Table of Contents

Анализ результатов A/B тестирования

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

Необходимо:

- оценить корректность проведения теста;
- проанализировать результаты теста.

Чтобы оценить корректность проведения теста, нужно проверить:

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

Техническое задание и описание данных

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: 6 000;

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

Описание данных

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

Структура файла:

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

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

Структура файла:

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

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

Структура файла:

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

`final_ab_participants.csv` — таблица участников тестов.

Структура файла:

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

Загрузка данных

In [1]:

```
# импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import datetime as dt
```

```
from datetime import datetime, timedelta

from scipy import stats as st
import math as mth

import plotly
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
```

In [2]:

```
try:
    user_events = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_events.csv')
    pm_events = pd.read_csv('https://code.s3.yandex.net/datasets/ab_project_marketing_ever
    new_users = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_new_users.csv')
    participants = pd.read_csv('https://code.s3.yandex.net/datasets/final_ab_participants.
except:
    user_events = pd.read_csv('final_ab_events.csv')
    pm_events = pd.read_csv('final_ab_marketing_events.csv')
    new_users = pd.read_csv('final_ab_new_users.csv')
    participants = pd.read_csv('final_ab_participants.csv')
```

In [3]:

```
datasets_list = [user_events, pm_events, new_users, participants]
name_list = ['user_events', 'pm_events', 'new_users', 'participants']
```

In [4]:

```
for df, name  in zip (datasets_list, name_list):
    print()
    print ('\033[1m'+'Общая информация о DataFrame', name)
    display (df.head())
    df.info()
    print ()
    print ('Количество дубликатов = ', df.duplicated().sum())
    print ('Количество строк/столбцов -> ', df.shape)
    print ()
```

Общая информация о DataFrame user_events

	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):
 #   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

Количество дубликатов =  0
Количество строк/столбцов ->  (440317, 4)
```

Общая информация о DataFrame pm_events

	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

```
<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

Количество дубликатов =  0
Количество строк/столбцов ->  (14, 4)
```

Общая информация о DataFrame new_users

	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):
 #   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

Количество дубликатов =  0
Количество строк/столбцов ->  (61733, 4)
```

Общая информация о DataFrame participants

	user_id	group	ab_test
0	D1ABA3E2887B6A73	A	recommender_system_test
1	A7A3664BD6242119	A	recommender_system_test

	user_id	group	ab_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):
 #   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

Количество дубликатов =  0
Количество строк/столбцов ->  (18268, 3)
```

Оценим количество пропусков в `events['details']`

In [5]:
```
print(round (1 - user_events['details'].count()/user_events.shape[0], 2))
```

```
0.86
```

In [6]:
```
user_events.groupby('event_name')['details'].count()
```

Out[6]:
```
event_name
login              0
product_cart       0
product_page       0
purchase       62740
Name: details, dtype: int64
```

В таблице `events` имеем большое количество пропусков (порядка 86%) в столбце `details`. Данное поле несет в себе информацию о стоимости покупки и заполнено только для события `purchase`. Т.е. пропуски абсолютно естественны.

Таким образом пропуски можно классифицировать `MNAR (Missing Not At Random / Отсутствует не случайно)` — пропуски зависят от данных, их нельзя отбрасывать, т.к. это приведёт к заметным искажениям

Для дальнейшей работы потребуется изменение типов данных в следующих столбцах:

- `pm_events['start_dt']`, `pm_events['finish_dt']`, `new_users['first_date']` к типу данных `datetime64[D]`
- в таблице `user_events` создадим столбец `event_dt_time`, а в стобце `event_dt` оставим только дату.

In [7]:
```
# преобразование типов данных
pm_events['start_dt'] = pm_events['start_dt'].astype('datetime64[D]')
pm_events['finish_dt'] = pm_events['finish_dt'].astype('datetime64[D]')

new_users['first_date'] = new_users['first_date'].astype('datetime64[D]')
```

```
user_events['event_dt_time'] = pd.to_datetime(user_events['event_dt'], format='%Y-%m-%d %H
user_events['event_dt'] = user_events['event_dt'].astype('datetime64[D]')
```

Вывод

На данном шаге познакомились с данными:

- дубликаты отсутсвуют;
- пропуски есть только в столбце `events['details']` , но носят естественный характер (не было покупки, значит нет и стоимости)
- провели преобразование типов данных, в столбцах, где это потребовалось.

Оценка корректности теста

Проверка корректности ТЗ

Проверим корректность ТЗ, есть ли у нас в наличии данные для того, чтобы его выполнить

In [8]:
```
# считаем количество записей по каждому тесту
participants.groupby('ab_test')['user_id'].count()
```

Out[8]:
```
ab_test
interface_eu_test           11567
recommender_system_test      6701
Name: user_id, dtype: int64
```

In [9]:
```
# считаем количество записей для групп A и B для теста 'recommender_system_test'
participants.query("ab_test == 'recommender_system_test'").groupby('group')['user_id'].cou
```

Out[9]:
```
group
A    3824
B    2877
Name: user_id, dtype: int64
```

Данные об искомом тесте `recommender_system_test` присутствуют 6 701 запись, разделение на группы в тесте также есть.

Далее проверим даты записей в таблице с данными о новых пользователях.

In [10]:
```
print (new_users['first_date'].min())
print (new_users['first_date'].max())
```

```
2020-12-07 00:00:00
2020-12-23 00:00:00
```

Даты регистрации в таблице с новыми пользователями соответствуют ТЗ (чуть больше на два дня, но тем не менее)

In [11]:
```
print (user_events['event_dt'].min())
print (user_events['event_dt'].max())
```

```
2020-12-07 00:00:00
2020-12-30 00:00:00
```

Записи в таблице с событиями завершаются датой 30-12-2020. Это не хорошо, ведь у нас не все пользователи будут иметь возможность совершать события в первые 14 дней после регистрации. Теоретически это может негативно сказаться на результатах теста.

In [12]:
```
print ("Количество новых пользователей из Европы соствляет",len (new_users.query("region =
print ("Всего новых пользователей", len(new_users))
print ("15% новых пользователей из региона EU", 15*len(new_users.query("region == 'EU'")),
```

```
Количество новых пользователей из Европы соствляет 46270
Всего новых пользователей 61733
15% новых пользователей из региона EU 6940.5
```

Количество новых пользователей из Европы достаточно для соответствия ТЗ (более 6000)

Более подробно оценим данные в следующих шагах пока можно предварительно отметить:

В целом необходимые данные присутствуют, но не в идеальном виде. В частности, недостает записей о событиях после 30.12.2020, . Плюс мы еще не проверили возможные пересечения групп, где мы можем потерять еще записи. А после этого нужно проверить сколько пользователей реально участвовала в тесте.

Время проведения

Проверим время и географию проведения маркетинговых компаний и сравним со временем проведения нашего теста. По условию задачи набор новых пользователей проводился с 07.12.2020 по 21.12.2020 и ориентирован на регион EU, окончен тест был 04.01.2021.

Можно отметить не самое удачное время проведения теста в предверии новогодних праздников, т.к. в это время покупательская активность сезонно увеличивается.

In [13]:
```
# сравниваем периоды маркетинговых активностей со временем набора пользователей
pm_events.query("start_dt>datetime(2020,12,7) & start_dt<datetime(2020,12,21)")
```

Out[13]:

name	regions	start_dt	finish_dt

In [14]:
```
# сравниваем периоды маркетинговых активностей с общим периодом проведения теста.
pm_events.query("start_dt>datetime(2020,12,7) & start_dt<datetime(2021,1,4)")
```

Out[14]:

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

Во время набора новых пользователей не было пересечения с маркетинговыми акциями. Однако до завершения теста имеем пересечение с одной промоакцией `Christmas&New Year Promo` , проводимой в том числе в Европе с 25 декабря 2020 по 3 января 2021.

Такого рода акция могла существенно повлиять на решении совершить покупку клиентом и на остальные ключевые метрики конверсии.

Не станем избавляться от данных за это время, но и не будем забывать об этой накладке

Проверка аудитории теста

Проверим нет ли у нас пользователей попавших в оба теста одновременно.

In [15]:
```
# отбираем группы
group_a=participants.query("ab_test == 'recommender_system_test'")
group_b=participants.query("ab_test == 'interface_eu_test'")
```

```
# получаем id пользователей, попавших в обе группы
group_intersections=list(np.intersect1d(group_a['user_id'], group_b['user_id']))
print ('Количество пользователей, попавших в обе группы = ', len(group_intersections))
print ('Доля пользователей, попавших в обе группы = ', round (len(group_intersections)/par
```

```
Количество пользователей, попавших в обе группы =  1602
Доля пользователей, попавших в обе группы =  0.09
```

Получаем 1 602 пользователя участвовавших в обоих тестах. Сказать какой из тестов повлиял на их решение мы не можем, поэтому оставлять информацию о них в данных нельзя, это может исказить результаты.

In [16]:
```
# удаляем данные о пользователях попавших в оба теста
participants = participants.query('user_id not in @group_intersections')
```

Теперь проверим нет ли пользователей, участвующих в двух группах одновременно

In [17]:
```
# отбираем группы
group_a=participants.query('group == "A"')
group_b=participants.query('group == "B"')

# получаем id пользователей, попавших в обе группы
group_intersections=list(np.intersect1d(group_a['user_id'], group_b['user_id']))
print ('Количество пользователей, попавших в обе группы = ', len(group_intersections))
print ('Доля пользователей, попавших в обе группы = ', round (len(group_intersections)/par
```

```
Количество пользователей, попавших в обе группы =  0
Доля пользователей, попавших в обе группы =  0.0
```

Отлично, таких пересечений не выявлено.

Также уберем из данных записи о `interface_eu_test` , т.к. для нашей работы они нам не нужны.

In [18]:
```
participants = participants.query("ab_test == 'recommender_system_test'")
len (participants)
```

Out[18]:
```
5099
```

Остались записи только 5 099 пользователях.

Проверим сколько пользователей у нас попали в каждую из групп.

In [19]:
```
print ('Количество пользователей в группе A = ', len(participants.query('group == "A"')))
print ('Количество пользователей в группе B = ', len(participants.query('group == "B"')))
print ('Доля пользователей группы A  = ', round(len(participants.query('group == "A"')) /
print ('Доля пользователей группы B  = ', round(len(participants.query('group == "B"')) /
```

```
Количество пользователей в группе A =  2903
Количество пользователей в группе B =  2196
Доля пользователей группы A  =  0.57
Доля пользователей группы B  =  0.43
```

Имеется перекос по разделению пользователей на группы. В группе А пользователей больше. В идеале пользователи должны быть разделены по группам поровну. Различие составляет около 7% от общего размера данных.

Далее посмотрим сколько пользователей все - таки участвовали в тесте, т.е. совершали какие-либо действия.

In [20]:
```
# объединяем таблицы с участниками теста и событиями
```

```
participants_events = user_events.merge(participants, how='left', on='user_id').dropna(axi
```

In [21]:
```
# смотрим сколько уникальных пользователей совершали действия
active_users = participants_events['user_id'].nunique()
active_users
```

Out[21]:
```
2788
```

Всего 2 788, т.е. гораздо меньше чем ожидалось в ТЗ.

Теперь итоговая проверка разделения на группы, в которой будут только те пользователи, которые участвовали в тесте.

In [22]:
```
df_count_group = participants_events.groupby('group', as_index=False).agg({'user_id':'nuni
part_users_A = round (participants_events.query('group == "A"')['user_id'].nunique() / act
part_users_B = 1-part_users_A
df_count_group['part']  = [part_users_A*100, part_users_B*100]
df_count_group['part'] = df_count_group['part'].astype('str')+'%'
df_count_group
```

Out[22]:

	group	user_id	part
0	A	2082	75.0%
1	B	706	25.0%

Если посмотреть на пользователей групп A и B которые реально участвовали в тесте (совершали какие-либо действия), то перекос в сторону группы A становится уже огромным 75% на 25%.

Вывод

На данном шаге проверили корректность теста. Были выявлены пересечения с другим тестом. Записи о пользователях попавших в оба теста пришло удалить.

Кроме этого, при проверке данных о тесте представленных для анализа выявлены серьезные проблемы, в частности:

- время и география проведения теста пересекается с маркетинговой акцией `Christmas&New Year Promo` ;
- количество участников теста более чем в два раза меньше чем указано в ТЗ;
- пользователи разделены по группам A/B теста в соотношении 75%/25% в пользу группы A.

Таким образом, дальнейший анализ результатов теста нужно воспринимать скептически. Делать из него серьезных выводов нельзя.

Исследовательский анализ данных EDA

Распределение количества событий на пользователя

Посмотрим сколько в среднем событий приходится на одного пользователя в каждой из групп

In [23]:
```
print ('Усредненное количество событий на пользователя в группе A = ', \
    round (participants_events.query('group == "A"')['event_name'].count() / participants_
```

```
Усредненное количество событий на пользователя в группе A =  7.0
```

In [24]:
```
print ('Усредненное количество событий на пользователя в группе B = ', \
    round (participants_events.query('group == "B"')['event_name'].count() / participants_
```

```
Усредненное количество событий на пользователя в группе B =  6.0
```

События распределены схоже но неравномерно, т.к. в группе А гораздо больше пользователей

Распределение числа событий по дням

Посмотрим на распределение числа событий по дням для каждой из групп.

In [25]:
```
# создаем DataFrame event_by_date, на основнаии которого будем строить гарфик
event_by_date = participants_events.groupby(['event_dt', 'group'])['user_id'].count().rese
```

In [26]:
```
# строим график
plt.figure(figsize=(15, 4))
sns.set_style("whitegrid")
ax = sns.lineplot(data=event_by_date, x='event_dt', y='user_id', hue='group')
ax.set_xticks(event_by_date['event_dt'])
ax.set_title('Распределение количества событий в группах по дням')
plt.xticks(rotation=45);
```

- наблюдается резкий рост числа событий в группе А в период с 13 по 21 декабря;
- 30 декабря количество событий в группе В падает до нуля, а по группе А вообще нет информации.

In [27]:
```
# соберем все необходимые для дальнейшей работы данные в одну таблицу
df_all_data = participants_events.merge(new_users, how='left', on='user_id').dropna(axis='
```

Чтобы попробовать объяснить этот всплеск, посмотрим на количество пользователей зарегестрировавшихся и попавших в группу А в период с 13 по 21 декабря

In [28]:
```
df_all_data.query("first_date > datetime(2020, 12, 13) & first_date < datetime(2020, 12, 2
```

Out[28]:
```
1311
```

Получается 1311 пользователей из 2082 зарегистрировались в этот промежуток времени. Т.е. больше половины. Похоже резкий всплеск связан с этим.

Если бы в этот промежуток времени пользователи распределялись между группами более равномерно, то у нас бы не было такого перекоса между группами.

Кроме того, что - то не так с данными за 29-30 декабря. Либо тест по какой - то причине закончили раньше, либо имеет место сбой при сборе данных.

Явно имеем технические проблемы при распределении пользователей и вероятно при сборе информации

Изменение конверсии в воронке

На этом шаге оценим сколько пользователей "терялись" при переходе от одного шага к другому. Вначале для всех, а затем в разрезе по группам.

In [29]:
```
df_total_users = df_all_data.groupby('event_name')['user_id'].nunique().sort_values(ascend
        .rename(columns={'user_id': 'total_users'})


total_users = df_all_data['user_id'].nunique()

df_total_users['percent'] = round ((df_total_users['total_users'] / total_users*100),2)
df_total_users['percent'] = df_total_users['percent'].astype('str')+'%'
df_total_users
```

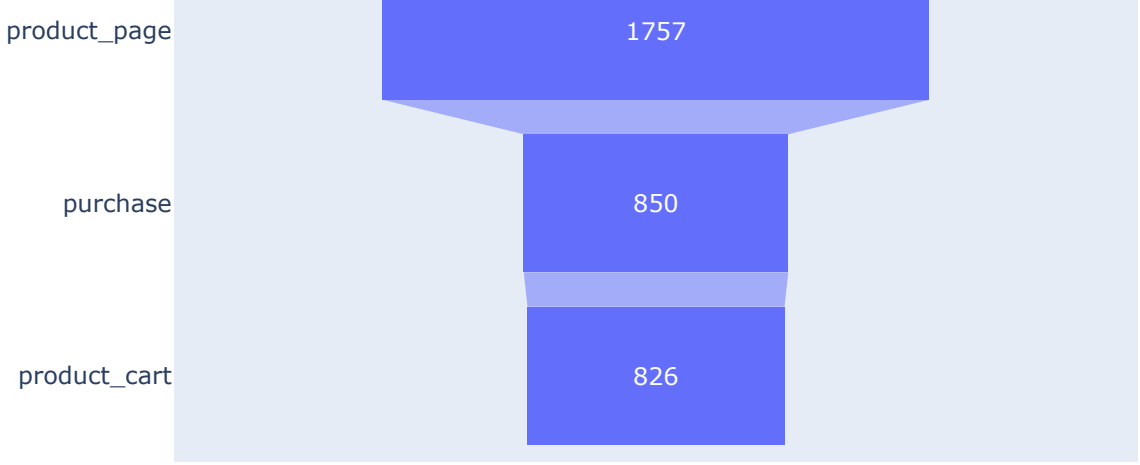
Out[29]:

	event_name	total_users	percent
0	login	2788	100.0%
1	product_page	1757	63.02%
2	purchase	850	30.49%
3	product_cart	826	29.63%

Только 63 % от всех залогинившихся пользователей посмотрели страницу продукта и около 30% совершили покупку. При этом судя по данным для покупки вовсе не обязательно добавлять товар в корзину.

Визуализируем наши расчеты.

In [30]:
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'All Users',
    y = df_total_users['event_name'],
    x = df_total_users['total_users'],
    ))
fig.show()
```



Теперь построим гарфик для иллюстрации конверсии при переходе от одного шага к другому, но уже в разрезе групп A/B.

Для начала соберем информацию о количетсве пользователей дошедших до каждого из шагов в группах A/B

In [31]:
```
group_A = df_all_data.query("group == 'A'").groupby('event_name')['user_id'].nunique().sor
        .rename(columns={'user_id': 'count'})
group_B = df_all_data.query("group == 'B'").groupby('event_name')['user_id'].nunique().sor
        .rename(columns={'user_id': 'count'})
```

Количество пользователей на каждом шаге в табличном виде.

In [32]:
```
all_groups = df_all_data.pivot_table(index='event_name', columns='group',values='user_id',

all_groups['percent_A'] = round ((all_groups['A'] / df_count_group.loc[0, 'user_id']*100),
all_groups['percent_A'] = all_groups['percent_A'].astype('str')+'%'

all_groups['percent_B'] = round ((all_groups['B'] / df_count_group.loc[1, 'user_id']*100),
all_groups['percent_B'] = all_groups['percent_B'].astype('str')+'%'

all_groups.sort_values('A', ascending = False)
```

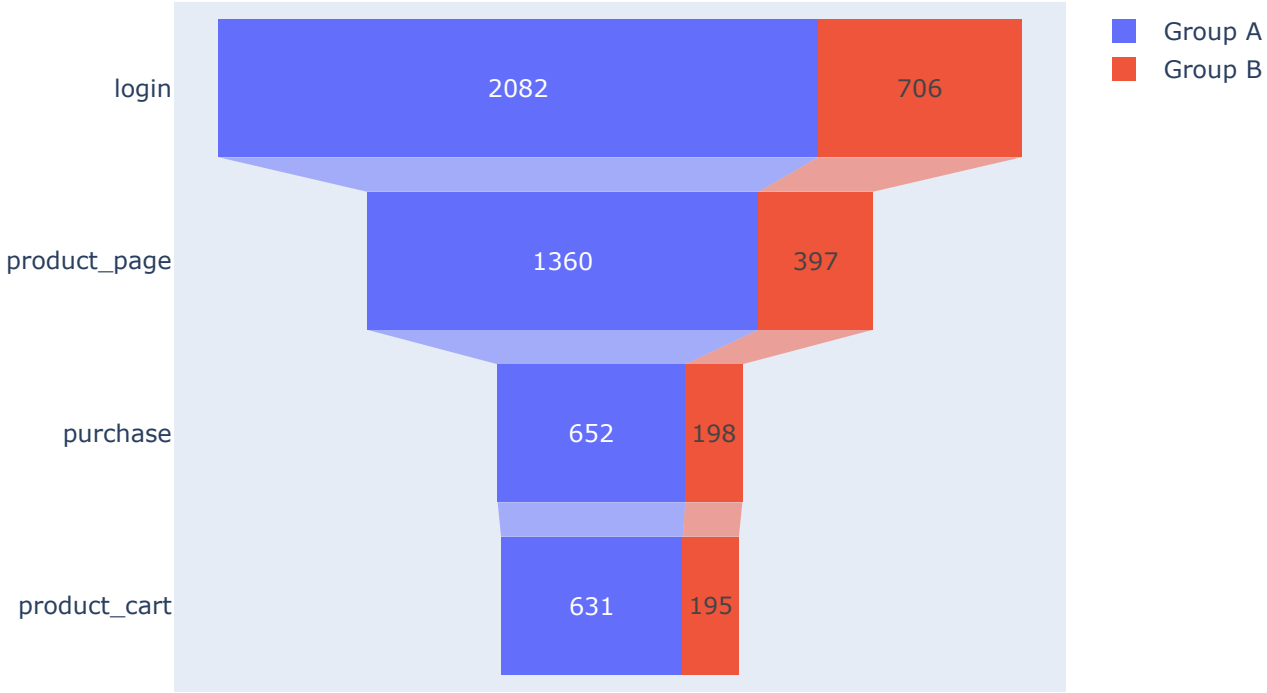
Out[32]:

group	A	B	percent_A	percent_B
event_name				
login	2082	706	100.0%	100.0%
product_page	1360	397	65.32%	56.23%
purchase	652	198	31.32%	28.05%
product_cart	631	195	30.31%	27.62%

Теперь переходим к графику.

In [33]:
```
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'Group A',
```

```
    y = group_A['event_name'],
    x = group_A['count'],
    ))

fig.add_trace(go.Funnel(
    name = 'Group B',
    y = group_B['event_name'],
    x = group_B['count'],
    ))



fig.show()
```



В первую очередь на "пирамиде" бросатеся в глаза неравномерное распределение пользователей между группами.

- немногим менее половины клиентов после авторизации не доходят до карточки товара;
- "корзина" и "покупка" поменялись местами, значит для осуществления покупки не обязательно добавлять товар в корзину;
- от просмотра страницы покупки на следующий шаг попадают также только около половины пользователей с предыдущего шага.

Вывод

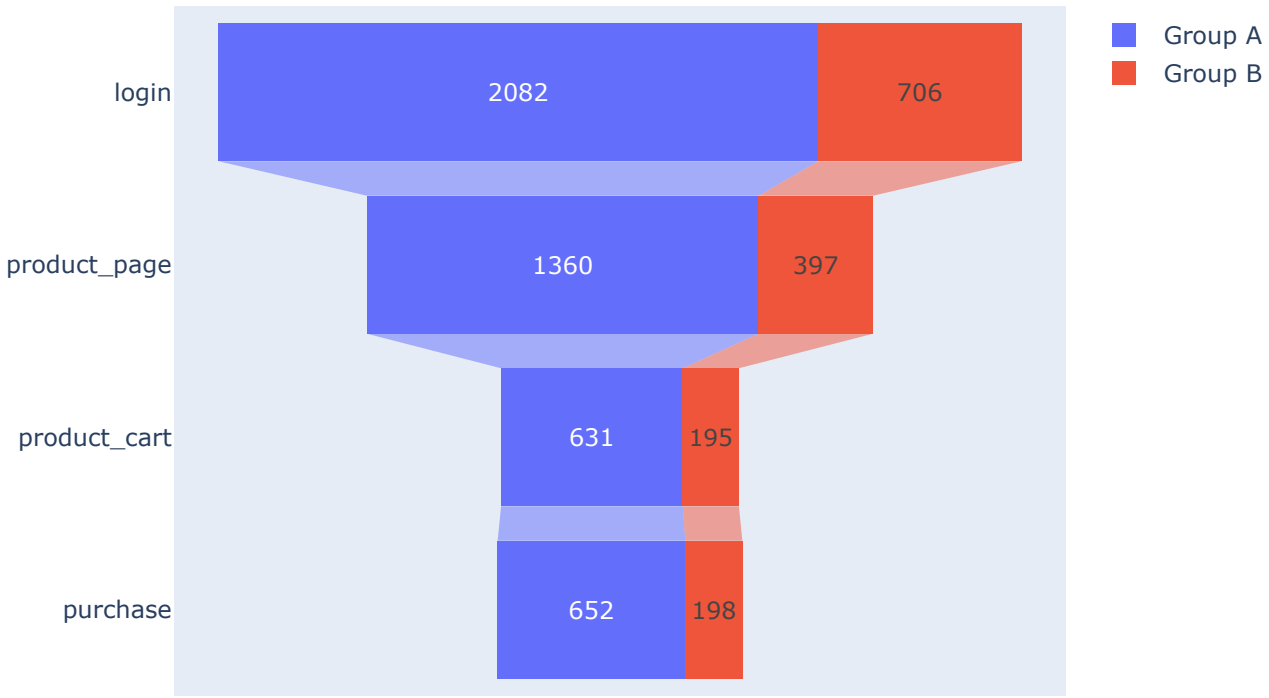
В целом воронка выглядит естественно.За исключением шагов "корзина" и "покупка". Вероятно, на сайте реализована возможность покупки без добавления в корзину, но при такой реализации мы не можем

однозначно выделить долю пользователей добавивших товар в корзину, но не совершивших покупку.

Также стоит отметить, что показатель конверсии оставляет желать лучшего, т.к. на каждом шаге у на отсеивается порядка половины пользователей.

In [34]:
```
# меняем строки местами
new_index = {2: 3, 3: 2}
group_A = group_A.rename(new_index).sort_index()
group_B = group_B.rename(new_index).sort_index()
```

In [35]:
```
# строим воронку
fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'Group A',
    y = group_A['event_name'],
    x = group_A['count'],
    ))

fig.add_trace(go.Funnel(
    name = 'Group B',
    y = group_B['event_name'],
    x = group_B['count'],
    ))



fig.show()
```



В таком представлении воронка действительно лучше иллюстрирует некорректность теста и в целом и в обоих гурппах. Анлизировать конверсию между `product_page -> product_cart`, `product_cart -> purchase` или даже `product_page -> purchase`, если уж корзина не обязательна, на этих данных невозможно. Т.к. нельзя однозначно сказать какое действие после product_page совершил пользователь.

Оценка результатов A/B тестирования

Конверсия

Изучим кумулятивную конверсию по значимым событиям для каждой из групп. Для это вначале опишем функции по сбору кумулятивных данных и построению графиков.

In [36]:
```
def cumulative_data(df, event):

    """
        Функция создает DataFrame с агрегированными кумулятивными данными о пользователях
        Таким образом при вызове функции получим DataFrame, где будут находиться такие дан
        как будто мы ежедневно считали результаты тестирования до выбранного дня включител
        их в строках таблиц.

    """
    # Выбираем из данных строки с заданным событием
    df_filtred_event = df.query('event_name == @event')
    # Создадим массив уникальных пар значений дат и групп теста
    datesGroups = df_filtred_event[['event_dt','group']].drop_duplicates()

    # Соберём агрегированные кумулятивные данные о покупках по дням
    df_purchase_agg = datesGroups.apply(lambda x: df_filtred_event[np.logical_and(df_filtr
                                                                                  df_filtred_ever
                                        .agg({'event_dt': 'max', 'group': 'max', 'user_id':
                                                                                  axis=1).sort_v

    # Соберём агрегированные кумулятивные данные о покупателях
    df_buyers_agg = datesGroups.apply(lambda x: df_filtred_event[np.logical_and(df_filtred
                                                                                df_filtred_event
                                      .agg({'event_dt': 'max', 'group': 'max', 'user_id': 'c
                                                                                  axis=1).sort_v

    # Объединим кумулятивные данные в одной таблице и переименуем столбцы
    result = df_purchase_agg.merge(df_buyers_agg, on=['event_dt','group'])
    result.columns = ['date', 'group', 'visitors', 'orders']
    return result
```

In [37]:
```
# Соберём кумулятивные значения для анализа относительных показателей и события "просмотр
cumulativeData_product_page = cumulative_data(df_all_data, 'product_page')

# добавляем столбец расчета конверсии
cumulativeData_product_page['conversion'] = cumulativeData_product_page['orders']/cumulati

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе A
cumulativeData_product_page_A = cumulativeData_product_page.query('group == "A"')

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B
cumulativeData_product_page_B = cumulativeData_product_page.query('group == "B"')
```

In [38]:
```
# Функция построения сравнительного графика
def plot_conversion(cumulativeData_A, cumulativeData_B, column, date, title, ylabel):
```

```
    """
        Функция для отображения кумулятивной конверсии в разбивке по группам

    """

    sns.set_style("whitegrid")
    fig, ax = plt.subplots(figsize=(18,6))
    ax.set(title=title, xlabel='дата', ylabel=ylabel)
    ax.plot(cumulativeData_A['date'], cumulativeData_A[column])
    ax.plot(cumulativeData_B['date'], cumulativeData_B[column])
    ax.plot(cumulativeData_A['date'], cumulativeData_A[column], '.-',label='A')
    ax.plot(cumulativeData_B['date'], cumulativeData_B[column], '.-',label='B' )
    ax.legend()


    plt.show()
```

In [39]:

```
# построение графика
plot_conversion(cumulativeData_product_page_A, cumulativeData_product_page_B,
                'conversion', 'date', 'Кумулятивная конверсия просмотр карточек', 'кумулят
```

взглянув на график, можем отметить следующее:

- до 9 декабря в обоих группах темп роста кумулятивной конверсии в просмотр карточек рос с одинаковым темпом, и был почти равен;
- затем данный показатель в группе B колебался возле значения 1,9-2,1 до 21 декабря, а в группе A наблюдался резкий спад 14 декабря;
- после этого темп роста показателя кумулятивной конверсии в просмотр карточек для группы A начал значительно увеличиваться и после 21 декабря перегнал показатель группы B.

In [40]:

```
# аналогично для корзины
cumulativeData_product_cart = cumulative_data(df_all_data, 'product_cart')
cumulativeData_product_cart['conversion'] = cumulativeData_product_cart['orders']/cumulati
cumulativeData_product_cart_A = cumulativeData_product_cart.query('group == "A"')
cumulativeData_product_cart_B = cumulativeData_product_cart.query('group == "B"')
```

In [41]:

```
# построение графика
plot_conversion(cumulativeData_product_cart_A, cumulativeData_product_cart_B,
                'conversion', 'date', 'Кумулятивная конверсия просмотр корзины', 'кумуляти
```

теперь опишем график кумулятивной конверсии просмотра корзины:

- до 13 декабря 2020 года наблюдался стабильный рост в обоих группах;
- в дальнейшие периоды конверсии групп росли с редкими небольшими просадками (в основном в группе А);
- рост наблюдался в целом с преобладанием кумулятивной конверсии просмотров корзины группы В над группой А со старта до 21 декабря;
- после 21 декабря кумулятивная конверсия просмотров корзины в группе А превысила показатель группы В;
- вероятно для группы A 30 декабря вообще убрали корзину и оставили только быструю покупку.

In [42]:
```
# анлогично для покупки
cumulativeData_purchase = cumulative_data(df_all_data, 'purchase')
cumulativeData_purchase['conversion'] = cumulativeData_purchase['orders']/cumulativeData_p
cumulativeData_purchase_A = cumulativeData_purchase.query('group == "A"')
cumulativeData_purchase_B = cumulativeData_purchase.query('group == "B"')
```

In [43]:
```
# построение графика
plot_conversion(cumulativeData_purchase_A, cumulativeData_purchase_B, 'conversion', 'date'
          'Кумулятивная конверсия покупки', 'кумулятивная конверсия')
```

Для показателя кумулятивной конверсии покупок наблюдается в целом схожая тенденция:

- до 13 декабря показатель группы B немного превышает показатель группы A;
- далее конверсия покупок в группе A устанавливается на постоянном уровне около 2, а конверсия группы B резко падает 14 декабря;
- после чего конверсия группы B стабильно растет и с 21 декабря превосходит показатель группы A (который тоже начинает расти, но медленнее).

Относительное изменение кумулятивной конверсии

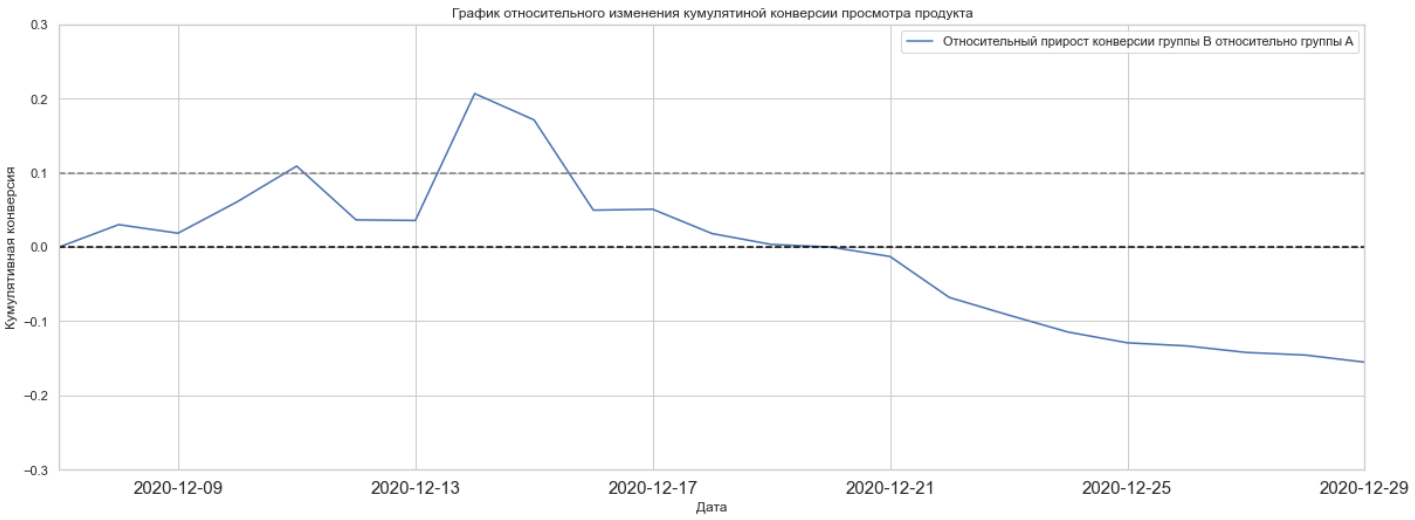
Согласно ТЗ ожидается увеличение не менее чем на 10% каждой из метрик:

- конверсии в просмотр карточек товаров — событие product_page ,
- просмотры корзины — product_cart ,
- покупки — purchase .

Чтобы оценить изменения построим графики относительного изменениея конверсий группы B к группе A на каждом из шагов.

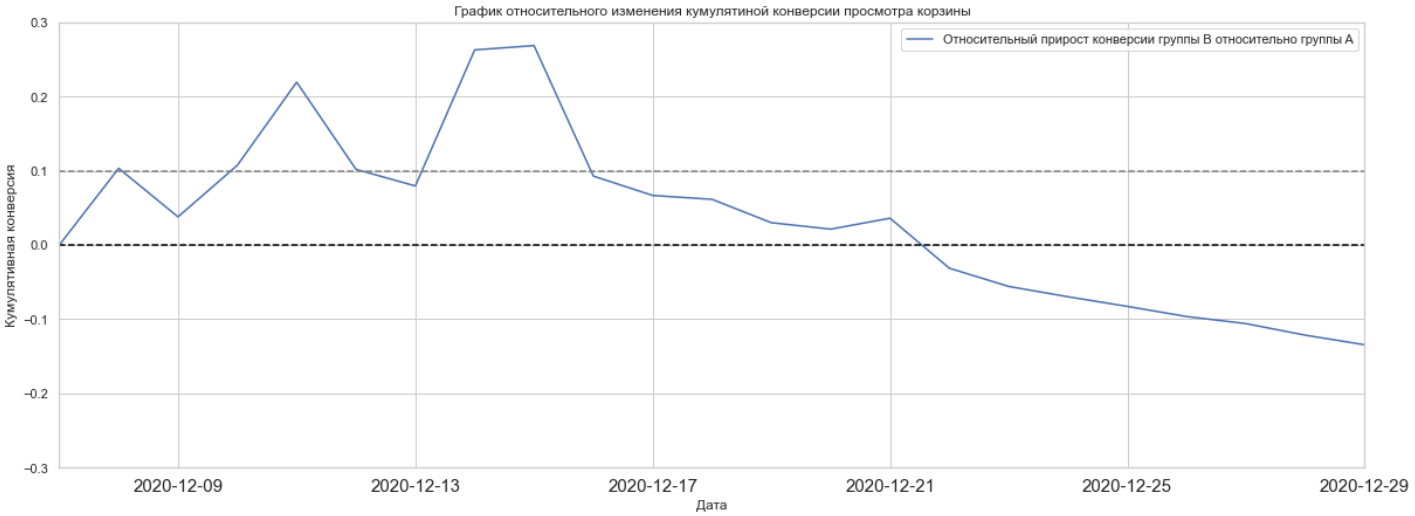
In [44]:
```
def plot_relative_change(cumulativeDataA, cumulativeDataB, title):

    """
        Функция для построение относительного изменения кумулятивной конверсии между двумя

    """

    # собираем данные в одном датафрейме
    mergedCumulativeConversions = cumulativeDataA[['date','conversion']].merge(cumulativeD

    # задаем размер поля
    plt.subplots(figsize=(20,7))

    # строим отношение средних данных
    plt.plot(mergedCumulativeConversions['date'], mergedCumulativeConversions['conversionB
    plt.legend()


    # добавляем ось X
    plt.axhline(y=0, color='black', linestyle='--')
    plt.axhline(y=0.1, color='grey', linestyle='--')
    plt.axis([dt.datetime(2020, 12, 7), dt.datetime(2020, 12, 29), -0.3, 0.3])

    # размер шрифта по оси x
    plt.tick_params(axis='x', which='major', labelsize=15)

    plt.title(title)
    plt.xlabel('Дата')
    plt.ylabel('Кумулятивная конверсия')
```

In [45]:
```
# построение графика относительного изменения кумулятиной конверсии просмотра продукта
plot_relative_change (cumulativeData_product_page_A, cumulativeData_product_page_B,
                      'График относительного изменения кумулятиной конверсии просмотра про
```



- в начале теста группа B превосходила группу A по показателю кумулятивной конверсии в просмотр карточек:
- с 14 декабря наблюдается стабильное снижение показателя метрики в группе B.
- с 20 декабря группа A вырвалась вперёд, а группа B стала показывать худшие результаты по относительному показателю, которые стабильно ухудшались до окончания теста.

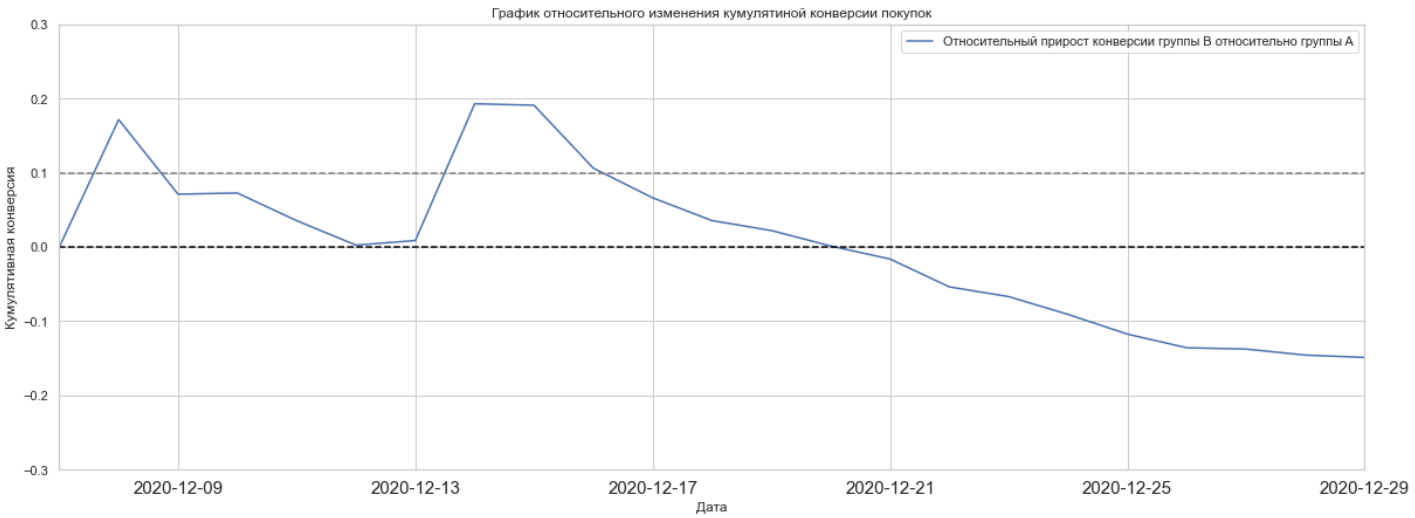
In [46]:
```
# построение графика относительного изменения кумулятиной конверсии просмотра корзины
plot_relative_change (cumulativeData_product_cart_A, cumulativeData_product_cart_B,
                      'График относительного изменения кумулятиной конверсии просмотра корз
```



Схожую ситуацию наблюдаем и по относительному показателю просмотров корзины:

- в начале теста группа B превосходила тоже группу A;
- с 16 декабря наблюдается стабильное снижение показателя метрики в группе B;
- с 22 декабря группа A вырвалась вперёд, а группа B стала показывать худшие результаты по относительному показателю, которые стабильно ухудшались до окончания теста.

In [47]:
```
# построение гарфика относительного изменения кумулятиной конверсии покупок
plot_relative_change (cumulativeData_purchase_A, cumulativeData_purchase_B,
                      'График относительного изменения кумулятиной конверсии покупок')
```



С покупками уже ожидаемо картина аналогична, необольшая разница только в датах, когда начало происходить снижение:

- в начале теста группа B превосходила тоже группу A;
- с 15 декабря наблюдается стабильное снижение показателя метрики в группе B;
- с 20 декабря группа A вырвалась вперёд, а группа B стала показывать худшие результаты по относительному показателю, которые стабильно ухудшались до окончания теста.

Статистическая разница долей

Оценим статистическую значимость разницы среднего количества пользователей, совершивших значимые действия `product_cart`, `product_page`, `purchase`. Согласно условию задачи, для проверки будем использовать `z-критерий`.

Опишем функция для проверки статистической разницы долей z-критерием.

In [48]:
```
def z_test(exp1, exp2, event, alpha):
    p1_ev = all_groups.loc[event, exp1]
    p2_ev = all_groups.loc[event, exp2]
    p1_us = df_count_group.loc[0, 'user_id']
    p2_us = df_count_group.loc[1, 'user_id']
    p1 = p1_ev / p1_us
    p2 = p2_ev / p2_us
    difference = p1 - p2
    p_combined = (p1_ev + p2_ev) / (p1_us + p2_us)
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1 / p1_us + 1 / p2_us
    distr = st.norm(0, 1)
    p_value = (1 - distr.cdf(abs(z_value))) * 2
    print('Проверка для {} и {}, событие: {}, p-значение: {p_value:.2f}'.format(exp1, exp2
    if (p_value < alpha):
        print("Отвергаем нулевую гипотезу")
        print("Среднее количество пользователей дошедших до {} в группах A и B значимо раз
    else:
        print("Не получилось отвергнуть нулевую гипотезу")
        print("Среднее количество пользователей дошедших до {} в группах A и B значимо не
```

Теперь сформулируем гипотезы. Под значимым событием понимаются действия `product_cart`, `product_page`, `purchase`.

$$\begin{cases} H_0 : \text{Среднее количество пользователей совершивших значимое событие в группах A и B равн} \\ H_1 : \text{Среднее количество пользователей совершивших значимое событие в группах A и B различае} \end{cases}$$

In [49]:
```
#  уровень значимости установим равным 0,05
a = .05

# для каждого события кроме события login вызываем функуию z_test
for event in all_groups.index:
    if event != 'login':
        z_test('A', 'B', event, a)
        print('')
```

```
Проверка для A и B, событие: product_cart, p-значение: 0.18
Не получилось отвергнуть нулевую гипотезу
Среднее количество пользователей дошедших до product_cart в группах A и B значимо не разли
чается

Проверка для A и B, событие: product_page, p-значение: 0.00
Отвергаем нулевую гипотезу
Среднее количество пользователей дошедших до product_page в группах A и B значимо различае
тся

Проверка для A и B, событие: purchase, p-значение: 0.10
Не получилось отвергнуть нулевую гипотезу
Среднее количество пользователей дошедших до purchase в группах A и B значимо не различает
ся
```

Вывод

На данном шаге мы построили кумулятивные графики конверсий и подробно описали их.

В целом относительные метрики показывают провальные результаты теста, целевые значения в группе B не только не увеличились на искомые 10% относительно A, а наоборот ухудшились.

Касаемо относительной конверсии по всем ключевым событиям можно сказать следующее:

- в начале теста группа B превосходила группу A по показателю кумулятивной конверсии в просмотр карточек:
- с 14 - 16 декабря наблюдается стабильное снижение всех показателей метрик в группе B.
- с 20 - 22 декабря группа A вырвалась вперёд, а группа B стала показывать худшие результаты по относительному показателю всех метрик, которые стабильно ухудшались до окончания теста.

Наблюдается статистическая разница долей групп A и B для события `product_page` . Для остальных событий статистической разницы не обнаружено.

Необходимо выяснить что поризошло на сайте 29-30 декабря. Похоже, что проходили какие - то обновления связанные с корзиной и покупкой, как следствие группа A перестала видеть корзину, такого рода изменения могли сказаться на конверсии в группах и дополнительно исказить результаты теста

Общий вывод

Проведя исследование, можно дать рекомендацию о том, что тест **необходимо продолжить** . Однако следует увеличить количество пользователей в группе B.

Несмотря на то, что все целевые показатели в группе B не только не увеличились, а наоборот ухудшились относительно группы `A` , однозначно утверждать, что тест неудачен нельзя. Показатели разнились долгое время и окончательное утверждение группы `A` в качестве лидера по конверсии всех показателей

произошло только в районе 20-22 декабря, когда в эту группу влили большое количество новых пользователей.

В случае принятия решения о продлении тестирования по его завершении дополнительно следует изучить данные о среднем чеке и ключевые метрики в разрезе устройств. Это может полезно, однако в рамках данного исследования такая задача не ставилась.

На сегодняшний день можно дополнительно дать следующие рекомендации:

Данные:

- пользователи в группах распределены неравномерно имется значительный перекос в пользу группы A 75 / 25 %;
- нет данных о последней неделе теста;
- тест по времени пересекается с новогодней маркетинговой акцией.

Рекомендации:

- проверить и по возможности исправить технические алгоритмы по разделению на группы;
- выяснить у ответственных сотрудников, что произошло с данными за последниюю неделю теста и что за работы происходили 29-30 декабря.

Конверсия:

- на каждом шаге мы теряем довольно большое количество пользователей (порядка половины)
- оценить конверсию между корзиной и покупкой невозможно из - за особенностей сайта заказчика

Рекомендации:

Нужно подумать о "клиентском пути" на сайте, дополнительно проанализировать действия и время нахождения клиента на сайте. Довольно странно, что авторизовавшись половина клиентов даже не посмотрела товар. Возможно есть технические проблемы в целом или проблемы на каких - то устройствах.