Проверка результатов AB-тестирования

Содержание

Цель исследования: на основе данных о новых пользователях, их активности, а также с учётом данных о маркетинговых активностях компании за 2020 год оценить результаты проведённого AB-теста новой системы рекомендаций для пользователей.

Задачи исследования:

1. Оценить корректность проведения теста.
2. Проанализировать результаты теста.

Техническое задание на AB-тест:

- название теста: `recommender_system_test`;
- группы: A — контрольная, B — новая платёжная воронка;
- дата запуска: 2020-12-07;
- дата остановки набора новых пользователей: 2020-12-21;
- дата остановки: 2021-01-04;
- аудитория: 15% новых пользователей из региона EU;
- назначение теста: тестирование изменений, связанных с внедрением улучшенной рекомендательной системы;
- ожидаемое количество участников теста: 6000.
- ожидаемый эффект: за 14 дней с момента регистрации пользователи покажут улучшение каждой метрики не менее, чем на 10%:
  - конверсии в просмотр карточек товаров — событие `product_page`,
  - просмотры корзины — `product_cart`,
  - покупки — `purchase`.

Описание имеющихся в наличии данных:

1. **`final_ab_new_users.csv`** — пользователи, зарегистрировавшиеся с 7 по 21 декабря 2020 года.

Структура файла:

- `user_id` — идентификатор пользователя;
- `first_date` — дата регистрации;
- `region` — регион пользователя;
- `device` — устройство, с которого происходила регистрация.

1. **`final_ab_participants.csv`** — таблица участников тестов.

Структура файла:

- `user_id` — идентификатор пользователя;
- `ab_test` — название теста;
- `group` — группа пользователя.

1. **`final_ab_events.csv`** — действия новых пользователей в период с 7 декабря 2020 по 4 января 2021 года.

Структура файла:

- `user_id` — идентификатор пользователя;
- `event_dt` — дата и время покупки;
- `event_name` — тип события;
- `details` — дополнительные данные о событии. Например, для покупок, `purchase,` в этом поле хранится стоимость покупки в долларах.

1. ** `ab_project_marketing_events.csv` — календарь маркетинговых событий на 2020 год.

Структура файла:

- `name` — название маркетингового события;
- `regions` — регионы, в которых будет проводиться рекламная кампания;
- `start_dt` — дата начала кампании;
- `finish_dt` — дата завершения кампании.

Примерный план исследования:

1. Загрузить и изучить данные.
2. Оценить корректность проведения теста.
3. Провести EDA.
4. Оценить результаты A/B-тестирования.
5. Описать выводы по этапу исследовательского анализа данных и по проведённой оценке результатов A/B-тестирования. Сделать общее заключение о корректности проведения теста.

Библиотеки, используемые в исследовании

In [1]:
```
import os

# основные библиотеки DA
import pandas as pd
import numpy as np
import math as mth

# библиотеки работы с датой и временем
from datetime import timedelta
from datetime import datetime

# библиотеки визуализации
import plotly.express as px
from plotly import graph_objects as go

# статистические библиотеки
from scipy import stats as st
```

Загрузка и обзор данных

Определение функций загрузки и обзора данных

In [2]:
```
# определение функции загрузки данных
# ====================================================
# на вход подаётся:
#     file_name - имя файла
# на выходе - датафрейм с загруженными данными
# в случае ошибки при загрузке файла бросается искоючение
# ValueError
# ====================================================
def open_file(file_name, sep=','):
    pth1 = '/datasets/' + file_name             # яндексовский путь
    pth2 = os.path.join('datasets', file_name)  # мой путь

    if os.path.exists(pth1):
        return pd.read_csv(pth1, sep=sep)
    elif os.path.exists(pth2):
        return pd.read_csv(pth2, sep=sep)

    # ни один путь не доступен
    raise ValueError("ERROR: Neither Yandex nor local path is reacable...")
```

In [3]:
```
# определение функции обзора данных
# ================================================
# на вход подаётся датафрейм df
# на выходе:
#     - 10 случайных строк df
#     - информация df.info()
#     - количество явных дубликатов в строках df
#     - процент пропусков данных в столбцах df
# ================================================
def data_observe(df):
    row_num = 5    # количество отображаемых строк таблицы

    print('Размерность данных (row, col):', df.shape)
    print('============================\n')

    print('Произвольные строки таблицы:')
    print('============================')
    if len(df) >= row_num:
        display(df.sample(row_num))
    else:
        display(df)

    print('\nИнформация о таблице:')
    print('=====================')
    df.info()

    print('\nКоличество явных дубликатов в таблице:')
    print('======================================')
    print(df.duplicated().sum())

    print('\nПроцент пропусков в столбцах:')
    print('=============================')
    display(pd.DataFrame(
                round((df.isna().mean()*100),2), columns=['NaNs, %'])
                .sort_values(by='NaNs, %', ascending=False
```

```
            )
        .style.format('{:.2f}')
        .background_gradient('coolwarm')
    )
```

Обзор файла `final_ab_new_users.csv`

Откроем и изучим содержимое файла `final_ab_new_users.csv`:

In [4]:
```
try:
    new_users = open_file('final_ab_new_users.csv', sep=',')
    data_observe(new_users)
except ValueError as err:
    print(err)
```

```
Размерность данных (row, col): (61733, 4)
============================

Произвольные строки таблицы:
============================
```

	user_id	first_date	region	device
32400	FB08394C20381E32	2020-12-23	N.America	Mac
36179	94780B0323120EE4	2020-12-10	N.America	PC
32676	74F16DCF9FE79A89	2020-12-23	EU	Android
30168	1378122034154 6B3	2020-12-16	EU	Android
21234	22B322EF0FED6148	2020-12-15	EU	PC

```
Информация о таблице:
====================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61733 entries, 0 to 61732
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   user_id     61733 non-null  object
 1   first_date  61733 non-null  object
 2   region      61733 non-null  object
 3   device      61733 non-null  object
dtypes: object(4)
memory usage: 1.9+ MB

Количество явных дубликатов в таблице:
======================================
0

Процент пропусков в столбцах:
=============================
```

	NaNs, %
user_id	0.00
first_date	0.00
region	0.00
device	0.00

Итак, в таблице новых пользователей, зарегистрировавшихся с 7 по 21 декабря 2020 года, 61733 строки и 4 столбца. Столбцы поименованы в хорошем стиле snake_case, все имеют тип `object`.

В таблице отсутствуют явные дубликаты и пропуски.

Для дальнейшей работы столбец `'first_date'` целесообразно привести к типу `date`.

Обзор файла `final_ab_participants.csv`

Откроем и изучим содержимое файла `final_ab_participants.csv`:

In [5]:
```
try:
    participants = open_file('final_ab_participants.csv', sep=',')
    data_observe(participants)
except ValueError as err:
    print(err)
```

```
Размерность данных (row, col): (18268, 3)
============================

Произвольные строки таблицы:
============================
```

	user_id	group	ab_test
11574	3EE718562551ECDD	B	interface_eu_test
11317	CA640B1407337D93	B	interface_eu_test
10881	5B6CAF9FF9207A0D	A	interface_eu_test
3115	9D7EACF1BBB179BC	B	recommender_system_test
10698	75654D851F8C8F59	A	interface_eu_test

```
Информация о таблице:
====================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18268 entries, 0 to 18267
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   user_id  18268 non-null  object
 1   group    18268 non-null  object
 2   ab_test  18268 non-null  object
dtypes: object(3)
memory usage: 428.3+ KB

Количество явных дубликатов в таблице:
======================================
0

Процент пропусков в столбцах:
=============================
```

	NaNs, %
user_id	0.00
group	0.00
ab_test	0.00

В таблице участников AB-тестов 18268 строк и 3 столбца. Столбцы поименованы в хорошем стиле snake_case, все имеют тип `object`.

В таблице отсутствуют явные дубликаты и пропуски.

Приведение типов не требуется.

Обзор файла `final_ab_events.csv`

Откроем и изучим содержимое файла `final_ab_events.csv`:

In [6]:
```
try:
    events = open_file('final_ab_events.csv', sep=',')
    data_observe(events)
except ValueError as err:
    print(err)
```

```
Размерность данных (row, col): (440317, 4)
==========================

Произвольные строки таблицы:
==========================
```

	user_id	event_dt	event_name	details
74147	06F6F2EC08558DC0	2020-12-12 15:43:52	product_cart	NaN
131856	7C14AF59F1C215C0	2020-12-08 09:06:49	product_page	NaN
436079	CB649BD92A6AC43E	2020-12-29 15:51:46	login	NaN
255937	39FB64068022CECD	2020-12-08 10:08:55	login	NaN
42919	203773F2A53233A8	2020-12-22 17:47:33	purchase	4.99

```
Информация о таблице:
====================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440317 entries, 0 to 440316
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   user_id     440317 non-null  object 
 1   event_dt    440317 non-null  object 
 2   event_name  440317 non-null  object 
 3   details     62740 non-null   float64
dtypes: float64(1), object(3)
memory usage: 13.4+ MB

Количество явных дубликатов в таблице:
======================================
0

Процент пропусков в столбцах:
=============================
```

	NaNs, %
details	85.75
user_id	0.00
event_dt	0.00
event_name	0.00

Итак, в логе действий новых пользователей в период с 7 декабря 2020 по 4 января 2021 года 440317 строк и 4 столбца. Столбцы поименованы в хорошем стиле snake_case, все, за исключением `'details'` имеют тип `object`.

В таблице отсутствуют явные дубликаты.

Столбец `'details'` имеет тип `float64` и содержит 85.75% пропусков. Согласно описанию данных, это дополнительные данные о событии - необязательное поле. Заполнение пропусков не требуется.

Для дальнейшей работы столбец `'event_dt'` целесообразно привести к типу `datetime`.

Обзор файла `ab_project_marketing_events.csv`

Откроем и изучим содержимое файла `ab_project_marketing_events.csv`:

In [7]:
```
try:
```

```
    marketing_events = open_file('ab_project_marketing_events.csv', sep=',')
    data_observe(marketing_events)
except ValueError as err:
    print(err)
```

```
Размерность данных (row, col): (14, 4)
==========================

Произвольные строки таблицы:
==========================
```

	name	regions	start_dt	finish_dt
8	International Women's Day Promo	EU, CIS, APAC	2020-03-08	2020-03-10
7	Labor day (May 1st) Ads Campaign	EU, CIS, APAC	2020-05-01	2020-05-03
2	St. Patric's Day Promo	EU, N.America	2020-03-17	2020-03-19
13	Chinese Moon Festival	APAC	2020-10-01	2020-10-07
9	Victory Day CIS (May 9th) Event	CIS	2020-05-09	2020-05-11

```
Информация о таблице:
====================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   name       14 non-null     object
 1   regions    14 non-null     object
 2   start_dt   14 non-null     object
 3   finish_dt  14 non-null     object
dtypes: object(4)
memory usage: 576.0+ bytes

Количество явных дубликатов в таблице:
======================================
0

Процент пропусков в столбцах:
============================
```

	NaNs, %
name	0.00
regions	0.00
start_dt	0.00
finish_dt	0.00

В календаре маркетинговых событий на 2020 год 14 строк и 4 столбца. Столбцы поименованы в хорошем стиле snake_case, все имеют тип `object`.

В таблице отсутствуют явные дубликаты и пропуски.

Для дальнейшей работы столбцы `'start_dt'` и `'finish_dt'` целесообразно привести к типу `date`.

Выводы

Предварительно данные выглядят полными. В них отсутствуют явные дубликаты и пропуски данных в основных столбцах.

Столбец `'details'` таблицы `events` содержит 85.75% пропусков. Согласно описанию данных, это дополнительные данные о событии - необязательное поле. Заполнение пропусков не требуется.

В таблицах `new_users`, `events` и `marketing_events` столбцы, отвечающие за дату и время, целесообразно привести к соответствующему типу - `date` и `datetime`.

Предварительная обработка данных

На этапе обзора данных мы установили, что некоторые столбцы таблиц `new_users`, `events` и `marketing_events` целесообразно привести к типу `date` или `datetime`.

In [8]:
```
# приведём дату регистрации пользователей
new_users.first_date = pd.to_datetime(new_users.first_date)
print(new_users.first_date.dtype)

# приведём дату и время событий
events.event_dt = pd.to_datetime(events.event_dt)
print(events.event_dt.dtype)

# приведём дату и время начала и окончания маркетинговых событий
marketing_events.start_dt = pd.to_datetime(marketing_events.start_dt)
marketing_events.finish_dt = pd.to_datetime(marketing_events.finish_dt)
print(marketing_events.start_dt.dtype)
print(marketing_events.finish_dt.dtype)
```

```
datetime64[ns]
datetime64[ns]
datetime64[ns]
datetime64[ns]
```

Выводы

Типы даты и времени откорректированы во всех таблицах.

Перейдём к исследованию данных.

Исследовательский анализ данных (EDA)

Изучение состава значений столбцов

Посмотрим внимательнее на значения в таблице `new_users`:

In [9]:
```
print('Количество привлечённых уникальных пользователей:',
      new_users.user_id.nunique())
print('Количество уникальных дней привлечения:',
      new_users.first_date.nunique())
print('Дни, в которые приходили пользователи: с',
      new_users.first_date.min(), 'по', new_users.first_date.max())
print('Количество уникальных регионов привлечения:',
      new_users.region.nunique())
print('Список регионов:',
      new_users.region.unique())
print('Количество уникальных названий устройств:',
      new_users.device.nunique())
print('Список устройств:',
      new_users.device.unique())
```

```
Количество привлечённых уникальных пользователей: 61733
Количество уникальных дней привлечения: 17
Дни, в которые приходили пользователи: с 2020-12-07 00:00:00 по 2020-12-23 00:00:00
Количество уникальных регионов привлечения: 4
Список регионов: ['EU' 'N.America' 'APAC' 'CIS']
Количество уникальных названий устройств: 4
Список устройств: ['PC' 'Android' 'iPhone' 'Mac']
```

Итак, за 17 дней (с 7 по 23 декабря 2020 года) из 4 регионов ('EU' 'N.America' 'APAC' 'CIS') пришли 61733 новых пользователей, регистрировавшихся с устройств 'PC' 'Android' 'iPhone' 'Mac'.

Количество уникальных привлечённых пользователей совпадает с общим количеством привлечённых пользователей, следовательно, нет неявных дубликатов и сбои в системе регистрации пользователей не зафиксированы.

Отметим, что **данные расходятся с требованиями технического задания**: привлечение пользователей должно было закончиться 21 декабря 2020 года.

Тем не менее, выделим целевой регион (EU) и снова оценим количество привлечённых пользователей и даты привлечения:

In [10]:
```
print('Количество привлечённых уникальных пользователей:',
      new_users.query('region == "EU"').user_id.nunique())
print('Количество уникальных дней привлечения:',
      new_users.query('region == "EU"').first_date.nunique())
print('Дни, в которые приходили пользователи: с',
      new_users.first_date.min(), 'по', new_users.first_date.max())
print('Список регионов:',
      new_users.query('region == "EU"').region.unique())
```

```
Количество привлечённых уникальных пользователей: 46270
Количество уникальных дней привлечения: 17
Дни, в которые приходили пользователи: с 2020-12-07 00:00:00 по 2020-12-23 00:00:00
Список регионов: ['EU']
```

Вниманию маркетологов: В целевом регионе АВ-теста набор новых пользователей длился дольше предписанного техническим заданием.

В дальнейшем для соответствия ТЗ необходимо будет отсечь пользователей, привлечённых 22 и 23 декабря.

Взглянем на значения в таблице `participants`:

In [11]:
```
print('Количество уникальных участников тестов:',
      participants.user_id.nunique())
print('Количество уникальных групп:',
      participants.group.nunique())
print('Список групп:',
      participants.group.sort_values().unique())
print('Количество уникальных названий тестов:',
      participants.ab_test.nunique())
print('Список тестов:',
      participants.ab_test.sort_values().unique())
```

```
Количество уникальных участников тестов: 16666
Количество уникальных групп: 2
Список групп: ['A' 'B']
Количество уникальных названий тестов: 2
Список тестов: ['interface_eu_test' 'recommender_system_test']
```

Итак, в таблице зарегистрированы 16666 участников двух АВ-тестов (`interface_eu_test`, `recommender_system_test`), разделённых на контрольную и тестовую группы.

Для оценки тестов потребуется:

- разделить участников тестов;
- проверить, являются ли участники теста `recommender_system_test` участниками также и теста `interface_eu_test`;
- проверить, не пересекаются ли группы A и B теста `recommender_system_test`;
- проверить, соответствуют ли участники теста `recommender_system_test` требованиям технического задания (регион, % новых пользователей, ожидаемое количество участников).

Изучим значения в таблице `events`:

In [12]:
```
print('Количество уникальных пользователей:',
      events.user_id.nunique())
print('Количество уникальных названий событий:',
      events.event_name.nunique())
print('Список названий событий:',
      events.event_name.sort_values().unique())
print('Количество уникальных примечаний:',
      events.details.nunique())
print('Список примечаний:',
      events.details.sort_values().unique())
print('Начало логирования событий:',
      events.event_dt.min())
print('Конец логирования событий:',
      events.event_dt.max())
```

```
Количество уникальных пользователей: 58703
Количество уникальных названий событий: 4
Список названий событий: ['login' 'product_cart' 'product_page' 'purchase']
Количество уникальных примечаний: 4
Список примечаний: [  4.99   9.99  99.99 499.99    nan]
Начало логирования событий: 2020-12-07 00:00:33
Конец логирования событий: 2020-12-30 23:36:33
```

Итак, в таблице событий зарегистрированы действия 58703 пользователей.

Четыре зарегистрированных типа событий могут быть выстроены в логическую воронку: **login -> product_page -> product_cart -> purchase**.

Состав примечаний показывает, что пользователи делали покупки на 4.99, 9.99, 99.99, 499.99 долл.

События логировались с 7 по 30 декабря 2020 года включительно.

Проанализируем дату и время на дубликаты:

In [13]:
```
# посчитаем количество событий на каждую временную метку
event_dup = (
    events
    .groupby(by='event_dt')
    .agg({'user_id':['count', 'nunique'], 'event_name':['count', 'nunique']})
)

event_dup.columns = ['user_count', 'user_unique', 'event_count', 'event_unique']

# удалим метки с единственным событием
event_dup = (
    event_dup
    .query('user_count > 1 or event_count > 1')
    .sort_values(by=['user_count', 'event_count'], ascending=False)
)

event_dup
```

Out[13]:

event_dt	user_count	user_unique	event_count	event_unique
2020-12-14 18:54:55	10	4	10	4
2020-12-23 02:37:24	10	3	10	4
2020-12-13 06:00:54	9	3	9	4
2020-12-20 02:51:18	9	3	9	4
2020-12-21 21:14:13	9	4	9	4
...	...	...	...	...
2020-12-30 15:26:13	2	1	2	2
2020-12-30 16:47:56	2	1	2	2
2020-12-30 18:28:55	2	1	2	2
2020-12-30 20:41:37	2	1	2	2
2020-12-30 23:02:44	2	1	2	2

120353 rows × 4 columns

Итак, мы видим, что около 27% временных меток имеют дубликаты:

In [14]:
```
round(len(event_dup) * 100 / len(events), 2)
```

Out[14]:
```
27.33
```

При этом, часть дубликатов соответствует разным пользователям (это нормальная ситуация), а часть - одному и тому же уникальному пользователю. Это может свидетельствовать о сбое в системе логирования. Взглянем на дубликаты:

In [15]:
```
events[events.event_dt.duplicated()].sort_values(by=['user_id', 'event_dt']).head(10)
```

Out[15]:

	user_id	event_dt	event_name	details
157324	000199F1887AE5E6	2020-12-14 09:56:09	product_page	NaN
299212	000199F1887AE5E6	2020-12-14 09:56:09	login	NaN
81751	000199F1887AE5E6	2020-12-15 07:22:56	product_cart	NaN
163536	000199F1887AE5E6	2020-12-15 07:22:56	product_page	NaN
308595	000199F1887AE5E6	2020-12-15 07:22:56	login	NaN
97349	000199F1887AE5E6	2020-12-20 06:36:35	product_cart	NaN
355840	000199F1887AE5E6	2020-12-20 06:36:35	login	NaN
367197	000199F1887AE5E6	2020-12-21 02:11:23	login	NaN
202344	000199F1887AE5E6	2020-12-21 02:11:24	product_page	NaN
385742	0002499E372175C7	2020-12-22 03:49:52	login	NaN

Уже в начале отсортированной таблицы дубликатов видно, что для одного пользователя (`user_id = 000199F1887AE5E6`) в момент времени `2020-12-15 07:22:56` произошли одновременно 3 события: `login` , `product_page` , `product_cart` , что противоречит логике приложения даже с учётом быстродействия сетевого доступа.

Следует уведомить продуктовую команду о сбоях в системе логирования!

Отметим, однако, что посекундный учёт событий не входит в нашу задачу и не должен оказать существенного влияния на результаты АВ-теста. Поэтому продолжим изучение данных.

Рассмотрим данные в таблице `marketing_events` :

In [16]:
```
print('Общее количество маркетинговых акций:', len(marketing_events))
print('Количество уникальных названий акций:', marketing_events.name.nunique())
print('Количество уникальных регионов:', marketing_events.regions.nunique())
print('Состав регионов:', marketing_events.regions.unique())
print('Количество уникальных дат начала:', marketing_events.start_dt.nunique())
print('Количество уникальных дат начала:', marketing_events.finish_dt.nunique())
```

```
Общее количество маркетинговых акций: 14
Количество уникальных названий акций: 14
Количество уникальных регионов: 6
Состав регионов: ['EU, N.America' 'EU, CIS, APAC, N.America' 'N.America' 'APAC'
 'EU, CIS, APAC' 'CIS']
Количество уникальных дат начала: 14
Количество уникальных дат начала: 14
```

Итак, мы имеем 14 уникальных маркетинговых акций. Регионы проведения части акций заданы перечислением. ***Для проверки влияния маркетинговых действий на AB-тест необходимо выделить акции в регионе EU в период проведения теста.***

Оценка корректности проведения AB-теста

Проверим полученные данные об AB-тесте на соответствие техническому заданию:

1. Тест должен был проводиться с 7 декабря 2020 года по 4 января 2021 года. По факту в таблице событий зарегистрированы действия пользователей с 7 по 30 декабря 2020 года. Следовательно, ***тест был остановлен раньше времени***.
2. Набор новых пользователей должны были остановить 21 декабря 2020 года. По факту - 23 декабря 2020 г. ***Набор новых пользователей длился дольше требуемого.*** Пользователей, набранных 22 и 23 декабря, целесообразно удалить для приведения данных к требованиям ТЗ. Кроме того, целесообразно проверить, не попали ли эти "лишние" пользователи в тестовые группы.
3. Для проверки состава аудитории оценим её количество и качество:

In [17]:
```
# выделим пользователей из EU
eu_new_users = new_users.query('region == "EU"').copy()
print('Количество новых пользователей из EU:', len(eu_new_users))

# выделим из них пользователей, привлечённых по 21 декабря включительно
good_eu_new_users = eu_new_users.query('first_date < "2020-12-22"')
print('Количество пользователей из EU, привлечённых по 21 декабря:',
      len(good_eu_new_users), '(',
      round(len(good_eu_new_users) * 100 / len(eu_new_users), 2), '% )'
)

# выделим пользователей, привлечённых после 21 декабря
bad_eu_new_users = eu_new_users.query('first_date > "2020-12-21"')
print('Количество пользователей из EU, привлечённых по 21 декабря:',
      len(bad_eu_new_users), '(',
      round(len(bad_eu_new_users) * 100 / len(eu_new_users), 2), '% )'
)
```

```
Количество новых пользователей из EU: 46270
Количество пользователей из EU, привлечённых по 21 декабря: 42340 ( 91.51 % )
Количество пользователей из EU, привлечённых по 21 декабря: 3930 ( 8.49 % )
```

Итак, из 46270 привлечённых в EU пользователей 8.49% пришли в сервис после формального окончания набора новых пользователей. Проверим, попали ли они в тестовые группы:

In [18]:
```
(
    bad_eu_new_users
    .merge( # добавим информацию о тестах
        participants, on='user_id', how='left'
    )
    .dropna() # удалим пользователей без тестов
    .groupby(by=['ab_test', 'group'])
    .agg({'user_id':'nunique'})
)
```

Out[18]:

		user_id
ab_test	group	
interface_eu_test	A	489
	B	513

Излишне набранные пользователи попали в конкурирующий тест. ***Сообщить об этом его команде!*** Возможно, там срок набора дольше.

В дальнейшем мы можем работать только с пользователями, привлечёнными по 21 декабря включительно - `good_eu_new_users`.

Замечание: С учётом новых обстоятельств условие ТЗ об остановке набора новых пользователей для теста `recommender_system_test` 2020-12-21 соблюдено!

Проверим, в какие тесты и группы попали пользователи `good_eu_new_users`:

In [19]:
```
# выделим информацию по участникам тестов
eu_participants = (
    good_eu_new_users
    .merge( # добавим информацию о тестах
        participants, on='user_id', how='left'
    )
    .dropna() # удалим пользователей без тестов
)
# сгруппируем по тестам и группам
eu_participant_groups = (
    eu_participants
    .groupby(by=['ab_test', 'group'], as_index=False)
    .agg({'user_id':'nunique'})
    .merge( # добавим общее количество пользователей теста
        eu_participants
        .groupby(by=['ab_test'], as_index=False)
        .agg({'user_id':'nunique'})
        .rename(columns={'user_id':'total_users'}),
        on='ab_test', how='left'
    )
)
eu_participant_groups['percent'] = (
    (eu_participant_groups['user_id'] * 100 /
     eu_participant_groups['total_users'])
```

```
    .round(2)
)

eu_participant_groups
```

Out[19]:

	ab_test	group	user_id	total_users	percent
0	interface_eu_test	A	5342	10565	50.56
1	interface_eu_test	B	5223	10565	49.44
2	recommender_system_test	A	3634	6351	57.22
3	recommender_system_test	B	2717	6351	42.78

В целом, в конкурирующий тест попало больше людей - 10565 человек против 6351 в целевом тесте.

В контрольную группу A целевого теста `recommender_system_test` попало 3634 человека, в группу B - 2717 человек. В целом группы можно считать сбалансированными (57.22% участников против 42.78%).

Выделим участников целевого теста и проверим, как они распределены по группам и сколько из них могли принять участие в двух тестах:

In [20]:
```
# выделяем целевых участников
target_eu_participants = (eu_participants
                          .query('ab_test == "recommender_system_test"'))

# выясним, есть ли пересечение групп
display(
    target_eu_participants
    .groupby(by='user_id').agg({'group':'count'})
    .query('group > 1')
)

#  добавим для них заново информацию о тестах
target_eu_participants = (
    target_eu_participants
    # уберём лишние столбцы
    .drop(columns=['first_date', 'region', 'device', 'group', 'ab_test'])
    # добавим вновь информацию о тестах
    .merge(
        eu_participants, on='user_id', how='left'
    )
)

# выделим "двойных агентов"
double_test_perticipants = (
    target_eu_participants
    .groupby(by='user_id', as_index=False).agg({'ab_test':'count'})
    .query('ab_test > 1')
    .drop(columns=['ab_test']) # уберём лишние столбцы
    .merge( # добавим вновь информацию о тестах
        eu_participants, on='user_id', how='left'
    )
)

# добавим количество двойных участников
eu_participant_groups = (
    eu_participant_groups
    .merge(
        # сгруппируем по тестам и группам
        double_test_perticipants
        .groupby(by=['ab_test', 'group'], as_index=False)
        .agg({'user_id':'nunique'})
        .rename(columns={'user_id':'double_user_count'}),
        on=['ab_test', 'group'], how='left'
    )
)
# вычислим % от численности группы теста
eu_participant_groups['double_user_pct'] = (
    (eu_participant_groups['double_user_count'] * 100 /
     eu_participant_groups['user_id'])
    .round(2)
)
eu_participant_groups
```

user_id	group

Out[20]:

	ab_test	group	user_id	total_users	percent	double_user_count	double_user_pct
0	interface_eu_test	A	5342	10565	50.56	819	15.33
1	interface_eu_test	B	5223	10565	49.44	783	14.99
2	recommender_system_test	A	3634	6351	57.22	921	25.34
3	recommender_system_test	B	2717	6351	42.78	681	25.06

Итак:

- **пользователи групп A и B целевого теста не пересекаются**;
- **в два конкурирующих теста попало примерно по 25% участников из каждой группы целевого теста**.

Само по себе это не должно сильно повлиять на тест `recommender_system_test`.

Поэтому логичное, с одной стороны, удаление двойных участников из целевого теста, вероятно, не приведёт к существенным изменениям результатов, а кроме того, сократит количество пользователей в тесте в среднем на 25%, что не позволить выполнить требование ТЗ по общей численности теста (6000 человек).

Однако нужно проверить как распределились устройства для всех участников теста `recommender_system_test` и для двойных участников:

In [21]:
```
# снова удалим информацию о втором тесте
target_eu_participants = (eu_participants
                          .query('ab_test == "recommender_system_test"'))

# посчитаем количество устройств в группах
target_eu_participant_devices = (
```

```
    target_eu_participants
    .groupby(by=['group', 'device']).agg({'user_id':'count'})
    .rename(columns={'user_id':'participant_count'})
    # добавим количества устройств у двойных участников
    .join(
        double_test_perticipants
        .groupby(by=['group', 'device']).agg({'user_id':'count'})
        .rename(columns={'user_id':'double_participant_count'}),
        how='left'
    )
)
# вычислим %
target_eu_participant_devices['pct'] = (
    (target_eu_participant_devices['double_participant_count'] * 100 /
     target_eu_participant_devices['participant_count'])
    .round(2)
)

target_eu_participant_devices
```

Out[21]:

group	device	participant_count	double_participant_count	pct
A	Android	1590	766	48.18
	Mac	354	188	53.11
	PC	964	461	47.82
	iPhone	726	325	44.77
B	Android	1228	612	49.84
	Mac	250	174	69.60
	PC	657	381	57.99
	iPhone	582	297	51.03

Из полученной таблицы следует, что **по устройствам двойные пользователи распределены по группам не так равномерно, как по общему количеству**. Это может оказать некоторое воздействие на результат. Попробуем учесть это при оценке результатов AB-теста.

Итак, общее количество участников теста `recommender_system_test` равно 6351, что удовлетворяет требованиям технического задания с оговорками, сделанными выше.

1. Оценим, какую долю от общего количества привлечённых в регионе EU пользователей составляют участники теста:

In [22]:
```
len(target_eu_participants) / len(good_eu_new_users)
```

Out[22]:
```
0.15
```

Доля участников от привлечённых в EU пользователей составляет 15%, что удовлетворяет требованиям ТЗ.

Оценка соответствия проведённого AB-теста условиям тезнического задания завершена.

Выделим события, совершённые участниками целевого AB-теста:

In [23]:
```
target_eu_participant_events = (
    target_eu_participants
    # удалим лишние данные
    .drop(columns=['first_date', 'region', 'device', 'ab_test'])
    .merge(
        events, on='user_id', how='left'
    )
)
# проверим, у всех ли участников были события
target_eu_participant_events.event_name.isna().sum()
```

Out[23]:
```
2870
```

In [24]:
```
target_eu_participant_events.event_name.isna().sum() / len(target_eu_participants)
```

Out[24]:
```
0.451897339001732
```

Оказывается 2870 из 6351 участников теста (свыше 45%) не совершили за указанный период ни одного события!

Уведомить разработчиков: Налицо наличие серьёзной ошибки в системе логирования событий!

Это может повлиять на результаты теста, но придумать недостающие данные мы не можем. Посмотрим, как пользователи без событий распределились между группами теста и по устройствам:

In [25]:
```
# выделим пользователей без событий
target_eu_participant_no_events = (
    target_eu_participant_events[
        target_eu_participant_events['event_name'].isna()
    ]
)

# посчитаем по группам теста
target_eu_participant_no_events_groups = (
    target_eu_participant_no_events
    .groupby(by='group').agg({'user_id':'count'})
    .rename(columns={'user_id':'no_event_users'})
    .join(
        target_eu_participants.groupby(by='group').agg({'user_id':'count'}),
        how='left'
    )
    .rename(columns={'user_id':'total_users'})
)
# оценим % пользователей без событий
target_eu_participant_no_events_groups['pct'] = (
    (target_eu_participant_no_events_groups['no_event_users'] * 100 /
     target_eu_participant_no_events_groups['total_users'])
    .round(2)
)
```

```
# оценим количество пользователей с событиями
target_eu_participant_no_events_groups['delta'] = (
    (target_eu_participant_no_events_groups['total_users'] -
     target_eu_participant_no_events_groups['no_event_users'])
)
target_eu_participant_no_events_groups
```

Out[25]:

group	no_event_users	total_users	pct	delta
A	1030	3634	28.34	2604
B	1840	2717	67.72	877

Итак, мы имеем свыше 28% от численности группы A и свыше 67% от численности группы B без зарегистрированных в логе событий. **Такое большое количество "мёртвых душ" может серьёзно исказить метрики, поскольку они будут учтены в знаменателе целевых метрик, но не учтены в числителе. При этом, с учётом того, что тестовая группа и так меньше контрольной, существует риск занижения метрик в тестовой группе.**

В этой связи представляется целесообразным убрать таких пользователей из лога. При этом снизится мощность теста, однако сохранится статистическая значимость - в группе сохранится 877 пользователей с активностями, свыше 33% от численности группы A:

In [26]:
```
(
    target_eu_participant_no_events_groups.loc['B','delta'] /
    target_eu_participant_no_events_groups.loc['A','delta']
)
```

Out[26]:
```
0.33678955453149
```

Вероятная причина наличия "мёртвых душ" - ошибки и сбои в системе логирования действий пользователей. Для поиска и локализации ошибок командой определим дни, в которые были привлечены эти пользователи:

In [27]:
```
(
    target_eu_participant_no_events
    .drop(columns=['group', 'event_dt', 'event_name', 'details'])
    .merge(
        new_users, on='user_id', how='left'
    )
    .sort_values(by='first_date')
    ['first_date'].unique()
)
```

Out[27]:
```
array(['2020-12-07T00:00:00.000000000', '2020-12-08T00:00:00.000000000',
       '2020-12-09T00:00:00.000000000', '2020-12-10T00:00:00.000000000',
       '2020-12-11T00:00:00.000000000', '2020-12-12T00:00:00.000000000',
       '2020-12-13T00:00:00.000000000', '2020-12-14T00:00:00.000000000',
       '2020-12-15T00:00:00.000000000', '2020-12-16T00:00:00.000000000',
       '2020-12-17T00:00:00.000000000', '2020-12-18T00:00:00.000000000',
       '2020-12-19T00:00:00.000000000', '2020-12-20T00:00:00.000000000',
       '2020-12-21T00:00:00.000000000'], dtype='datetime64[ns]')
```

Очевидно, **сбои в логировании действий происходили на протяжении всего периода привлечения**.

Удалим "мёртвые души" из участников теста:

In [28]:
```
target_eu_participant_events = (
    target_eu_participant_events.dropna(axis=0, thresh=4)
)
target_eu_participant_events
```

Out[28]:

	user_id	group	event_dt	event_name	details
0	D72A72121175D8BE	A	2020-12-07 21:52:10	product_page	NaN
1	D72A72121175D8BE	A	2020-12-07 21:52:07	login	NaN
3	DD4352CDCF8C3D57	B	2020-12-07 15:32:54	product_page	NaN
4	DD4352CDCF8C3D57	B	2020-12-08 08:29:31	product_page	NaN
5	DD4352CDCF8C3D57	B	2020-12-10 18:18:27	product_page	NaN
...	...	...	...	...	...
26284	0416B34D35C8C8B8	A	2020-12-21 22:28:29	product_page	NaN
26285	0416B34D35C8C8B8	A	2020-12-24 09:12:51	product_page	NaN
26286	0416B34D35C8C8B8	A	2020-12-20 20:58:25	login	NaN
26287	0416B34D35C8C8B8	A	2020-12-21 22:28:29	login	NaN
26288	0416B34D35C8C8B8	A	2020-12-24 09:12:49	login	NaN

23420 rows × 5 columns

Для оценки возможного влияния маркетинговых активностей на результаты теста выделим те из них, которые проводились в регионе EU во время проведения теста. За дату окончания AB-теста теперь можно принять дату последнего события в таблице `target_eu_participant_events`. Условием, что акция могла повлиять на тест является тот факт, что она началась до конца теста и закончилась после его начала:

In [29]:
```
# начало теста по логам
ab_test_start_dt = target_eu_participant_events.event_dt.min()
# конец теста по логам
ab_test_end_dt = target_eu_participant_events.event_dt.max()
(
    marketing_events[marketing_events['regions'].str.contains('EU')]
    .query('start_dt <= @ab_test_end_dt and finish_dt >= @ab_test_start_dt')
)
```

Out[29]:

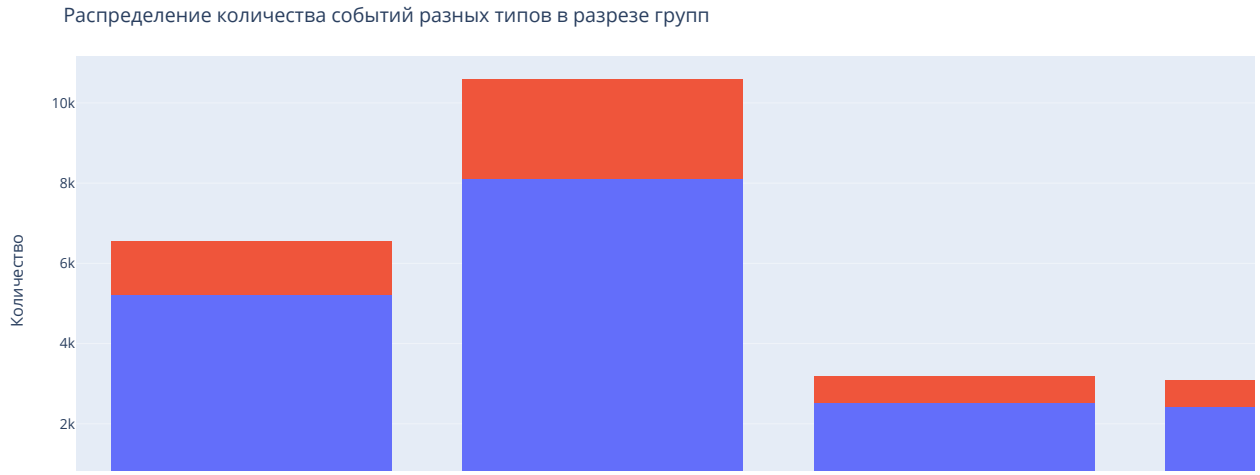
	name	regions	start_dt	finish_dt
0	Christmas&New Year Promo	EU, N.America	2020-12-25	2021-01-03

Итак, только **новогодняя маркетинговая акция из календаря могла повлиять на результаты теста**.

Исследование распределений и воронки

Проверим, одинаково ли распределено в выборках количество событий в разрезе групп:

In [30]:
```
# визуализируем распределение
fig = px.histogram(target_eu_participant_events, x="event_name", color="group")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение количества событий разных типов в разрезе групп',
    xaxis_title_text='События',
    yaxis_title_text='Количество'
)
fig.show()
```



Гистограмма распределения событий по типам в разрезе групп показывает, что события распределены между группами схожим образом.

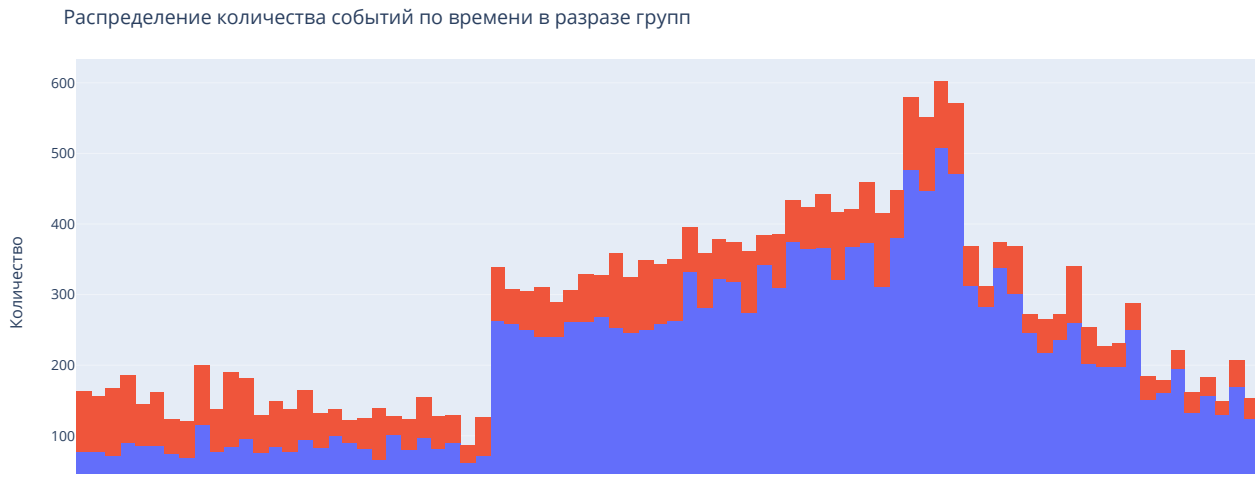
Посчитаем общее количество событий на пользователя для каждой из групп:

In [31]:
```
# посчитаем, сколько событий совершил каждый пользователь каждой группы
events_by_user = (
    target_eu_participant_events
    .groupby(by=['group', 'user_id'], as_index=False)
    .agg({'event_name':'count'})
    .rename(columns={'event_name':'event_count'})
)

# визуализируем распределение
fig = px.histogram(events_by_user, x="event_count", color="group")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение количества событий на пользователя в разрезе групп',
    xaxis_title_text='Количество событий',
    yaxis_title_text='Частота'
)
fig.show()
```

Гистограмма распределения количества событий на пользователя в разрезе групп показывает, что распределения для разных групп схожи, большая часть пользователей и в группе A и в группе B совершили от 2 до 12 событий за период теста, при этом распределение тестовой группы B чуть сильнее смещено влево.

Построим распределение количества событий по времени в разрезе групп:

In [32]:
```
# визуализируем распределение
fig = px.histogram(target_eu_participant_events, x="event_dt", color="group")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение количества событий по времени в разразе групп',
    xaxis_title_text='Дата и время',
    yaxis_title_text='Количество'
)
fig.show()
```



Мы видим, что количество событий по времени для групп A и B также распределено неодинаково:

- для группы A до 14 декабря количество событий было рапределено равномерно в районе 90, а 14 декабря наблюдается резкий скачок количества событий;
- В группе B события распределены равномерно в период набора новых пользователей, а по его завершении их количество равномерно убывает.

Визуализируем воронку событий. Ранее мы описывали стадии естественной для сервиса воронки следующим образом: **login -> product_page -> product_cart -> purchase**.

Отметим, что события в данной воронке упорядочены не лексикографически. Для правильного отображения воронки события нужно пронумеровать и отсортировать по номерам:

In [33]:
```
# зададим нумерацию событий в воронке
event_list = target_eu_participant_events.event_name.unique()
event_num_list = []

for evt in event_list:
    if evt == 'login':
        event_num_list.append(1)
    elif evt == 'product_page':
        event_num_list.append(2)
    elif evt == 'product_cart':
        event_num_list.append(3)
    elif evt == 'purchase':
        event_num_list.append(4)
    else:
        # исключительная ситуация, если в воронке прявятся новые события,
        # потребуется изменить код
        event_num_list.append(-1)

# соберём словарь нумерации для воронки
event_num = pd.DataFrame(dict(number=event_num_list, event_name=event_list))
event_num
```

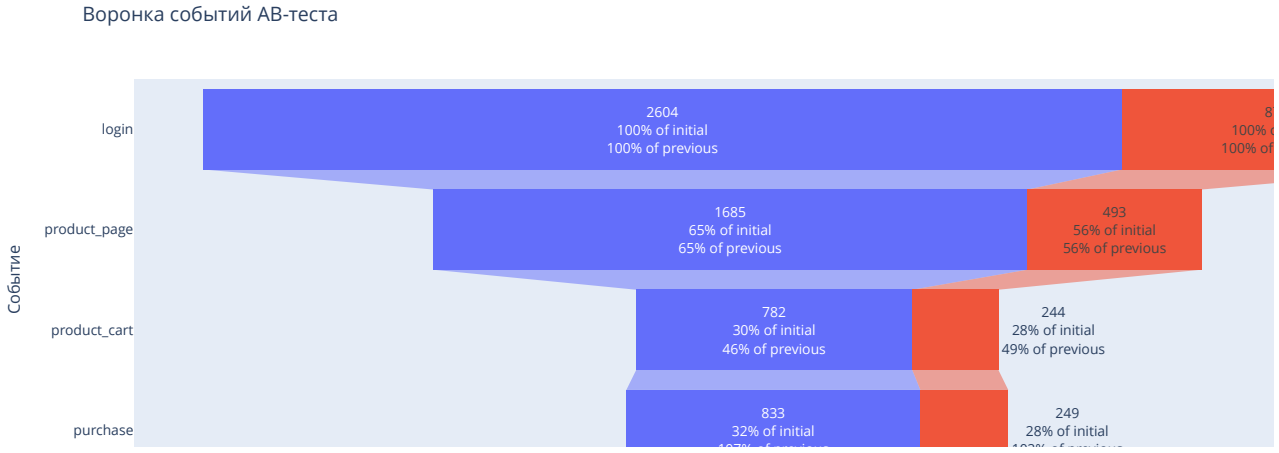
Out[33]:

	number	event_name
0	2	product_page
1	1	login
2	4	purchase
3	3	product_cart

In [34]:
```
# построим воронку
funnel = (
    target_eu_participant_events
    .groupby(by=['event_name', 'group'], as_index=False)
    .agg({'user_id':'nunique'})
    .merge(
        event_num, on='event_name', how='left'
    )
    .sort_values(by=['number', 'group'])
)
display(funnel)

# визуализируем воронку
fig = go.Figure()
groups = funnel.group.unique()

for g in groups:
    fig.add_trace(go.Funnel(
        name=g,
        y=list(funnel.query('group == @g')['event_name']),
        x=list(funnel.query('group == @g')['user_id']),
```

```
        textinfo = "value+percent initial+percent previous"
    ))

# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Воронка событий АВ-теста',
    yaxis_title_text='Событие'
)
fig.show()
```

	event_name	group	user_id	number
0	login	A	2604	1
1	login	B	877	1
4	product_page	A	1685	2
5	product_page	B	493	2
2	product_cart	A	782	3
3	product_cart	B	244	3
6	purchase	A	833	4
7	purchase	B	249	4



На основании построенной воронки событий опишем, как меняется конверсия в выборках на разных этапах:

1. Контрольная группа A:
    A. На этапе `login` представлено 2604 (100%) пользователей.
    B. На этап `product_page` перешли 1685 пользователей (65% от предыдущего этапа).
    C. На этап `product_cart` попали 782 человека (46% от предыдущего этапа и 30% от общего количества группы A).
    D. А вот покупку (`purchase`) совершили 833 пользователя (107% от предыдущего этапа и 32% от исходного количества).
2. Тестовая группа B:
    A. На этапе `login` представлено 877 (100%) пользователей.
    B. На этап `product_page` перешли 493 пользователя (56% от предыдущего этапа).
    C. На этап `product_cart` попали 244 человека (49% от предыдущего этапа и 28% от общего количества группы B).
    D. А вот покупку (`purchase`) совершили 249 пользователя (102% от предыдущего этапа и 28% от исходного количества).

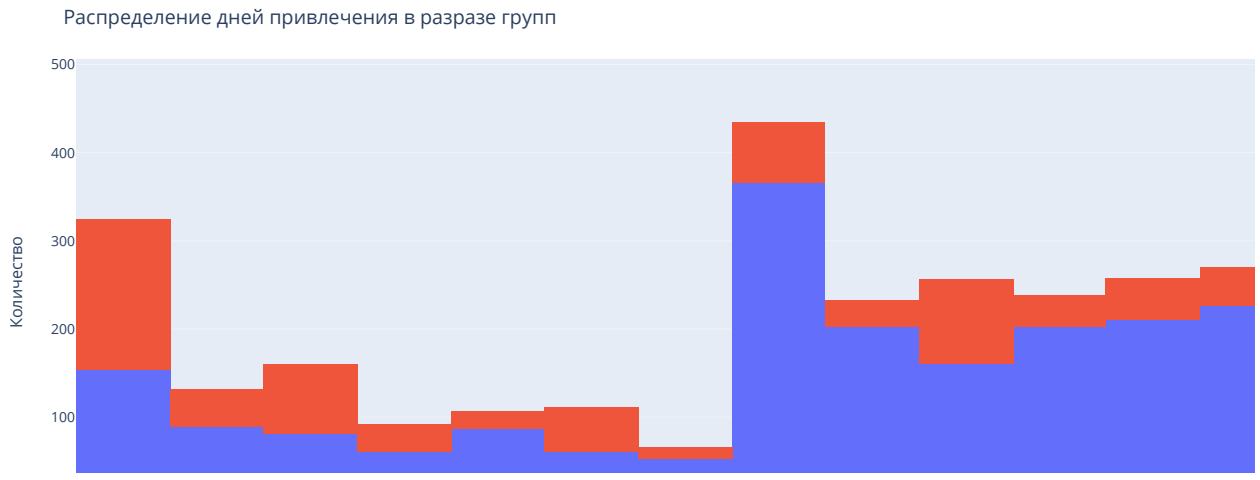
Из полученного сравнения можно заключить, что **в целом за период наблюдений конверсия тестовой группы B на каждом шаге воронки ниже по сравнению с контрольной группой A.**

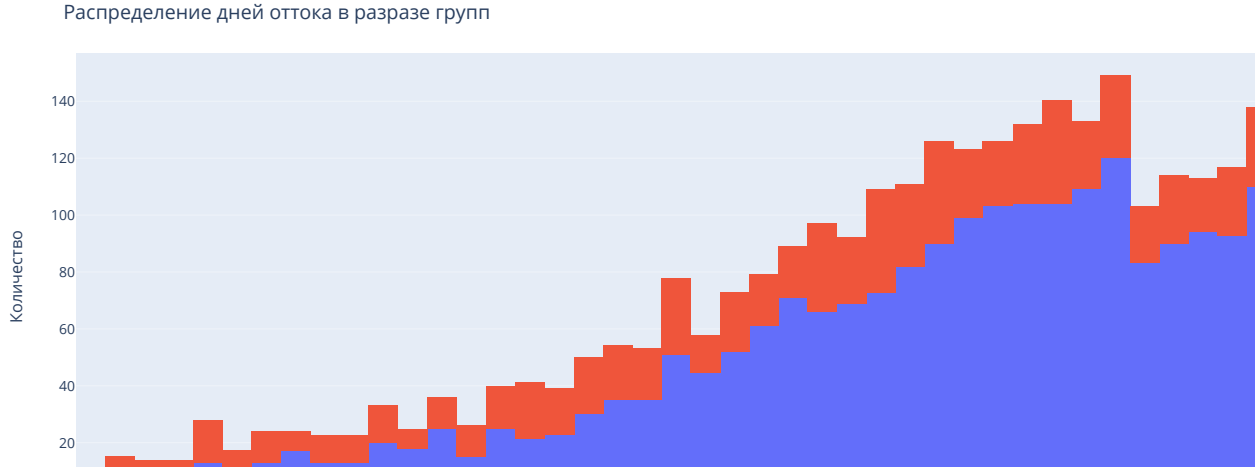
Более высокая конверсия покупок по сравнению с показами корзины объясняется, вероятно, наличием возможности быстрых покупок со страницы товара, минуя корзину.

Возможно, на такие низкие результаты тестовой группы оказало влияние удержание пользователей. Посмотрим, в какие даты приходили пользователи, и в какие прекращяли пользование сервисом.

In [35]:
```
# последние даты пользования и даты привлечения
target_retention = (
    target_eu_participant_events
    .groupby(by=['group', 'user_id'], as_index=False)
    .agg({'event_dt':'max'})
    .rename(columns={'event_dt':'last_event_dt'})
    .merge(
        new_users[['user_id', 'first_date']], on='user_id', how='left'
    )
)

# визуализируем привлечение
fig = px.histogram(target_retention, x="first_date", color="group")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение дней привлечения в разразе групп',
    xaxis_title_text='Дата',
    yaxis_title_text='Количество'
)
fig.show()

# визуализируем отток
fig = px.histogram(target_retention, x="last_event_dt", color="group")
```

```
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение дней оттока в разразе групп',
    xaxis_title_text='Дата и время',
    yaxis_title_text='Количество'
)
fig.show()
```





Сравнение гистограмм распределений дат привлечения и оттока показывают, что распределения привлечения и оттока между группами различаются:

- В контрольной группе А:
  1. Активное формирование группы началось на 3 неделе набора новых пользователей. **Это является недостатком системы отбора тестовых групп: распределение количества набранных участников тестирования во времени должно быть равномерным.**
  2. Отток участников тестирования начался ещё в период формирования группы, усиливался по мере набора участников теста и вышел на максимальные показатели по завершению набора.
- В тестовой группе В:
  1. Формирование группы и отток распределены во времени более равномерно, однако в распределении привлечения наблюдается некоторая цикличность, а в распределении оттока - смещение вправо, как и в случае с контрольной группой.
  2. Наибольшее количество участников были привлечены в первый день, ушли - по окончании периода привлечения.

Выводы

В процессе EDA нами были проведены оценка соответствия полученных данных АВ-тестирования техническому заданию, исследование распределений и воронки событий.

По результатам могут быть сделаны следующие выводы:

1. Исходные данные не в полной мере соответствуют ТЗ:
   - тест был остановлен раньше времени (30 декабря 2020 года вместо 4 января 2021 года);
   - наблюдается неравномерность распределения пользователей по устройствам;
   - свыше 28% от численности группы А и свыше 67% от численности группы В не имеют зарегистрированных в логе событий, их удаление из выборки нарушает требования ТЗ об общем количестве участников (6000) и доле от привлечённых из целевого региона (15%).

1. В ходе изучения распределений установлено:
   - события по типам распределены между группами схожим образом;

- количество событий на пользователя схоже распределено для групп A и B, большая часть пользователей в обеих группах совершили от 2 до 12 событий за период теста, при этом распределение тестовой группы B чуть сильнее смещено влево;
- количество событий по времени для групп A и B распределено неодинаково:
  - для группы A до 14 декабря количество событий было рапределено равномерно в районе 90, а 14 декабря наблюдается резкий скачок количества событий;
  - В группе B события распределены равномерно в период набора новых пользователей, а по его завершении их количество равномерно убывает.

1. Воронка событий выглядит следующим образом:
   A. Контрольная группа A:
      a. На этапе `login` представлено 2604 (100%) пользователей.
      b. На этап `product_page` перешли 1685 пользователей (65% от предыдущего этапа).
      c. На этап `product_cart` попали 782 человека (46% от предыдущего этапа и 30% от общего количества группы A).
      d. А вот покупку (`purchase`) совершили 833 пользователя (107% от предыдущего этапа и 32% от исходного количества).
   B. Тестовая группа B:
      a. На этапе `login` представлено 877 (100%) пользователей.
      b. На этап `product_page` перешли 493 пользователея (56% от предыдущего этапа).
      c. На этап `product_cart` попали 244 человека (49% от предыдущего этапа и 28% от общего количества группы B).
      d. А вот покупку (`purchase`) совершили 249 пользователя (102% от предыдущего этапа и 28% от исходного количества).

1. В целом за период наблюдений конверсия тестовой группы B на каждом шаге воронки ниже по сравнению с контрольной группой A.

1. Более высокая конверсия покупок по сравнению с показами корзины объясняется, вероятно, наличием возможности быстрых покупок со страницы товара, минуя корзину.

1. Распределения привлечения и оттока между группами различаются:

- В контрольной группе A:
  1. Активное формирование группы началось на 3 неделе набора новых пользователей. **Это является недостатком системы отбора тестовых групп: распределение количества набранных участников тестирования во времени должно быть равномерным.**
  2. Отток участников тестирования начался ещё в период формирования группы, усиливался по мере набора участников теста и вышел на максимальные показатели по завершению набора.
- В тестовой группе B:
  1. Формирование группы и отток распределены во времени более равномерно, однако в распределении привлечения наблюдается некоторая цикличность, а в распределении оттока - смещение вправо, как и в случае с контрольной группой.
  2. Наибольшее количество участников были привлечены в первый день, ушли - по окончании периода привлечения.

Кроме того, **выявлены сбои в системе логирования событий**, на которые указывают:

- часть событий одного пользователя, с высокой вероятностью разнесённых по времени, помечена одной временной меткой (например для пользователя user_id = 000199F1887AE5E6 в момент времени 2020-12-15 07:22:56 произошли одновременно 3 события: login, product_page и product_cart, что противоречит логике приложения даже с учётом быстродействия сетевого доступа);
- 2870 из 6351 участников теста (свыше 45%) не совершили за период его проведения ни одного события;
- сбои в логировании действий происходили, как минимум, на протяжении всего периода привлечения.

Отметим, что большое количество "мёртвых душ" может серьёзно исказить метрики и повлиять на результаты теста, поскольку они будут учтены в знаменателе целевых метрик, но не учтены в числителе. При этом, с учётом того, что тестовая группа и так меньше контрольной, существует риск занижения метрик в тестовой группе.

Оценка результатов AB-теста

Оценим соответствие достигнутых результатов AB-теста ожидаемому эффекту: за 14 дней *с момента регистрации* пользователи тестовой группы B покажут улучшение каждой метрики не менее, чем на 10%, по сравнению с контрольной группой A:

- конверсии в просмотр карточек товаров — событие `product_page` ,
- конверсии в просмотры корзины — `product_cart` ,
- конверсии в покупки — `purchase` .

Конверсии будем считать по формулам:

$$CR_{pp} = \frac{Users_{pp}}{Users_{total}},$$

$$CR_{pc} = \frac{Users_{pc}}{Users_{total}},$$

$$CR_{p} = \frac{Users_{p}}{Users_{total}},$$

где:

- CR_{pp} - конверсия в просмотр карточек товаров;
- CR_{pc} - конверсия в просмотры корзины;
- CR_{p} - конверсия в покупки;
- $Users_{pp}$ - количество пользователей группы, совершивших событие `product_page` за некоторый промежуток времени;
- $Users_{pc}$ - количество пользователей группы, совершивших событие `product_cart` за некоторый промежуток времени;
- $Users_{p}$ - количество пользователей группы, совершивших событие `purchase` за некоторый промежуток времени;
- $Users_{total}$ - общее количество пользователей в группе.

Для решения задачи нам необходимо:

- определить дневной лайфтайм каждого события для каждого пользователя,
- ограничить лог событий лайфтаймами с 0 по 13 (14 дней),
- построить воронку событий,
- посчитать соответствующие конверсии для групп A и B (CR_i^A и CR_i^B, $i \in \{A, B\}$),
- проверить равенства $\frac{CR_i^B}{CR_i^A} \geq 1.1$.

Построение и анализ воронки за 14 дней теста

1. Определим дневные лайфтаймы событий для каждого пользователя с момента регистрации:

In [36]:
```
# добавим даты привлечения
target_eu_participant_events = (
    target_eu_participant_events
    # добавим даты привлечения
    .merge(
        new_users[['user_id', 'first_date']],
        on='user_id', how='left'
    )
)

# посчитаем дневные лайфтаймы
target_eu_participant_events['lifetime'] = (
    target_eu_participant_events['event_dt'] -
    target_eu_participant_events['first_date']
).dt.days

target_eu_participant_events.sample(10)
```

Out[36]:

	user_id	group	event_dt	event_name	details	first_date	lifetime
4943	4D2B0FBB9E528D60	A	2020-12-15 09:18:40	product_page	NaN	2020-12-14	1
1511	1AA90C1AD5727610	B	2020-12-07 17:17:15	login	NaN	2020-12-07	0
13874	60E68A4F33951169	A	2020-12-20 00:59:28	login	NaN	2020-12-16	4
21336	6F70F06DD209676B	A	2020-12-26 06:11:19	product_page	NaN	2020-12-13	13
20087	72742C5F312A1FEC	B	2020-12-20 14:13:18	login	NaN	2020-12-19	1
4157	79913E8816E3DA5D	A	2020-12-17 12:11:29	product_cart	NaN	2020-12-14	3
3395	9DB76819BAB65ED6	A	2020-12-25 06:36:15	login	NaN	2020-12-14	11
8660	8EB70A4733EED0BA	A	2020-12-21 05:41:41	purchase	4.99	2020-12-08	13
15522	6687D5BCA6CD3141	A	2020-12-18 23:31:46	product_cart	NaN	2020-12-17	1
4865	B85F36F6515DAB84	A	2020-12-16 19:05:05	purchase	4.99	2020-12-14	2

Взглянем на распределения событий по лайфтаймам для каждой группы тестирования:

In [37]:
```
# визуализируем распределение событий группы A
fig = px.histogram(target_eu_participant_events.query('group == "A"'),
                   x="lifetime", color="event_name")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение событий группы A по лайфтаймам',
    xaxis_title_text='Лайфтайм',
    yaxis_title_text='Количество'
)
fig.show()
```

В группе A:

- событие `product_page` распределено по пуассоновскому закону, основная доля этих событий произошла в первые 17 дней с момента привлечения, единичные события происходили вплоть до 23 лайфтайма;
- событие `product_cart` распределено по пуассоновскому закону, основная доля этих событий произошла в первые 16 дней с момента привлечения, единичные события происходили вплоть до 21 лайфтайма;
- событие `purchase` распределено по пуассоновскому закону, основная доля этих событий произошла в первые 17 дней с момента привлечения, единичные события происходили вплоть до 21 лайфтайма.

In [38]:
```
# визуализируем распределение событий группы B
fig = px.histogram(target_eu_participant_events.query('group == "B"'),
                   x="lifetime", color="event_name")
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Распределение событий группы B по лайфтаймам',
    xaxis_title_text='Лайфтайм',
    yaxis_title_text='Количество'
```

```
)
fig.show()
```

В группе B:

- событие `product_page` распределено по закону, приближенному к пуассоновскому, основная доля этих событий произошла в первые 21 день с момента привлечения;
- событие `product_cart` распределено по закону, приближенному к пуассоновскому, основная доля этих событий произошла в первые 21 день с момента привлечения;
- событие `purchase` распределено по закону, приближенному к пуассоновскому, вплоть до 23 лайфтайма.

Промежуточный вывод: целевые события воронки для группы A расположены более компактно к началу тестирования, что может свидетельствовать о более высокой конверсии по сравнению с группой B в условиях временных ограничений.

2. Ограничим лог событий лайфтаймами с 0 по 13 (14 дней):

In [39]:
```
target_events_14 = (
    target_eu_participant_events.query('lifetime < 14')
)
target_events_14
```

Out[39]:

	user_id	group	event_dt	event_name	details	first_date	lifetime
0	D72A72121175D8BE	A	2020-12-07 21:52:10	product_page	NaN	2020-12-07	0
1	D72A72121175D8BE	A	2020-12-07 21:52:07	login	NaN	2020-12-07	0
2	DD4352CDCF8C3D57	B	2020-12-07 15:32:54	product_page	NaN	2020-12-07	0
3	DD4352CDCF8C3D57	B	2020-12-08 08:29:31	product_page	NaN	2020-12-07	1
4	DD4352CDCF8C3D57	B	2020-12-10 18:18:27	product_page	NaN	2020-12-07	3
...	...	...	...	...	...	...	...
23415	0416B34D35C8C8B8	A	2020-12-21 22:28:29	product_page	NaN	2020-12-20	1
23416	0416B34D35C8C8B8	A	2020-12-24 09:12:51	product_page	NaN	2020-12-20	4
23417	0416B34D35C8C8B8	A	2020-12-20 20:58:25	login	NaN	2020-12-20	0
23418	0416B34D35C8C8B8	A	2020-12-21 22:28:29	login	NaN	2020-12-20	1
23419	0416B34D35C8C8B8	A	2020-12-24 09:12:49	login	NaN	2020-12-20	4

22620 rows × 7 columns

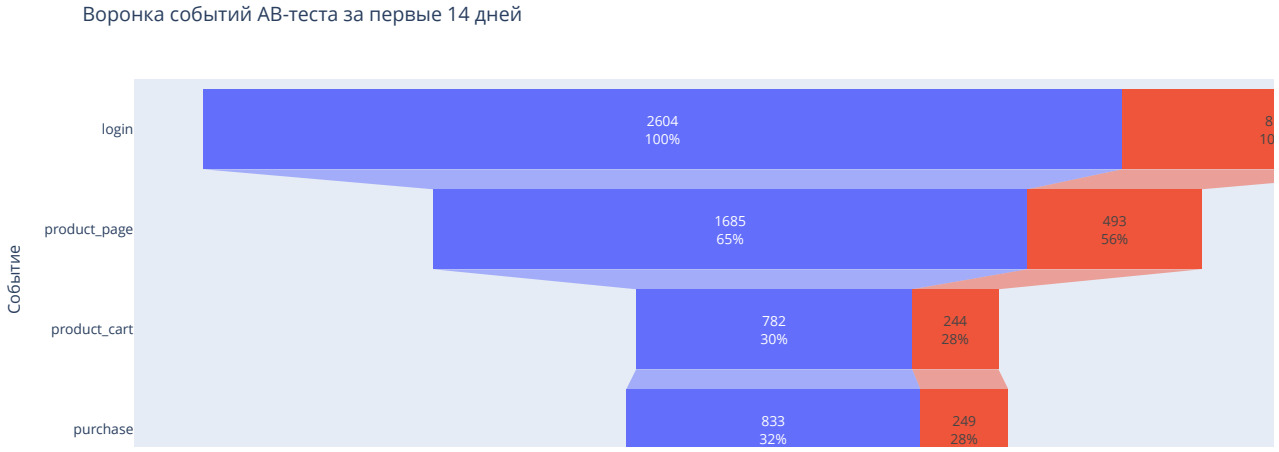
Итак у нас в логе осталось 22620 событий, совершённых пользователями групп A и B в первые 14 дней с момента привлечения.

3. Построим воронку событий, посчитаем требуемые конверсии для групп A и B:

In [40]:
```
# построим воронку
funnel_14 = (
    target_events_14
    .groupby(by=['event_name', 'group'], as_index=False)
    .agg({'user_id':'nunique'})
    .merge(
        event_num, on='event_name', how='left'
    )
    .sort_values(by=['number', 'group'])
)
display(funnel_14)

# визуализируем воронку
fig = go.Figure()
groups = funnel_14.group.unique()

for g in groups:
    fig.add_trace(go.Funnel(
        name=g,
        y=list(funnel_14.query('group == @g')['event_name']),
        x=list(funnel_14.query('group == @g')['user_id']),
        textinfo = "value+percent initial"
    ))
```

```
# зададим названия гистограммы и осей
fig.update_layout(
    title_text='Воронка событий АВ-теста за первые 14 дней',
    yaxis_title_text='Событие'
)
fig.show()
```

	event_name	group	user_id	number
0	login	A	2604	1
1	login	B	876	1
4	product_page	A	1685	2
5	product_page	B	493	2
2	product_cart	A	782	3
3	product_cart	B	244	3
6	purchase	A	833	4
7	purchase	B	249	4



Итак, мы видим, что **по имеющимся данным о событиях конверсии всех этапов для тестовой группы B ниже, чем в контрольной группе A**.

Предварительно можно утверждать, что ожидаемый эффект не достигнут.

4. Оценим разницу в конверсиях:

In [41]:
```
# сгруппируем данные о событиях
funnel_14_comparison = (
    # посчитаем количество пользователей группы A по этапам воронки
    funnel_14.query('group == "A"')
    .merge( # добавим общую численность группы A
        target_events_14
        .groupby(by='group', as_index=False).agg({'user_id':'nunique'})
        .rename(columns={'user_id':"total_users"}),
        on='group', how='left'
    )
    .rename(columns={'user_id':'stage_users_a', 'total_users':'total_users_a'})
    .drop(columns=['group', 'number'])
    .merge( # добавим количество пользователей группы B по этапам воронки
        funnel_14.query('group == "B"')
        .merge( # добавим общую численность группы B
            target_events_14
            .groupby(by='group', as_index=False).agg({'user_id':'nunique'})
            .rename(columns={'user_id':"total_users"}),
            on='group', how='left'
        )
        .rename(columns={'user_id':'stage_users_b', 'total_users':'total_users_b'})
        .drop(columns=['group', 'number']),
        on='event_name', how='left'
    )
)

# посчитаем отношение конверсий по этапам
funnel_14_comparison['ratio'] = (
    (funnel_14_comparison['stage_users_b'] * funnel_14_comparison['total_users_a']) /
    (funnel_14_comparison['total_users_b'] * funnel_14_comparison['stage_users_a'])
).round(2)

funnel_14_comparison
```

Out[41]:

	event_name	stage_users_a	total_users_a	stage_users_b	total_users_b	ratio
0	login	2604	2604	876	877	1.00
1	product_page	1685	2604	493	877	0.87
2	product_cart	782	2604	244	877	0.93
3	purchase	833	2604	249	877	0.89

Анализируя полученную таблицу, можно заметить, что:

1. Конверсия группы B на этап `product_page` ниже конверсии группы A на 13%.
2. Конверсия группы B на этап `product_cart` ниже конверсии группы A на 7%.
3. Конверсия группы B на этап `purchase` ниже конверсии группы A на 11%.

Вывод: ожидаемый эффект теста не достигнут.

Оценка статистической разницы долей

Оценим статистическую разницу конверсий на разных этапах воронки с помощью теста на равенство долей. Реализуем тест в виде функции:

In [42]:
```
# определение функции статистического критерия на равенство долей
# ======================================================================
def prop_difference_criteria(
    df,          # датафрейм с данными пропорций
    part_col,    # числитель пропорции
    full_col,    # знаменатель пропорции
    alpha=.05    # критический уровень статистической значимости

):
    alpha = alpha

    successes = np.array(df[part_col])
    trials = np.array(df[full_col])

    # 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 / mth.sqrt(p_combined * (1 - p_combined) *
                                    (1/trials[0] + 1/trials[1]))

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

    # считаем вероятность того, что статистика "уехала" от 0 на заданную величину
    # или больше, с использованием кумулятивной функции распределения (для
    # нормального распределения)
    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

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

Гипотеза о разнице конверсий в этап `product_page` для групп A и B

Гипотеза состоит в том, что конверсии в этап `product_page` для групп A и B различаются. Сформулируем основную статистическую гипотезу и альтернативу:

- **Основная гипотеза H0**: конверсии в этап `product_page` для групп A и B одинаковы;
- **Альтернативная гипотеза H1**: конверсии в этап `product_page` для групп A и B различаются.

Для проверки гипотезы нам нужны общее количество пользователей, для каждой группы, а также количество тех из них, которые совершили целевое действие. Количество пользователей, совершавших события, хранится в таблице `funnel_14`, к которой необходимо добавить общую численность групп для запуска функции `prop_difference_criteria`:

In [43]:
```
# сформируем данные для теста
test_data = (
    funnel_14
    .merge(
        target_events_14
        .groupby(by='group', as_index=False).agg({'user_id':'nunique'})
        .rename(columns={'user_id':"total_users"}),
        on='group', how='left'
    )
    .drop(columns=['number'])
    .query('event_name == "product_page"')
)
display(test_data)

# запустим тест на равенство долей
prop_difference_criteria(test_data, 'user_id', 'total_users')
```

	event_name	group	user_id	total_users
2	product_page	A	1685	2604
3	product_page	B	493	877

```
p-значение:  6.942739359416805e-06
Отвергаем нулевую гипотезу: между долями есть значимая разница
```

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 есть основания считать конверсии в этап `product_page` для групп A и B различными.

Гипотеза о разнице конверсий в этап `product_cart` для групп A и B

Гипотеза состоит в том, что конверсии в этап `product_cart` для групп A и B различаются. Сформулируем основную статистическую гипотезу и альтернативу:

- **Основная гипотеза H0**: конверсии в этап `product_cart` для групп A и B одинаковы;
- **Альтернативная гипотеза H1**: конверсии в этап `product_cart` для групп A и B различаются.

Сформируем тестовые данные и запустим функцию `prop_difference_criteria`:

In [44]:
```
# сформируем данные для теста
test_data = (
    funnel_14
    .merge(
        target_events_14
        .groupby(by='group', as_index=False).agg({'user_id':'nunique'})
        .rename(columns={'user_id':"total_users"}),
        on='group', how='left'
    )
    .drop(columns=['number'])
    .query('event_name == "product_cart"')
)
display(test_data)

# запустим тест на равенство долей
prop_difference_criteria(test_data, 'user_id', 'total_users')
```

	event_name	group	user_id	total_users
4	product_cart	A	782	2604
5	product_cart	B	244	877

```
p-значение:  0.21469192029582396
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии в этап `product_cart` для групп A и B различными.

Гипотеза о разнице конверсий в этап `purchase` для групп A и B

Гипотеза состоит в том, что конверсии в этап `purchase` для групп A и B различаются. Сформулируем основную статистическую гипотезу и альтернативу:

- **Основная гипотеза H0**: конверсии в этап `purchase` для групп A и B одинаковы;
- **Альтернативная гипотеза H1**: конверсии в этап `purchase` для групп A и B различаются.

Сформируем тестовые данные и запустим функцию `prop_difference_criteria`:

In [45]:
```
# сформируем данные для теста
test_data = (
    funnel_14
    .merge(
        target_events_14
        .groupby(by='group', as_index=False).agg({'user_id':'nunique'})
        .rename(columns={'user_id':"total_users"}),
        on='group', how='left'
    )
    .drop(columns=['number'])
    .query('event_name == "purchase"')
)
display(test_data)

# запустим тест на равенство долей
prop_difference_criteria(test_data, 'user_id', 'total_users')
```

	event_name	group	user_id	total_users
6	purchase	A	833	2604
7	purchase	B	249	877

```
p-значение:  0.04652482738393027
Отвергаем нулевую гипотезу: между долями есть значимая разница
```

На имеющихся данных при заданном критическом уровне статистической значимости 0.05 есть основания считать конверсии в этап `purchase` для групп A и B различными.

Выводы

В ходе оценки результатов AB-теста мы построили воронку событий на глубину 14 лайфтаймов с момента привлечения пользователя и установили, что:

- целевые события воронки для группы A расположены более компактно к началу тестирования, что может свидетельствовать о более высокой конверсии по сравнению с группой B в условиях временных ограничений;
- анализируя полученную воронку, можно заметить, что:
  - конверсия группы B на этап `product_page` ниже конверсии группы A на 13%.
  - конверсия группы B на этап `product_cart` ниже конверсии группы A на 7%.
  - конверсия группы B на этап `purchase` ниже конверсии группы A на 11%.

Вывод: ожидаемый эффект AB-теста не достигнут.

В подтверждение этого мы проверили три статистических гипотезы:

- конверсии в этап `product_page` для групп A и B различаются;
- конверсии в этап `product_cart` для групп A и B различаются;
- конверсии в этап `purchase для` групп A и B различаются.

Для проверки использован критерий равенства долей (`z-test`):

1. На имеющихся данных при заданном критическом уровне статистической значимости 0.05 есть основания считать конверсии в этап `product_page` для групп A и B различными.

1. На имеющихся данных при заданном критическом уровне статистической значимости 0.05 нет оснований считать конверсии в этап `product_cart` для групп A и B различными.

1. На имеющихся данных при заданном критическом уровне статистической значимости 0.05 есть основания считать конверсии в этап `purchase` для групп A и B различными.

Итоговые выводы исследования и рекомендации

В процессе анализа проведённого AB-теста мы изучили (EDA) предоставленные данные, построили целевую воронку событий и посчитали и сравнили между собой конверсии в различные этапы воронки групп A и B.

Подробные выводы по этапам исследования представлены в соответствующих разделах.

Резюмируя данные выводы, целесообразно признать собранные данные не соответствующими техническому заданию на AB-тест. Основной причиной с высокой долей вероятности является некорректная работа логирования событий - нам пришлось удалить из выборки свыше 28% от исходной численности группы A и свыше 67% от численности группы B.

Вместе с тем, оставшиеся данные сохранили статистическую значимость, а проведённое статистическое сравнение целевых метрик между группами A и B позволяет считать ожидаемый эффект AB-теста не достигнутым.

Ввиду того, что основная доля событий для пользователей распределена в течение первых 14-17 лайфтаймов, досрочное прекращение теста могло оказать незначительное влияние только на пользователей, пришедших в конце третьей недели привлечения, что не должно отразиться на статистической значимости результатов теста в целом.

По результатам анализа можно сформулировать следующие рекомендации:

- признать тест состоятельным и зафиксировать снижение конверсии для тестовой группы B по сравнению с контрольной группой A;
- признать новую систему рекомендаций неудачной и направить на доработку;
- доработать систему логирования пользовательских событий с целью повышения качества последующих AB-тестов.