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."

Оглавление

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

Шаг 1. Откроем файл с данными и изучим общую информацию

Загрузим датасет `logs_exp`, выведем общую информацию и первые десять строк на экран.

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

In [3]:
```
logs_exp.info()
```

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

In [4]:
```
logs_exp.head(10)
```

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
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

Таблица `logs_exp` содержит следующие столбцы:

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

К оглавлению

Шаг 2. Подготовим данные

Дататасет состоит из 4 столбцов и 244126 строк. Названия столбцов одностилевые, не стану их заменять.

Проверим на дубликаты.

In [5]:
```
logs_exp.duplicated().sum()
```

Out[5]:
```
413
```

Дубликаты составляют меньше 0,2% датасета, удалим их.

In [6]:
```
logs_exp = logs_exp.drop_duplicates()
```

Пропуски отсутствуют. Изменим формат поля `ExpId` с `int64` на `int16`.

In [7]:
```
logs_exp['ExpId'] = logs_exp['ExpId'].astype(np.int16())
```

Добавим столбец даты и времени, а также отдельный столбец дат методами `to_datetime` и `astype`.

In [8]:
```
logs_exp['Datetime'] = pd.to_datetime(logs_exp['EventTimestamp'], unit='s')
logs_exp['Date'] = logs_exp['Datetime'].astype('datetime64[D]')
logs_exp.head()
```

Out[8]:

	EventName	DeviceIDHash	EventTimestamp	ExpId	Datetime	Date
0	MainScreenAppear	4575588528974610257	1564029816	246	2019-07-25 04:43:36	2019-07-25
1	MainScreenAppear	7416695313311560658	1564053102	246	2019-07-25 11:11:42	2019-07-25
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019-07-25
3	CartScreenAppear	3518123091307005509	1564054127	248	2019-07-25 11:28:47	2019-07-25
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248	2019-07-25 11:48:42	2019-07-25

Вывод: дататасет состоит из 4 столбцов и 244126 строк. Названия столбцов одностилевые, не были заменены. Удалены 413 пропусков (0,2% датасета). Изменён формат поля `ExpId` с `int64` на `int16`. Добавлены столбцы даты и времени.

К оглавлению

Шаг 3. Изучим и проверим данные

Посчитаем, сколько всего событий в логе.

In [9]:
```
logs_exp['EventName'].unique()
```

Out[9]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)
```

In [10]:
```
logs_exp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243713 entries, 0 to 244125
Data columns (total 6 columns):
EventName         243713 non-null object
DeviceIDHash      243713 non-null int64
EventTimestamp    243713 non-null int64
ExpId             243713 non-null int16
Datetime          243713 non-null datetime64[ns]
Date              243713 non-null datetime64[ns]
dtypes: datetime64[ns](2), int16(1), int64(2), object(1)
memory usage: 11.6+ MB
```

Вывод: в датасете 243713 события, уникальных событий всего пять.

Посчитаем, сколько всего пользователей в логе.

In [11]:
```
len(logs_exp['DeviceIDHash'].unique())
```

Out[11]:
```
7551
```

Вывод: в логе всего 7551 пользователь.

Сгруппируем таблицу `logs_exp` по идентификаторам пользователей и посчитаем количество событий. Сохраним в переменную `user_eventnames`.

In [12]:
```
user_eventnames = logs_exp.groupby('DeviceIDHash').agg({'EventName': 'count'})
user_eventnames.head()
```

Out[12]:

	EventName
DeviceIDHash	
6888746892508752	1
6909561520679493	5
6922444491712477	47
7435777799948366	6
7702139951469979	137

Посчитаем, сколько в среднем событий приходится на пользователя.

In [13]:
```
user_eventnames.mean()
```

Out[13]:
```
EventName    32.275593
dtype: float64
```

Вывод: в среднем на пользователя приходится 32 события.

Исследуем количество пользовательских событий методом `describe()`.

In [14]:
```
user_eventnames.describe()
```

Out[14]:

	EventName
count	7551.000000
mean	32.275593
std	65.154219
min	1.000000
25%	9.000000
50%	20.000000
75%	37.000000
max	2307.000000

Найдём моду.

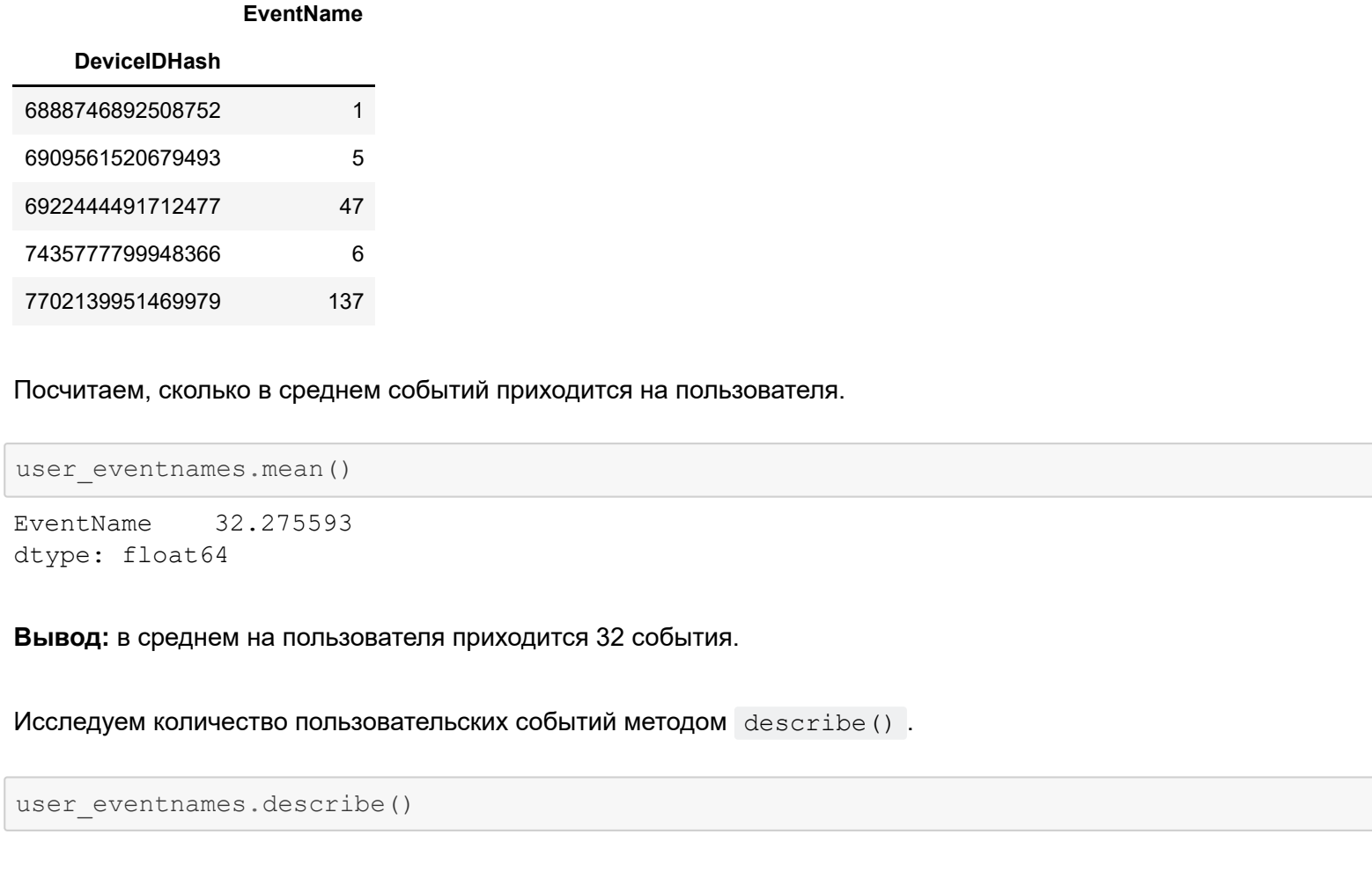
In [15]:
```
user_eventnames.mode()
```

Out[15]:

	EventName
0	5

Построим гистограмму, ограничив события на одного пользователя значением 101.

In [16]:
```
user_eventnames[user_eventnames['EventName'] < 101].hist(bins=100, figsize=(12, 7))
plt.grid(True)
ax = plt.gca()
ax.set_xlabel('Пользовательские события')
ax.set_ylabel('Количество событий')
plt.title('Гистограмма пользовательских событий');
```



Вывод: чаще всего на пользователя приходится 5 событий. Распределение Пуассона.

Найдём максимальную и минимальную дату.

In [17]:
```
logs_exp['Date'].min()
```

Out[17]:
```
Timestamp('2019-07-25 00:00:00')
```

In [18]:
```
logs_exp['Date'].max()
```

Out[18]:
```
Timestamp('2019-08-07 00:00:00')
```

Сгруппируем таблицу `logs_exp` по дате, посчитаем количество пользовательских событий, сбросим индексы методом `reset_index()` и сохраним в переменную `logs_exp_date`. Преобразуем столбец `Date` в формат строки.

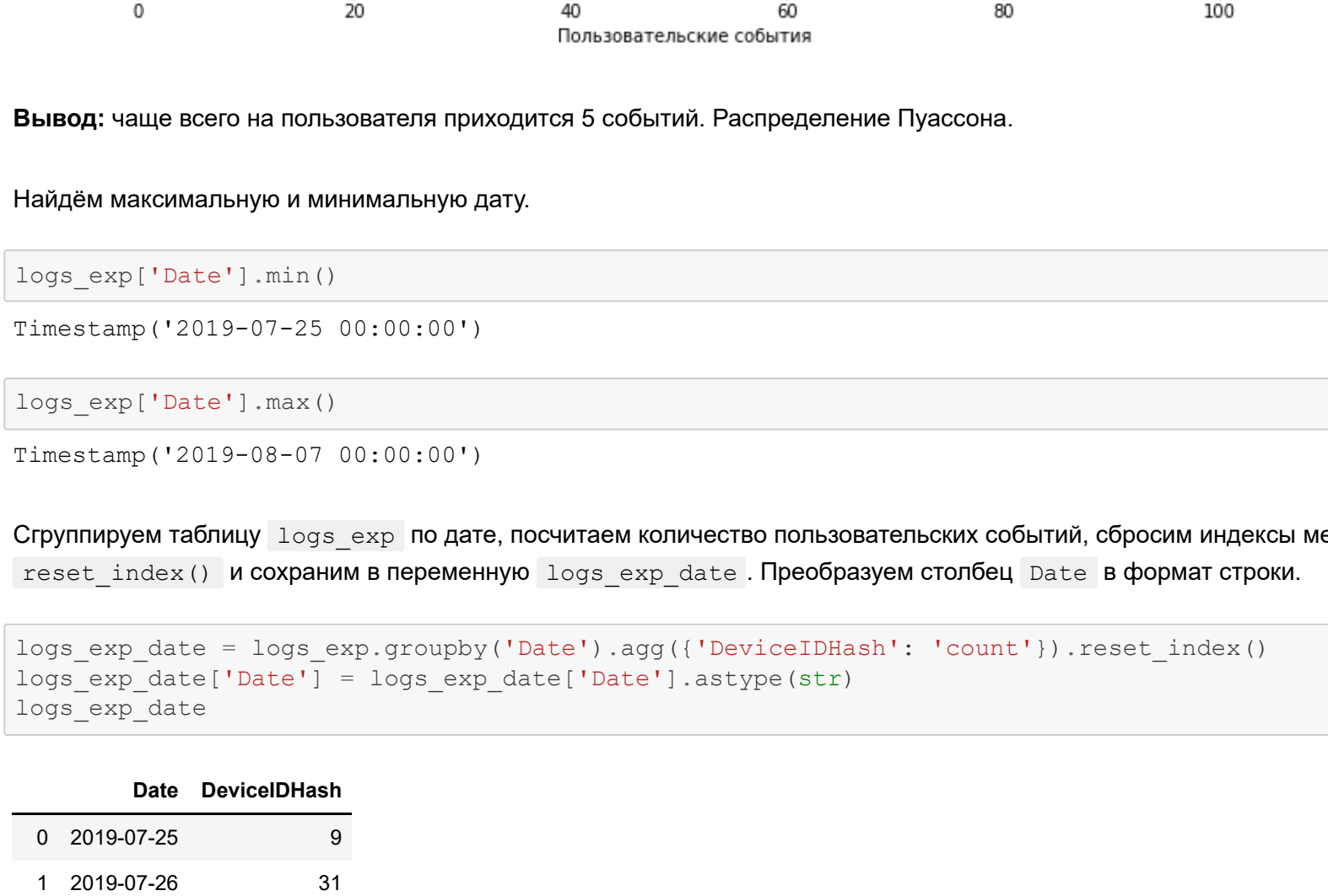
In [19]:
```
logs_exp_date = logs_exp.groupby('Date').agg({'DeviceIDHash': 'count'}).reset_index()
logs_exp_date['Date'] = logs_exp_date['Date'].astype(str)
logs_exp_date
```

Out[19]:

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

Построим гистограмму.

In [20]:
```
plt.figure(figsize=(12, 7))
plt.bar(logs_exp_date['Date'], logs_exp_date['DeviceIDHash'])
plt.grid(True)
ax = plt.gca()
ax.set_xlabel('Дата')
ax.set_ylabel('Количество пользовательских событий')
ax.set_xticklabels(logs_exp_date['Date'], rotation = 90, verticalalignment = 'top')
plt.title('Гистограмма пользовательских событий по дате');
```



Отбросим июльские данные. Сохраним результат в переменную `logs_exp_filtered`.

In [21]:
```
logs_exp_filtered = logs_exp[logs_exp['Date'] >= '2019-08-01']
```

Вывод: данные полны с 1-го августа 2019.

Посчитаем количество отброшенных событий и пользователей при помощи метода `shape`.

In [22]:
```
logs_exp.shape[0] - logs_exp_filtered.shape[0]
```

Out[22]:
```
2826
```

Вывод: количество отброшенных событий и пользователей составляет 1% данных.

Посчитаем количество пользователей из всех трёх экспериментальных групп в отброшенных данных и в полных.

In [23]:
```
logs_exp[logs_exp['Date'] < '2019-08-01'].groupby('ExpId').agg({'DeviceIDHash': ['count', 'nunique']})
```

Out[23]:

	DeviceIDHash	
	count	nunique
ExpId		
246	879	459
247	928	484
248	1019	508

In [24]:
```
logs_exp[logs_exp['Date'] < '2019-08-01']['DeviceIDHash'].drop_duplicates().reset_index(drop = True)
```

Out[24]:
```
0       4575588528974610257
1       7416695313311560658
2       3518123091307005509
3       6217807653094995999
4       8351860793733343758
                ...
1446    1879130487676858089
1447    5801359697252720810
1448     874398610899880804
1449    2238430701269950372
1450    2413191300976302222
Name: DeviceIDHash, Length: 1451, dtype: int64
```

In [25]:
```
logs_exp_filtered['DeviceIDHash'].drop_duplicates().reset_index(drop = True)
```

Out[25]:
```
0       3737462046622621720
1       1433840883824088890
2       4899590676214355127
3       1182179323890311443
4       4613461174774205834
                ...
7529    5811573131275421338
7530    5365227480683749189
7531    6660805781687343085
7532    7823752606740475984
7533    3454683894921357834
Name: DeviceIDHash, Length: 7534, dtype: int64
```

In [26]:
```
np.setdiff1d(logs_exp[logs_exp['Date'] < '2019-08-01']['DeviceIDHash'].drop_duplicates().reset_index(dr
op = True), \
                  logs_exp_filtered['DeviceIDHash'].drop_duplicates().reset_index(drop = True))
```

Out[26]:
```
array([  19151588103329050, 2141951348073372988, 2271703761362823687,
       2284191870135457477, 3551033798723561677, 3732627634014826395,
       4038665841336196550, 5114978858019388942, 5505110279185484837,
       6802370866140138742, 6991707383660138921, 7098993074549869464,
       7145262973832191693, 7539563624051296370, 8759002907241723567,
       8774423546023392346, 9192059464363071054])
```

In [28]:
```
logs_exp_filtered.groupby('ExpId').agg({'DeviceIDHash': ['count', 'nunique']})
```

Out[28]:

	DeviceIDHash	
	count	nunique
ExpId		
246	79302	2484
247	77022	2513
248	84563	2537

Найдём отношение.

In [30]:
```
logs_exp[logs_exp['Date'] < '2019-08-01'].groupby('ExpId').agg({'DeviceIDHash': ['count', 'nunique']})
/ \
logs_exp_filtered.groupby('ExpId').agg({'DeviceIDHash': ['count', 'nunique']})
```

Out[30]:

	DeviceIDHash	
	count	nunique
ExpId		
246	0.011084	0.184783
247	0.012049	0.192598
248	0.012050	0.200236

Вывод: в полных данных есть пользователи из всех трёх экспериментальных групп.

Вывод: в логе всего 7551 пользователь, чаще всего на пользователя приходится 5 событий. Данные полны с 1-го августа 2019, в полных данных есть пользователи из всех трёх экспериментальных групп.

К оглавлению

Шаг 4. Изучим воронку событий

Посмотрим, какие события есть в логах:

- `MainScreenAppear` — переход на главную страницу;
- `OffersScreenAppear` — переход на страницу оформления заказа;
- `CartScreenAppear` — переход на страницу оплаты картой;
- `PaymentScreenSuccessful` — переход на страницу успешной оплаты;
- `Tutorial` — обучение.

Посчитаем, как часто встречаются события методом `value_counts()`.

In [31]:
```
logs_exp_filtered['EventName'].value_counts()
```

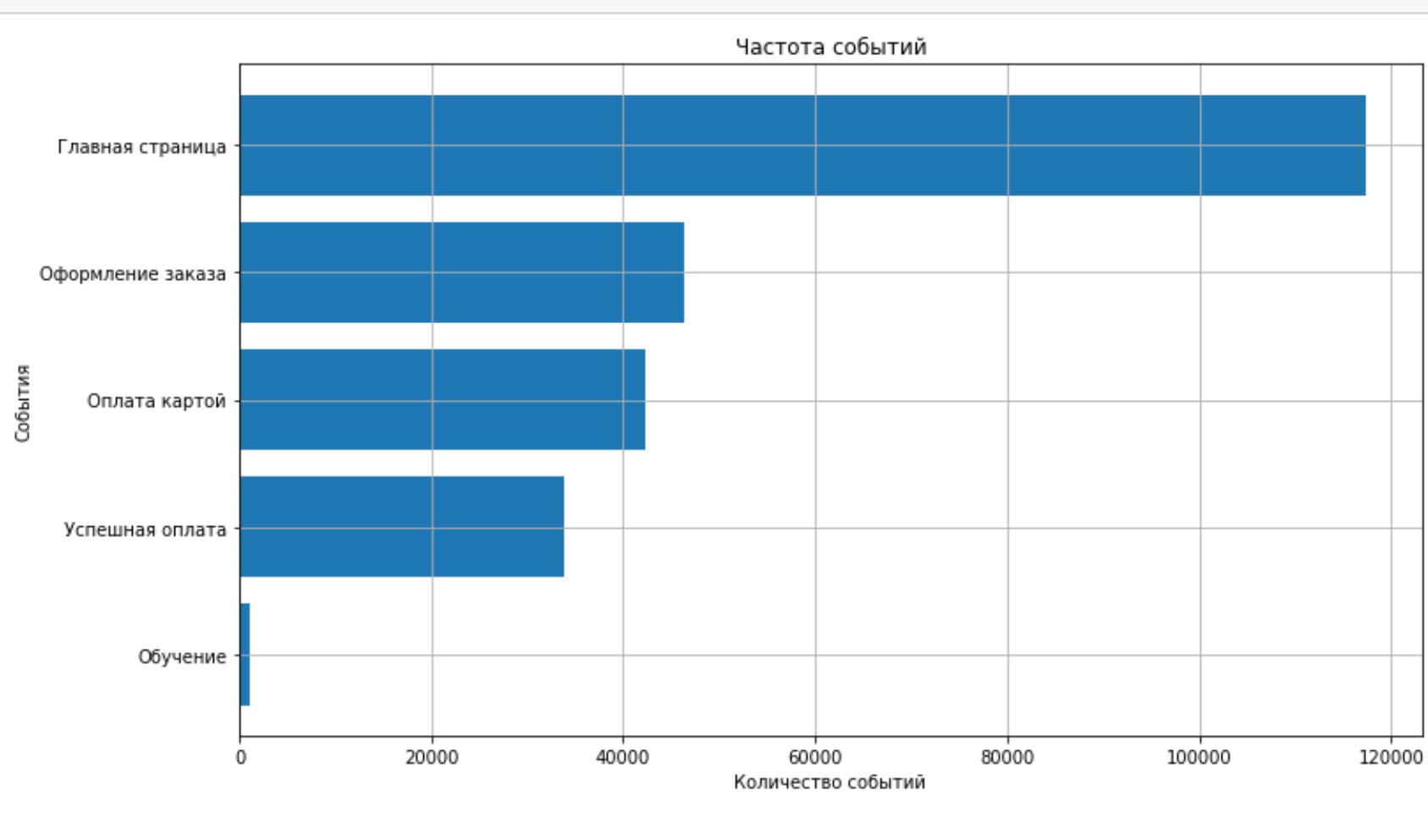
Out[31]:
```
MainScreenAppear           117328
OffersScreenAppear          46333
CartScreenAppear            42303
PaymentScreenSuccessful     33918
Tutorial                     1005
Name: EventName, dtype: int64
```

Создадим массив названий событий.

In [32]:
```
EventName = ['Обучение', 'Успешная оплата', 'Оплата картой', 'Оформление заказа', 'Главная страница']
```

Построим гистограмму.

In [33]:
```
plt.figure(figsize=(12, 7))
plt.barh(EventName, logs_exp_filtered['EventName'].value_counts(ascending=True))
ax = plt.gca()
plt.grid(True)
ax.set_xlabel('Количество событий')
ax.set_ylabel('События')
plt.title('Частота событий');
```



Аналогично найдём количество событий по числу пользователей. Сгруппируем таблицу `logs_exp_filtered` по событиям, посчитаем количество пользователей, отсортируем по убыванию методом `sort_values()`, сохраним в переменную `all_users`.

In [34]:
```
all_users = logs_exp_filtered.groupby('EventName').agg({'DeviceIDHash': 'count'}).sort_values(
    by='DeviceIDHash', ascending=False)
all_users
```

Out[34]:

	DeviceIDHash
EventName	
MainScreenAppear	117328
OffersScreenAppear	46333
CartScreenAppear	42303
PaymentScreenSuccessful	33918
Tutorial	1005

Посчитаем долю пользователей, которые хоть раз совершали событие. Сгруппируем таблицу `logs_exp_filtered` по событиям, посчитаем количество уникальных пользователей, отсортируем по убыванию методом `sort_values()`, сохраним в переменную `unique_users`.

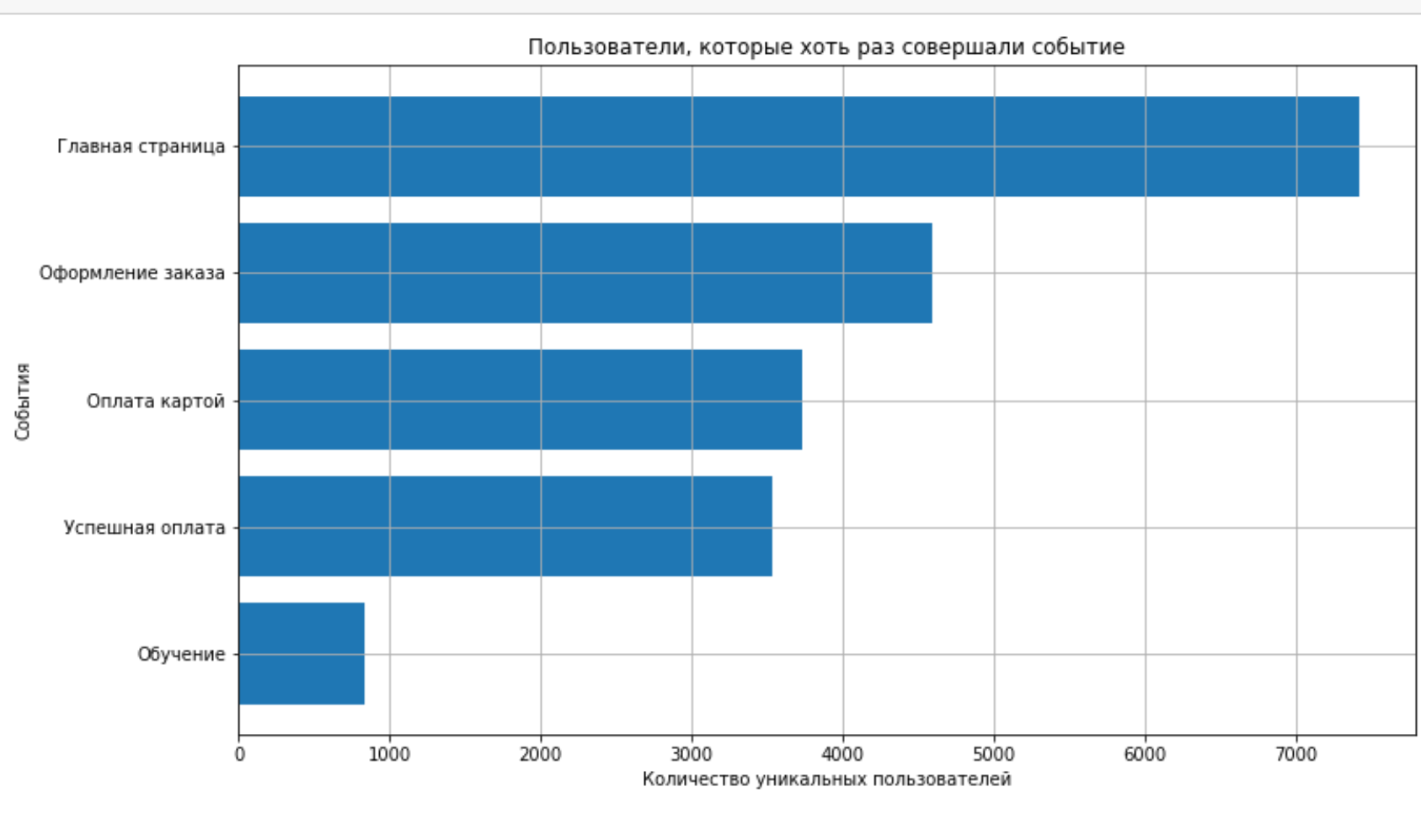
In [35]:
```
unique_users = logs_exp_filtered.groupby('EventName').agg({'DeviceIDHash': 'nunique'}).sort_values(
    by='DeviceIDHash', ascending=False)
unique_users
```

Out[35]:

	DeviceIDHash
EventName	
MainScreenAppear	7419
OffersScreenAppear	4593
CartScreenAppear	3734
PaymentScreenSuccessful	3539
Tutorial	840

Построим гистограмму.

In [36]:
```
plt.figure(figsize=(12, 7))
plt.barh(EventName, \
         unique_users.sort_values(by='DeviceIDHash', ascending=True)['DeviceIDHash'])
ax = plt.gca()
plt.grid(True)
ax.set_xlabel('Количество уникальных пользователей')
ax.set_ylabel('События')
plt.title('Пользователи, которые хоть раз совершали событие');
```



Вывод: пользователи чаще всего обновляют главную страницу, что вполне очевидно.

Предположительно события происходят в следующем порядке: переход на главную страницу → переход на страницу оформления заказа → переход на страницу оплаты картой → переход на страницу успешной оплаты. Обучение не встроено в цепочку и очень немногие (0,4%) пользователи его проходят. Удалим строку `Tutorial` из таблицы `all_users`.

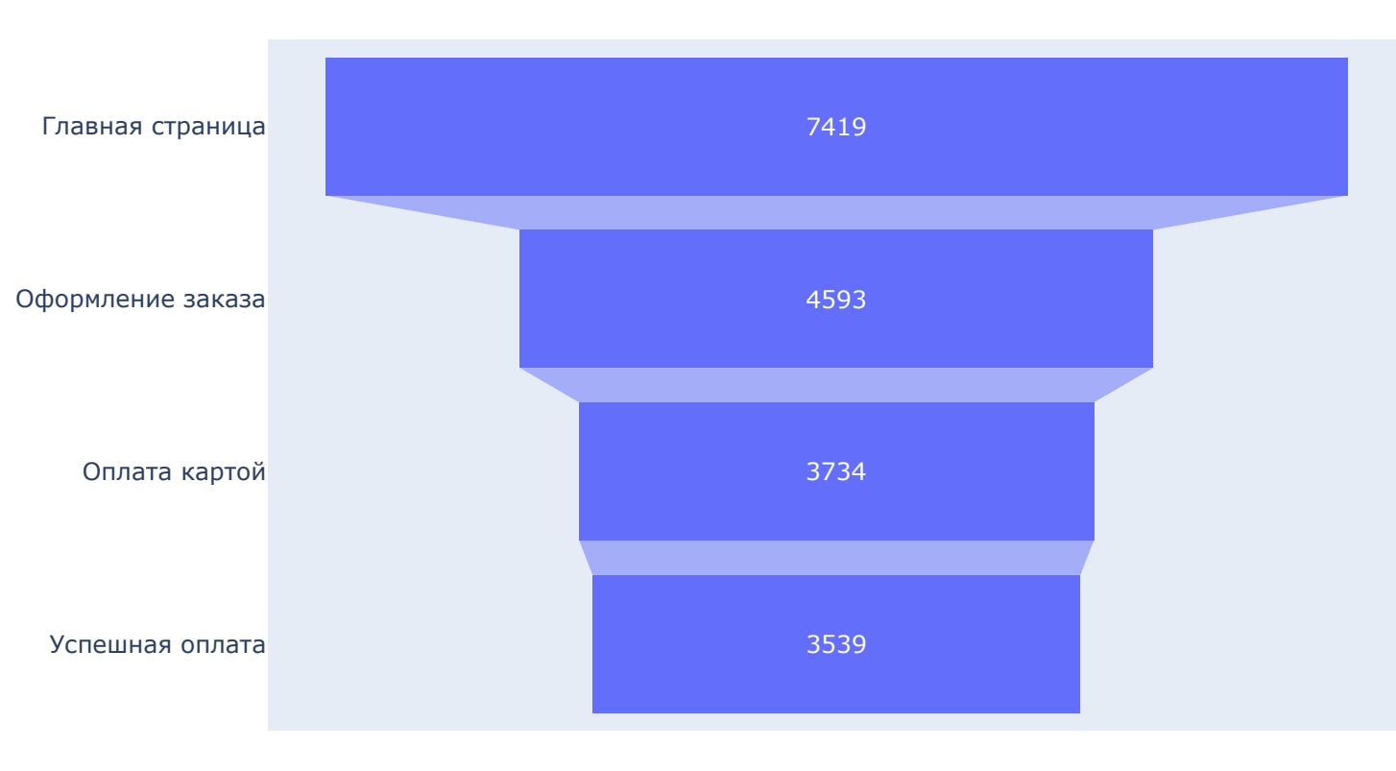
In [37]:
```
all_users = all_users.drop('Tutorial')
```

Изменим массив названий событий.

In [38]:
```
EventName = ['Главная страница', 'Оформление заказа', 'Оплата картой', 'Успешная оплата']
```

Построим воронку событий методом `Funnel()`.

In [39]:
```
fig = go.Figure(go.Funnel(
    y = EventName,
    x = unique_users['DeviceIDHash'],
    ))
fig.show()
```



Добавим в таблицу `unique_users` столбцы `Previous` и `Initial` с отформатированными значениями.

In [40]:
```
unique_users['Previous'] = unique_users['DeviceIDHash'] / unique_users['DeviceIDHash'].shift()
unique_users['Previous'] = unique_users['Previous'].fillna(1)
unique_users['Previous'] = unique_users['Previous'].apply(lambda x: "{0:.1f}%".format(x * 100))

unique_users['Initial'] = unique_users['DeviceIDHash'] / unique_users.loc['MainScreenAppear', 'DeviceID
Hash']
unique_users['Initial'] = unique_users['Initial'].apply(lambda x: "{0:.1f}%".format(x * 100))
unique_users = unique_users.drop('Tutorial')
unique_users
```

Out[40]:

	DeviceIDHash	Previous	Initial
EventName			
MainScreenAppear	7419	100.0%	100.0%
OffersScreenAppear	4593	61.9%	61.9%
CartScreenAppear	3734	81.3%	50.3%
PaymentScreenSuccessful	3539	94.8%	47.7%

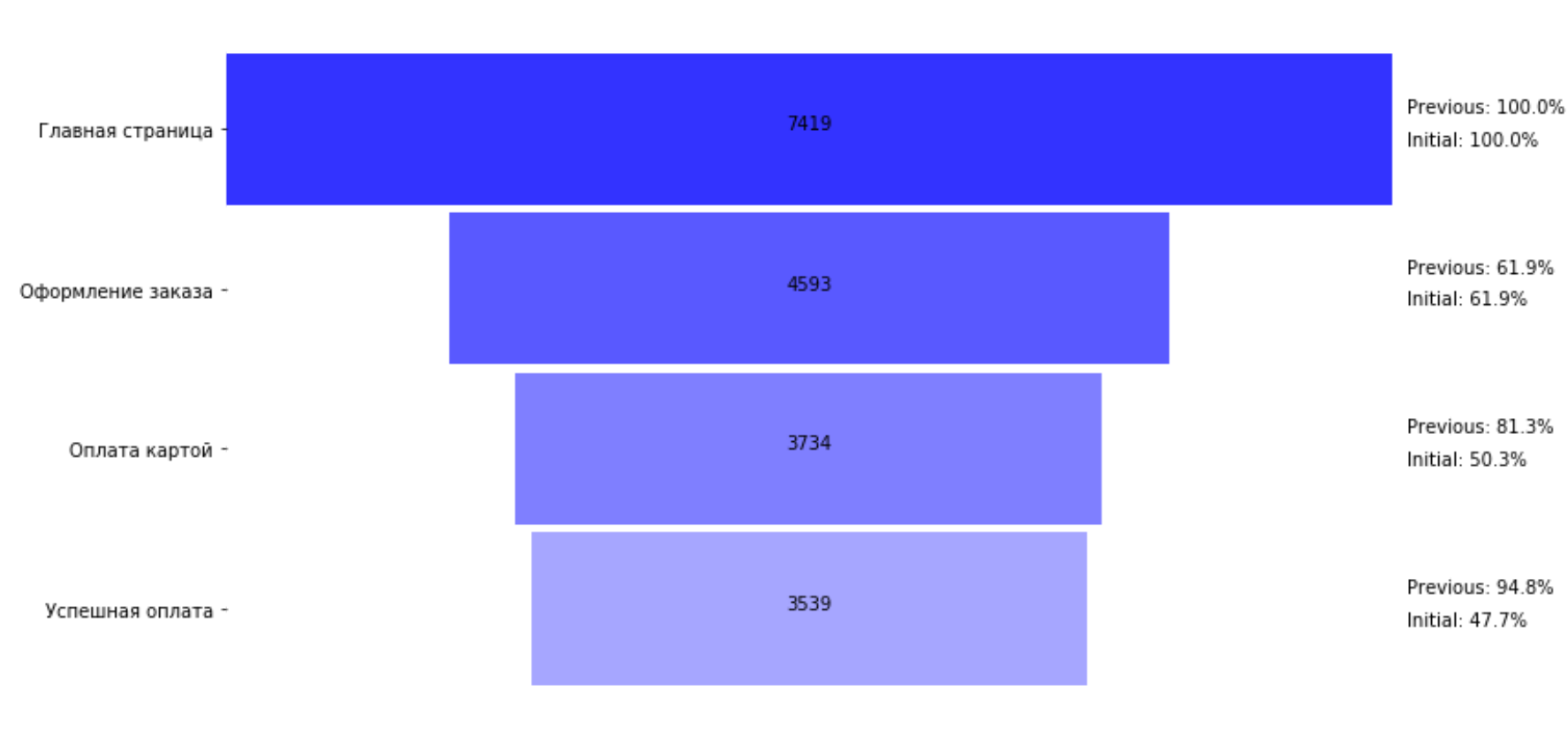
Построим воронку.

In [41]:
```
fig, ax = plt.subplots(figsize=(12,7));

EventNameEng = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessful'
]

for i in reversed(range(len(EventName))):
    ax.barh(EventName[i], unique_users.loc[EventNameEng[i], 'DeviceIDHash'], \
            height=0.95, align='center', left=(unique_users['DeviceIDHash'].max() - \
                                               unique_users.loc[EventNameEng[i], 'DeviceIDHash'])/2, \
            color='blue', alpha=(0.8-i*0.15))
    ax.annotate('Initial: ' + unique_users.loc[EventNameEng[i], 'Initial'], \
                xy=(unique_users['DeviceIDHash'].max() + 100, len(EventName) - i - 1.1), horizontalalig
nment='left')
    ax.annotate('Previous: ' + unique_users.loc[EventNameEng[i], 'Previous'], \
                xy=(unique_users['DeviceIDHash'].max() + 100, len(EventName) - i - 0.9), horizontalalig
nment='left')
    ax.annotate(unique_users.loc[EventNameEng[i], 'DeviceIDHash'], \
                xy=(unique_users['DeviceIDHash'].max()/2, len(EventName) - i-1), horizontalalignment='c
enter')

ax.get_xaxis().set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
fig.show();
```



Вывод: больше всего пользователей теряется на шаге оформления заказа — 38,1%. От первого события до оплаты доходит 47,7% пользователей.

К оглавлению

Шаг 5. Изучим результаты эксперимента

Посчитаем, сколько пользователей в каждой экспериментальной группе. Сгруппируем таблицу `logs_exp_filtered` по номерам экспериментов, посчитаем количество пользователей, сохраним в переменную `ExpId_users` и выведем на экран.

In [42]:
```
ExpId_users = logs_exp_filtered.groupby('ExpId').agg({'DeviceIDHash': 'nunique'})
ExpId_users
```

Out[42]:

	DeviceIDHash
ExpId	
246	2484
247	2513
248	2537

Создадим сводную таблицу `EventNumber`, сгруппируем данные по номерам эксперимента, в качестве столбцов — названия события, посчитаем количество пользователей и выведем на экран.

In [43]:
```
EventNumber = logs_exp_filtered.pivot_table(
    index='ExpId', columns='EventName', values='DeviceIDHash', aggfunc='nunique')
EventNumber
```

Out[43]:

EventName	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
ExpId					
246	1266	2450	1542	1200	278
247	1238	2476	1520	1158	283
248	1230	2493	1531	1181	279

Создадим функцию `z_test_func`, которая проверяет нулевую гипотезу по статистической значимости различий и выводит результат.

In [44]:

```
def z_test_func(first_exp, second_exp):
    """
    Функция получает на вход номера экспериментальных групп и название события, по котором идёт проверк
а нулевой гипотезы.
    Проверяет нулевую гипотезу по статистической значимости различий и выводит результат.

    """
    alpha = .05 # критический уровень статистической значимости

    EventName = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear', 'PaymentScreenSuccessfu
l']

    for i in EventName:
        # успехи
        successes = np.array([EventNumber.loc[first_exp, i], EventNumber.loc[second_exp, i]])

        # всего
        trials = np.array([ExpId_users.loc[first_exp, 'DeviceIDHash'], ExpId_users.loc[second_exp, 'Dev
iceIDHash']])

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

        # пропорция успехов во второй 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

        # считаем статистику в ст.отклонениях стандартного нормального распределения
        z_value = difference / math.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: ', "{0:.3f}".format(p_value))

        if (p_value < alpha):
            print("Отвергаем нулевую гипотезу: между долями выборок", first_exp, "и", second_exp, "по с
обытию", i, \
                  "есть значимая разница")
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок", firs
t_exp, "и", \
                  second_exp, "по событию", i, "разными")
```

Проверим, находят ли статистические критерии разницу между выборками 246 и 247. Проверим по количеству событий успешной оплаты.

1. Сформулируем нулевую гипотезу:

H₀: нет оснований считать разными доли пользователей по каждому событию в выборках 246 и 247.

Hₐ: есть значимая разница между долями пользователей по каждому событию в выборках 246 и 247.

In [45]:

```
z_test_func(246, 247)
```

```
p-значение:  0.757
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 247 по событию Ма
inScreenAppear разными
p-значение:  0.248
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 247 по событию Of
fersScreenAppear разными
p-значение:  0.229
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 247 по событию Ca
rtScreenAppear разными
p-значение:  0.115
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 247 по событию Pa
ymentScreenSuccessful разными
```

Вывод: разбиение на группы работает корректно. Доли пользователей по каждому событию в контрольных выборках одинаковые.

1. Сформулируем нулевую гипотезу:

H₀: нет оснований считать разными доли пользователей по каждому событию в выборках 246 и 248.

Hₐ: есть значимая разница между долями пользователей по каждому событию в выборках 246 и 248.

In [46]:

```
z_test_func(246, 248)
```

```
p-значение:  0.295
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 248 по событию Ма
inScreenAppear разными
p-значение:  0.208
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 248 по событию Of
fersScreenAppear разными
p-значение:  0.078
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 248 по событию Ca
rtScreenAppear разными
p-значение:  0.212
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246 и 248 по событию Pa
ymentScreenSuccessful разными
```

Вывод: Доли пользователей по каждому событию в контрольной выборке 246 и в экспериментальной выборке 248 одинаковые.

1. Сформулируем нулевую гипотезу:

H₀: нет оснований считать разными доли пользователей по каждому событию в выборках 247 и 248.

Hₐ: есть значимая разница между долями пользователей по каждому событию в выборках 247 и 248.

In [47]:

```
z_test_func(247, 248)
```

```
p-значение:  0.459
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 247 и 248 по событию Ма
inScreenAppear разными
p-значение:  0.920
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 247 и 248 по событию Of
fersScreenAppear разными
p-значение:  0.579
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 247 и 248 по событию Ca
rtScreenAppear разными
p-значение:  0.737
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 247 и 248 по событию Pa
ymentScreenSuccessful разными
```

Вывод: Доли пользователей по каждому событию в контрольной выборке 247 и в экспериментальной выборке 248 одинаковые.

Объединим контрольные группы 246, 247 и сравним с экспериментальной группой 248. Добавим в таблицу `ExpId_users` суммарное число пользователей выборок 246 и 247 под индексом 246247.

In [48]:

```
ExpId_users.loc[246247, 'DeviceIDHash'] = ExpId_users.loc[246, 'DeviceIDHash'] + ExpId_users.loc[247,
'DeviceIDHash']
ExpId_users
```

Out[48]:

ExpId	DeviceIDHash
246	2484.0
247	2513.0
248	2537.0
246247	4997.0

Добавим в таблицу `EventNumber` суммарное число пользователей выборок 246 и 247 по каждому событию под индексом 246247.

In [49]:

```
EventNumber.loc[246247, :] = EventNumber.loc[246, :] + EventNumber.loc[247, :]
EventNumber
```

Out[49]:

EventName	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
ExpId					
246	1266.0	2450.0	1542.0	1200.0	278.0
247	1238.0	2476.0	1520.0	1158.0	283.0
248	1230.0	2493.0	1531.0	1181.0	279.0
246247	2504.0	4926.0	3062.0	2358.0	561.0

1. Сформулируем нулевую гипотезу:

H₀: нет оснований считать разными доли пользователей по каждому событию в объединённой контрольной группе 246, 247 и экспериментальной группе 248.

Hₐ: есть значимая разница между долями пользователей по каждому событию в объединённой контрольной группе 246, 247 и экспериментальной группе 248.

In [50]:

```
z_test_func(246247, 248)
```

```
p-значение:  0.294
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246247 и 248 по событию
MainScreenAppear разными
p-значение:  0.434
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246247 и 248 по событию
OffersScreenAppear разными
p-значение:  0.182
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246247 и 248 по событию
CartScreenAppear разными
p-значение:  0.600
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли выборок 246247 и 248 по событию
PaymentScreenSuccessful разными
```

Вывод: доли пользователей по каждому событию в объединённой контрольной группе 246, 247 и экспериментальной группе 248 одинаковые.

Общий вывод: тест следует признать неуспешным. Доли пользователей по каждому событию в контрольных группах и экспериментальной группе одинаковые.

При проверке статистических гипотез выше был выбран уровень значимости 0.05. Было сделано 16 проверок статистических гипотез. Уровень значимости меньше 0.1, не будем его менять.

Проверим результаты эксперимента критерием Манна-Уитни.

Найдём количество уникальных пользователей и совершённых ими событий. Сгруппируем таблицу `logs_exp_filtered` по пользователям, посчитаем количество событий, сбросим индексы методом `reset_index()`, сохраним в переменную `logs_exp_filtered_users`.

In [51]:

```
logs_exp_filtered_users = logs_exp_filtered.groupby('DeviceIDHash').agg({'EventName': 'count'}).reset_i
ndex()
logs_exp_filtered_users
```

Out[51]:

	DeviceIDHash	EventName
0	6888746892508752	1
1	6909561520679493	5
2	6922444491712477	47
3	7435777799948366	6
4	7702139951469979	137
...	...	...
7529	9217594193087726423	9
7530	9219463515465815368	17
7531	9220879493065341500	6
7532	9221926045299980007	7
7533	9222603179720523844	59

7534 rows × 2 columns

Создадим сводную таблицу `logs_exp_filtered_246`, сгруппируем данные из выборки 246 по пользователям, посчитаем общее количество событий по каждому из них, сбросим индексы методом `reset_index()`, выведем на экран.

In [52]:

```
logs_exp_filtered_246 = logs_exp_filtered[logs_exp_filtered['ExpId'] == 246].\
pivot_table(index='DeviceIDHash', columns='EventName', values='ExpId', aggfunc='count').reset_index()
logs_exp_filtered_246.head(5)
```

Out[52]:

EventName	DeviceIDHash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
0	6888746892508752	NaN	1.0	NaN	NaN	NaN
1	6922444491712477	8.0	19.0	12.0	8.0	NaN
2	8740973466195562	NaN	8.0	1.0	NaN	NaN
3	12692216027168046	NaN	7.0	3.0	NaN	NaN
4	15708180189885246	38.0	27.0	38.0	23.0	NaN

Создадим сводную таблицу `logs_exp_filtered_247`, сгруппируем данные из выборки 247 по пользователям, посчитаем общее количество событий по каждому из них, сбросим индексы методом `reset_index()`, выведем на экран.

In [53]:

```
logs_exp_filtered_247 = logs_exp_filtered[logs_exp_filtered['ExpId'] == 247].\
pivot_table(index='DeviceIDHash', columns='EventName', values='ExpId', aggfunc='count').reset_index()
logs_exp_filtered_247.head(5)
```

Out[53]:

EventName	DeviceIDHash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
0	6909561520679493	1.0	2.0	1.0	1.0	NaN
1	7702139951469979	5.0	40.0	87.0	5.0	NaN
2	28534696657485531	6.0	11.0	7.0	5.0	NaN
3	28755862496905658	2.0	6.0	NaN	NaN	NaN
4	29094035245869447	2.0	10.0	10.0	1.0	1.0

Создадим сводную таблицу `logs_exp_filtered_248`, сгруппируем данные из выборки 248 по пользователям, посчитаем общее количество событий по каждому из них, сбросим индексы методом `reset_index()`, выведем на экран.

In [54]:

```
logs_exp_filtered_248 = logs_exp_filtered[logs_exp_filtered['ExpId'] == 248].\
pivot_table(index='DeviceIDHash', columns='EventName', values='ExpId', aggfunc='count').reset_index()
logs_exp_filtered_248.head(5)
```

Out[54]:

EventName	DeviceIDHash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
0	7435777799948366	NaN	6.0	NaN	NaN	NaN
1	8486814028069281	4.0	4.0	2.0	NaN	NaN
2	9841258664663090	8.0	10.0	8.0	2.0	1.0
3	26317307137967461	NaN	5.0	NaN	NaN	NaN
4	27899413433550864	NaN	9.0	NaN	NaN	NaN

Примечание: бывает, что количество событий успешной оплаты больше количества переходов на главную страницу. Неполные данные?

In [55]:

```
logs_exp_filtered_246[logs_exp_filtered_246['PaymentScreenSuccessful'] > logs_exp_filtered_246['MainScr
eenAppear']].head()
```

Out[55]:

EventName	DeviceIDHash	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
5	18658818197810381	7.0	3.0	63.0	6.0	NaN
24	91292479590032512	45.0	30.0	28.0	36.0	NaN
56	197027893265565660	931.0	93.0	107.0	865.0	NaN
71	263550556164486172	13.0	3.0	9.0	10.0	NaN
92	366093607806007780	35.0	13.0	7.0	32.0	NaN

Добавим к таблице уникальных пользователей `logs_exp_filtered_users` столбцы из выборки 246 по пользователям `logs_exp_filtered_246`. Добавим столбец `Total` — отношение числа успешной оплаты к общему числу событий, заменим пропуски нулями методом `fillna()`. Выведем пять строк на экран.

In [56]:

```
sample_246 = logs_exp_filtered_users.merge(logs_exp_filtered_246, on='DeviceIDHash', how='left')
sample_246['Total'] = (sample_246['PaymentScreenSuccessful'] / sample_246['EventName']).fillna(0)
sample_246.sample(5)
```

Out[56]:

	DeviceIDHash	EventName	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
1515	1902634890028391543	38	NaN	NaN	NaN	NaN	NaN
7074	8652127031019540255	191	NaN	NaN	NaN	NaN	NaN
5667	7033128542556787609	7	NaN	NaN	NaN	NaN	NaN
2917	3662441162486667021	25	NaN	NaN	NaN	NaN	NaN
3366	4231316200676149778	47	NaN	NaN	NaN	NaN	NaN

Добавим к таблице уникальных пользователей `logs_exp_filtered_users` столбцы из выборки 247 по пользователям `logs_exp_filtered_246`. Добавим столбец `Total` — отношение числа успешной оплаты к общему числу событий, заменим пропуски нулями методом `fillna()`. Выведем пять строк на экран.

In [57]:

```
sample_247 = logs_exp_filtered_users.merge(logs_exp_filtered_247, on='DeviceIDHash', how='left')
sample_247['Total'] = (sample_247['PaymentScreenSuccessful'] / sample_247['EventName']).fillna(0)
sample_247.sample(5)
```

Out[57]:

	DeviceIDHash	EventName	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
2647	3357371372909105330	9	NaN	NaN	NaN	NaN	NaN
187	250419112136655315	56	7.0	13.0	30.0	6.0	NaN
6552	8059769296163008703	13	NaN	13.0	NaN	NaN	NaN
3021	3794977356240836925	107	NaN	NaN	NaN	NaN	NaN
3088	3881891539348553720	17	1.0	8.0	7.0	1.0	NaN

Добавим к таблице уникальных пользователей `logs_exp_filtered_users` столбцы из выборки 248 по пользователям `logs_exp_filtered_246`. Добавим столбец `Total` — отношение числа успешной оплаты к общему числу событий, заменим пропуски нулями методом `fillna()`. Выведем пять строк на экран.

In [58]:

```
sample_248 = logs_exp_filtered_users.merge(logs_exp_filtered_248, on='DeviceIDHash', how='left')
sample_248['Total'] = (sample_248['PaymentScreenSuccessful'] / sample_248['EventName']).fillna(0)
sample_248.sample(5)
```

Out[58]:

	DeviceIDHash	EventName	CartScreenAppear	MainScreenAppear	OffersScreenAppear	PaymentScreenSuccessful	Tutorial
6661	8183993768433290804	12	NaN	NaN	NaN	NaN	NaN
6220	7670816421888043072	3	NaN	3.0	NaN	NaN	NaN
2550	3226469711415915535	5	NaN	NaN	NaN	NaN	NaN
475	634894491679664239	31	NaN	NaN	NaN	NaN	NaN
2272	2862153300066037949	21	NaN	NaN	NaN	NaN	NaN

In [59]:

```
alpha = .05 # критический уровень статистической значимости
```

Сформулируем следующие гипотезы.

H₀: нет статистически значимой разницы в количестве событий успешной оплаты между выборками 246 и 247.

Hₐ: есть статистически значимая разница в количестве событий успешной оплаты между выборками 246 и 247.

In [60]:

```
results = st.mannwhitneyu(sample_246['Total'], \
                          sample_247['Total'])

print('p-значение: ', "{0:.3f}".format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу: разница статистически значима")
else:
    print("Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя")
```

```
p-значение:  0.141
Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя
```

Сформулируем следующие гипотезы.

H₀: нет статистически значимой разницы в количестве событий успешной оплаты между выборками 246 и 248.

Hₐ: есть статистически значимая разница в количестве событий успешной оплаты между выборками 246 и 248.

In [61]:

```
results = st.mannwhitneyu(sample_246['Total'], \
                          sample_248['Total'])

print('p-значение: ', "{0:.3f}".format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу: разница статистически значима")
else:
    print("Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя")
```

```
p-значение:  0.379
Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя
```

Сформулируем следующие гипотезы.

H₀: нет статистически значимой разницы в количестве событий успешной оплаты между выборками 247 и 248.

Hₐ: есть статистически значимая разница в количестве событий успешной оплаты между выборками 247 и 248.

In [62]:

```
results = st.mannwhitneyu(sample_247['Total'], \
                          sample_248['Total'])

print('p-значение: ', "{0:.3f}".format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу: разница статистически значима")
else:
    print("Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя")
```

```
p-значение:  0.225
Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя
```

Вывод: тест следует признать неуспешным. Изменение шрифта не повлияло на количество событий успешной оплаты.

К оглавлению

Выводы

Тест следует признать неуспешным. Доли пользователей по каждому событию в контрольных группах и экспериментальной группе одинаковые. Изменение шрифта не повлияло на количество событий успешной оплаты.

В датасете 243713 события, уникальных событий всего пять:

- `MainScreenAppear` — переход на главную страницу;
- `OffersScreenAppear` — переход на страницу оформления заказа;
- `CartScreenAppear` — переход на страницу оплаты картой;
- `PaymentScreenSuccessful` — переход на страницу успешной оплаты;
- `Tutorial` — обучение.

В логе всего 7551 пользователь, чаще всего на пользователя приходится 5 событий.

Данные полны с 1-го августа 2019, в полных данных есть пользователи из всех трёх экспериментальных групп. Количество отброшенных событий и пользователей составляет 1% данных.

Пользователи чаще всего обновляют главную страницу, что вполне очевидно. Больше всего пользователей теряется на шаге оформления заказа — 38,1%. От первого события до оплаты доходит 47,7% пользователей.

Разбиение на группы работает корректно.

К оглавлению