Оценка результатов A/B - тестирования

Описание проекта

Задача — провести оценку результатов A/B-теста. В вашем распоряжении есть датасет с действиями пользователей, техническое задание и несколько вспомогательных датасетов.

- Оцените корректность проведения теста
- Проанализируйте результаты теста

Чтобы оценить корректность проведения теста, проверьте:

- пересечение тестовой аудитории с конкурирующим тестом,
- совпадение теста и маркетинговых событий, другие проблемы временных границ теста.

Техническое задание

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`.

Описание данных

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

Структура файла:

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

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

Структура файла:

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

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

Структура файла:

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

`final_ab_participants.csv` — таблица участников тестов.

Структура файла:

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

Импорт библиотек и чтение данных

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats as st
import numpy as np
import plotly.express as px
from plotly import graph_objects as go
from datetime import datetime
from datetime import date
from datetime import timedelta
import plotly.graph_objects as go
from statsmodels.stats.proportion import proportions_ztest
```

In [2]:
```
ab_project_marketing_events = pd.read_csv('https://code.s3.yandex.net/datasets/ab_pro
final_ab_new_users = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_new_us
final_ab_events = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_events.cs
final_ab_participants = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_par
```

In [3]:
```
ab_project_marketing_events
```

Out[3]:

	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
5	Black Friday Ads Campaign	EU, CIS, APAC, N.America	2020-11-26	2020-12-01
6	Chinese New Year Promo	APAC	2020-01-25	2020-02-07
7	Labor day (May 1st) Ads Campaign	EU, CIS, APAC	2020-05-01	2020-05-03
8	International Women's Day Promo	EU, CIS, APAC	2020-03-08	2020-03-10
9	Victory Day CIS (May 9th) Event	CIS	2020-05-09	2020-05-11
10	CIS New Year Gift Lottery	CIS	2020-12-30	2021-01-07
11	Dragon Boat Festival Giveaway	APAC	2020-06-25	2020-07-01
12	Single's Day Gift Promo	APAC	2020-11-11	2020-11-12
13	Chinese Moon Festival	APAC	2020-10-01	2020-10-07

In [4]:
```
final_ab_new_users.head()
```

Out[4]:

	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

In [5]:
```
final_ab_events.head()
```

Out[5]:

	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

In [6]:
```
final_ab_participants.head()
```

Out[6]:

	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

Предобработка

- Исследуйте данные:
  - Требуется ли преобразование типов?
  - Опишите природу пропущенных значений и дубликатов, если их обнаружите.

In [7]:
```
# функция для краткого ознакомления с датасетом
def research(df):
    print(df.info(), '\n')
    print('Дуликаты:', df.duplicated().sum())
    print('\n Пропуски:\n', df.isna().sum())
    display(df.head())
```

In [8]:
```
research(ab_project_marketing_events)
```

```
<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

Дуликаты: 0

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

	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

In [9]:
```
research(final_ab_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
None

Дуликаты: 0

 Пропуски:
 user_id       0
first_date    0
region        0
device        0
dtype: int64
```

	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

In [10]:
```
research(final_ab_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
None

Дуликаты: 0

 Пропуски:
 user_id            0
event_dt           0
event_name         0
details       377577
dtype: int64
```

	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

In [11]:
```
research(final_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
None

Дуликаты: 0

 Пропуски:
 user_id    0
group      0
ab_test    0
dtype: int64
```

	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

Первое что бросается в глаза - типы данных для колонок с датами. Также в датафрейме `final_ab_events` есть пропуски. Сначала преобразую типы данных, а далее подробнее ознакомимся с пропусками

In [12]:
```
# Приведем данные к нужному типу
ab_project_marketing_events['start_dt'] = pd.to_datetime(ab_project_marketing_events[
ab_project_marketing_events['finish_dt'] = pd.to_datetime(ab_project_marketing_events
final_ab_new_users['first_date'] = pd.to_datetime(final_ab_new_users['first_date'])
final_ab_events['event_dt'] = pd.to_datetime(final_ab_events['event_dt'])
```

Теперь посмотрим на пропуски в датафрейме `final_ab_events`

In [13]:
```
def count_nulls(s):
    return s.size - s.count()
```

In [14]:
```
final_ab_events.groupby('event_name', as_index=False, dropna=False).agg({'details': c
```

Out[14]:

	event_name	details
0	login	189552
1	product_cart	62462
2	product_page	125563
3	purchase	0

Пропуски встречаются везде, кроме данных о покупках. Вероятно, в дополнительных событиях указывается лишь стоимость покупки. Предлагаю заменить пропуски на 'no info'

In [15]:
```
final_ab_events = final_ab_events.fillna('no info')
```

In [16]:
```
# И сразу добавим колонку с простой датой
final_ab_events['event_date'] = final_ab_events['event_dt'].dt.date
```

Оценка корректности теста

- Оцените корректность проведения теста. Обратите внимание на:
  - Соответствие данных требованиям технического задания. Проверьте корректность всех пунктов технического задания.
  - Время проведения теста. Убедитесь, что оно не совпадает с маркетинговыми и другими активностями.
  - Аудиторию теста. Удостоверьтесь, что нет пересечений с конкурирующим тестом и нет пользователей, участвующих в двух группах теста одновременно. Проверьте равномерность распределения по тестовым группам и правильность их формирования.

Для начала соберем общий датафрейм. Соединими информацию о событиях, данные о пользователях и данные о тестах в один датафрейм

In [17]:
```
final_ab_new_users
```

Out[17]:

	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
...	...	...	...	...
61728	1DB53B933257165D	2020-12-20	EU	Android
61729	538643EB4527ED03	2020-12-20	EU	Mac
61730	7ADEE837D5D8CBBD	2020-12-20	EU	PC
61731	1C7D23927835213F	2020-12-20	EU	iPhone
61732	8F04273BB2860229	2020-12-20	EU	Android

61733 rows × 4 columns

In [18]:
```
df = (
    pd.merge(pd.merge(final_ab_events, final_ab_participants[['user_id', 'group', 'ab
                      on='user_id', how='right'),
    final_ab_new_users[['user_id', 'first_date', 'region', 'device']], on='user_id',
)
```

In [19]:
```
# Проверка на пропуски и дубли
research(df)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110368 entries, 0 to 110367
Data columns (total 10 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   user_id     110368 non-null  object        
 1   event_dt    106625 non-null  datetime64[ns]
 2   event_name  106625 non-null  object        
 3   details     106625 non-null  object        
 4   event_date  106625 non-null  object        
 5   group       110368 non-null  object        
 6   ab_test     110368 non-null  object        
 7   first_date  110368 non-null  datetime64[ns]
 8   region      110368 non-null  object        
 9   device      110368 non-null  object        
dtypes: datetime64[ns](2), object(8)
memory usage: 9.3+ MB
None

Дуликаты: 0

 Пропуски:
 user_id          0
event_dt       3743
event_name     3743
details        3743
event_date     3743
group             0
ab_test           0
first_date        0
region            0
device            0
dtype: int64
```

	user_id	event_dt	event_name	details	event_date	group	ab_test	f
0	D1ABA3E2887B6A73	2020-12-07 14:43:27	purchase	99.99	2020-12-07	A	recommender_system_test	
1	D1ABA3E2887B6A73	2020-12-25 00:04:56	purchase	4.99	2020-12-25	A	recommender_system_test	
2	D1ABA3E2887B6A73	2020-12-07 14:43:29	product_cart	no info	2020-12-07	A	recommender_system_test	
3	D1ABA3E2887B6A73	2020-12-25 00:04:57	product_cart	no info	2020-12-25	A	recommender_system_test	
4	D1ABA3E2887B6A73	2020-12-07 14:43:27	product_page	no info	2020-12-07	A	recommender_system_test	

Есть пропуски в колонках, связанных с событиями. Веротяно, существуют пользователи, либо не совершившие дейсвтий, либо информация об их активности просто отсутствует. Немного позже проверим как эти пользователи распределены между группами заданного в ТЗ теста

Данные собраны. Начнем проверять соответствие ТЗ. Во-первых проверим какие тесты у нас доступны и есть ли пересечение между тестами и группами

In [20]:
```
df.groupby('ab_test', as_index=False)['user_id'].nunique()
```

Out[20]:

	ab_test	user_id
0	interface_eu_test	11567
1	recommender_system_test	6701

Всего есть данные о двух тестах.
Посмотрим есть ли пользователи, участвуют в нескольких тестах

In [21]:
```
df.groupby('user_id', as_index=False)['ab_test'].nunique().loc[lambda x: x['ab_test']
```

Out[21]:

	user_id	ab_test
2	001064FEAAB631A1	2
10	00341D8401F0F665	2
12	003B6786B4FF5B03	2
29	0082295A41A867B5	2
46	00E68F103C66C1F7	2
...	...	...
16638	FF7BE2897FC0380D	2
16644	FF9A81323FA67D6E	2
16652	FFC53FD45DDA5EE8	2
16662	FFED90241D04503F	2
16664	FFF28D02B1EACBE1	2

1602 rows × 2 columns

1602 пользователя участвуют сразу в обоих тестах. Считаю, что это не очень критично, тк тесты запускают достаточно часто и пользователь может участвовать сразу в нескольких тестах.
Поскольку ТЗ говорит о том, что нас интересует только `recommender_system_test`, предлагаю сразу удалить `interface_eu_test` из датафрейма.

In [22]:
```
# список пользователей теста recommender_system_test
users_recommender_system_test = (
    final_ab_participants[final_ab_participants['ab_test'] == 'recommender_system_tes
)
```

In [23]:
```
# распределение пользователей по группам.
(
    final_ab_participants[(final_ab_participants['ab_test'] == 'interface_eu_test')
                          & (final_ab_participants['user_id'].isin(users_recommender_
).groupby('group', as_index=False)['user_id'].nunique()
```

Out[23]:

	group	user_id
0	A	819
1	B	783

Пользователи тесат `recommender_system_test` попали в обе группы теста `interface_eu_test`, причем в достаточно близком кол-ве

In [24]:
```
df = df[df['ab_test'] == 'recommender_system_test']
```

Теперь подцепим информацию о кол-ве тестах, в которых участвует пользователь и посмотрим сколько пользователей из каждой группы участвуют сразу в 2х тестах

In [25]:
```
df = df.merge(final_ab_participants.groupby('user_id', as_index=False)['ab_test'].nun
              .rename(columns={'ab_test': 'count_ab_test'}), on='user_id', how='left'
df.query('count_ab_test > 1').groupby('group', as_index=False)['user_id'].nunique()
```

Out[25]:

	group	user_id
0	A	921
1	B	681

В группе А на 240 пользователей больше участвует сразу в двух тестах. Будем иметь это в виду в дальнейшем

Теперь посмотрим на кольичество пользователей в каждой группе

In [26]:
```
df.groupby('group', as_index=False)['user_id'].nunique()
```

Out[26]:

	group	user_id
0	A	3824
1	B	2877

Количество участников более 6000, что соответствует ТЗ

Проверим пересечение пользователей между группами

In [27]:
```
df.groupby('user_id', as_index=False)['group'].nunique().loc[lambda x: x['group'] > 1
```

Out[27]:

user_id	group

С этой точки зрения все корректно. Пересечения между группами нет.

Также тут важно вспонмить, что есть пользователи, о действиях которых нам ничего не известно. Посмотрим как они распределены между группами

In [28]:
```
df.query('event_dt.isna()').groupby('group', as_index=False)['user_id'].count()
```

Out[28]:

	group	user_id
0	A	1077
1	B	1949

Таких пользователей достаточно много. Но поскольку конверсию из регистрации в логирование нам оценивать не предстоит, то предлагаю пока оставить таких пользоавателей, чтобы соответствовать условиям ТЗ

Следующий пункт ТЗ: Дата остановки набора новых пользователей 21 декабря Проверим это

In [29]:
```
display(df['first_date'].min())
display(df['first_date'].max())
```

```
Timestamp('2020-12-07 00:00:00')
Timestamp('2020-12-21 00:00:00')
```

Все новые пользователи для теста дейсвтительно зарегистрированы до 21 декабря. Все ок

Следующий пункт ТЗ на проверку: Дата запуска 7 декабря, дата остановки 4 января.
Проверим

In [30]:
```
display(final_ab_events['event_dt'].min())
display(final_ab_events['event_dt'].max())
```

```
Timestamp('2020-12-07 00:00:33')
Timestamp('2020-12-30 23:36:33')
```

Фактической датой остановки теста будет 31 декабря. Возможно, в период с 21 декабря по 4 января были технические проблемы с сервисом и пользователи не могли им пользоваться. Возможно, была проблема со сбором данных.
Это может несколько повлиять на исследование в том плане, что например пользователь, зарегистрировавшийся 24 декабря уже "не доживет" до 14го дня лайфтама, который задан по ТЗ

В любом случае будем считать 31 декабря остановкой теста.

Проверим еще сразу пункта ТЗ:

- время набора новых пользователей с 7 по 21 декабря
- 15% новых пользователей из региона EU

По условию, время набора новых пользователей с 7 по 21 декабря. Проверим

In [31]:
```
display(final_ab_new_users['first_date'].min())
display(final_ab_new_users['first_date'].max())
```

```
Timestamp('2020-12-07 00:00:00')
Timestamp('2020-12-23 00:00:00')
```

Как мы видим, есть пользователи, которые регистрировались после 21го декабря.
Теперь проверим след. пункт (15% пользователей), но уже с учетом того, что в датафрейме с новыми пользователями есть "лишние"

In [32]:
```
# всего новых пользователей из Европы
(
    final_ab_new_users[(final_ab_new_users['region'] == 'EU') &
                       (final_ab_new_users['first_date'] <= ('2020-12-21'))]['user_id
```

Out[32]:
```
42340
```

In [33]:
```
# Пользователей из Европы в тесте
df[df['region'] == 'EU']['user_id'].nunique()
```

Out[33]:
```
6351
```

In [34]:
```
# Отношение пользователей из Европы в тесте к числу новых пользователей из Европы
(
    df[df['region'] == 'EU']['user_id'].nunique() /
    (
    final_ab_new_users[(final_ab_new_users['region'] == 'EU') &
                       (final_ab_new_users['first_date'] <= ('2020-12-21'))]['user_id
)
```

Out[34]:
```
0.15
```

В тесте участвуют ровно 15% пользователей из Европы.
Также посмотрим какие еще регионы представлены в тесте

In [35]:
```
(
    df.groupby('region', as_index=False)['user_id'].nunique()
    .assign(ratio=lambda x: x['user_id'] / x['user_id'].sum())
)
```

Out[35]:

	region	user_id	ratio
0	APAC	72	0.010745
1	CIS	55	0.008208
2	EU	6351	0.947769
3	N.America	223	0.033279

Пользователей из Европы 95%, но в тесте так же есть и представители других регионов. Посколько ЦА теста именно Европейцы, предлагаю избавиться от отсальных регионов

In [36]:
```
df = df[df['region'] == 'EU']
(
    df.groupby('region', as_index=False)['user_id'].nunique()
)
```

Out[36]:

	region	user_id
0	EU	6351

Пользователей из Европы в тесте 6351. Это удовлетворяет условиям ТЗ.

ТЗ по пунктам на соответствие данным выполнен. Далее проверим время проведения теста и убедимся, что оно не совпадает с маркетинговыми и другими активностями

In [37]:
```
# выведем все декабрьские маркетинговые активности
ab_project_marketing_events[ab_project_marketing_events['start_dt'].dt.month == 12]
```

Out[37]:

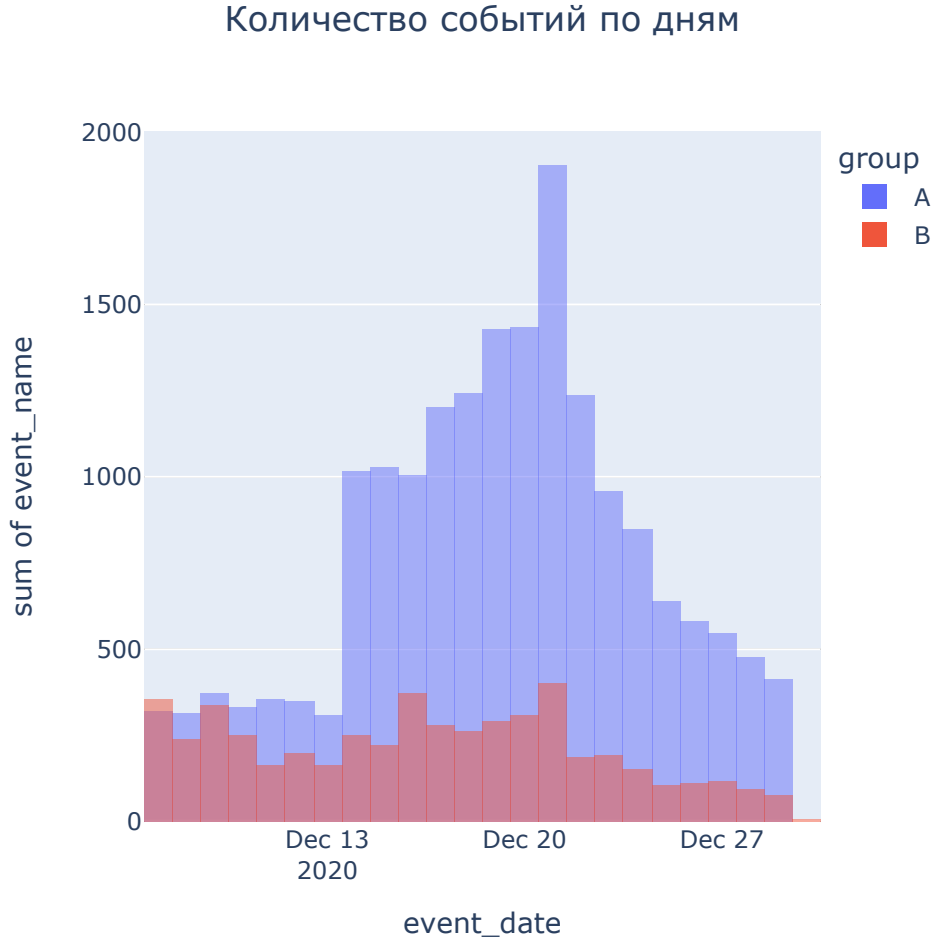
	name	regions	start_dt	finish_dt
0	Christmas&New Year Promo	EU, N.America	2020-12-25	2021-01-03
10	CIS New Year Gift Lottery	CIS	2020-12-30	2021-01-07

In [38]:
```
# Сколько останется пользователей, если отсечь данные до маркетинговых событий
df[df['event_dt'] <= '2020-12-24']['user_id'].nunique()
```

Out[38]: 3481

Маркетинговые компании достаточно обычное явление, особенно в канун Новогодних праздников. Можно так же посмотреть на распределение количества событий по дням для обоих групп и посмотрет как маркетинговые события повлияли на пользователей

In [39]:
```
fig = px.histogram(df.groupby(['group', 'event_date'], as_index=False)['event_name'].
                   x="event_date", y='event_name', color="group", nbins=30, opacity=0
                  title='Количество событий по дням')
fig.update_layout(title_x=0.5)
fig.update_layout(barmode='overlay')
fig.show()
```



Данные по обоим группам распределены достаточно равномерно относительно друг друга. Маркетинговые акции точно не оказали явного влияния на поведение пользователей. Итого у нас есть выбор: отчесь данные по датам маркетинговых событий и нарушить условие из ТЗ о минимальном кол-ве пользователей для теста, или проигнорировать пункт о пересечении с маркетинговыми событиями.

Думаю, верным решением будет оставить пересечение с маркетинговыми событиями

Еще один важный пункт ТЗ звучит так: **Ожидаемый эффект: за 14 дней с момента регистрации в системе пользователи покажут улучшение каждой метрики не менее, чем на 10%**

Соответсвенно, нам надо для каждого пользователя отчесь события, которые он совершил позже, чем через 14 дней после регистрации.
Для этого добавлю колонку с датой регистрации + 14 дней и отфильтрую по ней события для каждого пользователя

In [40]:
```
df['last_date_fot_test'] = df['first_date'] + pd.to_timedelta(14, unit="D")
```

Посмотрим сколько пользователей будет при условии выполнения данного пункта ТЗ

In [41]:
```
df[df['event_date'] <= df['last_date_fot_test']]['user_id'].nunique()
```

Out[41]:
```
3481
```

In [42]:
```
df[df['event_date'] <= df['last_date_fot_test']].groupby('group', as_index=False)['us
```

Out[42]:

	group	user_id
0	A	2604
1	B	877

В действительности мы сможем оценивать результаты теста лишь по 3481 пользователю.

Проведу исследовательский анализ без отсечения этих пользователей, а перед оценкой теста уберу их, чтобы выполнить данный пункт ТЗ

Исследовательский анализ данных

Количество событий на пользователя

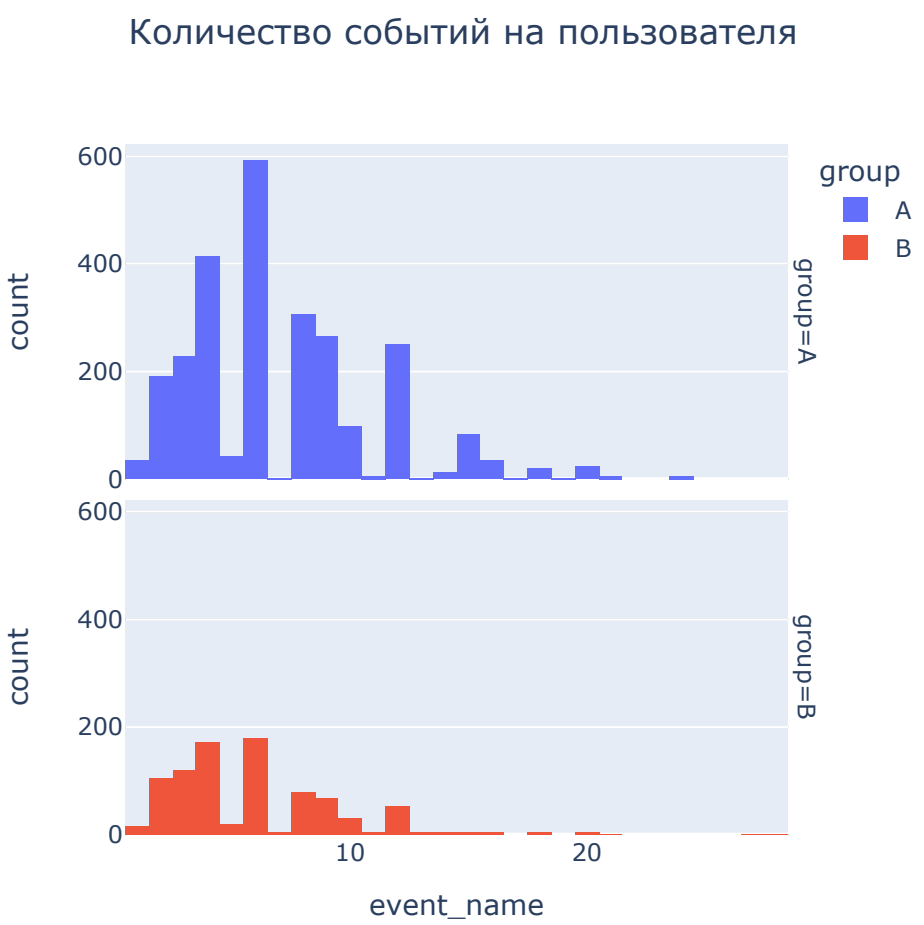
Посмотрим на количество событий для каждого пользователя.

In [43]:
```
# Гистрограмма количества дейсвтий на пользователя
fig_01 = px.histogram(df.groupby(['group', 'user_id'], as_index=False)['event_name'].
                   x="event_name", facet_col="group", color='group',
                  title='Количество событий на пользователя')
fig_01.update_layout(title_x=0.5)
fig_01.update_layout(barmode='overlay')
fig_01.show()
```

Как мы отмечали ранее, есть пользователи, которые не совершают дейсвтий. Предлагаю посмотреть на распределение без учета таких пользователей.

In [44]:
```
# Гистрограмма количества дейсвтий на пользователя без учета пользователей без дейсвт
```

```
fig_01 = px.histogram(df.groupby(['group', 'user_id'], as_index=False)['event_name'].
                      .query('event_name > 0'),
                    x="event_name", facet_row="group", color='group',
                    title='Количество событий на пользователя')
fig_01.update_layout(title_x=0.5)
fig_01.show()
```



In [45]:
```
# Посмотрим на основные статистики
display(df.groupby(['group', 'user_id'], as_index=False)['event_name'].count()
        .query('group == "A" and event_name > 0').describe().T)
display(df.groupby(['group', 'user_id'], as_index=False)['event_name'].count()
        .query('group == "B" and event_name > 0').describe().T)
```

	count	mean	std	min	25%	50%	75%	max
event_name	2604.0	7.031106	3.872263	1.0	4.0	6.0	9.0	24.0

	count	mean	std	min	25%	50%	75%	max
event_name	877.0	5.827822	3.492164	1.0	3.0	6.0	8.0	28.0

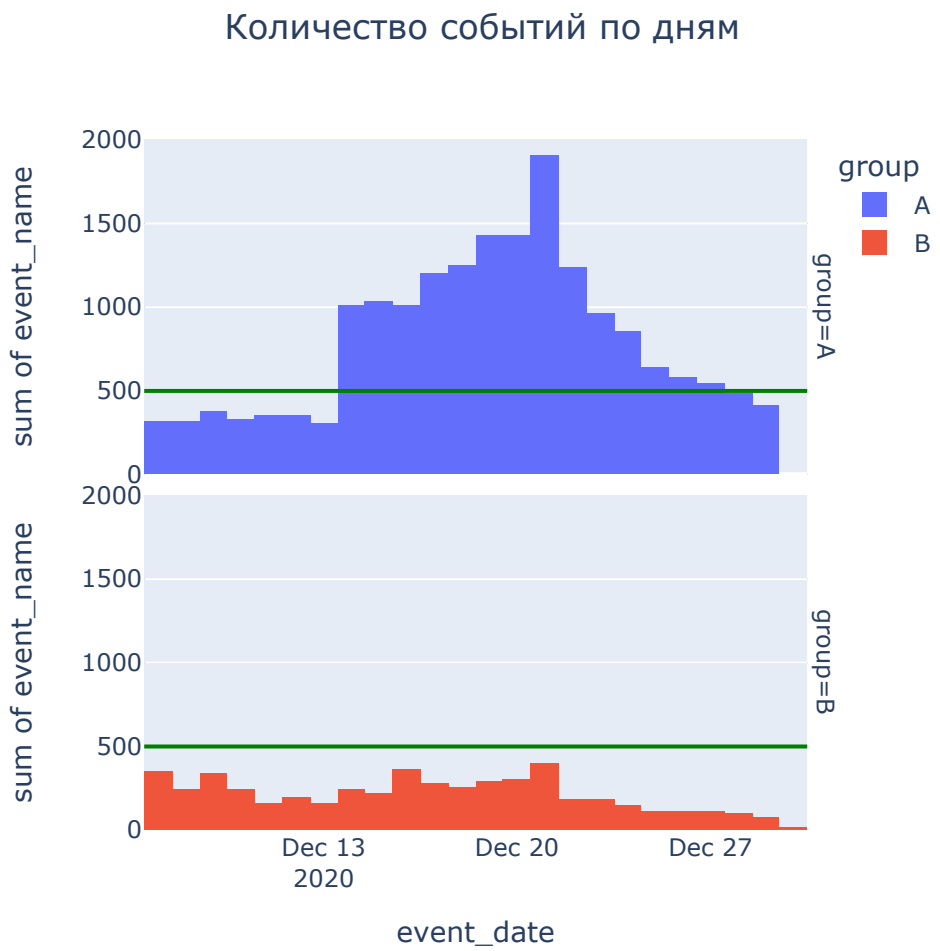
Распределния в целом схожи друг с другом. Если сравнивать основные статистики, то в группе А среднее количество дейсвтий на пользователя больше, чем в группе В (7 против почти 6). Однако медиана у обеих групп равна 6. Но и в целом пользователи из группы А совершают почти в 3 раза больше действий
При достаточно больших выборках, коими являются наши выборки, и такой картине распредления, можно сказать, что количество событий на пользователя распределены практически одинаково.

События по дням

In [46]:
```
# Гистрограмма количества дейсвтий на пользователя без учета пользователей без дейсвт
fig_02 = px.histogram(df.groupby(['group', 'event_date'], as_index=False)['event_name
                   x="event_date", y="event_name", facet_row="group", color='group',
                  title='Количество событий по дням')
fig_02.update_layout(title_x=0.5)

# Добавим линию среднего значения кол-ва дейсвтий в день
fig_02.add_hline(y=df.groupby(['group', 'event_date'], as_index=False)['event_name']
                 .count()['event_name'].mean(), line_color="green")

fig_02.show()
```



In [47]:
```
# Посмотрим на основные статистики
display(df.groupby(['group', 'event_date'], as_index=False)['event_name'].count()
        .query('group == "A"').describe().T)
display(df.groupby(['group', 'event_date'], as_index=False)['event_name'].count()
        .query('group == "B"').describe().T)
```

	count	mean	std	min	25%	50%	75%	max
event_name	23.0	796.043478	459.349488	308.0	363.5	639.0	1115.0	1903.0

	count	mean	std	min	25%	50%	75%	max
event_name	24.0	212.958333	102.057307	4.0	140.25	210.5	282.75	401.0

Изучили распределение событий по дням. Заметно, что пользователи из группы А совершают намного больше дейсвтий. Особенно это заметно с 14го декабря, когда кол-во событий для этой группы перевалило за 1000 в день и такой показатель держался до 23 декабря. Без этого скачка данные были бы распределены более гладко и имели бы больше сходства с группой В. Возможно, дело в том, что в группе а больше пользователей, которые участвуют в двух тестах

одновременно. 30 декабря в группе A вообще не было событий, а в группе B лишь 4. Максимальное количество действий в обоих группах было совершено 21го декабря. Никаких маркетинговых событий, объясяющих это явление в компании запланирвоа не было.

Все основные статистики выше у группы A:

- Среднее 796 против 212
- Медиана 639 против 210

Пользователи по устройствам

Теперь посмотрим на пользователей в разрезе устройств.

In [48]:
```
(
    df.groupby('device', as_index=False).agg({'user_id': 'nunique', 'event_name': 'co
    .assign(ratio_users=lambda x: x['user_id'] / x['user_id'].sum())
    .assign(event_by_users=lambda x: x['event_name'] / x['user_id'])
)
```

Out[48]:

	device	user_id	event_name	ratio_users	event_by_users
0	Android	2818	10264	0.443710	3.642300
1	Mac	604	2263	0.095103	3.746689
2	PC	1621	6012	0.255235	3.708822
3	iPhone	1308	4881	0.205952	3.731651

Пользователей с андройдом больше всех - почти 45% от общего числа. Интресно то, что пользователей компьютеров (ПК и мак) в сумме около 35%. Мир все стремительнее стремится к "мобильности".
Что касается активности, то тут нет определнной разницы по устройствам - пользователи всех устрйств примерно одинаково активны.
Теперь посмотрим на такие же показатели в разрезе групп

In [49]:
```
display(
    df.query('group == "A"')
    .groupby('device', as_index=False).agg({'user_id': 'nunique', 'event_name': 'coun
    .assign(ratio_users=lambda x: x['user_id'] / x['user_id'].sum())
    .assign(event_by_users=lambda x: x['event_name'] / x['user_id'])
)
display(
    df.query('group == "B"')
    .groupby('device', as_index=False).agg({'user_id': 'nunique', 'event_name': 'coun
    .assign(ratio_users=lambda x: x['user_id'] / x['user_id'].sum())
    .assign(event_by_users=lambda x: x['event_name'] / x['user_id'])
)
```

	device	user_id	event_name	ratio_users	event_by_users
0	Android	1590	7910	0.437534	4.974843
1	Mac	354	1855	0.097413	5.240113
2	PC	964	4823	0.265272	5.003112
3	iPhone	726	3721	0.199780	5.125344

	device	user_id	event_name	ratio_users	event_by_users
0	Android	1228	2354	0.451969	1.916938
1	Mac	250	408	0.092013	1.632000
2	PC	657	1189	0.241811	1.809741
3	iPhone	582	1160	0.214207	1.993127

Тут картина еще более показательна, чем в исследовании по регионам. Группа A намного активнее. Наименее активыне пользователи группы A (андроид) соврешают почти по 5 действий в среднем, в то время как самые активные пользователи группы B (айфон) совершают менее 2х действий в среднем.

Предлагаю взглянуть на сводник с учетом фильтра. (оставим только тех, пользователей которые совершили минимум 1 действие)

In [50]:
```
display(
    df[~df['event_name'].isna()]
    .query('group == "A"')
    .groupby('device', as_index=False).agg({'user_id': 'nunique', 'event_name': 'coun
    .assign(ratio_users=lambda x: x['user_id'] / x['user_id'].sum())
    .assign(event_by_users=lambda x: x['event_name'] / x['user_id'])
)
display(
    df[~df['event_name'].isna()]
    .query('group == "B"')
    .groupby('device', as_index=False).agg({'user_id': 'nunique', 'event_name': 'coun
    .assign(ratio_users=lambda x: x['user_id'] / x['user_id'].sum())
    .assign(event_by_users=lambda x: x['event_name'] / x['user_id'])
)
```

	device	user_id	event_name	ratio_users	event_by_users
0	Android	1139	7910	0.437404	6.944688
1	Mac	255	1855	0.097926	7.274510
2	PC	689	4823	0.264593	7.000000
3	iPhone	521	3721	0.200077	7.142035

	device	user_id	event_name	ratio_users	event_by_users
0	Android	405	2354	0.461802	5.812346
1	Mac	74	408	0.084379	5.513514
2	PC	212	1189	0.241733	5.608491
3	iPhone	186	1160	0.212087	6.236559

Здесь картина уже более ровная. Но в любом случае пользователи группы A немного активнее. Самыми активными в группе A являются пользователи маков, а в группе B андроидов

Воронка

Изучим общую воронку событий

In [51]:
```
funnel = (
    df.groupby('event_name', as_index=False)['user_id'].nunique()
    .sort_values('user_id', ascending=False)
```

```
#     процент от предыдущего действия
    .assign(ratio=lambda x: round((x['user_id'] / x['user_id'].shift(1)), 2))
    .reset_index(drop=True)
    .fillna(100)
)
funnel
```

Out[51]:

	event_name	user_id	ratio
0	login	3481	100.00
1	product_page	2178	0.63
2	purchase	1082	0.50
3	product_cart	1026	0.95

Воронка выглядит алогичной. Если по первым двум шагам все понятно (логин, страница продукта), то далее не совсем логично. Дело в том, что дейсвтий с заказами больше, чем переходов в корзину. Скорее всего на сервисе предусмотрен функционал, благодаря которому можно оформить заказ без перехода в корзину.
Предлагаю составить логичную воронку со следующим порядком шагов - логирование, страница продукта, корзина, заказ. Пусть выглядеть она будет не совсем привычно, но зато логика будет соблюдена.

In [52]:
```
# Добавим колонку с порядком дейсвтий
funnel['event_number'] = (
    funnel['event_name'].replace({'login':0, 'product_page':1, 'product_cart':2, 'pur
)
funnel = (
    funnel.sort_values('event_number')
    .assign(ratio_new=lambda x: round((x['user_id'] / x['user_id'].shift(1)), 2))
    .fillna(100)
)
funnel
```

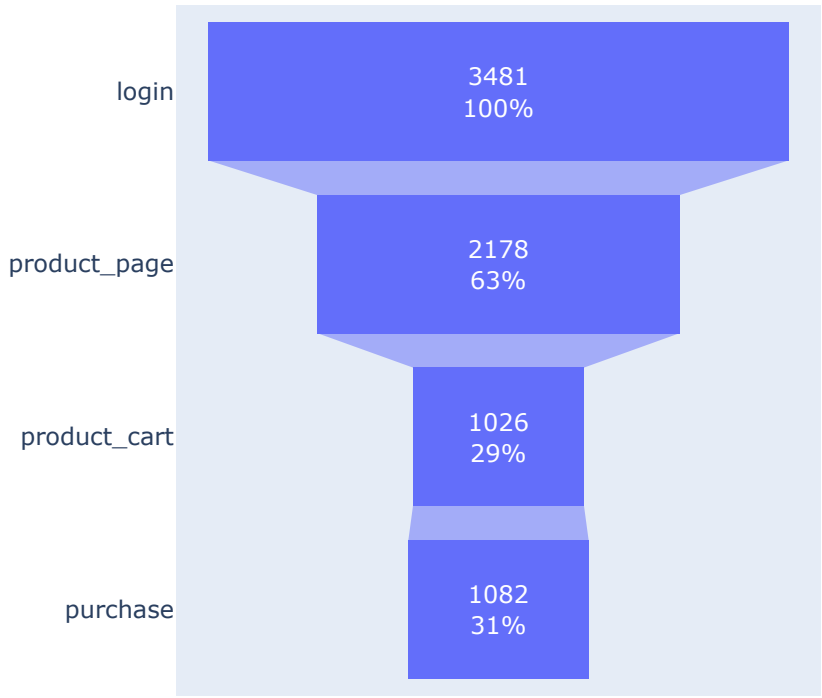
Out[52]:

	event_name	user_id	ratio	event_number	ratio_new
0	login	3481	100.00	0	100.00
1	product_page	2178	0.63	1	0.63
3	product_cart	1026	0.95	2	0.47
2	purchase	1082	0.50	3	1.05

In [53]:
```
# Теперь визуализируем воронку
fig_03 = go.Figure(
    go.Funnel(
        y=funnel['event_name'],
        x=funnel['user_id'],
        textposition = "inside",
        textinfo = "value+percent initial",
    )
)

fig_03.update_layout(
    title_text='Общая воронка событий'
)
fig_03.update_layout(title_x=0.5)

fig_03.show()
```



Да, как мы и предполагали, воронка выглядит непривычно, но зато отражает логику дейсвтвий.
В абсолютном значении большинство пользователей теряется при переходе с первого шага на второй. В относительном значении больше всего теряется пользователей между просмотром продукта и переходом в корзину.
Общая конвесия из логирования в заказ составляет 31%, что весьма неплохо. Залогинились лишь 44% от всех пользователей (помним о том, что есть пользователи, которые не совершили дейсвтий в принципе)

Теперь посмотрим на воронку событий по группам

In [54]:
```
# Создадим сводник
funnel_exp = (
    df.pivot_table(index='group', columns='event_name' ,values='user_id', aggfunc='nu
)

# Нужный порядок колонок
funnel_exp = funnel_exp[
    ['login', 'product_page', 'product_cart', 'purchase']]

# Перевернем для удобства
funnel_exp = funnel_exp.transpose()

# Избавимся от мультииндекса
funnel_exp.reset_index(inplace=True)


funnel_exp
```

Out[54]:

group	event_name	A	B
0	login	2604	877
1	product_page	1685	493
2	product_cart	782	244
3	purchase	833	249

In [55]:

```
fig_04 = go.Figure()
for i in ['A', 'B']:
    fig_04.add_trace(go.Funnel(
        name = str(i),
        y = funnel_exp['event_name'],
        x = funnel_exp[i],
        textposition = "inside",
        textinfo = "value+percent initial+percent previous")
                )
    fig_04.update_layout(
    title_text='Воронка событий для исследуемых групп')
    fig_04.update_layout(title_x=0.5)
fig_04.show()
```

И тут группа А показывает более лучшие результаты. Общая конверсия из логирования в заказ 32% против 28%. У обеих групп больше всего пользователей отваливается при переходе с первого шага на второй. Однако у группы В созраняется больше пользователей при переходе со страницы продукта в корзину (49% против 46%)

Вывод по исследовательскому анализу данных:

- Всего в тесте учатсвует 6351 пользователь.
- Из них 2870 пользователей не совершили ни одного дейсвтия
- Более активные пользователи группы А с устройствами на мак
- В среднем пользователи совершают 796 дейсвтий в день для группы А и 212 дейсвтий в день для группы В
- Самый активый день - 21 декабря
- В среднем каждый пользователь совершает 7 дейсвтий в день для группы А и 5 дейсвтий в день для группы В
- Больше всего пользователей теряется при переходе с первого на второй шаг ворокни

Какие особенности данных нужно учесть перед тестированием?

дата набора пользоваталей, дата остановки, процент от европы, аактивность групп и тд пользователи без дейсвтий

В первую очередь необходимо обратить внимание на:

- Лишь 44% пользователей совершали какие-либо дейсвтия.
- Датой остановки теста считаем 31.12, а не 4.01
- Проведение теста совпало с маркетинговым событием
- Пользователи в группе А показали более высокую активность
- Тк одно из условий - изменение метрик в первые 14 дней после регистрации, значительная часть пользователей не будет учитываться в итоговой оценке теста

Оценка результатов тестирования

Сначала уберем действия пользователей, которые они совершали после 14го дня с момента регистрации

In [62]:
```
df = df[df['event_date'] <= df['last_date_fot_test']]
df.head()
```

Out[62]:

	user_id	event_dt	event_name	details	event_date	group	ab_test
0	D1ABA3E2887B6A73	2020-12-07 14:43:27	purchase	99.99	2020-12-07	A	recommender_system_test
2	D1ABA3E2887B6A73	2020-12-07 14:43:29	product_cart	no info	2020-12-07	A	recommender_system_test
4	D1ABA3E2887B6A73	2020-12-07 14:43:27	product_page	no info	2020-12-07	A	recommender_system_test
6	D1ABA3E2887B6A73	2020-12-07 14:43:27	login	no info	2020-12-07	A	recommender_system_test
8	A7A3664BD6242119	2020-12-20 15:46:06	product_page	no info	2020-12-20	A	recommender_system_test

Соберем сводники по группам, в которых укажем кол-во пользователей на каждом шаге, количество всех активынх пользователей и конверсию от всех активынх пользователей для каждого шага

In [63]:
```
# сводник для гурппы A
test_a = (
    df.query('group == "A"').groupby('event_name', as_index=False)['user_id'].nuniqu
     .rename(columns={'user_id': 'users_count'})
)
test_a['event_number'] = (
    test_a['event_name'].replace({'login':0, 'product_page':1, 'product_cart':2, 'pu
)

test_a = test_a.sort_values('event_number')
# Общее кол-во пользователей = кол-во активных пользователей
test_a['total_users'] = df[(df['group'] == 'A') & (~df['event_name'].isna())]['user_
test_a['ratio'] = round(((test_a['users_count'] / test_a['total_users'])*100),2)
test_a = test_a.drop('event_number', axis=1)
test_a = test_a.fillna(100)

test_a
```

Out[63]:

	event_name	users_count	total_users	ratio
0	login	2604	2604	100.00
2	product_page	1685	2604	64.71
1	product_cart	782	2604	30.03
3	purchase	833	2604	31.99

In [64]:
```
# сводник для группы B
test_b = (
    df.query('group == "B"').groupby('event_name', as_index=False)['user_id'].nuniqu
     .rename(columns={'user_id': 'users_count'})
)
test_b['event_number'] = (
    test_b['event_name'].replace({'login':0, 'product_page':1, 'product_cart':2, 'pu
)

test_b = test_b.sort_values('event_number')
# Общее кол-во пользователей = кол-во активных пользователей
test_b['total_users'] = df[(df['group'] == 'B') & (~df['event_name'].isna())]['user_
test_b['ratio'] = round(((test_b['users_count'] / test_b['total_users'])*100),2)
test_b = test_b.drop('event_number', axis=1)
test_b = test_b.fillna(100)

test_b
```

Out[64]:

	event_name	users_count	total_users	ratio
0	login	876	877	99.89
2	product_page	493	877	56.21
1	product_cart	244	877	27.82
3	purchase	249	877	28.39

In [65]:
```
test_AB = pd.merge(test_a, test_b, on='event_name' ,suffixes=('_A', '_B'))
# уберем строчку с login, тк тут она нам не нужна
test_AB = test_AB[test_AB['event_name'] != 'login']
test_AB
```

Out[65]:

	event_name	users_count_A	total_users_A	ratio_A	users_count_B	total_users_B	ratio_B
1	product_page	1685	2604	64.71	493	877	56.21
2	product_cart	782	2604	30.03	244	877	27.82
3	purchase	833	2604	31.99	249	877	28.39

In [66]:
```
# Создадим функцию, которая будет брать на вход строку и сравнивать четыре значения.
alpha = 0.05 # критический уровень статистической значимости

# в функцию будем передавать кол-во пользователей в каждой группе совершивших опреде
# и общее количество пользователей в группе
def p_value(row):
    successes = np.array([row['users_count_A'], row['users_count_B']]) # кол-во уник
    trials = np.array([row['total_users_A'], row['total_users_B']]) # общее кол-во у
    stat, pval = proportions_ztest(successes, trials)
    return pval

# в функцию будем передавать полученное p-value и сравнивать его с уровнем критическ
# функция будет возврващать результат
def result(row, coefficient):
    pval = row['p-value'] # столбец с p-значением из z-теста
    if (pval < coefficient):
        return("Отвергаем нулевую гипотезу: между долями есть значимая разница")
    else:
        return("Не получилось отвергнуть нулевую гипотезу, нет оснований считать дол
```

Сформируем гипотезы:

- **H0** - между выборками нет значимой разницы в конверсии
- **H1** - между выборками есть значимая разница

In [67]:
```
# Добавим результат работы функции в отдельный столбец.
test_AB['p-value'] = test_AB.apply(p_value, axis=1)
test_AB['result'] = test_AB.apply(result, coefficient=alpha, axis=1)
test_AB
```

Out[67]:

	event_name	users_count_A	total_users_A	ratio_A	users_count_B	total_users_B	ratio_B	p-v
1	product_page	1685	2604	64.71	493	877	56.21	0.000
2	product_cart	782	2604	30.03	244	877	27.82	0.214
3	purchase	833	2604	31.99	249	877	28.39	0.046

Между выборками A и B статистическая значимость различий достигнута только для событий "product_page" и "purchase". Для конверсии в корзину статистической разницы нет.

Оценка с учетом поправки

Можно применить поправку Бонферрони, Идея метода заключается в том, чтобы снизить вероятность ошибки первого рода.

Чтобы применить поправку Бонферрони нужно допустимый уровень значимости разделить на кол-во сравнений.

В нашем случае, допустимый уровень значимости будет равен 0.016

In [68]:
```
# Применим поправку Бонферрони
alpha = 0.05/3
test_AB['result_repair'] = test_AB.apply(result, coefficient=alpha, axis=1)
test_AB
```

Out[68]:

	event_name	users_count_A	total_users_A	ratio_A	users_count_B	total_users_B	ratio_B	p-v
1	product_page	1685	2604	64.71	493	877	56.21	0.000
2	product_cart	782	2604	30.03	244	877	27.82	0.214
3	purchase	833	2604	31.99	249	877	28.39	0.046

С учетом поправки Бонферрони лишь в одном случае мы смогли отвергнуть нулевую гипотезу

Выводы

Была проделана исследовательская работа со следующими этапами:

- Ознакомление с данными
- Оценка корректности теста и сверка с ТЗ
- Исследовательский анализ данных
- Оценка результатов тестирования

Теперь можно сделать основные выводы по работе:

- Всего в тесте учатсвует 6351 пользователь.
- Из них 2870 пользователей не совершили ни одного дейсвтия
- Более активные пользователи группы А с устройствами на мак
- В среднем пользователи совершают 796 дейсвтий в день для группы А и 212 дейсвтий в день для группы В
- Самый активый день - 21 декабря
- В среднем каждый пользователь совершает 7 дейсвтий в день для группы А и 5 дейсвтий в день для группы В
- Больше всего пользователей теряется при переходе с первого на второй шаг ворокни

Оценка результатов тестирования:

- Статистически значимые показатели были замечены лишь в конверсии в корзину, а не по всем исследуемым параметрам. Даже с учетом поправки Бонферрони мы не достигли изменений по всем показателям.
- Изменения, связанные с внедрением улучшенной рекомендательной системы не достигли желаемого результата

Тест остановить и признать провальным, тк:

- Есть несколько пунктов ТЗ, которые не были достигнуты:
  - Маркетинговые события
  - Кол-во пользователей для теста
- Во всех показателях конверсии, конверсии группы В ниже, чем результаты контрольной группы А (ожидаемый эффект не достигнут)
- Группы получились слишком разными по активности.