Анализ результатов A/B-теста

Нужна проверить тестирование изменений, связанных с внедрением улучшенной рекомендательной системы. Оцените корректность проведения теста и проанализируйте результаты теста.

- Первый этап - изучение общий информации.
- Второй этап - предобработка данных.
- Третий этап - оценка корректности проведения теста.
- Четвертый этап - исследовательский анализ данных теста.
- Пятый этап - оценка результатов A/B-тестирования.
- Шестой этап - общий вывод.

Среди данных у нас есть идентификаторы пользователя, тип события, дата и время события, дата регистрации, регион пользователя и устройство регистрации, таблица участников тестов и календарь маркетинговых событий на 2020 год. Данные разбиты на две группы A и B.

Изучение общий информации.

In [1]:
```
import pandas as pd
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
from plotly import graph_objects as go
from scipy import stats as st
import math as mth
import warnings
import seaborn as sns
```

Загрузим данные действий новых пользователей в период с 7 декабря 2020 по 4 января 2021 года.

In [2]:
```
try:
    events = pd.read_csv('/datasets/final_ab_events.csv')
except:
    events = pd.read_csv('C:\\Users\\User\\Desktop\\fin_pr\\final_ab_events.csv')

events.head()
```

Out[2]:

	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

Загрузим календарь маркетинговых событий на 2020 год.

In [3]:
```
try:
    marketing_events = pd.read_csv('/datasets/ab_project_marketing_events.csv')
except:
```

```
    marketing_events = pd.read_csv('C:\\Users\\User\\Desktop\\fin_pr\\ab_project_marketi

marketing_events.head()
```

Out[3]:

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

Пользователи, зарегистрировавшиеся с 7 по 21 декабря 2020 года.

In [4]:
```
try:
    new_users = pd.read_csv('/datasets/final_ab_new_users.csv')
except:
    new_users = pd.read_csv('C:\\Users\\User\\Desktop\\fin_pr\\final_ab_new_users.csv')

new_users.head()
```

Out[4]:

	user_id	first_date	region	device
0	D72A72121175D8BE	2020-12-07	EU	PC
1	F1C668619DFE6E65	2020-12-07	N.America	Android
2	2E1BF1D4C37EA01F	2020-12-07	EU	PC
3	50734A22C0C63768	2020-12-07	EU	iPhone
4	E1BDDCE0DAFA2679	2020-12-07	N.America	iPhone

Таблица участников тестов.

In [5]:
```
try:
    participants = pd.read_csv('/datasets/final_ab_participants.csv')
except:
    participants = pd.read_csv('C:\\Users\\User\\Desktop\\fin_pr\\final_ab_participants.

participants.head()
```

Out[5]:

	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

Название столбцов коректны, нужна проверить типы данных, следует изучить наличие пропусков и дубликатов.

Предобработка данных.

Посмотрим на наличие дубликатов и пропусков.

In [6]:
```
events.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]:
```
print(events.isna().sum())
print('Количество дубликатов:', events.duplicated().sum());
```

```
user_id            0
event_dt           0
event_name         0
details       377577
dtype: int64
Количество дубликатов: 0
```

Есть пропуски в details, но они связаны с тем что не все события имеют дополнительные данные. Они не помешают нашему анализу оставим их как есть.

In [8]:
```
marketing_events.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 [9]:
```
print(marketing_events.isna().sum())
print('Количество дубликатов:', marketing_events.duplicated().sum())
```

```
name         0
regions      0
start_dt     0
finish_dt    0
dtype: int64
Количество дубликатов: 0
```

In [10]:
```
new_users.info()
new_users['user_id'].nunique()
```

```
<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
```

Out[10]:
```
61733
```

In [11]:
```
print(new_users.isna().sum())
print('Количество дубликатов:', new_users.duplicated().sum())
new_users['user_id'].nunique()
```

```
user_id       0
first_date    0
region        0
device        0
dtype: int64
Количество дубликатов: 0
```

Out[11]:
```
61733
```

In [12]:
```
participants.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 [13]:
```
print(participants.isna().sum())
print('Количество дубликатов:', participants.duplicated().sum())
```

```
user_id    0
group      0
ab_test    0
dtype: int64
Количество дубликатов: 0
```

Проверим на неявные дубликаты

In [14]:
```
new_users['user_id'].value_counts().head()
```

Out[14]:
```
BC1E96104DDE433A    1
D45554BA350E10C4    1
CC311F9ED000F25E    1
CA84538479857DBD    1
7F59E027E41EB119    1
Name: user_id, dtype: int64
```

In [15]:
```
participants['user_id'].value_counts().head()
```

Out[15]:
```
9CBD8387C8A1DDDF    2
4D269D6E438C6D22    2
B70E5E2275EEAA7F    2
06C6018D3CB3E903    2
87314190D7FC4E12    2
Name: user_id, dtype: int64
```

Есть дубликаты, видимо пользователи пересекаются, отфильтруем их на следующем этапе.

Поменяем типы данных.

In [16]:
```
events['event_dt']  = pd.to_datetime(
    events['event_dt'] , format='%Y-%m-%dT%H:%M:%S'
```

```
)
```

In [17]:
```
events.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 [18]:
```
marketing_events['start_dt'] = marketing_events['start_dt'].map(
    lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
)
```

In [19]:
```
marketing_events['finish_dt'] = marketing_events['finish_dt'].map(
    lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
)
```

In [20]:
```
marketing_events.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 [21]:
```
new_users['first_date'] = new_users['first_date'].map(
    lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
)
new_users.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
```

У столбцов с датами изменены типы на более подходящие, пропусков нет дубликаты нет.

Оценка корректности проведения теста.

Проревем корректность всех пунктов технического задания.

- Название теста: `recommender_system_test`;

In [22]:
```
print(participants.groupby(['ab_test', 'group']).count())
```

```
                                user_id
ab_test                 group
interface_eu_test       A         5831
                        B         5736
recommender_system_test A         3824
                        B         2877
```

У нас есть два теста, среди них есть recommender_system_test и есть группы A — контрольная, B — новая платёжная воронка.

Проверим;

- дата запуска: 2020-12-07;
- дата остановки набора новых пользователей: 2020-12-21;
- дата остановки: 2021-01-04;

In [23]:
```
print('Дата запуска:', new_users['first_date'].min())
```

```
Дата запуска: 2020-12-07 00:00:00
```

Дата запуска корректна.

In [24]:
```
print('Дата остановки набора новых пользователей:', new_users['first_date'].max())
```

```
Дата остановки набора новых пользователей: 2020-12-23 00:00:00
```

Дата остановки чуть позже на пару дней, возможно эта связано со вторым тестом который шёл параллельно. У нас есть нужная нам число 2020-12-21.

In [25]:
```
print('Дата остановки:', events['event_dt'].max())
```

```
Дата остановки: 2020-12-30 23:36:33
```

Тест остановился раньше положеного на пять дней.

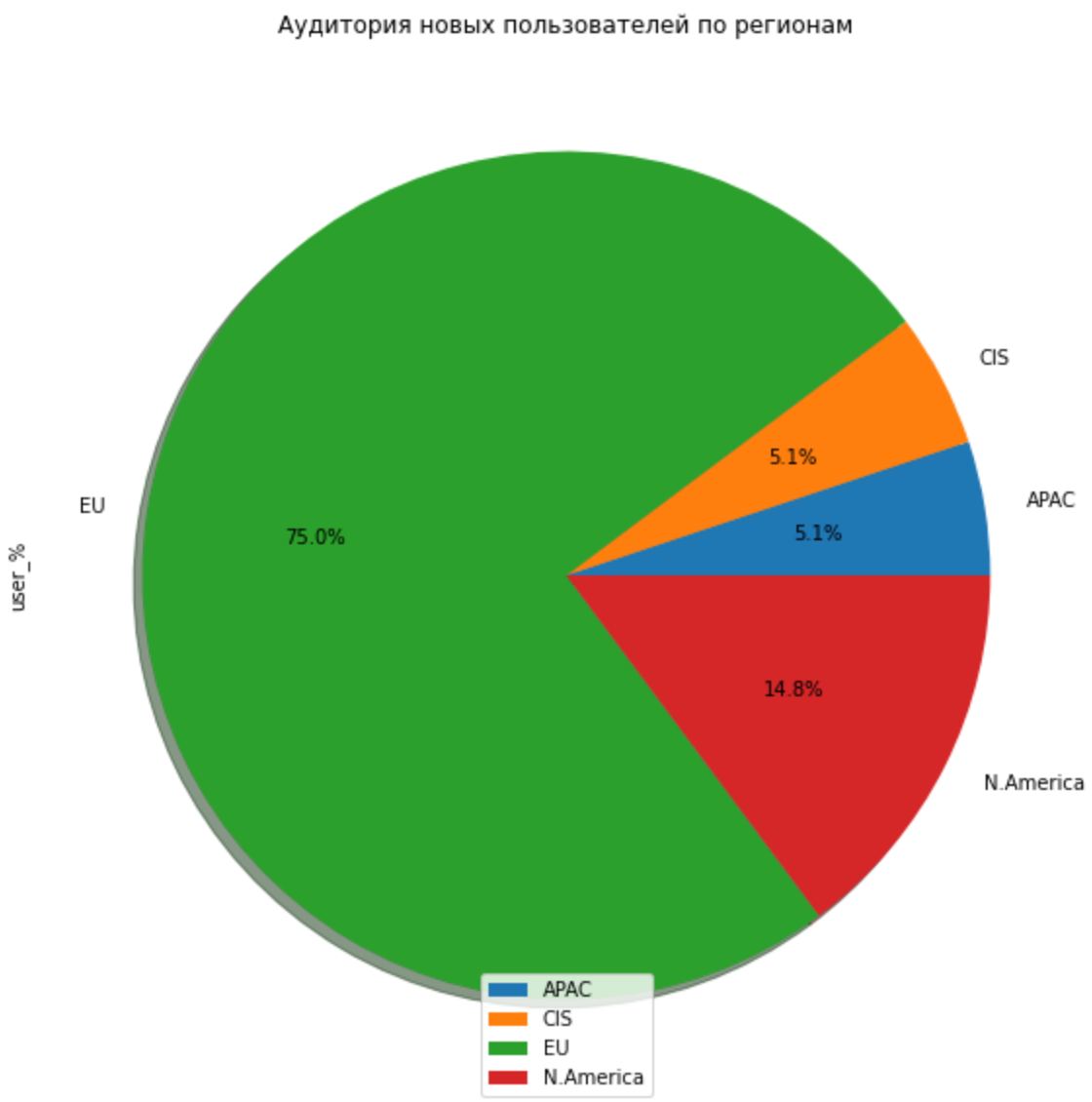
Аудитория: 15% новых пользователей из региона EU;

In [26]:
```
users_data = new_users.pivot_table(index='region', values='user_id', aggfunc='nunique')
count = new_users['user_id'].nunique()
users_data['user_%'] = (users_data['user_id'] / count) * 100
users_data.reset_index()
```

Out[26]:

	region	user_id	user_%
0	APAC	3153	5.107479
1	CIS	3155	5.110719
2	EU	46270	74.951809
3	N.America	9155	14.829994

In [27]:
```
users_data.plot(kind='pie', x='region', y='user_%',
                figsize=(15, 10),
                autopct='%1.1f%%',
                shadow=True)
plt.legend(loc=8, fontsize=10)
plt.title('Аудитория новых пользователей по регионам')
plt.show()
```



75% пользователей из региона EU;

In [28]:
```
new_users = new_users.loc[(new_users['first_date'] <= '2020-12-21')]
new_eu_users = new_users.query('region == "EU"')
new_eu_users
new_eu_users.head()
new_eu_users['user_id'].nunique()
```

Out[28]:
```
42340
```

In [29]:
```
user_all = new_users.query('user_id in @participants.user_id')
print(user_all.nunique())
#new_eu_users = new_eu_users.query('user_id in @test_group_eu.user_id')
#new_eu_users.count()
#print(participants.groupby(['ab_test', 'group']).count())
```

```
user_id       15664
first_date       15
region            4
device            4
dtype: int64
```

Оставим нужных нам пользователей.

In [30]:
```
#yu = new_users.query('user_id in @participants.user_id')
new_eu_users.nunique()
print(5532/42340*100)
```

```
13.065658951346245
```

Фактически 13% новых пользователей из региона EU.

Удалим пересекающихся пользователей которые попали в два теста одновременно.

In [31]:
```
t1 = participants.query('ab_test == "recommender_system_test"')
t2 = participants.query('ab_test != "recommender_system_test" and group != "B"')
```

In [32]:
```
#df.query('Price <= @maximum_price')
test_group = t1.query('user_id not in @t2.user_id')
print(test_group.groupby(['ab_test', 'group']).count())
test_group['user_id'].nunique()
```

```
                                user_id
ab_test                 group
recommender_system_test A         3342
                        B         2540
```

Out[32]:
```
5882
```

Оставляем пользователей только из eu региона.

In [33]:
```
test_group_eu = test_group.query('user_id in @new_eu_users.user_id')
test_group_eu['user_id'].nunique()
```

Out[33]:
```
5532
```

Ожидаемое количество участников теста 6000 после удаления пересикающихся пользователей 5532.

In [34]:
```
events_s = events.query('user_id in @new_eu_users.user_id')
events_s = events.query('user_id in @test_group_eu.user_id')
print(events_s.head())
```

```
             user_id            event_dt event_name  details
5   831887FE7F2D6CBA 2020-12-07 06:50:29   purchase     4.99
17  3C5DD0288AC4FE23 2020-12-07 19:42:40   purchase     4.99
58  49EA242586C87836 2020-12-07 06:31:24   purchase    99.99
74  A640F31CAC7823A6 2020-12-07 18:48:26   purchase     4.99
93  2F46396B6766CFDB 2020-12-07 13:29:30   purchase     4.99
```

In [35]:
```
print(events_s.count())
print(events_s['user_id'].nunique())
```

```
user_id       20382
event_dt      20382
event_name    20382
details        2740
dtype: int64
3025
```

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`.

Для ожидаемого эффект нам нужно оставить клиентов прожившие лайфтайм 14 дней. Удалим события которые находятся за пределами 2020-12-16.

In [36]:
```
ad_costs = test_group_eu.merge(new_eu_users, on=['user_id'], how='left')
ad_data = events_s.merge(ad_costs, on=['user_id'], how='left')
ad_data['event_day'] = pd.to_datetime(ad_data['event_dt']).dt.normalize();
ad_data.head()
#ad_data = ad_data.loc[(ad_data['first_date'] <= '2020-12-16')]
```

Out[36]:

	user_id	event_dt	event_name	details	group	ab_test	first_date	region	device
0	831887FE7F2D6CBA	2020-12-07 06:50:29	purchase	4.99	A	recommender_system_test	2020-12-07	EU	Android
1	3C5DD0288AC4FE23	2020-12-07 19:42:40	purchase	4.99	A	recommender_system_test	2020-12-07	EU	PC
2	49EA242586C87836	2020-12-07 06:31:24	purchase	99.99	B	recommender_system_test	2020-12-07	EU	iPhone
3	A640F31CAC7823A6	2020-12-07 18:48:26	purchase	4.99	B	recommender_system_test	2020-12-07	EU	PC
4	2F46396B6766CFDB	2020-12-07 13:29:30	purchase	4.99	A	recommender_system_test	2020-12-07	EU	PC

In [37]:
```
ad_data['lifetime'] = (
    ad_data['event_day'] - ad_data['first_date']
).dt.days
#ad_data = ad_data.loc[(ad_data['lifetime'] == 14)]
ad_data = ad_data.query('lifetime < 14 ')
ad_data
```

Out[37]:

	user_id	event_dt	event_name	details	group	ab_test	first_date	region	d
0	831887FE7F2D6CBA	2020-12-07 06:50:29	purchase	4.99	A	recommender_system_test	2020-12-07	EU	An
1	3C5DD0288AC4FE23	2020-12-07 19:42:40	purchase	4.99	A	recommender_system_test	2020-12-07	EU	
2	49EA242586C87836	2020-12-07 06:31:24	purchase	99.99	B	recommender_system_test	2020-12-07	EU	i
3	A640F31CAC7823A6	2020-12-07 18:48:26	purchase	4.99	B	recommender_system_test	2020-12-07	EU	
4	2F46396B6766CFDB	2020-12-07 13:29:30	purchase	4.99	A	recommender_system_test	2020-12-07	EU	
...	...	...	...	...	...	...	...	...	...
20376	930EACAE048DFF45	2020-12-29 06:56:00	login	NaN	A	recommender_system_test	2020-12-20	EU	
20377	E5589EAE02ACD150	2020-12-29 22:17:08	login	NaN	A	recommender_system_test	2020-12-20	EU	
20378	D21F0D4FDCD82DB2	2020-12-	login	NaN	A	recommender_system_test	2020-12-	EU	i

		29 02:17:00					20		
20379	96BDD55846D1F7F6	2020-12-29 16:53:42	login	NaN	A	recommender_system_test	2020-12-20	EU	i
20380	553BAE96C6EB6240	2020-12-29 14:09:14	login	NaN	A	recommender_system_test	2020-12-20	EU	A

19689 rows × 11 columns

In [38]:
```
ad_data['user_id'].nunique()
```

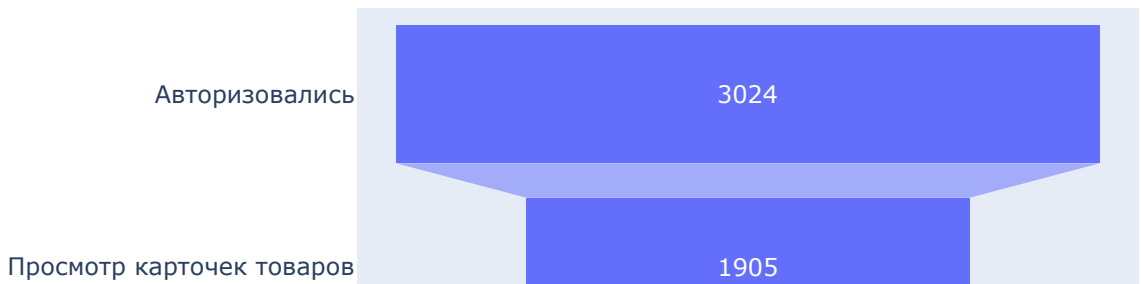
Out[38]:
```
3025
```

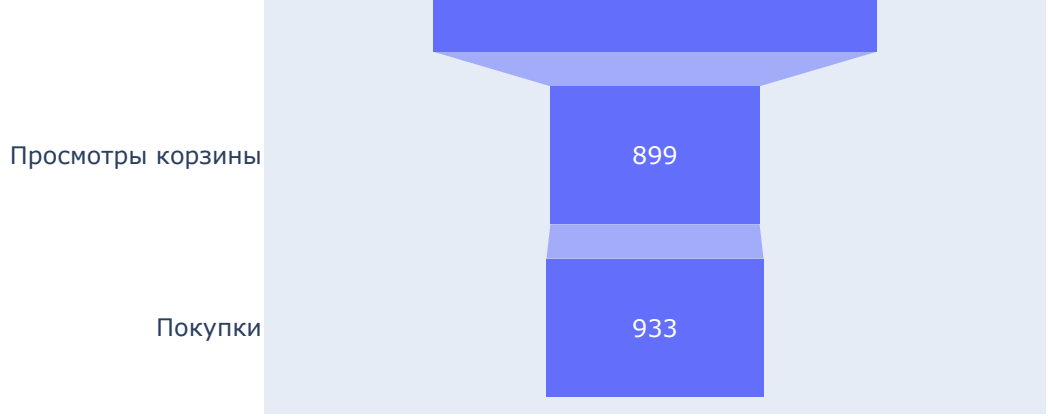
In [39]:
```
#events_count = events_s.groupby('event_name').agg({'user_id': 'count'})
events_count = (ad_data.groupby('event_name')
                .agg({'user_id': 'nunique'})
                .sort_values(by='user_id', ascending=False)
                .reset_index()
               )
events_count = events_count.iloc[[0,1,3,2]]
events_count
```

Out[39]:

	event_name	user_id
0	login	3024
1	product_page	1905
3	product_cart	899
2	purchase	933

In [40]:
```
fig = go.Figure(
    go.Funnel(
        y=[
            'Авторизовались',
            'Просмотр карточек товаров',
            'Просмотры корзины',
            'Покупки',
        ],
        x=[3024, 1905, 899, 933],
    )
)
fig.show()
```





За 14 дней с момента регистрации улучшение каждой метрики более 10%. В конце воронки покупок больше чем в пред идущем событии эта скорее всего из за покупок в один клик.

Итог соответствия данных техническому заданию:

- Название теста: `recommender_system_test`;
- группы: A — контрольная, B — новая платёжная воронка;
- дата запуска: 2020-12-07;
- дата остановки набора новых пользователей: 2020-12-23;
- дата остановки: 2020-12-30;
- аудитория: 13% новых пользователей из региона EU;
- фактическое количество участников теста: 5532.
- 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` равняется 64%,
  - просмотры корзины — `product_cart` равняется 47%,
  - покупки — `purchase` равняется 101%.

Убедимся, что время проведения теста не совпадает с маркетинговыми и другими активностями.

In [41]:
```
marketing_events
```

Out[41]:

	name	regions	start_dt	finish_dt
0	Christmas&New Year Promo	EU, N.America	2020-12-25	2021-01-03
1	St. Valentine's Day Giveaway	EU, CIS, APAC, N.America	2020-02-14	2020-02-16
2	St. Patric's Day Promo	EU, N.America	2020-03-17	2020-03-19
3	Easter Promo	EU, CIS, APAC, N.America	2020-04-12	2020-04-19
4	4th of July Promo	N.America	2020-07-04	2020-07-11
5	Black Friday Ads Campaign	EU, CIS, APAC, N.America	2020-11-26	2020-12-01
6	Chinese New Year Promo	APAC	2020-01-25	2020-02-07
7	Labor day (May 1st) Ads Campaign	EU, CIS, APAC	2020-05-01	2020-05-03
8	International Women's Day Promo	EU, CIS, APAC	2020-03-08	2020-03-10

9	Victory Day CIS (May 9th) Event	CIS	2020-05-09	2020-05-11
10	CIS New Year Gift Lottery	CIS	2020-12-30	2021-01-07
11	Dragon Boat Festival Giveaway	APAC	2020-06-25	2020-07-01
12	Single's Day Gift Promo	APAC	2020-11-11	2020-11-12
13	Chinese Moon Festival	APAC	2020-10-01	2020-10-07

In [42]:
```
A = events['event_dt'].max()  # начало теста
B = events['event_dt'].min()  # конец теста

print(marketing_events.query("start_dt <= @A" and "finish_dt >= @B"))
print(' ')
print('Тест совпадает с маркетинговыми и другими активностям:',
    marketing_events.query("start_dt <= @A" and "finish_dt >= @B")['name'].nunique());
```

```
                         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

Тест совпадает с маркетинговыми и другими активностям: 2
```

Тест совпадает с Christmas&New Year Promo которое начинается с 2020-12-25, также очень близко проходит New Year Gift Lottery (2020-12-30 по 2021-01-07), но он коснется только региона CIS. За неделю до исследование в европейском регионе заканчивается Black Friday Ads Campaign.

Проверим аудиторию теста.

- Удостоверемся, что нет пересечений с конкурирующим тестом и нет пользователей, участвующих в двух группах теста одновременно.
- Проверем равномерность распределения по тестовым группам и правильность их формирования.

Если хотитим увеличить показатель на 10% с помощью изменения, понадобится выборка минимум из 659 человек.

Проверим на пересечения в тестах.

In [43]:
```
#participants['user_id'].value_counts().head()
participants['user_id'].duplicated().sum()
```

Out[43]:
```
1602
```

In [44]:
```
test_p = participants.query('ab_test != "recommender_system_test"')
#test_p
```

In [45]:
```
old = [test_p['user_id']]
for new in ad_costs['user_id']:
    if new == old:
        print('Пересечений с конкурирующим тестом есть')
    else:
        end = 'Пересечений с конкурирующим тестом нет'
print(end)
```

```
Пересечений с конкурирующим тестом нет
```

In [46]:
```
test_p['user_id'].duplicated().sum()
```

Out[46]:
```
0
```

In [47]:
```
test_group_eu['user_id'].value_counts().head()
```

Out[47]:
```
CB3289BB00E5E465    1
9C2A5E3BF66CBB97    1
6070727198404A40    1
6BAD4743388F4545    1
EB3589638ED1D779    1
Name: user_id, dtype: int64
```

Пользователей одновременно участвующих в двух группах теста нет.

In [48]:
```
a_data = ad_data.query('group == "A"')
b_data = ad_data.query('group == "B"')
```

In [49]:
```
a_cr = (a_data.groupby('event_name')
               .agg({'user_id': 'nunique'})
               .sort_values(by='user_id', ascending=False)
               .reset_index()
              )
#a_cr
```

In [50]:
```
b_cr = (b_data.groupby('event_name')
               .agg({'user_id': 'nunique'})
               .sort_values(by='user_id', ascending=False)
               .reset_index()
              )
#b_cr
```

In [51]:
```
old_a = (a_data.groupby('first_date')
               .agg({'event_name': 'count', 'user_id': 'nunique'})
               .sort_values(by='first_date', ascending=False)
               .reset_index()
              )
old_a
```

Out[51]:

	first_date	event_name	user_id
0	2020-12-21	2257	339
1	2020-12-20	1470	228
2	2020-12-19	1342	204
3	2020-12-18	1441	184
4	2020-12-17	1269	172
5	2020-12-16	1046	143
6	2020-12-15	1346	170
7	2020-12-14	2414	316
8	2020-12-13	212	47
9	2020-12-12	250	51
10	2020-12-11	443	79
11	2020-12-10	263	54
12	2020-12-09	414	68
13	2020-12-08	504	74
14	2020-12-07	812	135

In [52]:
```
new_b = (b_data.groupby('first_date')
                .agg({'event_name': 'count', 'user_id': 'nunique'})
                .sort_values(by='first_date', ascending=False)
                .reset_index()
               )
new_b
```

Out[52]:

	first_date	event_name	user_id
0	2020-12-21	328	68
1	2020-12-20	274	53
2	2020-12-19	196	39
3	2020-12-18	223	44
4	2020-12-17	163	33
5	2020-12-16	627	85
6	2020-12-15	123	27
7	2020-12-14	290	60
8	2020-12-13	34	12
9	2020-12-12	222	45
10	2020-12-11	48	15
11	2020-12-10	129	29
12	2020-12-09	379	66
13	2020-12-08	194	37
14	2020-12-07	976	148

Проверем равномерность распределения по тестовым группам и правильность их формирования.

Сформулируем гипотезы:

- Нулевая: различий в количестве распределения клиентов между группами нет.
- Альтернативная: различия в количестве распределения клиентов между группами есть.

In [53]:
```
#from scipy import stats as st

old = old_a['user_id']
new = new_b['user_id']

alpha = 0.05

results = st.mannwhitneyu(old, new, True, 'less')

print('p-значение: ', results.pvalue)

if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу: разница статистически значима')
else:
    print(
        'Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя'
    )
```

```
p-значение:  0.9998581436703973
Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя
```

In [54]:
```
sns.set_style('whitegrid')
# назначаем размер графика
plt.figure(figsize=(10, 4))
# строим линейный график средствами seaborn
sns.lineplot(x='first_date', y='event_name', data=old_a, marker='D', label = 'group A')
sns.lineplot(x='first_date', y='event_name', data=new_b, marker='D', label = 'group B')
# формируем заголовок графика и подписи осей средствами matplotlib
plt.title('График распределения по тестовым группам ')
plt.xlabel('Дата')
plt.ylabel('Количество')# отображаем график на экране

plt.show()
```

Большой скачек с 13 числа в группе А возможна эта из за маркетинговых событий, однако у группы В таково скачка нет.

Исследовательский анализ результатов теста.

In [55]:
```
ax_data = ad_data.query('group == "A"')
bx_data = ad_data.query('group == "B"')
```

In [56]:
```
ax_data['user_id'].nunique()
```

Out[56]:
```
2264
```

In [57]:
```
bx_data['user_id'].nunique()
```

Out[57]:
```
761
```

In [58]:
```
id_event = (
    ax_data.groupby('user_id')
    .agg({'event_name' : 'count'})
    .sort_values(by='event_name', ascending=False)
)

fig, ax = plt.subplots()
id_event['event_name'].hist(figsize=(8, 5), bins=(25))
ax.set_title('Количество событий на пользователя группа A')
ax.set_xlabel('Количество событий')
ax.set_ylabel('Количество пользователей')
plt.show();
```

```
display(id_event.reset_index().head(10))
print('Медиана событий на пользователя групп A:',id_event['event_name'].median())
```

	user_id	event_name
0	CED71698585A2E46	24
1	B8EF6F0325A9979F	21
2	A25712EE46AD443A	20
3	97AD409895906A32	20
4	7347C03E6A300EFD	20
5	2F0639CBF0C3C249	20
6	109FE65EE47113C9	20
7	19F5032292917412	20
8	77FC0E20AEAC1506	20
9	1BFEE479308EFF44	20

```
Медиана событий на пользователя групп A: 6.0
```

In [59]:
```
id_event = (
    bx_data.groupby('user_id')
    .agg({'event_name' : 'count'})
    .sort_values(by='event_name', ascending=False)
)

fig, ax = plt.subplots()
id_event['event_name'].hist(figsize=(8, 5), bins=(25))
ax.set_title('Количество событий на пользователя группа B')
ax.set_xlabel('Количество событий')
ax.set_ylabel('Количество пользователей')
plt.show();
display(id_event.reset_index().head(10))
print('Медиана событий на пользователя групп B:',id_event['event_name'].median())
```

	user_id	event_name
0	1198061F6AF34B7B	24
1	115EBC1CA027854A	21
2	89545C7F903DBA34	21
3	7E8720DB6A21CF66	20
4	2C2BE85372033F77	20
5	C8460FF8BEF553A4	18
6	4EFB5E89AC11AC6D	16
7	A9908F62C41613A8	16
8	37094134968B2013	16
9	FE76759FE6BF8C68	16

```
Медиана событий на пользователя групп B: 4.0
```

У групп разброс разный, у группы B разброс сильнее смещен в лево на меньшие количество событий чем у группы A.

In [60]:
```
day_a = ax_data.groupby('event_day').agg({'event_name': 'count'})
day_b = bx_data.groupby('event_day').agg({'event_name': 'count'})
day_a
```

Out[60]:

event_day	event_name
2020-12-07	276
2020-12-08	269
2020-12-09	322
2020-12-10	283
2020-12-11	311
2020-12-12	300

2020-12-13	268
2020-12-14	890
2020-12-15	895
2020-12-16	885
2020-12-17	1029
2020-12-18	1067
2020-12-19	1272
2020-12-20	1235
2020-12-21	1640
2020-12-22	1056
2020-12-23	824
2020-12-24	693
2020-12-25	518
2020-12-26	458
2020-12-27	450
2020-12-28	303
2020-12-29	239

In [61]:
```
day_b
```

Out[61]:

event_day	event_name
2020-12-07	309
2020-12-08	200
2020-12-09	299
2020-12-10	213
2020-12-11	139
2020-12-12	180
2020-12-13	145
2020-12-14	214
2020-12-15	199
2020-12-16	322
2020-12-17	245
2020-12-18	235
2020-12-19	265
2020-12-20	274
2020-12-21	331
2020-12-22	147
2020-12-23	130

2020-12-24	105
2020-12-25	58
2020-12-26	55
2020-12-27	56
2020-12-28	50
2020-12-29	35

In [62]:
```
sns.set_style('whitegrid')
# назначаем размер графика
plt.figure(figsize=(10, 4))
# строим линейный график средствами seaborn
sns.lineplot(x='event_day', y='event_name', data=day_a, marker='D', label = 'group A')
sns.lineplot(x='event_day', y='event_name', data=day_b, marker='D', label = 'group B')
# формируем заголовок графика и подписи осей средствами matplotlib
plt.title('График распределения событий по дням')
plt.xlabel('Дата')
plt.ylabel('Количество')# отображаем график на экране

plt.show()
```

По какой то причини в группе А очень большой скачек клиентов и совершаемыми ими события с 13.12 по 21.12 число, возможно это влияние приближающегося рождества, в группе В такого аномального скачка не наблюдается.

Посмотрим на распределения по девайсам в группах.

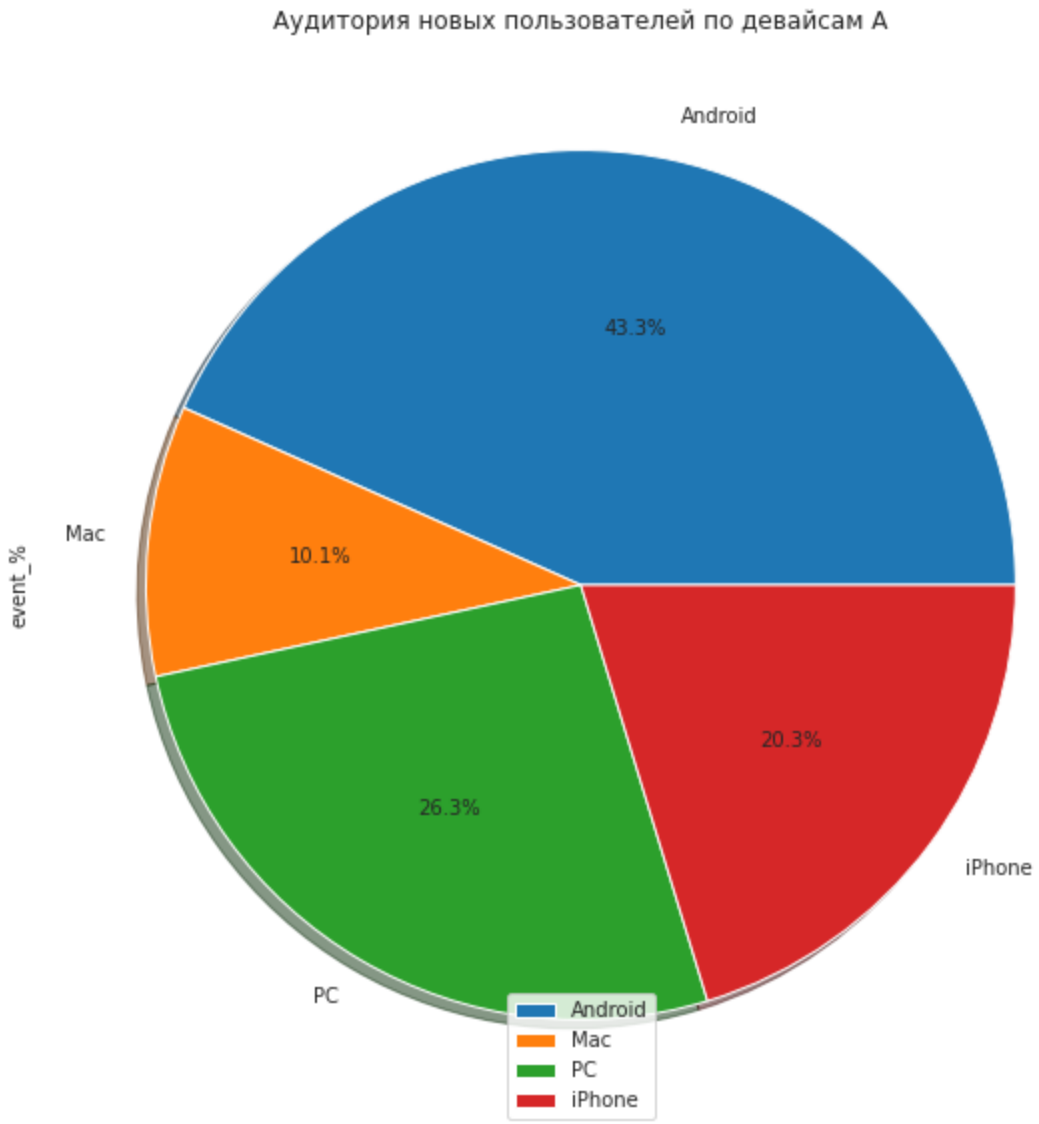
In [63]:
```
dev_a = ax_data.groupby('device').agg({'event_name': 'count', 'user_id': 'nunique'})
sum_a = dev_a['event_name'].sum()
dev_a['event_%'] = (dev_a['event_name'] / sum_a) * 100
dev_b = bx_data.groupby('device').agg({'event_name': 'count', 'user_id': 'nunique'})
sum_b = dev_b['event_name'].sum()
dev_b['event_%'] = (dev_b['event_name'] / sum_b) * 100
```

In [64]:
```
sum_a = dev_a['user_id'].sum()
dev_a['user_%'] = (dev_a['user_id'] / sum_a) * 100
sum_b = dev_b['user_id'].sum()
dev_b['user_%'] = (dev_b['user_id'] / sum_b) * 100
```

In [65]:
```
dev_a.plot(kind='pie', x='device', y='event_%',
```

```
                  figsize=(15, 10),
                  autopct='%1.1f%%',
                  shadow=True)
plt.legend(loc=8, fontsize=10)
plt.title('Аудитория новых пользователей по девайсам A')
plt.show()

dev_a.reset_index()
```



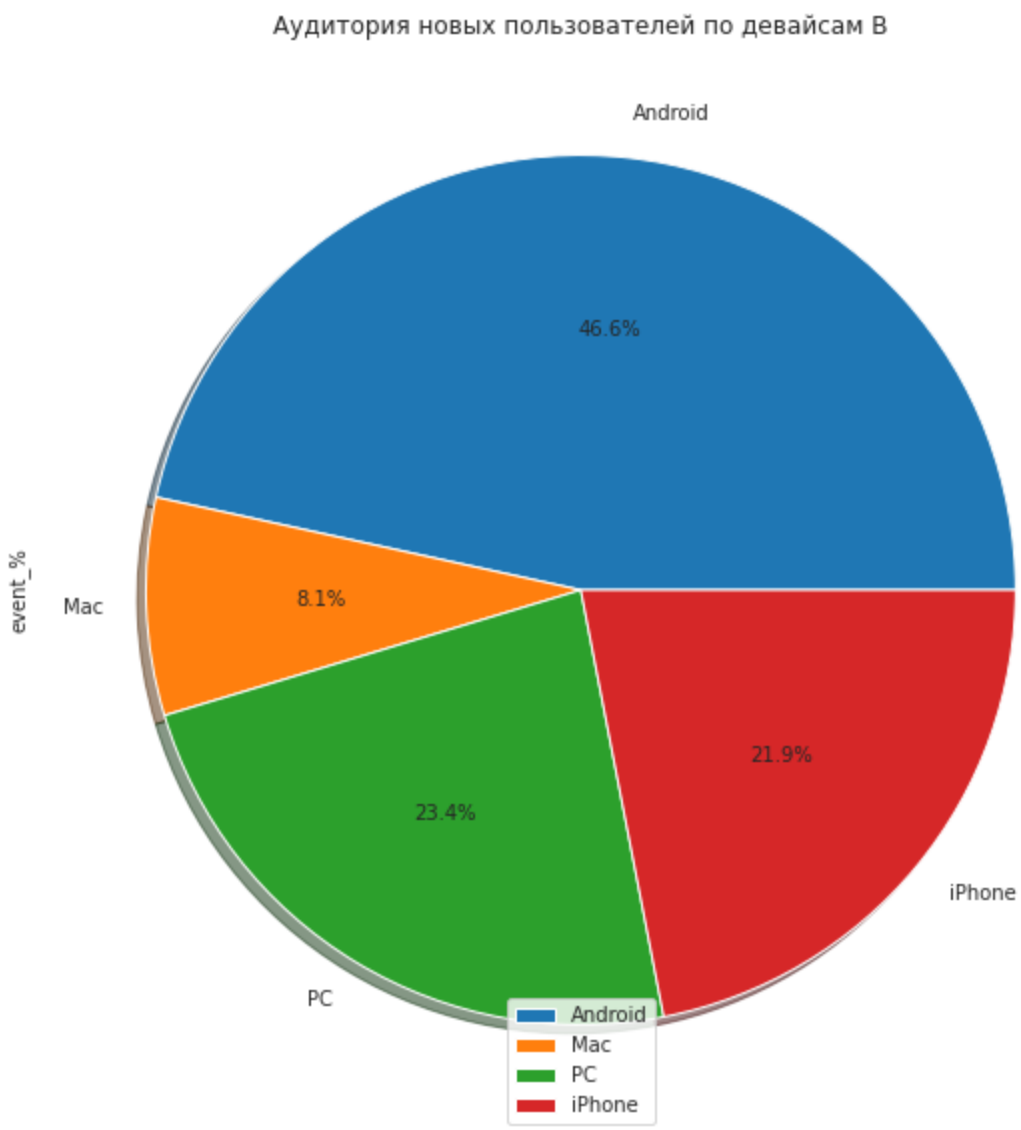
Out[65]:

	device	event_name	user_id	event_%	user_%
0	Android	6707	994	43.318478	43.904594
1	Mac	1558	221	10.062649	9.761484
2	PC	4076	600	26.325647	26.501767
3	iPhone	3142	449	20.293225	19.832155

In [66]:

```
dev_b.plot(kind='pie', x='device', y='event_%',
                  figsize=(15, 10),
                  autopct='%1.1f%%',
                  shadow=True)
plt.legend(loc=8, fontsize=10)
plt.title('Аудитория новых пользователей по девайсам B')
plt.show()

dev_b.reset_index()
```



Out[66]:

	device	event_name	user_id	event_%	user_%
0	Android	1959	354	46.576320	46.517740
1	Mac	340	65	8.083690	8.541393
2	PC	984	185	23.395150	24.310118
3	iPhone	923	157	21.944841	20.630749

В группе B пользователей на мобильных устройствах больше примерно на 3%.

In [67]:

```
a_v = (ax_data.groupby('event_name')
                .agg({'user_id': 'nunique'})
                .sort_values(by='user_id', ascending=False)
                .reset_index()
      )
a_v = a_v.iloc[[0,1,3,2]]
display(a_v)

fig = go.Figure(
    go.Funnel(
        y=[
            'Авторизовались',
            'Просмотр карточек товаров',
            'Просмотры корзины',
            'Покупки',
        ],
```

```
        x=[2264, 1474, 685, 712],
    )
)
fig.show()
```

	event_name	user_id
0	login	2264
1	product_page	1474
3	product_cart	685
2	purchase	712

Наибольший отток в группе А приходится между переходом от просмотра карточек товаров к просмотру корзины, переходя на этапе покупок идет прирост. От авторизация до просмотра карточек теряется 36,6%. От авторизации до покупки дошли 31%.

In [68]:
```
b_v = (bx_data.groupby('event_name')
                .agg({'user_id': 'nunique'})
                .sort_values(by='user_id', ascending=False)
                .reset_index()
              )
b_v = b_v.iloc[[0,1,3,2]]
display(b_v)

fig = go.Figure(
    go.Funnel(
        y=[
            'Авторизовались',
```

```
            'Просмотр карточек товаров',
            'Просмотры корзины',
            'Покупки',
        ],
        x=[760, 431, 214, 221],
    )
)
fig.show()
```

	event_name	user_id
0	login	760
1	product_page	431
3	product_cart	214
2	purchase	221

В группе B конверсия не такая большая от авторизации до просмотра теряется около 43.7% клиентов на следующем этапе от карточек до корзины 50%, но на последней этапе идет прирост на 3% видимо рекомендации группы B показывают хорошие показатели на быстрых покупках в один клик. От авторизации до покупки у группы B дошли 29% эти показатели немного ниже чем у группы A.

In [69]:
```
ad_data.head(1)
```

Out[69]:

	user_id	event_dt	event_name	details	group	ab_test	first_date	region	device
0	831887FE7F2D6CBA	2020-12-07	purchase	4.99	A	recommender_system_test	2020-12-07	EU	Android

06:50:29

In [70]:
```
details_a = ax_data.groupby('event_day').agg({'details': 'mean'})
details_b = bx_data.groupby('event_day').agg({'details': 'mean'})
#ax_data['details'].value_counts()
#bx_data['details'].value_counts()
details_a
```

Out[70]:

event_day	details
2020-12-07	24.434444
2020-12-08	24.490000
2020-12-09	17.199302
2020-12-10	18.147895
2020-12-11	27.704286
2020-12-12	24.490000
2020-12-13	9.990000
2020-12-14	26.373929
2020-12-15	21.455517
2020-12-16	21.697317
2020-12-17	16.874058
2020-12-18	18.133939
2020-12-19	22.392235
2020-12-20	27.630449
2020-12-21	26.400788
2020-12-22	29.450432
2020-12-23	12.665439
2020-12-24	28.626364
2020-12-25	32.284118
2020-12-26	22.760270
2020-12-27	18.990000
2020-12-28	12.431860
2020-12-29	8.538387

In [71]:
```
details_b
```

Out[71]:

event_day	details
2020-12-07	12.778462
2020-12-08	12.748621
2020-12-09	11.460588

2020-12-10	26.240000
2020-12-11	5.990000
2020-12-12	12.990000
2020-12-13	58.948333
2020-12-14	8.955517
2020-12-15	53.138148
2020-12-16	22.063171
2020-12-17	31.823333
2020-12-18	9.097143
2020-12-19	12.409355
2020-12-20	23.777879
2020-12-21	47.924783
2020-12-22	9.390000
2020-12-23	11.101111
2020-12-24	5.823333
2020-12-25	19.275714
2020-12-26	18.561429
2020-12-27	126.101111
2020-12-28	6.656667
2020-12-29	4.990000

In [72]:
```
sns.set_style('whitegrid')
# назначаем размер графика
plt.figure(figsize=(12, 6))
# строим линейный график средствами seaborn
sns.lineplot(x='event_day', y='details', data=details_a, marker='D', label = 'group A')
sns.lineplot(x='event_day', y='details', data=details_b, marker='D', label = 'group B')
# формируем заголовок графика и подписи осей средствами matplotlib
plt.title('График распределения  стоимость покупки по дням')
plt.xlabel('Дата')
plt.ylabel('Количество')# отображаем график на экране

plt.show()
```

In [73]:
```
details_a['details'].mean()
```

Out[73]:
```
21.441783882419678
```

In [74]:
```
details_b['details'].mean()
```

Out[74]:
```
24.01063901839584
```

В группе В можно видеть сильные скачки по суммам покупок, однака общая средняя у нее меньше чем у группы В, возможна новые рекомендации влияют больше на покупки нежили на привлечение новых клиентов, однако они по какой-та причине не стабильны.

Прежде чем приступать к A/B-тестированию нужно учесть некоторые особенности данных:

- Не все данные соответствуют требованиям технического задания. Дата остановки теста не корректна, заявлена дата 2021-01-04, фактическая 2020-12-30, скорей всего эта связана с начавшейся рождественскими признаками. В тесте 13% новых пользователей из региона EU, а не 15% как заявлено в ТЗ. Фактическое количество участников теста 5532 вместо 6000.
- В группе А очень большой скачек клиентов и совершаемыми ими события с 13.12 числа.
- Нужна учитывать что есть быстрые покупки в один клик из-за чего некоторые события можно игнорировать сразу переходя к оплате.
- Тест совпадает с маркетинговыми событиями Christmas&New Year Promo которое начинается с 2020-12-25 эта может влиять на поведения пользователей.

Оценка результатов A/B-тестирования.

По результаты A/B-тестирования можно подвести итоги:

- Количество событий на пользователя не одинаково распределены в выборках у группы В разброс сильнее смещен в лева на меньшие количество событий, у группы А он более равномерный и ближе к нормальному распределению. У групп медианное количество событий на пользователя разное.

- События по дням со временем падает в обоих группах, но с 13 числа у группы А очень большой скачке событий который выбивается из общего тренда.
- На этапе просмотра карточек товаров у группы А(63%) конверсия лучше чем у группы В (57%), на этап просмотра корзин переходят 46% у группы А и 49% у группы В, на этапе покупки конверсия от прошлого этапа 103% у обеих групп. Группа В конвертируется не лучше чем группа А на всех этапах кроме этапа просмотра корзин.
- В группе В пользователей на мобильных устройствах больше, так же в этой группе можно видеть сильные скачки по суммам покупок, однако общая средняя у нее меньше чем у группы В, по какой-та причине они не стабильны.

Сформулируем гипотезы:

- H_0: Между долями есть значимая разница
- H_1: Между долями нет значимой разницы

alpha = 0.05

In [75]:
```
purchases = np.array([431, 1474])
leads = np.array([760, 2264])

p1 = purchases[0] / leads[0]
p2=  purchases[1] / leads[1]
p_combined = (purchases[0] + purchases[1]) / (leads[0] + leads[1])
difference = p1 - p2

z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/leads[0] + 1/leads[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 гипотезу: между долями есть значимая разница')
else:
    print(
        'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
    )
```

```
p-значение:  3.3566961849640364e-05
Отвергаем нулевую гипотезу: между долями есть значимая разница
```

In [76]:
```
purchases = np.array([214, 685])
leads = np.array([760, 2264])

p1 = purchases[0] / leads[0]
p2=  purchases[1] / leads[1]
p_combined = (purchases[0] + purchases[1]) / (leads[0] + leads[1])
difference = p1 - p2

z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/leads[0] + 1/leads[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 гипотезу: между долями есть значимая разница')
```

```
else:
    print(
        'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
    )
```

```
p-значение:  0.27348595720377333
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

In [77]:
```
purchases = np.array([212, 712])
leads = np.array([760, 2264])

p1 = purchases[0] / leads[0]
p2=  purchases[1] / leads[1]
p_combined = (purchases[0] + purchases[1]) / (leads[0] + leads[1])
difference = p1 - p2

z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/leads[0] + 1/leads[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 гипотезу: между долями есть значимая разница')
else:
    print(
        'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
    )
```

```
p-значение:  0.06571034551803678
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Разница между конверсиями оказалось значимой только на первом этапе, дальше разницы между конверсиями нет.

Общие вывод.

Общие результаты исследования такие:

- Не смотря на одинаковые медианы цифры в группе B участники теста в основном сосредоточены от одного до четырех событий, когда в группе A они распределены более равномерно.
- События по дням со временем падает в обоих группах, с 13 числа у группы A очень большой скачке событий, у B такого скачка нет.
- Конверсия B на всех этапах хуже A корми первого, если посмотреть конверсию от первого события авторизации до последнего покупок то разницы между группами нет. Проверка z-критерием статистическую разность долей показала что между долями нет значимой разницы корми первого этапа.

Хоть по аудитории и есть не большие отклонения в целом участников достаточна для проведения теста, но тест прерывается не дав возможности отследить лайфтайм всех участников. Также тест проходит во время маркетингово события Christmas&New Year Promo перед рождественскими праздниками что сильно могло повлиять на поведения людей в группах, поэтому тест сложна считать корректным. Тесты нужна проводить с полным лайфтаймом и выбрать окно так чтобы не до, не вовремя теста не было не каких маркетинговых событий и акций.