Выбор гипотезы и анализ A/B теста для интернет-магазина

Заказчик: интернет-магазин

Цель проекта: выбрать гипотезу из списка улучшений от отдела маркетинга и проверить помогает ли выбранное изменение повысить выручку.

Основные шаги:

- проведем приоритизацию гипотез, с использованием фреймворков ICE и RICE
- проведем A/B-тест и анализ результатов.

Загрузка и предобработка данных

Описание данных

Файл hypothesis.csv содержит информацию о 9 гипотезах по увеличению выручки интернет-магазина с указанными параметрами `Reach`, `Impact`, `Confidence`, `Effort`.\
`hypothesis.csv`:

```
Hypothesis — краткое описание гипотезы;
Reach — охват пользователей по 10-балльной шкале;
Impact — влияние на пользователей по 10-балльной шкале;
Confidence — уверенность в гипотезе по 10-балльной шкале;
Efforts — затраты ресурсов на проверку гипотезы по 10-балльной шкале. Чем
больше значение Efforts, тем дороже проверка гипотезы.
```

В файлах orders.csv и visitors.csv содержится информация о результатах A/B теста.
`orders.csv`:

```
transactionId — идентификатор заказа;
visitorId — идентификатор пользователя, совершившего заказ;
date — дата, когда был совершён заказ;
revenue — выручка заказа;
group — группа A/B-теста, в которую попал заказ.
```

`visitors.csv`:

```
date — дата;
group — группа A/B-теста;
visitors — количество пользователей в указанную дату в указанной группе A/B-
теста
```

Чтение данных

In [1]:
```
# импортируем библиотеки
import pandas as pd
import numpy as np
import math as mth
from datetime import datetime
from matplotlib import pyplot as plt
from scipy import stats as st
```

In [2]:
```
# отключаем предупреждения
import warnings
warnings.filterwarnings(action='ignore')
```

In [3]:
```
# выгружаем данные
hypothesis = pd.read_csv('/datasets/hypothesis.csv')
orders = pd.read_csv('/datasets/orders.csv')
visitors = pd.read_csv('/datasets/visitors.csv')
```

In [4]:
```
data = [hypothesis, orders, visitors]
data_names = ['hypothesis', 'orders', 'visitors']
```

In [5]:
```
# смотрим на первые пять строк, инфо и количество пропусков всех таблиц чтобы оценить корректн
for i in range(len(data)):
    display(data_names[i])
    display(data[i].head(), data[i].info())
    display(data[i].isna().sum())
    print( )
```

```
'hypothesis'
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Hypothesis  9 non-null      object
 1   Reach       9 non-null      int64 
 2   Impact      9 non-null      int64 
 3   Confidence  9 non-null      int64 
 4   Efforts     9 non-null      int64 
dtypes: int64(4), object(1)
memory usage: 488.0+ bytes
```

	Hypothesis	Reach	Impact	Confidence	Efforts
0	Добавить два новых канала привлечения трафика,...	3	10	8	6
1	Запустить собственную службу доставки, что сок...	2	5	4	10
2	Добавить блоки рекомендаций товаров на сайт ин...	8	3	7	3
3	Изменить структура категорий, что увеличит кон...	8	3	3	8
4	Изменить цвет фона главной страницы, чтобы уве...	3	1	1	1

```
None
Hypothesis    0
Reach         0
Impact        0
Confidence    0
Efforts       0
dtype: int64
```

```
'orders'
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   transactionId  1197 non-null   int64
 1   visitorId      1197 non-null   int64
 2   date           1197 non-null   object
 3   revenue        1197 non-null   int64
 4   group          1197 non-null   object
dtypes: int64(3), object(2)
memory usage: 46.9+ KB
```

	transactionId	visitorId	date	revenue	group
0	3667963787	3312258926	2019-08-15	1650	B
1	2804400009	3642806036	2019-08-15	730	B
2	2961555356	4069496402	2019-08-15	400	A
3	3797467345	1196621759	2019-08-15	9759	B
4	2282983706	2322279887	2019-08-15	2308	B

```
None
transactionId    0
visitorId        0
date             0
revenue          0
group            0
dtype: int64
```

```
'visitors'
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      62 non-null     object
 1   group     62 non-null     object
 2   visitors  62 non-null     int64
dtypes: int64(1), object(2)
memory usage: 1.6+ KB
```

	date	group	visitors
0	2019-08-01	A	719
1	2019-08-02	A	619
2	2019-08-03	A	507
3	2019-08-04	A	717
4	2019-08-05	A	756

```
None
date        0
group       0
visitors    0
dtype: int64
```

Пропусков в таблицах нет.

В целом типы данных соответствуют содержимому, стоит заменить только тип данных в столбцах `date` в таблицах `orders` и `visitors` на тип datetime.

Дубликаты, ошибки

In [6]:
```
# проверяем количество дубликатов в таблицах
for df in data:
    print('Количество дубликатов:', df.duplicated().sum())
```

```
Количество дубликатов: 0
Количество дубликатов: 0
Количество дубликатов: 0
```

In [7]:
```
# смотрим как распределены значения числовых столбцов, чтобы проверить есть ли ошибки
for df in data:
    display(df.describe())
```

	Reach	Impact	Confidence	Efforts
count	9.000000	9.000000	9.000000	9.000000
mean	4.777778	4.777778	5.555556	4.888889
std	3.153481	3.192874	3.045944	2.803767
min	1.000000	1.000000	1.000000	1.000000
25%	3.000000	3.000000	3.000000	3.000000
50%	3.000000	3.000000	7.000000	5.000000
75%	8.000000	7.000000	8.000000	6.000000
max	10.000000	10.000000	9.000000	10.000000

	transactionId	visitorId	revenue
count	1.197000e+03	1.197000e+03	1.197000e+03
mean	2.155621e+09	2.165960e+09	8.348006e+03
std	1.229085e+09	1.236014e+09	3.919113e+04
min	1.062393e+06	5.114589e+06	5.000000e+01
25%	1.166776e+09	1.111826e+09	1.220000e+03
50%	2.145194e+09	2.217985e+09	2.978000e+03
75%	3.237740e+09	3.177606e+09	8.290000e+03
max	4.293856e+09	4.283872e+09	1.294500e+06

	visitors
count	62.000000
mean	607.290323
std	114.400560
min	361.000000
25%	534.000000
50%	624.500000
75%	710.500000
max	770.000000

Минимальные и максимальные значения числовых столбцов находятся в разумных пределах, нет нулевых и отрицательных значений.

In [8]:
```
# смотрим уникальные значения категориальных столбцов на предмет ошибок
for i in range(len(data)):
    for col in data[i].select_dtypes(include=['object']).columns:
        if not 'dt' in col:
            display('Таблица {}, уникальные значения столбца {}:'.format(data_names[i], col))
            display(data[i][col].unique())
            print('---------------------------')
```

```
'Таблица hypothesis, уникальные значения столбца Hypothesis:'
array(['Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше
пользователей',
       'Запустить собственную службу доставки, что сократит срок доставки заказов',
       'Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверс
ию и средний чек заказа',
       'Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найду
т нужный товар',
       'Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей',
       'Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество зака
зов',
       'Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увел
ичить конверсию',
       'Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для ema
il-рассылок',
       'Запустить акцию, дающую скидку на товар в день рождения'],
      dtype=object)
---------------------------
'Таблица orders, уникальные значения столбца date:'
array(['2019-08-15', '2019-08-16', '2019-08-01', '2019-08-22',
       '2019-08-17', '2019-08-23', '2019-08-02', '2019-08-18',
       '2019-08-24', '2019-08-03', '2019-08-25', '2019-08-28',
       '2019-08-19', '2019-08-06', '2019-08-26', '2019-08-29',
       '2019-08-04', '2019-08-20', '2019-08-09', '2019-08-07',
       '2019-08-30', '2019-08-05', '2019-08-27', '2019-08-21',
       '2019-08-08', '2019-08-10', '2019-08-31', '2019-08-11',
       '2019-08-12', '2019-08-13', '2019-08-14'], dtype=object)
---------------------------
'Таблица orders, уникальные значения столбца group:'
array(['B', 'A'], dtype=object)
---------------------------
'Таблица visitors, уникальные значения столбца date:'
```

```
array(['2019-08-01', '2019-08-02', '2019-08-03', '2019-08-04',
       '2019-08-05', '2019-08-06', '2019-08-07', '2019-08-08',
       '2019-08-09', '2019-08-10', '2019-08-11', '2019-08-12',
       '2019-08-13', '2019-08-14', '2019-08-15', '2019-08-16',
       '2019-08-17', '2019-08-18', '2019-08-19', '2019-08-20',
       '2019-08-21', '2019-08-22', '2019-08-23', '2019-08-24',
       '2019-08-25', '2019-08-26', '2019-08-27', '2019-08-28',
       '2019-08-29', '2019-08-30', '2019-08-31'], dtype=object)
--------------------------
'Таблица visitors, уникальные значения столбца group:'
array(['A', 'B'], dtype=object)
--------------------------
```

Всё нормально.

In [9]:
```
# смотрим диапазон дат в таблицах, чтобы исключить ошибки
for i in range(len(data)):
    for col in data[i].select_dtypes(include=['object', 'datetime']).columns:
        if 'date' in col:
            display('В таблице {} в столбце {} диапазон дат от {} до {}'.format(data_names[i],
```

```
'В таблице orders в столбце date диапазон дат от 2019-08-01 до 2019-08-31'
'В таблице visitors в столбце date диапазон дат от 2019-08-01 до 2019-08-31'
```

В таблицах с заказами и посетителями даты от 1 до 31 августа 2019 года. Ошибок не обнаружено.

Преобразование типов

In [10]:
```
# преобразуем даты
for df in data:
    for col in df.columns:
        if 'date' in col:
            df[col] = pd.to_datetime(df[col], yearfirst=True)
```

In [11]:
```
# проверим типы данных после преобразования
display(orders.info())
display(visitors.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   transactionId  1197 non-null   int64         
 1   visitorId      1197 non-null   int64         
 2   date           1197 non-null   datetime64[ns]
 3   revenue        1197 non-null   int64         
 4   group          1197 non-null   object        
dtypes: datetime64[ns](1), int64(3), object(1)
memory usage: 46.9+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   date      62 non-null     datetime64[ns]
 1   group     62 non-null     object        
 2   visitors  62 non-null     int64         
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 1.6+ KB
```

None

Итог:

Ошибок, пропусков, дубликатов нет, привели даты к типу `datetime` .

Приоритизация гипотез

In [12]:
```
hypothesis
```

Out[12]:

	Hypothesis	Reach	Impact	Confidence	Efforts
0	Добавить два новых канала привлечения трафика,...	3	10	8	6
1	Запустить собственную службу доставки, что сок...	2	5	4	10
2	Добавить блоки рекомендаций товаров на сайт ин...	8	3	7	3
3	Изменить структура категорий, что увеличит кон...	8	3	3	8
4	Изменить цвет фона главной страницы, чтобы уве...	3	1	1	1
5	Добавить страницу отзывов клиентов о магазине,...	3	2	2	3
6	Показать на главной странице баннеры с актуаль...	5	3	8	3
7	Добавить форму подписки на все основные страни...	10	7	8	5
8	Запустить акцию, дающую скидку на товар в день...	1	9	9	5

Проведем приоритизацию гипотез с помощью подхода ICE по формуле:
ICE score = (Impact x Confidence) / Efforts

In [13]:
```
hypothesis['ice_score'] = hypothesis['Impact'] * hypothesis['Confidence'] / hypothesis['Effor
hypothesis.sort_values(by='ice_score', ascending=False)
```

Out[13]:

	Hypothesis	Reach	Impact	Confidence	Efforts	ice_score
8	Запустить акцию, дающую скидку на товар в день...	1	9	9	5	16.200000
0	Добавить два новых канала привлечения трафика,...	3	10	8	6	13.333333
7	Добавить форму подписки на все основные страни...	10	7	8	5	11.200000
6	Показать на главной странице баннеры с актуаль...	5	3	8	3	8.000000
2	Добавить блоки рекомендаций товаров на сайт ин...	8	3	7	3	7.000000
1	Запустить собственную службу доставки, что сок...	2	5	4	10	2.000000
5	Добавить страницу отзывов клиентов о магазине,...	3	2	2	3	1.333333
3	Изменить структура категорий, что увеличит кон...	8	3	3	8	1.125000
4	Изменить цвет фона главной страницы, чтобы уве...	3	1	1	1	1.000000

На первом месте по методу ICE гипотеза №8 с рейтингом 16,2 балла. На 2: гипотеза 0 и на 3: гипотеза 7.

Теперь применим формулу RICE score = (Reach x Impact x Confidence) / Efforts.

In [14]:
```
hypothesis['rice_score'] = hypothesis['Reach'] * hypothesis['Impact'] * hypothesis['Confidence
hypothesis.sort_values(by='rice_score', ascending=False)
```

Out[14]:

	Hypothesis	Reach	Impact	Confidence	Efforts	ice_score	rice_score
7	Добавить форму подписки на все основные страни...	10	7	8	5	11.200000	112.0
2	Добавить блоки рекомендаций товаров на сайт ин...	8	3	7	3	7.000000	56.0
0	Добавить два новых канала привлечения трафика,...	3	10	8	6	13.333333	40.0
6	Показать на главной странице баннеры с актуаль...	5	3	8	3	8.000000	40.0
8	Запустить акцию, дающую скидку на товар в день...	1	9	9	5	16.200000	16.2
3	Изменить структура категорий, что увеличит кон...	8	3	3	8	1.125000	9.0
1	Запустить собственную службу доставки, что сок...	2	5	4	10	2.000000	4.0
5	Добавить страницу отзывов клиентов о магазине,...	3	2	2	3	1.333333	4.0
4	Изменить цвет фона главной страницы, чтобы уве...	3	1	1	1	1.000000	3.0

На первом месте по методу RICE гипотеза №7 - 112 баллов, на 2-ом: гипотеза 2 и на 3-ем: гипотезы 0 и 6.

Гипотеза №7 по методу ICE занимает 3-е место и по методу RICE 1-е благодаря тому, что у неё при похожих значениях влияния, уверенности и требуемых усилий больший охват. Т.е. значение Reach - показатель количества пользователей, которое затронет изменение - 10 - максимальное из всех гипотез и сильно больше чем у гипотезы №8.
Если количество пользователей для нас важно, стоит предпочесть 7-ю гипотезу.

In [15]:
```
# гипотеза №7
hypothesis['Hypothesis'][7]
```

Out[15]:
```
'Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок'
```

In [16]:
```
# гипотеза №8
hypothesis['Hypothesis'][8]
```

Out[16]:
```
'Запустить акцию, дающую скидку на товар в день рождения'
```

Анализ A/B-теста

Выручка по группам

Создадим датафрейм с комбинацией дата-группа для каждого числа за август 2019-го.

In [17]:
```
# создаем датайфрейм с датами за август 2019
df = pd.DataFrame(pd.date_range('2019-8-1','2019-8-31'))
# задаем ключ для объединения
df['key'] = 1
df.rename({0: 'date'},axis=1,inplace=True)
# создаем названия групп
ab = pd.DataFrame({'group': ['A', 'B']})
# задаем ключ для объединения
ab['key'] = 1
ab
```

Out[17]:

	group	key
0	A	1
1	B	1

In [18]:
```
# объединяем даты с названиями групп
df = df.merge(ab , how='left',on='key')
df.drop('key', axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62 entries, 0 to 61
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    62 non-null     datetime64[ns]
 1   group   62 non-null     object        
dtypes: datetime64[ns](1), object(1)
memory usage: 1.5+ KB
```

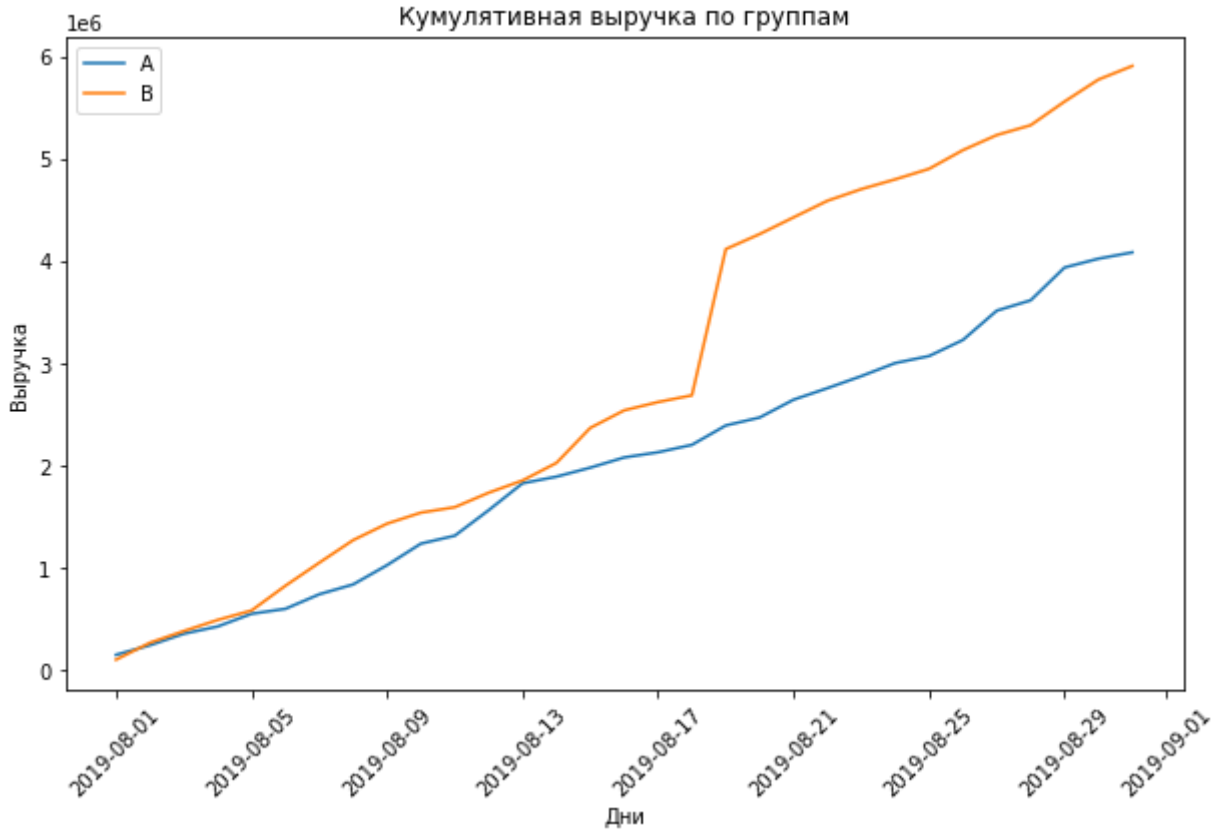
В таблице 62 строки: 31 день х на 2 группы. Всё верно.

In [19]:
```
# # создаем массив уникальных пар значений дат и групп теста
# datesGroups = orders[['date','group']].drop_duplicates()

# получаем агрегированные кумулятивные по дням данные о заказах
ordersAggregated = (df.apply(lambda x: orders[(orders['date'] <= x['date']) &
                                              (orders['group'] == x['group'])].agg(
    {'date' : 'max', 'group' : 'max', 'transactionId' : 'nunique', 'visitorId' : 'nunique', '
    .sort_values(by=['date','group']))

# получаем агрегированные кумулятивные по дням данные о посетителях интернет-магазина
visitorsAggregated = (df.apply(lambda x: visitors[(visitors['date'] <= x['date']) &
                                                  (visitors['group'] == x['group'])]
                                .agg({'date' : 'max', 'group' : 'max', 'visitors' : 'sum'
                                .sort_values(by=['date','group']))

# объединяем кумулятивные данные в одной таблице и присваиваем ее столбцам понятные названия
cumulativeData = ordersAggregated.merge(visitorsAggregated, left_on=['date', 'group'], right_
cumulativeData.columns = ['date', 'group', 'orders', 'buyers', 'revenue', 'visitors']

print(cumulativeData.head(5))
```

```
        date group  orders  buyers  revenue  visitors
0 2019-08-01     A      24      20   148579       719
1 2019-08-01     B      21      20   101217       713
2 2019-08-02     A      44      38   242401      1338
3 2019-08-02     B      45      43   266748      1294
4 2019-08-03     A      68      62   354874      1845
```

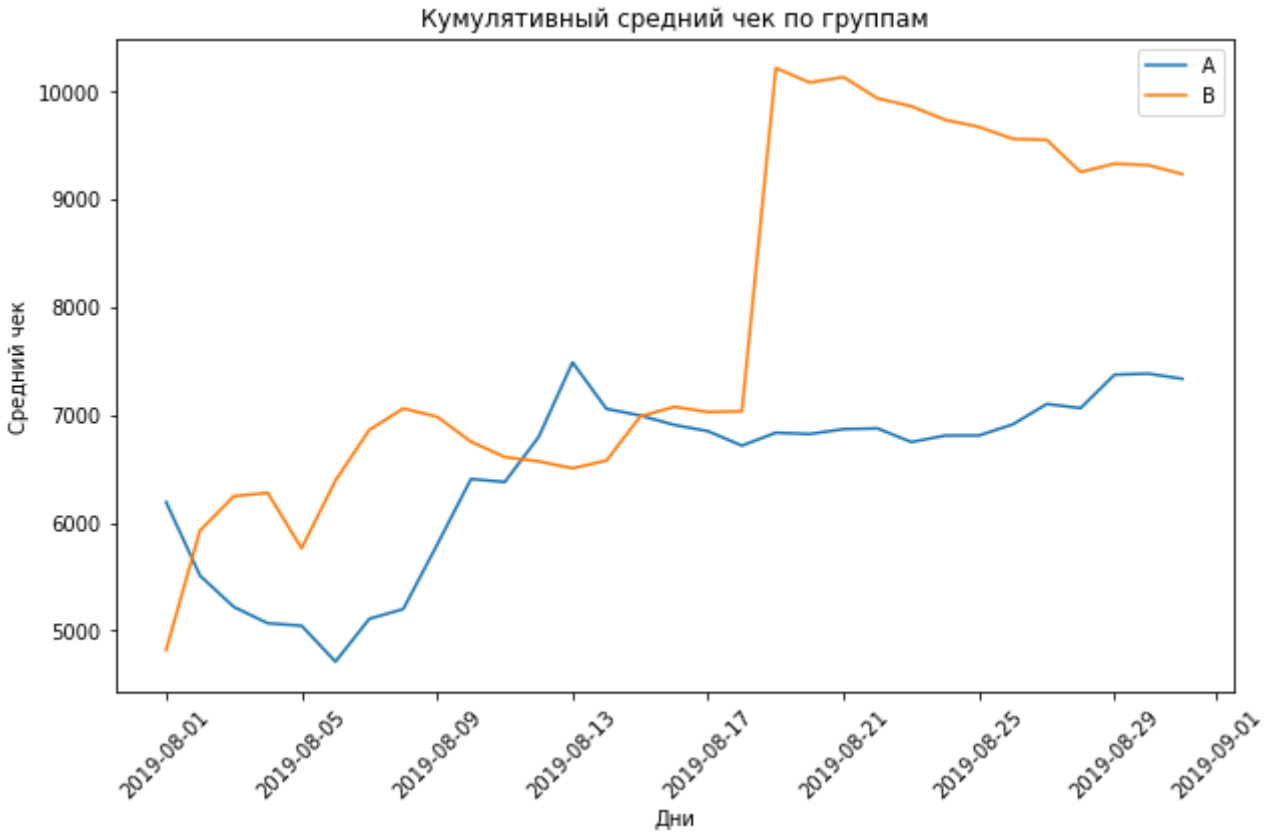
In [20]:
```
plt.rcParams["figure.figsize"] = [10, 6]
```

In [21]:
```
# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе A
cumulativeRevenueA = cumulativeData[cumulativeData['group']=='A'][['date','revenue', 'orders'

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B
cumulativeRevenueB = cumulativeData[cumulativeData['group']=='B'][['date','revenue', 'orders'

# Строим график выручки группы A
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue'], label='A')

# Строим график выручки группы B
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue'], label='B')
plt.title('Кумулятивная выручка по группам')
plt.rcParams["figure.figsize"] = [10, 6]
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Выручка')
plt.legend()
plt.show()
```



Выручка росла в обеих группах. В середине месяца кумулятивная выручка группы B стала превышать выручку группы A. Особенно резкий скачок был в один день. Возможно это была очень крупная покупка. Построим графики среднего чека по группам — разделим кумулятивную выручку на кумулятивное число заказов.

In [22]:
```
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']/cumulativeRevenueA['orders
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']/cumulativeRevenueB['orders
plt.title('Кумулятивный средний чек по группам')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Средний чек')
plt.legend()
plt.show()
```



Средний чек в группе B был выше в начале месяца, ближе к середине он снизился, но потом крупная продажа подняла его выше 10_000 и после этого он снова немного снизился, но остается выше 9000. Эта крупная покупка искажает результаты.
В группе A средний чек сначала рос, но с середины месяца он немного снизился и находится в районе 7000.

Построим график относительного различия для среднего чека.

In [23]:
```
# собираем данные в одном датафрейме
mergedCumulativeRevenue = cumulativeRevenueA.merge(cumulativeRevenueB, left_on='date', right_
```

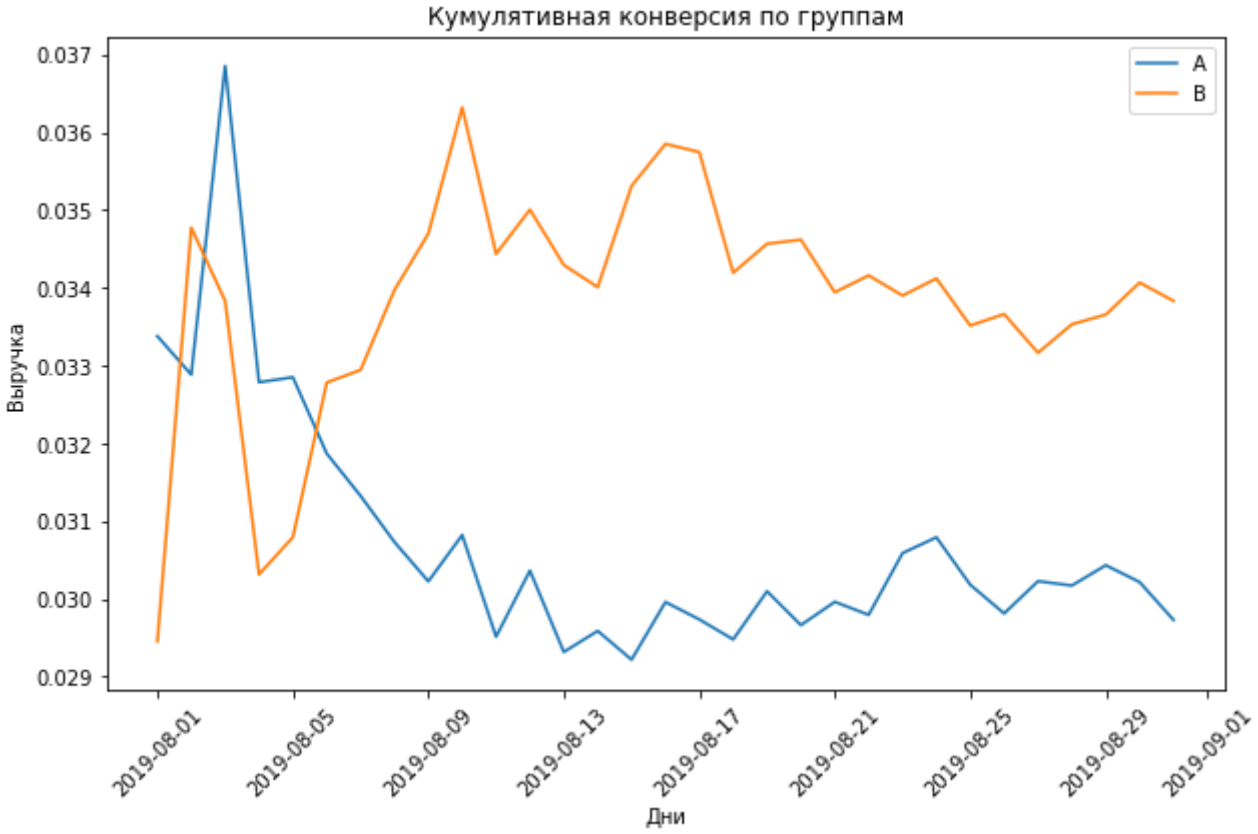
In [24]:
```
# строим отношение средних чеков
plt.plot(mergedCumulativeRevenue['date'], (mergedCumulativeRevenue['revenueB']/mergedCumulati

# добавляем ось X
plt.axhline(y=0, color='black', linestyle='--')
plt.title('Относительное различие среднего чека группы B и чека группы A')
# plt.rcParams["figure.figsize"] = [10, 6]
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('прирост ср. чека B к ср. чеку A')
plt.show()
```

Очень резко меняется отношение средних чеков между группами в первой трети месяца, в середине и в начале третьей декады. Очевидно в данных есть выбросы.

Конверсия по группам

Построим график кумулятивной конверсии.

In [25]:
```
# считаем кумулятивную конверсию
cumulativeData['conversion'] = cumulativeData['orders']/cumulativeData['visitors']

# отделяем данные по группе A
cumulativeDataA = cumulativeData[cumulativeData['group']=='A']

# отделяем данные по группе B
cumulativeDataB = cumulativeData[cumulativeData['group']=='B']

# строим графики
plt.plot(cumulativeDataA['date'], cumulativeDataA['conversion'], label='A')
plt.plot(cumulativeDataB['date'], cumulativeDataB['conversion'], label='B')
plt.legend()

# # задаем масштаб осей
# plt.axis(['2019-08-01','2019-08-31', 0, 0.05])
plt.title('Кумулятивная конверсия по группам')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Выручка')
plt.show()
```



Конверсия группы B выросла в первой трети месяца и с некоторыми рывками постепенно снижалась. Конверсия группы A упала в начале, начала небольшой рост в середине месяца и остается низкой в конце. Оба показателя еще подвижны, но заметно преимущество группы B.

Построим график относительного различия кумулятивных конверсий:

In [26]:
```
mergedCumulativeConversions = (cumulativeDataA[['date','conversion']]
.merge(cumulativeDataB[['date','conversion']], left_on='date', right_on='date', how='left', su
```

In [27]:
```
plt.plot(mergedCumulativeConversions['date'],
         mergedCumulativeConversions['conversionB']/mergedCumulativeConversions['conversionA'
         label="Относительный прирост конверсии группы B относительно группы A")
plt.legend()

plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=-0.1, color='grey', linestyle='--')
plt.title('Относительное различие кумулятивных конверсий группы B к группе A')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Отношение конверсий')
plt.show()
```

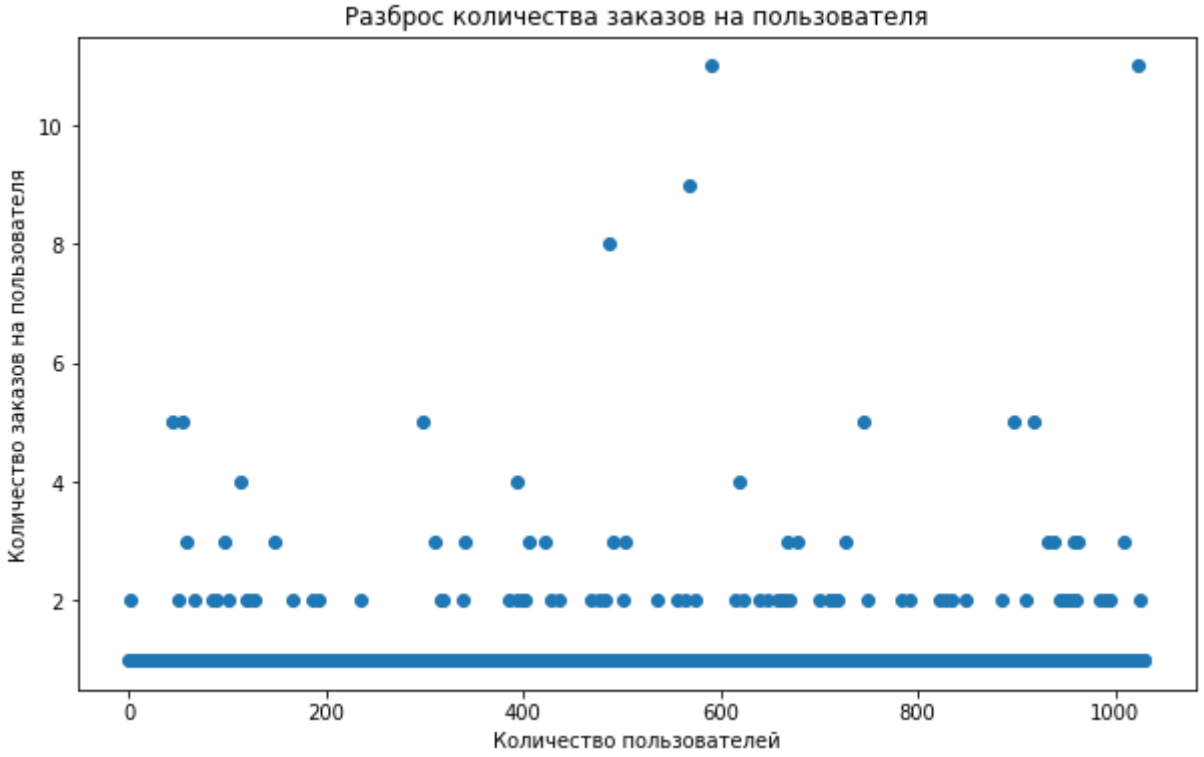
Конверсия группы B очень хорошо выросла относительно конверсии A в конце первой недели, с тех пор она постепенно с переменным успехом снижается. Но все равно она больше конверсии A как минимум на 10% в самой низкой точке и начала расти в конце месяца. Тут преимущество конверсии группы B очевидно.

Анализ выбросов

Количество заказов на покупателя

Посмотрим распределение числа заказов на одного покупателя.

In [28]:
```
# считаем количество заказов по пользователям
ordersByUsers = (
    orders.groupby('visitorId', as_index=False)
    .agg({'transactionId': 'nunique'})
)
ordersByUsers.columns = ['visitorId', 'orders']

# Построим точечную диаграмму числа заказов на одного пользователя:
x_values = pd.Series(range(0,len(ordersByUsers)))
plt.scatter(x_values, ordersByUsers['orders'])
plt.title('Разброс количества заказов на пользователя')
plt.xlabel('Количество пользователей')
plt.ylabel('Количество заказов на пользователя')
plt.show()
```



Похоже, что чаще всего покупатель делает один заказ, иногда два, редко три или больше. Чтобы точно определить границы выбросов найдем перцентили.

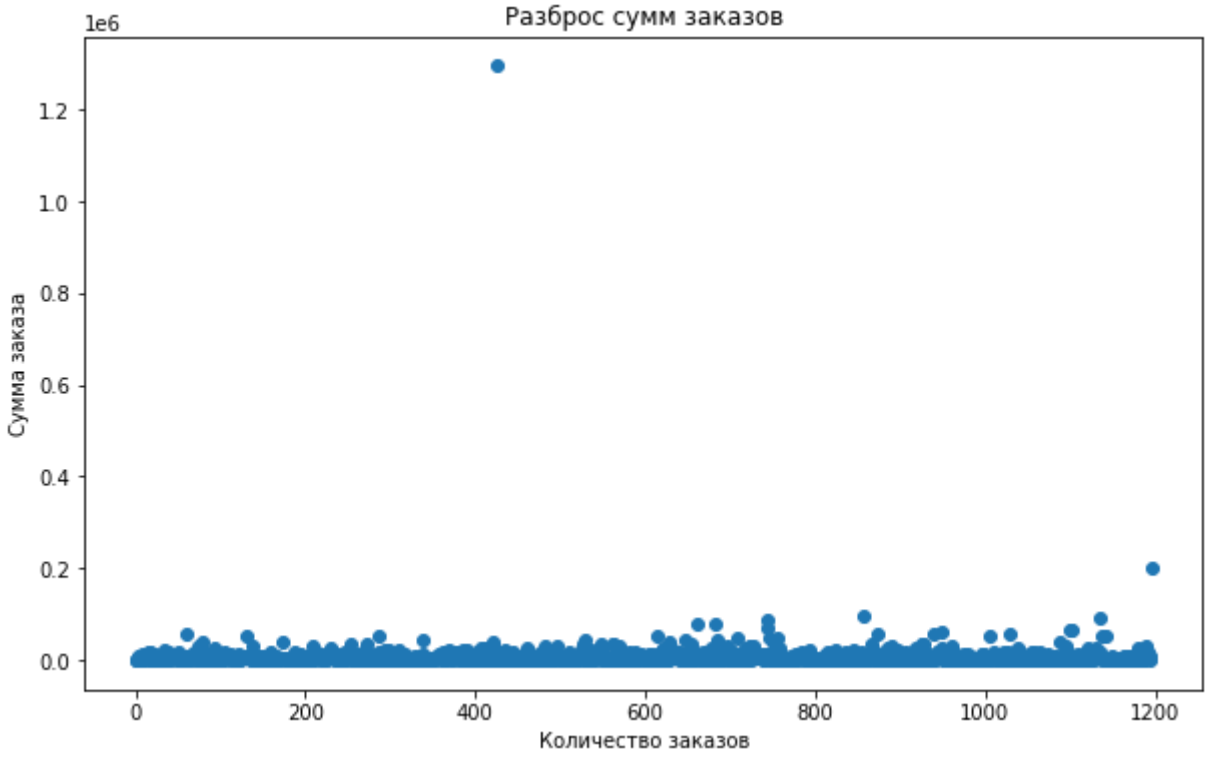
In [29]:
```
np.percentile(ordersByUsers['orders'], [95, 99])
```

Out[29]:
```
array([2., 4.])
```

95% покупателей делает 2 и менее заказов и только 1% делает 4 или больше заказа. Можем считать всех пользователей, кто покупает более 3-х раз, аномальными.

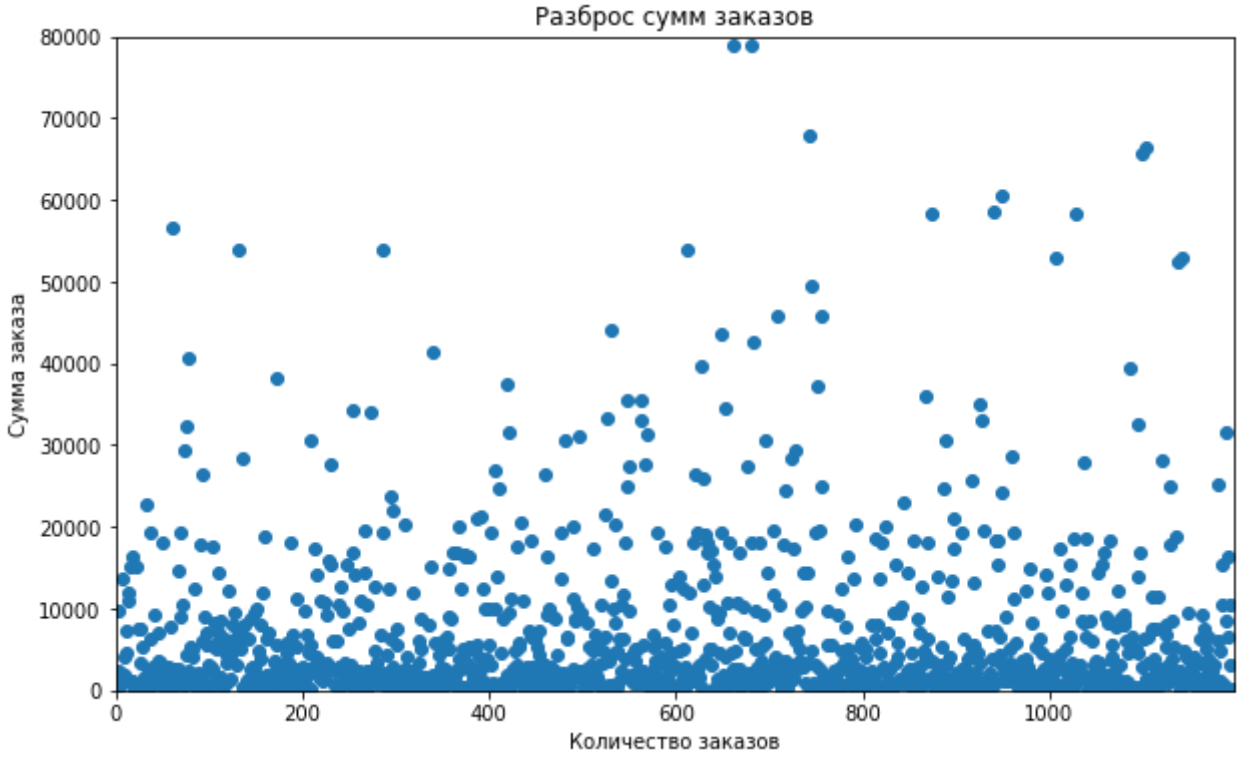
Сумма заказа

In [30]:
```
# Построим точечную диаграмму сумм заказов:
x_values = pd.Series(range(0,len(orders)))
plt.scatter(x_values, orders['revenue'])
plt.title('Разброс сумм заказов')
plt.xlabel('Количество заказов')
plt.ylabel('Сумма заказа')
plt.show()
```



Мы видим, что вся масса заказов находится в диапазоне до 100000, но из-за выбросов график стал нечитаемым. Сократим размер отображаемых заказов, чтобы посмотреть более детально.

In [31]:
```
# Построим точечную диаграмму сумм заказов:
x_values = pd.Series(range(0,len(orders)))
plt.scatter(x_values, orders['revenue'])
plt.title('Разброс сумм заказов')
# ограничим верхнюю границу суммы заказа, чтобы график был более подробным
plt.axis([0, len(orders), 0, 80000])
plt.xlabel('Количество заказов')
plt.ylabel('Сумма заказа')
plt.show()
```



Большая часть заказов по сумме меньше 20000. Найдем точные значения выбросов с помощью перцентилей.

In [32]:
```
np.percentile(orders['revenue'], [95, 99])
```

Out[32]:
```
array([28000. , 58233.2])
```

Только 5% заказов превышают 28000 и есть 1% заказов на суммы свыше 58000. Примем за аномальные заказы свыше 30000.

Анализ A/B теста

Конверсия по сырым данным

Посчитаем количество посетителей в таблице с заказами.

In [33]:
```
len(orders['visitorId'].unique())
```

Out[33]:
```
1031
```

In [34]:
```
visitorsA = orders[orders['group'] == 'A']['visitorId'].unique()
visitorsB = orders[orders['group'] == 'B']['visitorId'].unique()
print('В группе A {} посетителей, в группе B {}. В сумме {}.'.format(len(visitorsA), len(visi
```

```
В группе A 503 посетителей, в группе B 586. В сумме 1089.
```

Похоже, что посетители в группах действительно пересекаются, т.к. их сумма больше числа посетителей в таблице `orders`. Очистим от них датафрейм с заказами.

In [35]:
```
# убираем посетителей, которые есть в обеих группах одновременно
orders = orders[~(orders['visitorId'].isin(visitorsB) & orders['visitorId'].isin(visitorsA))]
# считаем итоговое количество уникальных посетителей
len(orders['visitorId'].unique())
```

Out[35]:
```
973
```

In [36]:
```
visitorsA = orders[orders['group'] == 'A']['visitorId'].unique()
visitorsB = orders[orders['group'] == 'B']['visitorId'].unique()
print('В группе A {} посетителей, в группе B {}. В сумме {}.'.format(len(visitorsA), len(visi
```

```
В группе A 445 посетителей, в группе B 528. В сумме 973.
```

Очистили таблицу от посетителей, попавших в обе группы. Их стало меньше на 58.

Проверим изменилось ли количество покупателей в группе B относительно группы A. Для этого посчитаем количество покупателей в каждой группе и сравним с количеством посетителей по группам.

In [37]:
```
# находим общее количество покупателей для каждой группы
orders_grouped = orders.groupby('group', as_index=False).agg(
orders = ('transactionId' , 'nunique'), buyers = ('visitorId', 'nunique'), revenue = ('revenu
)
orders_grouped
```

Out[37]:

	group	orders	buyers	revenue
0	A	468	445	3364656
1	B	548	528	5068972

In [38]:
```
# находим общее количество посетителей для каждой группы
visitors_gr = visitors.groupby('group')['visitors'].sum()
visitors_gr
```

Out[38]:
```
group
A    18736
B    18916
Name: visitors, dtype: int64
```

In [39]:
```
# пропорция покупателей в группе A:
p1 = orders_grouped['buyers'][0] / visitors_gr[0]
print('пропорция покупателей в группе A: {:.2%}'.format(p1))

# пропорция покупателей в группе B:
p2 = orders_grouped['buyers'][1] / visitors_gr[1]
print('пропорция покупателей в группе B: {:.2%}'.format(p2))

# пропорция покупателей в комбинированном датасете:
p_combined = (orders_grouped['buyers'][0] + orders_grouped['buyers'][1]) / (visitors_gr[0] + v
print('пропорция покупателей во всем датасете: {:.2%}'.format(p_combined))

#разница пропорций в датасетах
difference = p1 - p2
print('разница пропорций в группах: {:.2%}'.format(difference))
```

```
пропорция покупателей в группе A: 2.38%
пропорция покупателей в группе B: 2.79%
пропорция покупателей во всем датасете: 2.58%
разница пропорций в группах: -0.42%
```

Похоже, что доля покупателей в группе B выше. Проверим, можно ли считать эту разницу статистически значимой.

Примем за нулевую гипотезу то, что доли покупателей в группах равны.

Альтернативная гипотеза: конверсия в группах отличается.

In [40]:
```
#считаем статистику в ст.отклонениях стандартного нормального распределения
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/visitors_gr[0] + 1/visitors

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

alpha = .05 # критический уровень статистической значимости
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:  0.01093208956553604
Отвергаем нулевую гипотезу: между долями есть значимая разница
```

In [41]:
```
print('Конверсия группы B больше конверсии группы A на {:.2%}'.format(p2/p1 - 1 ))
```

```
Конверсия группы B больше конверсии группы A на 17.52%
```

P-значение равно 1,67%, что меньше стат значимости (5%), значит можно сказать, что конверсия групп A и B отличается.
По неочищенным данным конверсия группы B 3,1%, группы A - 2,68%. Конверсия группы B выше конверсии группы A на 15%.

Средний чек по сырым данным

Посчитаем средние чеки по группам и сравним их.

In [42]:
```
# считаем средние чеки
print('Средний чек по группе B:', round(orders[orders['group']=='B']['revenue'].mean()))
print('Средний чек по группе A:', round(orders[orders['group']=='A']['revenue'].mean()))
# считаем относительные различия:
print('Средний чек группы B превышает средний чек группы A на {0:.1%}'.format(orders[orders['
```

```
Средний чек по группе B: 9250
Средний чек по группе A: 7189
Средний чек группы B превышает средний чек группы A на 28.7%
```

Неплохой результат, похоже, что средний чек хорошо вырос. По крайней мере по неочищенным от выбросов данным.
Проверим статзначимость этих различий с помощью теста Манна-Уитни.

Проверяем отличаются ли средние чеки двух групп. Если полученный параметр будет в пределах выбранного значения статзначимости, то мы сможем отбросить нулевую гипотезу.
Нулевая гипотеза: средние значения двух выборок равны,
альтернативная: средние двух выборок отличаются.
Значение стат значимости α = 0,05.

In [43]:
```
print('P-value равно {0:.3f}.'.format(st.mannwhitneyu(orders[orders['group']=='A']['revenue'],
                        orders[orders['group']=='B']['revenue'])[1]))
```

```
P-value равно 0.829.
```

P-value очень высоко, при таком значении мы не можем отбросить гипотезу о равенстве средних чеков, несмотря на сравнение абсолютных величин. Видимо на них сильно отразились очень высокие суммы нетипичных заказов.

Конверсия по очищенным данным

In [44]:
```
# выделяем покупателей с большим количеством заказов
usersWithManyOrders = ordersByUsers[ordersByUsers['orders'] > 3]['visitorId']
# выделяем покупателей с крупными заказами
usersWithExpensiveOrders = orders[orders['revenue'] > 30000]['visitorId']
# соединяем списки
abnormalUsers = (
    pd.concat([usersWithManyOrders, usersWithExpensiveOrders], axis=0)
    .drop_duplicates()
    .sort_values()
)
print(abnormalUsers.head(5))
print(abnormalUsers.shape[0])
```

```
1099    148427295
44      199603092
928     204675465
55      237748145
684     358944393
Name: visitorId, dtype: int64
55
```

Мы выделили список нетипичных покупателей, их получилось 55 из общего числа покупателей 973.

In [45]:
```
# убираем нетипичных покупателей
orders_filtered = orders[~orders['visitorId'].isin(abnormalUsers)]
print('Удалили из таблицы orders', len(orders) - len(orders_filtered), 'строк.')
```

```
Удалили из таблицы orders 43 строк.
```

Посчитаем конверсию по группам на очищенных данных.

In [46]:
```
# находим общее количество покупателей для каждой группы
orders_grouped_f = orders_filtered.groupby('group', as_index=False).agg(
orders_filtered = ('transactionId' , 'nunique'), buyers = ('visitorId', 'nunique'), revenue =
)
orders_grouped_f
```

Out[46]:

	group	orders_filtered	buyers	revenue
0	A	448	426	2385548
1	B	525	505	2709781

In [47]:
```
# общее количество посетителей для каждой группы мы уже считали
visitors_gr
```

Out[47]:
```
group
A    18736
B    18916
Name: visitors, dtype: int64
```

In [48]:
```
# пропорция покупателей в группе A:
p1 = orders_grouped_f['buyers'][0] / visitors_gr[0]
print('пропорция покупателей в группе A: {:.2%}'.format(p1))

# пропорция покупателей в группе B:
p2 = orders_grouped_f['buyers'][1] / visitors_gr[1]
print('пропорция покупателей в группе B: {:.2%}'.format(p2))

# пропорция покупателей в комбинированном датасете:
p_combined = (orders_grouped_f['buyers'][0] + orders_grouped_f['buyers'][1]) / (visitors_gr[0
print('пропорция покупателей во всем датасете: {:.2%}'.format(p_combined))

#разница пропорций в датасетах
difference = p1 - p2
print('разница пропорций в группах: {:.2%}'.format(difference))
```

```
пропорция покупателей в группе A: 2.27%
пропорция покупателей в группе B: 2.67%
пропорция покупателей во всем датасете: 2.47%
разница пропорций в группах: -0.40%
```

Конверсия в группе B несколько выше. Посчитаем статистику в стандартных отклонениях стандартного нормального распределения. Нулевая гипотеза: конверсии групп равны, альтернативная: конверсии отличаются.

In [49]:
```
#считаем статистику в ст.отклонениях стандартного нормального распределения
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/visitors_gr[0] + 1/visitors

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

alpha = .05 # критический уровень статистической значимости
p_value = (1 - distr.cdf(abs(z_value))) * 2

print('p-значение: ', p_value)

if (p_value < alpha):
    print("Отвергаем нулевую гипотезу: между конверсиями есть значимая разница")
else:
    print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать конверсии разными"
```

```
p-значение:  0.013358748838567358
Отвергаем нулевую гипотезу: между конверсиями есть значимая разница
```

In [50]:
```
print('Конверсия группы B больше конверсии группы A на {:.2%}'.format(p2/p1 - 1 ))
```

```
Конверсия группы B больше конверсии группы A на 17.42%
```

P-значение после очистки данных немного увеличилось - 1,79% (было 1,67%), но оно всё ещё меньше 5% (стат значимости), а значит можно сказать, что конверсии групп отличаются.
После очистки данных конверсии обеих групп стали меньше, но превышение конверсии B относительно A чуть увеличилось на 0,5% до 15,88%.

Средний чек по очищенным данным

Посчитаем средние чеки по группам и сравним их.

In [51]:
```
# считаем средние чеки
print('Средний чек по группе B:', round(orders_filtered[orders_filtered['group']=='B']['revenu
print('Средний чек по группе A:', round(orders_filtered[orders_filtered['group']=='A']['revenu
# считаем относительные различия:
print('Средний чек группы B превышает средний чек группы A на {0:.1%}'.format(orders_filtered
```

```
Средний чек по группе B: 5161
Средний чек по группе A: 5325
Средний чек группы B превышает средний чек группы A на -3.1%
```

После того как мы убрали нетипичных покупателей, средний чек по группе B стал меньше среднего чека по группе A на 3,1%.
Проверим статзначимость этих различий с помощью теста Манна-Уитни.

Нулевая гипотеза: средние чеки двух выборок равны,
альтернативная: средние чеки двух выборок отличаются.
Значение стат значимости α = 0,05.

In [52]:
```
print('p-value равно {0:.3f}.'.format(st.mannwhitneyu(orders_filtered[orders_filtered['group']
                        orders_filtered[orders_filtered['group']=='B']['revenue'])[1]))
```

`p-value равно 0.842.`

После очистки данных p-значение стало ещё выше, а средний чек группы B оказался меньше среднего чека группы A, но не значительно, разница около 3%. А значит, мы не можем утверждать, что в группе B чек отличается от среднего чека группы A.

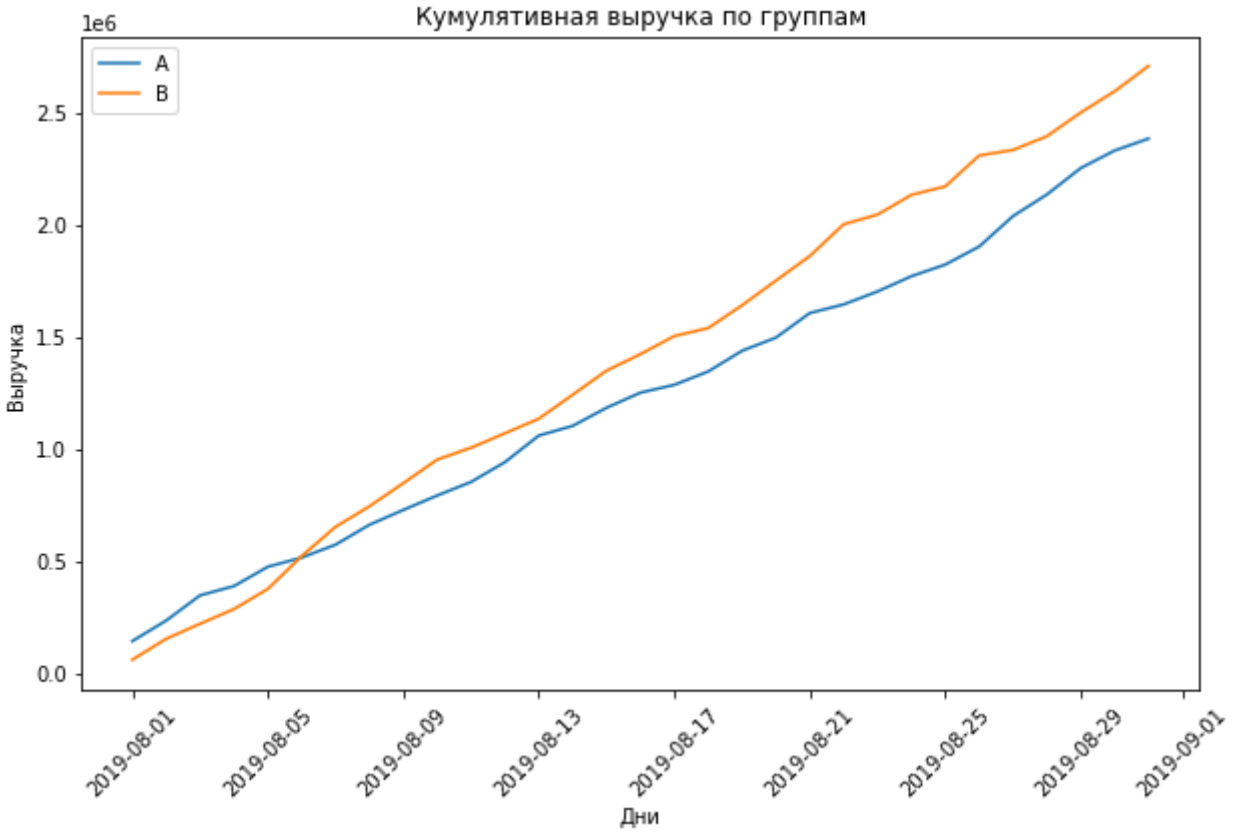
Анализ A/B теста после очистки данных

График кумулятивной выручки и среднего чека

Построим заново графики кумулятивной выручки на очищенных данных.

In [53]:
```
# получаем агрегированные кумулятивные по дням данные о заказах
ordersAggregated = (df.apply(lambda x: orders_filtered[(orders_filtered['date'] <= x['date']) 
                                                       (orders_filtered['group'] == x['group'])].agg(
    {'date' : 'max', 'group' : 'max', 'transactionId' : 'nunique', 'visitorId' : 'nunique', '
    .sort_values(by=['date','group']))

# получаем агрегированные кумулятивные по дням данные о посетителях интернет-магазина
visitorsAggregated = (df.apply(lambda x: visitors[(visitors['date'] <= x['date']) &
                                                  (visitors['group'] == x['group'])]
                                        .agg({'date' : 'max', 'group' : 'max', 'visitors' : 'sum'
                                        .sort_values(by=['date','group']))

# объединяем кумулятивные данные в одной таблице и присваиваем ее столбцам понятные названия
cumulativeData = ordersAggregated.merge(visitorsAggregated, left_on=['date', 'group'], right_
cumulativeData.columns = ['date', 'group', 'orders', 'buyers', 'revenue', 'visitors']
```

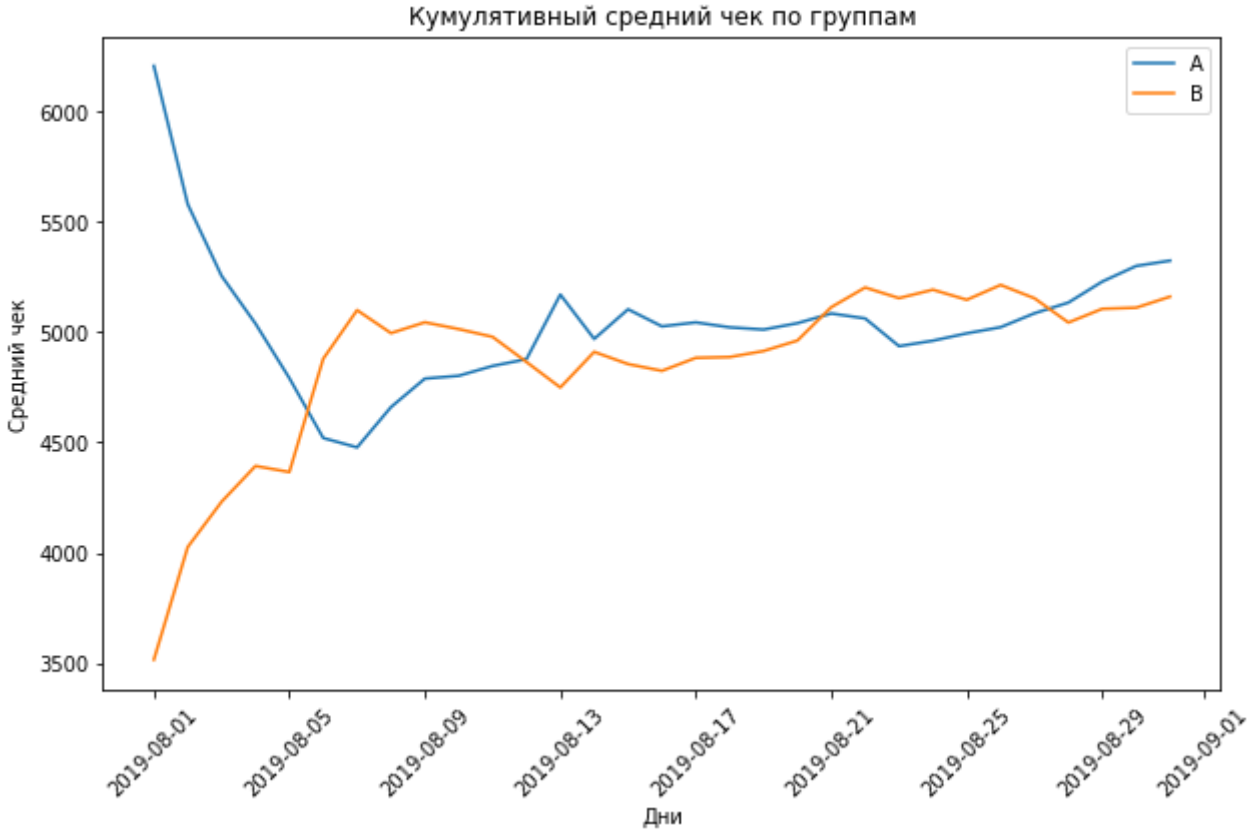
In [54]:
```
# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе A
cumulativeRevenueA = cumulativeData[cumulativeData['group']=='A'][['date','revenue', 'orders'

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B
cumulativeRevenueB = cumulativeData[cumulativeData['group']=='B'][['date','revenue', 'orders'

# Строим график выручки группы A
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue'], label='A')

# Строим график выручки группы B
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue'], label='B')
plt.title('Кумулятивная выручка по группам')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Выручка')
plt.legend()
plt.show()
```



Выручка группы B начала опережать выручку группы A уже через неделю и с тех пор увеличивает разрыв.

Построим графики среднего чека по группам — разделим кумулятивную выручку на кумулятивное число заказов.

In [55]:
```
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']/cumulativeRevenueA['orders
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']/cumulativeRevenueB['orders
plt.title('Кумулятивный средний чек по группам')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Средний чек')
plt.legend()
plt.show()
```

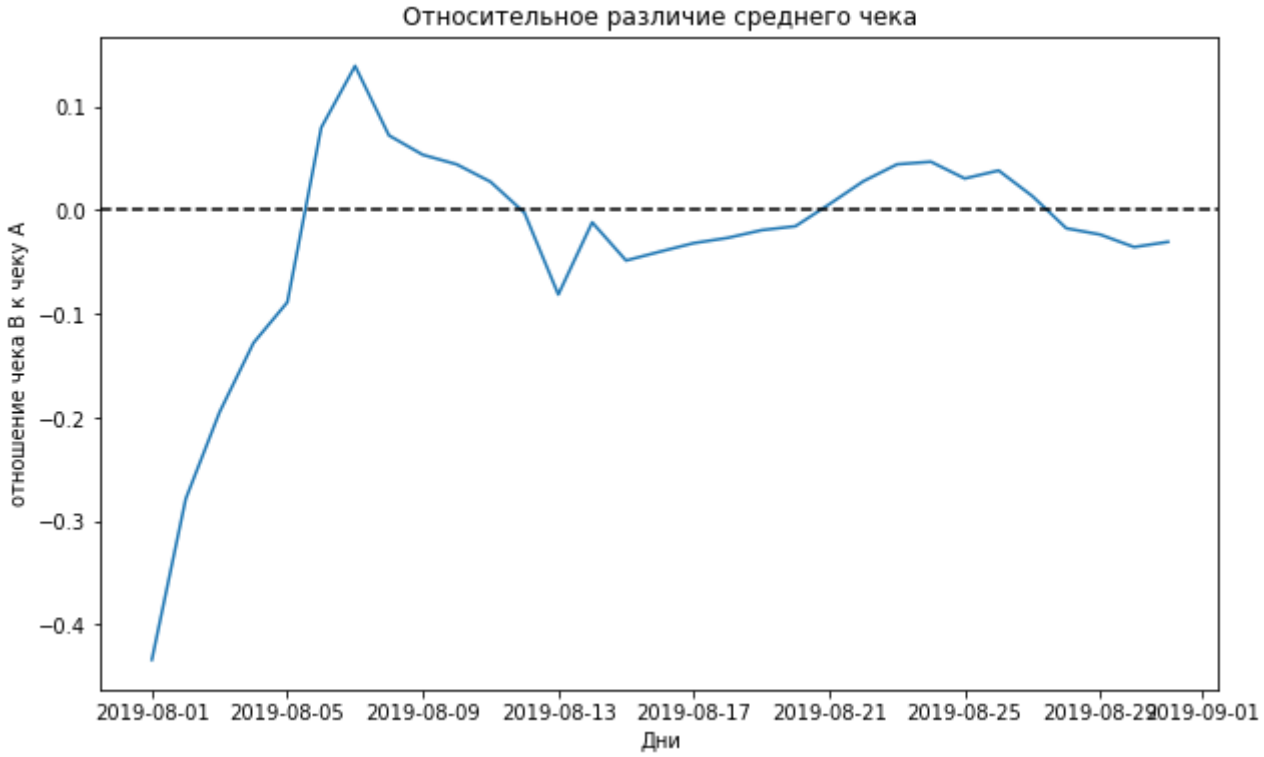


Средние чеки в группах попеременно обгоняют друг друга, можно сказать, что они сближаются в районе 5200-5300.

Построим график относительного различия для среднего чека.

In [56]:
```
# собираем данные в одном датафрейме
mergedCumulativeRevenue = cumulativeRevenueA.merge(cumulativeRevenueB, left_on='date', right_
```

In [57]:
```
# строим отношение средних чеков
plt.plot(mergedCumulativeRevenue['date'], (mergedCumulativeRevenue['revenueB']/mergedCumulati

# добавляем ось X
plt.axhline(y=0, color='black', linestyle='--')
plt.title('Относительное различие среднего чека')
plt.xlabel('Дни')
plt.ylabel('отношение чека B к чеку A')
plt.show()
```

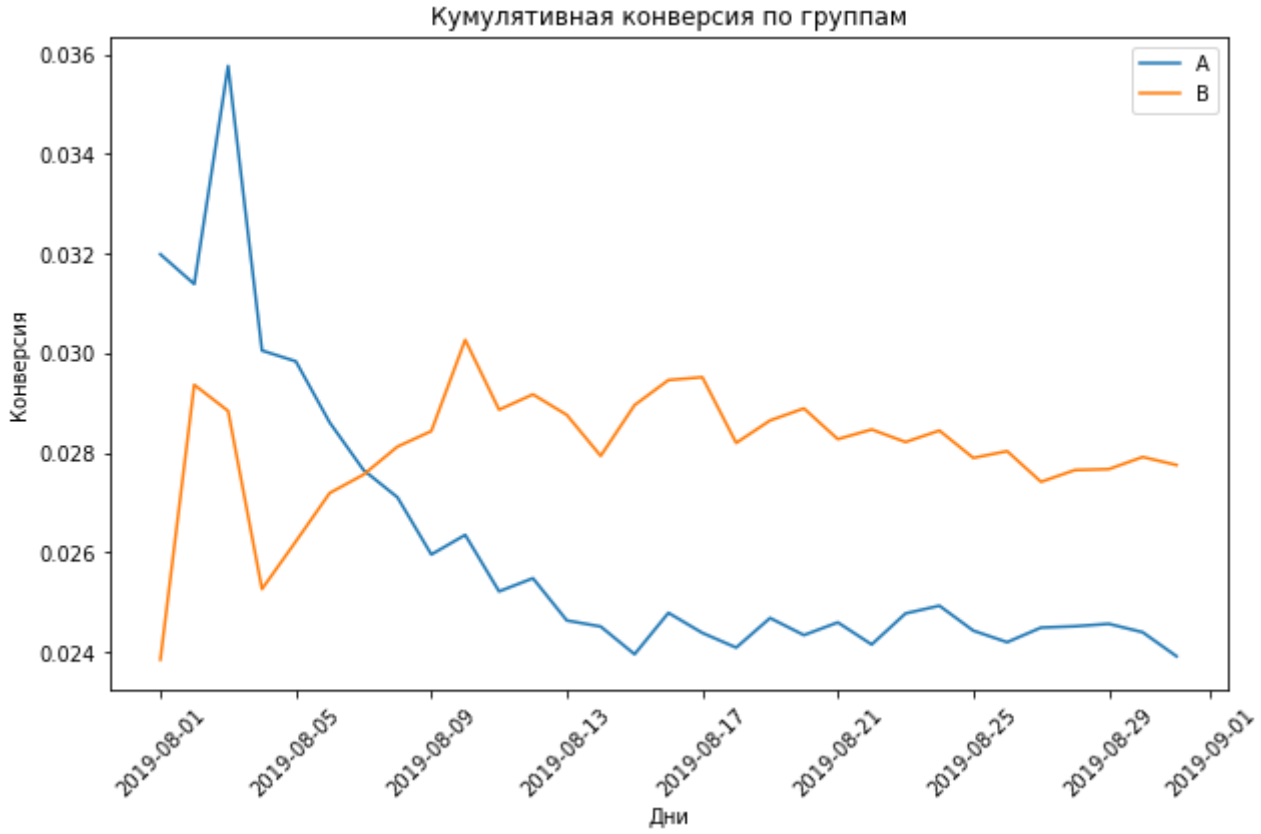


Отношение средних чеков между группами колеблется вокруг 0 от -10% до +10% и похоже стабилизируется ближе к 0.

График кумулятивной конверсии

Построим график кумулятивной конверсии.

In [58]:
```
# считаем кумулятивную конверсию
cumulativeData['conversion'] = cumulativeData['orders']/cumulativeData['visitors']

# отделяем данные по группе A
cumulativeDataA = cumulativeData[cumulativeData['group']=='A']

# отделяем данные по группе B
cumulativeDataB = cumulativeData[cumulativeData['group']=='B']

# строим графики
plt.plot(cumulativeDataA['date'], cumulativeDataA['conversion'], label='A')
plt.plot(cumulativeDataB['date'], cumulativeDataB['conversion'], label='B')
plt.legend()
plt.title('Кумулятивная конверсия по группам')
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('Конверсия')
# # задаем масштаб осей
# plt.axis(['2019-08-01','2019-08-31', 0, 0.05])
plt.show()
```



Конверсия группы B выросла в начале месяца и обогнала конверсию группы A уже в первую неделю и с тех пор более-менее стабильно находится в районе 2,8%. Конверсия группы A снизилась к середине августа до 2,4% и продолжает оставаться на этом уровне.
Конверсия группы B стабильно лучше конверсии группы A по очищенным данным.

Построим график относительного различия кумулятивных конверсий:

In [59]:
```
mergedCumulativeConversions = (cumulativeDataA[['date','conversion']]
.merge(cumulativeDataB[['date','conversion']], left_on='date', right_on='date', how='left', su
```

In [60]:
```
plt.plot(mergedCumulativeConversions['date'],
         mergedCumulativeConversions['conversionB']/mergedCumulativeConversions['conversionA']

plt.title("Относительный прирост конверсии группы B относительно группы A")
plt.xticks(rotation=45)
plt.xlabel('Дни')
plt.ylabel('прирост конверсии')
plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=-0.1, color='grey', linestyle='--')
plt.show()
```

Конверсия группы B очень хорошо обогнала группу A к середине месяца, на 15-20%, чуть снизилось их соотношение к концу, но все равно она больше конверсии A как минимум на 10%. Преимущество конверсии группы B очевидно и на очищенных данных.

Вывод

При анализе в данных мы обнаружили нетипично высокие заказы, а также покупателей с количеством покупок больше нормального. Эти покупки выразились в резком росте среднего чека в группе B в середине месяца. Чтобы исключить их влияние на результаты теста мы очистили данные от покупателей с 4 и более заказами, а также тех, кто покупал на суммы свыше 30000. Это 55 покуателей из тысячи.
Также пришлось убрать из анализа данные 58 покупателей, которые попали в обе группы.
В результате анализа A/B теста мы можем утверждать, что средняя конверсия группы B (2,67%) выше конверсии группы A (2,27%) на 17%. Средний чек в группе B (5161) не настолько отличается о чека группы A (5325) чтобы считать это отличие статистически значимым.
При более высокой конверсии и одинаковом среднем чеке покупатели группы B приносят больше выручки чем покупатели группы A, что можно видеть на графике кумулятивной выручки.
Думаю, что тест можно остановить и зафиксировать преимущество группы B.