A/A/B-тест для проверки гипотезы о том, что изменение шрифта в приложении повлияет на пользователей

Описание проекта: В мобильном приложении по продаже продуктов питания 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.

Цель проекта: Необходимо изучить воронку продаж, чтобы узнать, как пользователи доходят до покупки, а также провести анализ результатов A/A/B-теста и узнать, какой шрифт для приложения будет лучше.

Описание данных: В проекте один датасет с логами, каждый из которых содержит информацию о действии пользователя или событии.Путь к файлу: `/datasets/logs_exp.csv`.

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

Импортируем библиотеки

In [1]:
```
import pandas as pd
import datetime as dt
import numpy as np
import math as mth
import matplotlib.pyplot as plt
import scipy.stats as st
from pandas.plotting import register_matplotlib_converters
import warnings
from plotly import graph_objects as go
# Настроим полное отображение записи в датафрейме
pd.set_option('display.max_colwidth', None)
```

Шаг 1. Откройте файл с данными и изучите общую информацию

In [2]:
```
# Откроем файл
data = pd.read_csv('C:/Users/dimch/OneDrive/Рабочий стол/Яндекс Практикум/Datasets/logs_exp.csv', sep='\t')
# Выведем первые 20 строк
data.head(20)
```

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
5	CartScreenAppear	6217807653094995999	1564055323	248
6	OffersScreenAppear	8351860793733343758	1564066242	246
7	MainScreenAppear	5682100281902512875	1564085677	246
8	MainScreenAppear	1850981295691852772	1564086702	247
9	MainScreenAppear	5407636962369102641	1564112112	246
10	MainScreenAppear	9484657125123903 82	1564119214	247
11	MainScreenAppear	2547684315586332355	1564123826	248
12	MainScreenAppear	8885295911290764495	1564124085	248
13	MainScreenAppear	2140904690380565988	1564125732	247
14	MainScreenAppear	4444236400320272864	1564135560	246
15	MainScreenAppear	8947220251154009657	1564140668	248
16	MainScreenAppear	5839517684026830712	1564141421	247
17	MainScreenAppear	7540130374989658208	1564144283	248
18	CartScreenAppear	2575393697599976818	1564148945	247
19	MainScreenAppear	4651149533106703820	1564149866	246

In [3]:
```
# Выведем информацию о данных
data.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
```

Вывод: Необходимо поменять тип данных в столбце `EventTimestamp`, а также привести названия столбцов к "змеиному" регистру.

Шаг 2. Подготовьте данные

In [4]:
```
# Приведём названия столбцов к "змеиному" регистру
data.columns = ['event', 'device_id', 'event_time', 'exp_id']
```

In [5]:
```
# Приведём столбец с датами к формату datetime
data['event_time'] = pd.to_datetime(data['event_time'], unit='s')
# Создадим столбец с датами событий
data['dt'] = data['event_time'].dt.strftime('%Y-%m-%d')
data.head(5)
```

Out[5]:

	event	device_id	event_time	exp_id	dt
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 [6]:
```
# Проверим на дубликаты
data.duplicated().sum()
```

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

In [7]:
```
# Удалим дубликаты
data = data.drop_duplicates().reset_index(drop=True)
```

```
data.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
# Проверим данные на наличие пропусков
data.isna().sum()
```

Out[8]:
```
event         0
device_id     0
event_time    0
exp_id        0
dt            0
dtype: int64
```

Вывод: Названия столбцов приведены к "змеиному" регистру, столбец с датами приведён к формату datetime, создан столбец с датами событий, датафрейм очищен от дубликатов, пропусков не обнаружено.

Шаг 3. Изучите и проверьте данные

In [9]:
```
# Посчитаем события в логе
print('Всего событий в логе:', data['event'].count())
print('Уникальных событий в логе:', data['event'].nunique())
print('Событий в среднем на одного пользователя:', round(data.groupby('device_id')['event'].count().mean(), 0))
# Посчитаем количество пользователей в логе
print('Всего пользователей в логе:', data['device_id'].nunique())
print('Всего пользователей в группе контрольной 246:', data[data['exp_id']==246]['device_id'].nunique())
print('Всего пользователей в группе контрольной 247:', data[data['exp_id']==247]['device_id'].nunique())
print('Всего пользователей в группе экпериментальной 248:', data[data['exp_id']==248]['device_id'].nunique())
```

```
Всего событий в логе: 243713
Уникальных событий в логе: 5
Событий в среднем на одного пользователя: 32.0
Всего пользователей в логе: 7551
Всего пользователей в группе контрольной 246: 2489
Всего пользователей в группе контрольной 247: 2520
Всего пользователей в группе экпериментальной 248: 2542
```

In [10]:
```
# Найдём максимальную и минимальную дату
print('Минимальная дата', data['dt'].min())
print('Максимальная дата', data['dt'].max())
```

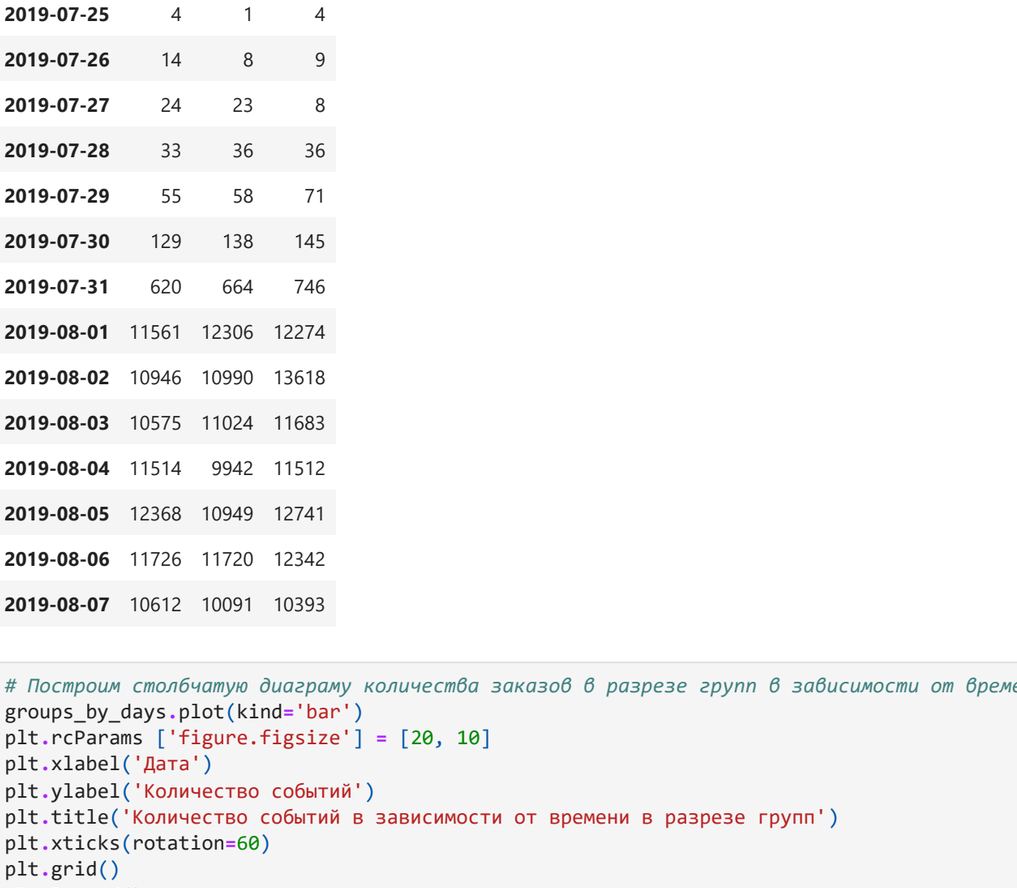
```
Минимальная дата 2019-07-25
Максимальная дата 2019-08-07
```

In [11]:
```
# Создаём датафрейм с агрегированными данными с количеством событий, разбитым по наименованию тестовой группы
groups_by_days = data.pivot_table(values='device_id', index='dt', columns='exp_id', aggfunc='count')
groups_by_days
```

Out[11]:

exp_id	246	247	248
dt			
2019-07-25	4	1	4
2019-07-26	14	8	9
2019-07-27	24	23	8
2019-07-28	33	36	36
2019-07-29	55	58	71
2019-07-30	129	138	145
2019-07-31	620	664	746
2019-08-01	11561	12306	12274
2019-08-02	10946	10990	13618
2019-08-03	10575	11024	11683
2019-08-04	11514	9942	11512
2019-08-05	12368	10949	12741
2019-08-06	11726	11720	12342
2019-08-07	10612	10091	10393

In [12]:
```
# Построим столбчатую диаграму количества заказов в разрезе групп в зависимости от времени
groups_by_days.plot(kind='bar')
plt.rcParams ['figure.figsize'] = [20, 10]
plt.xlabel('Дата')
plt.ylabel('Количество событий')
plt.title('Количество событий в зависимости от времени в разрезе групп')
plt.xticks(rotation=60)
plt.grid()
plt.legend();
```



По графику видно, что с 25.07.2019 по 31.07.2019 данных об экспериментах по группам сильно меньше, чем с 01.08.2019 по 07.08.2019. Данные за первую неделю могут негативно повлиять на исследование. Отбросим их, так как только за период с 01.08 по 07.08 мы располагаем полными данными.

In [13]:
```
# Отбросим неполные данные
data_filtr = data.query('dt >= "2019-08-01"').reset_index(drop=True)
```

In [14]:
```
# Проверим как поменялось количество событий в логе
print('Всего событий в логе:', data_filtr['event'].count())
print('Уникальных событий в логе:', data_filtr['event'].nunique())
print('Событий в среднем на одного пользователя:', round(data_filtr.groupby('device_id')['event'].count().mean(), 0))
# Посчитаем процент, на который уменьшилось количество событий
ev_delta = "{:.2%}".format(round((data['event'].count() - data_filtr['event'].count())/data['event'].count(), 3))
ev_dlt = data['event'].count() - data_filtr['event'].count()
print(f'Количество событий уменьшилось на {ev_delta} или на {ev_dlt}')
# Проверим как поменялось количество пользователей в логе
print('Всего пользователей в логе:', data_filtr['device_id'].nunique())
```

```
print('Всего пользователей в группе контрольной 246:', data_filtr[data_filtr['exp_id']==246]['device_id'].nunique())
print('Всего пользователей в группе контрольной 247:', data_filtr[data_filtr['exp_id']==247]['device_id'].nunique())
print('Всего пользователей в группе экспериментальной 248:', data_filtr[data_filtr['exp_id']==248]['device_id'].nunique())
dev_delta = "{:.2%}".format(round((data['device_id'].nunique() - data_filtr['device_id'].nunique())/data['device_id'].nunique(), 3))
dev_dlt = data['device_id'].nunique() - data_filtr['device_id'].nunique()
print(f'Количество пользователей уменьшилось на {dev_delta} или на {dev_dlt}')
```

```
Всего событий в логе: 240887
Уникальных событий в логе: 5
События в среднем на одного пользователя: 32.0
Количество событий уменьшилось на 1.20% или на 2826
Всего пользователей в логе: 7534
Всего пользователей в группе контрольной 246: 2484
Всего пользователей в группе контрольной 247: 2513
Всего пользователей в группе экспериментальной 248: 2537
Количество пользователей уменьшилось на 0.20% или на 17
```

Вывод: В ходе проверки данных выяснилось, что в логи новых дней по некоторым пользователям «доехали» события из прошлого. Для дальнейшего исследования пришлось отбросить эти данные и оставить только данные за период с 01.08.2019 по 07.08.2019. После очистки количество событий уменьшилось на 1,2%, а количество пользователей на 0,2%.

Шаг 4. Изучите воронку событий

In [15]:
```
# Посмотрим, какие события есть в логах, как часто они встречаются
data_filtr.pivot_table(index='event', values='device_id', aggfunc=pd.Series.count).sort_values(by='device_id', ascending=False).rename(columns={'device_id':'count'})
```

Out[15]:

event	count
MainScreenAppear	117328
OffersScreenAppear	46333
CartScreenAppear	42303
PaymentScreenSuccessful	33918
Tutorial	1005

- MainScreenAppear- просмотр главной страницы, самое популярное событие;
- OffersScreenAppear- просмотр страницы с предложением;
- CartScreenAppear- просмотр корзины;
- PaymentScreenSuccessful- успешная оплата;
- Tutorial- прохождение обучения.

In [16]:
```
# Посчитаем, сколько пользователей совершали каждое из этих событий
events_by_users = data_filtr.query('event != "Tutorial"').pivot_table(index='event', values='device_id', aggfunc=pd.Series.nunique)
events_by_users['share%'] = round(events_by_users['device_id']/data_filtr['device_id'].nunique()*100, 2)
events_by_users = events_by_users.rename(columns={'device_id':'users'})
events_by_users = events_by_users.sort_values(by='users', ascending=False)
events_by_users
```

Out[16]:

event	users	share%
MainScreenAppear	7419	98.47
OffersScreenAppear	4593	60.96
CartScreenAppear	3734	49.56
PaymentScreenSuccessful	3539	46.97

Порядок событий почти как в полученной таблице, за исключением обучения, скорей всего большинство пользователей его просто пропускает, исключим из списка данное событие для дальнейшего исследования. На главную страницу заходят 98,47% пользователей, к странице товара приходят уже 60,96%, просматривают корзину 49,56%, а совершают оплату заказа 46,97%.

In [17]:
```
# Построим диаграмму воронки
fig = go.Figure(go.Funnel(x = events_by_users['users'], y = events_by_users.index, textinfo = "value+percent initial+percent previous"))
fig.update_layout(title='Диаграмма воронки событий', title_x = 0.5)
fig.show();
```

Все пользователи из тех, кто совершил хотя бы одно действие, заходили на главную страницу, страницу с предложениями просмотрели 62% пользователей, просмотрели корзину 50%(81% от просмотревших страницу предложений), а оплатили заказ 48%(95% от просмотревших корзину).

Шаг 5. Изучите результаты эксперимента

In [18]:
```
# Посмотрим, сколько пользователей в каждой экспериментальной группе
users_by_groups = users_by_groups = data_filtr.query('event != "Tutorial"').pivot_table(columns='exp_id', values='device_id', aggfunc=pd.Series.nunique).reset_index()
users_by_groups =users_by_groups.drop('index', axis=1)
# Создадим столбец с суммой пользователей контрольных групп
users_by_groups['246&247'] = users_by_groups[246] +users_by_groups[247]
users_by_groups
```

Out[18]:

exp_id	246	247	248	246&247
0	2483	2512	2535	4995

In [19]:
```
# Посмотрим, сколько пользователей в каждой экспериментальной группе по событиям
events_by_group = data_filtr.query('event != "Tutorial"').pivot_table(index='event', columns='exp_id', values='device_id', aggfunc=pd.Series.nunique).sort_values(246, ascending=False).reset_index()
# Сделаем столбец с объединенной контрольной группой
events_by_group['246&247'] = events_by_group[246] + events_by_group[247]
events_by_group
```

Out[19]:

exp_id	event	246	247	248	246&247
0	MainScreenAppear	2450	2476	2493	4926
1	OffersScreenAppear	1542	1520	1531	3062
2	CartScreenAppear	1266	1238	1230	2504
3	PaymentScreenSuccessful	1200	1158	1181	2358

In [20]:
```
# Проверим вероятность попадания пользователей в несколько групп
def intersection_list(A1, A2, B):
    intersection = [value for value in A1 if value in A2 or B]
    return intersection
intersection_users = intersection_list(data_filtr.query('exp_id == "246"')['device_id'].unique(), data_filtr.query('exp_id == "247"')['device_id'].unique(), data_filtr.query('exp_id == "248"')['device_id
print(len(intersection_users))
```

```
0
```

Группы не пересекаются по пользователям.

Проведём A/A-тест.

Сформулируем гипотезы

- H_0 : статистически значимых различий между контрольными группами нет
- H_1 : статистически значимые различия между контрольными группами есть

Уровень статистической значимости возьму 1%, так как для успешного A/A-теста различие ключевых метрик не должно превышать этого значения

In [21]:
```
# Напишем функцию для проведения z-теста
def z_test(exp_1, exp_2, alpha):
    for e in events_by_group.index:
        # Количество успехов в групах
        successes1 = events_by_group[exp_1][e]
        successes2 = events_by_group[exp_2][e]
        # Количество попыток в групах
        trials1 = users_by_groups.at[0, exp_1]
        trials2 = users_by_groups.at[0, exp_2]
        # пропорция успехов в первой группе:
        p1 = successes1 / trials1
        # пропорция успехов во второй группе:
        p2 = successes2 / trials2
        # пропорция успехов в комбинированном датасете:
        p_combined = (successes1 + successes2) / (trials1 + trials2)
        # 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/trials1 + 1/trials2))
        # 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: {}'.format(events_by_group['event'][e], p_value))
        if (p_value < alpha):
            print("Отвергаем нулевую гипотезу: между долями есть статистически значимая разница")
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными")
        print('')
# alpha берём 1%, так как для успешного A/A-теста различие ключевых метрик не должно превышать этого значения
z_test(246, 247, 0.01)
```

```
MainScreenAppear p-значение: 0.7526703436483038
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.24786096925282264
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.22867643757335676
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.11446627829276612
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

По всем событиям статистически значимых различий между контрольными группами нет, это значит, что A/A-тест прошёл успешно.

Проведём A1/B-тест

СформулируемДальше гипотезы остаются такими же

- H_0 : статистически значимыхразличий между группами нет
- H_1 : статистически значимые различия между группами есть

Уровень статистической значимости возьму 5%.

In [22]:
```
# alpha берём 5%
z_test(246, 248, 0.01)
```

```
MainScreenAppear p-значение: 0.3387114076159288
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.21442476639710506
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.08067367598823139
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.21693033984516674
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Между первой контрольной группой и экспериментальной группой нет статистически значимых различий.

Проведём A2/B-тест

In [23]:
```
z_test(247, 248, 0.05)
```

```
MainScreenAppear p-значение: 0.5194964354051703
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.9333751305879443
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.5878284605111943
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.7275718682261119
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Между второй контрольной группой и экспериментальной группой нет статистически значимых различий.

Проведём A1,2/B-тест

In [24]:
```
z_test(246&247, 248, 0.05)
```

```
MainScreenAppear p-значение: 0.3387114076159288
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

OffersScreenAppear p-значение: 0.21442476639710506
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

CartScreenAppear p-значение: 0.08067367598823139
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

PaymentScreenSuccessful p-значение: 0.21693033984516674
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Аналогично происходит и с объединенной контрольной группой и экспериментальной группой, статистически значимые различия снова отсутствуют.

Вывод: При иследовании воронки продаж выяснилось, что при переходе от главной страницы к странице предложений теряется больше всего пользователей (38%), а при переходе от корзины к оплате товара теряется всего 2% от всех пользователей, по итогу к оплате приходят 48% пользователей.

Было проведено 16 проверок (12 с уровнем статистической значимости 0,05, а 4 с 0,01), из которых следует, что между всеми группами нет статистически значимой разницы. Получается, что использование новых шрифтов никак не повлияло на желание пользователей совершать первые покупки.

Также хочется отметить, что использование поправок для снижения вероятности ложнопозитивного результата при множественном тестировании гипотез не требуется, так как p-значения достаточно велики во всех случаях.

В целом эксперимент можно признать успешным, новый шрифт не повлиял на поведение пользователей в приложении.