Анализ корректности A/B теста

Наша задача — провести оценку результатов A/B теста по изменению системы рекомендаций товаров в интернет-магазине, а именно:

1. Оценить корректность проведения теста (пересечение тестовой аудитории с конкурирующим тестом, совпадение теста и маркетинговых событий, другие проблемы),
2. Проанализировать результаты теста.

Источники информации: техническое задание по проведению A/B теста; датасет с действиями новых пользователей в период с 7 декабря 2020 по 4 января 2021 года; календарь маркетинговых событий на 2020 год; реестр пользователей, зарегистрировавшихся с 7 по 21 декабря 2020 года; реестр участников тестов.

Техническое задание по A/B тесту:

- название теста: `recommender_system_test`;
- группы: A — контрольная, B — экспериментальная;
- дата запуска теста: 2020-12-07;
- дата остановки набора новых пользователей: 2020-12-21;
- дата остановки теста: 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`,
  - конверсии в просмотры корзины - `product_cart`,
  - конверсии в покупки - `purchase`.

В рамках поставленных задач будут выполнены следующие шаги:

- обзор данных,
- предобработка данных,
- оценка корректности проведения теста,
- исследовательский анализ данных,
- оценка результатов A/B-тестирования,
- выводы.

In [1]:
```
# импортируем библиотеки
import pandas as pd
import numpy as np
import datetime as dt
import scipy.stats as st
from statsmodels.stats.proportion import proportions_ztest

# импортируем библиотеки для построения графиков
import plotly.express as px
from plotly import graph_objects as go
from plotly.subplots import make_subplots

# настраиваем отображение всех колонок датафрейма при выводе данных
pd.options.display.max_columns = None

# отключаем предупреждения
import warnings
warnings.filterwarnings('ignore')

# зададим общую палитру для графиков
colours = ['#74AB5C','#B3D7A3','#C7C0DF','#9C8CD4','#5A5387','#FFE47B','#F4CCCC','#E73333','#796BC9']
```

Загрузка и обзор данных

In [2]:
```
# загружаем данные

try:
    events = pd.read_csv('/datasets/final_ab_events.csv')
    new_users = pd.read_csv('/datasets/final_ab_new_users.csv')
    ab_participants = pd.read_csv('/datasets/final_ab_participants.csv')
    marketing = pd.read_csv('/datasets/ab_project_marketing_events.csv')
except:
    events = pd.read_csv(
        '/Users/mrmrzpn/Desktop/Yandex Praktikum/2. Проекты/10. Финальный проект/2. A:B тест/final_ab_events.csv')
    new_users = pd.read_csv(
        '/Users/mrmrzpn/Desktop/Yandex Praktikum/2. Проекты/10. Финальный проект/2. A:B тест/final_ab_new_users.csv')
    ab_participants = pd.read_csv(
        '/Users/mrmrzpn/Desktop/Yandex Praktikum/2. Проекты/10. Финальный проект/2. A:B тест/final_ab_participants.csv')
    marketing = pd.read_csv(
        '/Users/mrmrzpn/Desktop/Yandex Praktikum/2. Проекты/10. Финальный проект/2. A:B тест/ab_project_marketing_events.csv')
```

In [3]:
```
# смотрим информацию о датафреймах

for data in [events, new_users, ab_participants, marketing]:
    name = [key for key in globals() if globals()[key] is data]
    print('Датасет', name[0], '\n')
    data.info()
    display(data.head())
    print('--'*50)
```

```
Датасет events

<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
```

	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

```
--------------------------------------------------------------------------------------------------------------
Датасет new_users

<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
```

	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

```
--------------------------------------------------------------------------------------------------------------
Датасет ab_participants

<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
```

	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

```
--------------------------------------------------------------------------------------------------------------
Датасет marketing

<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
```

	name	regions	start_dt	finish_dt
0	Christmas&New Year Promo	EU, N.America	2020-12-25	2021-01-03
1	St. Valentine's Day Giveaway	EU, CIS, APAC, N.America	2020-02-14	2020-02-16
2	St. Patric's Day Promo	EU, N.America	2020-03-17	2020-03-19
3	Easter Promo	EU, CIS, APAC, N.America	2020-04-12	2020-04-19
4	4th of July Promo	N.America	2020-07-04	2020-07-11

Описание данных:

`events` — действия новых пользователей в период с 7 декабря 2020 по 4 января 2021 года:

- `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.

`new_users` — пользователи, зарегистрировавшиеся с 7 по 21 декабря 2020 года:

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

`ab_participants` — таблица участников тестов:

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

`marketing` — календарь маркетинговых событий на 2020 год:

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

Наблюдения:

- Пропуски есть только в датасете `events` в столбце `details`;
- Тип данных в столбцах с датами исправим на datetime и datetime.date.

Предобработка данных

Изменение типа данных

In [4]:
```
# изменим тип данных в столбцах с датами

events['event_dt'] = pd.to_datetime(events['event_dt'])
new_users['first_date'] = pd.to_datetime(new_users['first_date']).dt.date
marketing['start_dt'] = pd.to_datetime(marketing['start_dt']).dt.date
marketing['finish_dt'] = pd.to_datetime(marketing['finish_dt']).dt.date
```

Проверка на наличие дубликатов

Полные дубликаты

In [5]:
```
# считаем число полных дубликатов
for data in [events, new_users, ab_participants, marketing]:
    name = [key for key in globals() if globals()[key] is data]
    print('Число полных дубликатов в датасете {} -'.format(name[0]),
          data.duplicated().sum()
         )
```

```
Число полных дубликатов в датасете events - 0
Число полных дубликатов в датасете new_users - 0
Число полных дубликатов в датасете ab_participants - 0
Число полных дубликатов в датасете marketing - 0
```

Наблюдения: Явные дубликаты не найдены.

Неявные дубликаты

Посмотрим на наличие неявных дубликатов в названиях событий, регионов, девайсов, тестов и маркетинговых кампаний:

In [6]:
```
list(events['event_name'].unique())
```

Out[6]:
```
['purchase', 'product_cart', 'product_page', 'login']
```

In [7]:
```
list(new_users['region'].unique())
```

Out[7]:
```
['EU', 'N.America', 'APAC', 'CIS']
```

In [8]:
```
list(new_users['device'].unique())
```

Out[8]:
```
['PC', 'Android', 'iPhone', 'Mac']
```

In [9]:
```
list(ab_participants['ab_test'].unique())
```

Out[9]:
```
['recommender_system_test', 'interface_eu_test']
```

In [10]:
```
sorted(marketing['regions'].unique())
```

Out[10]:
```
['APAC',
 'CIS',
 'EU, CIS, APAC',
 'EU, CIS, APAC, N.America',
 'EU, N.America',
 'N.America']
```

In [11]:
```
sorted(marketing['name'].unique())
```

Out[11]:
```
['4th of July Promo',
 'Black Friday Ads Campaign',
 'CIS New Year Gift Lottery',
 'Chinese Moon Festival',
 'Chinese New Year Promo',
 'Christmas&New Year Promo',
 'Dragon Boat Festival Giveaway',
 'Easter Promo',
 "International Women's Day Promo",
 'Labor day (May 1st) Ads Campaign',
 "Single's Day Gift Promo",
 "St. Patric's Day Promo",
 "St. Valentine's Day Giveaway",
 'Victory Day CIS (May 9th) Event']
```

Наблюдения: Неявные дубликаты отсутствуют. В логе присутствуют только 4 вида событий. Реестр участников представлен по 2 тестам.

Анализ пропусков

При обзоре данных мы обнаружили пропуски только в столбце `details` датасета `events`. В данном столбце представлена дополнительная информация о событиях, и пропуски обусловлены тем, что не для каждого события дополнительные сведения могут быть применимы и нужны.

Мы знаем, что для покупок ("purchase") в этом поле хранится стоимость покупки в долларах. Посмотрим, для каких событий указана дополнительная информация и сколько таких событий:

In [12]:
```
(events.groupby('event_name')
 .agg({'details':'count','event_dt':'count'})
 .rename(columns={'details':'details_number','event_dt':'events_number'})
)
```

Out[12]:

event_name	details_number	events_number
login	0	189552
product_cart	0	62462
product_page	0	125563
purchase	62740	62740

Наблюдения: Дополнительные сведения есть только о покупках, и представлены они по всем покупкам.

Общий вывод: На данном этапе мы не увидели критических проблем с качеством данных: логи собираются автоматически, что исключает влияние человеческого фактора; пропуски есть только в столбце с дополнительными сведениями, которые не являются необходимыми для нашего анализа; дубликаты отсутствуют.

Оценка корректности проведения теста

Анализ соответствия требованиям технического задания

Период тестирования

In [13]:
```
# соберем в один датафрейм данные по исследуемому тесту - recommender_system_test

rs_test = (ab_participants.query('ab_test == "recommender_system_test"')
           .merge(new_users, on='user_id', how='left')
           .merge(events, on='user_id', how='left')
          )
```

In [14]:
```
# посмотрим на фактический период тестирования

print('Фактическое начало тестирования - {}, конец - {}'
      .format(rs_test['event_dt'].min(), rs_test['event_dt'].max())
     )
```

```
Фактическое начало тестирования - 2020-12-07 00:05:57, конец - 2020-12-30 12:42:57
```

Наблюдения: Фактическое начало тестирования соответствует техническому заданию, однако период тестирования закончился раньше, чем того требовало ТЗ: 30 декабря 2020 г. вместо 4 января 2021 г.

Период теста в ТЗ устанавливался на основании того, что каждый пользователь должен "прожить" 14 дней с момента регистрации, и уменьшение периода тестирования приводит к тому, что в логах могут быть пользователи, которые не "дожили" до установленного лайфтайма. Если такие пользователи не успели пройти по всем этапам воронки за свой ограниченный лайфтайм, то они, возможно, будут снижать показатели конверсии: мы не можем быть уверены, что эти пользователи всё равно бы не прошли воронку, даже если бы "прожили" все 14 дней.

Рассчитаем лайфтайм событий по пользователям и посмотрим, сколько из участников теста не "дожили" до 14 дней, а также сколько этапов воронки они успели пройти.

Лайфтайм считается с момента регистрации, поэтому первым шагом воронки будем считать регистрацию, и добавим в датафрейм регистрацию как отдельное событие.

In [15]:
```
# добавим в датасет дополнительные строки с пустыми ячейками в названии и дате события,
# которые потом заполним событием "регистрация" и соответсвующей датой
rs_test = pd.concat([rs_test, (ab_participants
                               .query('ab_test == "recommender_system_test"')
                               .merge(new_users, on='user_id', how='left')
                              )
                    ], axis=0)

# заполним пропуски: название события - 'registration', время события - время регистрации
rs_test.loc[rs_test['event_name'].isna(), 'event_name'] = 'registration'
rs_test.loc[rs_test['event_name']=='registration', 'event_dt'] = (rs_test
                                                                  .loc[rs_test['event_name']=='registration',
                                                                       'first_date'])
rs_test['event_dt'] = pd.to_datetime(rs_test['event_dt'])

# убираем дубликаты
rs_test = rs_test.drop_duplicates()
```

In [16]:
```
# добавим столбец с датой события
rs_test['event_date'] = rs_test['event_dt'].dt.date

# отсортируем данные в датасете по дате регистрации пользователя, id пользователя и времени события
rs_test = rs_test.sort_values(by=['first_date', 'user_id', 'event_dt'])

# добавим столбец с лайфтаймом
rs_test['lifetime'] = (rs_test['event_date'] - rs_test['first_date']).dt.days
```

In [17]:
```
# посмотрим, сколько пользователей "прожили" менее 14 дней - дата регистрации с 18 декабря

max_registration_date = dt.date(2020, 12, 17)

print('Число пользователей, "проживших" менее 14 дней до остановки теста, -',
      rs_test.query('first_date > @max_registration_date')['user_id'].nunique()
     )
```

```
Число пользователей, "проживших" менее 14 дней до остановки теста, - 2028
```

In [18]:
```
# посмотрим, сколько пользователей, "проживших" менее 14 дней, не прошли все этапы воронки

print('Число пользователей, "проживших" менее 14 дней и не прошедших все этапы воронки, -',
      (rs_test.query('first_date > @max_registration_date')
       .groupby('user_id').agg({'event_name':'nunique'})
       .query('event_name < 5')['event_name'].count())
     )
```

```
Число пользователей, "проживших" менее 14 дней и не прошедших все этапы воронки, - 1939
```

In [19]:
```
# посмотрим, сколько пользователей, "проживших" менее 14 дней, только зарегистрировались
# и далее не были активны

print('Число пользователей, "проживших" менее 14 дней и прошедших только 1 этап воронки (регистрацию), -',
      (rs_test.query('first_date > @max_registration_date')
       .groupby('user_id').agg({'event_name':'nunique'})
       .query('event_name == 1')['event_name'].count())
     )
```

```
Число пользователей, "проживших" менее 14 дней и прошедших только 1 этап воронки (регистрацию), - 625
```

Наблюдения: Количество не "доживших" до 14 дней пользователей существенное (2 тыс. человек), их исключение снизит необходимый размер выборки для тестирования. Почти все эти пользователи (не считая 89 человек) не прошли все этапы воронки, при этом только чуть меньше 1/3 не были активны после регистрации. Поскольку основная часть данных пользователей была активна, предполагаем, что они не исказят уровень конверсии.

Поскольку период наблюдения за каждым пользователем - 14 дней, исключим те события, которые выходят за пределы 14 дней лайфтайма:

In [20]:
```
rs_test = rs_test.query('lifetime <= 14').reset_index(drop=True)
```

Аудитория теста

Аудитория теста должна соответствовать следующим критериям:

- новые пользователи, зарегистрировавшиеся до 21 декабря 2020 г. включительно,
- регион - EU,
- размер выборки - 15% новых пользователей из региона EU, зарегистрировавшихся в период с 7 по 21 декабря 2020 г.,
- размер выборки - не менее 6000 человек,
- группы пользователей - A и B, где A — контрольная, B — новая платёжная воронка.

1. Дата регистрации пользователей

In [21]:
```
# проверим даты регистрации пользователей

print('Начало набора новых пользователей – {}, конец – {}'
      .format(rs_test['first_date'].min(), rs_test['first_date'].max())
     )
```

```
Начало набора новых пользователей – 2020-12-07, конец – 2020-12-21
```

Наблюдения: Окончание набора новых пользователей соответствует ТЗ.

2. Регион пользователей

In [22]:
```
# проверим регион пользователей в тесте

print('Регионы участников теста:', list(rs_test['region'].unique()))
```

```
Регионы участников теста: ['EU', 'N.America', 'APAC', 'CIS']
```

Наблюдения: В тест попали пользователи из регионов помимо EU.

Исключим лишних пользователей из датасета для корректного анализа результатов теста и проверки соответствия объема выборки тех.заданию:

In [23]:
```
# исключим пользователей не из EU

rs_test = rs_test.query('region == "EU"').reset_index(drop=True)
```

3. Размер аудитории теста

In [24]:
```
# проверим, что число пользователей в тесте более 6000

print('Фактический размер аудитории теста – {}'.format(rs_test['user_id'].nunique()))
```

```
Фактический размер аудитории теста – 6351
```

In [25]:
```
# проверим, что число пользователей в тесте соответствует 15% новых пользователей из EU

registration_start = dt.date(2020, 12, 7)
registration_end = dt.date(2020, 12, 21)

print('Доля аудитории теста в количестве новых пользователей из EU – {:.0%}'
      .format(rs_test['user_id'].nunique() /
              (new_users.query('region == "EU" and @registration_start <= first_date <= @registration_end')
               ['user_id'].nunique())
             ))
```

```
Доля аудитории теста в количестве новых пользователей из EU – 15%
```

Наблюдения: Размер аудитории теста соответствует ТЗ.

4. Группы теста

In [26]:
```
# проверим, что пользователи разделены только на 2 группы – A и B

print('Группы теста –', list(rs_test['group'].unique()))
```

```
Группы теста – ['A', 'B']
```

Наблюдения: Разделение на группы соответствует ТЗ.

Анализ наличия параллельных маркетинговых активностей

In [27]:
```
# посмотрим, какие маркетинговые кампании проводились параллельно с тестом:
# в регионе EU, с датой завершения 7 декабря 2020 г. и позже

marketing.query('regions.str.contains("EU") and finish_dt >= @rs_test.event_date.min()')
```

Out[27]:

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

Наблюдения: Одновременно с тестом проводилась только одна маркетинговая кампания, посвященная Рождеству и Новому году. Кампания была запущена за 5 дней до окончания теста и могла повлиять на активность участников теста и, возможно, конверсию в этапы воронки. Однако весь декабрь - это время подготовки к праздникам, активной закупки подарков и, соответственно, роста спроса. Влияние сезонности и данной маркетинговой акции на обе группы должно быть одинаковым и не исказит результаты теста.

Анализ аудитории теста

Пересечение с конкурирующим тестом

Пересечение одного теста с другим создает вероятность, что на активность пользователей и конверсию может повлиять не проверяемое в анализируемом тесте изменение, а изменение, тестируемое параллельно.

In [28]:
```
# проверим, участвуют ли в другом тесте пользователи анализируемого теста, и сколько таких пользователей

users_2_tests = (ab_participants
                 .query('ab_test != "recommender_system_test" and user_id.isin(@rs_test.user_id.unique())')
                 ['user_id'])

print('Число пользователей, участвующих в другом тесте -', users_2_tests.count())
```

```
Число пользователей, участвующих в другом тесте - 1602
```

Если данные пользователи попали только в контрольную группу (A) конкурирующего теста, то влияние на наш тест будет нулевым, ведь в другом тесте они не видели никаких изменений. Проверим, в каких группах конкурирующего теста участвуют данные пользователи:

In [29]:
```
print('Пользователи нашего теста попали в следующие группы конкурирующего теста:')

(ab_participants
 .query('ab_test != "recommender_system_test" and user_id.isin(@users_2_tests)')
 .value_counts('group')
)
```

```
Пользователи нашего теста попали в следующие группы конкурирующего теста:
```

Out[29]:
```
group
A    819
B    783
dtype: int64
```

Наблюдения: Почти четверть участников нашего теста попали в другое тестирование и присутствовали там и в контрольной, и в экспериментальной группах. Мы не располагаем информацией о втором тесте: какие изменения проверяются, каков период тестирования, - поэтому не можем сказать, что этот тест проводился параллельно с анализируемым и мог исказить его результаты.

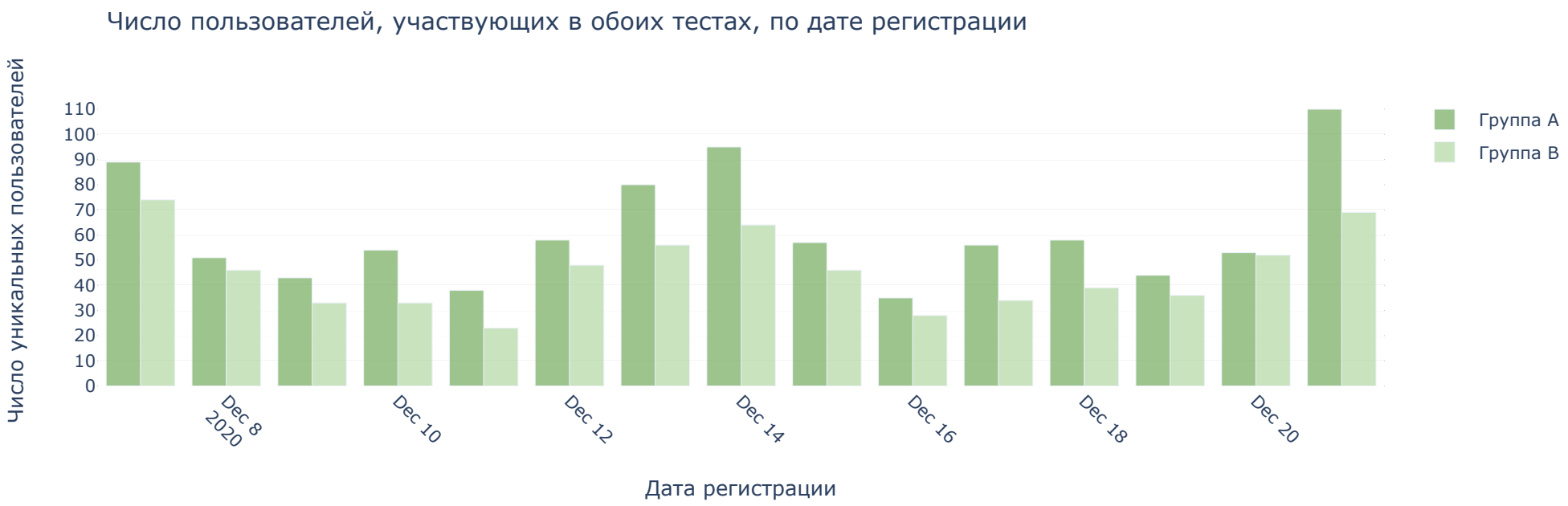
Посмотрим, как такие пользователи распределены по группам нашего теста: если их доля от размера группы сопоставима между A и B, влияние на разницу конверсии между группами не должно быть критическим.

In [30]:
```
# соберем данные по количеству таких пользоватлей в разрезе группы теста и даты регистрации
ab_users_2_tests = (rs_test.query('user_id.isin(@users_2_tests)')
                    .groupby(['group','first_date'], as_index=False).agg({'user_id':'nunique'})
                   )

# построим график распределения данных пользователей по датам регистрации и группам
fig = go.Figure()

for i, group in enumerate(["A", "B"]):
    fig.add_trace(go.Bar(x=ab_users_2_tests.query('group == @group')['first_date'],
                         y=ab_users_2_tests.query('group == @group')['user_id'],
                         marker_color=colours[i],
                         name='Группа {}'.format(group),
                         opacity=.7)
                  )
fig.update_layout(height=350, width=950, plot_bgcolor='white', font_size=10,
                      title={'text':
                             'Число пользователей, участвующих в обоих тестах, по дате регистрации',
                             'y':.85,
                             'x':.09,
                             'font.size':14}
                  )
fig.update_yaxes(title_text='Число уникальных пользователей', dtick=10)
fig.update_xaxes(title_text='Дата регистрации', tickangle=45)

fig.show()
```



In [31]:
```
# посчитаем долю данных пользователей в каждой группе теста

print('Доля пользователей, участвующих в двух тестах: \nгруппа A – {:.2%} \nгруппа B – {:.2%}'
      .format((ab_users_2_tests.query('group == "A"')['user_id'].sum() /
               rs_test.query('group == "A"')['user_id'].nunique()),
              (ab_users_2_tests.query('group == "B"')['user_id'].sum() /
               rs_test.query('group == "B"')['user_id'].nunique())
             ))
```

```
Доля пользователей, участвующих в двух тестах:
группа A – 25.34%
группа B – 25.06%
```

Наблюдения: Доля пользователей, участвующих в двух тестах, и динамика их набора (регистрации) сопоставимы в группах A и B. Таким образом, предполагаем, что участие в другом тесте не должно существенно повлиять на результаты анализируемого теста.

Пересечение между группами теста

Для чистоты эксперимента группы теста не должны пересекаться. Проверим, что в нашем тесте это требование соблюдается:

In [32]:
```
# рассчитаем, сколько пользователей попали одновременно в группу A и группу B

print('Число пользователей, попавших одновременно в группу A и группу B, –',
      rs_test.groupby('user_id').agg({'group':'nunique'}).query('group > 1').shape[0]
     )
```

```
Число пользователей, попавших одновременно в группу A и группу B, – 0
```

Наблюдения: Группы пользователей не пересекаются.

Распределение по группам теста

In [33]:
```
# посмотрим на размер групп

rs_test.groupby('group').agg({'user_id':'nunique'}).rename(columns={'user_id':'users_number'})
```

Out[33]:

group	users_number
A	3634
B	2717

Наблюдения: Размеры групп сопоставимы.

Активность пользователей

Посмотрим, сколько пользователей не были активны после регистрации:

In [34]:
```
# сохраним список неактивных пользователей в отдельной переменной
inactive_users = list(rs_test.groupby('user_id').agg({'event_name':'nunique'}).query('event_name == 1').index)

# считаем число таких пользователей
print('Количество неактивных пользователей –', len(inactive_users))
print('Из них в группе A – {}, в группе B – {}'
      .format(rs_test.query('user_id.isin(@inactive_users) and group == "A"').shape[0],
              rs_test.query('user_id.isin(@inactive_users) and group == "B"').shape[0])
     )
```

```
Количество неактивных пользователей – 2870
Из них в группе A – 1030, в группе B – 1840
```

Наблюдения: В группе B неактивных пользователей больше, а размер группы при этом меньше.

Посмотрим на динамику регистрации пользователей в разрезе групп и признака активности, чтобы понять, равномерна ли она по группам и нет ли здесь каких-нибудь инсайтов, объясняющих разницу в неактивных пользователях.

In [35]:
```
# добавим в датасет признак активности пользователя

rs_test['active_user'] = ~rs_test['user_id'].isin(inactive_users)
rs_test['active_user'] = rs_test['active_user'].replace({True:'active', False:'inactive'})
```

In [36]:
```
# сгруппируем данные для графика

registration_dynamics = (rs_test.groupby(['group','active_user','first_date'], as_index=False)
                         .agg({'user_id':'nunique'}))
```

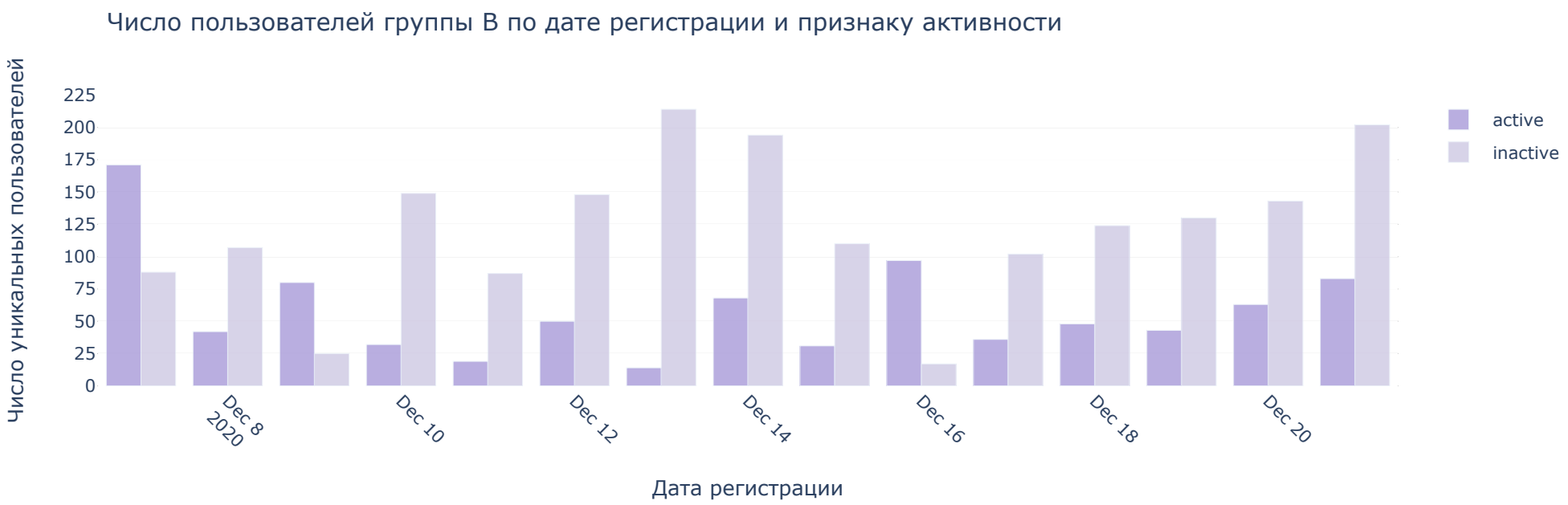
In [37]:
```
# построим график распределения пользователей по датам регистрации в разрезе групп и признака активности

# группа A
fig1 = go.Figure()
for i, activity in enumerate(['active', 'inactive']):
    fig1.add_trace(go.Bar(x=registration_dynamics.query('group == "A" and active_user == @activity')['first_date'],
                          y=registration_dynamics.query('group == "A" and active_user == @activity')['user_id'],
                          marker_color=colours[i],
                          name=activity,
                          opacity=.7)
                   )
fig1.update_layout(height=350, width=950, plot_bgcolor='white', font_size=10,
                   title={'text':
                          'Число пользователей группы A по дате регистрации и признаку активности',
                          'y':.85,
                          'x':.09,
                          'font.size':14}
                   )
fig1.update_yaxes(title_text='Число уникальных пользователей', dtick=50)
fig1.update_xaxes(title_text='Дата регистрации', tickangle=45)

fig1.show()
```

```
# группа B
fig2 = go.Figure()
for i, activity in enumerate(['active','inactive']):
    fig2.add_trace(go.Bar(x=registration_dynamics.query('group == "B" and active_user == @activity')['first_date'],
                          y=registration_dynamics.query('group == "B" and active_user == @activity')['user_id'],
                          marker_color=colours[3-i],
                          name=activity,
                          opacity=.7)
                   )
fig2.update_layout(height=350, width=950, plot_bgcolor='white', font_size=10,
                   title={'text':
                          'Число пользователей группы B по дате регистрации и признаку активности',
                          'y':.85,
                          'x':.09,
                          'font.size':14}
                   )
fig2.update_yaxes(title_text='Число уникальных пользователей', dtick=25)
fig2.update_xaxes(title_text='Дата регистрации', tickangle=45)

fig2.show()
```



Наблюдения: До 13 декабря включительно наблюдалось большое число регистраций неактивных пользователей в обеих группах (это число периодически было выше количества зарегистрировавшихся активных пользователей). 14 декабря ситуация изменилась: в группе A неактивные пользователи пропали, а количество активных пользователей существенно выросло. Предполагаем, что при записи событий возникали технические проблемы, которые были исправлены для группы A, но остались актуальными для группы B.

Если мы оставим неактивных пользователей в выборке как есть, это исказит результаты теста: ощутимо занизит конверсию для группы B и покажет далекую от реальной конверсию для группы A. Следовательно, дальнейший анализ и проверку результатов теста мы будем осуществлять на основе данных, учитывающих только активных пользователей.

In [38]:
```
# исключим из датасета неактивных пользователей

rs_test = rs_test.query('active_user == "active"').drop(columns='active_user').reset_index(drop=True)
```

In [39]:
```
# посмотрим на размер выборки и размер групп после исключения неактивных пользователей

print('Оставшееся число участников теста -', rs_test['user_id'].nunique())

users_by_group = rs_test.groupby('group').agg({'user_id':'nunique'}).rename(columns={'user_id':'users_number'})
users_by_group
```

```
Оставшееся число участников теста - 3481
```

Out[39]:

group	users_number
A	2604
B	877

Наблюдения: Группа B в итоге меньше группы A почти в 3 раза. Изменение баланса выборок и их размера снижает мощность теста, но если тестируемое изменение конверсии большое, то вероятность его обнаружения будет высока.

Общий вывод: По итогам оценки корректности проведенного теста мы выявили, что:

- Фактическое начало тестирования соответствует техническому заданию, однако период тестирования закончился раньше, чем того требовало ТЗ: 30 декабря 2020 г. вместо 4 января 2021 г. В результате число пользователей, "проживших" менее установленных по ТЗ 14 дней до остановки теста, - 2028 человек;
- Период набора новых пользователей соответствует ТЗ. В тесте участвовали пользователи из регионов, отличных от EU, и мы исключили их из анализа. Размер аудитории теста (до проверки качества данных) соответствует ТЗ - 6351 человек, пользователи разделены на 2 группы;
- Одновременно с тестом проводилась только одна маркетинговая кампания, посвященная Рождеству и Новому году, которая была запущена за 5 дней до окончания теста. Учитывая сезонный рост активности пользователей в декабре, мы считаем, что данная кампания не должна исказить результаты теста;
- Почти четверть участников нашего теста попали в другое тестирование и присутствовали там в контрольной и экспериментальной группах. Мы не располагаем информацией о втором тесте: какие изменения проверяются, каков период тестирования, - поэтому не можем сказать, что он проводился параллельно с анализируемым и был способен повлиять на наш тест. Доля пользователей, участвующих в двух тестах, и динамика их набора (регистрации) сопоставимы в группах A и B. Таким образом, предполагаем, что участие в другом тесте не должно существенно повлиять на результаты анализируемого теста;
- Участники групп A и B не пересекаются;
- До 13 декабря включительно наблюдалось большое число регистраций неактивных пользователей в обеих группах (это число было выше количества зарегистрировавшихся активных пользователей). 14 декабря ситуация изменилась: в группе A неактивные пользователи пропали, а количество активных пользователей существенно выросло. Предполагаем, что при записи событий возникали технические проблемы, которые были исправлены для группы A, но остались актуальными для группы B. Данная проблема исказит результаты теста: ощутимо занизит конверсию для группы B и покажет далекую от реальной конверсию для группы A. Следовательно, дальнейший анализ и проверку результатов теста мы будем осуществлять на основе данных, учитывающих только активных пользователей;
- После исключения неактивных пользователей аудитория теста сократилась до 3481 человека, а размер группы B стал меньше группы A почти в 3 раза.

Исследовательский анализ данных

Количество событий на пользователя в разрезе групп

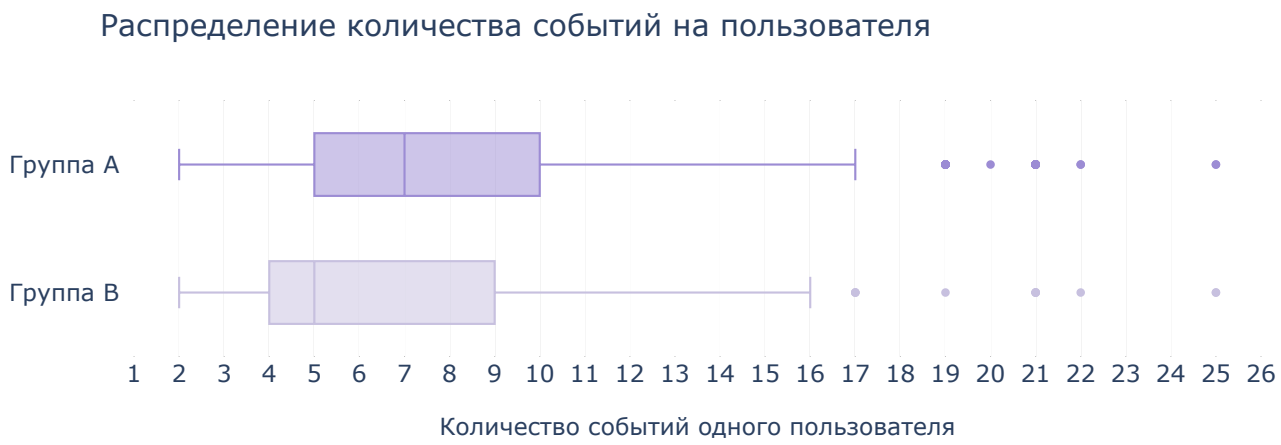
In [40]:
```
# рассчитаем число событий на пользователя в разрезе групп

events_by_user = (rs_test
                  .groupby(['group','user_id'], as_index=False)
                  .agg({'event_name':'count'})
                  .rename(columns={'event_name':'events_number'})
                 )
```

In [41]:
```
# построим график распределения числа событий на пользователя в разрезе групп

fig = go.Figure()

for i, group in enumerate(["B", "A"]):
    fig.add_trace(go.Box(x=events_by_user.query('group == @group')['events_number'],
                         marker_color=colours[i+2],
                         hoverinfo='x',
                         showlegend=False,
                         name='Группа {}'.format(group),
                         marker_size=4,
                         line_width=1)
                  )
fig.update_layout(height=300, width=700, plot_bgcolor='white', font_size=11,
                  title = {'text':'Распределение количества событий на пользователя',
                           'y':.8,
                           'x':.1,
                           'font.size':14
                          })
fig.update_xaxes(title='Количество событий одного пользователя', title_font_size=11, dtick=1)
fig.show()
```



Наблюдения: Пользователи группы A немного активнее пользователей группы B: медианное число событий для группы A - 7, максимальное стандартное (без учета выбросов) - 17, для группы B - 5 и 16 событий, соответственно.

Проверим значимость разницы в среднем числе событий на человека между группами. Поскольку в выборках A и B не наблюдается существенное число выбросов, для статистического анализа используем t-критерий Стьюдента:

- H0: Среднее количество событий на одного пользователя не различается между группами A и B
- H1: Среднее количество событий на одного пользователя в группах A и B различно

In [42]:
```
# проверим значимость разницы в среднем числе событий на человека между группами

# уровень значимости
alpha = .05

# проверка гипотезы
events_by_user_test = st.ttest_ind(events_by_user.query('group == "A"')['events_number'],
                                   events_by_user.query('group == "B"')['events_number'],
                                   equal_var=False)

print('p-значение:', events_by_user_test.pvalue)
if events_by_user_test.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
p-значение: 1.212317821640462e-23
Отвергаем нулевую гипотезу
```

Наблюдения: Полученное p-value ниже уровня значимости - отвергаем нулевую гипотезу о том, что среднее количество событий на одного пользователя не различается между группами A и B: разница статистически значима.

Количество событий по дням в разрезе групп

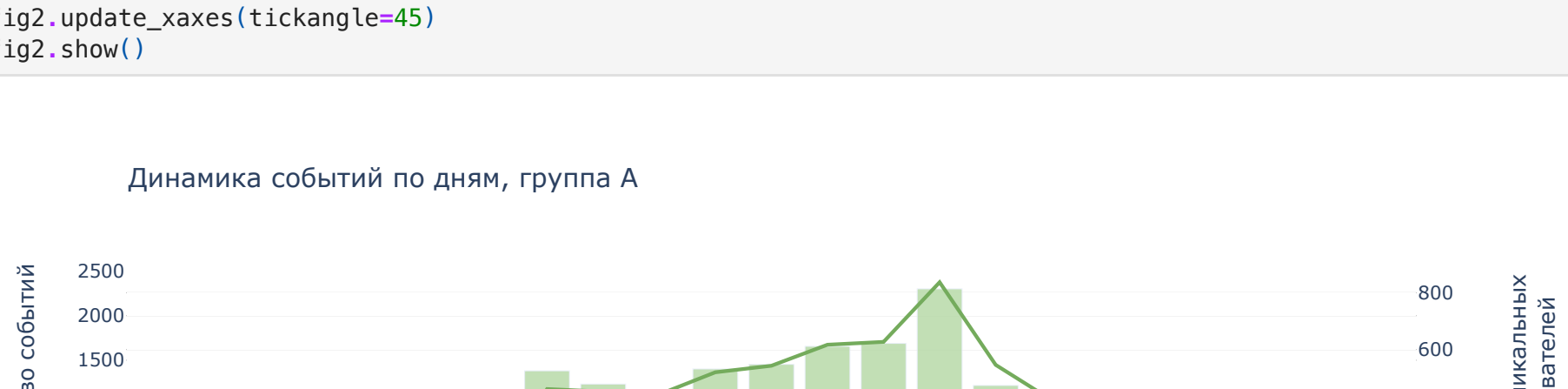
In [43]:
```
# сгруппируем данные о событиях и уникальных пользователях в разрезе групп и дат

events_daily = (rs_test
                .groupby(['group','event_date'], as_index=False)
                .agg({'event_name':'count', 'user_id':'nunique'})
                .rename(columns={'event_name':'events_number', 'user_id':'unique_users'})
               )
```

In [44]:
```
# построим график динамики числа событий и уникальных пользователей по дням

# группа A
fig1 = make_subplots(specs=[[{"secondary_y": True}]])
fig1.add_trace(go.Bar(x=events_daily.query('group == "A"')['event_date'],
                      y=events_daily.query('group == "A"')['events_number'],
                      marker_color=colours[1],
                      name='Число событий',
                      opacity=.8)
              )
fig1.add_trace(go.Scatter(x=events_daily.query('group == "A"')['event_date'],
                          y=events_daily.query('group == "A"')['unique_users'],
                          mode='lines',
                          name='Число уникальных пользователей',
                          marker_color=colours[0]),
               secondary_y=True)
fig1.update_layout(height=300, width=900, plot_bgcolor='white', font_size=10,
                   title={'text':'Динамика событий по дням, группа A',
                          'y':.85,
                          'x':.09,
                          'font.size':14},
                   legend=dict(orientation='h', y=-.3, x=0))
fig1.update_yaxes(secondary_y=False, title_text='Количество событий', range=[0,2500], dtick=500)
fig1.update_yaxes(secondary_y=True, title_text='Число уникальных<br>пользователей')
fig1.update_xaxes(tickangle=45)
fig1.show()

# группа B
fig2 = make_subplots(specs=[[{"secondary_y": True}]])
fig2.add_trace(go.Bar(x=events_daily.query('group == "B"')['event_date'],
                      y=events_daily.query('group == "B"')['events_number'],
                      marker_color=colours[2],
                      name='Число событий',
                      opacity=.8)
              )
fig2.add_trace(go.Scatter(x=events_daily.query('group == "B"')['event_date'],
                          y=events_daily.query('group == "B"')['unique_users'],
                          mode='lines',
                          name='Число уникальных пользователей',
                          marker_color=colours[3]),
               secondary_y=True)
fig2.update_layout(height=300, width=900, plot_bgcolor='white', font_size=10,
                   title={'text':'Динамика событий по дням, группа B',
                          'y':.85,
                          'x':.09,
                          'font.size':14},
                   legend=dict(orientation='h', y=-.3, x=0))
fig2.update_yaxes(secondary_y=False, title_text='Количество событий', range=[0,500], dtick=100)
fig2.update_yaxes(secondary_y=True, title_text='Число уникальных<br>пользователей', range=[0,200])
fig2.update_xaxes(tickangle=45)
fig2.show()
```



Наблюдения: Из графиков выше видно, что динамика количества событий за день зависит от числа активных пользователей. Также заметен резкий рост активности группы A с 14 декабря, причиной которого является исправление технической ошибки при записи событий (см. пункт 3.3.4).

Снижение активности в обеих группах после 22 декабря связано с прекращением набора в тест новых пользователей. До 21 числа аудитория теста росла, из-за чего наблюдается и рост активности. После прекращения набора пользователей число событий перестало расти, поскольку пользователи наиболее активны в первые дни после регистрации. См. график ниже:

In [45]:
```
# построим график распределения числа событий по дням лайфтайма в разрезе групп

fig = go.Figure()

for i, group in enumerate(["A", "B"]):
    fig.add_trace(go.Histogram(x=rs_test.query('group == @group')['lifetime'],
                               marker_color=colours[i+2],
                               name='Группа {}'.format(group),
                               hoverinfo='x+y',
                               showlegend=True)
                  )
fig.update_layout(height=350, width=800, plot_bgcolor='white', font_size=11,
                  title = {'text':'Распределение числа событий по дням лайфтайма',
                           'y':.88,
                           'x':0.09,
                           'font.size':14,
                          })
fig.update_xaxes(title='Лайфтайм', title_font_size=11, dtick=1)
fig.update_yaxes(title='Число событий', title_font_size=11)
fig.show()
```

Конверсия на разных этапах воронки в разрезе групп

In [46]:
```
# посмотрим еще раз на типы событий в воронке

list(rs_test['event_name'].unique())
```

Out[46]:
```
['registration', 'login', 'product_page', 'purchase', 'product_cart']
```

Логика воронки должна быть следующая: 1) регистрация (registration), 2) вход в личный кабинет (login), 3) просмотр страницы товара (product_page), 4) просмотр корзины (product_cart), 5) покупка (purchase).

Построим воронку продаж и посмотрим на конверсию по этапам для обеих групп.

In [47]:
```
# соберем число уникальных пользователей в разрезе групп теста и типов событий
funnel = (rs_test
          .pivot_table(index=['group','event_name'], values='user_id', aggfunc='nunique')
          .rename(columns={'user_id':'unique_users'})
          .reset_index()
         )
# рассчитаем долю пользователей группы по типам событий
funnel.loc[funnel['group'] == "A", 'total_users'] = users_by_group.loc['A', 'users_number']
funnel.loc[funnel['group'] == "B", 'total_users'] = users_by_group.loc['B', 'users_number']
funnel['share'] = funnel['unique_users'] / funnel['total_users']

# выведем результат
(funnel
 .pivot_table(index='event_name', columns='group', values=['unique_users', 'share'])
 .reindex(['registration', 'login', 'product_page', 'product_cart', 'purchase'])
 .iloc[:,[2,3,0,1]]
 .style.format({('share','A'):'{:.1%}', ('share','B'):'{:.1%}'})
)
```

Out[47]:

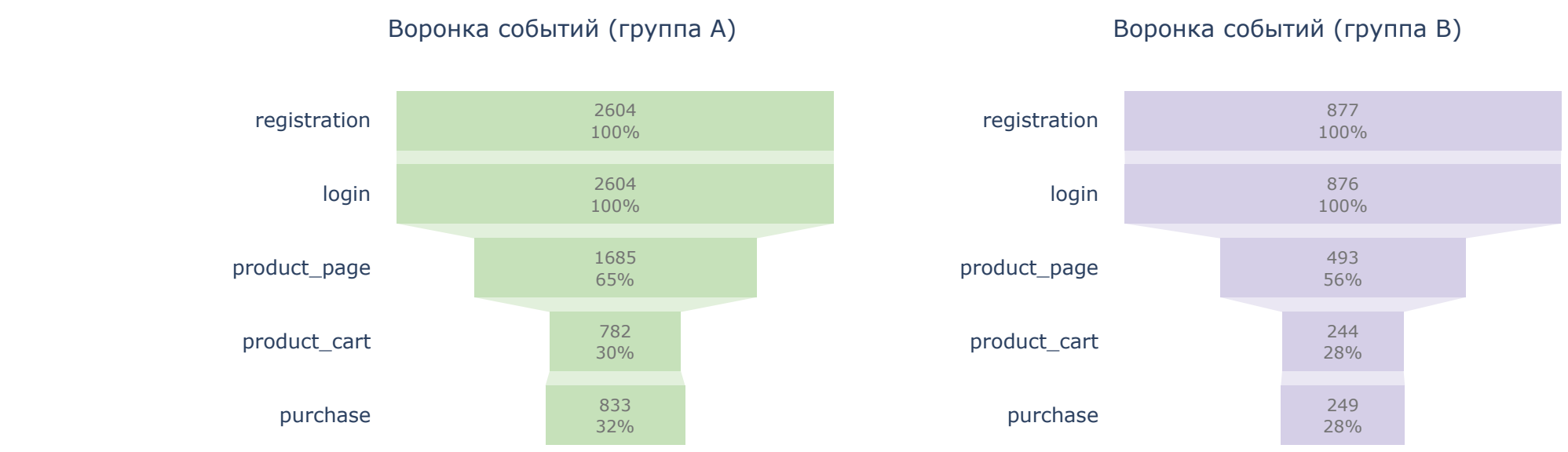
	unique_users		share	
group	A	B	A	B
event_name				
registration	2604	877	100.0%	100.0%
login	2604	876	100.0%	99.9%
product_page	1685	493	64.7%	56.2%
product_cart	782	244	30.0%	27.8%
purchase	833	249	32.0%	28.4%

In [48]:
```
# сохраним воронки по группам в отдельных датафреймах
funnel_A = (funnel.query('group == "A"').set_index('event_name')
             .reindex(['registration', 'login', 'product_page', 'product_cart', 'purchase'])
            )
funnel_B = (funnel.query('group == "B"').set_index('event_name')
             .reindex(['registration', 'login', 'product_page', 'product_cart', 'purchase'])
            )

# рисуем воронку событий
fig = make_subplots(rows=1, cols=2, horizontal_spacing=.2)

fig.add_trace(go.Funnel(y=funnel_A.index,
                        x=funnel_A['unique_users'],
                        textinfo='value + percent initial',
                        textfont={'size': 10},
                        hoverinfo='percent initial + percent previous',
                        opacity=.75,
                        marker_color=colours[1],
                        name='группа A',
                        showlegend=False
                       ),
              1, 1)
fig.add_trace(go.Funnel(y=funnel_B.index,
                        x=funnel_B['unique_users'],
                        textinfo='value + percent initial',
                        textfont={'size': 10},
                        hoverinfo='percent initial + percent previous',
                        opacity=.75,
                        marker_color=colours[2],
                        name='группа B',
                        showlegend=False
                       ),
              1, 2)
# отформатируем график
fig.update_layout(height=400, width=900, plot_bgcolor='white')

# добавляем заголовки
fig.add_annotation(text="Воронка событий (группа A)",
                   font_size=14,
                   xref="paper", yref="paper",
                   x=.01, y=1.2, showarrow=False)
fig.add_annotation(text="Воронка событий (группа B)",
                   font_size=14,
                   xref="paper", yref="paper",
                   x=.9, y=1.2, showarrow=False)
fig.show()
```



Наблюдения: Конверсия группы A на всех этапах воронки выше конверсии группы B, из чего следует, что новая рекомендательная система не оправдала ожиданий. До покупки в группе A дошло 32% пользователей, в группе B - 28.4%. Конверсия в покупку в обеих группах получилась выше конверсии в просмотр корзины, вероятно, потому что есть опция быстрой покупки: оформление заказа минуя корзину.

Общий вывод: По итогам анализа действий активных пользователей мы выявили, что:

- Пользователи группы A немного активнее пользователей группы B: медианное число событий для группы A - 7, максимальное стандартное (без учета выбросов) - 17, для группы B - 5 и 16 событий, соответственно;
- Динамика количества событий за день зависит от числа активных пользователей. Резкий рост активности группы A с 14 декабря связан с исправлением технической ошибки при записи событий;
- Конверсия группы A на всех этапах воронки выше конверсии группы B, из чего следует, что новая рекомендательная система не оправдала ожиданий. До покупки в группе A дошло 32% пользователей, в группе B - 28.4%. Конверсия в покупку в обеих группах получилась выше конверсии в просмотр корзины, вероятно, потому что есть опция быстрой покупки: оформление заказа минуя корзину.

Оценка результатов A/B-тестирования

Особенности данных

Прежде чем приступать к оценке результатов теста, вспомним и учтем следующие особенности данных:

- **Период тестирования закончился раньше, чем того требовало ТЗ**: 30 декабря 2020 г. вместо 4 января 2021 г. В результате часть пользователей "прожили" менее установленных 14 дней до остановки теста, **что может снизить наблюдаемый уровень конверсии**;
- **Почти четверть участников нашего теста попали в другое тестирование**. Мы не располагаем информацией о втором тесте: какие изменения проверяются, каков период тестирования, - поэтому не можем сказать, что он проводился параллельно с анализируемым и был способен повлиять на наш тест. Доля пользователей, участвующих в двух тестах, и динамика их набора сопоставимы в группах A и B. Таким образом, предполагаем, что **участие в другом тесте не должно существенно повлиять на результаты анализируемого теста**;
- До 13 декабря включительно наблюдалось большое число регистраций неактивных пользователей в обеих группах. 14 декабря ситуация изменилась: в группе A неактивные пользователи пропали, а количество активных пользователей существенно выросло. Предполагаем, что **при записи событий возникали технические проблемы, которые были исправлены для группы A, но остались актуальными для группы B**. Если бы мы оставили неактивных пользователей в выборке как есть, это исказило бы результаты теста: ощутимо занизило конверсию для группы B и показало далекую от реальной конверсию для группы A. Следовательно, проверку результатов теста мы будем осуществлять на основе данных, учитывающих только активных пользователей. После исключения неактивных пользователей аудитория теста сократилась до 3481 человека, а размер группы B стал меньше группы A почти в 3 раза. Таким образом, **фактический размер аудитории теста не соответствует ТЗ из-за технических проблем при логировании событий**.

Результаты теста

Подход к анализу A/B-теста
Поскольку мы сравниваем конверсию между группами, а это доля пользователей, осуществивших действие, для проверки статистической значимости разницы в конверсии между группами применим z-тест:

Нулевая гипотеза H0: доля пользователей, совершивших событие, не отличается между группами теста.
Альтернативная гипотеза H1: доля пользователей, совершивших событие, различается между группами теста.

In [49]:
```
# объявим функцию для расчета p-value и оценки статистической значимости различий между группами

def ztest_analysis(df, event, alpha):
    successes = np.array([df.query('group == "A" and event_name == @event')['unique_users'].iloc[0],
                          df.query('group == "B" and event_name == @event')['unique_users'].iloc[0]
                         ])
    trials = np.array([df.query('group == "A"')['total_users'].iloc[0],
                       df.query('group == "B"')['total_users'].iloc[0]
                      ])
    p_value = proportions_ztest(successes, trials)[1]

    print('Событие {}: \nКонверсия группы A - {:.2%} \nКонверсия группы B - {:.2%} \np-value = {:.5f}'
          .format(event,
                  df.query('group == "A" and event_name == @event')['share'].iloc[0],
                  df.query('group == "B" and event_name == @event')['share'].iloc[0],
                  p_value)
         )
    if p_value < alpha:
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница\n')
    else:
        print('Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными\n')
```

Корректировка уровня значимости
В целом нам предстоит выполнить 3 проверки на одних и тех же данных: конверсия в события product_page, product_cart, purchase. Таким образом, наш тест является множественным, и с каждой новой проверкой гипотезы растет вероятность ошибки первого рода (H0 отвергается, когда на самом деле различий между сравниваемыми группами нет).

Чтобы снизить вероятность ошибки первого рода, скорректируем уровень значимости alpha по методу Шидака. Он не так грубо снижает alpha для одного теста, как простой метод Бонферрони: чем ниже уровень значимости, тем ниже мощность теста, и тем больше различий между группами можно упустить. За базовый уровень значимости возьмем 0.05.

In [50]:
```
# корректируем alpha

alpha = .05
alpha_sidak = 1 - (1 - alpha)**(1/3)
```

In [51]:
```
# посмотрим на скорректированный показатель alpha

alpha_sidak
```

Out[51]: 0.016952427508441503

Для каждого события проверям следующую гипотезу:

H0: Группы A и B демонстрируют одинаковую конверсию в событие.

H1: Группы A и B демонстрируют разную конверсию в событие.

In [52]:
```
# тестируем гипотезы для групп A и B

for event in ['product_page', 'product_cart', 'purchase']:
    ztest_analysis(funnel, event, alpha_sidak)
```

```
Событие product_page:
Конверсия группы A - 64.71%
Конверсия группы B - 56.21%
p-value = 0.00001
Отвергаем нулевую гипотезу: между долями есть значимая разница

Событие product_cart:
Конверсия группы A - 30.03%
Конверсия группы B - 27.82%
p-value = 0.21469
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие purchase:
Конверсия группы A - 31.99%
Конверсия группы B - 28.39%
p-value = 0.04652
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Наблюдения: Статистически значимая разница между уровнем конверсии в группах A и B наблюдается только на шаге просмотра страницы продукта (product_page). В остальном изменение системы рекомендаций не повлияло на уровень конверсии.

Вывод

Мы исследовали корректность A/B теста и выявили, что **тест был проведен с нарушениями**, а именно:

- Период тестирования закончился раньше, чем того требовало ТЗ: 30 декабря 2020 г. вместо 4 января 2021 г. В результате часть пользователей "прожили" менее установленных 14 дней до остановки теста, что может снижать наблюдаемый уровень конверсии;
- В тесте участвовали пользователи из регионов, отличных от EU, но мы исключили их из анализа;
- До 13 декабря включительно наблюдалось большое число регистраций неактивных пользователей в обеих группах. С 14 декабря ситуация изменилась: в группе A неактивные пользователи пропали, а количество активных пользователей существенно выросло. Предполагаем, что при записи событий возникали технические проблемы, которые были исправлены для группы A, но остались актуальными для группы B. Это снижает качество данных и искажает результаты теста: ощутимо занижает конверсию для группы B и показывает далекую от реальной конверсию для группы A;
- Из-за проблемы качества данных мы были вынуждены исключить неактивных пользователей из анализа, в результате чего аудитория теста сократилась до 3481 человека, что не соответствует ТЗ, а размер группы B стал меньше группы A почти в 3 раза.

Кроме того, **почти четверть участников нашего теста попали в другое тестирование**. Мы не располагаем информацией о втором тесте: какие изменения проверяются, каков период тестирования, - поэтому не можем сказать, что он проводился параллельно с анализируемым и был способен повлиять на наш тест. Доля пользователей, участвующих в двух тестах, и динамика их набора (регистрации) сопоставимы в группах A и B. Таким образом, предполагаем, что участие в другом тесте не должно существенно повлиять на результаты анализируемого теста.

Результаты теста вышли следующими:

- Наблюдаемая конверсия группы A на всех этапах воронки выше конверсии группы B, из чего следует, что новая рекомендательная система в целом не оправдала ожиданий. До покупки в группе A дошло 32% пользователей, в группе B - 28.4%;
- Статистически значимая разница между уровнем конверсии в группах A и B наблюдается только на шаге просмотра страницы продукта (в группе A конверсия - 64.7%, в группе B - 56.2%). В остальном изменение системы рекомендаций не повлияло на уровень конверсии.

Для будущих тестов **рекомендуем**:

- соблюдать заданный период тестирования для сокращения вероятных искажений результата,
- отслеживать наличие технических проблем при записи логов во время теста (например, сопоставлять динамику регистраций и активности между группами),
- стараться избегать ситуаций, когда пользователи, попавшие в параллельные тестирования, присутствуют в экспериментальных группах (присутствие в контрольной группе теста ок).

In [ ]: