Содержание:

Проект выполнил: Алексей Становой, в рамках учебного курса "Аналитик данных" Yandex.Practicum Москва, Февраль 2021г. +7 (962) 985-51-77 Телефон - WhatsApp - Telegram - e-mail: a.v.stanovoy@yandex.ru

Проект:

Оценка результатов A/B-теста.

Цель:

- Оценить корректность проведения A/B теста.
- Проанализировать результаты теста.

Техническое задание проведения A/B теста:

- 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 в системе пользователи покажут улучшение каждой метрики не менее, чем на 10%:
  - конверсии в просмотр карточек товаров — событие product_page
  - просмотры корзины — product_card
  - покупки — purchase.

Материалы:

- /datasets/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;
  - `inish_dt` — дата завершения кампании.
- /datasets/final_ab_new_users.csv — все пользователи, зарегистрировавшиеся в интернет-магазине в период 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.
- /datasets/final_ab_events.csv — все события новых пользователей в период с 7 декабря 2020 по 4 января 2021 года; Структура файла:
  - `user_id` — идентификатор пользователя;
  - `event_dt` — дата и время события;
  - `event_name` — тип события;
  - `details` — дополнительные данные о событии. Например, для покупок, purchase, в этом поле хранится стоимость покупки в долларах.
- /datasets/final_ab_participants.csv — таблица участников тестов. Структура файла:
  - `user_id` — идентификатор пользователя;
  - `ab_test` — название теста;
  - `group` — группа пользователя.

1. Загрузка и проверка данных.

In [1]:
```
# Библиотеки
import pandas as pd
import datetime as dt
from IPython.display import display
from plotly import graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
import numpy as np
import math as mth
from scipy import stats as st
import scipy.stats as stats
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import seaborn as sns
from matplotlib import pyplot as plt
```

Календарь маркетинговых событий на 2020 год.

In [2]:
```
# извлечение данныех
events_calender = pd.read_csv('/datasets/ab_project_marketing_events.csv')
# выводим таблицу
display(events_calender)
# вывод информацию о таблице
events_calender.info()
print('')
# проверка наличия задвоенных строк
print('Количество дублированных строк:', events_calender.duplicated().sum())
# проверка наличия задвоенных "user_id"
print('Количество дублированных "name":', events_calender['name'].duplicated().sum())
```

	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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 4 columns):
name         14 non-null object
regions      14 non-null object
start_dt     14 non-null object
finish_dt    14 non-null object
dtypes: object(4)
memory usage: 576.0+ bytes

Количество дублированных строк: 0
Количество дублированных "name": 0
```

ВЫВОД:

- Таблица `events_calender` содержит календарь маркетинговых событий на 2020 год.
- В таблице 4 столбца и 14 строк.
- Пропущенных значений нет. Дублированных строк, а также названий мероприятий в столбце `name` нет.
- Названия столбцов соответствуют содержанию, записаны нижним регистром.
- Тип данных во всех столбцах указан как "object", хотя в столбцах `start_dt` и `finish_dt` содержатся даты. Нужно преобразовать их в подходящий тип данных.

In [3]:
```
# приведение 'start_dt' и 'finish_dt' к формату datetime
events_calender['start_dt'] = pd.to_datetime(events_calender['start_dt'])
events_calender['finish_dt'] = pd.to_datetime(events_calender['finish_dt'])
events_calender.info()
```

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

Все пользователи, зарегистрировавшиеся в интернет-магазине в период с 7 по 21 декабря 2020 года

In [4]:
```
# извлекаем данные
all_users = pd.read_csv('/datasets/final_ab_new_users.csv')
# выводим таблицу
display(all_users.head(5))
# выводим информацию о таблице
all_users.info()
print('')
# проверка наличия задвоенных строк
print('Количество дублированных строк:', all_users.duplicated().sum())
# проверка наличия задвоенных "user_id"
print('Количество дублированных "user_id":', all_users['user_id'].duplicated().sum())
```

	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

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

Количество дублированных строк: 0
Количество дублированных "user_id": 0
```

ВЫВОД:

- Таблица `all_users` содержит данные о пользователях, зарегистрировавшихся в интернет-магазине в период с 7 по 21 декабря 2020 года.
- В таблице 4 столбца и 61733 строки.
- Пропущенных значений нет. Дублированных строк, а также ID клиентов в столбце `user_id` нет.
- Названия столбцов соответствуют содержанию, записаны нижним регистром.
- Тип данных во всех столбцах указан как "object", хотя в столбце `first_date` содержатся даты. Нужно преобразовать их в подходящий тип данных.

In [5]:
```
# приведение 'first_date' к формату datetime
all_users['first_date'] = pd.to_datetime(all_users['first_date'])
```

Все события новых пользователей в период с 7 декабря 2020 по 4 января 2021 года

In [6]:
```
# извлекаем данные
ab_events = pd.read_csv('/datasets/final_ab_events.csv')
# выводим таблицу
display(ab_events.head(5))
# выводим информацию о таблице
ab_events.info()
print('')
# проверка наличия задвоенных строк
print('Количество дублированных строк:', ab_events.duplicated().sum())
# проверка наличия задвоенных "user_id"
print('Количество дублированных "user_id":', ab_events['user_id'].duplicated().sum())
```

	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

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

Количество дублированных строк: 0
Количество дублированных "user_id": 381614
```

In [7]:
```
# группировка по столбцу "event_name" поможет выявить причину пропусков.
ab_events.groupby('event_name')['details'].count()
```

Out[7]:
```
event_name
login              0
product_cart       0
product_page       0
purchase       62740
Name: details, dtype: int64
```

ВЫВОД:

- Таблица `ab_events` содержит данные о всех событиях новых пользователей в период с 7 декабря 2020 по 4 января 2021 года
- В таблице 4 столбца и 440317 строк.
- Пропущенные значения есть только в столбце `details`, который содержит дополнительные данные о событиях. В представленной таблице присутствуют дополнительные данные только для события "purchase", покупки. По условию в этом поле хранится стоимость покупки в долларах. Так как для других событий эта информация не характерна, то в таблице присутствуют пропуски.
- Дублированных строк нет. Дублируются ID клиентов в столбце `user_id`, так как один клиент может совершать несколько событий, которые записаны в разных строчках.
- Названия столбцов соответствуют содержанию, записаны нижним регистром.
- Тип данных в столбце `event_dt` указан как "object", хотя содержатся в нем содержатся даты. Нужно преобразовать их в подходящий тип данных. В остальных столбцах типы данных сообветствуют содержанию.

In [8]:
```
# приведение 'event_dt' к формату datetime
ab_events['event_dt'] = pd.to_datetime(ab_events['event_dt'])
```

Таблица участников тестов.

In [9]:
```
# извлекаем данные
ab_participants = pd.read_csv('/datasets/final_ab_participants.csv')
# выводим таблицу
display(ab_participants.head(5))
# выводим информацию о таблице
ab_participants.info()
print('')
# проверка наличия задвоенных строк
print('Количество дублированных строк:', ab_participants.duplicated().sum())
# проверка наличия задвоенных "user_id"
print('Количество пользователей, принявших участие в тестах более одного раза:',
      ab_participants['user_id'].duplicated().sum())
```

	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

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

Количество дублированных строк: 0
Количество пользователей, принявших участие в тестах более одного раза: 1602
```

Чтобы проверить природу повторного участия пользователей в тестах нужно взглянуть на содержание столбца `ab_test` , в котором хранится название проводимого теста.

In [10]:
```
# группировка по столбцу "ab_test" с подсчетом количества 'user_id'.
ab_participants.groupby('ab_test')['user_id'].count().reset_index()
```

Out[10]:

	ab_test	user_id
0	interface_eu_test	11567
1	recommender_system_test	6701

Имеется информация о двух тестах. Нужна проверка наличия повторяющихся участников в одном и том же тесте.

In [11]:
```
print('Количество участников "recommender_system_test", принявших повторное участие в тесте:',
      ab_participants.query('ab_test == "recommender_system_test"').duplicated().sum())
print('Количество участников "interface_eu_test", принявших повторное участие в тесте:',
      ab_participants.query('ab_test == "interface_eu_test"').duplicated().sum())
```

```
Количество участников "recommender_system_test", принявших повторное участие в тесте: 0
Количество участников "interface_eu_test", принявших повторное участие в тесте: 0
```

Итак, в таблице представлены данные двух тестов. Данный анализ касается только 'recommender_system_test'. Будет проведена проверка в срезе данных по этому параметру.

ВЫВОД:

- Таблица `ab_participants` содержит данные об участниках двух тестов: "recommender_system_test" и "interface_eu_test".
- В таблице 3 столбца и 18268 строк.
- Названия столбцов соответствуют содержанию, записаны нижним регистром.
- Тип данных во всех столбцах указан как "object", что соответствует содержанию.
- Пропущенных значений нет.
- Дублированных строк нет.
- Дубликаты выявлены в столбце 'user_id' среди участников тестов.
- Проверка показала, что повторного участия в одном и том же тесте среди пользователей нет. А значит некоторые пользователи приняли участие в обоих тестах, что может исказить результаты.
- Для дальнейшего анализа понадобятся участники только теста "recommender_system_test", которые не принимали участие в параллельном тесте.

Для дальнейшего анализа нужно удалить из таблицы данные участников параллельного теста.

In [12]:
```
# методу drop_duplicates() будет передан парамметр keep=False для удаления всех строк
# с повторяющимися пользователями.
no_duplicates_participants = ab_participants.drop_duplicates(subset=['user_id'],keep=False)
# проверка работы метода
print('Количество пользователей, принявших участие в тестах более одного раза:',
      no_duplicates_participants['user_id'].duplicated().sum())
# итоговое распределение участников теста.
display(no_duplicates_participants.groupby('ab_test')['user_id'].count().reset_index())
```

```
Количество пользователей, принявших участие в тестах более одного раза: 0
```

	ab_test	user_id
0	interface_eu_test	9965
1	recommender_system_test	5099

Таблица очищена от повторяющихся значений.

2. Исследовательский анализ данных.

Для начала необходимо собрать из таблиц данные, которые отвечают техническому заданию анализа.

Подготовка списка участников анализируемого теста: _recommender_system*test*.

In [13]:
```
# сортировка методом query
test_participes=no_duplicates_participants.query('ab_test == "recommender_system_test"')
```

Подготовка списка пользователей региона, соответствующего техническому заданию: *EU*

In [14]:
```
# сортировка методом query
eu_users=all_users.query('region == "EU"')
```

Объединение данных в одну таблицу.

In [15]:
```
# Использование полседовательно метода merge
events_eu_ab=ab_events.merge(eu_users, on='user_id').merge(test_participes, on='user_id')
# вывод общей информации о новой таблице
print(events_eu_ab.info())
# проверка наличия задвоенных строк
print('')
print('Количество дублированных строк:', events_eu_ab.duplicated().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17526 entries, 0 to 17525
Data columns (total 9 columns):
user_id       17526 non-null object
event_dt      17526 non-null datetime64[ns]
event_name    17526 non-null object
details       2348 non-null float64
first_date    17526 non-null datetime64[ns]
region        17526 non-null object
device        17526 non-null object
group         17526 non-null object
ab_test       17526 non-null object
dtypes: datetime64[ns](2), float64(1), object(6)
memory usage: 1.3+ MB
None

Количество дублированных строк: 0
```

Соответствует ли диапозон дат условию?

In [16]:
```
print("Дата и время первого события: ", events_eu_ab['event_dt'].min())
print("Дата и время последнего события: ", events_eu_ab['event_dt'].max())
print("Начало набора новых пользователей: ", events_eu_ab['first_date'].min())
print("Окончание набора новых пользователей: ", events_eu_ab['first_date'].max())
```

```
Дата и время первого события:  2020-12-07 00:16:00
Дата и время последнего события:  2020-12-30 06:42:52
Начало набора новых пользователей:  2020-12-07 00:00:00
Окончание набора новых пользователей:  2020-12-21 00:00:00
```

ВЫВОД:

- Сборная таблица `events_eu_ab` для анализа содержит 9 столбцов и 17526 строк.
- Дублированных строк нет.
- Пропуски содержатся только в столбце `details` , который содержит информацию о стоимости покупки в долларах для события "purchase".
- Диапазон дат соответствует техническому заданию.

Конверсия в воронке на разных этапах.

Подсчет количества событий для каждого этапа.

In [17]:
```
# Группировка методом groupby() и count()
event_stages=events_eu_ab.groupby('event_name')['user_id'].count().reset_index()
# переименование столбца
event_stages.columns=('event_name', 'count_events' )
# добавление столбца с долей событий на каждом этапе от общего их числа.
event_stages['rate_%'] = ((event_stages['count_events']/event_stages['count_events'].sum())*100).round(1)
# сортировка по количеству событий
event_stages.sort_values(by='count_events', ascending=False)
```

Out[17]:

	event_name	count_events	rate_%
0	login	7906	45.1
2	product_page	4922	28.1
1	product_cart	2350	13.4
3	purchase	2348	13.4

Подсчет количества уникальных пользователей для каждого события.

In [18]:
```
# Группировка методом groupby() и nunique()
event_type=events_eu_ab.groupby('event_name')['user_id'].nunique().reset_index()
# переименование событий
event_type.columns=('event_name', 'count_users' )
# добавление столбца с долей уникальных пользователей каждого события от общего их числа.
event_type['rate_%'] = ((event_type['count_users']/events_eu_ab['user_id'].nunique())*100).round(1)
# сортировка по количеству пользователей
event_funnel=event_type.sort_values(by='count_users', ascending=False)
event_funnel
```

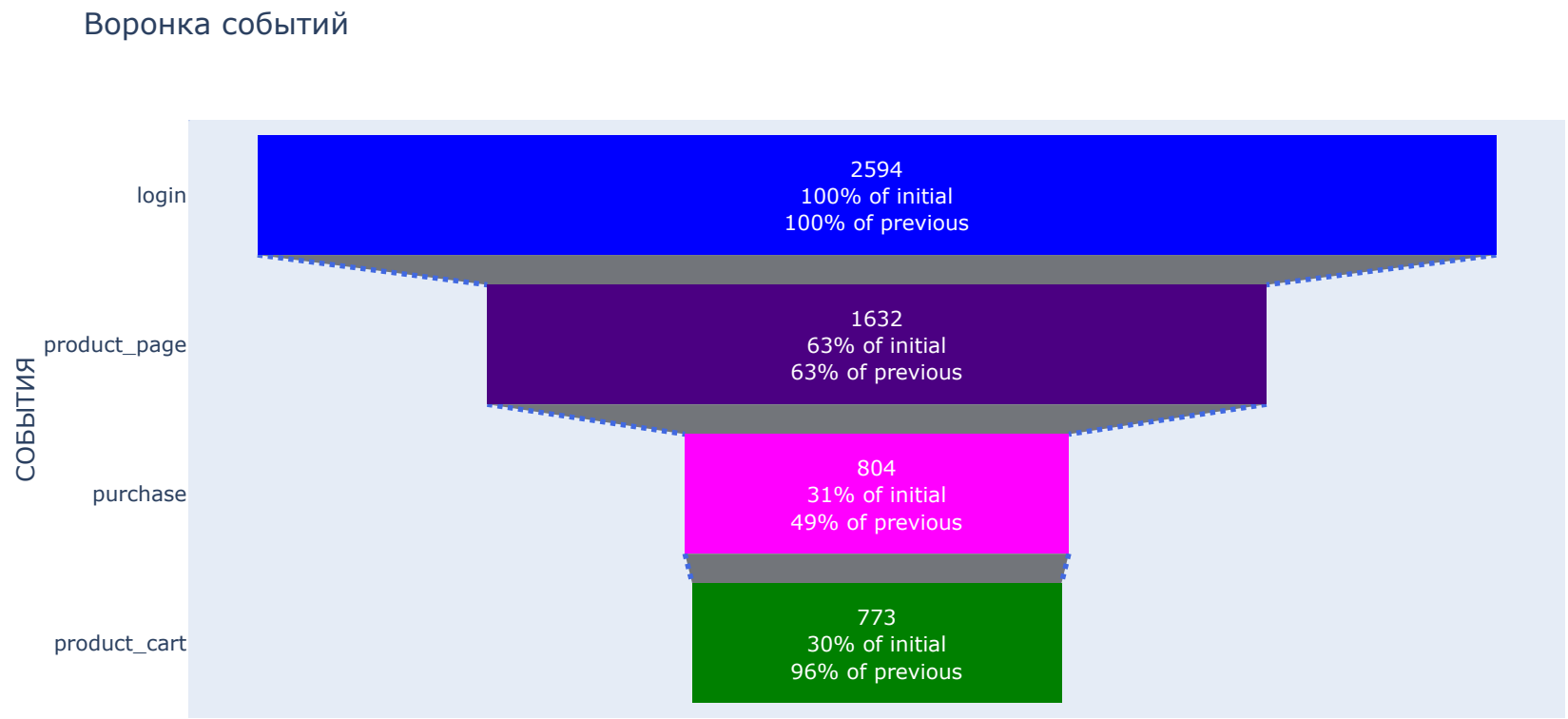
Out[18]:

	event_name	count_users	rate_%
0	login	2594	100.0
2	product_page	1632	62.9
3	purchase	804	31.0
1	product_cart	773	29.8

Построение воронки.

In [19]:
```
# вывод воронки при помощи "plotly"
fig = go.Figure(go.Funnel(
    y = event_funnel['event_name'], marker={'color': ['blue', 'indigo', 'fuchsia', 'green']},
    textinfo='percent previous+percent initial+value',
    x = event_funnel['count_users'],
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}
    ))

fig.update_layout(title='Воронка событий', yaxis = dict(title='СОБЫТИЯ'))
fig.show()
```



ВВЫВОД:

- Имеется 4 события: login - вход пользователя в систему, product_page - просмотр картинки продукта, purchase - покупка, product_cart - просмотр корхины товаров.
- По количеству событий выделяется:
  - login - можно рассматривать как первое событие.
  - product_page - второе событие.
  - далее product_cart,
  - purchase - последнее.
- Но воронка по количеству уникальных пользователей показывает преоблодание их числа на этапе **purchase** по сравнению с **product_cart**.
- Скорее всего это связано с тем, что часть пользователей перескакивает через шаг просмотр корзины и идет по короткому пути оформления заказа. Возможно у них небольшое количество товаров или малая сумма заказа.
- Следовательно переход на этап **product_cart** не является обязательным, тем более что проверка конверсии его перехода в **purchase** не показательна.
- Конверсия на шаге **login - product_page** составляет **63%.**
- Конверсия на шаге **product_page - purchase** составляет **49%.**

Конверсия в воронке на разных этапах для каждой из групп.

Подсчет количества событий для каждого этапа в разных группах эксперимента.

In [20]:
```
# методу pivot_table() передаются данные 'event_name' в качестве индексов
ab_stages=events_eu_ab.pivot_table(
    index='event_name',
    columns='group',
    values='user_id',
    aggfunc='count').sort_values(by='A', ascending=False).reset_index()
# переименование столбца
ab_stages.columns=('event_name', 'A_count_events', 'B_count_events')
# добавление столбца с долей событий на каждом этапе от общего их числа.
ab_stages['A_rate_%'] = ((ab_stages['A_count_events']/ab_stages['A_count_events'].sum())*100).round(1)
ab_stages['B_rate_%'] = ((ab_stages['B_count_events']/ab_stages['B_count_events'].sum())*100).round(1)
ab_stages
```

Out[20]:

	event_name	A_count_events	B_count_events	A_rate_%	B_rate_%
0	login	6083	1823	44.3	48.2
1	product_page	3952	970	28.8	25.6
2	purchase	1854	494	13.5	13.1
3	product_cart	1853	497	13.5	13.1

Подсчет количества уникальных пользователей для каждого этапа в разных группах эксперимента.

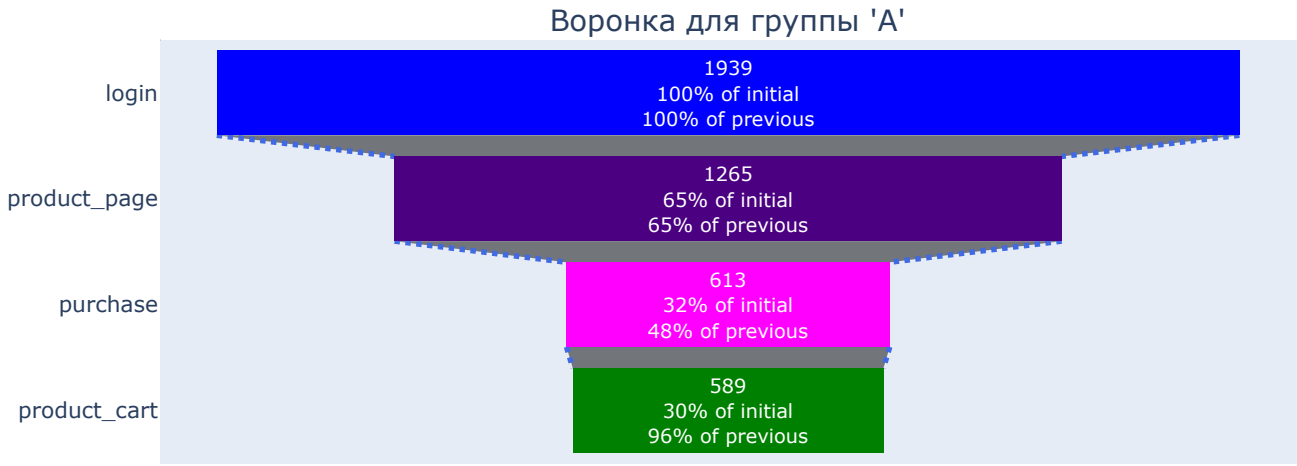
In [21]:
```
# методу pivot_table() передаются данные 'event_name' в качестве индексов
ab_funnel=events_eu_ab.pivot_table(
    index='event_name',
    columns='group',
    values='user_id',
    aggfunc='nunique').sort_values(by='A', ascending=False).reset_index()
# переименование столбца
ab_funnel.columns=('event_name', 'A_count_users', 'B_count_users')
# добавление столбца с долей участников на каждом этапе от общего их числа.
ab_funnel['A_rate_%'] = ((ab_funnel['A_count_users']/events_eu_ab.query('group == "A"')['user_id'].nunique())*100).round(1)
ab_funnel['B_rate_%'] = ((ab_funnel['B_count_users']/events_eu_ab.query('group == "B"')['user_id'].nunique())*100).round(1)
ab_funnel
```

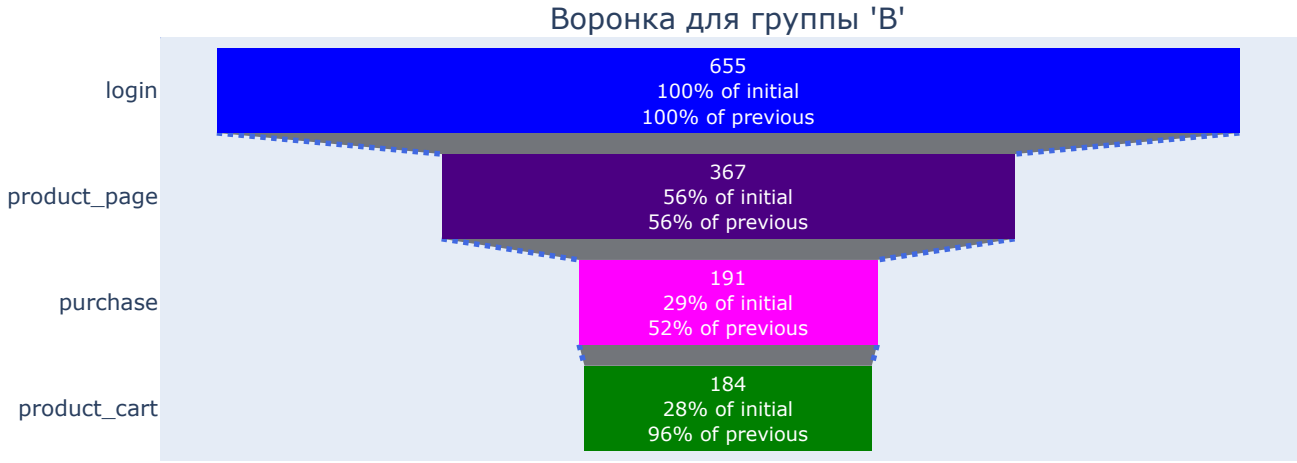
Out[21]:

	event_name	A_count_users	B_count_users	A_rate_%	B_rate_%
0	login	1939	655	100.0	100.0
1	product_page	1265	367	65.2	56.0
2	purchase	613	191	31.6	29.2
3	product_cart	589	184	30.4	28.1

In [22]:
```
#  Построение матрицы
fig = make_subplots(rows=2, cols=1, subplot_titles=("Воронка для группы 'A'", "Воронка для группы 'B'"))
# вывод воронок при помощи go.Funnel из Plotly
# для "A"
fig.add_trace(go.Funnel(
    y = ab_funnel['event_name'], marker={'color': ['blue', 'indigo', 'fuchsia', 'green']},
    textinfo='percent previous+percent initial+value',
    x = ab_funnel['A_count_users'],
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}
    ), row=1, col=1)
# для "B"
fig.add_trace(go.Funnel(
    y = ab_funnel['event_name'], marker={'color': ['blue', 'indigo', 'fuchsia', 'green']},
    textinfo='percent previous+percent initial+value',
    x = ab_funnel['B_count_users'],
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}
    ), row=2, col=1)
fig.update_layout(height=800, width=800, showlegend=False,
                  title_text="Воронки событий для двух групп эксперимента.")
fig.show()
```

Воронки событий для двух групп эксперимента.





ВЫВОД:

- Распределение пропорций количества событий по отдельности в группах схоже с распределением отмеченным в общей массе.
- По количеству событий и уникальных пользователей выделяется:
  - login - можно рассматривать как первое событие.
  - product_page - второе событие.
  - далее по логике должно идти product_cart, но количество событий и пользователей в purchase перевешивает.
- Пропорции количества событий на каждом этаме между груммами отличаются незначительно, но заметно что увеличение коснулось только доли события "login" (**на 4%**), тогда как на других этапах в группе "B" отмечается снижение доли событий.
- Доля количества пользователей переходящих на каждый этап в группе "B" также ниже чем в "A".
- По воронкам видно, что большие чем в группе "A" потери в группе "B" происходят на переходе к событию "product_page". Конверсия **снизилась на 9%.**
- При этом конверсия при переходе **product_page - purchase** повысилась **на 4%.**

Обладают ли выборки одинаковыми распределениями количества событий на пользователя?

In [23]:
```
# подсчет уникальных пользователей и количества событий в 2-х группах методом groupby
ab_group=events_eu_ab.groupby('group').agg({'user_id': ('nunique'), 'region': ('count')})
# переименование столбцов
ab_group.rename(columns = {'user_id':'count_user', 'region': 'count_event'}, inplace = True)
# добавление данных о среднем количестве событий на пользователя.
ab_group['events_per_user']=(ab_group['count_event']/ab_group['count_user']).round(1)
ab_group.T
```

Out[23]:

group	A	B
count_user	1939.0	655.0
count_event	13742.0	3784.0
events_per_user	7.1	5.8

ВЫВОД:

- Количество уникальных пользователей и событий в группе **B** почти в 3 раза меньше чем в **A**.
- В группе **B** на одного пользователя приходится меньше событий чем в **A** (5,8 вместо 7,1)

Присутствуют ли в выборках одни и те же пользователи?

Для проверки наличия одних и тех же пользователей в выборках следует обратиться к таблице `test_participes` , которая содержит признак разделения на группы и которая использовалась для выделения участников теста "recommender_system_test" в объединеной таблице.

In [24]:
```
# проверка наличия задвоенных "user_id" в выборках методом duplicated().sum()
print('Количество дублированных "user_id":',
      test_participes['user_id'].duplicated().sum())
```

```
Количество дублированных "user_id": 0
```

ВЫВОД:

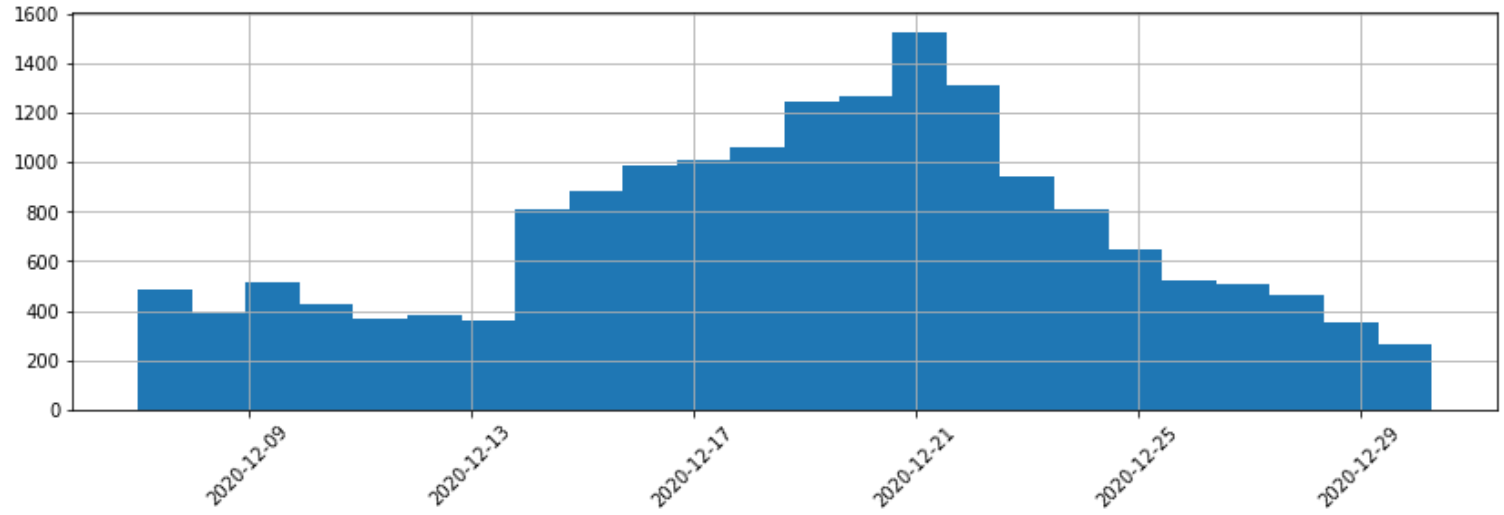
- Проверка не выявила дубликатов по 'user_id' в группах среди участников анализируемого теста 'recommender_system_test'.

Как число событий распределено по дням?

Распределение общего числа событий.

In [25]:
```
# построение столбчатой диаграммы распределения
events_eu_ab['event_dt'].hist(bins=24, figsize=(14,4), xrot=45)
```

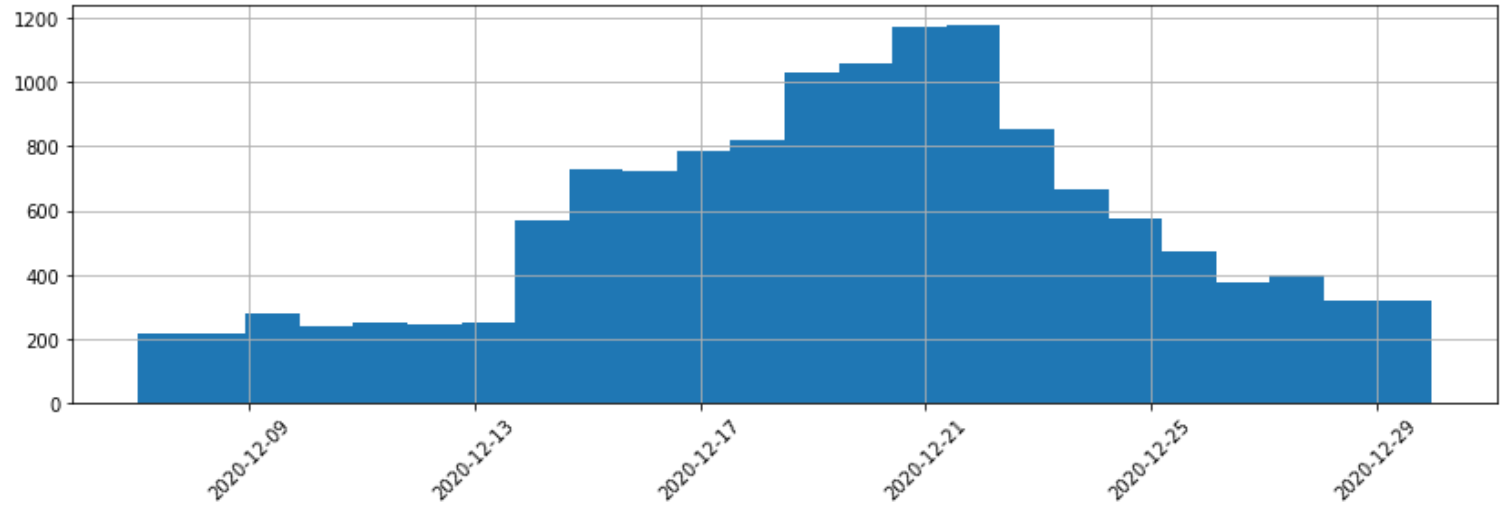
Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f473228db90>
```



Распределение по дням событий в группе "A".

In [26]:
```
# построение столбчатой диаграммы распределения
events_eu_ab.query('group == "A"')['event_dt'].hist(bins=24, figsize=(14,4), xrot=45)
```

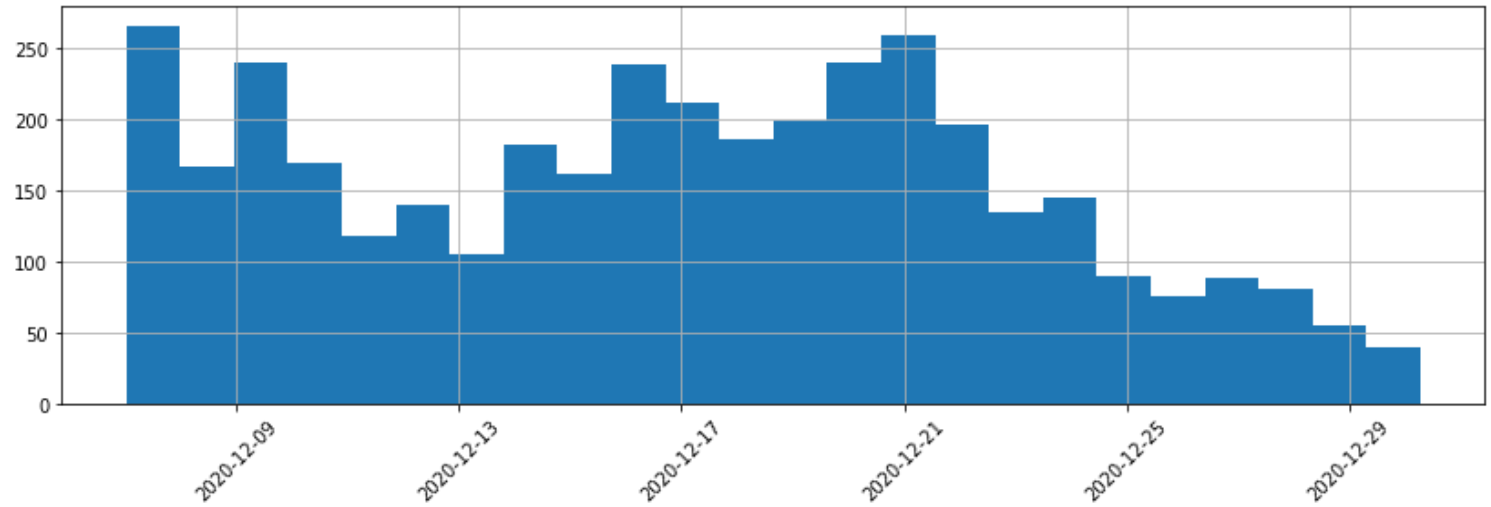
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f473117f450>



Распределение по дням событий в группе "B".

In [27]:
```
# построение столбчатой диаграммы распределения
events_eu_ab.query('group == "B"')['event_dt'].hist(bins=24, figsize=(14,4), xrot=45)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47310c1b50>



ВЫВОД:

- Количество событий в обеих группах эксперимента распределено по-разному.
- для группы "A" характерно следующее:
  - События распределены неравномерно по дням.
  - Первую неделю количество событий колеблется в коридоре 220-280 событий.
  - Во вторую неделю, с 14.12.2020 число событий начинает резко расти вплоть до 21.12.2020 (с 250 событий до 1200).
  - После чего количество событий резко снижается до 300 к 30.12.2020.
- для группы "B" характерно следующее:
  - первая неделя - нисходящий тренд, с 270 до 100 событий в день.
  - вторая неделя - восходящий тренд, со 100 до 260 событий в день.
  - после 21 декабря - спад, до 40 событий к 30 декабря.
- заметно, что обе группы начинают со схожих показателей по количеству событий. 220-270 событий - 7 декабря.
- Рост в обеих группах начинается с 14 декабря, но в группе "A" он более динамичный. Количество событий к 21 декабря увеличилось почти в 5 раз, тогда как в группе "B" только в 2,6 раза.
- Это связано с окончанием срока набора новых пользователей после 21 декабря.

Нюансы данных, которые нужно учесть, прежде чем приступать к A/B-тестированию?

Важный нюанс перед проведением A/B тестирования - исключить возможность влияния на собираемые для теста данные со стороны проводимых компанией маркетинговых акций.

Какие маркетинговые активности проводились в 2020 году в регионе EU?

In [28]:
```
# выборка мероприятий с признаком "EU" в "regions" методом str.contains()
events_calender_eu=events_calender[events_calender['regions'].str.contains('EU')]
events_calender_eu
```

Out[28]:

	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
5	Black Friday Ads Campaign	EU, CIS, APAC, N.America	2020-11-26	2020-12-01
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

Определение дат начала теста и окончания набора новых участников теста согласно технического задания.

In [29]:
```
# формат даты значений нужен для сравнения дат
start_test = pd.to_datetime('2020-12-07')
stop_new_user= pd.to_datetime('2020-12-21')
```

Запуск цикла проверки пересечения дат теста и дат проведения мероприятий.

In [30]:
```
# Вывод заголовка для результатов проверки
print('Результат проверки пересечения дат теста и дат проведения мероприятий:')
print('')
# цикл с условиями
for start, finish, name in zip(events_calender_eu['start_dt'],
                           events_calender_eu['finish_dt'],
                           events_calender_eu['name']):
    if stop_new_user >= start >= start_test or stop_new_user >= finish >= start_test:
        print("Есть пересечения с", name)
    else:
        print("Нет пересечений с", name)
```

```
Результат проверки пересечения дат теста и дат проведения мероприятий:

Нет пересечений с Christmas&New Year Promo
Нет пересечений с St. Valentine's Day Giveaway
Нет пересечений с St. Patric's Day Promo
Нет пересечений с Easter Promo
Нет пересечений с Black Friday Ads Campaign
Нет пересечений с Labor day (May 1st) Ads Campaign
Нет пересечений с International Women's Day Promo
```

ВЫВОД:

Пересечений с маркетинговыми активностями нет.

Визуально в таблице `events_calender_eu` заметно,

что все активности проходили или до запуска теста 7 декабря 2020

или уже после окончания набора новых пользователей 21 декабря 2020 года.

Так, например: "Christmas&New Year Promo", старт 25 декабря.

3. Оценка результатов A/B-тестирования.

Для начала необходимо свести данные по событиям с подсчетом уникальных пользователей в каждом эксперименте.

In [31]:
```
# методу pivot_table() передаются данные 'event_name' в качестве индексов
ab=events_eu_ab.pivot_table(
    index='event_name',
    columns='group',
    values='user_id',
    aggfunc='nunique').sort_values(by='A', ascending=False).reset_index()
# добавление строки с количеством уникальных пользователей в каждой группе
ab.loc[4]=ab_group.T.reset_index().loc[0]
# изменение обозначения в ячейке
ab.loc[4, 'event_name'] = 'total_unique_user'
ab['ratio B/A'] = ab['B']/ab['A']
ab['total'] = ab['A'] + ab['B']
ab# выведем полученный результат
```

Out[31]:

group	event_name	A	B	ratio B/A	total
0	login	1939.0	655.0	0.337803	2594.0
1	product_page	1265.0	367.0	0.290119	1632.0
2	purchase	613.0	191.0	0.311582	804.0
3	product_cart	589.0	184.0	0.312394	773.0
4	total_unique_user	1939.0	655.0	0.337803	2594.0

ВЫВОД:

- Общее число уникальных пользователей A/B теста составляет **2594**. Что более чем в 2 раза меньше ожидаемого.
- Количество участников группы **B** составляет **одну третью часть** от числа участников группы **A**.

Формулировка гипотез.

Нулевая гипотеза H_0: **Доли уникальных польтзователей, совершивших одно и тоже событие в 2-х разных выборках статистически одинаковы.**
Альтернативная гипотеза H_1: **Доли уникальных польтзователей, совершивших одно и тоже событие в 2-х разных выборках различаются.**

Функция для проверки гипотезы.

In [32]:
```python
# напишем функцию для проверки гипотезы о равенстве долей
def check_ab_test(part, alpha, ex1, ex2):

    alpha = alpha # критический уровень статистической значимости

    #количество уникальных пользователей по выборкам
    total1=ab.loc[4, ex1]
    total2=ab.loc[4, ex2]

    # доля пользователей в первой группе:
    p1 = part[0]/total1

    # доля пользователей успехов во второй группе:
    p2 = part[1]/total2

    # доля пользователей в комбинированном датасете:
    p_combined = (part[0] + part[1]) / (total1 + total2)

    # 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/total1 + 1/total2))

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

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

    if (p_value < alpha):
        print('p_value=', p_value.round(5), "Отвергаем нулевую гипотезу: между долями есть значимая разница")
    else:
        print('p_value=', p_value.round(5), "Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными")
```

Цикл с функцией для проверки гипотезы о равенстве долей.

In [33]:
```
# заданные параметры теста
alpha=.05 # критический уровень статистической значимости
group1="A" # указание на столбец с данными для группы "A"
group2="B" # указание на столбец с данными для группы "B"
# цикл перебирает строки таблицы "ab", задает переменную part и выводит результат функции
for i in range(len(ab)-1):
    part=np.array(ab[[group1, group2]].iloc[i])
    print('Сравнение долей для этапа: ', ab.iloc[i, 0])
    check_ab_test(part, alpha, group1, group2)
    print('')
```

```
Сравнение долей для этапа:  login
p_value= nan Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравнение долей для этапа:  product_page
p_value= 2e-05 Отвергаем нулевую гипотезу: между долями есть значимая разница

Сравнение долей для этапа:  purchase
p_value= 0.24036 Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравнение долей для этапа:  product_cart
p_value= 0.26899 Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными


/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:22: RuntimeWarning:

invalid value encountered in double_scalars
```

ВЫВОД:

- Тест показал наличие ошибки при сравнении долей на этапе **"login"**, так как в данном случае количество вошедших в систему уникальных пользователей равно общему количеству уникальных пользователей в группе. Т.е. перейти на любой другой этап без прохождения **"login"** невозможно.
- Ррозличие долей тест фиксирует только на этапе **"product_page"**.
- Для самого важного этапа **"purchase"** нет причин говорить о разной доли уникальных пользователей в обеих группах теста.
- Также нет различий в обеих группах между долями **"product_cart"**.

4. Общие выводы:

- Тест проведен в соответствии со сроками, указанными в техническом задании.
- Пересечений по срокам с периодами проведения маркетинговых активностей не выявлено.
- В сроки проведения анализируемого теста шел параллельно в те же даты _interface_eu*test*, но в данном исследовании не обнаружено возможное его влияние на результаты анализируемого теста.
- Обнаружено неравномерное распределение событий во время теста.
- Распределение событий в группах теста не совпадает. Возможно предположить, что неучтенные мероприятия, начавшиеся 14 декабря, в меньшей степени повлияли на рост количества событий только в группе **"B"** в следствии того, что пользователи из группы "A" в большей степени участвовали в этих мероприятиях. А значит разделение на группы могло произойти некорректно.
- В **A/B тесте** приняли участие **2594** пользователя, вместо ожидаемых 6000. Этот факт также мог оказать влияние на результат.
- Выделено 4 этапа: **"login"**, **"product_page"**, **"purchase"** и **"product_cart"**.
- Логика продаж позволяет пропускать этап просмотра корзины **"product_cart"** и сразу переходить к покупке **"purchase"**. Что подтверждается воронкой по числу уникальных пользователей.
- Увеличение конверсии переходов в группе **B** на 10% в сравнении с группой **"A"** не выявлено. Вместо этого зафиксировано существенное снижение на **9%** конверсии при переходе пользователей на этап **"product_page"** и повышение на **4%** при переходе с **"product_page"** на **"purchase"**.
- Таким образом установлено, что проводимые в рамках теста изменения, связанные с внедрением улучшенной рекомендательной системы, оказывают влияние на конверсию переходов. Но это влияние не отражает ожидаемого результата.
- Анализ **A/B теста**, оценивающий доли уникальных пользователей между тестовыми группами, показал, что основания считать разными имеются только для долей на этапе **"product_page"**. Пропорции на остальных этапах схожие. Оценка теста так же, как и изужчение конверсий переходов, показывает отсутствие значимого влияния тестируемых изменений на результаты продаж.
- Причиной невнятного результата теста могли стать:
  - количество участников ниже ожидаемого, в связи с неправильным расчетом сроков проведения теста, следовало увеличить срок как минимум вдвое.
  - влияние неучтенных факторов, которые повлияли на рост количества событий в середине теста.
  - разделение на группы произошло неравномерно, возможно из-за ошибки при подготовке к **A/B тесту**.

In [ ]: