Recommnder system test

В нашем распоряжении результаты A/B теста. Регион проведение теста - Евросоюз. Задача - оценить корректность его проведения и проанализировать результаты.

Содержание

In [1]:
```python
import pandas as pd
from plotly import graph_objects as go
from plotly.subplots import make_subplots
from matplotlib import pyplot as plt
import seaborn as sns
from warnings import filterwarnings
filterwarnings("ignore")
import numpy as np
import math
from scipy import stats as st
```

Предобработка данных

In [2]:
```python
# Расписание промоакций
schedule = pd.read_csv('ab_project_marketing_events.csv')
display(schedule)
```

	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 [3]:
```python
# Приведм даты к корректному типу данных
schedule.start_dt = pd.to_datetime(schedule.start_dt)
schedule.finish_dt = pd.to_datetime(schedule.finish_dt)
print(schedule.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
None
```

In [4]:
```python
# Выберем интересующий нас период
schedule = schedule.sort_values(by='start_dt').reset_index(drop=True)
actual_schedule = schedule.query('finish_dt >= "2020-12-07" and start_dt <= "2021-01-04"')
display(actual_schedule)
```

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

На интересующий нас период выпадает рождественско-новогодняя промоакция с 25 декабря по 3 января

In [5]:
```python
# Новые пользователи
new_users = pd.read_csv('final_ab_new_users.csv')
display(new_users.head())
```

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

In [6]:
```python
# Приведем даты к корректному типу
new_users.first_date = pd.to_datetime(new_users.first_date)
print(new_users.info())
print(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  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
None
61733
```

In [7]:
```python
# Есть ли дубликаты?
print(new_users.duplicated().sum())
print(new_users.region.unique())
```

```
0
['EU' 'N.America' 'APAC' 'CIS']
```

Дубликатов нет. Интересующий нас Евросоюз обозначен только одним способом.

In [8]:
```python
# Новые события
new_events = pd.read_csv('final_ab_events.csv')
display(new_events.head())
```

	user_id	event_dt	event_name	details
0	E1BDDCE0DAFA2679	2020-12-07 20:22:03	purchase	99.99
1	7B6452F081F49504	2020-12-07 09:22:53	purchase	9.99
2	9CD9F34546DF254C	2020-12-07 12:59:29	purchase	4.99
3	96F27A054B191457	2020-12-07 04:02:40	purchase	4.99
4	1FD7660FDF94CA1F	2020-12-07 10:15:09	purchase	4.99

In [9]:
```python
# Приведем даты к корректному типу
new_events.event_dt = pd.to_datetime(new_events.event_dt)
print(new_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
None
```

In [10]:
```python
# Есть ли дубликаты?
print(new_events.duplicated().sum())
# В столбце details очень много пропусков. Вероятно, "детали" - это только доходы от покупок. Проверим.

purchase = new_events.query('event_name == "purchase"')
print(purchase.shape)
```

```
0
(62740, 4)
```

Дубликатов нет. Число покупок равно числу "деталей". Значит, переименуем "детали" в "доход". Пропуски оставим, здесь они не мешают.

In [11]:
```python
new_events = new_events.rename(columns = {'details': 'income'})
```

In [12]:
```python
# Таблица тестов
test_groups = pd.read_csv('final_ab_participants.csv')
display(test_groups.head())
```

	user_id	group	ab_test
0	D1ABA3E2887B6A73	A	recommender_system_test
1	A7A3664BD6242119	A	recommender_system_test
2	DABC14FDDFADD29E	A	recommender_system_test
3	04988C5DF189632E	A	recommender_system_test
4	482F14783456D21B	B	recommender_system_test

In [13]:
```python
print(test_groups.info())
print()
print(test_groups.duplicated().sum())
print()
print(test_groups.ab_test.unique())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18268 entries, 0 to 18267
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   user_id  18268 non-null  object
 1   group    18268 non-null  object
 2   ab_test  18268 non-null  object
dtypes: object(3)
memory usage: 428.3+ KB
None

0

['recommender_system_test' 'interface_eu_test']
```

Дубликтов нет, пропусков нет. Тестов - два вида.

Исследовательский анализ данных

Корректность набора участников теста

In [14]:
```python
# Сделаем срез по нужному нам тесту
recommender_system_test = test_groups.query('ab_test == "recommender_system_test"').reset_index(drop=True)
# Из среза уберем столбец "ab_test"
recommender_system_test = recommender_system_test.drop('ab_test', axis=1)
# Сравним количество строк с количеством уникальных значений user_id
print(recommender_system_test.shape)
print(recommender_system_test.user_id.nunique())
```

```
(6701, 2)
6701
```

Пользователей, попавших и в контрольную и в тестовую группы, нет: число уникальных пользователей в таблице recommender_system_test равно числу строк в ней.

In [15]:
```python
# Создадим срез пользователям из ЕС
eu_users = new_users.query('region == "EU"').reset_index(drop=True)
eu_users = eu_users.drop('region', axis =1)
# Объединим таблицы пользователей из ЕС и пользователей, участвующих в нашем тесте,
# получим участников интересующего нас теста из нужного нам региона
eu_users_tests = pd.merge(eu_users, recommender_system_test, on= 'user_id' )
print(eu_users_tests.shape)
```

```
(6351, 4)
```

Несколько сотен участников теста были не из ЕС. Теперь остались только нужные, их чуть больше необходимых 6000

In [16]:
```python
# Проверим, есть ли общие пользователи в параллельном тесте
interface_eu_test = test_groups.query('ab_test == "interface_eu_test"').reset_index(drop=True)

total = 0
for i in eu_users_tests.user_id.unique():
    if i in interface_eu_test.user_id.unique():
        total += 1
print(total)
```

```
1602
```

1602 пользователя принимают участие в обоих тестах. То есть примерно каждый четвертый пользователь из нашего теста также есть в другом, это ставит под сомнение корректность проведения теста, потому что участие в другом тесте может влиять на поведение этих пользователей.

Продуктовая воронка

Для воронки нам понадобится вычислить, сколько уникальных пользователей приходится на события каждого типа в каждой группе, а также общее число пользователей. включая, не совершивших ни одного действия - тоже для обеих групп.

In [17]:
```python
# Сделаем срезы участников тестов по группе
eu_users_tests_a = eu_users_tests.query('group == "A"')
eu_users_tests_b = eu_users_tests.query('group == "B"')
# Общее число пользователей обеих групп
print(eu_users_tests_a.user_id.nunique())
print(eu_users_tests_b.user_id.nunique())
```

```
3634
2717
```

In [18]:
```python
# Теперь найдем пользователей, которые совершили хотя бы одно действие
users_and_events = pd.merge(new_users, new_events, on = 'user_id')
# Выделим европейский сегмент
eu_users_and_events = users_and_events.query('region == "EU"')
# Уберем из европейского среза столбец region
eu_users_and_events = eu_users_and_events.drop('region', axis = 1)
# Объединим европейский срез пользователей с хотя бы одним действием и таблицу нашего теста
# Получим участников теста с хотя бы одним действием
eu_tests_with_events = pd.merge(eu_users_and_events, recommender_system_test, on = 'user_id')

display(eu_tests_with_events.head())
# Разделим получившуюся таблицу по группам
eu_tests_with_events_a = eu_tests_with_events.query('group == "A"')
eu_tests_with_events_b = eu_tests_with_events.query('group == "B"')
```

	user_id	first_date	device	event_dt	event_name	income	group
0	D72A72121175D8BE	2020-12-07	PC	2020-12-07 21:52:10	product_page	NaN	A
1	D72A72121175D8BE	2020-12-07	PC	2020-12-07 21:52:07	login	NaN	A
2	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-07 15:32:54	product_page	NaN	B
3	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-08 08:29:31	product_page	NaN	B
4	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-10 18:18:27	product_page	NaN	B

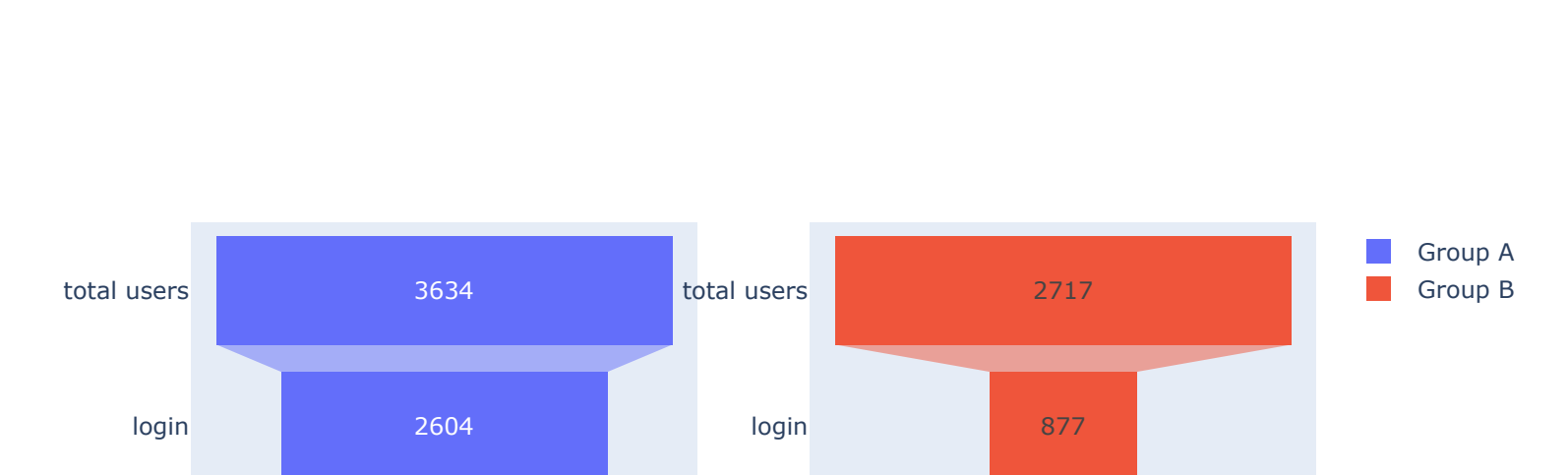
In [19]:
```python
# В обеих группах подсчитаем число уникальных пользователей для каждого типа события.
# Отсортируем по убыванию числа событий
users_by_event_types_a = eu_tests_with_events_a.groupby('event_name', as_index = False).agg({'user_id':
 'nunique'}).sort_values(by='user_id', ascending = False)
users_by_event_types_b = eu_tests_with_events_b.groupby('event_name', as_index = False).agg({'user_id':
 'nunique'}).sort_values(by='user_id', ascending = False)
# Переименуем столбец с аггрегированными данными
users_by_event_types_a = users_by_event_types_a.rename(columns={'user_id': 'num_users'})
users_by_event_types_b = users_by_event_types_b.rename(columns={'user_id': 'num_users'})

# Сделаем воронки по группам из общего числа пользователей и количества пользователей по типам событий
# Переменная total_users включает в себя как тех пользователей, которые совершали все, некоторые или хо
тя бы один из этих шагов, так и тех, которые не совершали никаких.
funnel_a = pd.concat([pd.DataFrame(data = [['total users', eu_users_tests_a.user_id.nunique()]], column
s = ['event_name', 'num_users']), users_by_event_types_a]).reset_index(drop=True)
funnel_b = pd.concat([pd.DataFrame(data = [['total users', eu_users_tests_b.user_id.nunique()]], column
s = ['event_name', 'num_users']), users_by_event_types_b]).reset_index(drop=True)
# Посмотрим, что вышло, на примере воронки контрольной группы
display(funnel_a)
```

	event_name	num_users
0	total users	3634
1	login	2604
2	product_page	1685
3	purchase	833
4	product_cart	782

Теперь визуализируем получившиеся воронки

In [20]:
```python
fig = make_subplots(rows=1, cols=2)
fig.add_trace(go.Funnel(name = "Group A", y = funnel_a.event_name, x=funnel_a.num_users), row=1, col=1)
fig.add_trace(go.Funnel(name = "Group B", y = funnel_b.event_name, x=funnel_b.num_users), row=1, col=2)
fig.show()
```



In [21]:
```python
# Число пользователей, совершивших хотя бы один шаг, совпадает с числом логинов (см. воронку)
print(eu_tests_with_events_b.user_id.nunique())
```

```
877
```

Очевидно, обязательным действием, без которого не возможны другие шаги, является вход (login). И действительно, число логинов совпадает с количеством уникальных пользователей в таблице пользователей, совершавших хотя бы одно действие. Из этого можно заключить, что 1840 пользователей из группы В не совершили ни одного действия, даже не залогинились ни разу. Из-за этих-то 1840 случаев получилось, что total_users больше суммы шагов.

Дальше, при переходе к следующим шагам могут быть варианты. Например, переход в корзину - не обязательный шаг, судя по тому, что перешедших в корзину меньше, чем совершивших покупку.

Тестовая группа демонстрирует ярко выраженную отрицательную тенденцию по сравнению с контрольной. Конверсия в покупателей 9% против 22% в группе А. Переход на продуктовую страницу - 46 % в контрольной группе, в тестовой - 18%. Метрики ухудшились в два с половиной раза.

Распределение событий по пользователям

In [22]:
```python
# Количество событий по пользователям
events_by_users = new_events.groupby('user_id', as_index = False).agg({'event_dt': 'count'})
events_by_users = events_by_users.rename(columns = {'event_dt': 'num_events'})
display(events_by_users.head())
```

	user_id	num_events
0	0001710F4DDB1D1B	6
1	000199F1887AE5E6	16
2	0002499E372175C7	9
3	0002CE61FF2C4011	12
4	000456437D0EFFE1	4

In [23]:
```python
# Объединим таблицу всех участников теста с таблицей количества событий на пользователя,
# сохранив всех пользователей из левой таблицы.
# Получим события для всех участников теста с пропусками для тех, кто не делал ничего
events_by_ab_groups_users = pd.merge(eu_users_tests, events_by_users, on='user_id', how='left')
display(events_by_ab_groups_users.head())
```

	user_id	first_date	device	group	num_events
0	D72A72121175D8BE	2020-12-07	PC	A	2.0
1	E6DE857AFBDC6102	2020-12-07	PC	B	NaN
2	DD4352CDCF8C3D57	2020-12-07	Android	B	12.0
3	831887FE7F2D6CBA	2020-12-07	Android	A	8.0
4	4CB179C7F847320B	2020-12-07	iPhone	B	6.0

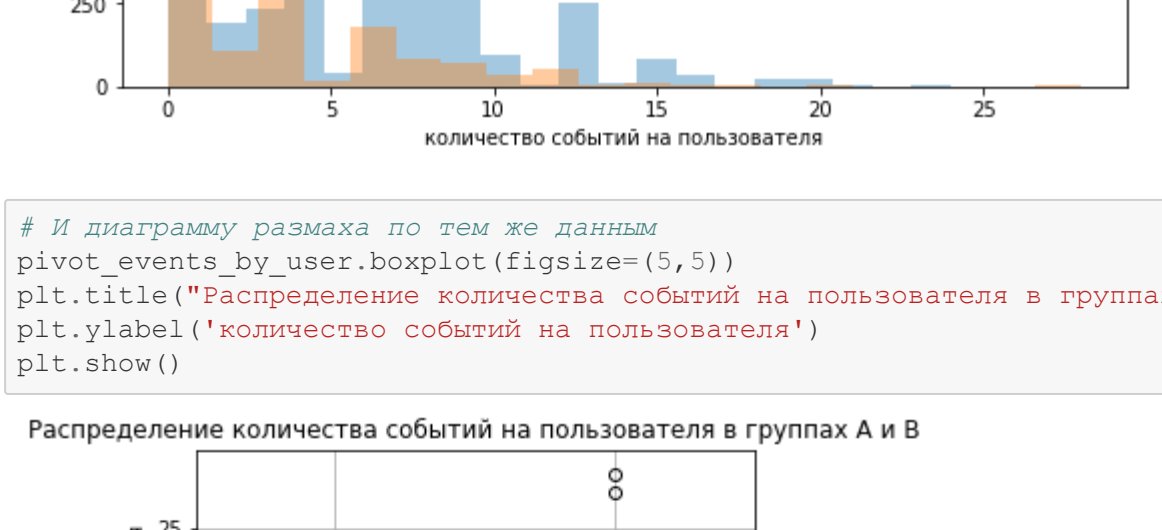
In [24]:
```python
# Заменим пропуски на нули
events_by_ab_groups_users = events_by_ab_groups_users.fillna(0)
display(events_by_ab_groups_users.head())
```

	user_id	first_date	device	group	num_events
0	D72A72121175D8BE	2020-12-07	PC	A	2.0
1	E6DE857AFBDC6102	2020-12-07	PC	B	0.0
2	DD4352CDCF8C3D57	2020-12-07	Android	B	12.0
3	831887FE7F2D6CBA	2020-12-07	Android	A	8.0
4	4CB179C7F847320B	2020-12-07	iPhone	B	6.0

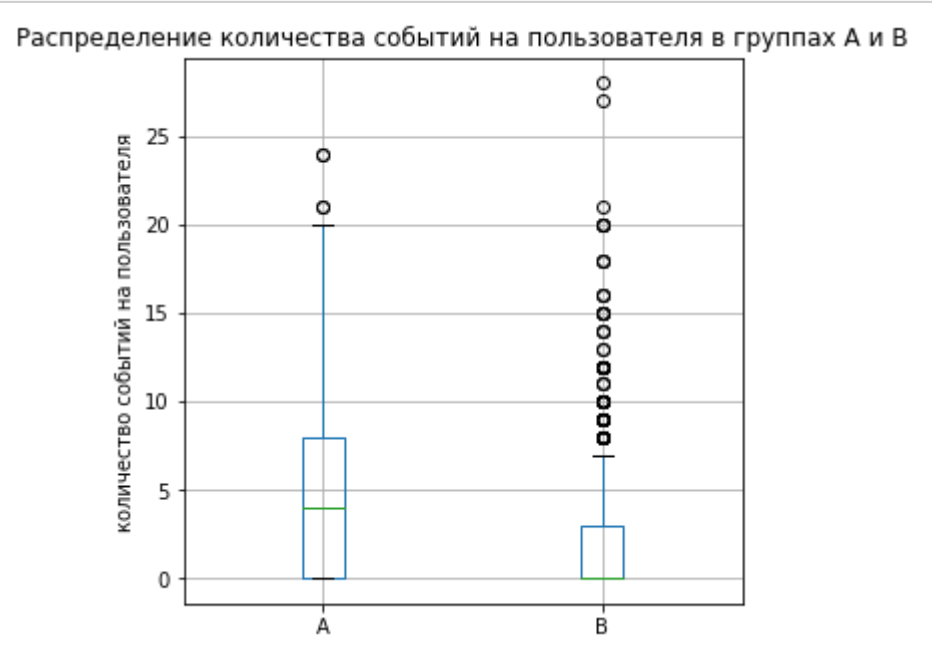
In [25]:
```python
# Сделаем сводную таблицу событий на пользователя
pivot_events_by_user = events_by_ab_groups_users.pivot_table(index = 'user_id', columns = 'group', valu
es = 'num_events')
display(pivot_events_by_user.head())
```

group	A	B
user_id		
000ABE35EE11412F	0.0	NaN
001064FEAAB631A1	NaN	6.0
0010A1C096941592	12.0	NaN
001C05E87D336C59	0.0	NaN
00341D8401F0F665	2.0	NaN

In [26]:
```python
# Построим график распределения значений количества событий на пользователя для обеих групп
plt.figure(figsize= (9,6))
sns.distplot(pivot_events_by_user['A'], bins=20, kde=False)
sns.distplot(pivot_events_by_user['B'], bins=20, kde=False)
plt.title("Распределение количества событий на пользователя в группах А и В")
plt.legend(labels = ['A', 'B'])
plt.xlabel('количество событий на пользователя')
plt.ylabel('частота значений')
plt.show()
```



In [27]:
```python
# И диаграмму размаха по тем же данным
pivot_events_by_user.boxplot(figsize=(5,5))
plt.title("Распределение количества событий на пользователя в группах А и В")
plt.ylabel('количество событий на пользователя')
plt.show()
```



In [28]:
```python
print("Среднее число событий в группе А: ", end ='')
print(np.round (pivot_events_by_user.A.mean(),2))
print()

print("Среднее число событий в группе В: ", end ='')
print(np.round(pivot_events_by_user.B.mean(), 2))
print()

print("Медианное число событий в группе А: ", end ='')
print(np.round (pivot_events_by_user.A.median(), 2))
print()

print("Медианное число событий в группе В: ", end ='')
print(np.round (pivot_events_by_user.B.median(), 2))
print()

print("Мода событий в группе А: ", end ='')
print(np.round (float(pivot_events_by_user.A.mode()), 2))
print()

print("Мода событий в группе В: ", end ='')
print(np.round(float(pivot_events_by_user.B.mode()), 2))
```

```
Среднее число событий в группе А: 5.04

Среднее число событий в группе В: 1.88

Медианное число событий в группе А: 4.0

Медианное число событий в группе В: 0.0

Мода событий в группе А: 0.0

Мода событий в группе В: 0.0
```

Гистограмма и диаграмма размаха показывают, что распределиние количества событий на пользователя у двух групп заметно различается. В контрольной группе медианное значение количества событий на пользователя - 4, в тестовой - 0. Среднее арифметическое также заметно больше в контрольной группе. Совпадает только мода (0), но на гистограмме хорошо видно, что нулевой столбик гораздо выше в тестовой группе. Три четверти значений тестовой группы лежат между 0 и 3, у контрольной группы - между 0 и 8. При этом в целом разброс значений в группе В несколько больше.

Распределение событий по дням

In [29]:
```python
# Добавим столбец дня в таблицу участников теста
eu_tests_with_events['event_day'] = eu_tests_with_events['event_dt'].dt.date
display(eu_tests_with_events.head())
```

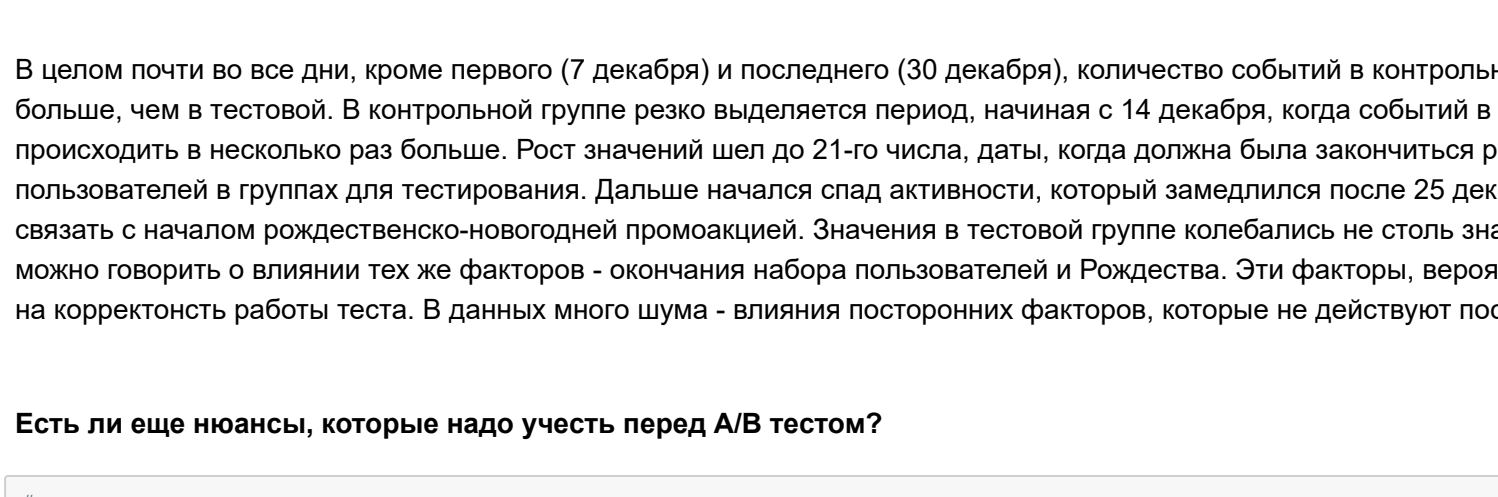
	user_id	first_date	device	event_dt	event_name	income	group	event_day
0	D72A72121175D8BE	2020-12-07	PC	2020-12-07 21:52:10	product_page	NaN	A	2020-12-07
1	D72A72121175D8BE	2020-12-07	PC	2020-12-07 21:52:07	login	NaN	A	2020-12-07
2	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-07 15:32:54	product_page	NaN	B	2020-12-07
3	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-08 08:29:31	product_page	NaN	B	2020-12-08
4	DD4352CDCF8C3D57	2020-12-07	Android	2020-12-10 18:18:27	product_page	NaN	B	2020-12-10

In [30]:
```python
# Сводная таблица событий по дням для двух групп
pivot_events_by_days = eu_tests_with_events.pivot_table(index = 'event_day', columns = 'group', values
= 'event_name', aggfunc = 'count')
display(pivot_events_by_days)
```

group	A	B
event_day		
2020-12-07	318.0	356.0
2020-12-08	313.0	238.0
2020-12-09	371.0	338.0
2020-12-10	331.0	249.0
2020-12-11	356.0	161.0
2020-12-12	346.0	199.0
2020-12-13	308.0	164.0
2020-12-14	1015.0	248.0
2020-12-15	1030.0	222.0
2020-12-16	1007.0	369.0
2020-12-17	1200.0	281.0
2020-12-18	1244.0	259.0
2020-12-19	1430.0	288.0
2020-12-20	1434.0	309.0
2020-12-21	1903.0	401.0
2020-12-22	1236.0	185.0
2020-12-23	959.0	189.0
2020-12-24	850.0	149.0
2020-12-25	639.0	107.0
2020-12-26	580.0	113.0
2020-12-27	547.0	114.0
2020-12-28	479.0	95.0
2020-12-29	413.0	73.0
2020-12-30	NaN	4.0

Вопреки заявленному в ТЗ, у нас есть данные только по 30 декабря, а не по 4 января.

In [31]:
```python
# График событий по дням
pivot_events_by_days.plot(kind='bar', figsize = (12,5))
plt.title("Количество событий по дням")
plt.ylabel("Количество событий")
plt.xlabel("Дата")
plt.xticks(rotation=60)
plt.show()
```



В целом почти во все дни, кроме первого (7 декабря) и последнего (30 декабря), количество событий в контрольной группе было больше, чем в тестовой. В контрольной группе резко выделяется период, начиная с 14 декабря, когда событий в группе стало происходить в несколько раз больше. Рост значений шел до 21-го числа, даты, когда должна была закончиться регистрация новых пользователей в группах для тестирования. Дальше начался спад активности, который замедлился после 25 декабря, что можно связать с началом рождественско-новогодней промоакцией. Значения в тестовой группе колебались не столь значительно, хотя можно говорить о влиянии тех же факторов - окончания набора пользователей и Рождества. Эти факторы, вероятно, могут влиять на корректонсть работы теста. В данных много шума - влияния посторонних факторов, которые не действуют постоянно.

Есть ли еще нюансы, которые надо учесть перед A/B тестом?

In [32]:
```python
# Проверим распределение дохода
print("Средний доход от покупки, группа А: ", end='')
print (np.round (eu_tests_with_events_a.income.mean(), 2))
print()
print("Средний доход от покупки, группа В: ", end='')
print (np.round(eu_tests_with_events_b.income.mean(), 2))
print()
print("Медианный доход от покупки, группа А: ", end='')
print (eu_tests_with_events_a.income.median())
print()
print("Медианный доход от покупки, группа В: ", end='')
print (eu_tests_with_events_b.income.median())
print()
print("Стандартное отклонение, доход от покупки, группа А: ", end='')
print(np.round(eu_tests_with_events_a.income.std(), 2))
print()
print("Стандартное отклонение, доход от покупки, группа В: ", end='')
print(np.round(eu_tests_with_events_b.income.std(), 2))
```

```
Средний доход от покупки, группа А: 23.21

Средний доход от покупки, группа В: 22.37

Медианный доход от покупки, группа А: 4.99

Медианный доход от покупки, группа В: 4.99

Стандартное отклонение, доход от покупки, группа А: 70.18

Стандартное отклонение, доход от покупки, группа В: 69.92
```

Меры центральной тенденции и меры разброса для доходов от покупок в группах А и В не различаются. Очевидно, сумма доходов с покупки распредилилась примерно одинаково в обеих группах.

In [33]:
```python
# Проверим, есть ли разница в том, с каких устройств заходят на сайт пользователи двух групп
devices = events_by_ab_groups_users.pivot_table(index = 'device', columns = 'group', values = 'user_id'
, aggfunc = 'count').sort_values(by='A', ascending = False)
display(devices)
```

group	A	B
device		
Android	1590	1228
PC	964	657
iPhone	726	582
Mac	354	250

Пропорции в использовании компьютеров и телефонов разных видов в обеих группах примерно одинаковы. Значимого влияния на результаты теста это не должно оказывать.

A/B тест

In [34]:
```python
# Данные необходимые для z-теста
display(funnel_a)
display(funnel_b)
```

	event_name	num_users
0	total users	3634
1	login	2604
2	product_page	1685
3	purchase	833
4	product_cart	782

	event_name	num_users
0	total users	2717
1	login	877
2	product_page	493
3	purchase	249
4	product_cart	244

Из выведенных таблиц нам нужны данные total_users, product_page, purchase и product_cart. Три последних показателя - интересующие нас шаги в воронке, где мы будем искать статистически значимые различия между группами.

In [35]:
```python
# Напишем функцию для определения стат. значимости разницы между долями выборок с помощью z-теста
# Аргументы функции - количество уникальных пользователей и количество пользователей, дошедших до
# интересующего нас шага (события) - для двух сравниваемых групп

def calculate_some_stats(total_users_1, current_step_1, total_users_2, current_step_2):
    # Всего предстоит делать три теста - для таких метрик, как конверсия в просмотр карточек,
    # конверсия в просмотр корзины, конверсия в покупки
    # Обычный уровень стат. значимости разделим на 3. Получим число чуть большее, чем 0.015
    alpha = 0.015
    # Пропорции выборок
    p1 = current_step_1/total_users_1
    p2 = current_step_2/total_users_2
    # Общая пропорция
    p_combined = (current_step_1 +  current_step_2) / (total_users_1 + total_users_2)
    difference = p1 - p2
    # Зед-значение
    z_value = difference / math.sqrt(p_combined * (1 - p_combined) * (1/total_users_1 + 1/total_users_2
))
    distr = st.norm(0, 1)
    # p-значение
    p_value = (1 - distr.cdf(abs(z_value))) * 2
    # Вывод
    print ('p-значение ', p_value)
    if p_value < alpha:
        print("Отвергаем нулевую гипотезу, разница между долями статистически значима")
    else:
        print("Не получилось отвергнуть нулевую гипотезу, разница между долями может быть случайной")
```

In [36]:
```python
# Просмотр карточек

# Нулевая гипотеза: доли просмотревших карточк в обеих выборках равны.
# Альтернативная гипотеза: между долями есть статистически значимая разница.

print("Гипотеза о равенстве долей просмотревших карточки:")
print()
calculate_some_stats(funnel_a['num_users'][0], funnel_a['num_users'][2], funnel_b['num_users'][0], funn
el_b['num_users'][2])
```

```
Гипотеза о равенстве долей просмотревших карточки:

p-значение  0.0
Отвергаем нулевую гипотезу, разница между долями статистически значима
```

Получили p-значение "равное нулю" из-за ограничения на число знаков после запятой: значение distr.cdf(abs(z_value) стремится к единице. P-значение стремится к нулю.

In [37]:

```
# Просмотр корзины

# Нулевая гипотеза: доли просмотревших корзину в обеих выборках равны.
# Альтернативная гипотеза: между долями есть статистически значимая разница.

print("Гипотеза о равенстве долей просмотревших корзину:")
print()
calculate_some_stats(funnel_a['num_users'][0], funnel_a['num_users'][4], funnel_b['num_users'][0], funn
el_b['num_users'][4])
```

```
Гипотеза о равенстве долей просмотревших корзину:

p-значение  0.0
Отвергаем нулевую гипотезу, разница между долями статистически значима
```

In [38]:

```
# Совершившие покупку

# Нулевая гипотеза: доли совершивших покупку в обеих выборках равны.
# Альтернативная гипотеза: между долями есть статистически значимая разница.

print("Гипотеза о равенстве долей совершивших покупку:")
print()
calculate_some_stats(funnel_a['num_users'][0], funnel_a['num_users'][3], funnel_b['num_users'][0], funn
el_b['num_users'][3])
```

```
Гипотеза о равенстве долей совершивших покупку:

p-значение  0.0
Отвергаем нулевую гипотезу, разница между долями статистически значима
```

Различия между выборками по всем трем показателям статистически значимы, но при этом мы получили отрицательный результат: метрики в группе B значительно хуже, чем в группе A.

Выводы

Исследовательский анализ данных показал, и A/B тест подтвердил, что внедрение новой рекомендательной системы не пошло на пользу ключевым метрикам. Конверсия сократилась по всем показателям в два с половиной раза.

В то же время корректность проведения теста вызывает сомнения: во-первых, из-за 1602 общих участников с параллельным тестом, во-вторых из-за влияния рождественской промоакции, в-третьих, из-за прекращения регистрации новых участников 21-го декабря.

In [ ]: