In [1]:
```
#загружаем необходимые библиотеки
import pandas as pd
import matplotlib.pyplot as plt
from plotly import graph_objects as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"
import numpy as np
import seaborn as sns
sns.set(style="whitegrid")
colors =["#ef476f","#ffd166","#06d6a0","#118ab2","#073b4c"]
sns.set_palette(sns.color_palette(colors))
import re
from scipy import stats as st
import math as mth
import re
from scipy import stats as st
import math as mth
import warnings
warnings.filterwarnings('ignore')
```

Цель исследования - провести оценку результатов A/B-теста.

- Оценить корректность проведения теста
- Проанализировать результаты теста
- Проверить:
  - пересечение тестовой аудитории с конкурирующим тестом,
  - совпадение теста и маркетинговых событий, другие проблемы временных границ теста
- Тестирование изменений, связанных с внедрением улучшенной рекомендательной системы.

Загружаем данные

Первый датафрейм

In [2]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df1 = pd.read_csv('/datasets/ab_project_marketing_events.csv')
df1.head()
```

Out[2]:

	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 [3]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   name       14 non-null     object
 1   regions    14 non-null     object
 2   start_dt   14 non-null     object
```

```
 3   finish_dt  14 non-null     object
dtypes: object(4)
memory usage: 576.0+ bytes
```

In [4]:
```
#Выведем размер
print(df1.shape)
```

```
(14, 4)
```

В фрейме 14 строки, 4 столбца.Тип данных - object.

- Имеем столбцы:
  - name - название маркетингового события
  - regions - регионы, в которых будет проводиться рекламная кампания
  - start_dt - дата начала кампании
  - finish_dt - дата завершения кампании

Второй датафрейм

In [5]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df2 = pd.read_csv('/datasets/final_ab_events.csv')
df2.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]:
```
df2.info()
```

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

In [7]:
```
#Выведем размер
print(df2.shape)
```

```
(440317, 4)
```

В фрейме 440317 строк, 4 столбца.Тип данных - object и float64.

- Имеем столбцы:
  - 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.

Третий датафрейм

In [8]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df3 = pd.read_csv('/datasets/final_ab_participants.csv')
df3.head()
```

Out[8]:

	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

In [9]:
```
df3['ab_test'].head()
```

Out[9]:
```
0    recommender_system_test
1    recommender_system_test
2    recommender_system_test
3    recommender_system_test
4    recommender_system_test
Name: ab_test, dtype: object
```

In [10]:
```
df3.info()
```

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

In [11]:
```
#Выведем размер
print(df3.shape)
```

```
(18268, 3)
```

В фрейме 18268 строк, 3 столбца.Тип данных - object .

- Имеем столбцы:
  - user_id - идентификатор пользователя
  - group - группа пользователя
  - ab_test - название теста

Четвертый датафрейм

In [12]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df4 = pd.read_csv('/datasets/final_ab_new_users.csv')
df4.head()
```

Out[12]:

	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

	user_id	first_date	region	device
4	E1BDDCE0DAFA2679	2020-12-07	N.America	iPhone

In [13]:
```
df4.info()
```

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

In [14]:
```
#Выведем размер
print(df4.shape)
```

```
(61733, 4)
```

В фрейме 61733 строк, 4 столбца.Тип данных - object

- Имеем столбцы:
  - user_id - идентификатор пользователя
  - first_date - дата регистрации
  - region - регион пользователя
  - device - устройство, с которого происходила регистрация

Предобработка и исследовательский анализ данных

Предобработка

Первый датасет:

In [15]:
```
# Названия столбцов:
df1.columns
```

Out[15]:
```
Index(['name', 'regions', 'start_dt', 'finish_dt'], dtype='object')
```

Названия корректные

In [16]:
```
# проверим на дубликаты
df1.duplicated(subset=['name', 'regions', 'start_dt', 'finish_dt']).sum()
```

Out[16]:
```
0
```

Дубликатов нет

In [17]:
```
# проверим на пропуски
df1.isna().sum()
```

Out[17]:
```
name         0
regions      0
start_dt     0
finish_dt    0
dtype: int64
```

Пропусков нет.

In [18]:
```
# Заменим типы данных
df1['start_dt'] = df1['start_dt'].astype('datetime64')
df1['finish_dt'] = df1['finish_dt'].astype('datetime64')
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   name       14 non-null     object
 1   regions    14 non-null     object
 2   start_dt   14 non-null     datetime64[ns]
 3   finish_dt  14 non-null     datetime64[ns]
dtypes: datetime64[ns](2), object(2)
memory usage: 576.0+ bytes
```

In [19]:
```
df1.sample(3)
```

Out[19]:

	name	regions	start_dt	finish_dt
6	Chinese New Year Promo	APAC	2020-01-25	2020-02-07
8	International Women's Day Promo	EU, CIS, APAC	2020-03-08	2020-03-10
10	CIS New Year Gift Lottery	CIS	2020-12-30	2021-01-07

Второй датасет:

In [20]:
```
# Названия столбцов:
df2.head()
```

Out[20]:

	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 [21]:
```
df2['event_name'].value_counts()
```

Out[21]:
```
login           189552
product_page    125563
purchase         62740
product_cart     62462
Name: event_name, dtype: int64
```

Названия корректные

In [22]:
```
# проверим на дубликаты
df2.duplicated(subset=['user_id', 'event_dt', 'event_name', 'details']).sum()
```

Out[22]:
```
0
```

Дубликатов нет

In [23]:
```
df2['event_name'].value_counts()
```

Out[23]:
```
login           189552
```

```
product_page    125563
purchase         62740
product_cart     62462
Name: event_name, dtype: int64
```

In [24]:
```
# проверим на пропуски
df2.isna().sum()
```

Out[24]:
```
user_id            0
event_dt           0
event_name         0
details       377577
dtype: int64
```

Пропуски есть в столбце details - 377577. details — дополнительные данные о событии. Например, для покупок, purchase, в этом поле хранится стоимость покупки в долларах.Можно предположить, что не все покупки совершены.

In [25]:
```
# Заменим типы данных
df2['event_dt'] = df2['event_dt'].astype('datetime64')
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440317 entries, 0 to 440316
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   user_id     440317 non-null  object        
 1   event_dt    440317 non-null  datetime64[ns]
 2   event_name  440317 non-null  object        
 3   details     62740 non-null   float64       
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 13.4+ MB
```

In [26]:
```
# найдем,сколько всего уникальных событий в логе
df2['event_name'].unique()
```

Out[26]:
```
array(['purchase', 'product_cart', 'product_page', 'login'], dtype=object)
```

In [27]:
```
df2['event_name'].value_counts()
```

Out[27]:
```
login           189552
product_page    125563
purchase         62740
product_cart     62462
Name: event_name, dtype: int64
```

Имеем:

- purchase - покупки
- product_cart - просмотры корзины
- product_page - просмотр карточек товаров
- login - вход

In [28]:
```
df2['user_id'].nunique()
```

Out[28]:
```
58703
```

In [29]:
```
df2.sample(3)
```

Out[29]:

	user_id	event_dt	event_name	details
189187	591B94FC6F751001	2020-12-19 07:53:40	product_page	NaN

	user_id	event_dt	event_name	details
273684	A1EE8A62DA8344B0	2020-12-11 10:13:59	login	NaN
406081	DEBE8475879660F1	2020-12-24 13:38:48	login	NaN

Третий датасет:

In [30]:
```
# Названия столбцов:
df3.columns
```

Out[30]:
```
Index(['user_id', 'group', 'ab_test'], dtype='object')
```

Названия корректные

In [31]:
```
# проверим на дубликаты
df3.duplicated(subset=['user_id', 'group', 'ab_test']).sum()
```

Out[31]:
```
0
```

Дубликатов нет

In [32]:
```
# проверим на пропуски
df3.isna().sum()
```

Out[32]:
```
user_id    0
group      0
ab_test    0
dtype: int64
```

Пропусков нет.

In [33]:
```
df3['user_id'].nunique()
```

Out[33]:
```
16666
```

In [34]:
```
df3.sample(3)
```

Out[34]:

	user_id	group	ab_test
8619	2030EBE446D2CF39	A	interface_eu_test
134	3B9B00EA8303F6DA	A	recommender_system_test
945	6647A4395CED21C4	A	recommender_system_test

Четвертый датасет:

In [35]:
```
# Названия столбцов:
df4.columns
```

Out[35]:
```
Index(['user_id', 'first_date', 'region', 'device'], dtype='object')
```

Названия корректные

In [36]:
```
# проверим на дубликаты
df4.duplicated(subset=['user_id', 'first_date', 'region', 'device']).sum()
```

Out[36]:
```
0
```

Дубликатов нет

In [37]:
```
df4['user_id'].nunique()
```

Out[37]:
```
61733
```

In [38]:
```
# проверим на пропуски
df4.isna().sum()
```

Out[38]:
```
user_id       0
first_date    0
region        0
device        0
dtype: int64
```

Пропусков нет.

In [39]:
```
# Заменим типы данных
df4['first_date'] = df4['first_date'].astype('datetime64')
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61733 entries, 0 to 61732
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   user_id     61733 non-null  object        
 1   first_date  61733 non-null  datetime64[ns]
 2   region      61733 non-null  object        
 3   device      61733 non-null  object        
dtypes: datetime64[ns](1), object(3)
memory usage: 1.9+ MB
```

In [40]:
```
df4.sample(3)
```

Out[40]:

	user_id	first_date	region	device
13999	556326B7FE314166	2020-12-21	N.America	PC
41107	16C8661E42FCE4AD	2020-12-11	EU	iPhone
53563	7B5C1EDF3EC1F686	2020-12-13	EU	Android

Вывод.

- В процессе предобработки:
- проверили на дубли и отсутствующие значения
- изменили тип данных в формат datetime.
- Тип данных в каждой колонке – правильный Дубликатов нет.

Изучим и проверим данные

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 — `purchas`

In [41]:
```
# Распределение пользователей по регионам
reg_name = df4.groupby('region')['user_id'].nunique().sort_values()
reg_name
```

Out[41]:
```
region
APAC          3153
CIS           3155
N.America     9155
EU           46270
Name: user_id, dtype: int64
```

In [42]:
```
# дата остановки набора новых пользователей:
df4['first_date'].max()
```

Out[42]:
```
Timestamp('2020-12-23 00:00:00')
```

- дата остановки набора новых пользователей: 2020-12-23 не совпадает с техзаданием

Рассмотрим совершенные события

In [43]:
```
df3.head()
```

Out[43]:

	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

In [44]:
```
df3['user_id'].nunique()
```

Out[44]:
```
16666
```

In [45]:
```
df3['group'].value_counts()
```

Out[45]:
```
A    9655
B    8613
Name: group, dtype: int64
```

In [46]:
```
df3['ab_test'].value_counts()
```

Out[46]:
```
interface_eu_test          11567
recommender_system_test     6701
Name: ab_test, dtype: int64
```

- Итак, у нас есть
  - 16 666 уникальных пользователей
  - разбивка на две группы A и B
  - два A/B-теста interface_eu_test и recommender_system_test
- Так как по условиям задачи нам интересен только тест recommender_system_test, то чтобы не делать лишних манипуляций с данными, отберем только целевых пользователей (участников теста). Всю дальнейшую работу будем строить вокруг этих пользователей.

In [47]:
```
# Начнем собирать в новый DataFrame всю необходимую информацию
df = df3[df3['ab_test']=='recommender_system_test']
```

Если тесты проводились параллельно, а не последовательно, то пользователи могли оказаться в другом эксперименте, что привело бы к искажению результатов. Нужно проверить, попадают ли наши пользователи во второй эксперимент? И если попадает, то можем ли мы с этим что-то сделать? Начнем с проверки пересечения сроков проведения тестов.

In [48]:
```
check = df4.merge(df3, on='user_id')
check.head()
```

Out[48]:

	user_id	first_date	region	device	group	ab_test
0	D72A72121175D8BE	2020-12-07	EU	PC	A	recommender_system_test
1	2E1BF1D4C37EA01F	2020-12-07	EU	PC	A	interface_eu_test
2	50734A22C0C63768	2020-12-07	EU	iPhone	B	interface_eu_test
3	E6DE857AFBDC6102	2020-12-07	EU	PC	B	recommender_system_test
4	E6DE857AFBDC6102	2020-12-07	EU	PC	B	interface_eu_test

In [49]:
```
check.pivot_table(index='first_date', columns='ab_test', aggfunc={'user_id':'nunique'})
```

Out[49]:

	user_id	
ab_test	**interface_eu_test**	**recommender_system_test**
first_date		
2020-12-07	1031.0	645.0
2020-12-08	612.0	364.0
2020-12-09	426.0	258.0
2020-12-10	551.0	391.0
2020-12-11	450.0	277.0
2020-12-12	800.0	489.0
2020-12-13	874.0	581.0
2020-12-14	1045.0	665.0
2020-12-15	589.0	359.0
2020-12-16	387.0	285.0
2020-12-17	581.0	359.0
2020-12-18	627.0	397.0
2020-12-19	618.0	423.0

	user_id	
ab_test	interface_eu_test	recommender_system_test
first_date		
2020-12-20	794.0	485.0
2020-12-21	1180.0	723.0
2020-12-22	587.0	NaN
2020-12-23	415.0	NaN

Опасения подтвердились, тесты шли параллельно. Наш тест (recommender_system_test) закончился на два дня раньше, поэтому, чтобы уменьшить объем анализируемой информации оставлю данные только по 2020-12-21

In [50]:
```
check = check[check['first_date']<'2020-12-22']
```

- Проверим дату регистрации пользователей (first_date), возможно, что по этой метрике удастся восстановить к какому тесту относить пользователя. Логика такая:
  - если пользователь только в одном тесте, то у него будет только одна дата
  - если пользователь в двух тестах, то у него будет две даты (для каждого теста) и тогда можно будет понять в какой тест пользователь был записан раньше.

In [51]:
```
check_date = check.pivot_table(index='user_id', columns='ab_test', aggfunc={'first_date':'min'}
check_date.columns=['interface', 'recommender']

# отбираем только тех пользователей, которые оказались в двух тестах
check_date = check_date[(check_date['interface'].isnull()==False)&(check_date['recommender'].is
check_date.reset_index(inplace=True)
check_date
```

Out[51]:

	user_id	interface	recommender
0	001064FEAAB631A1	2020-12-20	2020-12-20
1	00341D8401F0F665	2020-12-21	2020-12-21
2	003B6786B4FF5B03	2020-12-13	2020-12-13
3	0082295A41A867B5	2020-12-16	2020-12-16
4	00E68F103C66C1F7	2020-12-18	2020-12-18
...	...	...	...
1597	FF7BE2897FC0380D	2020-12-13	2020-12-13
1598	FF9A81323FA67D6E	2020-12-09	2020-12-09
1599	FFC53FD45DDA5EE8	2020-12-19	2020-12-19
1600	FFED90241D04503F	2020-12-08	2020-12-08
1601	FFF28D02B1EACBE1	2020-12-16	2020-12-16

1602 rows × 3 columns

1602 участника относятся сразу к двум тестам. Для нашего теста (recommender_system_test) это очень существенная доля. Посмотрим, что можно с этим сделать.

In [52]:
```
def find_first_date(row):
```

```
    '''
    Функция сравнивает даты и возвращает 0 в случае их равенства и 1 в случае неравенства
    '''
    interface=row['interface']
    recommender=row['recommender']

    if interface == recommender:
        return 0
    else:
        return 1
```

In [53]:
```
check_date['find_first_date'] = check_date.apply(find_first_date, axis=1)
```

In [54]:
```
check_date[check_date['find_first_date']==1]
```

Out[54]:

user_id	interface	recommender	find_first_date

Результат такой проверки показал, что у всех пользователей которые относятся сразу к двум тестам одинаковая дата регистрации. Прежде чем перейти к выводам по этому факту хочется посмотреть на динамику появления таких пользователей.

In [55]:
```
check_date_day = check_date.groupby('interface').agg({'user_id':'count'})

fig = px.bar(check_date_day, x=check_date_day.index, y=check_date_day['user_id'])
fig.update_xaxes(tickangle=-90, tickmode = 'linear')

fig.update_layout(title='Распределение количества пользователей по датам',
                  yaxis_title='Количество пользователей',
                  margin=dict(l=0, r=0))

fig.show()
```

- Выводы данного расследования:
  - на протяжении всего рассматриваемого периода происходит постоянное «дублирование» пользователей в два параллельно идущих A/B-теста. Либо это какая-то техническая ошибка в сборе данных, либо что-то не так с предоставленной выгрузкой данных (может в сырых логах все корректно?)
  - не нашли никакой возможности однозначно идентифицировать данных пользователей по тестам которые проходили параллельно.
- Проведя дополнительные проверки (в черновике) выяснили, что удаление данных пользователей:
  - уменьшает выборки групп A и B в целевом тесте, но пропорционально это деление не меняется.
  - не меняет оценку результатов A/B-теста. Уменьшение количества пользователей в группах не привело к заметному изменению конверсий на всех этапах воронки и к изменению результатов Z-теста.
- Поэтому решили не удалять данных пользователей, потому что невозможно на 100% установить их причастность или непричастность к целевому тесту и их удаление не приносит изменения в итоговый результатах.

In [56]:
```
df4.head()
```

Out[56]:

	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

объединим данные по user_id участников теста.

In [57]:
```
df = df.merge(df4, on='user_id')
```

In [58]:
```
df = df.merge(df2, on='user_id', how='left')
df.head()
```

Out[58]:

	user_id	group	ab_test	first_date	region	device	event_dt	event_name	detai
0	D1ABA3E2887B6A73	A	recommender_system_test	2020-12-07	EU	PC	2020-12-07 14:43:27	purchase	99.9
1	D1ABA3E2887B6A73	A	recommender_system_test	2020-12-07	EU	PC	2020-12-25 00:04:56	purchase	4.9
2	D1ABA3E2887B6A73	A	recommender_system_test	2020-12-07	EU	PC	2020-12-07 14:43:29	product_cart	Na
3	D1ABA3E2887B6A73	A	recommender_system_test	2020-12-07	EU	PC	2020-12-25 00:04:57	product_cart	Na

	user_id	group	ab_test	first_date	region	device	event_dt	event_name	detai
4	D1ABA3E2887B6A73	A	recommender_system_test	2020-12-07	EU	PC	2020-12-07 14:43:27	product_page	Na

Распределение пользователей по группам

Для более полного понимания ситуации посмотрим на ежедневную динамику наполнения групп (A и B) пользователями.

In [59]:
```
user_by_group = df.pivot_table(index='first_date', columns='group', aggfunc={'user_id':'nunique
user_by_group.columns = ['A', 'B']
```

In [60]:
```
fig = go.Figure()

for col in user_by_group.columns:
    fig.add_trace(go.Scatter(x=user_by_group.index, y=user_by_group[col],
                             name=col, mode='lines+markers'))

fig.update_layout(title='Динамика распределения пользователей по группам',
                  yaxis_title='Количество пользователей в день')
fig.update_xaxes(tickangle=-90, tickmode = 'linear')

fig.show()
```

- Данный график позволяет вывести два наблюдения:
  - очень четко прослеживается сезонность выходных дней и понедельника.

- все понедельники - это максимальные значения в группе A и группе B
- равенства групп (по количеству пользователей) нет. Но чтобы окончательно в этом убедиться сделаем дополнительную проверку.

In [61]:
```
count_a = df[df['group']=='A']['user_id'].nunique()
count_b =  df[df['group']=='B']['user_id'].nunique()
print('Количество пользователей в группе A: {}'.format(count_a),
      '\nКоличество пользователей в группе B: {}'.format(count_b),
      '\nПроцент прироста группы B к группе A: {:.2%}'.format((count_b-count_a)/count_a))
```

```
Количество пользователей в группе A: 3824
Количество пользователей в группе B: 2877
Процент прироста группы B к группе A: -24.76%
```

Количество пользователей в группах совсем не одинаковое.

Проверим, что есть пользователи из обеих экспериментальных групп.

In [62]:
```
# проверим пользователей на наличие их в нескольких тестах и группах одновременно
intersection_test = df.groupby('user_id').agg({'ab_test':'nunique'}).query('ab_test > 1').count
print('Пересечений в тестах:' , intersection_test)
```

```
Пересечений в тестах: ab_test    0
dtype: int64
```

In [63]:
```
# нас интересует тест "recommender_system_test"
intersection_test = df.query('ab_test == "recommender_system_test"').groupby('user_id').agg({'a
print('Пересечений в группах теста:' , intersection_test)
```

```
Пересечений в группах теста: ab_test    0
dtype: int64
```

Пересечений в группах теста "recommender_system_test" нет

Всего пользователей в тесте "recommender_system_test"

In [64]:
```
df3_test = df.query('ab_test == "recommender_system_test"')
users_test = df3_test.groupby('user_id').agg({'user_id':'nunique'}).count()
users_test
```

Out[64]:
```
user_id    6701
dtype: int64
```

В тесте участвуют 6701 пользователь

In [65]:
```
# найдем,сколько всего событий в логе
event =  df3_test['event_name'].count()
event
```

Out[65]:
```
24698
```

In [66]:
```
# найдем,сколько в среднем событий приходится на пользователя
int(df3_test.groupby('user_id')['event_name'].agg('count').median())
```

Out[66]:
```
2
```

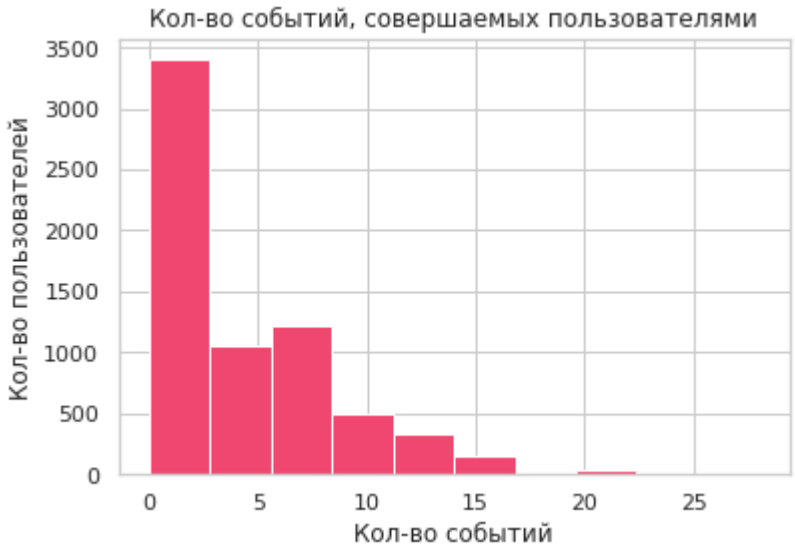
In [67]:
```
# выведем статистические данные через метод describe()
df3_test.groupby('user_id')['event_name'].agg('count').describe()
```

Out[67]:
```
count    6701.000000
mean        3.685719
std         4.376568
min         0.000000
```

```
25%          0.000000
50%          2.000000
75%          6.000000
max         28.000000
Name: event_name, dtype: float64
```

В среднем на пользователя приходится 2 события. Max 28 событий

In [68]:
```
# построим гистограмму
df3_test.groupby('user_id').agg({'event_name':'count'}).hist(bins=10)
df3_test.groupby('user_id').agg({'event_name':'count'}).median()
plt.title('Кол-во событий, совершаемых пользователями')
plt.xlabel('Кол-во событий')
plt.ylabel('Кол-во пользователей')
plt.show()
```



In [69]:
```
# минимальная дата привлечения пользователей.
df3_test['event_dt'].min()
```

Out[69]:
```
Timestamp('2020-12-07 00:05:57')
```

In [70]:
```
# максимальная дата привлечения пользователей.
df3_test['event_dt'].max()
```

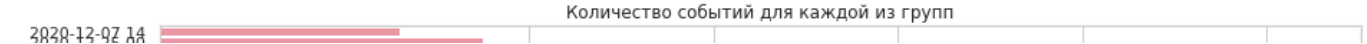
Out[70]:
```
Timestamp('2020-12-30 12:42:57')
```

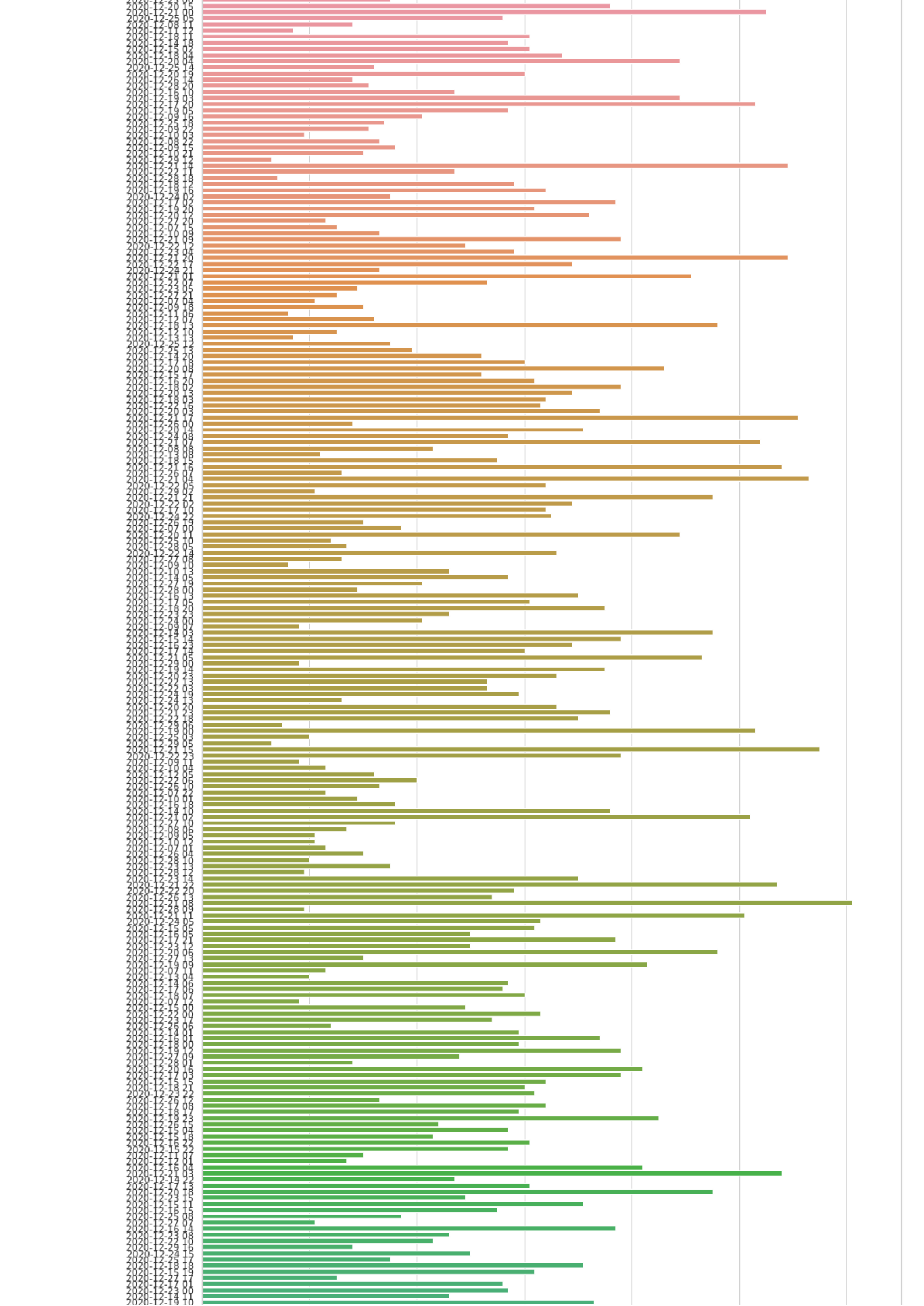
In [71]:
```
# период
df3_test['event_dt'].max() - df3_test['event_dt'].min()
```

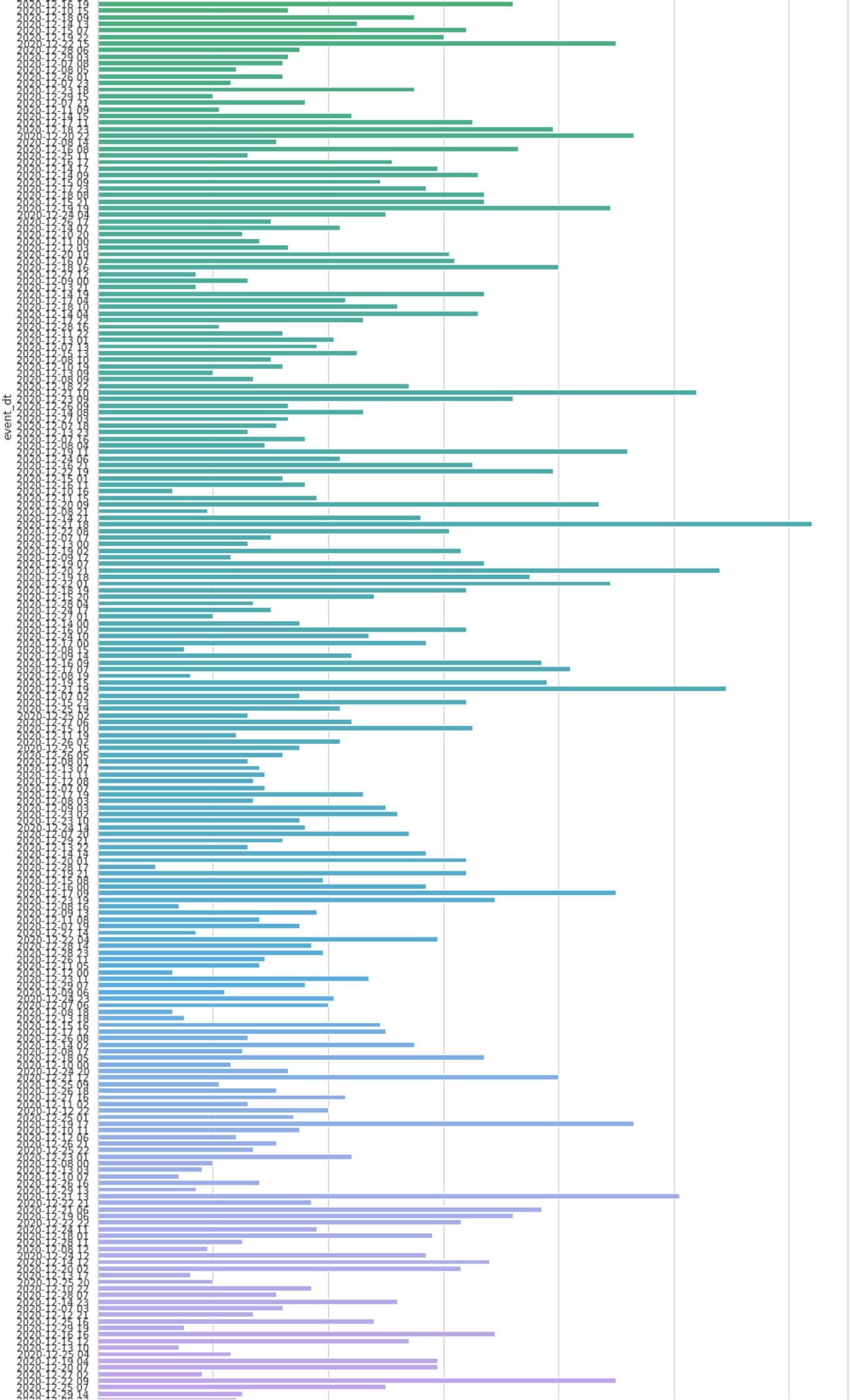
Out[71]:
```
Timedelta('23 days 12:37:00')
```

In [72]:
```
print('Всего событий в логе за актуальный период {}.'.format(df3_test['user_id'].count()))
print('')
print('Всего пользователей в логе за актуальный период {}.'.format(df3_test['user_id'].nunique(
print('')
plt.figure(figsize=(15, 75))
ax = sns.countplot(y=df3_test['event_dt'].dt.strftime('%Y-%m-%d %H'),  data=df3_test)
ax.set_title('Количество событий для каждой из групп')
plt.show()
```

```
Всего событий в логе за актуальный период 27724.

Всего пользователей в логе за актуальный период 6701.
```






event_dt

По графику видим , что тест длился не до конца планируемого срока .

Проверка конверсий пользовательской воронки

In [73]:

```
funnel = df.pivot_table(index='event_name', columns='group',aggfunc={'user_id':'nunique'})
funnel.columns = ['A','B']
funnel = funnel.sort_values(by='A', ascending=False)
```

In [74]:

```
def my_funnel(df, title):
    """
    Функция принимает df и название графика. Воронка строится по событиям, которые должны являт
    Если указанный df содержит несколько переменных, то воронка будет построена по всем данным
    """

    fig = go.Figure()

    for name in df.columns:
        fig.add_trace(go.Funnel(
            name = name,
            y = df.index,
            x = df[name],
            textposition = 'inside',
            textinfo = 'value+percent initial+percent previous'))
```

```
    fig.update_layout(title=title,
                      margin=dict(l=0, r=0))

    fig.show()
```

In [75]:
```
my_funnel(funnel, 'Пользовательская воронка, конверсии')
```

Несмотря на большую разницу в количестве пользователей, конверсия по этапа не сильно отличается. Из общей картины выпадает только второй этап воронки (product_page). Разница между группами A и B почти 10%. Но вот остальные этапы не имеют такого сильного отличия.

Найдем события, которые не выходят за рамки двухнедельного окна

In [76]:
```
# находим время первого дейсвия каждого посетителя
user_event_time_min = df3_test.groupby('user_id', as_index=False).agg({'event_dt':'min'})
user_event_time_min.columns = ['user_id', 'user_first_event_date']
user_event_time_min.head()
```

Out[76]:

	user_id	user_first_event_date
0	000ABE35EE11412F	NaT
1	001064FEAAB631A1	2020-12-20 14:43:27
2	0010A1C096941592	2020-12-17 21:07:27
3	001C05E87D336C59	NaT
4	00341D8401F0F665	2020-12-21 11:14:50

In [77]:
```
# добавляем эти данные в общий фрейм
```

```
df3_new = df3_test.merge(user_event_time_min, on='user_id', how='left')
```

In [78]:
```
#разница
df3_new['diff'] = df3_new['event_dt'] - df3_new['user_first_event_date']
df3_new['diff'] = df3_new['diff'].dt.days
print('Всего событий:', df3_new['event_dt'].count())
diff = df3_new.query('diff <=14')
print('количество событий с длиной действий  14 дней:', diff['diff'].count())
```

```
Всего событий: 24698
количество событий с длиной действий  14 дней: 24155
```

In [79]:
```
diff.groupby('user_id').agg({'user_id':'nunique'}).count()
```

Out[79]:
```
user_id    3675
dtype: int64
```

В тесте остается 24155события 3675 пользователя

В дальнейшем будем работать с этим фреймом:

- только в 14-дневный период

Всего пользователей в тесте

In [80]:
```
users_test = diff.groupby('user_id').agg({'user_id':'nunique'}).count()
users_test
```

Out[80]:
```
user_id    3675
dtype: int64
```

Всего пользователей участвуют в тесте 3675

Количество пользователей в группах

In [81]:
```
diff.groupby('group').agg({'user_id': 'nunique'}).reset_index()
```

Out[81]:

	group	user_id
0	A	2747
1	B	928

- Количество пользователей в группах:
  - A - 2747
  - B - 928

По заданию - аудитория: 15% новых пользователей из региона EU. Найдем новых пользователей из Европы

In [82]:
```
df4['region'].value_counts()
```

Out[82]:
```
EU           46270
N.America     9155
CIS           3155
APAC          3153
Name: region, dtype: int64
```

In [83]:
```
# посчитаем сколько всего из региона EU
user_eu =diff.query('region == "EU"'). groupby('user_id').agg({'user_id':'nunique'}).count()
user_eu
```

Out[83]:
```
user_id    3481
dtype: int64
```

In [84]:
```
user_eu_tot = diff.query('region == "EU"')['user_id'].count()
user_eu_tot
```

Out[84]:
```
22910
```

In [85]:
```
# найдем долю тех, кто в тесте
print(f'Пользователей из EU региона - {round(user_eu/user_eu_tot*100,2)}%')
```

```
Пользователей из EU региона - user_id    15.19
dtype: float64%
```

Доля новых пользователей из Европы 15.2%. Совпадает с ТЗ

Даты маркетинговых событий

In [86]:
```
df1.sort_values(by='start_dt')
```

Out[86]:

	name	regions	start_dt	finish_dt
6	Chinese New Year Promo	APAC	2020-01-25	2020-02-07
1	St. Valentine's Day Giveaway	EU, CIS, APAC, N.America	2020-02-14	2020-02-16
8	International Women's Day Promo	EU, CIS, APAC	2020-03-08	2020-03-10
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
7	Labor day (May 1st) Ads Campaign	EU, CIS, APAC	2020-05-01	2020-05-03
9	Victory Day CIS (May 9th) Event	CIS	2020-05-09	2020-05-11
11	Dragon Boat Festival Giveaway	APAC	2020-06-25	2020-07-01
4	4th of July Promo	N.America	2020-07-04	2020-07-11
13	Chinese Moon Festival	APAC	2020-10-01	2020-10-07
12	Single's Day Gift Promo	APAC	2020-11-11	2020-11-12
5	Black Friday Ads Campaign	EU, CIS, APAC, N.America	2020-11-26	2020-12-01
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

Время проведения теста совпадает с маркетинговыми и другими активностями - "Christmas&New Year Promo"с 2020-12-25 по 2021-01-03

Оценим корректность проведения теста.

Т.к. не мы планировали тест и не следили за его процессом, нужно проверить адекватность его проведения. Первым делом проверим условия ТЗ (с исходными данными), отметим где ошибки в сборе и потом уже отфильтруем для чистоты анализа с обоснованием.

Проверим корректность всех пунктов технического задания.

- В логе 3675 пользователя. Не совпадает с ТЗ (6000)
- 24155 событие
- Дата запуска: 2020-12-07 совпадает
- Дата остановки: 2021-01-04 не совпадает. Фактически - 2020-12-30
- Дата дата остановки набора новых пользователей: 2020-12-23 не совпадает с техзаданием
- Новых пользователей из Европы 15.2%
- Время проведения теста совпадает с маркетинговыми активностями "Christmas&New Year Promo "
- В среднем на пользователя приходится 2 события

- Количество пользователей в группах равно
  - в группе A - 2747
  - в группе B - 928

Прежде чем приступать к A/B-тестированию нужно учесть,что данные для теста не совпадают с ТЗ

Изучим воронку событий.

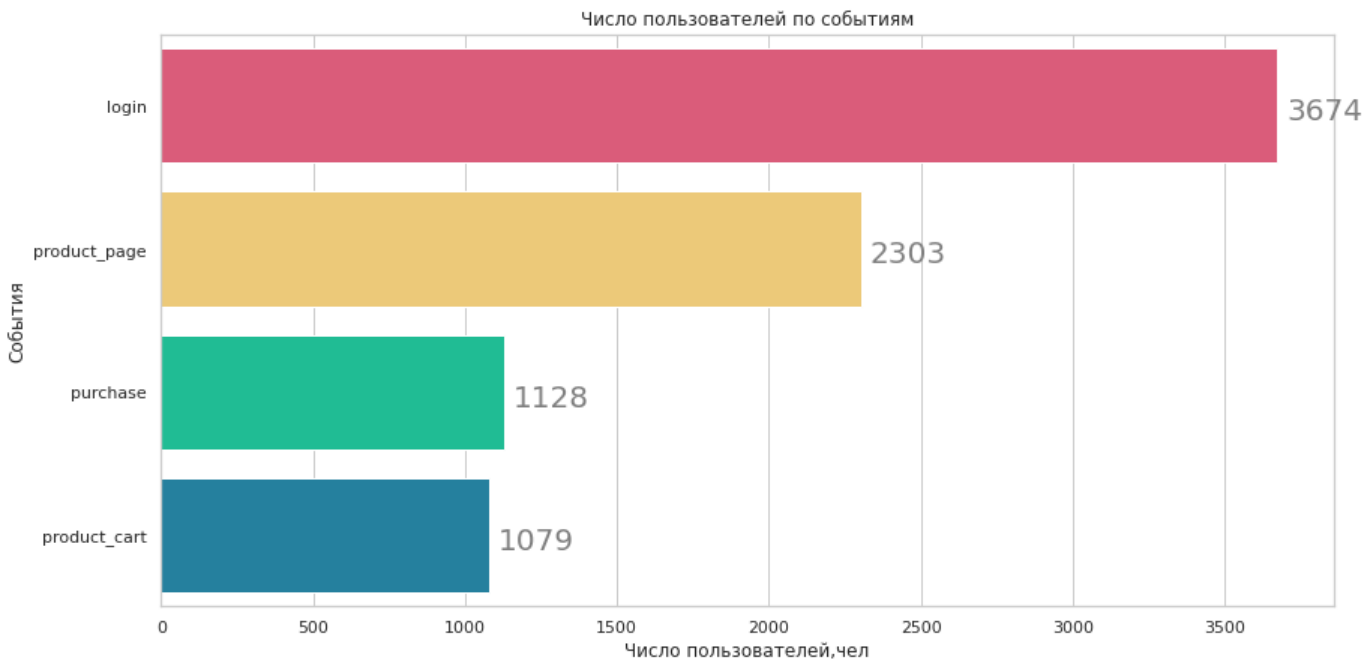
Посмотрим, какие события есть в логах, как часто они встречаются. Отсортируем события по частоте.

In [87]:
```
events = diff.groupby('event_name').agg({'user_id': 'nunique'}).reset_index()
events.columns =['event_name', 'n_users']
events
```

Out[87]:

	event_name	n_users
0	login	3674
1	product_cart	1079
2	product_page	2303
3	purchase	1128

In [88]:
```
n_users = {'all': diff['user_id'].nunique()
           }
plt.figure(figsize=(14, 7))
order = events.sort_values('n_users', ascending=False).reset_index(drop=True)['event_name']
ax = sns.barplot(y='event_name', x='n_users', order = order,  data=events)
ax.set_title('Число пользователей по событиям')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События')
for i in ax.patches:
    if i.get_width() > 20000:
        ax.text(i.get_width()-11000, i.get_y()+0.5,
                str(int(i.get_width())), fontsize=20, color='grey')
    else:
        ax.text(i.get_width()+30, i.get_y()+0.5,
                str(int(i.get_width())), fontsize=20, color='grey')
plt.show()
```



- Вывод:
  - Самое частое событие это появление основного экрана(login) случалось 3675 раза.
  - Второе по популярности событие это появление экрана с просмотром карточек продуктов(product_page)случалось 2303 раза.
  - Третье по популярности событие это появление экрана с покупками(purchase) случалось 1128 раз.
  - Четвёртое по популярности событие это появление экрана с просмотры корзины(product_cart) случалось 1079 раз.

Посчитаем, сколько пользователей совершали каждое из этих событий и долю пользователей, которые хоть раз совершали событие.

In [89]:
```
# Отсортируем события в логах по частоте
events.sort_values(by = 'n_users')
```

Out[89]:

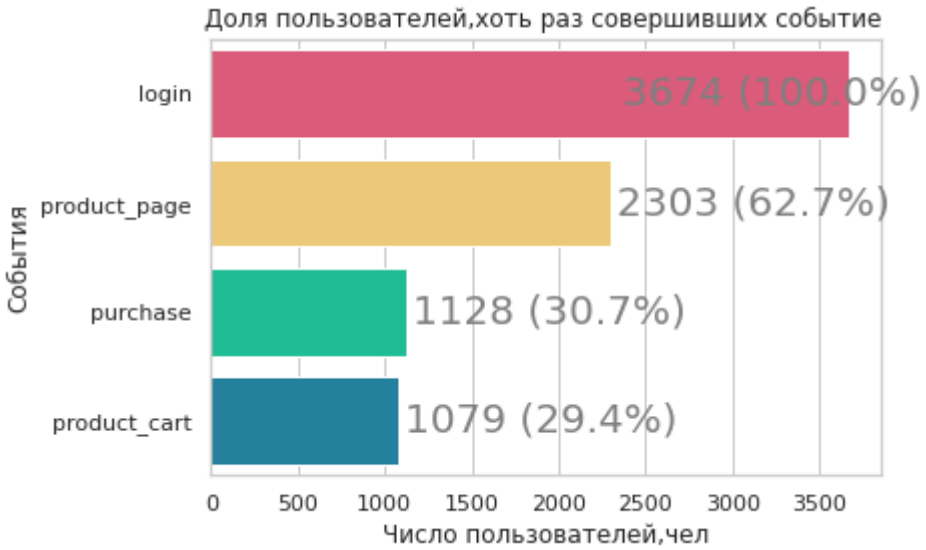
	event_name	n_users
1	product_cart	1079
3	purchase	1128
2	product_page	2303
0	login	3674

In [90]:
```
events['users_part'] = round(events['n_users']*100/len(df3_test['user_id'].unique()), 2)
```

In [91]:
```
diff['user_id'].nunique()
```

Out[91]: 3675

In [92]:
```
ax = sns.barplot(y='event_name', x='n_users', order = order,  data=events)
ax.set_title('Доля пользователей,хоть раз совершивших событие ')
ax.set_xlabel('Число пользователей,чел')
ax.set_ylabel('События')
for i in ax.patches:
    if i.get_width() > 3000:
        ax.text(i.get_width()-1300, i.get_y()+0.5,
            str(int(i.get_width()))+' ({:.1%})'.format(i.get_width() / n_users['all']), fontsiz
    else:
        ax.text(i.get_width()+30, i.get_y()+0.5,
            str(int(i.get_width()))+' ({:.1%})'.format(i.get_width() / n_users['all']), fontsiz
plt.figure(figsize=(14, 7))
plt.show()
```




```
<Figure size 1008x504 with 0 Axes>
```

In [93]:

```
login_users = diff.query('event_name =="login"')['user_id'].unique().tolist()
print('Кол-во пользователей все "login" =', len(login_users))

product_page = diff.query('event_name =="product_page"')['user_id'].unique().tolist()
print('Кол-во пользователей все "product_page" =', len(product_page))

purchase = diff.query('event_name =="purchase"')['user_id'].unique().tolist()
print('Кол-во пользователей все "purchase" =', len(purchase))

product_cart = diff.query('event_name =="product_cart"')['user_id'].unique().tolist()
print('Кол-во пользователей все "product_cart" =', len(product_cart))
```

```
Кол-во пользователей все "login" = 3674
Кол-во пользователей все "product_page" = 2303
Кол-во пользователей все "purchase" = 1128
Кол-во пользователей все "product_cart" = 1079
```

- Вывод:

  - 3675 пользователей хотя бы раз открывали главную страницу приложения(login)(54.8% от общего)

  - 2303 пользователей хотя бы раз открывали страницу с просмотром карточек товаров(product_page),это 34.4% всех пользователей.

  - Почти 38% пользователей не открыли каталог товаров.Возможно, приложение не на всех устройствах работает корректно. Следует проверить.

  - 1128 пользователей хотя бы раз попадали на страницу с успешной оплатой(purchase),это 16.8% всех пользователей.

  - 1079 пользователей хотя бы раз попадали на страницу с просмотром корзины(product_cart), это 16.1% всех пользователей.

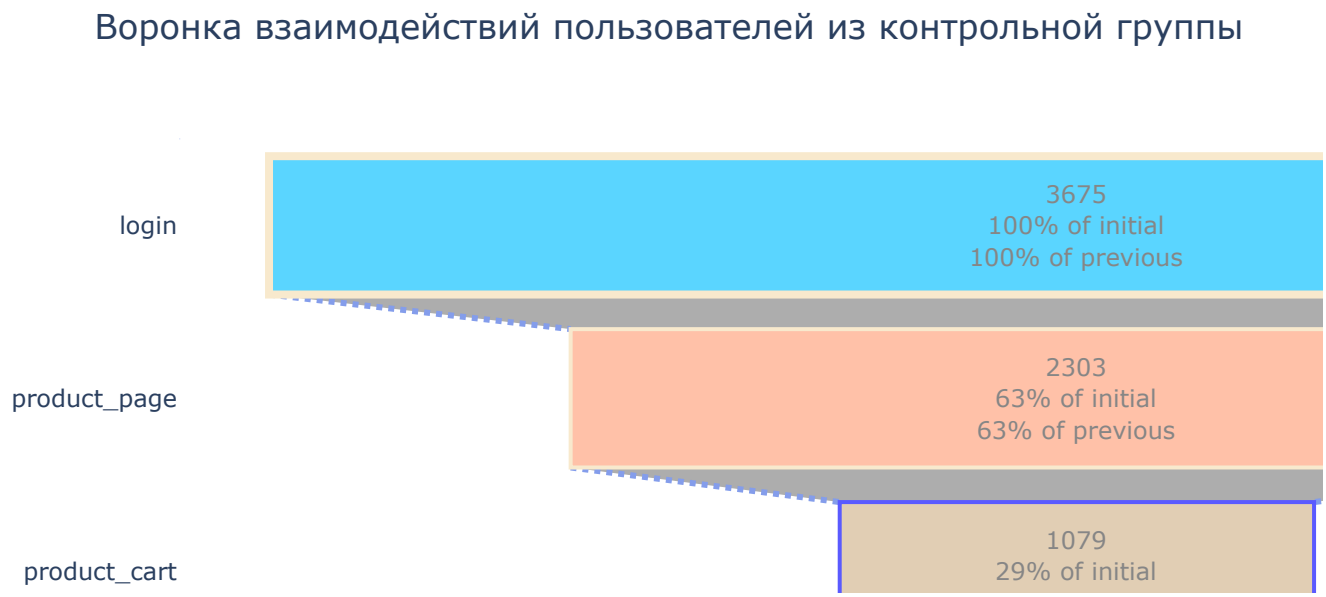
Предположим, в каком порядке происходят события

- Предполагаем,что события порядке происходят в таком порядке:
  - сначала открывает главную страницу приложения(login)
  - в каталоге выбирает товары(product_page)
  - кладёт их в корзину(product_cart)
  - оплачивает(purchase)

- Видим, что не все они выстраиваются в последовательную цепочку. Часть пользователей оплачивают сразу, не заходя в корзину.

По воронке событий посчитаем, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем)

In [94]:
```
fig = go.Figure(go.Funnel(y=['login', 'product_page','product_cart','purchase'], x=[3675,2303,1
    textinfo = "value+percent initial+percent previous",
    opacity = 0.65, marker = {"color": ["deepskyblue","lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}))
fig.update_layout(title='Воронка взаимодействий пользователей из контрольной группы')
fig.show()
```



- Вывод:
  - 63% пользователей заходят в каталог.
  - из них 47% заходят в корзину. Это 29% от общего
  - 31% пользователей совершают покупки
  - Больше всего пользователей теряется на третьем шаге - только около 47% переходят к корзине

Распределение событий по группам теста

In [95]:
```
diff.groupby(['group', 'event_name']).agg({'user_id':'nunique'})
```

Out[95]:

		user_id
group	**event_name**	

		user_id
group	**event_name**	
A	**login**	2747
	product_cart	824
	product_page	1780
	purchase	872
B	**login**	927
	product_cart	255
	product_page	523
	purchase	256

Изучим результаты эксперимента

Проверим, находят ли статистические критерии разницу между выборками A и B.

Построим воронку для каждой тестовой группы и сравним шаги этих воронок с помощью z-теста

Посчитаем количество пользователей в группах

In [96]:
```
group_a = diff.query('group =="A"')['user_id'].unique().tolist()
print('Кол-во пользователей все "A" =', len(group_a))

group_b = diff.query('group =="B"')['user_id'].unique().tolist()
print('Кол-во пользователей все "B" =', len(group_b))
```

```
Кол-во пользователей все "A" = 2747
Кол-во пользователей все "B" = 928
```

- Количество пользователей в группах:
  - группа A - 2747
  - группа B - 928

In [97]:
```
# Соотношение численности:
len(group_b)/len(group_a)
```

Out[97]: 0.33782307972333453

Численность группы B - 34% от численности группы A

Общее количество участников эксперимента

In [98]:
```
group_ = len(group_a) +len(group_b)
group_
```

Out[98]: 3675

Посчитаем количество пользователей в группах, открывших главный экран

In [99]:
```
df_login_a = diff.query('user_id in @group_a')
df_login_a[df_login_a['event_name'] == 'login']['user_id'].nunique()
df_login_a_b = df_login_a[df_login_a['event_name'] == 'login']['user_id'].nunique()
df_login_a_b
```

Out[99]: 2747

In [100…
```
df_login_b = diff.query('user_id in @group_b')
df_login_b[df_login_b['event_name'] == 'login']['user_id'].nunique()
df_login_a_bb = df_login_b[df_login_b['event_name'] == 'login']['user_id'].nunique()
df_login_a_bb
```

Out[100… 927

- Количество пользователей в группах, открывших главный экран :
  - В группе A - 2747 пользователей
  - В группе B - 927 пользователей

Посчитаем количество пользователей в группах,перешедших в просмотр каталога

In [101…
```
df_product_page_a = diff.query('user_id in @group_a')
df_product_page_a[df_product_page_a['event_name'] == 'product_page']['user_id'].nunique()
df_product_page_a_b = df_product_page_a[df_product_page_a['event_name'] == 'product_page']['use
df_product_page_a_b
```

Out[101… 1780

In [102…
```
df_product_page_b = diff.query('user_id in @group_b')
df_product_page_b[df_product_page_b['event_name'] == 'product_page']['user_id'].nunique()
df_product_page_a_bb = df_product_page_b[df_product_page_b['event_name'] == 'product_page']['us
df_product_page_a_bb
```

Out[102… 523

- В просмотр каталога прошли :
  - В группе A - 1780 пользователей
  - В группе B - 523 пользователей

Среднее количество событий на пользователя по группам

Контрольная группа A

In [103…
```
df_a = diff.query('user_id in @group_a')
df_a.groupby('user_id')['event_name'].agg('count').describe()
```

Out[103…
```
count    2747.000000
mean        6.917728
std         3.846632
min         1.000000
25%         4.000000
50%         6.000000
75%         9.000000
max        24.000000
Name: event_name, dtype: float64
```

Экспериментальная группа B

In [104…
```
df_b = diff.query('user_id in @group_b')
df_b.groupby('user_id')['event_name'].agg('count').describe()
```

Out[104…
```
count    928.000000
mean       5.551724
std        3.303020
min        1.000000
25%        3.000000
50%        4.000000
75%        8.000000
max       24.000000
Name: event_name, dtype: float64
```

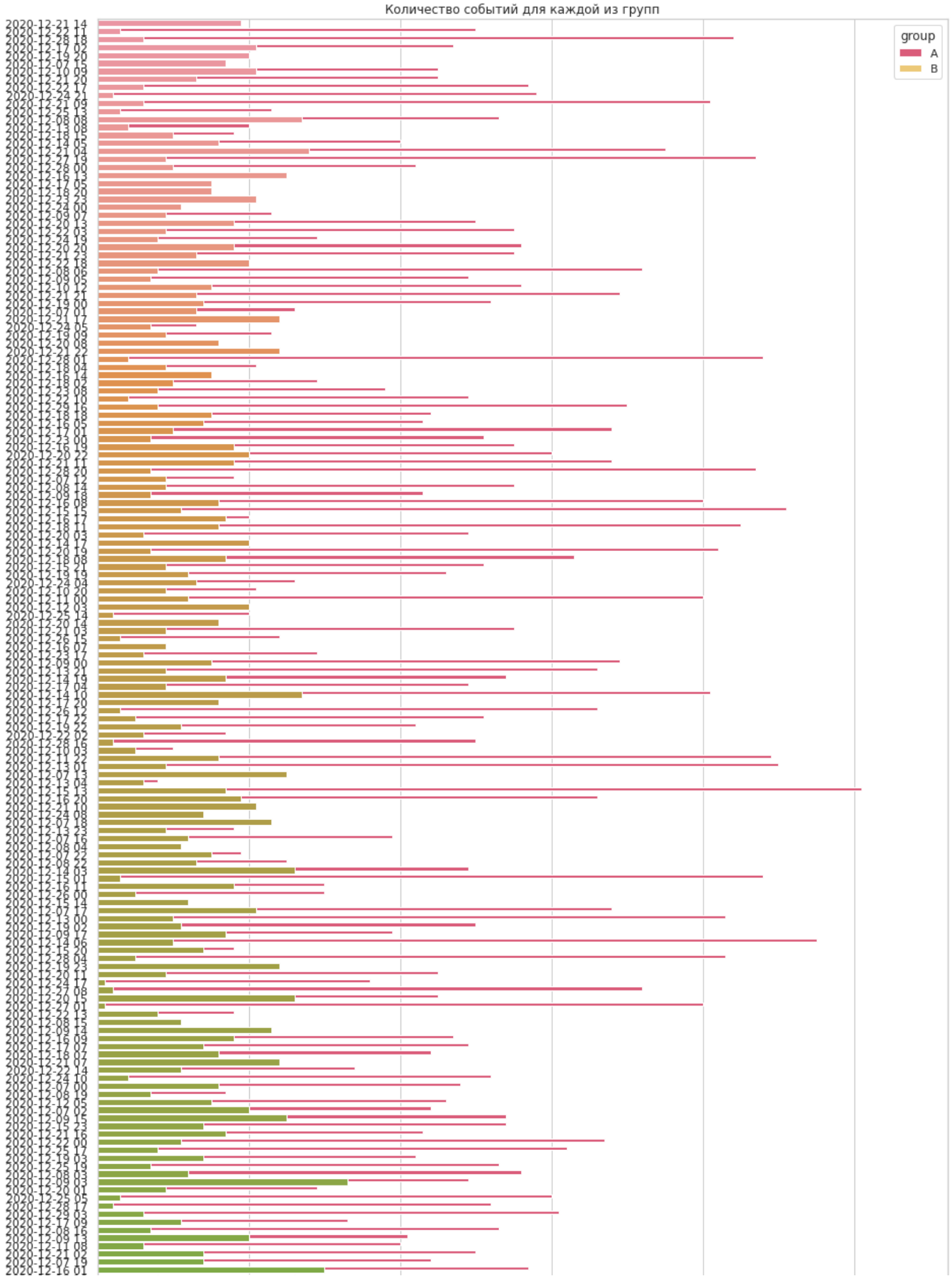
Количество событий на пользователя распределены в выборках неодинаково.В группе A - 6 и в группе B -4.

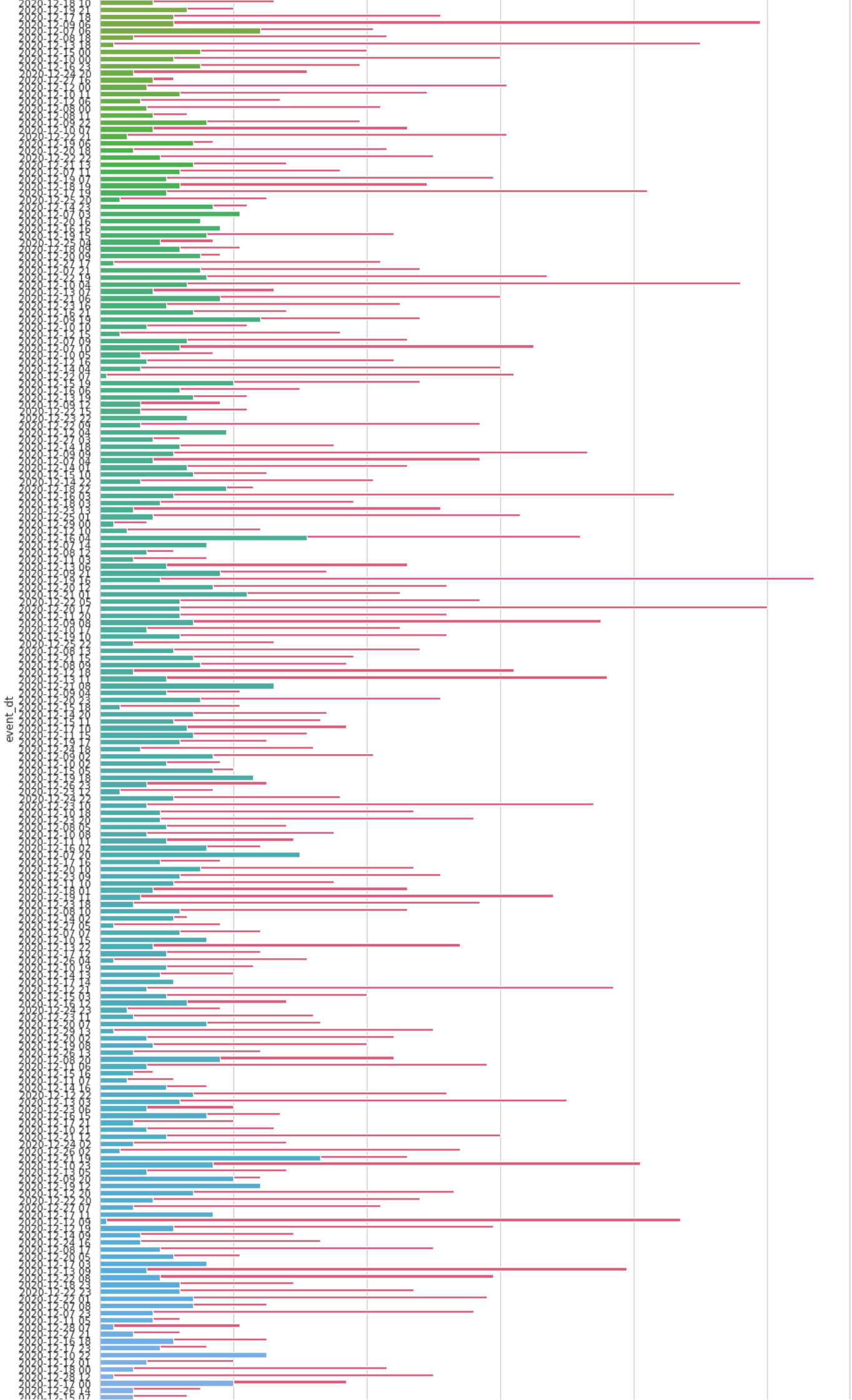
Распределение числа событий в выборках по дням

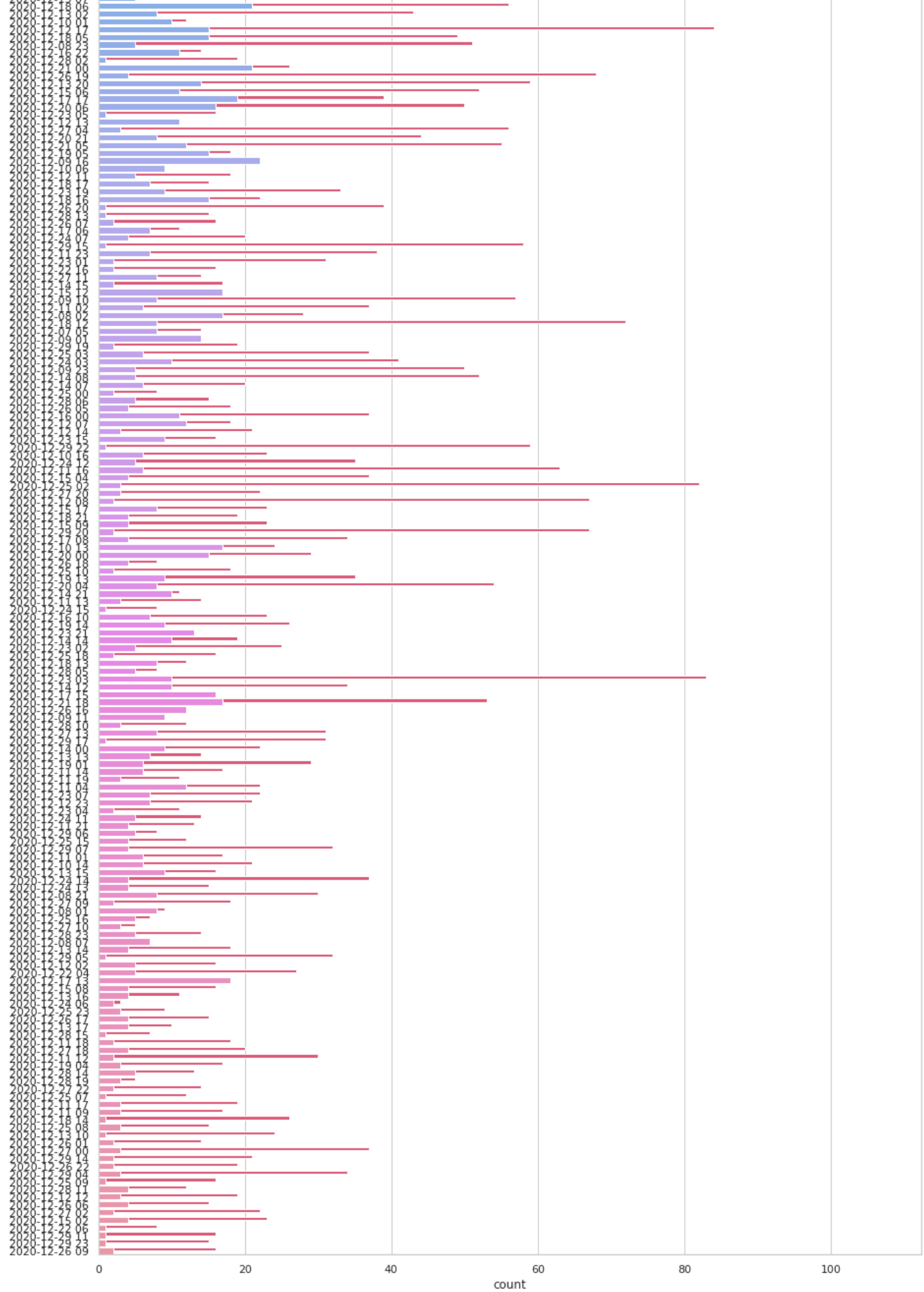
Построим общий график

In [105…

```
plt.figure(figsize=(15, 75))
ax = sns.countplot(y=df_a['event_dt'].dt.strftime('%Y-%m-%d %H'),  data=df3_new,hue='group')
ax = sns.countplot(y=df_b['event_dt'].dt.strftime('%Y-%m-%d %H'),  data=df3_new)
ax.set_title('Количество событий для каждой из групп')
plt.show()
```




event_dt



Гистограмма по дате и времени

In [106…

```
ax = diff['event_dt'].hist(bins = 100)
plt.title('Гистограмма по дате и времени', size = 15)
plt.ylabel('Частота', size = 30)
plt.xticks(rotation=45)
plt.show()
```

Видим плавный спад до 2020-12-13,затем резкий скачок в 3 раза. мах 2020-12-21. Затем спад.

Посчитаем конверсии

Конверсия общего числа пользователей в login

In [107…
```
print(f'Конверсия группы A в login  = {round(df_login_a_b/len(group_a)*100)}%')
```

```
Конверсия группы A в login  = 100%
```

In [108…
```
print(f'Конверсия группы B в login  = {round(df_login_a_bb/len(group_b)*100)}%')
```

```
Конверсия группы B в login  = 100%
```

Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей

Соберем данные для теста:

Кол-во всего пользователей в группе A = len(group_a)

Кол-во пользователей группы A,перешедших в просмотр каталога = df_login_a_b

Кол-во всего пользователей в группе B = len(group_b)

Кол-во пользователей группы B,перешедших в просмотр каталога = df_login_a_bb

Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)

In [109…
```
def recommender_system_test(successes, trials, alpha):
    alpha = alpha
    successes = successes
    trials = trials

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

    # пропорция успехов во второй группе:
    p2 = successes[1]/trials[1]

    # пропорция успехов в комбинированном датасете:
    p_combined = (successes[0] + successes[1]) / (trials[0] + trials[1])

    # разница пропорций в датасетах
```

```
    difference = p1 - p2


    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[0] + 1/trials[1]))

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


    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(
        p_combined * (1 - p_combined) * (1 / trials[0] + 1 / trials[1])
    )

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

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

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


alpha = .05

successes = [df_login_a_b,df_login_a_bb]

trials = [len(group_a),len(group_b)]

recommender_system_test(successes, trials, alpha)
print(trials)
print(successes)
```

```
p-значение:  0.08529860212027773
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
[2747, 928]
[2747, 927]
```

Конверсия login в product_page

In [130…

```
print(f'Конверсия группы A в product_page  = {round(df_product_page_a_b/len(group_a)*100)}%')
```

```
Конверсия группы A в product_page  = 65%
```

In [131…

```
print(f'Конверсия группы B в product_page  = {round(df_product_page_a_bb/len(group_b)*100)}%')
```

```
Конверсия группы B в product_page  = 56%
```

Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей

Соберем данные для теста:

Кол-во всего пользователей в группе A = len(group_a)

Кол-во пользователей группы A,перешедших в просмотр каталога = df_product_page_a_b

Кол-во всего пользователей в группе B = len(group_b)

Кол-во пользователей группы B,перешедших в просмотр каталога = df_product_page_a_bb

Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)

In [112…
```
alpha = .05

successes = [df_product_page_a_b,df_product_page_a_bb]

trials = [len(group_a),len(group_b)]

recommender_system_test(successes, trials, alpha)
print(trials)
print(successes)
```

```
p-значение:  4.310980554755872e-06
Отвергаем нулевую ни гипотезу: между долями есть значимая разница
[2747, 928]
[1780, 523]
```

Посчитаем количество пользователей в группах,совершивших покупки

In [113…
```
df_purchase_a = df3_new.query('user_id in @group_a')
df_purchase_a[df_purchase_a['event_name'] == 'purchase']['user_id'].nunique()
df_purchase_a_b = df_purchase_a[df_purchase_a['event_name'] == 'purchase']['user_id'].nunique()
df_purchase_a_b
```

Out[113… 872

In [114…
```
df_purchase_b = df3_new.query('user_id in @group_b')
df_purchase_b[df_purchase_b['event_name'] == 'purchase']['user_id'].nunique()
df_purchase_a_bb = df_purchase_b[df_purchase_b['event_name'] == 'purchase']['user_id'].nunique(
df_purchase_a_bb
```

Out[114… 256

- Покупки совершили :
  - В группе A - 872 пользователей
  - В группе B - 256 пользователей

In [115…
```
df_purchase_a_bb/df_purchase_a_b
```

Out[115… 0.29357798165137616

Конверсия login в purchase

In [116…
```
print(f'Конверсия группы A в  purchase = {round(df_purchase_a_b/len(group_a)*100)}%')
```

```
Конверсия группы A в  purchase = 32%
```

In [117…
```
print(f'Конверсия группы B в  purchase = {round(df_purchase_a_bb/len(group_b)*100)}%')
```

```
Конверсия группы B в  purchase = 28%
```

Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей

Соберем данные для теста:

Кол-во всего пользователей в группе A = len(group_a)

Кол-во пользователей группы A,совершивших покупку = df_purchase_a_b

Кол-во всего пользователей в группе B = len(group_b)

Кол-во пользователей группы B,совершивших покупку = df_purchase_a_bb

Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)

In [118…
```
alpha = .05

successes = [df_purchase_a_b,df_purchase_a_bb]

trials = [len(group_a),len(group_b)]
recommender_system_test(successes, trials, alpha)
print(trials)
print(successes)
```

```
p-значение:  0.017592402663314743
Отвергаем нулевую ни гипотезу: между долями есть значимая разница
[2747, 928]
[872, 256]
```

Посчитаем количество пользователей в группах,перешедших в просмотр корзины

In [119…
```
df_product_cart_a = df3_new.query('user_id in @group_a')
df_product_cart_a[df_product_cart_a['event_name'] == 'product_cart']['user_id'].nunique()
df_product_cart_a_b = df_product_cart_a[df_product_cart_a['event_name'] == 'product_cart']['use
df_product_cart_a_b
```

Out[119… 824

In [120…
```
df_product_cart_b = df3_new.query('user_id in @group_b')
df_product_cart_b[df_product_cart_b['event_name'] == 'product_cart']['user_id'].nunique()
df_product_cart_a_bb = df_product_cart_b[df_product_cart_b['event_name'] == 'product_cart']['us
df_product_cart_a_bb
```

Out[120… 255

- В просмотр корзины вошли:
  - В группе A - 824 пользователей
  - В группе B - 255 пользователей

In [121…
```
df_product_cart_a_bb/df_product_cart_a_b
```

Out[121… 0.3094660194174757

Конверсия login в product_cart

In [122…
```
print(f'Конверсия группы A в product_cart  = {round(df_product_cart_a_b/len(group_a)*100)}%')
```

```
Конверсия группы A в product_cart  = 30%
```

In [123…
```
print(f'Конверсия группы B в product_cart  = {round(df_product_cart_a_bb/len(group_b)*100)}%')
```

```
Конверсия группы B в product_cart  = 27%
```

Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей

Соберем данные для теста:

Кол-во всего пользователей в группе A = len(group_a)

Кол-во пользователей группы A,совершивших покупку = df_product_cart_a_b

Кол-во всего пользователей в группе B = len(group_b)

Кол-во пользователей группы B,совершивших покупку = df_product_cart_a_bb

Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)

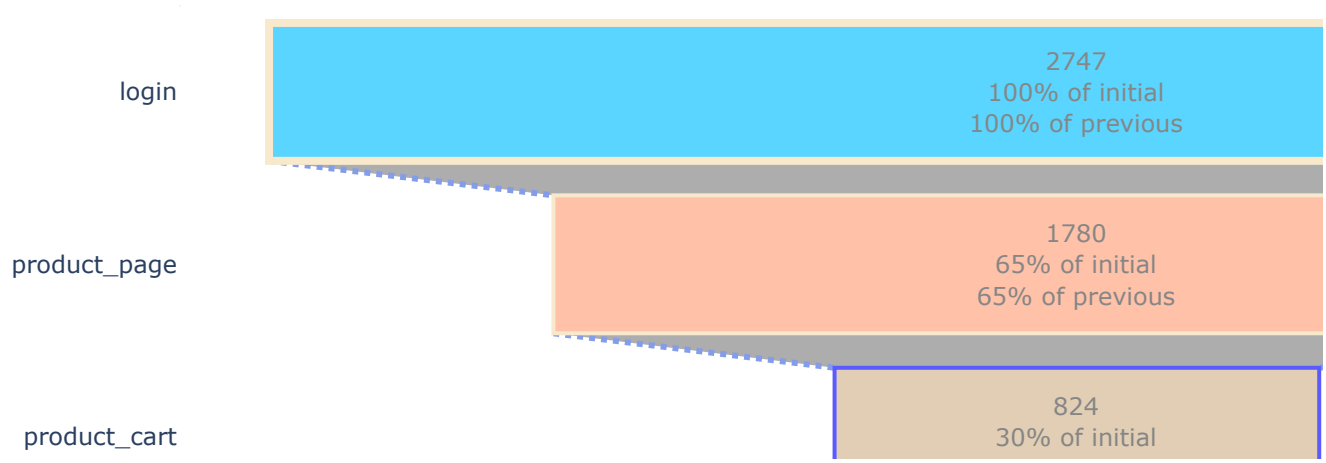
In [124…

```
alpha = .05

successes = [df_product_cart_a_b,df_product_cart_a_bb]

trials = [len(group_a),len(group_b)]

recommender_system_test(successes, trials, alpha)
print(trials)
print(successes)
```

```
p-значение:  0.14534814557238196
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
[2747, 928]
[824, 255]
```

In [125…

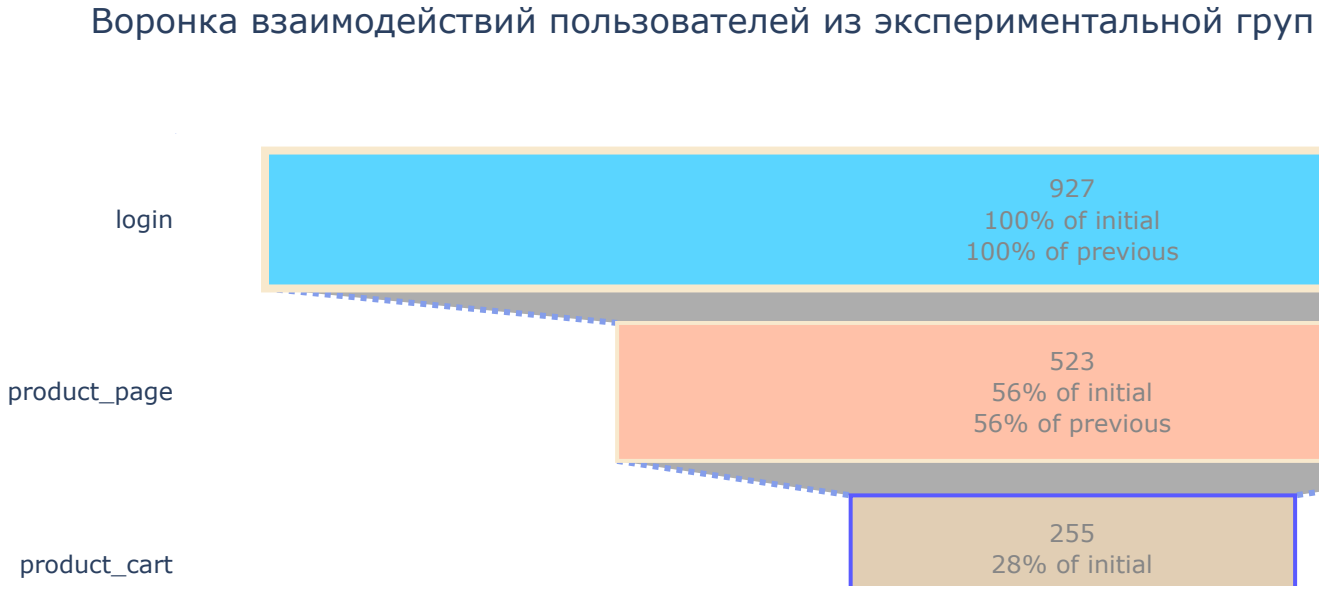
```
fig = go.Figure(go.Funnel(y=['login', 'product_page','product_cart','purchase'], x=[2747,1780,8
    textinfo = "value+percent initial+percent previous",
    opacity = 0.65, marker = {"color": ["deepskyblue","lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}))
fig.update_layout(title='Воронка взаимодействий пользователей из контрольной группы')
fig.show()
```



- В контрольной группе:
  - 65% пользователей заходят в каталог
  - из зашедших в каталог , 46% заходят в корзину. Это 30% от общего числа пользователей
  - оплачивают 32% от общего числа пользователей

In [126…

```
fig = go.Figure(go.Funnel(y=['login', 'product_page','product_cart','purchase'], x=[927,523,255
    textinfo = "value+percent initial+percent previous",
    opacity = 0.65, marker = {"color": ["deepskyblue","lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]}},
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}))
fig.update_layout(title='Воронка взаимодействий пользователей из экспериментальной группы')
fig.show()
```



- В экспериментальной группе:
  - 56% открывают каталог
  - из открывших каталог, 49% заходят в корзину.Это 28% от общего числа
  - 28% от общего числа пользователей оплачивают

In [127…

```
test_data = df3_new.groupby(['group', 'event_name']).agg({'user_id':'nunique'}).reset_index()
test_data
```

Out[127…

	group	event_name	user_id
0	A	login	2747
1	A	product_cart	824

	group	event_name	user_id
2	A	product_page	1780
3	A	purchase	872
4	B	login	928
5	B	product_cart	255
6	B	product_page	523
7	B	purchase	256

In [128…

```
result_conversion = pd.DataFrame(index=['product_cart', 'product_page', 'purchase'], \
                                 columns=['A_conversion, %','B_conversion, %','delta, %'])
```

In [129…

```
for i in [1,2,3]:
    result_conversion.iloc[i-1, 0] = round((test_data.iloc[i,2] / test_data.iloc[0,2] * 100), 1

for i in [1,2,3]:
    result_conversion.iloc[i-1, 1] = round((test_data.iloc[i+4,2] / test_data.iloc[4,2] * 100),

result_conversion['delta, %'] = (result_conversion['B_conversion, %'] / result_conversion['A_co

#result_conversion['delta, %'] = round( - result_conversion['delta, %'], 1)
result_conversion
```

Out[129…

	A_conversion, %	B_conversion, %	delta, %
product_cart	30.0	27.5	-8.333333
product_page	64.8	56.4	-12.962963
purchase	31.7	27.6	-12.933754

Конверсия в экспериментальной группе снизилась

Выводы:

- В процессе предобработки:
  - проверили на дубли и отсутствующие значения
  - изменили тип данных в формат datetime.
  - Тип данных в каждой колонке — правильный Дубликатов нет.
- Изучили и проверили данные:
  - всего событий в логе за актуальный период 24155 события,не выходящих за 14-дневный срок
  - всего пользователей в логе 6701
  - нашли максимальную и минимальную дату : 2020-12-07 и 2021-01-04.
  - в среднем на пользователя приходится 6 событий.
  - построили гистограмму по дате и времени.
- Оценили корректность проведения теста:

```
- Проверили корректность всех пунктов технического задания:
    - Дата остановки не совпадает - фактическая 2020-12-30
    - Дата дата остановки набора новых пользователей: 2020-12-23 не совпадает с
техзаданием
    - Ожидаемое количество участников теста: 6000 не совпадает. Фактически -3675
    - Количество новых пользователей из Европы в тесте 15.2%
    - Нашли совпадение теста и маркетингового события  "Christmas&New Year Promo"с
2020-12-25 по 2021-01-0
```

- Определили, что есть пересечение с конкурирующим тестом и нет пользователей, участвующих в двух группах теста одновременно
- Проверили равномерность распределения по тестовым группам и правильность их формирования
    - пользователей в группах: A -2747, B -938, разница в 3 раза

- Изучили воронку событий:

- Самое частое событие это появление основного экрана(login) случалось 3675 раз.
- Второе по популярности событие это появление экрана с просмотром карточек продуктов(product_page)случалось 2303 раза.
- Третье по популярности событие это появление экрана с покупками(purchase) случалось 1128 раз.
- Четвёртое по популярности событие это появление экрана с просмотры корзины(product_cart) случалось 1079 раз
- Предполагаем,что события порядке происходят в таком порядке:

    - сначала открывает главную страницу приложения(login)
    - в каталоге выбирает товары(product_page)
    - кладёт их в корзину(product_cart)
    - оплачивает(purchase)
    - Видим, что не все они выстраиваются в последовательную цепочку. Часть пользователей оплачивают сразу, не заходя в корзину.
- Больше всего пользователей уходит от просмотра каталога к корзине - половина из открывших каталог
- От первого события до оплаты доходит третья часть пользователей()
    - Изучили результаты эксперимента:
        - В контрольной группе:
            - 65% пользователей заходят в каталог
            - из зашедших в каталог , 46% заходят в корзину. Это 30% от общего числа пользователей
            - оплачивают 32% от общего числа пользователей
        - В экспериментальной группе:
            - 56% открывают каталог
            - из открывших каталог, 49% заходят в корзину.Это 28% от общего числа
            - 28% от общего числа пользователей оплачивают

- - Оценили результаты A/B-тестирования
        - По результатам A/B-тестирования можно сказать, что ожидаемый эффект:
            - за 14 дней с момента регистрации пользователи покажут улучшение каждой метрики не менее, чем на 10%:
                - конверсии в просмотр карточек товаров — событие `product_page` ,
                - просмотры корзины — 'product_cart',
                - покупки — `purchase` ,
            - не достигнут. Показатели экспериментальной группы хуже.
        - Проверили статистическую разницу долей z-критерием:
            - на этапе "login" и "product_cart" не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
            - на остальных этапах получилось отвергнуть нулевую гипотезу,между долями есть значимая разница

Главный вывод:

- Тест проведен некорректно.

- По результатам исслдования видно, что цели эксперимента не достигнуты, конверсии не увеличились.