Проект: A/B тестирование

1 Описание проекта

Цель проекта:

- оценка изменений, связанных с внедрением улучшенной рекомендательной системы.

Задача проекта:

- оценить корректность проведения A/B теста;
- провести анализ результатов A/B-теста.

Техническое задание:

- 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` .

Этапы выполнения проекта:

- постановка цели;

- анализ исходных данных;
- оценка корректности проведения теста;
- иследовательский анализ данных;
- оценка результатов A/B теста;
- формулировка выводов.

Источник данных:

- ab_project_marketing_events.csv - календарь маркетинговых событий на 2020 год.
- final_ab_new_users.csv - пользователи, зарегистрировавшиеся с 7 до 21 декабря 2020 года.
- final_ab_events.csv - действия новых пользователей в период с 7 декабря 2020 по 4 января 2021 года.
- final_ab_participants.csv - таблица участников тестов.

Описание данных(структура файлов):

- *ab_project_marketing_events.csv*:

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

- *final_ab_new_users.csv*:

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

- *final_ab_events.csv*:

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

- *final_ab_participants.csv*:

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

Итоговые результаты:

- заключение о корректности проведения теста;
- выводы по результатам исследовательского анализа;
- выводы по анализу результатов A/B теста.

К выводам

2 Подготовительный этап анализа

Задачи раздела:

- загрузить файлы;
- создать датасеты;
- проверить качество данных в датасетах;
- проверить соответствие типов данных характеру данных;
- выявить дубликаты: явные и неявные;
- выявить пропуски и принять решение, что с ними делать;
- познакомиться с данными в разрезе столбцов датасетов.

Импорт библиотек

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

In [2]:
```
# функция для первичного знакомства с новым датасетом
def first_info(df):
    print('Вывод первых 5 строк датасета:', end='\n\n')
    print(df.head(), end='\n\n')
    print('-------------------------------------------------------------------------------')
    print('Вывод общей информации по датасету:', end='\n\n')
    print(df.info(), end='\n\n')
    print('-------------------------------------------------------------------------------')
    print('Вывод информации о пропусках в столбцах датасета:', end='\n\n')
    print(df.isnull().sum(), end='\n\n')
    print('-------------------------------------------------------------------------------')
    print('Проверка данных методом describe:', end='\n\n')
    print(df.describe(include='all').round(2), end='\n\n')
    print('-------------------------------------------------------------------------------')
    print('Количество полных дубликатов по датасету:', df.duplicated().sum(), end='\n\n')
    print('-------------------------------------------------------------------------------')
    print('Список названий столбцов:', end='\n\n')
    print(*df.columns, sep='\n', end='\n\n')
    return
```

✓ **Комментарий ревьюера:** Здорово, что ты пишешь функцию для первичного изучения датасетов. Это помогает сократить код в случаях, когда датасетов несколько

2.1 Загрузка данных и проверка их качества

В данном разделе будут созданы 4 базовых датасета:

marketing_events - календарь маркетинговых событий на 2020 год

new_users - пользователи, зарегистрировавшиеся с 7 до 21 декабря 2020 года.

events - действия новых пользователей в период с 7 декабря 2020 по 4 января 2021 года.

participants - таблица участников тестов.

2.1.1 Файл ab_project_marketing_events.csv

In [3]:
```
# загрузим файл ab_project_marketing_events.csv и создадим датасет marketing_events
try:
    marketing_events = pd.read_csv('/datasets/ab_project_marketing_events.csv', sep=',')
except:
    marketing_events = pd.read_csv('https://ab_project_marketing_events.csv', sep=',')
```

In [4]:
```
# исследуем датасет с помощью функции first_info
first_info(marketing_events)
```

```
Вывод первых 5 строк датасета:

                          name                   regions    start_dt  \
0      Christmas&New Year Promo             EU, N.America  2020-12-25
1  St. Valentine's Day Giveaway  EU, CIS, APAC, N.America  2020-02-14
2        St. Patric's Day Promo             EU, N.America  2020-03-17
3                  Easter Promo  EU, CIS, APAC, N.America  2020-04-12
4             4th of July Promo                 N.America  2020-07-04

    finish_dt
0  2021-01-03
1  2020-02-16
2  2020-03-19
3  2020-04-19
4  2020-07-11

--------------------------------------------------------------------------------
Вывод общей информации по датасету:

<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
None

--------------------------------------------------------------------------------
Вывод информации о пропусках в столбцах датасета:

name         0
regions      0
start_dt     0
finish_dt    0
dtype: int64
```

```
--------------------------------------------------------------------------------
Проверка данных методом describe:

                            name regions    start_dt   finish_dt
count                         14      14          14          14
unique                        14       6          14          14
top     Christmas&New Year Promo    APAC  2020-12-30  2020-04-19
freq                           1       4           1           1

--------------------------------------------------------------------------------
Количество полных дубликатов по датасету: 0

--------------------------------------------------------------------------------
Список названий столбцов:

name
regions
start_dt
finish_dt
```

Выводы:

- названия столбцов соответствуют стандартному виду;
- в датасете 4 столбца, 14 строк;
- типы данных во всех столбцах object;
- **необходима замена типа данных в столбцах start_dt и finish_dt с object на datetime**;
- пропусков в датасете нет;
- полных дубликатов в датасете нет.

In [5]:
```
# выполним замену типа данных и проверим результат замены
marketing_events['start_dt'] = pd.to_datetime(marketing_events['start_dt'])
marketing_events['finish_dt'] = pd.to_datetime(marketing_events['finish_dt'])
marketing_events.info()
```

```
<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     datetime64[ns]
 3   finish_dt  14 non-null     datetime64[ns]
dtypes: datetime64[ns](2), object(2)
memory usage: 576.0+ bytes
```

```
# выполним замену типа данных
marketing_events['start_dt'] = marketing_events['start_dt'].map(
    lambda x: dt.datetime.strptime(x, '%d/%m/%Y')
)
```

2.1.2 Файл final_ab_new_users.csv

In [6]:
```
# загрузим файл final_ab_new_users.csv и создадим датасет new_users
try:
    new_users = pd.read_csv('/datasets/final_ab_new_users.csv', sep=',')
except:
    new_users = pd.read_csv('https://final_ab_new_users.csv', sep=',')
```

In [7]:
```
# исследуем датасет с помощью функции first_info
first_info(new_users)
```

```
Вывод первых 5 строк датасета:

            user_id  first_date     region   device
0  D72A72121175D8BE  2020-12-07         EU       PC
1  F1C668619DFE6E65  2020-12-07  N.America  Android
2  2E1BF1D4C37EA01F  2020-12-07         EU       PC
3  50734A22C0C63768  2020-12-07         EU   iPhone
4  E1BDDCE0DAFA2679  2020-12-07  N.America   iPhone

--------------------------------------------------------------------------------
Вывод общей информации по датасету:

<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
None

--------------------------------------------------------------------------------
Вывод информации о пропусках в столбцах датасета:

user_id       0
first_date    0
region        0
device        0
dtype: int64

--------------------------------------------------------------------------------
Проверка данных методом describe:

                user_id  first_date region   device
count             61733       61733  61733    61733
unique            61733          17      4        4
top     A6CBB5B57BB0C8C5  2020-12-21     EU  Android
```

```
freq                   1        6290  46270    27520

--------------------------------------------------------------------------------
Количество полных дубликатов по датасету: 0

--------------------------------------------------------------------------------
Список названий столбцов:

user_id
first_date
region
device
```

Выводы:

- названия столбцов соответствуют стандартному виду;
- в датасете 4 столбца, 61733 строк;
- типы данных во всех столбцах object;
- **необходима замена типа данных в столбце first_date с object на datetime**;
- пропусков в датасете нет;
- полных дубликатов в датасете нет.

In [8]:
```
# выполним замену типа данных и проверим результат замены
new_users['first_date'] = pd.to_datetime(new_users['first_date'])
new_users.info()
```

```
<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  datetime64[ns]
 2   region      61733 non-null  object        
 3   device      61733 non-null  object        
dtypes: datetime64[ns](1), object(3)
memory usage: 1.9+ MB
```

2.1.3 Файл final_ab_events.csv

In [9]:
```
# загрузим файл final_ab_events.csv и создадим датасет events
try:
    events = pd.read_csv('/datasets/final_ab_events.csv', sep=',')
except:
    events = pd.read_csv('https://final_ab_events.csv', sep=',')
```

In [10]:
```
# исследуем датасет с помощью функции first_info
first_info(events)
```

```
Вывод первых 5 строк датасета:

            user_id             event_dt event_name  details
0  E1BDDCE0DAFA2679  2020-12-07 20:22:03   purchase    99.99
1  7B6452F081F49504  2020-12-07 09:22:53   purchase     9.99
2  9CD9F34546DF254C  2020-12-07 12:59:29   purchase     4.99
3  96F27A054B191457  2020-12-07 04:02:40   purchase     4.99
4  1FD7660FDF94CA1F  2020-12-07 10:15:09   purchase     4.99

--------------------------------------------------------------------------------
Вывод общей информации по датасету:

<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
None

--------------------------------------------------------------------------------
Вывод информации о пропусках в столбцах датасета:

user_id            0
event_dt           0
event_name         0
details       377577
dtype: int64

--------------------------------------------------------------------------------
Проверка данных методом describe:

                user_id             event_dt event_name   details
count            440317               440317     440317  62740.00
unique            58703               267268          4       NaN
top     A3917F81482141F2  2020-12-23 02:37:24      login       NaN
```

```
freq                  36                  10     189552       NaN
mean                 NaN                 NaN        NaN     23.88
std                  NaN                 NaN        NaN     72.18
min                  NaN                 NaN        NaN      4.99
25%                  NaN                 NaN        NaN      4.99
50%                  NaN                 NaN        NaN      4.99
75%                  NaN                 NaN        NaN      9.99
max                  NaN                 NaN        NaN    499.99

--------------------------------------------------------------------------------
Количество полных дубликатов по датасету: 0

--------------------------------------------------------------------------------
Список названий столбцов:

user_id
event_dt
event_name
details
```

Выводы:

- названия столбцов соответствуют стандартному виду;
- в датасете 4 столбца, 440317 строк;
- типы данных в 3-х столбцах object, в одном - float64;
- **необходима замена типа данных в столбце event_dt с object на datetime**;
- **377577 пропусков в столбце details**;
- полных дубликатов в датасете нет.

In [11]:
```
# выполним замену типа данных и проверим результат замены
events['event_dt'] = pd.to_datetime(events['event_dt'])
events.info()
```

```
<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  datetime64[ns]
 2   event_name  440317 non-null  object        
 3   details     62740 non-null   float64       
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 13.4+ MB
```

2.1.4 Файл final_ab_participants.csv

In [12]:
```
# загрузим файл final_ab_participants.csv и создадим датасет participants
try:
    participants = pd.read_csv('/datasets/final_ab_participants.csv', sep=',')
except:
    participants = pd.read_csv('https://final_ab_participants.csv', sep=',')
```

In [13]:
```
# исследуем датасет с помощью функции first_info
first_info(participants)
```

```
Вывод первых 5 строк датасета:

            user_id group                  ab_test
0  D1ABA3E2887B6A73     A  recommender_system_test
1  A7A3664BD6242119     A  recommender_system_test
2  DABC14FDDFADD29E     A  recommender_system_test
3  04988C5DF189632E     A  recommender_system_test
4  482F14783456D21B     B  recommender_system_test

--------------------------------------------------------------------------------
Вывод общей информации по датасету:

<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
None

--------------------------------------------------------------------------------
Вывод информации о пропусках в столбцах датасета:

user_id    0
group      0
ab_test    0
dtype: int64

--------------------------------------------------------------------------------
Проверка данных методом describe:

                user_id  group            ab_test
count             18268  18268              18268
unique            16666      2                  2
top     2F3AC426BF87D50E      A  interface_eu_test
freq                  2   9655              11567
```

```
--------------------------------------------------------------------------------
Количество полных дубликатов по датасету: 0

--------------------------------------------------------------------------------
Список названий столбцов:

user_id
group
ab_test
```

Выводы:

- названия столбцов соответствуют стандартному виду;
- в датасете 3 столбца, 18268 строк;
- типы данных во всех столбцах object;
- пропусков в датасете нет;
- полных дубликатов в датасете нет.

2.2 Предварительный анализ данных

2.2.1 Календарь маркетинговых событий на 2020 год: *marketing_events*

In [14]:
```
marketing_events.info(2)
```

```
<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     datetime64[ns]
 3   finish_dt  14 non-null     datetime64[ns]
dtypes: datetime64[ns](2), object(2)
memory usage: 576.0+ bytes
```

In [15]:
```
# Изучим столбец name и узнаем названия маркетинговых событий
print('Количество уникальных маркетинговых событий в столбце name:', marketing_events['name'].nunique())
print()
print(*marketing_events['name'].unique(), sep = '\n')
```

```
Количество уникальных маркетинговых событий в столбце name: 14

Christmas&New Year Promo
St. Valentine's Day Giveaway
St. Patric's Day Promo
Easter Promo
4th of July Promo
Black Friday Ads Campaign
Chinese New Year Promo
Labor day (May 1st) Ads Campaign
International Women's Day Promo
Victory Day CIS (May 9th) Event
CIS New Year Gift Lottery
Dragon Boat Festival Giveaway
Single's Day Gift Promo
Chinese Moon Festival
```

In [16]:
```
# Изучим столбец regions и узнаем регионы, в которых проводятся маркетинговые события
print('Количество уникальных значений в столбце regions (название региона проведения РК):', marketing_events['regions'
print()
print(*marketing_events['regions'].unique(), sep = '\n')
```

```
Количество уникальных значений в столбце regions (название региона проведения РК): 6

EU, N.America
EU, CIS, APAC, N.America
N.America
APAC
EU, CIS, APAC
CIS
```

In [17]:
```
# Изучим столбец start_dt
print('Кол-во уникальных значений в столбце start_dt', marketing_events['start_dt'].nunique())
print('Дата начала проведения первого маркетингового события', min(marketing_events['start_dt']))
print('Дата начала проведения последнего маркетингового события',max(marketing_events['start_dt']))
```

```
Кол-во уникальных значений в столбце start_dt 14
Дата начала проведения первого маркетингового события 2020-01-25 00:00:00
Дата начала проведения последнего маркетингового события 2020-12-30 00:00:00
```

In [18]:
```
# Изучим столбец finish_dt
print('Кол-во уникальных значений в столбце finish_dt', marketing_events['finish_dt'].nunique())
print('Первая дата окончания проведения маркетингового события', min(marketing_events['finish_dt']))
print('Последняя дата окончания проведения маркетингового события',max(marketing_events['finish_dt']))
```

```
Кол-во уникальных значений в столбце finish_dt 14
Первая дата окончания проведения маркетингового события 2020-02-07 00:00:00
Последняя дата окончания проведения маркетингового события 2021-01-07 00:00:00
```

In [19]:
```
# Создадим сводную таблицу регион - кол-во маркетинговых событий
marketing_events.pivot_table(index='regions', values = 'name', aggfunc = 'count')
```

Out[19]:

regions	name
APAC	4
CIS	2
EU, CIS, APAC	2
EU, CIS, APAC, N.America	3
EU, N.America	2
N.America	1

Выводы:

- скрытых дубликатов в датасете не выявлено;
- всего в 2020 г. было 14 маркетинговых собятий;
- маркетинговые события проводились в 4 регионах: EU, CIS, APAC, N.America;
- некоторые маркетинговые события проводились одновременно в нескольких регионах.

2.2.2 Пользователи, зарегистрировавшиеся с 7 до 21 декабря 2020 года: new_users

In [20]:
```
new_users.info()
```

```
<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  datetime64[ns]
 2   region      61733 non-null  object        
 3   device      61733 non-null  object        
dtypes: datetime64[ns](1), object(3)
memory usage: 1.9+ MB
```

In [21]:
```
# Изучим столбец user_id
print('Количество уникальных значений в столбце user_id :', new_users['user_id'].nunique())
```

```
Количество уникальных значений в столбце user_id : 61733
```

In [22]:
```
# Изучим столбец first_date - дата регистрации пользователя
print('Первая дата регистрации пользователей', min(new_users['first_date']))
print('Последняя дата регистрации пользователей',max(new_users['first_date']))
```

```
Первая дата регистрации пользователей 2020-12-07 00:00:00
Последняя дата регистрации пользователей 2020-12-23 00:00:00
```

In [23]:
```
# Изучим столбец region - регион регистрации пользователя
print('Количество уникальных значений в столбце region:', new_users['region'].nunique())
print()
print('Уникальные наименования в столбце region:', *new_users['region'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце region: 4

Уникальные наименования в столбце region:
EU
N.America
APAC
CIS
```

Получить данные о распределении пользователей по значениям столбца можно с помощью метода value-counts()

Но выглядит это не красиво, поэтому здесь и далее буду ипользовать pivot_table

In [24]:
```
# Создадим сводну таблицу кол-во новых пользователей в разрезе регионов
new_users.pivot_table(index='region', values = 'user_id', aggfunc = 'count')
```

Out[24]:

region	user_id
APAC	3153
CIS	3155
EU	46270
N.America	9155

In [25]:
```
# Изучим столбец device - устройство, с которого происходила регистрация
print('Количество уникальных значений в столбце device:', new_users['device'].nunique())
print()
print('Уникальные наименования в столбце device:', *new_users['device'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце device: 4

Уникальные наименования в столбце device:
PC
Android
iPhone
Mac
```

In [26]:
```
# Создадим сводну таблицу кол-во пользователей в разрезе устройств регистрации
new_users.pivot_table(index='device', values = 'user_id', aggfunc = 'count')
```

Out[26]:

device	user_id
Android	27520
Mac	6084
PC	15599
iPhone	12530

In [27]:
```
# Создадим сводну таблицу кол-во пользователей в устройств регистрации и регионов
new_users.pivot_table(index = 'region', columns='device', values = 'user_id', aggfunc = 'count')
```

Out[27]:

region	device	Android	Mac	PC	iPhone
APAC		1401	316	803	633
CIS		1413	310	776	656
EU		20629	4575	11693	9373
N.America		4077	883	2327	1868

Выводы:

- скрытых дублей в датасете не выявленно - в столбце user_id дублей нет;
- пользователи регистрировались в 4-х регионах: APAC, CIS, EU, N.America;
- наибольшее количество зарегистрированных приходится на регион EU;
- регистрацию пользователи производили с 4-х устройств: Android, Mac, PC, iPhone;
- наибольшее количество зарегистрированных приходится на устройство Android;
- в интересующем нас регионе EU самое распространенное устройство - Android, на втором месте - PC, на третьем - iPhone;
- первая дата регистрации в датасете 2020-12-07, что соответствует описанию данных;

- **последняя дата регистрации в датасете 2020-12-23, что не соответствует описанию данных**. По условию ТЗ предполагалось, что последней датой должно быть 2020-12-21. Очистим данные позднее.

2.2.3 Действия новых пользователей в период с 7 декабря 2020 по 4 января 2021 года: *events*

In [28]:
```
events.info()
```

```
<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  datetime64[ns]
 2   event_name  440317 non-null  object        
 3   details     62740 non-null   float64       
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 13.4+ MB
```

In [29]:
```
# Изучим столбец user_id
print('Количество уникальных значений в столбце user_id :', events['user_id'].nunique())
```

```
Количество уникальных значений в столбце user_id : 58703
```

In [30]:
```
# Изучим столбец event_dt - дата и время события, совершенного пользователем
print('Кол-во уникальных значений в столбце event_dt', events['event_dt'].nunique())
print('Первая дата и время совершения события пользователем', min(events['event_dt']))
print('Последняя дата и время совершения события пользователем',max(events['event_dt']))
```

```
Кол-во уникальных значений в столбце event_dt 267268
Первая дата и время совершения события пользователем 2020-12-07 00:00:33
Последняя дата и время совершения события пользователем 2020-12-30 23:36:33
```

In [31]:
```
# Изучим столбец event_name - тип события
print('Количество уникальных значений в столбце event_name:', events['event_name'].nunique())
print()
print('Уникальные наименования в столбце event_name:', *events['event_name'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце event_name: 4

Уникальные наименования в столбце event_name:
purchase
product_cart
product_page
login
```

In [32]:
```
# Создадим сводну таблицу кол-во пользователей в разрезе типов совершаемых ими событий
events.pivot_table(index='event_name', values = 'user_id', aggfunc = ['count', 'nunique'])
```

Out[32]:

	count	nunique
	user_id	user_id
event_name		
login	189552	58697
product_cart	62462	19284
product_page	125563	38929
purchase	62740	19569

In [33]:
```
# Изучим столбец details - дополнительные данные о событии.
# Например, для покупок, purchase, в этом поле хранится стоимость покупки в долларах.
print('Количество уникальных значений в столбце details:', events['details'].nunique())
print()
print('Уникальные наименования в столбце details:', *events['details'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце details: 4

Уникальные наименования в столбце details:
99.99
9.99
4.99
499.99
nan
```

In [34]:
```
# Создадим сводну таблицу количество доп данных в разрезе событий
events.pivot_table(index='event_name', values = 'details', aggfunc = 'count')
```

Out[34]:

event_name	details
login	0
product_cart	0
product_page	0
purchase	62740

In [35]:
```
# Проверим все ли события типа purchase имеют доп. информацию
events.loc[events['event_name']=='purchase'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62740 entries, 0 to 62739
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   user_id     62740 non-null  object
 1   event_dt    62740 non-null  datetime64[ns]
 2   event_name  62740 non-null  object
 3   details     62740 non-null  float64
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 2.4+ MB
```

Выводы:

- скрытых дубликатов в датасете не выявлено, каждый пользователь может создать несколько событий, следовательно, нормально, что уникальных id меньше, чем строк в датасете;
- **в столбце details присутствуют пропуски, так как столбец содержит дополнительную информацию только к событию purchase, пропуски удалять или заменять не будем, оставим NaN**;
- события совершали 58703 пользователей;
- первая дата и время совершения события пользователем 2020-12-07 00:00:33;
- **последняя дата и время совершения события пользователем 2020-12-30 23:36:33 - хотя заявлено было, что данные до 2021-01-04**;
- в датасете 4 типа событий: purchase, product_cart, product_page, login;
- самое многочисленное событие - login 189552, на втором месте - product_page 125563, на третьем - purchase 62740, на четвертом - product_cart 62462;
- столбец details имеет всего 4 уникальных значения.

2.2.4 Таблица участников тестов: *participants*

In [36]:
```
participants.info()
```

```
<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
```

In [37]:
```
# Изучим столбец user_id
print('Количество уникальных значений в столбце user_id :', participants['user_id'].nunique())
```

```
Количество уникальных значений в столбце user_id : 16666
```

In [38]:
```
# Изучим столбец group - группа пользователя
print('Количество уникальных значений в столбце group:', participants['group'].nunique())
print()
print('Уникальные наименования в столбце group:', *participants['group'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце group: 2

Уникальные наименования в столбце group:
A
B
```

In [39]:
```
# Изучим столбец ab_test - название теста
print('Количество уникальных значений в столбце ab_test:', participants['ab_test'].nunique())
print()
print('Уникальные наименования в столбце ab_test:', *participants['ab_test'].unique(), sep='\n')
```

```
Количество уникальных значений в столбце ab_test: 2

Уникальные наименования в столбце ab_test:
recommender_system_test
interface_eu_test
```

In [40]:
```
# Создадим сводну таблицу кол-во пользователей в разрезе названий теста
participants.pivot_table(index='ab_test', values = 'user_id', aggfunc = 'count')
```

Out[40]:

ab_test	user_id
interface_eu_test	11567
recommender_system_test	6701

In [41]:
```
# Создадим сводну таблицу кол-во пользователей в разрезе названий теста и групп
participants.pivot_table(index='ab_test', columns = 'group', values = 'user_id', aggfunc = 'count')
```

Out[41]:

ab_test	group A	B
interface_eu_test	5831	5736
recommender_system_test	3824	2877

Выводы:

- скрытых дубликатов в столбце не выявлено, но **часть клиентов попали одновременно в несколько групп**, исследуем это подробнее на следующем этапе анализа;
- **в датасете информация о двух тестах: interface_eu_test и recommender_system_test**;
- в рамках каждого теста клиенты разделены на 2 группы A и B.

Правильность распределения клиентов по группам исследуем в следующем разделе.

3 Оценка корректности проведения теста

3.1 Проверка соответствия данных требованиям технического задания

Задача раздела:

- проверка корректности всех пунктов технического задания.

Техническое задание:

- 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: в тест должно быть отобрано 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` .

Не все пункты ТЗ требуют проверки, уже на данном этапе известно, что:

- все клиенты, действительно, разделены на две группы A и B;
- в датасете participants, действительно, есть данные о тесте recommender_system_test;
- назначение теста: тестирование изменений, связанных с внедрением улучшенной рекомендательной системы.

Нам нужно:

- сравнить дату запуска теста: 2020-12-07 с первой датой начала отбора клиентов в датасете new_users и первой датой события в датасете events;
- сравнить дату остановки набора новых пользователей для теста: 2020-12-21 с последней датой начала отбора клиентов в датасете new_users;
- сравнить дату остановки теста : 2021-01-04 с последней датой события в датасете events;
- расчитать % новых пользователей из региона EU, принявших участие в тесте - он должен быть равен 15%;
- очистить данные от данных с тестом interface_eu_test;
- проверить наличие пользователей, попавших в два теста одновременно;
- проверить ниличие пользователей попавших одновременно в группу A и B теста recommender_system_test;
- определить точное количество участников теста recommender_system_test и их распределение на группы;
- расчитать для каждого пользователя lifetime равный 14 дням с даты регистрации;
- расчитать % улучшения по трем метрикам: product_page, product_cart, purchase.

3.1.1 Проверка даты запуска теста на соответствие ТЗ

In [42]:
```
# Сверим дату начала теста с минимальной датой в датасете new_users
new_users['first_date'].min()
```

Out[42]:
```
Timestamp('2020-12-07 00:00:00')
```

In [43]:
```
# Сверим дату первого события в датасете events
events['event_dt'].min()
```

Out[43]:
```
Timestamp('2020-12-07 00:00:33')
```

Вывод:

Первая дата регистрации пользователя, попавшая в датасет new_users, и дата первого события в датасете events равны дате запуска теста - 07.12.2020 г., то есть **соответствуют ТЗ**.

3.1.2 Проверка даты остановки набора новых пользователей на соответствие ТЗ

In [44]:
```
# Сверим дату остановки  остановки набора новых пользователей с максимальной датой в датасете new_users
new_users['first_date'].max()
```

Out[44]:
```
Timestamp('2020-12-23 00:00:00')
```

Вывод:

Последняя дата регистрации пользователей, попавшая в датасет new_users (23.12.2020 г.), **не соответствует** дате остановки набора новых пользователей - должно быть 21.12.2020 г.

Приведем датасет new_users в соответствие с ТЗ.

In [45]:
```
# Приведем датасет new_users в соответствие с датой набора новых пользователей в ТЗ и проверим результат
new_users = new_users.loc[new_users['first_date']<='2020-12-21 00:00:00']
new_users['first_date'].max()
```

Out[45]:
```
Timestamp('2020-12-21 00:00:00')
```

3.1.3 Проверка дата остановки теста на соответствие ТЗ

In [46]:
```
# Сверим дату остановки теста с максимальной датой события в датасете events
events['event_dt'].max()
```

Out[46]:
```
Timestamp('2020-12-30 23:36:33')
```

Вывод:

- дата остановки теста 2021-01-04, указанная в ТЗ, **не соответствует** последней дате совершения пользователями события в датасете events 2020-12-30.

Причина отсутствия данных за период с 2020-12-31 по 2021-01-04 неизвестна.

Об отсутствии данных следует сообщить тимлиду и поставщику данных.

В настоящий момент 14-дневный тестовый период мы получили только по клиентам, зарегистрированным до 16 декабря, то есть по 9 когортам из 15. Такая ситуация ставит под сомнение результаты A\B теста в целом.

3.1.4 Проверка аудитории на соответствие ТЗ

In [47]:
```
# Расчитаем количество новых пользователей из региона EU
new_users_EU = new_users.loc[new_users['region']=='EU', 'user_id'].nunique()
new_users_EU
```

Out[47]: 42340

Присоединим к датасету **participants** датасет new_users по столбцу user_id

In [48]:
```
# Соединим датасет participants с датасетом new_users по столбцу user_id
participants = participants.merge(new_users, on = 'user_id', how = 'left').dropna()
participants.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17266 entries, 0 to 18267
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   user_id     17266 non-null  object        
 1   group       17266 non-null  object        
 2   ab_test     17266 non-null  object        
 3   first_date  17266 non-null  datetime64[ns]
 4   region      17266 non-null  object        
 5   device      17266 non-null  object        
dtypes: datetime64[ns](1), object(5)
memory usage: 944.2+ KB
```

In [49]:
```
# Расчитаем кол-во пользователей, принимающих участие в тесте recommender_system_test из региона EU
participants_EU = participants.loc[(participants['region']=='EU') & (participants['ab_test']=='recommender_system_test
participants_EU
```

Out[49]: 6351

In [50]:
```
# Расчитаем процент новых пользователей из EU, принимащих участие в тесте recommender_system_test
round(participants_EU/new_users_EU*100)
```

Out[50]: 15

Вывод:

- в тест отобрано 15% новых пользователей из региона EU, что **соответствует ТЗ**.

In [51]:
```
# Посмотрим на распределение пользователей в датасете participants по регионам
participants_region = pd.DataFrame(participants['region'].value_counts())
participants_region.columns = ['count']
participants_region['%_count'] = participants['region'].value_counts(normalize=True).round(2)*100
participants_region
```

Out[51]:

	count	%_count
EU	16916	98.0
N.America	223	1.0
APAC	72	0.0
CIS	55	0.0

Техническое задание гласит: в тест должно быть отобрано 15% новых пользователей из региона EU.
Следовательно, **удалим из датасета participants пользователей, регион которых не равен EU**.

Таких пользователей всего 2% или 350 человек

In [52]:
```
# Удалим пользователей из регионов не равных EU из датасета participants и проверим результат
participants = participants.loc[participants['region']=='EU']
print(participants.info())
print('-------------------------------------------------------------------------------')
participants['region'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16916 entries, 0 to 18267
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   user_id     16916 non-null  object        
 1   group       16916 non-null  object        
 2   ab_test     16916 non-null  object        
 3   first_date  16916 non-null  datetime64[ns]
 4   region      16916 non-null  object        
 5   device      16916 non-null  object        
dtypes: datetime64[ns](1), object(5)
memory usage: 925.1+ KB
None
-------------------------------------------------------------------------------
```

Out[52]:
```
EU    16916
Name: region, dtype: int64
```

Вывод:

- из датасета participants удалено 350 строк, осталось 16916 строк, т.е. 16916 пользователей.
- в датасете participants остались данные по пользователям одного региона - EU, что **соответствует ТЗ**.

3.2 Анализ времени проведения теста

Задача раздела:

- проверить, что время проведения теста не совпадает с маркетинговыми и другими активностями из датасета marketing_events.

In [53]:
```
marketing_events.info()
```

```
<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     datetime64[ns]
 3   finish_dt  14 non-null     datetime64[ns]
dtypes: datetime64[ns](2), object(2)
memory usage: 576.0+ bytes
```

In [54]:
```
# Найдем активности, время проведения которых, совпадает с временем проведения теста
marketing_events.loc[(marketing_events['start_dt']<=events['event_dt'].max()) & (marketing_events['finish_dt']>=events
```

Out[54]:

	name	regions	start_dt	finish_dt
0	Christmas&New Year Promo	EU, N.America	2020-12-25	2021-01-03
10	CIS New Year Gift Lottery	CIS	2020-12-30	2021-01-07

Вывод:

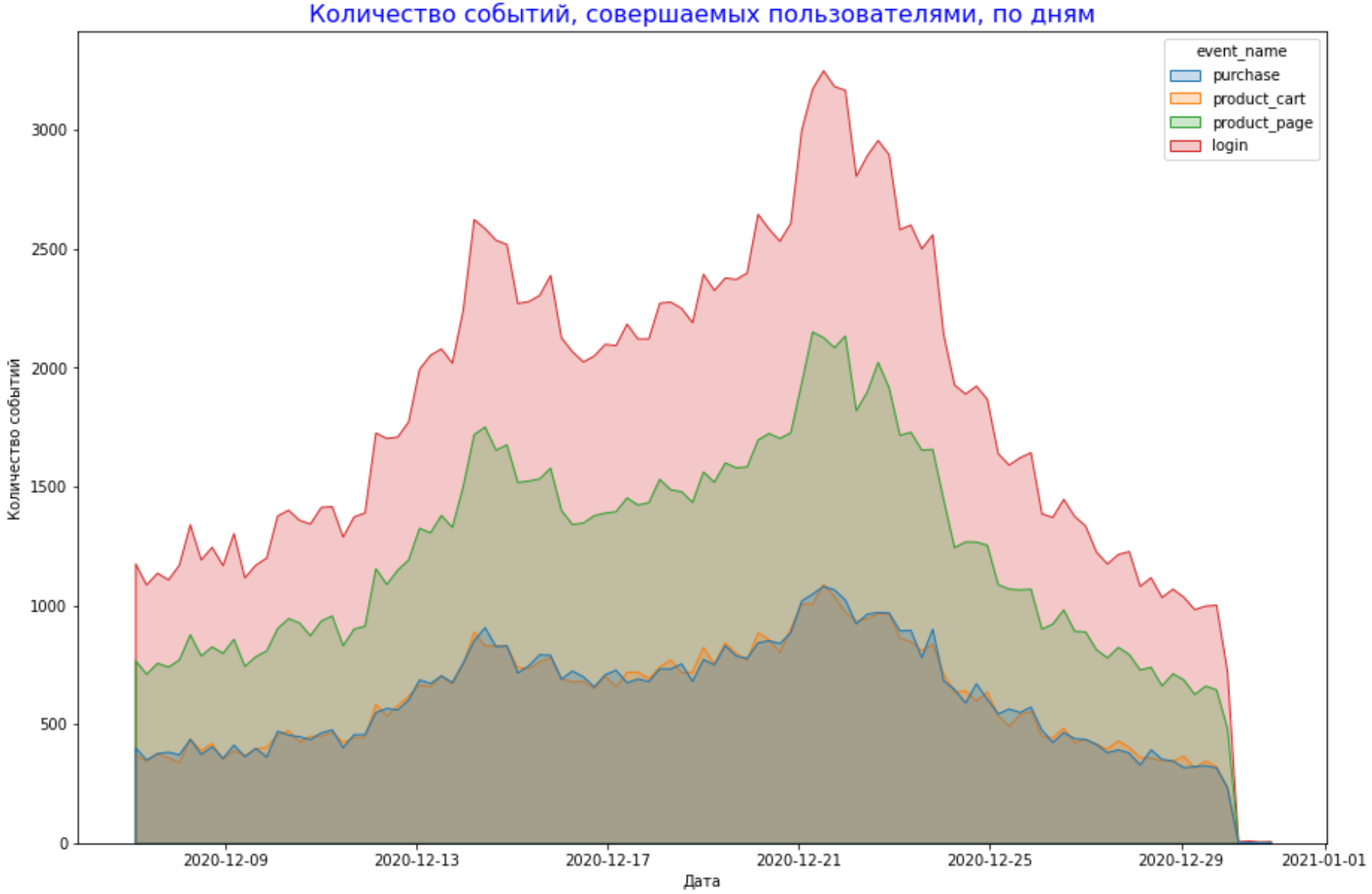
В период проведения теста проводились 2 маркетинговые активности.

Нас интересует только одна активность - Christmas&New Year Promo, так как только она проводилась в тестируемом регионе EU.

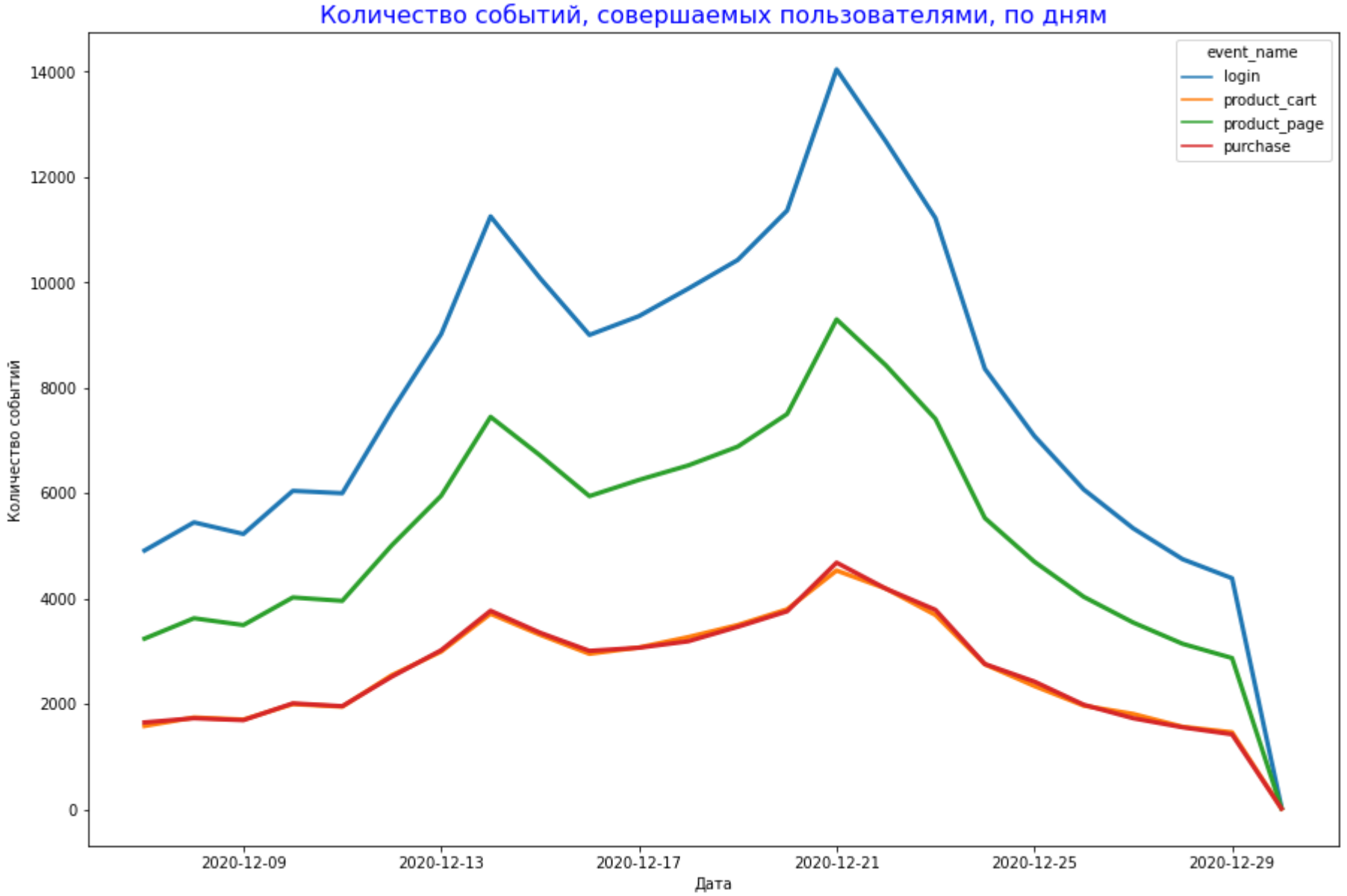
Проверим как данная активность сказалась на количесте событий, совершаемых пользователями.

In [55]:
```
# Создадим в датасете events столбец с датой
events['event_day'] = events['event_dt'].dt.date
events['event_day'] = pd.to_datetime(events['event_day'])
```

In [56]:
```
# Посмотрим на распределение количества событий, совершаемых пользователями, по дням
plt.figure(figsize=(15, 10))
sns.histplot(events, x = 'event_dt', hue = 'event_name', element="poly")
plt.title('Количество событий, совершаемых пользователями, по дням', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Дата')
plt.ylabel('Количество событий')
plt.show()
```


Количество событий, совершаемых пользователями, по дням
event_name
purchase
product_cart
product_page
login
Количество событий
0
500
1000
1500
2000
2500
3000
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
2021-01-01
Дата

In [57]:
```
# Вторая визуализация количества событий по дням в разрезе видов событий
day_events_all = events.pivot_table(index=['event_day', 'event_name'], values = 'user_id', aggfunc = 'count').reset_in
day_events_all.columns = ['event_day', 'event_name', 'count_events']
plt.figure(figsize=(15, 10))
sns.lineplot(data=day_events_all,
             x='event_day',
             y = "count_events",
             hue = 'event_name',
             linewidth=3,
             linestyle='solid',
             color=['darkblue', 'coral', 'lightblue', 'green'])
plt.title('Количество событий, совершаемых пользователями, по дням', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Дата')
plt.ylabel('Количество событий')
plt.show()
```



Чисто визуально я не нахожу на графике влияния маркетинговых событий на поведение пользователей.

Количество событий растет до 21.12, затем начинает снижаться и резко падает 30.12. Такое поведение скорее связано с рождественскими праздниками, чем с маркетинговым событием.

Следовательно,игнорирую факт совпадения маркетингового события с временим проведения теста.

3.3 Анализ аудитории теста

Задачи раздела:

- проверить наличие/отсутствие пересечений с конкурирующим тестом;
- проверить наличие/отсутствие пользователей, участвующих в двух группах теста одновременно;
- проверить равномерность распределения по тестовым группам и правильность их формирования.

3.3.1 Проверка наличия пересечений между конкурирующими тестами

В разделе 2.2.4 мы выяснили, что в датасете participants содержиться информация о двух тестах.
Согласно техзаданию нас интересует только тест `recommender_system_test`.

Проверим наличие пользователей, участвующих одновременно в двух тестах recommender_system_test и interface_eu_test

In [58]:
```
# Посмотрим на распределение пользователей по тестам
participants['ab_test'].value_counts()
```

Out[58]:
```
interface_eu_test          10565
recommender_system_test     6351
Name: ab_test, dtype: int64
```

In [59]:
```
# Найдем пользователей участвующих в двух тестах одновременно и посчитаем их кол-во
user_1 = set(participants.loc[participants['ab_test']=='recommender_system_test', 'user_id'].tolist())
user_2 = set(participants.loc[participants['ab_test']=='interface_eu_test', 'user_id'].tolist())
#len(user_1.intersection(user_2))
duble_id_user_test = list(user_1 & user_2)
len(duble_id_user_test)
```

Out[59]:
```
1602
```

In [60]:
```
# Посмотрим как пользователи участвующие в 2 тестах одновременно распределились на группы A и B в тесте interface_eu_t
participants.loc[(participants['user_id'].isin(duble_id_user_test)==True) & (participants['ab_test']=='interface_eu_te
```

Out[60]:
```
A    819
B    783
Name: group, dtype: int64
```

In [61]:
```
# Посмотрим как пользователи участвующие в 2 тестах одновременно распределились на группы A и B в тесте recommender_sy
participants.loc[(participants['user_id'].isin(duble_id_user_test)==True) & (participants['ab_test']=='recommender_sys
```

Out[61]:
```
A    921
B    681
Name: group, dtype: int64
```

In [62]:
```
# Пересчитаем список duble_id_user_test с учетом того, что нам нужны пользователи группы B места interface_eu_test и n
user_2 = set(participants.loc[(participants['ab_test']=='interface_eu_test') & (participants['group']=='B'), 'user_id'
duble_id_user_test = list(user_1 & user_2)
len(duble_id_user_test)
```

Out[62]: 783

In [63]:
```
# Снова посмотрим как пользователи участвующие в 2 местах одновременно распределились на группы A и B в месте recommen
a_b_duble = participants.loc[(participants['user_id'].isin(duble_id_user_test)==True) & (participants['ab_test']=='rec
a_b_duble = pd.DataFrame(a_b_duble)
a_b_duble['all'] = participants.loc[participants['ab_test'] == 'recommender_system_test']['group'].value_counts()
a_b_duble['%%'] = a_b_duble['group']/a_b_duble['all']*100
a_b_duble
```

Out[63]:

	group	all	%%
A	439	3634	12.080352
B	344	2717	12.661023

Выводы:

Выявлено 1602 пользователя, участвующих одновременно в двух тестах.

Распределение по группам теста:

- тест interface_eu_test: 819 чел. в группе A и 783 чел. в группе B;
- тест recommender_system_test: 921 чел. в группе A и 681 чел. в группе B.

Мы не знаем одновременно ли проходят 2 теста и как участие одновременно в двух тестах может сказаться на поведении пользователей. Следовательно, для чистоты проведения теста необходимо удалить пользователей участвующих одновременно в двух тестах из датасета participants.

Однако нужно определиться, что значит пользователи, участвующие в двух тестах одновременно.

Пользователи из группы A теста interface_eu_test не видят изменений, вызванных тестом interface_eu_test, слудовательно, могут участвовать и в тесте recommender_system_test.

Пользователи из группы B теста interface_eu_test равномерно распределены между группами A и B теста recommender_system_test:

- 12.08% в группе A и 12.66% в группе B

следовательно, их тоже можно оставить

Учитывая все вышеизложенное и требование ТЗ набрать в тест не менее 6000 пользователей, приму решение не удалять пользователей участвующих в двух тестах одновременно

Больше данные о тесте interface_eu_test нам не нужны, удалим их из датасета participants

In [64]:
```
#Удалим из дамасета participants данные о месте interface_eu_test
participants = participants.loc[participants['ab_test']=='recommender_system_test']
participants['ab_test'].value_counts()
```

Out[64]:
```
recommender_system_test    6351
Name: ab_test, dtype: int64
```

Вывод:

На данном этапе в тесте recommender_system_test участвуют 6351 пользователя, что **соответствует ТЗ**.

3.3.2 Проверка наличия пользователей, участвующих в двух группах теста одновременно

In [65]:
```
# Найдем пользователей, участвующих в двух группах теста одновременно
user_3 = set(participants.loc[participants['group']=='A', 'user_id'].tolist())
user_4 = set(participants.loc[participants['group']=='B', 'user_id'].tolist())
#len(user_1.intersection(user_2))
duble_id_user_group = list(user_3 & user_4)
len(duble_id_user_group)
```

Out[65]: 0

Вывод:

- пользователей, участвующих в двух группах теста одновременно, нет.

3.3.3 Проверка равномерности распределения пользователей по тестовым группам и правильность их формирования

In [66]:
```
# Еще раз посчитаем сколько всего пользователей осталось в нашей тестовой выборке
ab_count_user = participants['user_id'].count()
ab_count_user
```

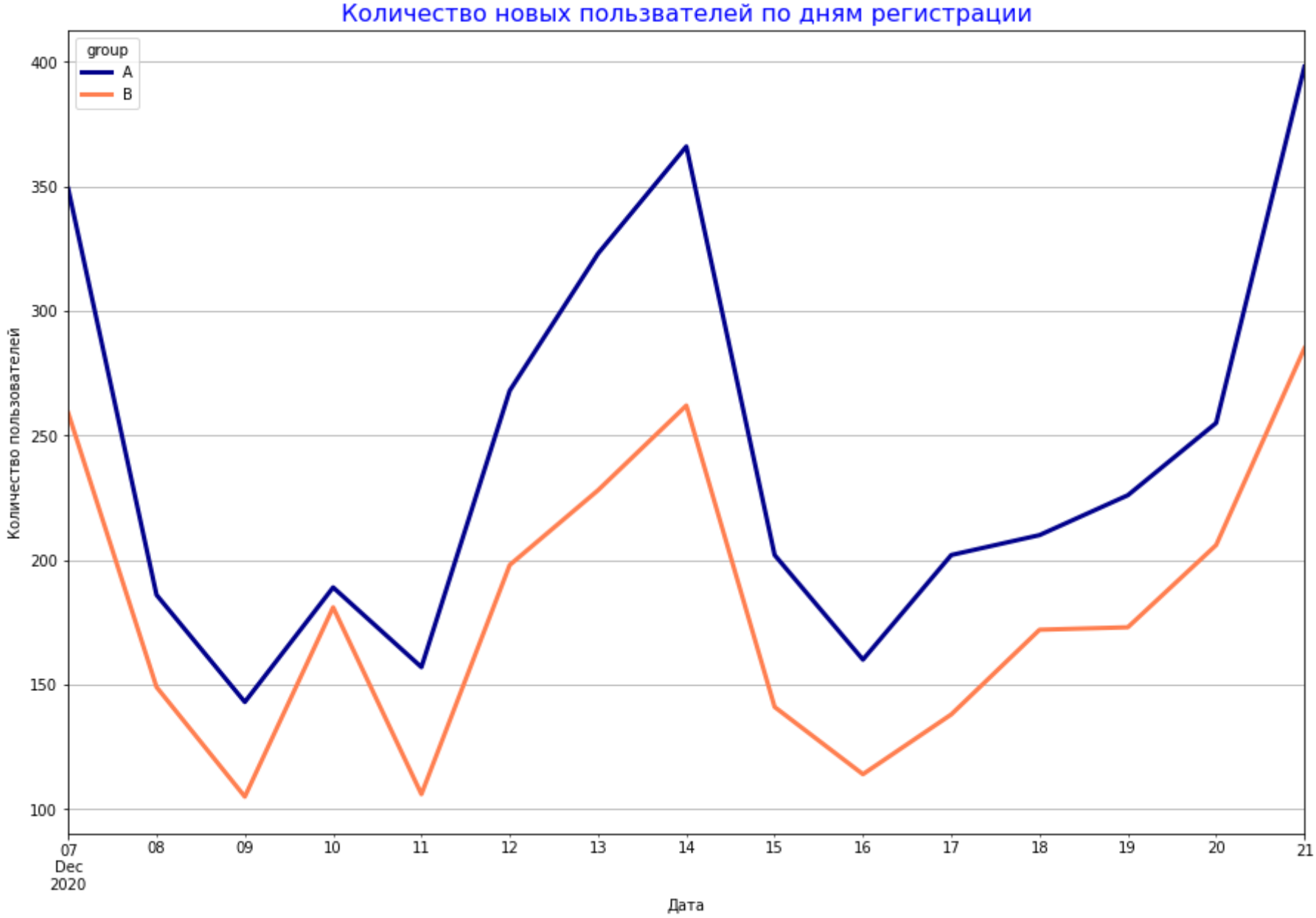
Out[66]: 6351

In [67]:
```
# Посмотрим на равномерность распределения пользователей по группам A и B
a_b_group = pd.DataFrame(participants['group'].value_counts())
a_b_group.columns = ['count']
a_b_group['%_count'] = participants['group'].value_counts(normalize=True).round(2)*100
a_b_group
```

Out[67]:

	count	%_count
A	3634	57.0
B	2717	43.0

In [68]:
```
# Построим график распределения количества новых пользователей по дням регистрации
new_users_day = participants.pivot_table(index='first_date', columns='group', values='user_id', aggfunc='count')
new_users_day.plot(grid=True,
                   figsize=(15, 10),
                   linewidth=3,
                   linestyle='solid',
                   color=['darkblue', 'coral'])
plt.title('Количество новых пользвателей по дням регистрации', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Дата')
plt.ylabel('Количество пользователей')
plt.show()
```

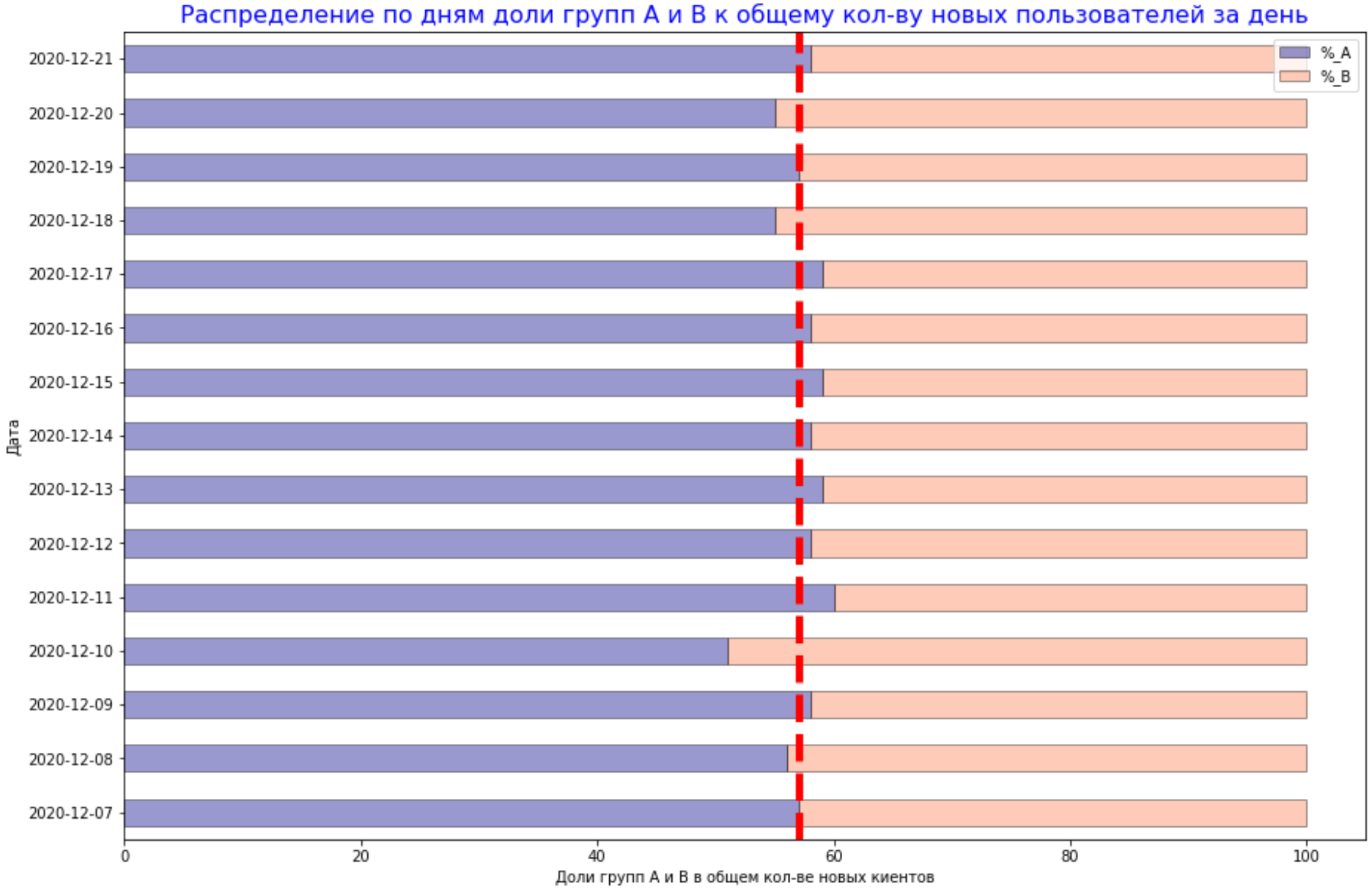

Количество новых пользвателей по дням регистрации
group
A
B
400
350
300
250
200
150
100
Количество пользователей
07
Dec
2020
08
09
10
11
12
13
14
15
16
17
18
19
20
21
Дата

In [69]:
```
# Создадим столбцы с относительными показателями в датасете new_users_day
new_users_day = new_users_day.reset_index()
new_users_day.columns = ['first_date', 'A', 'B']
new_users_day['%_A'] = round(new_users_day['A']/(new_users_day['A']+new_users_day['B'])*100)
new_users_day['%_B'] = round(new_users_day['B']/(new_users_day['A']+new_users_day['B'])*100)
new_users_day['first_date'] = new_users_day['first_date'].dt.date
new_users_day.head(2)
```

Out[69]:

	first_date	A	B	%_A	%_B
0	2020-12-07	349	259	57.0	43.0
1	2020-12-08	186	149	56.0	44.0

In [70]:
```
# Построим график распределения по дням доли групп A и B к общему кол-ву новых пользователей за день
new_users_day[['first_date', '%_A', '%_B']].plot(
  x = 'first_date',
  figsize=(15, 10),
  kind = 'barh',
  stacked = True,
  color = ['darkblue', 'coral'],
  edgecolor="black",
  alpha = 0.4,
  mark_right = True)
plt.axvline (x = 57, color = 'red', linestyle='--', linewidth= 5)
plt.title('Распределение по дням доли групп A и B к общему кол-ву новых пользователей за день', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Доли групп A и B в общем кол-ве новых киентов')
plt.ylabel('Дата')
plt.show()
```


Распределение по дням доли групп A и B к общему кол-ву новых пользователей за день
%_A
%_B
2020-12-21
2020-12-20
2020-12-19
2020-12-18
2020-12-17
2020-12-16
2020-12-15
2020-12-14
2020-12-13
2020-12-12
2020-12-11
2020-12-10
2020-12-09
2020-12-08
2020-12-07
Дата
0
20
40
60
80
100
Доли групп A и B в общем кол-ве новых киентов

Вывод:

- **пользователи неравномерно распределены по тестовым группам**: в группе A - 3385 чел (57%), в группе B - 2533 чел(43%);
- а вот дол новых пользователей из группы A и группы B по дням распределены почти равномерно.

Данный факт нужно учесть при анализе результатов A/B теста.

https://habr.com/ru/articles/664512/#6 (https://habr.com/ru/articles/664512/#6)

Почему АБ тест проходит не точно 50\50? Иногда во время АБ тестирования вы можете заметить, что количество трафика на каждой из переменных не идентично. Это не значит, что с тестом что-то не то, просто случайные выборки работают… случайно. Вспомните про подкидывание монеты. Шанс, что выпадет орел или решка – 50\50, но иногда выпадают три решки подряд. Чем больше трафика пройдет через ваш сайт – тем ближе цифры станут к соотношению 50\50.

3.4 Создание итогового датасета для проведения A/B теста

Задачи раздела:

- создать итоговый датасет, на основе которогоу будет проводится A/B тест;
- создать в новом датасете столбец с lifetime (согласно ТЗ это 14 дней со сдня регистрации нового пользователя);
- узнать количество новых пользователей, совершавших каки-либо события в тестовом периоде.

3.4.1 Анализ активности пользователей, попавших в выборку A/B теста

In [71]:
```
# Посчитаем активных пользователей в выборке A/B теста
ab_count_user_with_events = events.loc[events['user_id'].isin(participants['user_id'])==True, 'user_id'].nunique()
print('Общее кол-во пользователей, попавших в выборку A/B теста, чел.:', ab_count_user)
print()
print('Кол-во пользователей, который попали в выборку  A/B теста и совершали какое-либо действие в тестовом периоде, ч
print()
print('Удельный вес пользователей, участвующих в тесте и  совершивших события, %:', round(ab_count_user_with_events/ab
```

```
Общее кол-во пользователей, попавших в выборку A/B теста, чел.: 6351

Кол-во пользователей, который попали в выборку  A/B теста и совершали какое-либо действие в тестовом периоде, чел.: 3
481

Удельный вес пользователей, участвующих в тесте и  совершивших события, %: 55
```

In [72]:
```
# Посмотрим как распределяются по группам неактивные пользователи
a_b_without_event = participants.loc[participants['user_id'].isin(events['user_id'].unique())==False]
a_b_without_event = pd.DataFrame(a_b_without_event['group'].value_counts())
a_b_without_event['%_total_count'] = round(a_b_without_event['group']/ab_count_user*100)
a_b_without_event
```

Out[72]:

	group	%_total_count
B	1840	29.0
A	1030	16.0

Вывод:

- только 55% пользователей в выборке A/B теста совершали какие-либо действия в тестовом периоде;
- в группе B неактивных пользователей почти в два раза болше(29%), чем в группе A(16%).

Это **негативный показатель для тестируемого изменения рекомендательной системы**.

3.4.2 Создание итогового датасета для проведения A/B теста

Присоединим к датасету events датасет participants по столбцу user_id.

Сохраним получившийся датасет под именем **total_df**

In [73]:
```
# Создадим объединенный датасет total_df и изучем его
total_df = events.merge(participants, on = 'user_id')
total_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23420 entries, 0 to 23419
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   user_id     23420 non-null  object        
 1   event_dt    23420 non-null  datetime64[ns]
 2   event_name  23420 non-null  object        
 3   details     3196 non-null   float64       
 4   event_day   23420 non-null  datetime64[ns]
 5   group       23420 non-null  object        
 6   ab_test     23420 non-null  object        
 7   first_date  23420 non-null  datetime64[ns]
 8   region      23420 non-null  object        
 9   device      23420 non-null  object        
dtypes: datetime64[ns](3), float64(1), object(6)
memory usage: 2.0+ MB
```

In [74]:
```
# Изучим кол-во уникальных значений в разрезе столбцов датасета total_df
total_df.nunique()
```

Out[74]:
```
user_id        3481
event_dt      15628
event_name        4
details           4
event_day        24
group             2
ab_test           1
first_date       15
region            1
device            4
dtype: int64
```

Вывод:

- в объединенный датасет попали только активные пользователи - 3481 человек;
- в объединенныом датасете сохранилось 4 типа событий;
- количество дней в тестируемом периоде - 24 дня (с 07.12.2020 по 30.12.2020);
- количество дней набора клиентов - 15 дня (с 07.12.2020 по 21.12.2020);
- в объединенном датасете сохранилось 2 группы пользователей A и B;
- в объединенном датасете сохранилось 1 тест.

Удалим лишние столбцы из датасета total_df:

- ab_test;
- region;
- device.

Согласно ТЗ мы исследуем только тест recommender_system_test, в котором участвую пользователи только из одного региона EU, заданий связанных с device в ТЗ - нет.

In [75]:
```
# Удалим из итогового датасета лишние столбцы
total_df = total_df[['user_id', 'first_date', 'event_dt', 'event_day', 'event_name', 'details', 'group']]
total_df.head(2)
```

Out[75]:

	user_id	first_date	event_dt	event_day	event_name	details	group
0	831887FE7F2D6CBA	2020-12-07	2020-12-07 06:50:29	2020-12-07	purchase	4.99	A
1	831887FE7F2D6CBA	2020-12-07	2020-12-09 02:19:17	2020-12-09	purchase	99.99	A

3.4.3 Создание в итоговом датасете total_df столбца lifetime

Согласно ТЗ горизонт анализа установлен 14 дней.
Действительно, если дату окончания набора бользователей 21.12.2020 увеличить на 14 дней, получим дату завершения теста 04.01.2021.

Однако, по факту в предоставленных данных пользователи совершают события только до 30.12.2020 г.

Причина отсутствия данных с 31.12.2020 по 04.01.2021 неизвестна.

Информация об осутствии данных доведена до тимлида и поставщика данных раннее.

Создадим столбец lifetime, равный разнице между датой совершения события и датой регистрации пользователя/b>.

Затем удалим строки с lifetime больше 14.

In [76]:
```
# Создадим столбец с датой совершения события event_day на базе столбца event_dt
total_df['event_day'] = pd.to_datetime(total_df['event_day'])
# Создадим столбец lifetime
total_df['lifetime'] = (total_df['event_day'] - total_df['first_date']).dt.days
total_df.head(2)
```

Out[76]:

	user_id	first_date	event_dt	event_day	event_name	details	group	lifetime
0	831887FE7F2D6CBA	2020-12-07	2020-12-07 06:50:29	2020-12-07	purchase	4.99	A	0
1	831887FE7F2D6CBA	2020-12-07	2020-12-09 02:19:17	2020-12-09	purchase	99.99	A	2

In [77]:
```
# Посмотрим на распределение событий по дням lifetime
total_df['lifetime'].value_counts()
```

Out[77]:
```
0     7719
1     3559
2     2451
3     1712
4     1417
5     1149
6      999
7      924
8      725
9      578
10     481
12     341
11     322
13     243
14     208
15     163
16      94
17      78
18      77
20      62
19      54
21      31
22      29
23       4
Name: lifetime, dtype: int64
```

In [78]:
```
# Удалим события(строки) с lifetime больше 14 и изучим количество уникальных значений в столбцах итогового датасета
total_df = total_df.loc[total_df['lifetime']<=14]
total_df.nunique()
```

Out[78]:
```
user_id        3481
first_date       15
event_dt      15231
event_day        23
event_name        4
details           4
group             2
lifetime         15
dtype: int64
```

In [79]:
```
# Изучим итоговый датасет total_df, с которым мы будем проводить исследовательский анализ и анализ а/б теста
total_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22828 entries, 0 to 23419
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   user_id     22828 non-null  object        
 1   first_date  22828 non-null  datetime64[ns]
 2   event_dt    22828 non-null  datetime64[ns]
 3   event_day   22828 non-null  datetime64[ns]
 4   event_name  22828 non-null  object        
 5   details     3123 non-null   float64       
 6   group       22828 non-null  object        
 7   lifetime    22828 non-null  int64         
dtypes: datetime64[ns](3), float64(1), int64(1), object(3)
memory usage: 1.6+ MB
```

In [80]:
```
# Посмотрим на распределение по группам теста
total_df.groupby('group').agg({'user_id':'nunique'})
```

Out[80]:

group	user_id
A	2604
B	877

Вывод:

В total_df:

- 3481 пользователей;
- в группе A - 2604 чел., в группе B - 877 чел.
- 22620 событий.

Обратим внимание на неравномерное распределение пользователей по группам.

3.5 Общий вывод по разделу

Задача раздела - проверить данные на соответствие ТЗ.

Результаты проверки на соответствие данных техническому заданию

Характеристика	Техническое задание	Исходные данные	Корректировка
Название теста	recommender_system_test	recommender_system_test, interface_eu_test	удален тест interface_eu_test

Характеристика	Техническое задание	Исходные данные	Корректировка
группы	A, B	A, B	-
дата запуска	2020-12-07	2020-12-07	-
дата остановки набора новых пользователей	2020-12-21	2020-12-23	удалены 2 дня
дата остановки	2021-01-04	2020-12-30	инф-ция передана тимлиду
аудитория	15% новых пользователей из региона EU	4 региона, 15% новых пользователей EU	удалены 3 региона

Результат - создан итоговый датасет total_df для проведения исследовательского анализа и анализа результатов А/Б теста

4 Исследовательский анализ данных

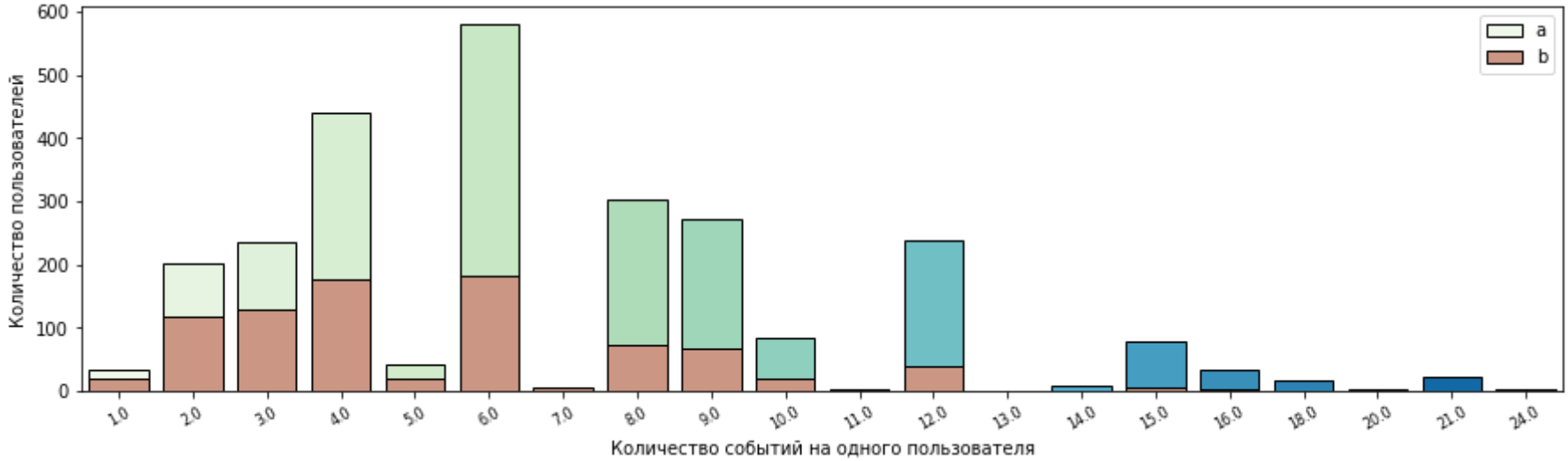
4.1 Анализ распределения количества событий на пользователя в выборках

In [81]:
```
# Сравним данные описательной статистики по кол-ву событий на пользователя из группы A и группы B
id_event_count = total_df.pivot_table(index='user_id', columns = 'group', values = 'event_name', aggfunc = 'count').re
id_event_count.columns = ['user_id', 'a_group', 'b_group']
id_event_count.describe()
```

Out[81]:

	a_group	b_group
count	2604.00000	877.000000
mean	6.90361	5.531357
std	3.84470	3.314281
min	1.00000	1.000000
25%	4.00000	3.000000
50%	6.00000	4.000000
75%	9.00000	8.000000
max	24.00000	24.000000

In [82]:
```
# 0 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B
plt.figure(figsize=(15, 4))
sns.countplot(x=id_event_count['a_group'], palette='GnBu', saturation=0.9, edgecolor = 'black')
sns.countplot(x=id_event_count['b_group'], color='coral', saturation=0.4, edgecolor = 'black')
plt.xlabel('Количество событий на одного пользователя')
plt.ylabel('Количество пользователей')
plt.xticks(rotation=30, size=8)
plt.legend('ab')
plt.show()
```



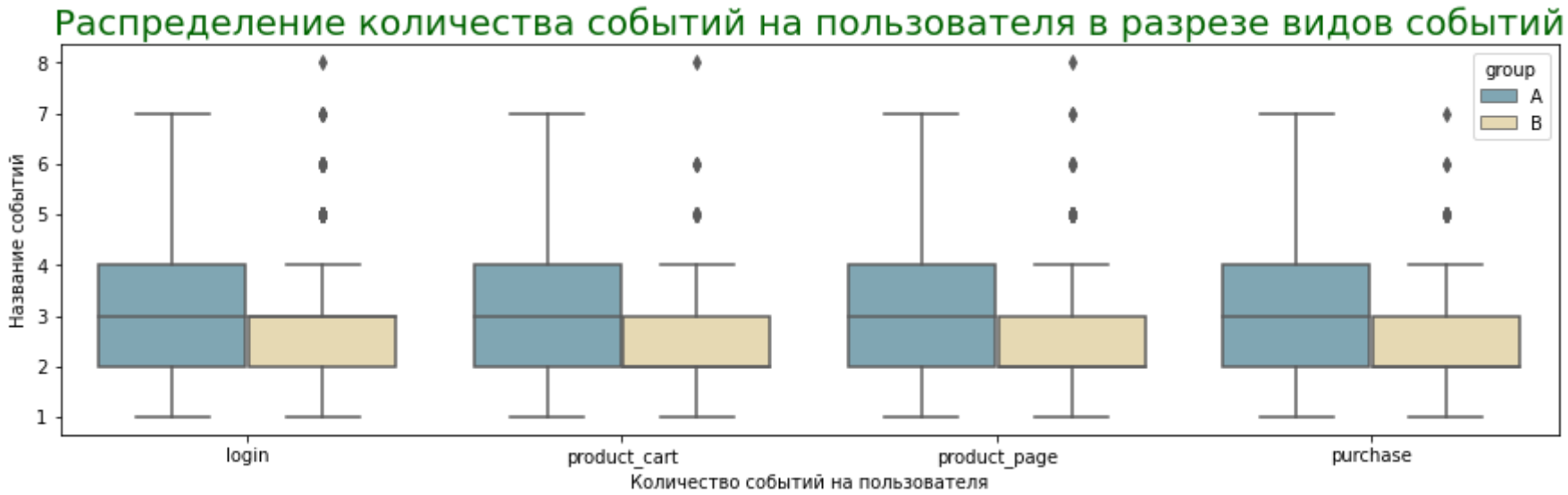

```
# 1 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B
for i in id_event_count.columns[1:]:
    plt.figure(figsize=(10, 4))
    sns.boxplot(x = id_event_count[i], palette='GnBu')
    plt.xlabel('Количество событий на пользователя')
    plt.title(f'Распределение количества событий на пользователя "{id_event_count[i].name}"', size=20,
color='darkgreen')
    plt.rcParams['axes.grid'] = False
    plt.show()
    plt.figure(figsize=(11, 4))
    sns.countplot(x=id_event_count[i], palette='GnBu', saturation=0.9, edgecolor = 'black')
    plt.xlabel('Количество событий на пользователя')
    plt.ylabel('Количество')
```

```
    plt.xticks(rotation=30, size=8)
    plt.show()
```

```
# 2 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B
plt.figure(figsize=(15, 4))
id_event_count2 = total_df.pivot_table(index=['group', 'user_id', ], values = 'event_name', aggfunc =
'count').reset_index()
id_event_count2.columns = ['group', 'user_id', 'count']
sns.boxplot(data = id_event_count2, y = 'group', x = 'count')
plt.title(f'Распределение количества событий на пользователя "{id_event_count[i].name}"', size=20, color='darkgreen')
plt.xlabel('Количество событий на пользователя')
plt.ylabel('Название группы')
plt.show()
```

```
# 3 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B
plt.figure(figsize=(15, 4))
sns.boxplot(data=id_event_count[['a_group', 'b_group']], orient="h")
plt.title(f'Распределение количества событий на пользователя "{id_event_count[i].name}"', size=20, color='darkgreen')
plt.xlabel('Количество событий на пользователя')
plt.ylabel('Название группы')
plt.show()
```

In [83]:
```
# 1 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B в разрезе ви
plt.figure(figsize=(15, 4))
id_event_count3 = total_df.pivot_table(index=['event_name', 'group', 'user_id'], values = 'event_dt', aggfunc = 'count
id_event_count3.columns = ['event_name', 'group', 'user_id', 'count']
sns.boxplot(data = id_event_count3, y = 'count', palette = "blend:#7AB,#EDA", x = 'event_name', hue = 'group', dodge=T
plt.title('Распределение количества событий на пользователя в разрезе видов событий', size=20, color='darkgreen')
plt.xlabel('Количество событий на пользователя')
plt.ylabel('Название событий')
plt.show()
```



In [84]:
```
# Сравним данные описательной статистики по кол-ву событий на пользователя из группы A и группы B в разрезе видов собы
id_event_count4 = total_df.pivot_table(index = 'user_id', columns = ['group', 'event_name'], values = 'event_dt', aggf
id_event_count4.columns = ['user_id', 'a_login', 'a_product_cart', 'a_product_page', 'a_purchase', 'b_login', 'b_produ
id_event_count4.describe()
```

Out[84]:

	a_login	a_product_cart	a_product_page	a_purchase	b_login	b_product_cart	b_product_page	b_purchase
count	2604.000000	782.000000	1685.000000	833.000000	876.000000	244.000000	493.000000	249.000000
mean	3.059908	3.049872	3.041543	3.000000	2.676941	2.561475	2.549696	2.506024
std	1.209937	1.184111	1.222583	1.201962	1.216822	1.183004	1.199621	1.205232
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
50%	3.000000	3.000000	3.000000	3.000000	3.000000	2.000000	2.000000	2.000000
75%	4.000000	4.000000	4.000000	4.000000	3.000000	3.000000	3.000000	3.000000
max	7.000000	7.000000	7.000000	7.000000	8.000000	8.000000	8.000000	7.000000

```
# 2 Визуализируем данные описательной статистики по кол-ву событий на пользователя из группы A и группы B в разрезе
видов событий
for i in total_df['event_name'].unique():
    plt.figure(figsize=(15, 4))
    id_event_count2 = total_df.loc[total_df['event_name'] == i].pivot_table(index=['group', 'user_id', ], values =
'event_name', aggfunc = 'count').reset_index()
    id_event_count2.columns = ['group', 'user_id', 'count']
    sns.boxplot(data = id_event_count2, y = 'group', x = 'count')
    plt.title(f'Распределение количества событий на пользователя событие {i}', size=20, color='darkgreen')
    plt.xlabel('Количество событий на пользователя')
    plt.ylabel('Название групп')
    plt.show()
```

Вывод:

- медианное количество событий в группе A на 1 пользователя - 6, в группе B - 4;

- медианное количество события purchase в группе A на 1 пользователя - 3, в группе B - 2.

Количество событий в группе B на одного пользователя примерно на 33% ниже, чем в группе A.

Это плохой показатель для тестируемой рекомендательной сичтемы.

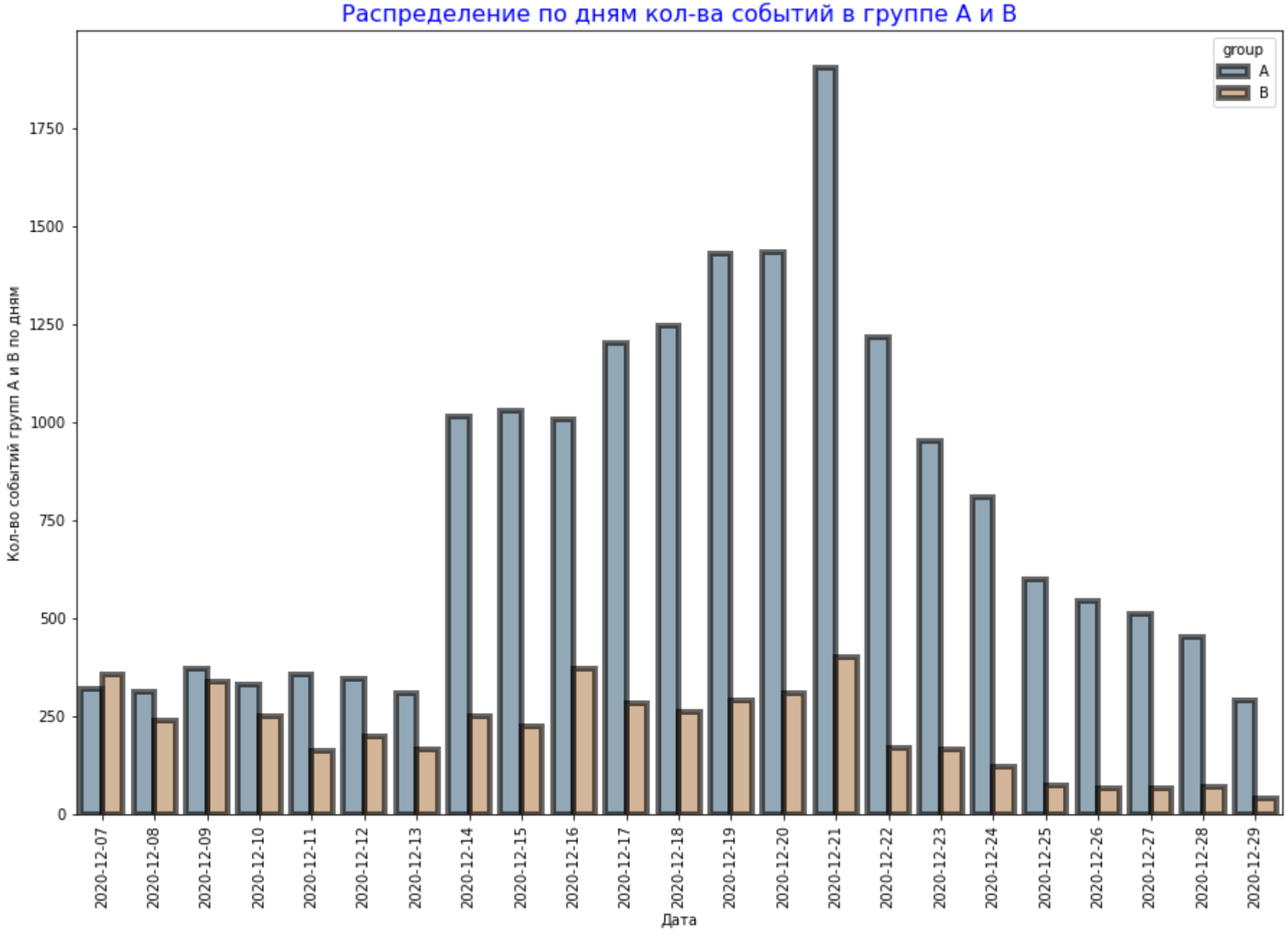
4.2 Анализ распределения по дням числа событий в выборках

In [85]:
```
# Построим таблицу 1 распределения по дням кол-ва событий в разрезе групп
total_df['event_day'] = total_df['event_day'].dt.date
day_events_group = total_df.pivot_table(index = ['event_day', 'group'], values = 'event_name', aggfunc = 'count').rese
day_events_group.columns = ['event_day', 'group', 'count']
day_events_group.head(2)
```

Out[85]:

	event_day	group	count
0	2020-12-07	A	318
1	2020-12-07	B	356

In [86]:
```
# Визуализируем распределение по дням и группам кол-ва событий
plt.figure(figsize=(15, 10))
sns.color_palette("blend:#7AB,#EDA", as_cmap=True)
sns.barplot(data = day_events_group,
            x = 'event_day',
            #palette = 'mako',
            saturation=0.4,
            alpha = 0.6,
            y = 'count',
            hue = 'group',
            linewidth=4,
            edgecolor="black",)
plt.xticks(rotation=45)
plt.title('Распределение по дням кол-ва событий в группе A и B', size=16, color='blue')
plt.xticks(rotation=90)
plt.ylabel('Кол-во событий групп A и B по дням')
plt.xlabel('Дата')
plt.show();
```

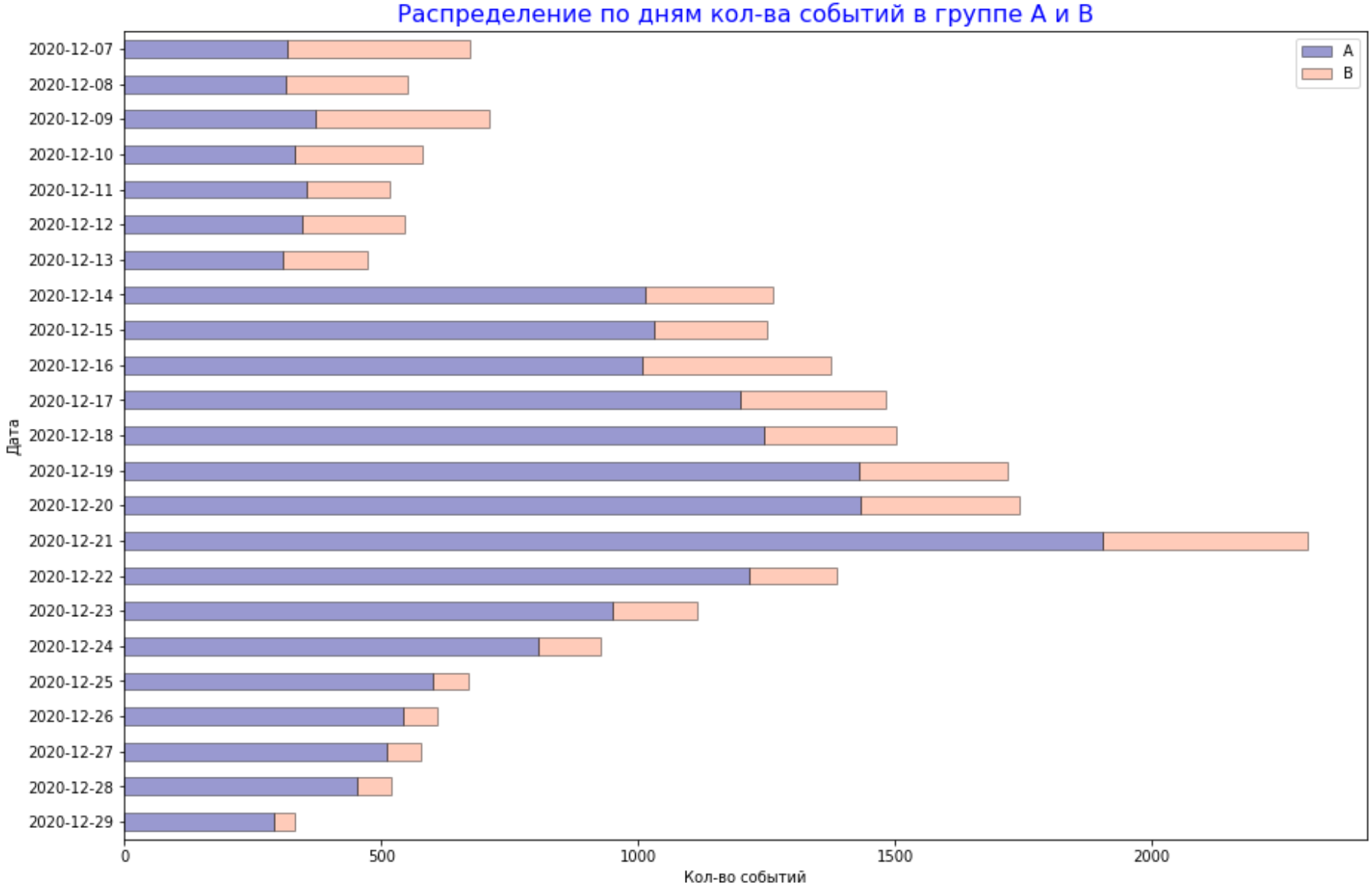

Распределение по дням кол-ва событий в группе A и B
group
A
B
Кол-во событий групп A и B по дням
0
250
500
750
1000
1250
1500
1750
2020-12-07
2020-12-08
2020-12-09
2020-12-10
2020-12-11
2020-12-12
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
Дата

In [87]:
```
# Построим таблицу распределения по дням кол-ва событий в разрезе групп
a_b_events_day = total_df.pivot_table(index = 'event_day', columns = 'group', values = 'event_name', aggfunc = 'count'
a_b_events_day.columns = ['event_day', 'A', 'B']
a_b_events_day['%_A'] = round(a_b_events_day['A']/(a_b_events_day['A'] + a_b_events_day['B'])*100)
a_b_events_day['%_B'] = round(a_b_events_day['B']/(a_b_events_day['A'] + a_b_events_day['B'])*100)
a_b_events_day = a_b_events_day.sort_values(by='event_day', ascending = False)
a_b_events_day.head(2)
```

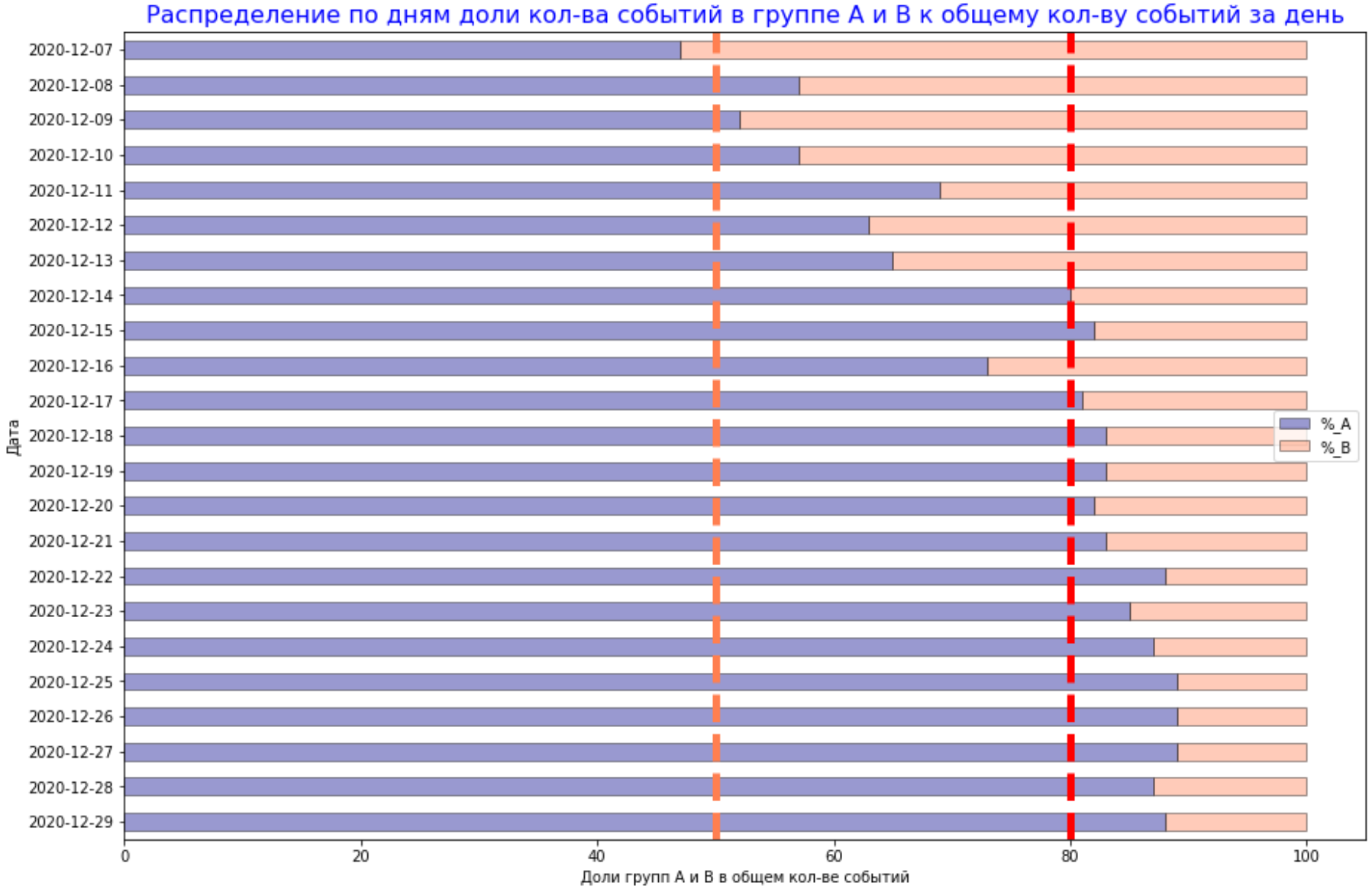
Out[87]:

	event_day	A	B	%_A	%_B
22	2020-12-29	291	41	88.0	12.0
21	2020-12-28	452	69	87.0	13.0

In [88]:
```
# Построим график распределения по дням кол-ва событий в группе A и B
a_b_events_day[['event_day', 'A', 'B']].plot(
  x = 'event_day',
  figsize=(15, 10),
  kind = 'barh',
  stacked = True,
  color = ['darkblue', 'coral'],
  edgecolor="black",
  alpha = 0.4,
  mark_right = True)
plt.title('Распределение по дням кол-ва событий в группе A и B', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Кол-во событий')
plt.ylabel('Дата')
plt.show()
```


Распределение по дням кол-ва событий в группе A и B
A
B
2020-12-07
2020-12-08
2020-12-09
2020-12-10
2020-12-11
2020-12-12
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
Дата
0
500
1000
1500
2000
Кол-во событий

In [89]:
```
# Построим график распределения по дням доли кол-ва событий в группе A и B к общему кол-ву событий за день
a_b_events_day[['event_day', '%_A', '%_B']].plot(
  x = 'event_day',
  figsize=(15, 10),
  kind = 'barh',
  stacked = True,
  color = ['darkblue', 'coral'],
  edgecolor="black",
  alpha = 0.4,
  mark_right = True)
plt.axvline (x = 80, color = 'red', linestyle='--', linewidth= 5)
plt.axvline (x = 50, color = 'coral', linestyle='--', linewidth= 5)
plt.title('Распределение по дням доли кол-ва событий в группе A и B к общему кол-ву событий за день', size=16, color='
plt.xticks(rotation=360)
plt.xlabel('Доли групп A и B в общем кол-ве событий')
plt.ylabel('Дата')
plt.show()
```


Распределение по дням доли кол-ва событий в группе A и B к общему кол-ву событий за день
2020-12-07
2020-12-08
2020-12-09
2020-12-10
2020-12-11
2020-12-12
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
Дата
%_A
%_B
0
20
40
60
80
100
Доли групп A и B в общем кол-ве событий

Вывод:

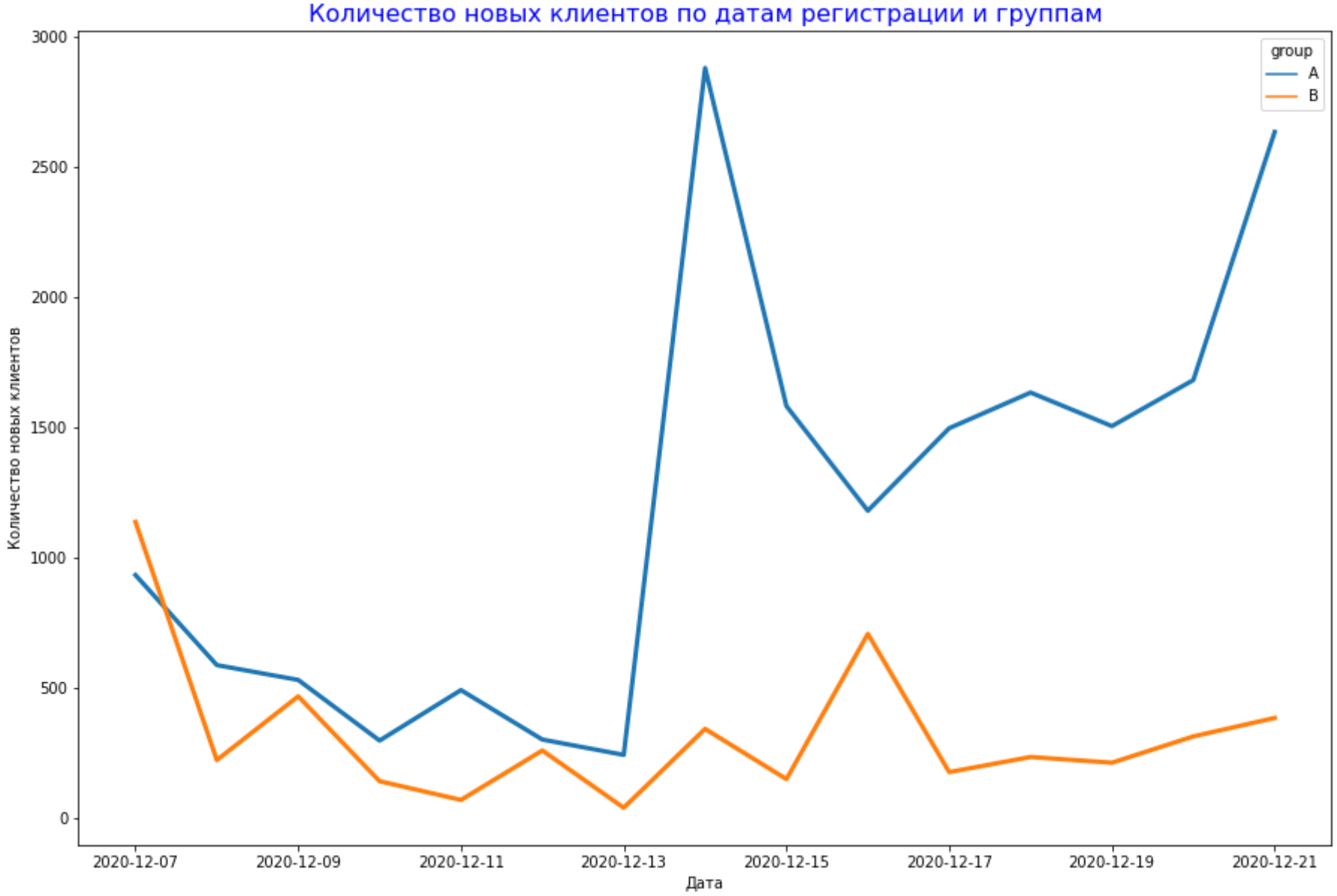
- пик кол-ва событий совпадает в группах А и В - 21.12.2020;
- только в певый день теста удельный вес кол-ва событий, совершенных пользователями из группы В, больше 50%;
- с течением времени удельный вес событий, совершенных пользователяи из группы А растет, а группы В - снижается. Уже с 14.12.2020 г. удельный вес событий, совершенных пользователми из группы, превышает 80% от общего кол-ва событий за день.

Снижение со временем доли событий, совершаемых пользователями из группы В, также является негативным показателем для тестируемого изменения рекомендательной системы.

Выясним причину резкого увеличения количества событий, совершенных клиентами из группы А. Скорее всего это связано с увеличением числа новых пользователей, зарегистрировавшихся и попавших в группу А. Проверим.

In [90]:

```
# Визуализируем кол-во новых клиентов по датам регитрации и группам
day_new_user = total_df.pivot_table(index=['first_date', 'group'], values = 'user_id', aggfunc = 'count').reset_index(
day_new_user.columns = ['first_date', 'group', 'count']
plt.figure(figsize=(15, 10))
sns.lineplot(data=day_new_user,
             x='first_date',
             y = "count",
             hue = 'group',
             linewidth=3,
             linestyle='solid',
             )
plt.title('Количество новых клиентов по датам регистрации и группам', size=16, color='blue')
plt.xticks(rotation=360)
plt.xlabel('Дата')
plt.ylabel('Количество новых клиентов')
plt.show()
```


Количество новых клиентов по датам регистрации и группам
group
A
B
Количество новых клиентов
3000
2500
2000
1500
1000
500
0
2020-12-07
2020-12-09
2020-12-11
2020-12-13
2020-12-15
2020-12-17
2020-12-19
2020-12-21
Дата

Вывод:

Действительно 14.12.2020 произошел резкий рост регистрации новых пользователей в группе A, в группе B увеличения количества зарегистрировавшихся пользователей не было. Рост кол-ва новых пользователей в группе A привел к росту количества совершенных ими событий.

4.3 Анализ конверсии в выборках на разных этапах воронки

In [91]:
```
# Создадим таблицу кол-во уникальных пользователей и событий в разрезе видов без разделения групп A и B
user_name_total = total_df.pivot_table(index='event_name', values = 'user_id', aggfunc = 'nunique')
user_name_total.columns = ['user_count']
user_name_total['events_count'] = total_df['event_name'].value_counts()
user_name_total['%_user_count'] = round(user_name_total['user_count']/user_name_total['user_count'].sum()*100)
display(user_name_total)
user_name_total_a = total_df.loc[total_df['group']=='A'].pivot_table(index='event_name', values = 'user_id', aggfunc =
user_name_total_a.columns = ['user_a_count']
user_name_total_a['events_a_count'] = total_df.loc[total_df['group']=='A']['event_name'].value_counts()
user_name_total_a['%_user_a_count'] = round(user_name_total_a['user_a_count']/user_name_total_a['user_a_count'].sum()*
display(user_name_total_a)
user_name_total_b = total_df.loc[total_df['group']=='B'].pivot_table(index='event_name', values = 'user_id', aggfunc =
user_name_total_b.columns = ['user_b_count']
user_name_total_b['events_b_count'] = total_df.loc[total_df['group']=='B']['event_name'].value_counts()
user_name_total_b['%_user_b_count'] = round(user_name_total_b['user_b_count']/user_name_total_b['user_b_count'].sum()*
display(user_name_total_b)
```

event_name	user_count	events_count	%_user_count
login	3480	10313	45.0
product_cart	1026	3010	13.0
product_page	2178	6382	28.0
purchase	1082	3123	14.0

event_name	user_a_count	events_a_count	%_user_a_count
login	2604	7968	44.0
product_cart	782	2385	13.0
product_page	1685	5125	29.0
purchase	833	2499	14.0

event_name	user_b_count	events_b_count	%_user_b_count
login	876	2345	47.0
product_cart	244	625	13.0
product_page	493	1257	26.0
purchase	249	624	13.0

Воронка событий:

1. login;
2. product_page;
3. product_cart;
4. purchase

In [92]:
```
# Создадим таблицу для построения воронки событий в разрезе групп: кол-во событий в разрезе видов событий и групп
event_name_group = total_df.pivot_table(index=['event_name', 'group'], values = 'user_id', aggfunc = 'nunique')
event_name_group = event_name_group.rename({'login':'1_login', 'product_cart':'3_product_cart', 'product_page':'2_prod
event_name_group = event_name_group.sort_values(by = 'event_name')
event_name_group
```

Out[92]:

	event_name	group	user_id
0	1_login	A	2604
1	1_login	B	876
4	2_product_page	A	1685
5	2_product_page	B	493
2	3_product_cart	A	782
3	3_product_cart	B	244
6	4_purchase	A	833
7	4_purchase	B	249

```
# Визуализация 1 воронки продаж в разрезе групп
fig = px.funnel(event_name_group,
                x='user_id',
                y='event_name',
                color = 'group')
fig.show()
```

In [93]:
```
# Визуализация 2 воронки продаж в разрезе групп
fig = go.Figure()

fig.add_trace(go.Funnel(
        name = 'A',
        y = event_name_group.loc[event_name_group['group'] == 'A']['event_name'],
        x = event_name_group.loc[event_name_group['group'] == 'A']['user_id'],
        textposition = "inside",
        textinfo = "value+percent initial",
        opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal"],
        "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "wheat", "coral"]}},
        connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
         )
fig.add_trace(go.Funnel(
        name = 'B',
        y = event_name_group.loc[event_name_group['group'] == 'B']['event_name'],
        x = event_name_group.loc[event_name_group['group'] == 'B']['user_id'],
        textposition = "inside",
        textinfo = "value+percent initial",
        opacity = 0.65, marker = {"color": ["teal", "tan", "lightsalmon", "deepskyblue"],
        "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "wheat", "coral"]}},
        connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
         )
fig.update_layout(title_text = 'Условная воронка событий',title_y=0.9, title_x= 0.55)
fig.show()
```

Вывод:

- наибольшее кол-во клиентов "отваливается" на этапах login - product_page и product_page - product_cart;
- в группе A на этап product_page прошли 65% клиентов, в группе B - 56%;
- в группе A на этап product_cart прошли 30% клиентов, в группе B - 28%;

- на этапе product_cart-purchase график не показывает потерь клиентов, даже наоборот в группе а % в purchase выше, чем в product_cart, следовательно, есть способ совершить покупку минуя этап product_cart.

На этапе login - product_page группа B показывает результат хуже группы A на 9%. Это негативный показатель для тестируемого изменения рекомендательной системы. Возможно, именно на этом этапе находится слабое место новой рекомендательной системы.

По результатам построеной воронки событи также видно, что цель, поставленная перед внедрением новой рекомендательной системы по ТЗ, а именно:

"Ожидаемый эффект: за 14 дней с момента регистрации в системе пользователи покажут улучшение каждой метрики не менее, чем на 10%"

не достигнута. .

4.4 Описание особенностей данных, которые нужно учесть, прежде чем приступать к A/B-тестированию

Особенности, выявленные в ходе предварительного и исследовательского анализа, которые необходимо учитывать при проведении А/Б теста:

- неравномерное распределение пользователей по группам A и B: 2604 чел. в группе A и 877 чел в группе B;
- в тестовой группе активных пользователей всего 3481 чел. из отобранных для участия в тесте 6351, в ТЗ предполагалось участие в тесте 6000 чел.;
- время проведения теста совпадает с временем проведения маркетингового события Christmas&New Year Promo, проводимого с 25.12.2020 г;
- в исходных данных нет информации о событиях, совершенных пользователями в период с 31.12.2020 до 04.01.2021;
- в A/B тесте принимает участие далеко не вся аудитория сайта, ТЗ предпалагало участие только 15% новых пользователей из одного из 4 регионов, использующих сайт, по факту доля участвующих в тесте пользователей еще ниже;

- тест проводится с 7 по 30 декабря и на поведение пользователей сильно сказываются внешние факторы, а именно традиции проведения Рождеста в EU регионе.

5 Оценка результатов A/B теста

5.1 Анализ кумулятивных метрик

Остался неизученным вопрос эффекта от внедрения усовершенствованной рекомендательной системы.
Согласно ТЗ:

"ожидаемый эффект: за 14 дней с момента регистрации пользователи покажут улучшение каждой метрики не менее, чем на 10%: о конверсии в просмотр карточек товаров — событие product_page, о просмотры корзины — product_cart, о покупки — purchase."

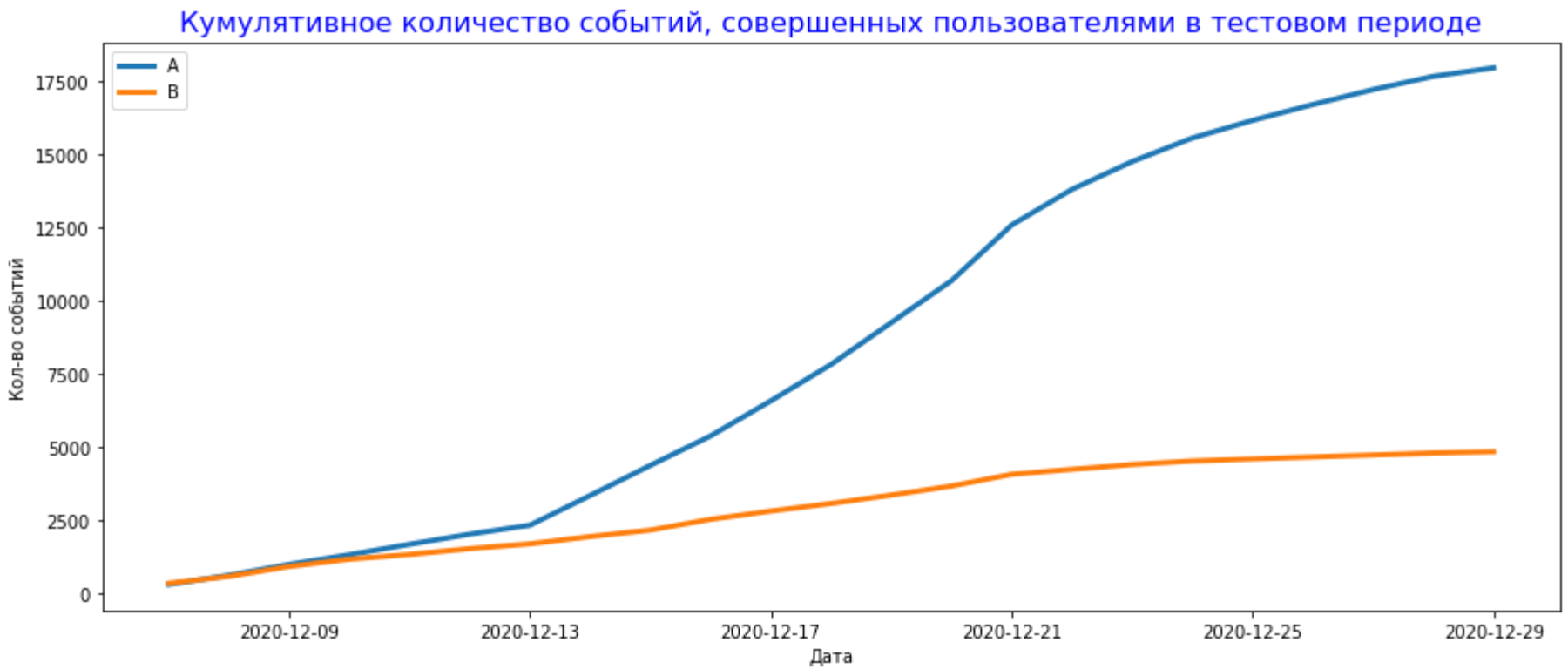
Проанализирузируем кумулятивную метрику: количество событий, совершенных пользователями.

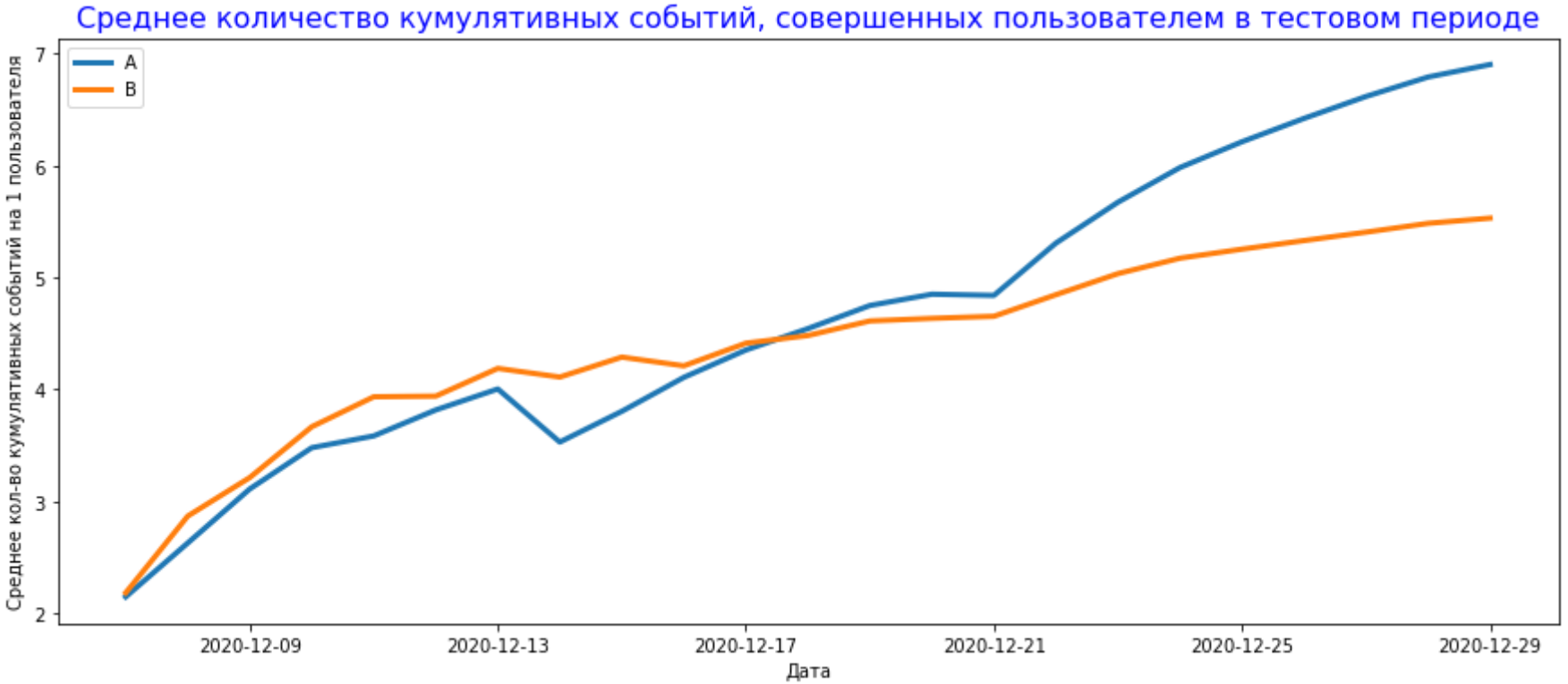
In [94]:
```
total_df['event_day'] = pd.to_datetime(total_df['event_day'])
```

In [95]:
```
def cumulative_metrics(df):
    # Создадим массив уникальных пар значений дат и групп теста
    datesGroups = df[['event_day','group']].drop_duplicates()
    # Получим агрегированные кумулятивные по дням данные о событиях
    cumulativeData = datesGroups.apply(lambda x: df[np.logical_and(df['event_day'] <= x['event_day'], df['group'] == x
    # Создадим датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе A
    cumulativeRevenueA = cumulativeData[cumulativeData['group']=='A'][['event_day','event_name', 'user_id']]
    # Создадим датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B
    cumulativeRevenueB = cumulativeData[cumulativeData['group']=='B'][['event_day','event_name', 'user_id']]

    # Построим график кол-во событий в группе A
    plt.figure(figsize=(15, 6))
    plt.plot(cumulativeRevenueA['event_day'], cumulativeRevenueA['event_name'], label='A', linewidth=3)
    # Построим график кол-во событий в группе B
    plt.plot(cumulativeRevenueB['event_day'], cumulativeRevenueB['event_name'], label='B', linewidth=3)
    plt.title('Кумулятивное количество событий, совершенных пользователями в тестовом периоде', size=16, color='blue')
    plt.xticks(rotation=360)
    plt.ylabel('Кол-во событий')
    plt.xlabel('Дата')
    plt.legend()
    plt.show()

    # Построим график среднего кол-ва кумулятивных событий на одного пользователя в разрезе групп
    plt.figure(figsize=(15, 6))
    plt.plot(cumulativeRevenueA['event_day'], cumulativeRevenueA['event_name']/cumulativeRevenueA['user_id'], label='A
    plt.plot(cumulativeRevenueB['event_day'], cumulativeRevenueB['event_name']/cumulativeRevenueB['user_id'], label='B
    plt.title('Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде', size=16, color=
    plt.xticks(rotation=360)
    plt.ylabel('Среднее кол-во кумулятивных событий на 1 пользователя')
    plt.xlabel('Дата')
    plt.legend()
    plt.show()

    plt.figure(figsize=(15, 6))
    # Соберем данные в одном датафрейме
    mergedCumulativeRevenue = cumulativeRevenueA.merge(cumulativeRevenueB, left_on='event_day', right_on='event_day',

    # Построим отношение среднего кол-ва событий
    plt.plot(mergedCumulativeRevenue['event_day'], (mergedCumulativeRevenue['event_nameB']/mergedCumulativeRevenue['us

    # Добавим ось X
```

```
    plt.axhline(y=0, color='black', linestyle='--', linewidth=3)
    plt.title('График относительного различия для среднего количества событий на одного пользователя в тестовом период
    plt.xticks(rotation=360)
    plt.ylabel('Относительное различие')
    plt.xlabel('Дата')
    plt.show()
```

In [96]:
```
# Изучим кумулятивные метрики по общему кол-ву событий
cumulative_metrics(total_df)
```




Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде
A
B
Среднее кол-во кумулятивных событий на 1 пользователя
7
6
5
4
3
2
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

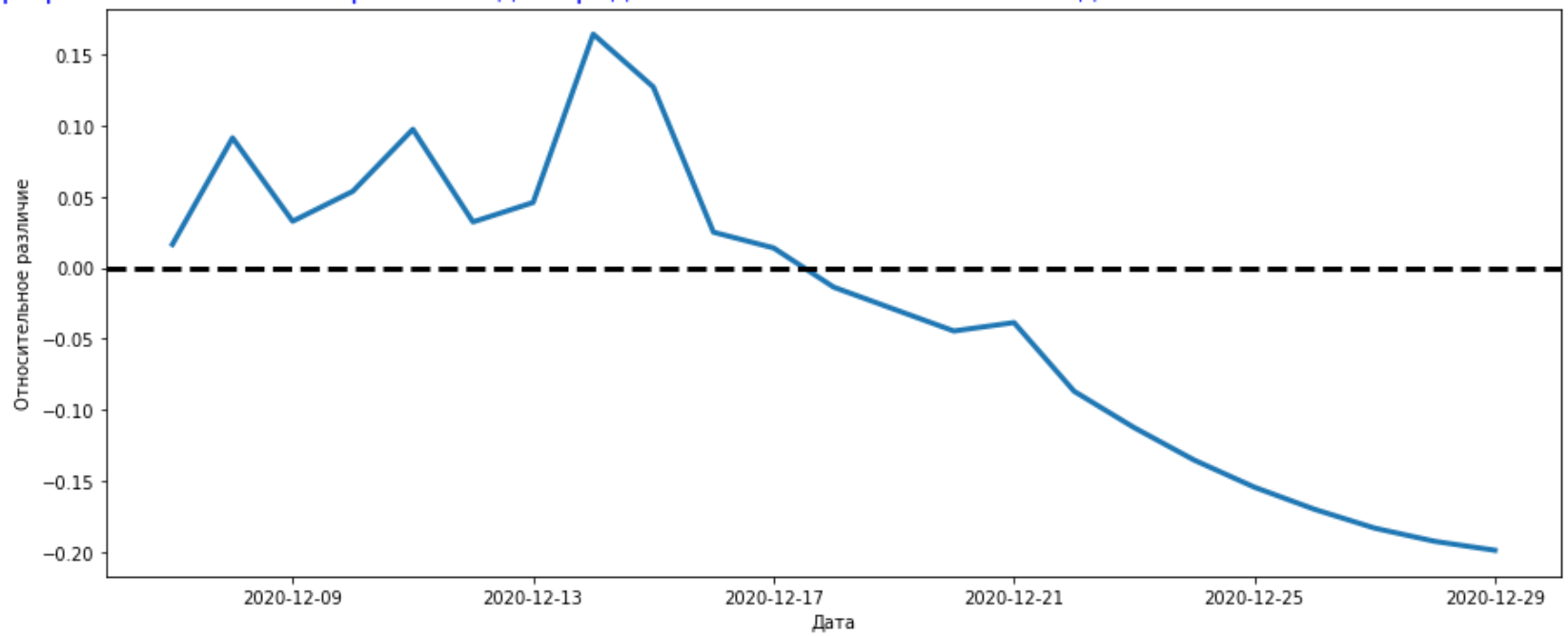
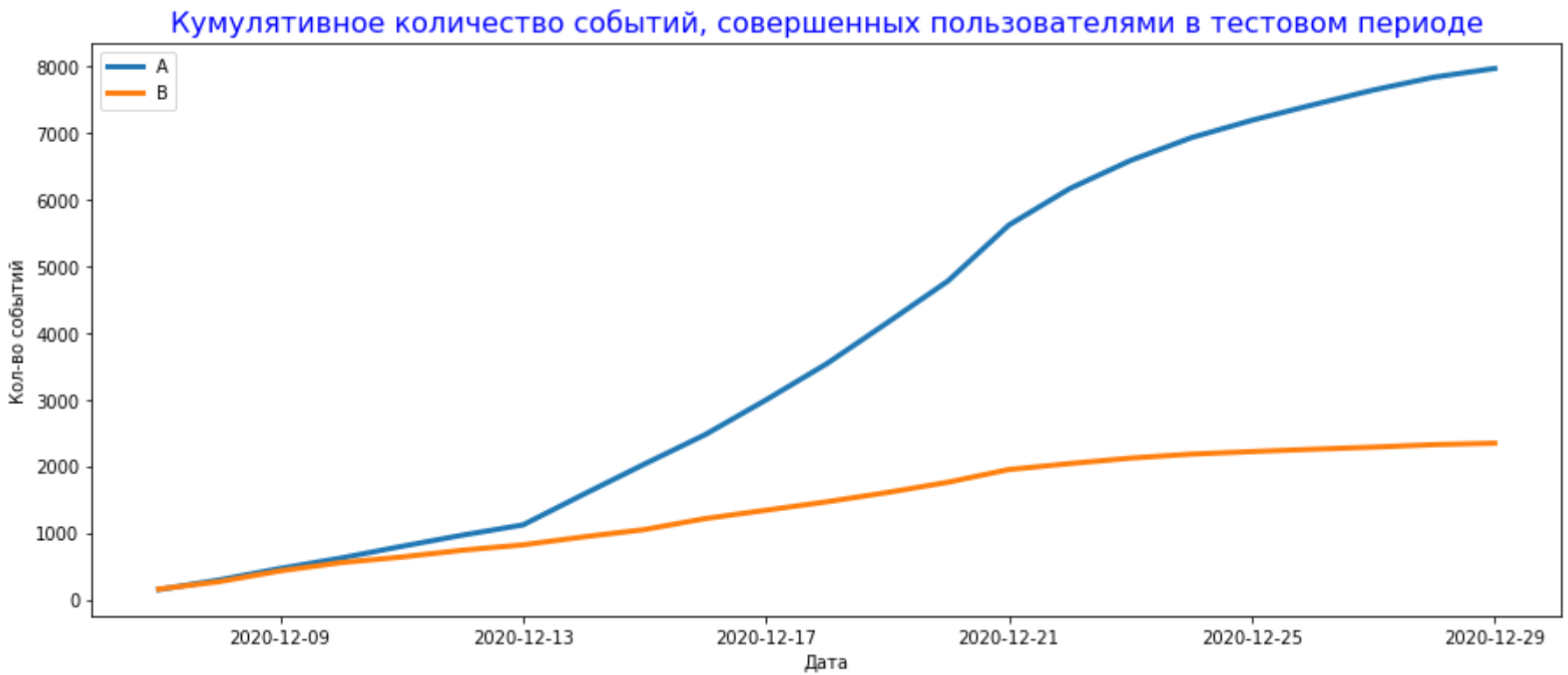
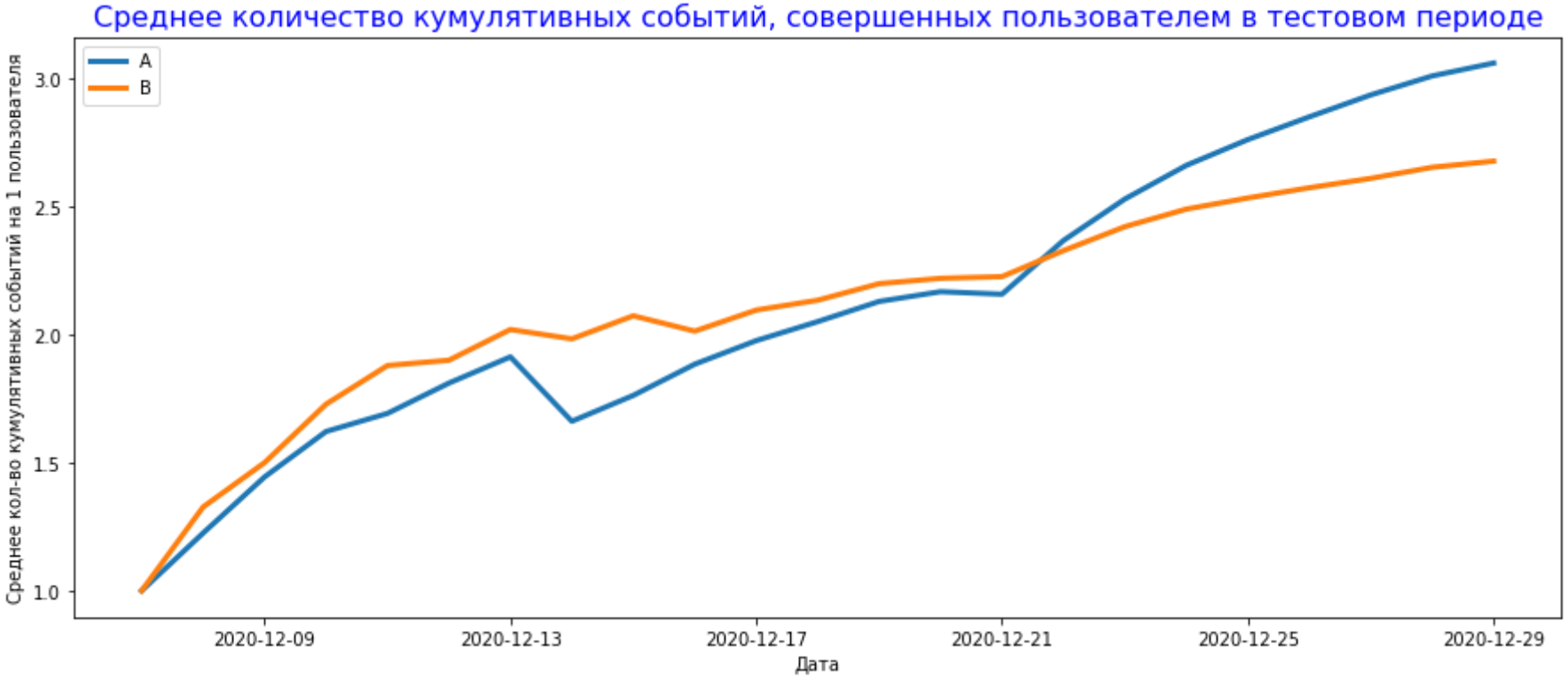


График относительного различия для среднего количества событий на одного пользователя в тестовом периоде
Относительное различие
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

In [97]:
```
# Изучим кумулятивные метрики по кол-ву событий на этапе воронки 'login'
cumulative_metrics(total_df.loc[total_df['event_name'] == 'login'])
```




Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде
A
B
Среднее кол-во кумулятивных событий на 1 пользователя
3.0
2.5
2.0
1.5
1.0
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

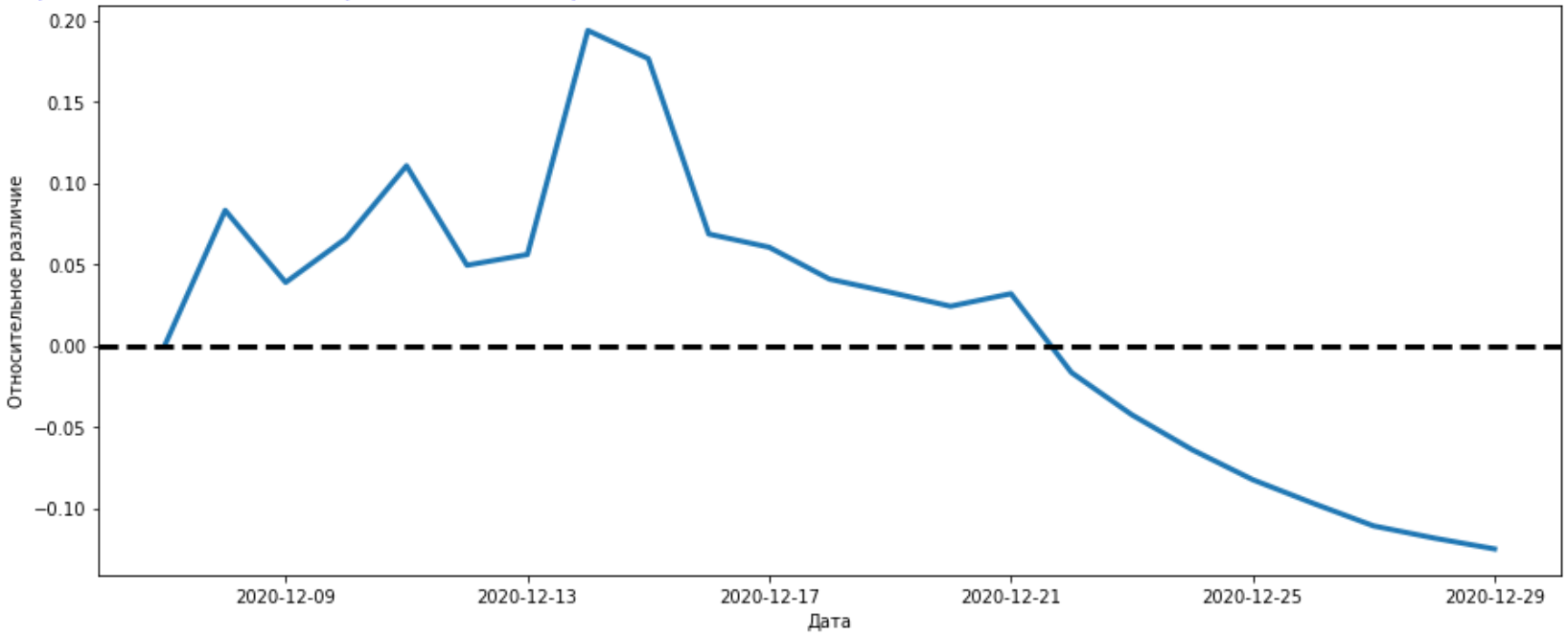
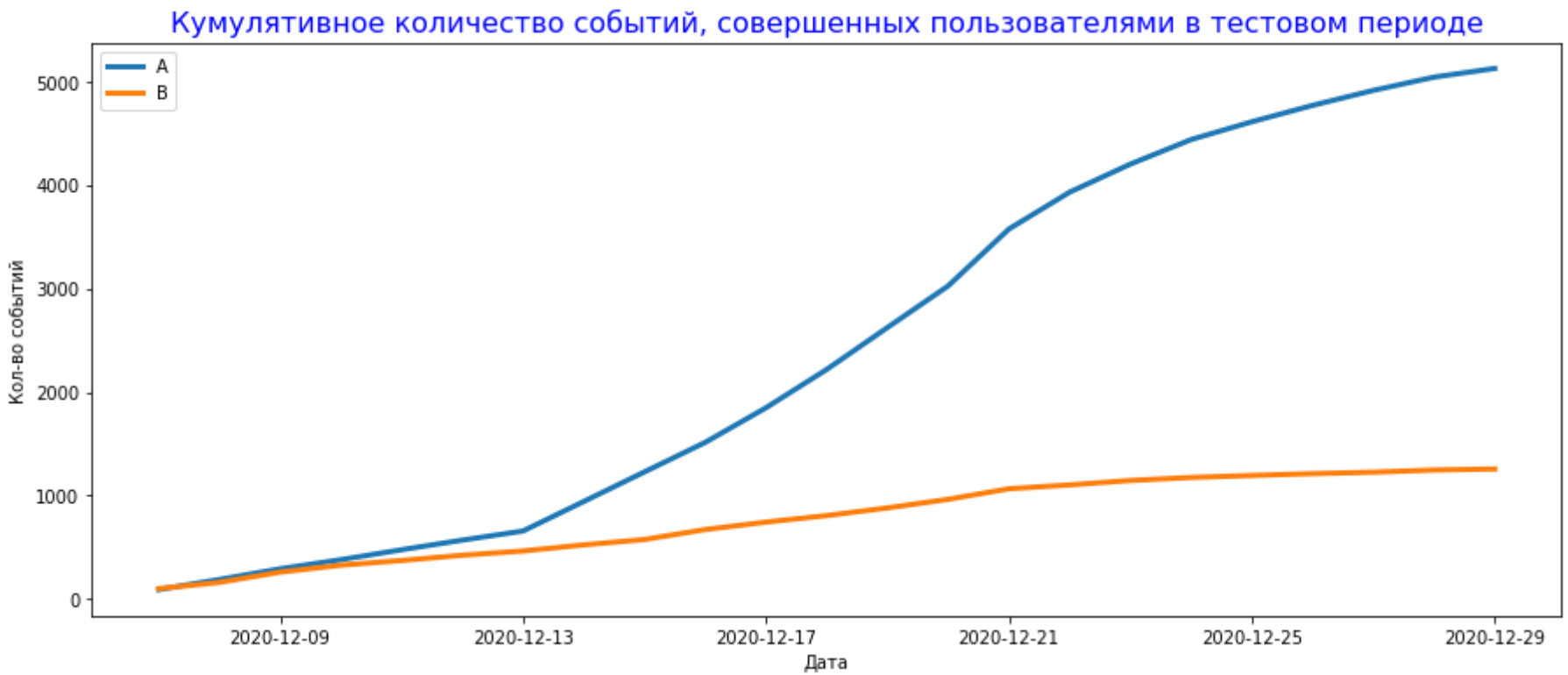
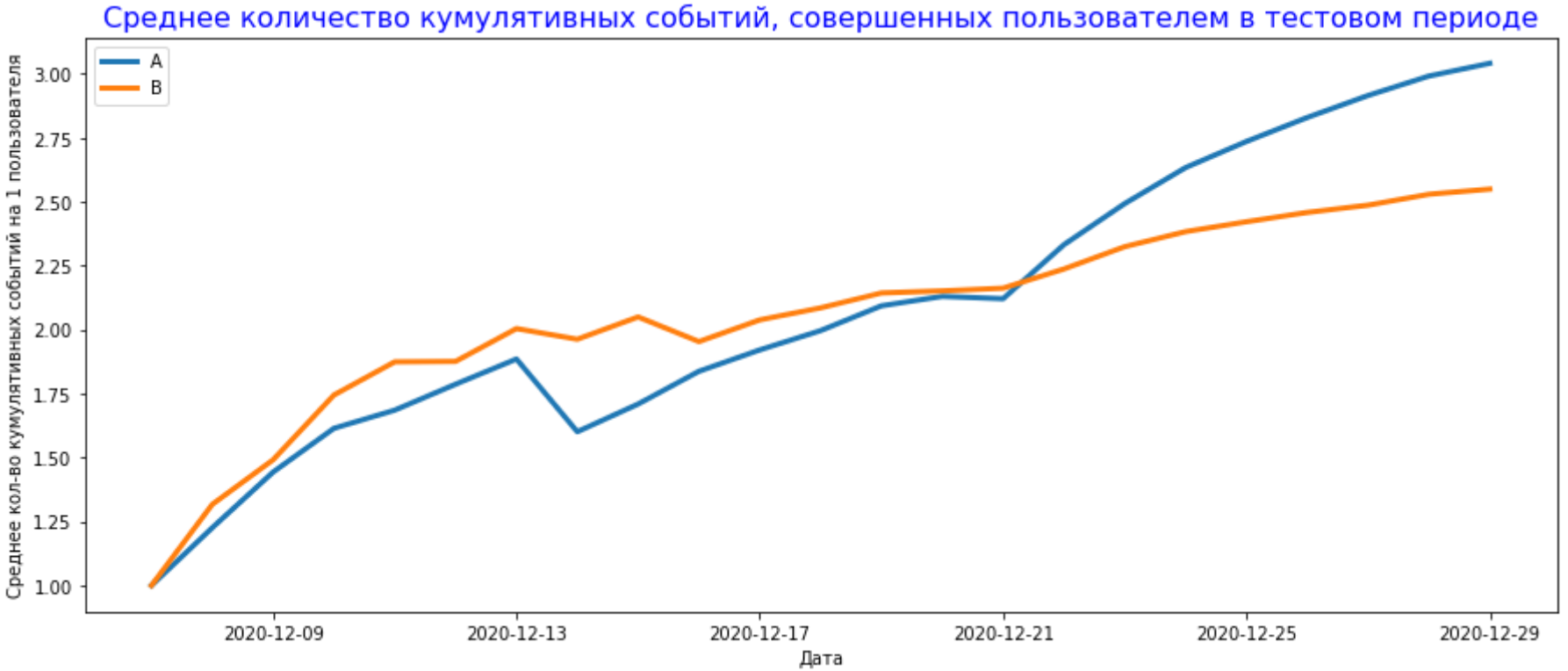


График относительного различия для среднего количества событий на одного пользователя в тестовом периоде
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
Относительное различие
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

In [98]:
```
# Изучим кумулятивные метрики по кол-ву событий на этапе воронки 'product_page'
cumulative_metrics(total_df.loc[total_df['event_name'] == 'product_page'])
```




Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде
A
B
Среднее кол-во кумулятивных событий на 1 пользователя
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

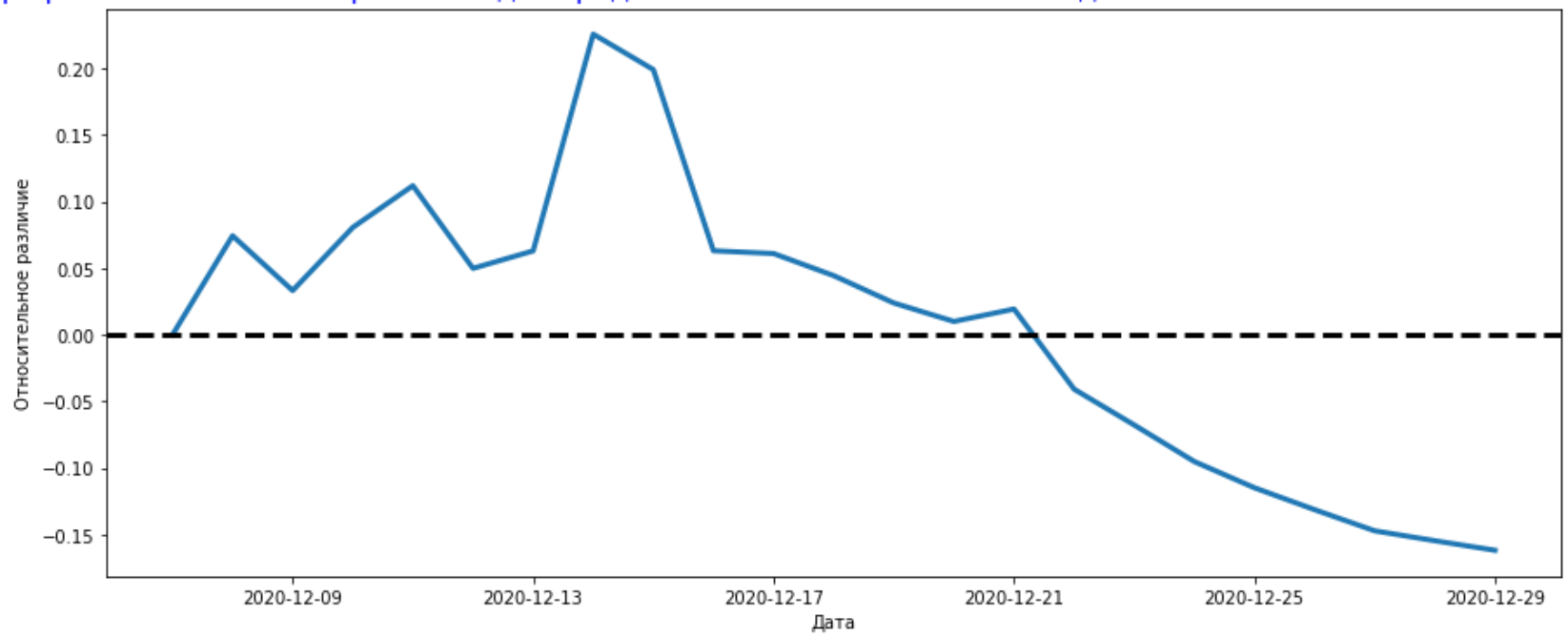
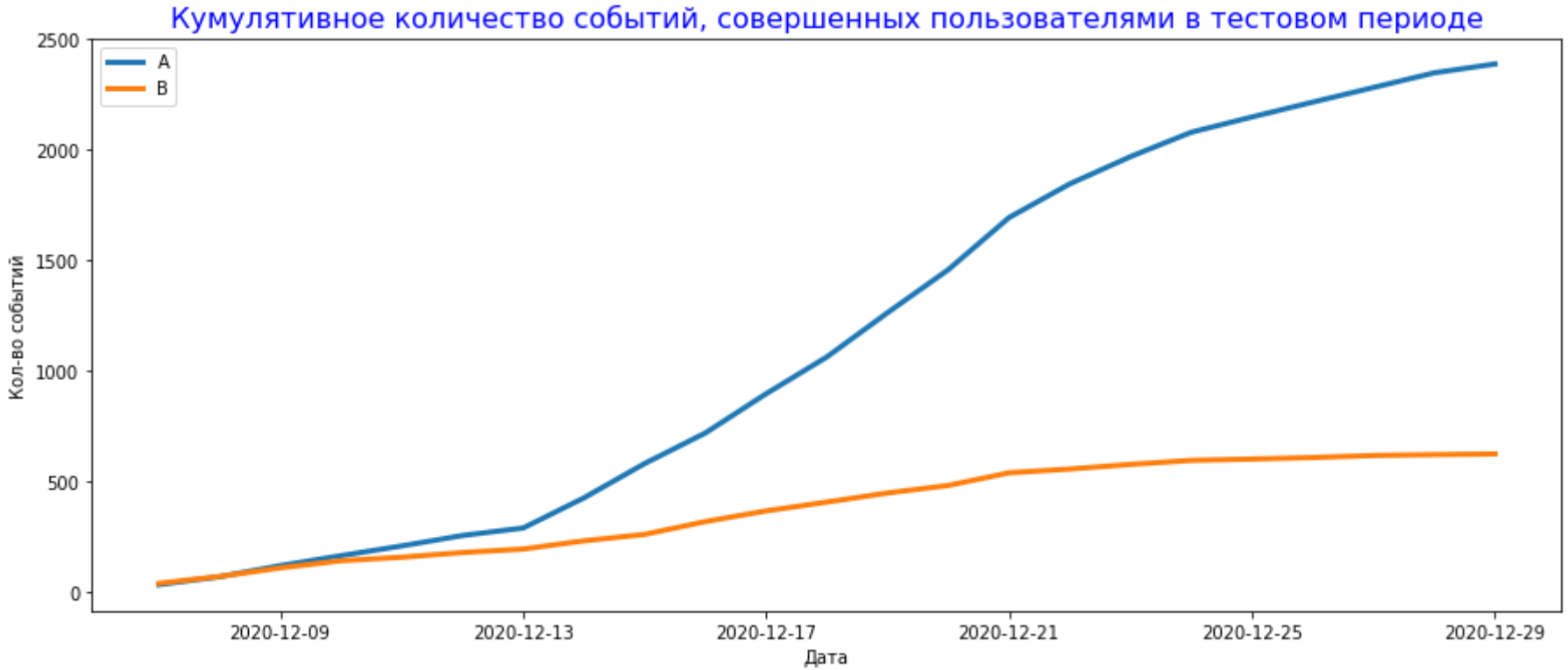
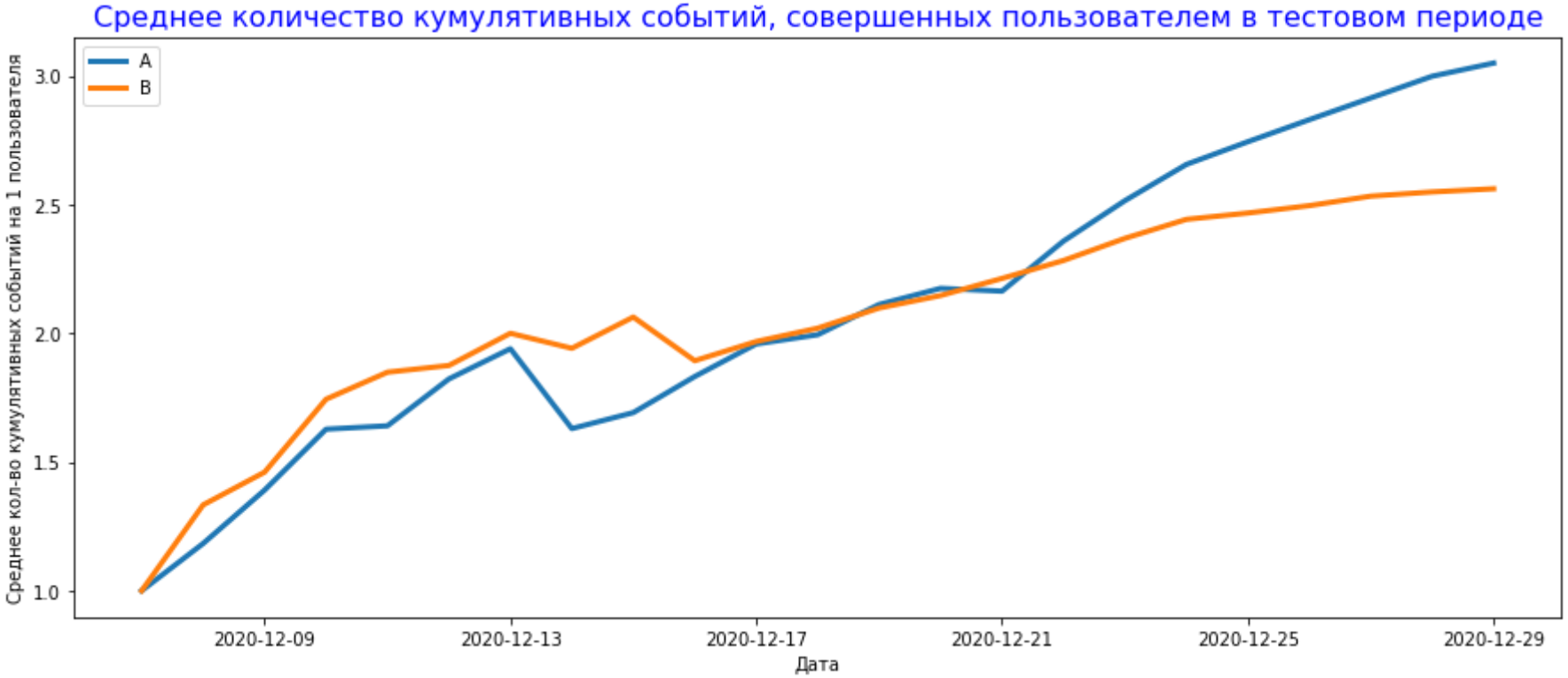


График относительного различия для среднего количества событий на одного пользователя в тестовом периоде
Относительное различие
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

In [99]:
```
# Изучим кумулятивные метрики по кол-ву событий на этапе воронки 'product_cart'
cumulative_metrics(total_df.loc[total_df['event_name'] == 'product_cart'])
```




Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде
A
B
Среднее кол-во кумулятивных событий на 1 пользователя
3.0
2.5
2.0
1.5
1.0
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

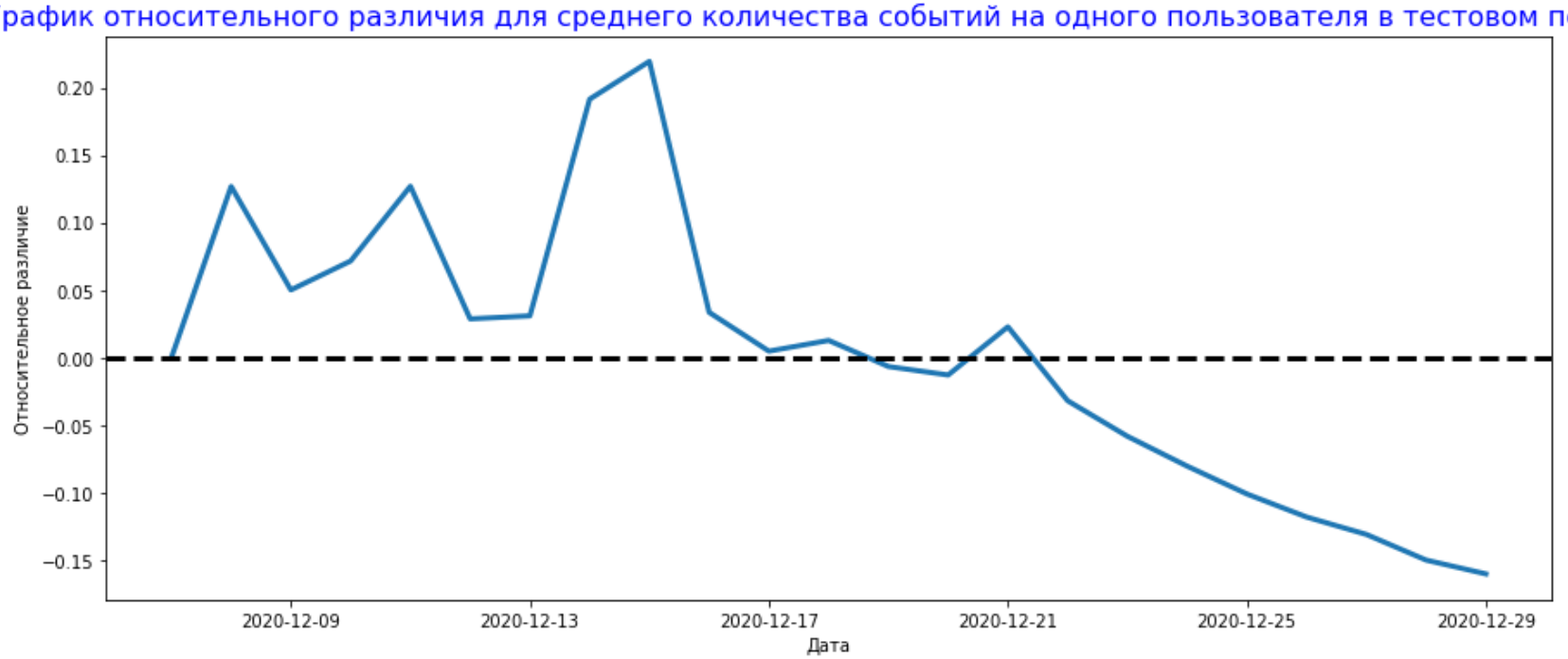
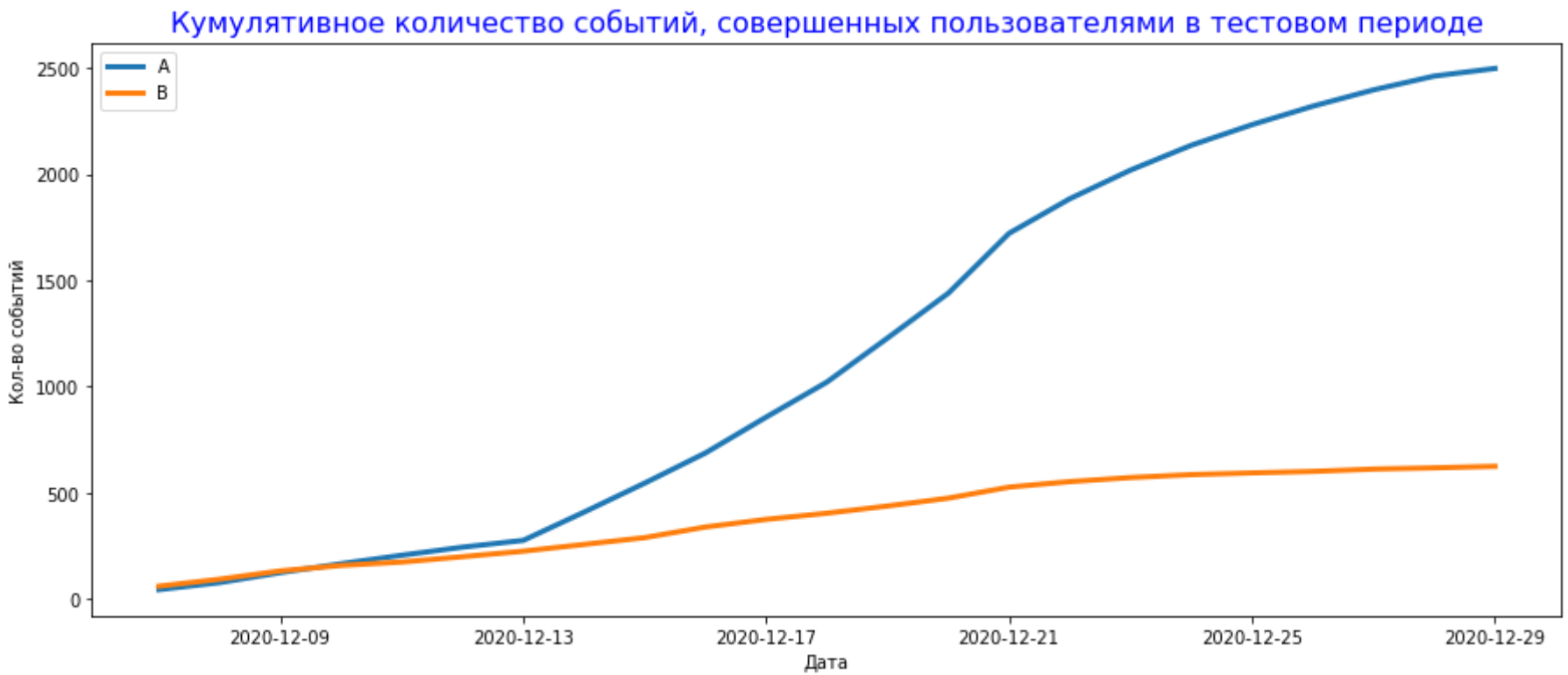
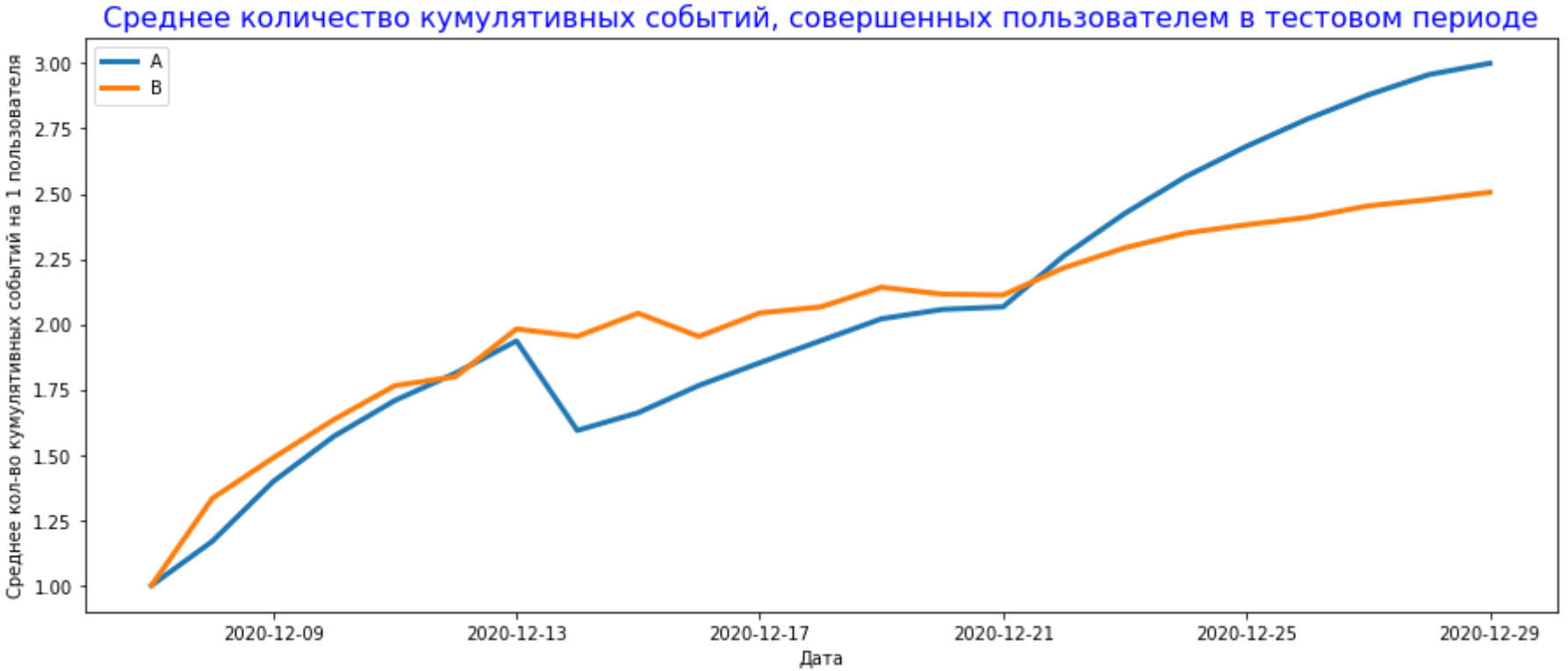
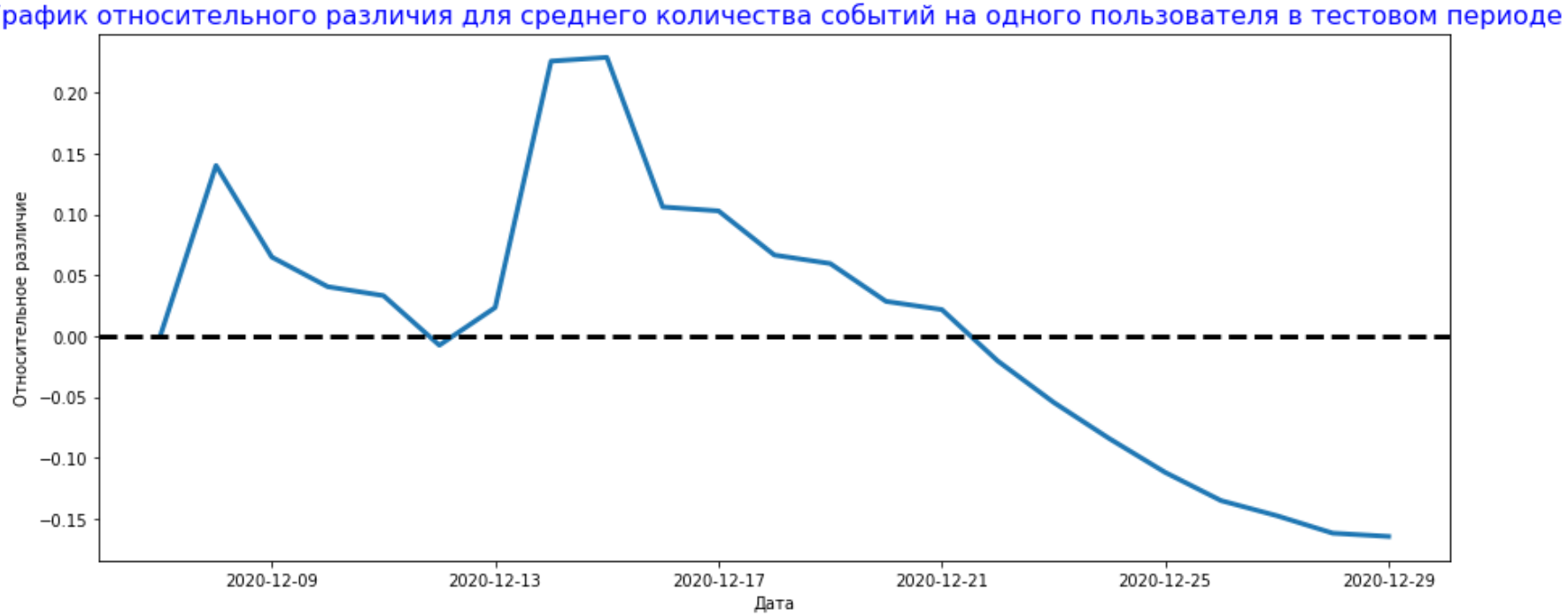


График относительного различия для среднего количества событий на одного пользователя в тестовом периоде
Относительное различие
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата

In [100]:
```
# Изучим кумулятивные метрики по кол-ву событий на этапе воронки 'purchase'
cumulative_metrics(total_df.loc[total_df['event_name'] == 'purchase'])
```




Среднее количество кумулятивных событий, совершенных пользователем в тестовом периоде
A
B
Среднее кол-во кумулятивных событий на 1 пользователя
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
2020-12-09
2020-12-13
2020-12-17
2020-12-21
2020-12-25
2020-12-29
Дата



Вывод:

- нет ни одного события в воронке, по которому группа B показывает результат лучше, чем группа A;
- с 21.12.2020 кумулятивные метрики группы B перестают показывать прирост, в то время как кумулятивные метрики группы A продолжают увеличиваться.

Ожидаемый эффект от внедрения новой рекомендательной системы не был достигнут.

5.2 Проверка статистической разницы долей метрик в группе A и группе B на значимость с помощью z-критерия

Сформулируем гипотезу:

H0:

доля уникальных посетителей из контрольной группы A, побывавших на анализируемом этапе воронки событий, равна доле уникальных посетителей из экспериментальной группы B, побывавших на этом же этапе воронки;

H1: доля уникальных посетителей из контрольной группы A, побывавших на анализируемом этапе воронки событий, не равна доле уникальных посетителей из экспериментальной группы B, побывавших на этом же этапе воронки.

Проверять гипотезу будем на трех этапах воронки:

- product_page;
- product_cart;
- purchase

Этап воронки login проверять на ниличие статистически значимой разницы в долях не имеет смысла, так как на этом этапе побывало 100% пользователей группы.

Проверку равенства долей уникальных пользователей группы A и группы B, побывавших на анализируемом этапе воронки

событий, проверять будем с помощью Z-теста.

Установим критический уровень статистической значимости (alpha) равный 5%.
Так как тест множественный (3 теста) и проводиться он будет на одном наборе данных, используем поправку Бонферрони для компенсации роста вероятности совершения ошибки 1-го рода, вызванного множественным тестом: alpha/3

In [101]:
```
# Объявим функцию z_test для проведения проверки гипотезы о равенстве долей
def z_test(successes, trials, alpha):

    # 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

    # посчитаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[0] + 1/trials[1]))
    # зададим стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2
    print('p-значение: ', p_value)
    print('Критический уровень статистической значимости равен:', round(alpha,4))

    if p_value < alpha:
        print('Отвергаем нулевую гипотезу о равенстве долей')
    else:
        print('Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными')
```

In [102]:
```
# Объявим функцию start_z_test для запуска Z-теста для A/B теста (2 группы A и B)
def start_z_test(df0, df, alpha):
    alpha = alpha/len(df.index)
    for i in df.index:
        successes = np.array([df['A'][i], df['B'][i]])
        trials = np.array([df0['A'], df0['B']])
        print(f'Результаты z-теста для этапа воронки событий - {i}:')
        z_test(successes, trials, alpha)
        print('-----------------------------------------------------------------------')
        print()
```

In [103]:
```
# Создадим сводную таблицу общее кол-во уникальных пользователей в разрезе групп A/B теста
total_users_nunique = total_df.pivot_table(columns='group', values='user_id', aggfunc='nunique')
total_users_nunique.columns = ['A', 'B']
total_users_nunique = total_users_nunique.rename({'user_id':'count_users'})
total_users_nunique
```

Out[103]:

	A	B
count_users	2604	877

In [104]:
```
# Создадим сводную таблицу кол-во уникальных пользователей в разрезе групп A/B теста и видов событий
event_users_nunique = total_df.pivot_table(index = 'event_name', columns='group', values='user_id', aggfunc='nunique')
event_users_nunique.columns = ['A', 'B']
event_users_nunique.index = ['login', 'product_cart', 'product_page', 'purchase']
event_users_nunique = event_users_nunique.drop (index = 'login')
event_users_nunique = event_users_nunique.reindex(['product_page','product_cart','purchase'])
event_users_nunique
```

Out[104]:

	A	B
product_page	1685	493
product_cart	782	244
purchase	833	249

In [108]:
```
s = total_df.pivot_table(index = 'event_name', columns='group', values='user_id', aggfunc='nunique')
s.columns = ['A', 'B']
s = s.reset_index()
s
```

Out[108]:

	event_name	A	B
0	login	2604	876
1	product_cart	782	244
2	product_page	1685	493
3	purchase	833	249

In [110]:
```
from statsmodels. stats.proportion import proportions_ztest
for i in range(1,4):
    success = [s.loc[i, 'A'], s.loc[i, 'B']]
    trials = [s.loc[0, 'A'], s.loc[0, 'B']]
    stat, pval = proportions_ztest(success, trials)
    print(pval)
```

```
0.2215941567364419
8.195976000351998e-06
0.0486476669504243
```

In [105]:
```
# Зададим критичесий уровень значимости
alpha = 0.05
```

In [106]:
```
# Запустим z тест с датасетами total_users_nunique и event_users_nunique
start_z_test(total_users_nunique, event_users_nunique, alpha)
```

```
Результаты z-теста для этапа воронки событий - product_page:
p-значение:  [6.94273936e-06]
Критический уровень статистической значимости равен: 0.0167
Отвергаем нулевую гипотезу о равенстве долей
----------------------------------------------------------------------

Результаты z-теста для этапа воронки событий - product_cart:
p-значение:  [0.21469192]
Критический уровень статистической значимости равен: 0.0167
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
----------------------------------------------------------------------

Результаты z-теста для этапа воронки событий - purchase:
p-значение:  [0.04652483]
Критический уровень статистической значимости равен: 0.0167
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
----------------------------------------------------------------------
```

✓ **Комментарий ревьюера:** Ты написала правильный код

Выводы:

- **по событию product_page** нулевая гипотеза была отвергнута z-тестом, следовательно, **возможно существует статистически значимая разница** между долей уникальных пользователей группы A, побывавших на этапе воронки product_page, и долей уникальных пользователей группы B, побывавших на этапе воронки product_page;
- **по событию product_cart** не получилось отвергнуть нулевую гипотезу, следовательно, **нет статистически значимой разницы** между долями уникальных пользователей группы A и группы B, побывавших на этапе product_cart воронки;
- **по событию purchase** не получилось отвергнуть нулевую гипотезу, следовательно, **нет статистически значимой разницы** между долями уникальных пользователей группы A и группы B, побывавших на этапе purchase воронки.

В техническом задании предполагалось, что эффект от внедрения новой улучшенной рекомендательной системы по каждой метрике будет не менее 10%. Однако, A/B тест не подтвердил наличия данного эффекта.

6 Общее заключение

В начало

Предварительный анализ данных:

Для проведения анализа результатов A/B теста было предоставлено 4 файла с данными:

- ab_project_marketing_events.csv - календарь маркетинговых событий на 2020 год.
- final_ab_new_users.csv - пользователи, зарегистрировавшиеся с 7 до 21 декабря 2020 года.
- final_ab_events.csv - действия новых пользователей в период с 07.12.2020 по 04.01.2021.
- final_ab_participants.csv - таблица участников тестов.

В ходе предварительно анализа установлено:

- дубликатов в данных нет;
- пропуски данных в столбце details файла final_ab_events не требуют обработки;
- всего в 2020 г. было 14 маркетинговых собятий;
- исходные данные содержат информацию по 4-м регионах: EU, CIS, APAC, N.America;
- регистрация новых пользователей в рамках A/B теста проводилась с 7 до 21 декабря 2020 года
- наибольшее количество зарегистрированных приходится на регион EU;
- регистрацию пользователи производили с 4-х устройств: Android, Mac, PC, iPhone;
- наибольшее количество зарегистрированных приходится на устройство Android;
- действия новых пользователей в период с 7 по 30 декабря 2020 года
- данные содержат 4 типа событий: purchase, product_cart, product_page, login;
- в данных информация о двух тестах interface_eu_test и recommender_system_test;
- в рамках каждого теста клиенты разделены на 2 группы A и B.

В данных выявлена информация, не имеющая отношения к техническому заданию, и информация, не соответствующая техническому заданию. Поэтому возникла необходимость до начала исследовательского анализа и анализа результатов A/B теста провести:

- детальную проверку соответствия данных ТЗ;
- анализ время проведения теста;
- анализ аудитории, участвующей в тесте

и создать новый датасет для проведения исследовательского анализа и анализа результатов A/B теста.

Оценка корректности проведения A/B теста:

В данных выявлены следующие несоответсвия ТЗ:

- последняя дата регистрации пользователей, попавшая в датасет new_users (23.12.2020 г.), не соответствует дате остановки набора новых пользователей - должно быть 21.12.2020 г.(устранено);
- дата остановки теста 2021-01-04, указанная в ТЗ, не соответствует последней дате совершения пользователями события в датасете events 2020-12-30(не удалось устранить);
- выявлено наличие данных по 4 регионам, в ТЗ указан только один - EU (устранено);
- выявлено наличие данных по двум тестам(устранено).

Анализ времени проведения теста показал, что:

- в период проведения теста в EU проводилась маркетинговая активность Christmas&New Year Promo, однако ее влияние на поведение пользователей не было подтверждено графиком.

Анализ аудитории выявил:

- 1602 пользователей, участвующих одновременно в двух тестах, анализ распределения этих пользователей по группам тестов позволил считать их пригодными для участии в тесте recommender_system_test;
- пользователей, участвующих в двух группах теста recommender_system_test одновременно, нет.
- неравномерное распределение пользователей по тестовым группам: в группе A - 3385 чел (57%), в группе B - 2533 чел(43%);
- только 55% пользователей в выборке recommender_system_test совершали какие-либо действия в тестовом периоде;

- в группе B неактивных пользователей почти в два раза болше(29%), чем в группе A(16%).

Данный этап анализа показал, что результаты A/B теста могут быть некорректными по причине:

- отсутствия данных о событиях, совершенных пользователями в период с 31.12.2020 по 04.01.2021;
- совпадения времени проведения теста с рождественскими праздниками;
- неравномерного распределения пользователей по группас A/B теста.

Исследовательский анализ данных:

1. Анализ распределения количества событий на пользователя в выборках

- медианное количество событий в группе A на 1 пользователя - 6, в группе B - 4;
- медианное количество события purchase в группе A на 1 пользователя - 3, в группе B - 2.
- количество событий в группе B на одного пользователя примерно на 33% ниже, чем в группе A.

2. Анализ распределения по дням числа событий в выборках

- пик кол-ва событий совпадает в группах A и B - 21.12.2020 (последний день набора новых пользователей для теста);
- только в первый день теста удельный вес кол-ва событий, совершенных пользователями из группы B, больше 50%;
- с течением времени удельный вес событий, совершенных пользователяи из группы A растет, а группы B - снижается. Уже с 14.12.2020 г. удельный вес событий, совершенных пользователми из группы, превышает 80% от общего кол-ва событий за день (причина - неравномерный приток новых клиентов: в группе A есть рост кол-ва зарегистрированных пользователей, в группе B - роста нет).

3. Анализ конверсии в выборках на разных этапах воронки

Воронка событий:

1. login;
2. product_page;
3. product_cart;
4. purchase

- наибольшее кол-во клиентов "отваливается" на этапах login - product_page и product_page - product_cart;
- в группе A на этап product_page прошли 65% клиентов, в группе B - 56%;
- в группе A на этап product_cart прошли 30% клиентов, в группе B - 28%;
- на этапе product_cart-purchase график не показывает потерь клиентов, даже наоборот в группе a % в purchase выше, чем в product_cart, следовательно, есть способ совершить покупку минуя этап product_cart.

На этапе login - product_page группа B показывает результат хуже группы A на 9%. Это негативный показатель для тестируемого изменения рекомендательной системы. Возможно, именно на этом этапе находится слабое место новой рекомендательной системы.

На всех этапах исследовательского анализа данных обнаружено, что группа A показывает лучшие результаты, чем группа B.

Следовательно, эффект от внедрения новой рекомендательной системы не только не достиг желаемых 10%, но и оказался отрицательным.

Оценка результатов A/B теста:

- **по событию product_page** нулевая гипотеза **была отвергнута**, следовательно, доля уникальных пользователей группы A, побывавших на этапе воронки product_page, не равна доле уникальных пользователей группы B, побывавших на этапе воронки product_page;
- **по событию product_cart** не получилось отвергнуть нулевую гипотезу, следовательно, **нет статистически значимой разницы** между долями уникальных пользователей группы A и группы B, побывавших на этапе product_cart воронки;
- **по событию purchase** не получилось отвергнуть нулевую гипотезу, следовательно, **нет статистически значимой разницы** между долями уникальных пользователей группы A и группы B, побывавших на этапе purchase воронки.

Таким образом, как и исследовательский анализ, анализ результатов A/B тест не подтвердил наличия ожидаемого в ТЗ эффекта от внедрения новой рекомендательной системы.

Заключение:

Считаю, что А/Б тест проведен некорректно:

- неудачно выбран тестовый период - рождественские праздники оказывают очень большое влияние на поведение пользователей;
- предоставлены неполные данные о событиях, совершенных пользователями - нет данных за период с 31.12.2020 по 04.01.2020;
- некорректно распределены пользователи по группам, группа А (2604 чел.) почти в 3 раза больше группы В(877 чел.).

Из-за некорретного проведения теста, его результаты (отсутствие ожидаемого эффекта от внедрения новой рекомендательной системы) нельзя считать достоверными.

Рекомендую провести повторный тест, устранив вышеуказанные замечания.