[10]:
```
event_name    0
user_id       0
event_time    0
group         0
dtype: int64
```

In [11]:
```
#Проверим наличие дубликатов
df.duplicated().sum()
```

Out[11]:
```
413
```

In [12]:
```
#Посмотрим на дубли поближе
df[df.duplicated()==True].sort_values(by=['user_id', 'event_time']).tail(15)
```

Out[12]:

	event_name	user_id	event_time	group
219011	PaymentScreenSuccessful	8755591450908418981	1565163744	246
89999	PaymentScreenSuccessful	8835108532520342368	1564833924	246
242329	MainScreenAppear	8870358373313968633	1565206004	247
67295	MainScreenAppear	8878573080913900043	1564766098	247
147568	CartScreenAppear	8942553003070671545	1564992297	247
198995	CartScreenAppear	8942553003070671545	1565102760	247
219655	PaymentScreenSuccessful	8942553003070671545	1565165423	247
236910	MainScreenAppear	8973626519929080220	1565195542	247
140009	OffersScreenAppear	8982013177812162195	1564953333	248
60224	MainScreenAppear	9062390201421847360	1564755334	248
56851	PaymentScreenSuccessful	9110248565804959041	1564750074	248
199849	CartScreenAppear	9160437016685643194	1565103970	247
200171	PaymentScreenSuccessful	9160437016685643194	1565104416	247
204830	PaymentScreenSuccessful	9187990861085277398	1565110888	247
204831	PaymentScreenSuccessful	9187990861085277398	1565110888	247

In [13]:
```
# Удалим дубликаты
df = df.drop_duplicates().reset_index(drop=True)
```

Добавим столбец даты и времени, а также отдельный столбец дат

In [14]:
```
# Преобразуем данные о времени
df['event_time'] = pd.to_datetime(df['event_time'], unit='s')
df['date'] = df['event_time'].dt.date
df['date'] = df['date'].astype('datetime64') #заменим тип данных
df.head()
```

Out[14]:

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

In [15]:
```
#Переименуем группы эксперимента

def sort_group(group):
    if group == 246:
        return 'A'
    elif group == 247:
        return 'A1'
    elif group == 248:
        return 'B'
    else:
        return 'Ошибка'

df['group'] = df['group'].apply(sort_group)
df
```

Out[15]:

	event_name	user_id	event_time	group	date
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	A	2019-07-25
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	A	2019-07-25
2	PaymentScreenSuccessful	3518123091307005509	2019-07-25 11:28:47	B	2019-07-25
3	CartScreenAppear	3518123091307005509	2019-07-25 11:28:47	B	2019-07-25
4	PaymentScreenSuccessful	6217807653094995999	2019-07-25 11:48:42	B	2019-07-25
...	...	...	...	...	...
243708	MainScreenAppear	4599628364049201812	2019-08-07 21:12:25	A1	2019-08-07
243709	MainScreenAppear	5849806612437486590	2019-08-07 21:13:59	A	2019-08-07
243710	MainScreenAppear	5746969938801999050	2019-08-07 21:14:43	A	2019-08-07
243711	MainScreenAppear	5746969938801999050	2019-08-07 21:14:58	A	2019-08-07
243712	OffersScreenAppear	5746969938801999050	2019-08-07 21:15:17	A	2019-08-07

243713 rows × 5 columns

In [16]:
```
#Посмотрим еще раз общую информацию
```

```
df.info()
```

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

Вывод

Данные в хорошем состоянии.

Пропущенных значений не было, дубликаты удалили, переименовали столбцы, добавили новые столбцы с датами и изменили тип данных. Изменили названия тестовых групп.

Можно приступать к анализу

Изучение и проверка данных

Количество событий в логе

In [17]:
```
# Сколько всего событий в логе
events = df['event_name'].count()
print('Всего в логе', events, 'событий.')
```

```
Всего в логе 243713 событий.
```

In [18]:
```
# Сколько уникальных вариантов событий в логе
df['event_name'].value_counts()
```

Out[18]:
```
MainScreenAppear           119101
OffersScreenAppear          46808
CartScreenAppear            42668
PaymentScreenSuccessful     34118
Tutorial                     1018
Name: event_name, dtype: int64
```

Количество пользователей в логе

In [19]:
```
# Сколько всего пользователей в логе
users = df['user_id'].nunique()
print('Количество пользователей в логе:', users)
```

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

Количество событий на пользователя

In [20]:
```
print('Количество событий в среднем на пользователя:', (events / users).round())
```

```
Количество событий в среднем на пользователя: 32.0
```

Определение периода данных

Выясним, данными за какой период мы располагаем. Найдем максимальную и минимальную дату.

In [21]:
```
print(f"Период данных которым мы располагаем: с {df['event_time'].min()} по {df['eve
print(f"Есть период размером : {df['event_time'].max() - df['event_time'].min()}")
```

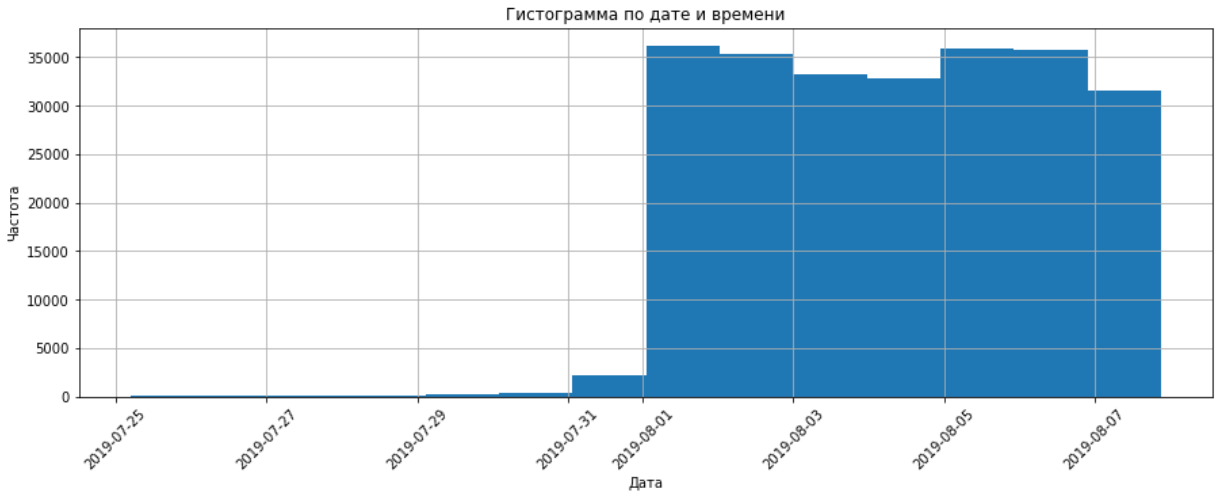
```
Период данных которым мы располагаем: с 2019-07-25 04:43:36 по 2019-08-07 21:15:17
Есть период размером : 13 days 16:31:41
```

Построим 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.

Выясним, данными за какой период времени мы располагаем на самом деле.

In [22]:
```
#Построим график
plt.figure(figsize=(15,5))
ax = df['event_time'].hist(bins=df['date'].nunique())
plt.title('Гистограмма по дате и времени')
plt.ylabel("Частота")
plt.xlabel("Дата")
plt.xticks(rotation=45)
plt.show()
```



Как видим на графике, мы владем данными только второй недели из дух, начиная с 1 августа.

Первую неделю можно удалить, т.к. по ней очень мало данных.

In [23]:
```
df_new = df.loc[df['date'] > '2019-07-31'].reset_index(drop=True)
df_new.sort_values(by='event_time')
```

Out[23]:

	event_name	user_id	event_time	group	date
0	Tutorial	3737462046622621720	2019-08-01 00:07:28	A	2019-08-01
1	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	A	2019-08-01
2	MainScreenAppear	3737462046622621720	2019-08-01 00:08:55	A	2019-08-01
3	OffersScreenAppear	3737462046622621720	2019-08-01 00:08:58	A	2019-08-01
4	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	A1	2019-08-01

	event_name	user_id	event_time	group	date
...	...	...	...	...	...
240882	MainScreenAppear	4599628364049201812	2019-08-07 21:12:25	A1	2019-08-07
240883	MainScreenAppear	5849806612437486590	2019-08-07 21:13:59	A	2019-08-07
240884	MainScreenAppear	5746969938801999050	2019-08-07 21:14:43	A	2019-08-07
240885	MainScreenAppear	5746969938801999050	2019-08-07 21:14:58	A	2019-08-07
240886	OffersScreenAppear	5746969938801999050	2019-08-07 21:15:17	A	2019-08-07

240887 rows × 5 columns

Оценка потерь событий и пользователей

Посмотрим сколько событий и пользователей мы потеряли, отбросив старые данные.

In [24]:
```
# Потеря событий
events_new = df_new['event_name'].count()
print('Всего потеряно', (events - events_new), 'событий.')
print('Что составляет', ((events - events_new)/events*100).round(), '% от изначально
```

```
Всего потеряно 2826 событий.
Что составляет 1.0 % от изначальной базы.
```

In [25]:
```
# Потеря пользователей
users_new = df_new['user_id'].nunique()
print('Потеряно пользователей :', (users - users_new))
print('Что составляет', (users - users_new)/users*100, '% от изначальной базы.')
```

```
Потеряно пользователей : 17
Что составляет 0.22513574361011784 % от изначальной базы.
```

Отбросив старые данные мы потеряли 17 пользователей и 2826 событий.

Проверка пользователи из экспериментальных групп

In [26]:
```
# Наличие данных из всех трёх экспериментальных групп
df_new['group'].value_counts()
```

Out[26]:
```
B     84563
A     79302
A1    77022
Name: group, dtype: int64
```

In [27]:
```
# Наличие пользователи из всех трёх экспериментальных групп
df_new.groupby('group')['user_id'].nunique()
```

Out[27]:
```
group
A     2484
A1    2513
B     2537
Name: user_id, dtype: int64
```

Имеются данные оользователях из всех трёх экспериментальных групп.

Выводы

Мы выяснили, что изначально в логе было 243713 событий и 7551 пользователей.
Определили, что имеется период данных с 2019-07-25 04:43:36 по 2019-08-07 21:15:17.
Но только начиная с 1 августа данные можно считать полными. Поэтому мы удалили часть данных за первую неделю.
Отбросив старые данные мы потеряли 17 пользователей и 2826 событий.
При этом данные оользователях есть из всех трёх экспериментальных групп.

Воронка событий

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

Отсортируем события по частоте

In [28]:
```
df_new['event_name'].value_counts()
```

Out[28]:
```
MainScreenAppear           117328
OffersScreenAppear          46333
CartScreenAppear            42303
PaymentScreenSuccessful     33918
Tutorial                     1005
Name: event_name, dtype: int64
```

Пользователи и события

Выясним, сколько пользователей совершали каждое из этих событий.
Отсортируем события по числу пользователей.
Посчитаем долю пользователей, которые хоть раз совершали событие.

In [29]:
```
events_funnel = df_new.groupby('event_name')['user_id'].nunique().sort_values(ascend
        .rename(columns={'user_id': 'total_users'})

events_funnel['percent_1event'] = (events_funnel['total_users'] / df_new['user_id'].
events_funnel
```

Out[29]:

	event_name	total_users	percent_1event
0	MainScreenAppear	7419	98.0
1	OffersScreenAppear	4593	61.0
2	CartScreenAppear	3734	50.0
3	PaymentScreenSuccessful	3539	47.0
4	Tutorial	840	11.0

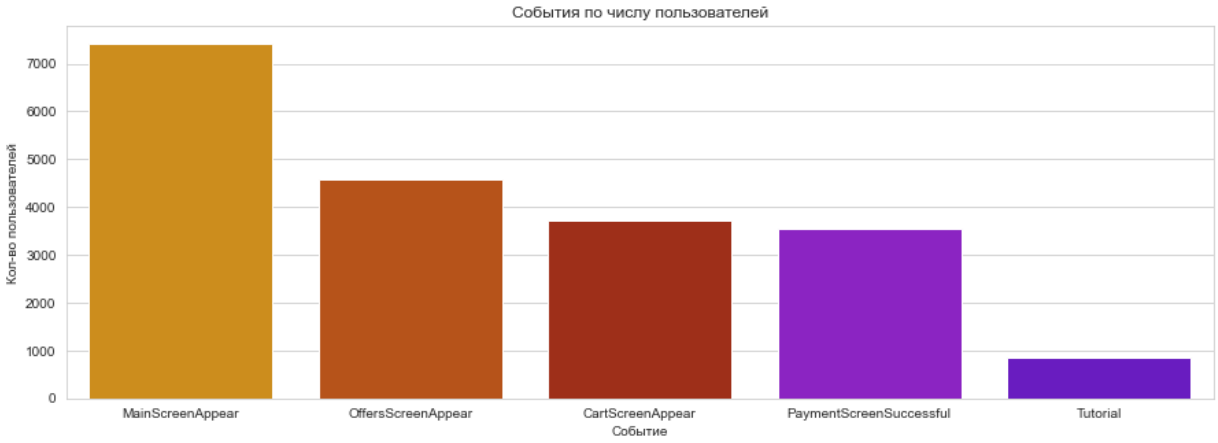
Предположим, в каком порядке происходят события

Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки

In [30]:
```
sns.set_style("whitegrid")
plt.figure(figsize=(15,5))
sns.barplot(y='total_users', x='event_name', data=events_funnel, palette='gnuplot_r'
```

```
plt.title('События по числу пользователей')
plt.xlabel('Событие')
plt.ylabel('Кол-во пользователей')

plt.show();
```



In [31]:

```
fig = go.Figure()
fig = go.Figure(go.Funnel(
    y = events_funnel['event_name'],
    x = events_funnel['total_users']))
fig.show()
```

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.

Событие Tutorial (Обучение) явно лишнее

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 [32]:
```
# Исключим событие Tutorial
events_funnel = events_funnel.query('event_name != "Tutorial"')
```

In [33]:
```
# Доля пользователей следующего шага
fig = go.Figure()
fig = go.Figure(go.Funnel(
    y = events_funnel['event_name'],
    x = events_funnel['total_users'],
    textinfo = "value+percent initial+percent previous"))
fig.show()
```

Из 7419 пользователей с первого шага до последнего доходят только 3539, что составляет 48%.
Это очень хороший показатель, особенно если учесть, что 38% отваливается сразу после первого этапа.

Как мы видим, на шаге OffersScreenAppear (экран с товаром) теряется больше всего пользователей - 38%. Отделу маркетинга стоит поработать с данным моментом, чтобы уменьшить отвал пользователей.

Изучение результатов эксперимента

Пользователи в каждой экспериментальной группе

Сколько пользователей в каждой экспериментальной группе

In [34]:
```
# Посмотрим сколько пользователей в каждой экспериментальной группе
# Исключим событие Tutorial
df_new = df_new.query('event_name != "Tutorial"')
users_group = df_new.groupby('group')['user_id'].nunique()
users_group['AA1'] = users_group['A'] + users_group['A1']
users_group
```

Out[34]:
```
group
A       2483
A1      2512
B       2535
AA1     4995
Name: user_id, dtype: int64
```

Количество пользователей в группах сопоставимо.

Оценка контрольных экспериментальных групп

Есть 2 контрольные группы для A/A-эксперимента, чтобы проверить корректность всех механизмов и расчётов.
Проверим, находят ли статистические критерии разницу между выборками A и A1.

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 [35]:
```
# Число пользователей в каждой группе
users_A = (df_new[df_new['group'] == 'A']['user_id']).nunique()
```

```
users_A1 = (df_new[df_new['group'] == 'A1']['user_id']).nunique()
users_B = (df_new[df_new['group'] == 'B']['user_id']).nunique()
```

In [36]:

```
print('Количество пользователей в группах A-A1 различается на', round(100 - (users_A
print('Количество пользователей в группах A-B различается на', round(100 - (users_A
print('Количество пользователей в группах A1-B различается на', round(100 - (users_A
```

```
Количество пользователей в группах A-A1 различается на 1.15 %
Количество пользователей в группах A-B различается на 2.05 %
Количество пользователей в группах A1-B различается на 0.91 %
```

Как мы выяснили, группы отличаются по количеству более, чем на 1%. Далее мы посмотрим на сколько эти различия статистически значимы.

Самое популярное событие

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

Посчитаем долю пользователей, совершивших это событие.

Проверим, будет ли отличие между группами статистически достоверным.

In [37]:

```
# Создадим таблицу с количеством и долей пользователей из каждой группы с разбивкой

df_test = df_new.pivot_table(
    index='event_name',
    columns='group',
    values='user_id',
    aggfunc='nunique').sort_values(by='A', ascending=False).reset_index()

df_test['AA1'] = df_test['A1'] + df_test['A']

df_test['share_A'] = (df_test['A'] / users_group['A'] * 100).round(1)
df_test['share_A1'] = (df_test['A1'] / users_group['A1'] * 100).round(1)
df_test['share_B'] = (df_test['B'] / users_group['B'] * 100).round(1)
df_test['share_AA1'] = ((df_test['A'] + df_test['A1']) / (users_group['A'] + users_g

df_test
```

Out[37]:

group	event_name	A	A1	B	AA1	share_A	share_A1	share_B	share_AA1
0	MainScreenAppear	2450	2476	2493	4926	98.7	98.6	98.3	98.6
1	OffersScreenAppear	1542	1520	1531	3062	62.1	60.5	60.4	61.3
2	CartScreenAppear	1266	1238	1230	2504	51.0	49.3	48.5	50.1
3	PaymentScreenSuccessful	1200	1158	1181	2358	48.3	46.1	46.6	47.2

Около 1.5% пользователей каждой группы не совершили первый шаг (Главный экран). Вероятно, это произошло из-за того, что мы обрезали часть старых данных.

In [38]:

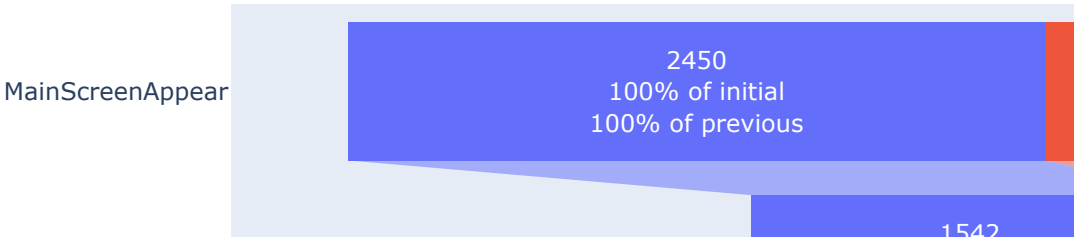
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'A',
    y = df_test['event_name'],
    x = df_test['A'],
    textinfo = "value+percent initial+percent previous"))
```

```
fig.add_trace(go.Funnel(
    name = 'A1',
    y = df_test['event_name'],
    x = df_test['A1'],
    textinfo = "value+percent initial+percent previous"))

fig.add_trace(go.Funnel(
    name = 'B',
    y = df_test['event_name'],
    x = df_test['B'],
    textinfo = "value+percent initial+percent previous"))

fig.show()
```



Судя по воронке продаж, данных всех трех групп очень близки по значениям.

Сравним доли клиентов, сделавших заказ.

- Выборка A: из 2450 зарегистрировавшихся пользователей, заказ оплатили 1200
- Выборка A1: из 2476 зарегистрировавшихся пользователей, заказ оплатили 1158

Проверем, находят ли статистические критерии разницу между выборками A и A1

Зададим гипотезы для наших тестов:

H0: Статистически значимой разницы между конверсиями пользователей в оплату заказа между выборками нет

H1: Статистически значимая разница между конверсиями пользователей в оплату заказа между группами есть

In [39]:
```
alpha = 0.01  # критический уровень статистической значимости

Payment = np.array([1200, 1158])
MainScreen = np.array([2450, 2476])

# ваш код
# пропорция успехов в первой группе:
p1 = Payment[0]/MainScreen[0]

# пропорция успехов во второй группе:
p2 = Payment[1]/MainScreen[1]

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

# разница пропорций в датасетах
difference = p1 - p2

# считаем статистику в ст.отклонениях стандартного нормального распределения
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1 / MainScreen[0] +

# 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 группами есть значимая разница')
else:
    print(
        'Не получилось отвергнуть нулевую гипотезу, нет оснований считать группы раз
    )
```

```
p-значение:  0.12044299485641763
Не получилось отвергнуть нулевую гипотезу, нет оснований считать группы разными
```

z-тест показывает, что нет статистически значимой разницы между конверсиями пользователей в оплату заказа между выборками A и A1.

Проверим, будет ли отличие между группами статистически достоверным для всех событий. Обернём проверку в отдельную функцию.
Решим, можно ли сказать, что разбиение на группы работает корректно.

In [40]:
```
def z_test(group1, group2, alpha):
    p1_event = df_test.loc[event_name, group1]
    p2_event = df_test.loc[event_name, group2]
    p1_users = users_group.loc[group1]
    p2_users = users_group.loc[group2]

    # пропорция успехов в первой группе:
    p1 = p1_event / p1_users
    # пропорция успехов во второй группе
    p2 = p2_event / p2_users

    # пропорция успехов в комбинированном датасете
    p_combined = (p1_event + p2_event) / (p1_users + p2_users)
```

```
    # разница пропорций в датасетах
    difference = p1 - p2

    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1 / p1_users +

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)
    p_value = (1 - distr.cdf(abs(z_value))) * 2

    # введем поправку Бонферрони, поделим альфа на число гипотез для всех проверок
    bonferroni_alpha = alpha / 17

    print('Проверка для групп {} и {}, событие: {}, p-значение: {p_value:.2f}'.forma
    if (p_value < bonferroni_alpha):
        print("Отвергаем нулевую гипотезу о равенстве данных в группах")
    else:
        print("Не получилось отвергнуть нулевую гипотезу о равенстве данных в группа
```

In [41]:

```
for event_name in df_test.index:
    z_test('A', 'A1', 0.01)
    print()
```

```
Проверка для групп A и A1, событие: 0, p-значение: 0.75
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп A и A1, событие: 1, p-значение: 0.25
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп A и A1, событие: 2, p-значение: 0.23
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп A и A1, событие: 3, p-значение: 0.11
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах
```

Вывод: разбивка на группы работает корректно, несмотря на отличия между группами.

Нет статистически значимой разницы для всех событий воронки продаж в группах.

Исследование группы B с изменённым шрифтом

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

In [42]:

```
# Сравним группу B с тестовой группой A
for event_name in df_test.index:
    z_test('B', 'A', 0.01)
    print()
```

```
Проверка для групп B и A, событие: 0, p-значение: 0.34
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A, событие: 1, p-значение: 0.21
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A, событие: 2, p-значение: 0.08
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A, событие: 3, p-значение: 0.22
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах
```

Вывод: Нет статистически значимой разницы для всех событий воронки продаж в группах B и A.

In [43]:
```
# Сравним группу B с тестовой группой A1
for event_name in df_test.index:
    z_test('B', 'A1', 0.01)
    print()
```

```
Проверка для групп B и A1, событие: 0, p-значение: 0.52
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A1, событие: 1, p-значение: 0.93
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A1, событие: 2, p-значение: 0.59
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп B и A1, событие: 3, p-значение: 0.73
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах
```

Вывод: Нет статистически значимой разницы для всех событий воронки продаж в группах B и A1.

Сравним результаты с объединённой контрольной группой.
Сделаем выводы из эксперимента

In [44]:
```
# Сравним группу B с объединённой контрольной группой A+A1
for event_name in df_test.index:
    z_test('AA1', 'B', 0.01)
    print()
```

```
Проверка для групп AA1 и B, событие: 0, p-значение: 0.35
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп AA1 и B, событие: 1, p-значение: 0.45
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп AA1 и B, событие: 2, p-значение: 0.19
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах

Проверка для групп AA1 и B, событие: 3, p-значение: 0.61
Не получилось отвергнуть нулевую гипотезу о равенстве данных в группах
```

Вывод: Нет статистически значимой разницы для всех событий воронки продаж между группой B с измененным шрифтом и объединенными данными двух контрольных групп A и A1.

Выводы

Перед нами стояла задача провести анализ влияния изменения шрифта на поведение пользоватлей.

Пропущенных значений в исходных даных не было, дубликаты мы удалили, переименовали столбцы, добавили новые столбцы с датами и изменили тип данных. Изменили названия тестовых групп.

Мы выяснили, что изначально в логе было 243713 событий и 7551 пользователей. Определили, что имеется период данных с 2019-07-25 04:43:36 по 2019-08-07 21:15:17. Но только начиная с 1 августа данные можно считать полными. Поэтому мы удалили часть данных за первую неделю. Отбросив старые данные мы потеряли 9 пользователей и 796 событий. При этом отмечено, что данные о пользователях есть из всех трёх экспериментальных групп. В каждой из которых примерно по 2500 пользователей.

Анализ воронки продаж показал: Из 7429 пользователей с первого шага до последнего доходят только 3542, что составляет 48%. Это очень хороший показатель, особенно если учесть, что 38% отваливается сразу после первого этапа.

На шаге OffersScreenAppear (экран с товаром) теряется больше всего пользователей - 38%. Отделу маркетинга/web-аналитикам тут стоит поработать, чтобы уменьшить отвал пользователей на данном шаге.

Событие "Tutorial" не имеет влияние на заказы, можно исключить отслеживание этого шага в дальнейшем.

Группы для A/A/B-тестов были корректными, несмотря на разницу 1.27% между группами A и A1 в количестве пользователей, что превышает рекомендованные 0.5-1%

При проверке статистических гипотез мы выбрали уровень значимости равный 1%. Мы сделали 17 проверок статистических гипотез и применили поправку Бонферрони, так как работали с множественным сравнением.

Статистически значимой разницы между группами во всех тестах выявлено не было. Значит, изменение шрифта не повлияло ни на один этап воронки продаж. Опасения менеджеров, что пользователям будет непривычно не оправдались. Изменения шрифтов для всех пользователей можно применять спокойно.