Проект: "Принятие решений в бизнесе"

Описание проекта

Интернет-магазин:
Вместе с отделом маркетинга подготовлен список гипотез для увеличения выручки, были получены результаты A/B-теста.

Задачи:

- приоритизировать гипотезы;
- запустить A/B-тест;
- проанализировать результаты.

Оглавление

1 Часть 1. Приоритизация гипотез

Задачи:

1. Применить фреймворк ICE для приоритизации гипотез.
2. Применить фреймворк RICE для приоритизации гипотез.
3. Указать, как изменилась приоритизация гипотез при применении RICE вместо ICE.

Файл `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.

In [1]:
```python
# Импорт библиотек
import pandas as pd
import scipy.stats as stats
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
# Чтение файлов
try:
    df = pd.read_csv('/datasets/hypothesis.csv')
# если не получилось прочитать файл из локальной папки, то загружаем данные из сети
except:
    df = pd.read_csv('datasets/hypothesis.csv')
```

In [3]:
```python
# Снимаем ограничение на вывод всех символов в записях
pd.set_option('display.max_colwidth', None)
```

In [4]:
```
df
```

Out[4]:

	Hypothesis	Reach	Impact	Confidence	Efforts
0	Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей	3	10	8	6
1	Запустить собственную службу доставки, что сократит срок доставки заказов	2	5	4	10
2	Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа	8	3	7	3
3	Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар	8	3	3	8
4	Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей	3	1	1	1
5	Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов	3	2	2	3
6	Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию	5	3	8	3
7	Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок	10	7	8	5
8	Запустить акцию, дающую скидку на товар в день рождения	1	9	9	5

1.1 Метод ICE

$$\text{ICE SCORE} = \frac{\text{Impact} \times \text{Confidence}}{\text{Efforts}}$$

In [5]:
```
# Создаем копию df для работы
df_ice = df[['Hypothesis', 'Impact','Confidence','Efforts']].copy()
```

In [6]:
```
# Расчёт ICE
df_ice['ICE'] = round(df_ice['Impact'] * df_ice['Confidence'] / df_ice['Efforts'], 2)

# Сортировка по ICE
df_ice = df_ice.sort_values(by='ICE', ascending=False)

# Вывод df_ice
df_ice
```

Out[6]:

	Hypothesis	Impact	Confidence	Efforts	ICE
8	Запустить акцию, дающую скидку на товар в день рождения	9	9	5	16.20
0	Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей	10	8	6	13.33
7	Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок	7	8	5	11.20
6	Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию	3	8	3	8.00
2	Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа	3	7	3	7.00
1	Запустить собственную службу доставки, что сократит срок доставки заказов	5	4	10	2.00
5	Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов	2	2	3	1.33
3	Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар	3	3	8	1.12
4	Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей	1	1	1	1.00

ВЫВОД:
Фреймворк ICE рекомендует начинать с 8-ой гипотезы. Последней рассматривается 4-ая.

1.2 Метод RICE

$$\text{RICE SCORE} = \frac{\text{Reach} \times \text{Impact} \times \text{Confidence}}{\text{Efforts}}$$

In [7]:
```
# Создаем копию df для работы
df_rice = df.copy()
```

In [8]:
```
# Расчёт RICE↔
```

Out[8]:

	Hypothesis	Reach	Impact	Confidence	Efforts	RICE
7	Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок	10	7	8	5	112.0
2	Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа	8	3	7	3	56.0
0	Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей	3	10	8	6	40.0
6	Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию	5	3	8	3	40.0
8	Запустить акцию, дающую скидку на товар в день рождения	1	9	9	5	16.2
3	Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар	8	3	3	8	9.0
1	Запустить собственную службу доставки, что сократит срок доставки заказов	2	5	4	10	4.0
5	Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов	3	2	2	3	4.0
4	Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей	3	1	1	1	3.0

ВЫВОД:
Фреймворк RICE рекомендует начинать с 7-ой гипотезы. Последней рассматривается 4-ая.

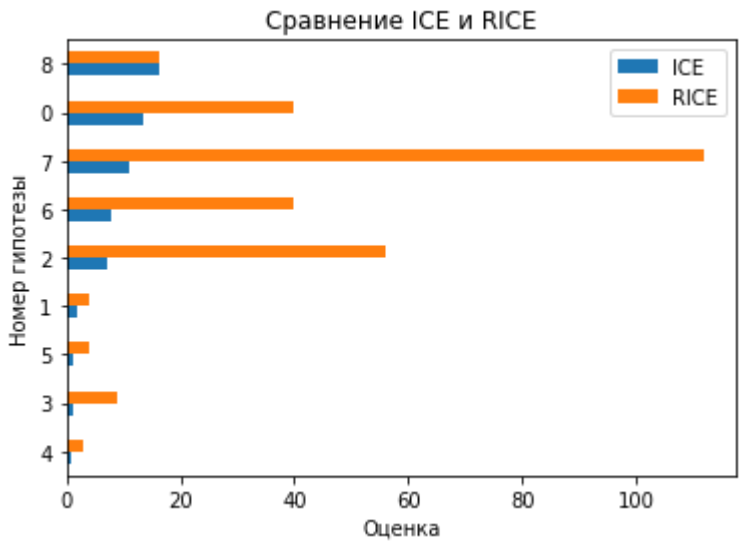
Соединим таблицы

In [9]:
```
# Объединение таблиц
df_ice.merge(df_rice)[['Hypothesis','ICE','RICE']]\
.set_axis(df_ice.index)\
.sort_values(by=['ICE','RICE'], ascending=False)
```

Out[9]:

	Hypothesis	ICE	RICE
8	Запустить акцию, дающую скидку на товар в день рождения	16.20	16.2
0	Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей	13.33	40.0
7	Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок	11.20	112.0
6	Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию	8.00	40.0
2	Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа	7.00	56.0
1	Запустить собственную службу доставки, что сократит срок доставки заказов	2.00	4.0
5	Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов	1.33	4.0
3	Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар	1.12	9.0
4	Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей	1.00	3.0

In [10]:
```
df_ice.merge(df_rice)[['Hypothesis','ICE','RICE']]\
.set_axis(df_ice.index)\
.sort_values(by=['ICE','RICE'], ascending=True).plot(kind='barh', title='Сравнение ICE и RICE');
plt.xlabel("Оценка")
plt.ylabel("Номер гипотезы");
```



Выводы:

Два фреймворка производят приоритизацию по-разному. Вариант RICE учитывает охват пользователей (REACH) - данный показатель поднимает оценки показателей (в данном случае).

Если есть возможноть оценить охват пользователей, то оптимальнее использовать RICE, однако, при одинаковом охвате достаточно ICE.

2 Часть 2. Анализ A/B-теста

Задача:

Проанализировать A/B-тест.

Файл `/datasets/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.

Файл `/datasets/visitors.csv` :

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

In [11]:
```
# Чтение файлов
try:
    orders = pd.read_csv('/datasets/orders.csv')
    visitors = pd.read_csv('/datasets/visitors.csv')
# если не получилось прочитать файл из локальной папки, то загружаем данные из сети
except:
    orders = pd.read_csv('datasets/orders.csv')
    visitors = pd.read_csv('datasets/visitors.csv')
```

Выполним предобработку данных:

In [12]:
```
orders.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   object
 3   revenue        1197 non-null   int64 
 4   group          1197 non-null   object
dtypes: int64(3), object(2)
memory usage: 46.9+ KB
```

In [13]:
```
visitors.info()
```

```
<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
```

In [14]:
```
orders.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
visitors.duplicated().sum()
```

Out[15]:
```
0
```

Полных дубликатов нет

In [16]:
```
orders.nunique()
```

Out[16]:
```
transactionId    1197
visitorId        1031
date               31
revenue           713
group               2
dtype: int64
```

In [17]:
```
visitors.nunique()
```

Out[17]:
```
date        31
group        2
visitors    58
dtype: int64
```

In [18]:
```
print('Начало теста:', orders['date'].min())
print('Конец теста:', orders['date'].max())
```

```
Начало теста: 2019-08-01
Конец теста: 2019-08-31
```

In [19]:
```
# Число ошибочных распределений
orders\
.pivot_table(index=['visitorId'],
             values=['group','revenue'],
             aggfunc='nunique'
            )\
.pivot_table(index=['group'],
             values=['revenue'],
             aggfunc='count'
            )
```

Out[19]:

group	revenue
1	973
2	58

In [20]:
```
# id "дефектных" пользователей
index_ord = orders.pivot_table(index=['visitorId'],
             values=['group'],
             aggfunc='nunique'
            )\
.query('group == 2').index
```

In [21]:
```
orders.query('visitorId in @index_ord').sort_values(by='visitorId')
```

Out[21]:

	transactionId	visitorId	date	revenue	group
703	4293855558	8300375	2019-08-07	1790	A
71	3679129301	8300375	2019-08-01	10510	B
823	2971973105	199603092	2019-08-27	2790	A
246	437656952	199603092	2019-08-02	3488	B
26	2223239646	199603092	2019-08-15	3488	A
...	...	...	...	...	...
187	2048878902	4256040402	2019-08-17	1550	A
114	1120327437	4256040402	2019-08-01	5800	A
60	1421016313	4256040402	2019-08-16	56650	B
662	1811671147	4266935830	2019-08-29	78990	A
682	1216533772	4266935830	2019-08-29	78990	B

181 rows × 5 columns

Итоги:

- есть повторяющиеся клиенты;
- групп в тесте 2;
- 31 день: с 01/08/2019 по 31/08/2019;
- в двух группах находятся 58 покупателей;
- "дефектные" покупатели занимают 181 запись.

Так как в данных не должно быть покупателей, которые попадают в обе группы, то удалим их.

In [22]:
```
print('Кол-во "дефектных" записей в группе A:',
      orders.query('visitorId in @index_ord & group == "A"')['visitorId'].count()
     )

print('Кол-во "дефектных" записей в группе B:',
      orders.query('visitorId in @index_ord & group == "B"')['visitorId'].count()
     )
```

```
Кол-во "дефектных" записей в группе A: 89
Кол-во "дефектных" записей в группе B: 92
```

In [23]:
```
print('Кол-во записей в группе A:',
      orders.query('group == "A"')['visitorId'].count()
     )

print('Кол-во записей в группе B:',
      orders.query('group == "B"')['visitorId'].count()
     )
```

```
Кол-во записей в группе A: 557
Кол-во записей в группе B: 640
```

In [24]:
```
print('Потери записей в группе A после очистки:',
      round((1 - (orders.query('group == "A"')['visitorId'].count()
      - orders.query('visitorId in @index_ord & group == "A"')['visitorId'].count())
      /orders.query('group == "A"')['visitorId'].count()) * 100, 2),
      '%'
     )

print('Потери записей в группе B после очистки:',
      round((1 - (orders.query('group == "B"')['visitorId'].count()
      - orders.query('visitorId in @index_ord & group == "B"')['visitorId'].count())
      / orders.query('group == "B"')['visitorId'].count()) * 100, 2),
      '%'
     )
```

```
Потери записей в группе A после очистки: 15.98 %
Потери записей в группе B после очистки: 14.38 %
```

In [25]:
```
# Данные для вычитания из visitors
delete_visit = orders.query('visitorId in @index_ord').pivot_table(index=['date','group'], values='revenue', aggfunc='coun
```

In [26]:
```
orders.head()
```

Out[26]:

	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

In [27]:
```
delete_visit.head()
```

Out[27]:

		revenue
date	**group**	
2019-08-01	**A**	1
	B	4
2019-08-02	**A**	1
	B	1
2019-08-03	**B**	2

In [28]:
```
visitors.head()
```

Out[28]:

	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

In [29]:
```
# Вычитаем из общего кол-ва посетителей число "дефектных"
for visitor_ind in visitors.index:
    # Попытка вычитания
    try:
        dt = visitors.loc[visitor_ind, 'date']
        gr = visitors.loc[visitor_ind, 'group']

        visitors.loc[visitor_ind, 'visitors'] = visitors.loc[visitor_ind, 'visitors'] - delete_visit.loc[dt, gr]['revenue'
    # если не получилось, выводи дату
    except:
        print(visitors.loc[visitor_ind, 'date'])
```

```
2019-08-03
2019-08-13
2019-08-16
2019-08-26
2019-08-19
2019-08-20
```

In [30]:
```
visitors.head()
```

Out[30]:

	date	group	visitors
0	2019-08-01	A	718
1	2019-08-02	A	618
2	2019-08-03	A	507
3	2019-08-04	A	712
4	2019-08-05	A	753

In [31]:
```
# Удаляем проблемный записи
orders = orders.query('visitorId not in @index_ord').sort_values(by='visitorId')
print('Кол-во записей в orders после очистки:', orders.shape[0])
```

```
Кол-во записей в orders после очистки: 1016
```

2.1 График кумулятивной выручки по группам

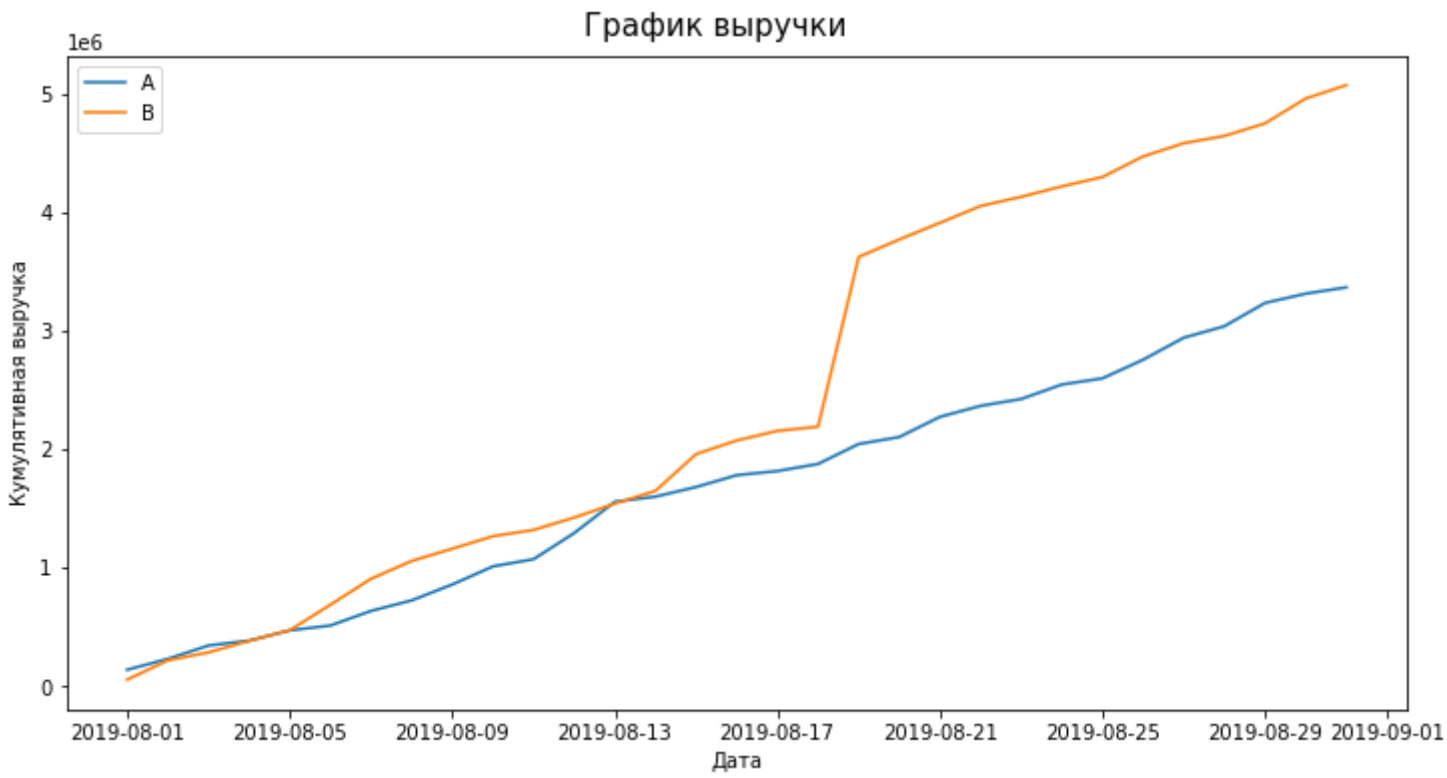
In [32]:
```
# Создаем массив уникальных пар значений дат и групп теста
datesGroups = orders[['date','group']].drop_duplicates()

# Получаем агрегированные кумулятивные по дням данные о заказах
ordersAggregated = datesGroups\
.apply(lambda x: orders[np.logical_and(orders['date'] <= x['date'], orders['group'] == x['group'])]\
       .agg({'date': 'max', 'group': 'max', 'transactionId': 'nunique', 'visitorId': 'nunique', 'revenue': 'sum'}), axis=1
.sort_values(by=['date','group'])

# Получаем агрегированные кумулятивные по дням данные о посетителях интернет-магазина
visitorsAggregated = datesGroups\
.apply(lambda x: visitors[np.logical_and(visitors['date'] <= x['date'], visitors['group'] == x['group'])]\
       .agg({'date': 'max', 'group': 'max', 'visitors': 'sum'}), axis=1)\
.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_on=['date', 'group'])
cumulativeData.columns = ['date', 'group', 'orders', 'buyers', 'revenue', 'visitors']

cumulativeData.head(5)
```

Out[32]:

	date	group	orders	buyers	revenue	visitors
0	2019-08-01	A	23	19	142779	718
1	2019-08-01	B	17	17	59758	709
2	2019-08-02	A	42	36	234381	1336
3	2019-08-02	B	40	39	221801	1289
4	2019-08-03	A	66	60	346854	1843

In [33]:
```
# Перевод в тип даты
cumulativeData['date'] = pd.to_datetime(cumulativeData['date'])
```

In [34]:
```
# 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']]

# Размер поля построения
plt.figure(figsize=(12, 6))

# 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.legend()
plt.ylabel("Кумулятивная выручка")
plt.xlabel("Дата")
plt.suptitle('График выручки', x=0.5, y=0.93, fontsize=15);
```



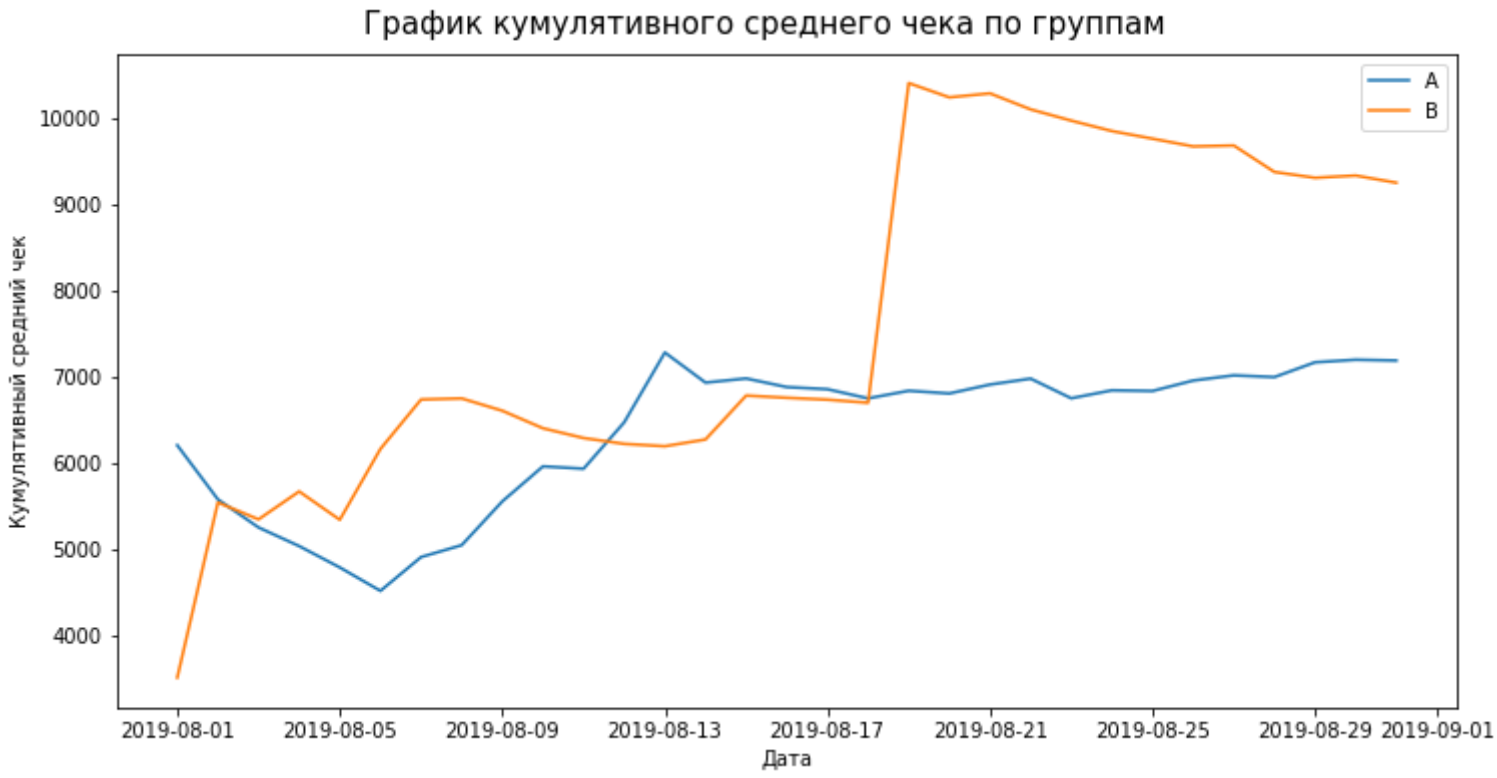
Выводы:

Изначально результаты групп были близкими, однако с 18-го августа наблюдается скачок группы B, после чего графики идут практически параллельно.

Заметка: *необходимо понять причину скачка*

2.2 График кумулятивного среднего чека по группам

In [35]:
```
# Размер поля построения
plt.figure(figsize=(12, 6))

plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']/cumulativeRevenueA['orders'], label='A')
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']/cumulativeRevenueB['orders'], label='B')
plt.legend()
plt.ylabel("Кумулятивный средний чек")
plt.xlabel("Дата")
plt.suptitle('График кумулятивного среднего чека по группам', x=0.5, y=0.93, fontsize=15);
```



Выводы:

Наблюдается скачок в величине среднего чека около 18-го числа. Вероятно, кто-то произвел крупную покупку, что отобразилось на среднем чеке - **аномалия?**

In [36]:
```
# Создадим таблицу
combo_rev = cumulativeRevenueA.merge(cumulativeRevenueB, on='date')\
.rename(columns={'revenue_x':'revenue_A',
                 'revenue_y':'revenue_B',
                 'orders_x':'orders_A',
                 'orders_y':'orders_B'}
       )
```

In [37]:
```
# Доп. колонки =  delta
combo_rev['d_rev'] = combo_rev['revenue_B'] - combo_rev['revenue_A']
combo_rev['d_ord'] = combo_rev['orders_B'] - combo_rev['orders_A']
```

In [38]:
```
# Выведем таблицу
combo_rev.style.background_gradient(cmap='RdYlGn')
```

Out[38]:

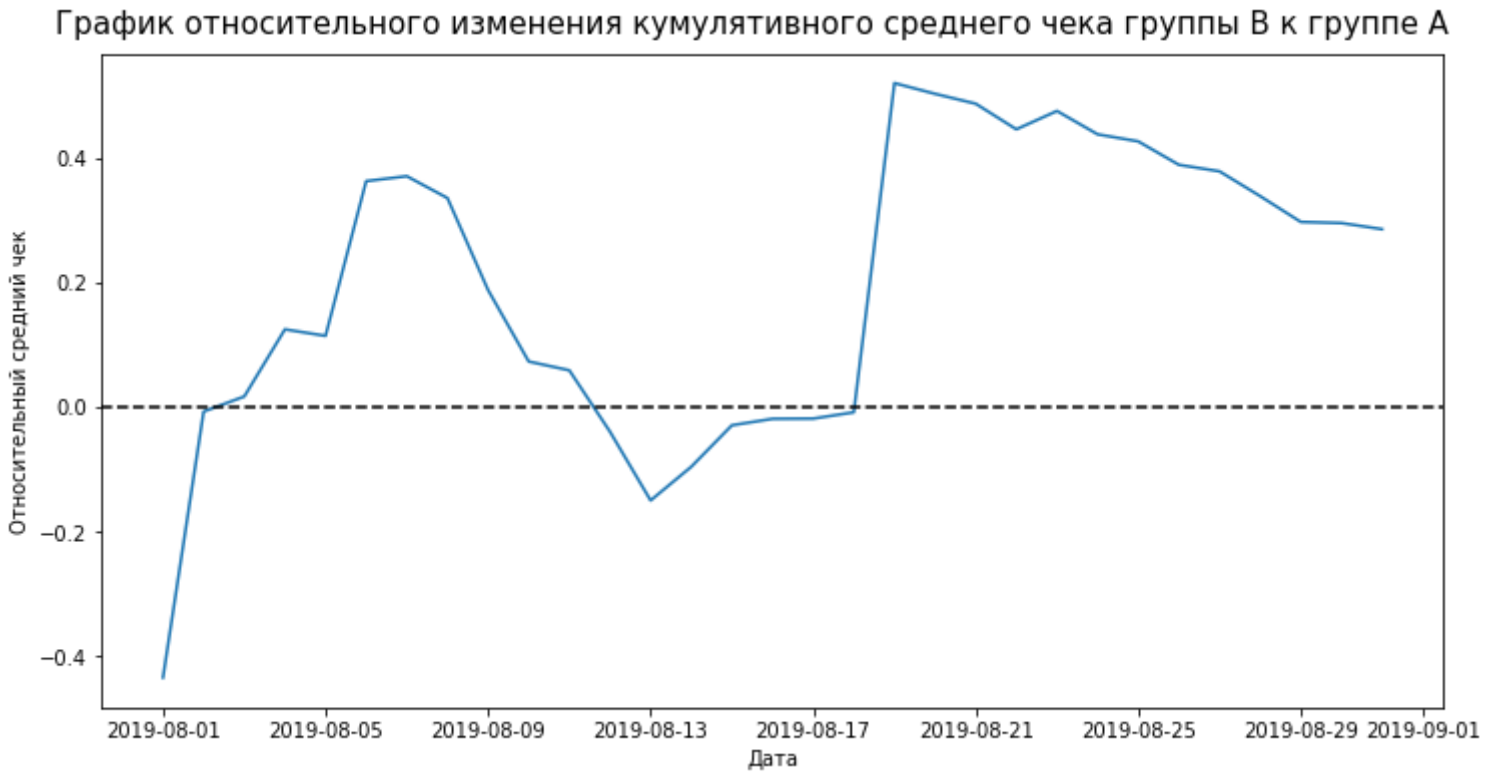
	date	revenue_A	orders_A	revenue_B	orders_B	d_rev	d_ord
0	2019-08-01 00:00:00	142779	23	59758	17	-83021	-6
1	2019-08-02 00:00:00	234381	42	221801	40	-12580	-2
2	2019-08-03 00:00:00	346854	66	288850	54	-58004	-12
3	2019-08-04 00:00:00	388030	77	385740	68	-2290	-9
4	2019-08-05 00:00:00	474413	99	475648	89	1235	-10
5	2019-08-06 00:00:00	515332	114	690490	112	175158	-2
6	2019-08-07 00:00:00	638580	130	909654	135	271074	5
7	2019-08-08 00:00:00	727219	144	1059795	157	332576	13
8	2019-08-09 00:00:00	861456	155	1162961	176	301505	21
9	2019-08-10 00:00:00	1013731	170	1268123	198	254392	28
10	2019-08-11 00:00:00	1074396	181	1321183	210	246787	29
11	2019-08-12 00:00:00	1294788	200	1425237	229	130449	29
12	2019-08-13 00:00:00	1558426	214	1542928	249	-15498	35
13	2019-08-14 00:00:00	1601692	231	1650268	263	48576	32
14	2019-08-15 00:00:00	1682569	241	1960427	289	277858	48
15	2019-08-16 00:00:00	1782420	259	2074677	307	292257	48
16	2019-08-17 00:00:00	1817160	265	2155542	320	338382	55
17	2019-08-18 00:00:00	1876741	278	2190865	327	314124	49
18	2019-08-19 00:00:00	2044934	299	3620785	348	1575851	49
19	2019-08-20 00:00:00	2103613	309	3768059	368	1664446	59
20	2019-08-21 00:00:00	2273782	329	3908406	380	1634624	51
21	2019-08-22 00:00:00	2366147	339	4050134	401	1683987	62
22	2019-08-23 00:00:00	2424010	359	4127403	414	1703393	55
23	2019-08-24 00:00:00	2546023	372	4215269	428	1669246	56
24	2019-08-25 00:00:00	2597907	380	4294526	440	1696619	60
25	2019-08-26 00:00:00	2755121	396	4467965	462	1712844	66
26	2019-08-27 00:00:00	2940471	419	4579312	473	1638841	54
27	2019-08-28 00:00:00	3036933	434	4640510	495	1603577	61
28	2019-08-29 00:00:00	3233233	451	4746610	510	1513377	59
29	2019-08-30 00:00:00	3311413	460	4955833	531	1644420	71
30	2019-08-31 00:00:00	3364656	468	5068972	548	1704316	80

Выводы:

В целом, наблюдаются лучшие показатели у группы B, но нужно внимательнее рассмотреть 19-ое число, так как они могут изменить картину.

2.3 График относительного изменения кумулятивного среднего чека группы B к группе A

In [39]:
```
# Размер поля построения
plt.figure(figsize=(12, 6))

# Собираем данные в одном датафрейме
mergedCumulativeRevenue = cumulativeRevenueA.merge(cumulativeRevenueB,
                                                   left_on='date',
                                                   right_on='date',
                                                   how='left',
                                                   suffixes=['A', 'B']
                                                  )

# Строим отношение средних чеков
plt.plot(mergedCumulativeRevenue['date'],
         (mergedCumulativeRevenue['revenueB'] / mergedCumulativeRevenue['ordersB'])\
         / (mergedCumulativeRevenue['revenueA'] / mergedCumulativeRevenue['ordersA']) - 1)

# Добавляем ось X
plt.axhline(y=0, color='black', linestyle='--')

plt.suptitle('График относительного изменения кумулятивного среднего чека группы B к группе A',
             x=0.5,
             y=0.93,
             fontsize=15
            )

plt.ylabel("Относительный средний чек")
plt.xlabel("Дата");
```

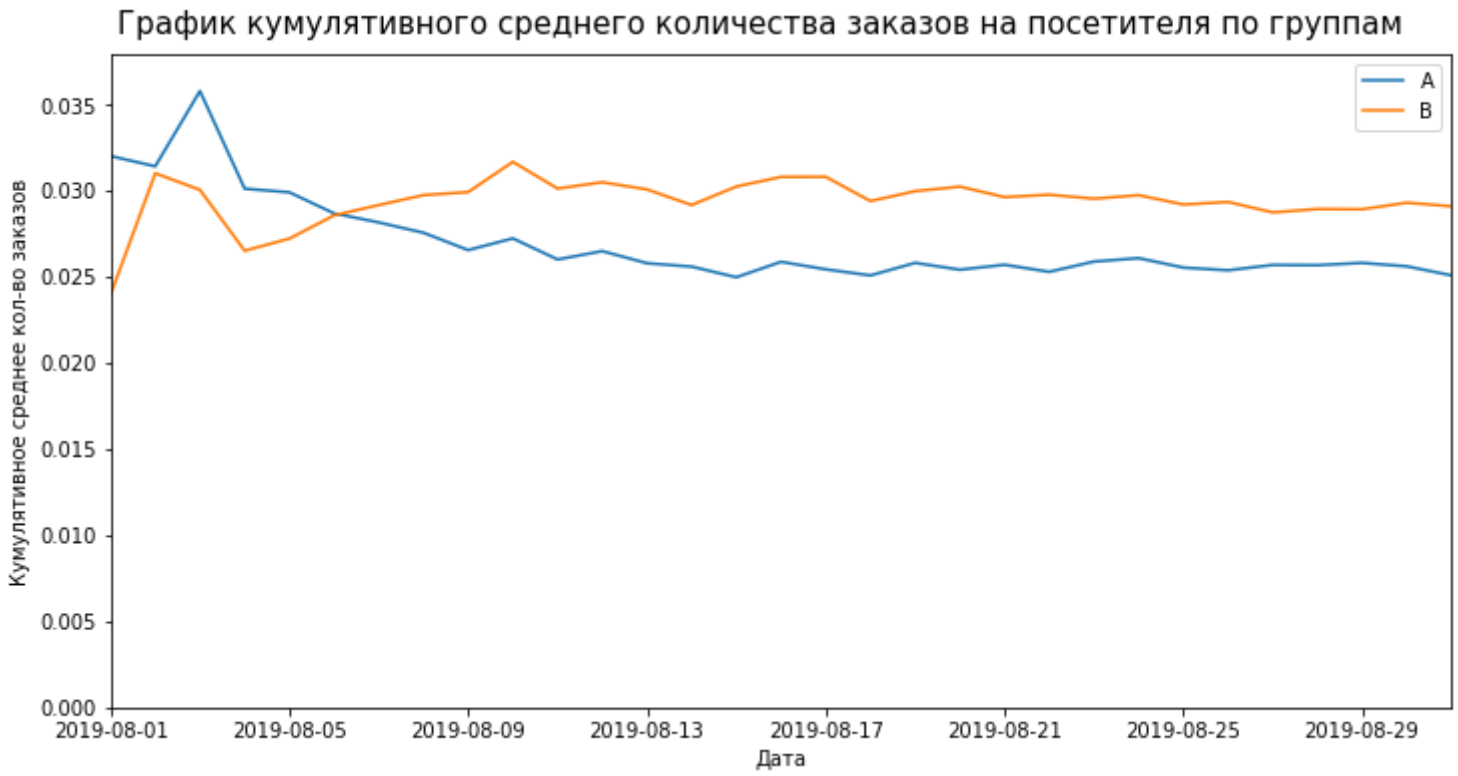


Выводы:

Рост, падение, малый рост, скачок, плавный спуск - скачок 19-го числа. Установившегося нет.

2.4 График кумулятивного среднего количества заказов на посетителя по группам

In [40]:
```
import datetime as dt

# Размер поля построения
plt.figure(figsize=(12, 6))

# 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([dt.date(2019, 8, 1), dt.date(2019, 8, 31), 0, 0.038])

# Название
plt.suptitle('График кумулятивного среднего количества заказов на посетителя по группам',
             x=0.5,
             y=0.93,
             fontsize=15
            )

plt.ylabel("Кумулятивное среднее кол-во заказов")
plt.xlabel("Дата");
```



Выводы:

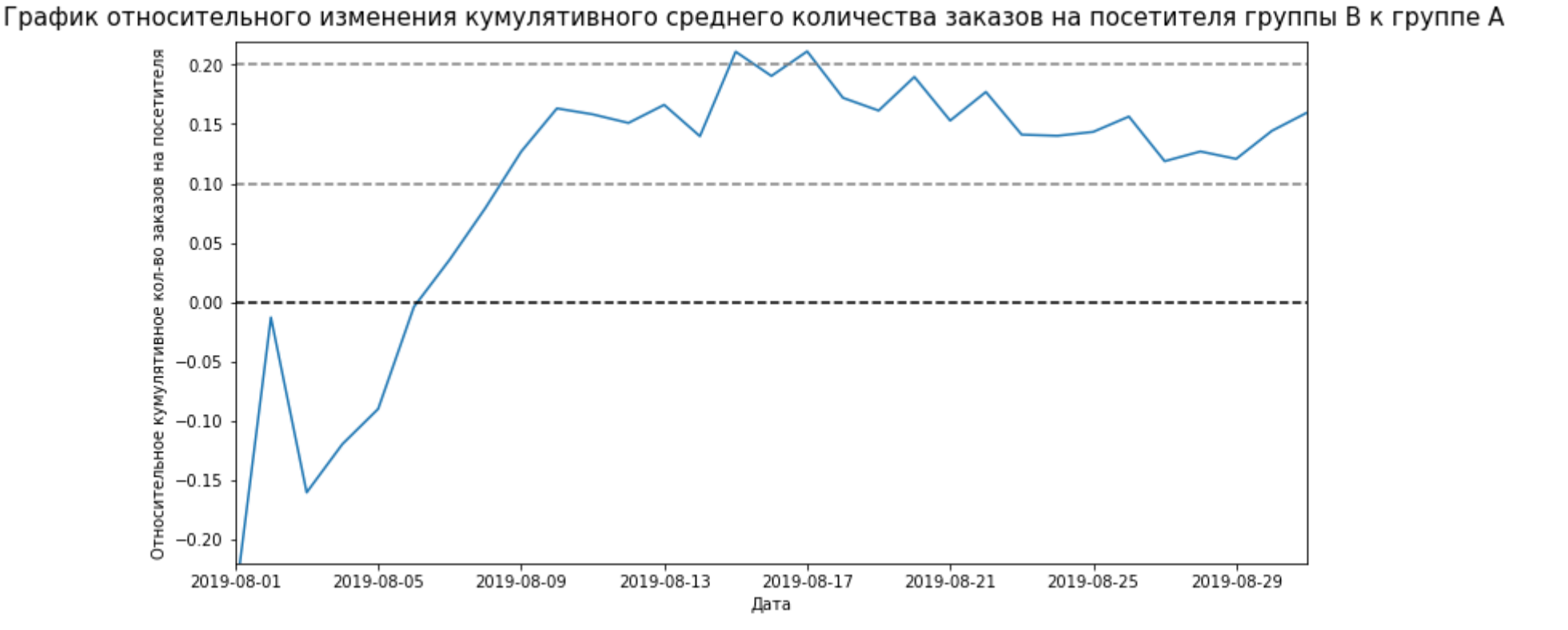
Количество заказов в группе B стабильно больше, чем в группе A, начиная с 9-го числа.

График количества заказов уже больше половины месяца похож на прямую, не имеет резких изменений. Количество заказов можно считать стабилизировавшимся.

(Конечно, дальше сентябрь, а это август - предшкольная пора, вероятно, дальше будет *синхронный* спуск).

2.5 График относительного изменения кумулятивного среднего количества заказов на посетителя группы B к группе A

In [41]:
```
# Размер поля построения
plt.figure(figsize = (12, 6))

mergedCumulativeConversions = cumulativeDataA[['date','conversion']]\
.merge(cumulativeDataB[['date','conversion']],
       left_on='date',
       right_on='date',
       how='left',
       suffixes=['A', 'B']
      )

plt.plot(mergedCumulativeConversions['date'],
         mergedCumulativeConversions['conversionB']/mergedCumulativeConversions['conversionA']-1,
         label="Относительный прирост конверсии группы B относительно группы A"
        )

plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=0.1, color='grey', linestyle='--')
plt.axhline(y=0.2, color='grey', linestyle='--')

plt.axis([dt.date(2019, 8, 1), dt.date(2019, 8, 31), -0.22, 0.22])

# Название
plt.suptitle('График относительного изменения кумулятивного среднего количества заказов на посетителя группы B к группе A'
             x=0.5,
             y=0.93,
             fontsize=15
            )

plt.ylabel("Относительное кумулятивное кол-во заказов на посетителя")
plt.xlabel("Дата");
```



Выводы:

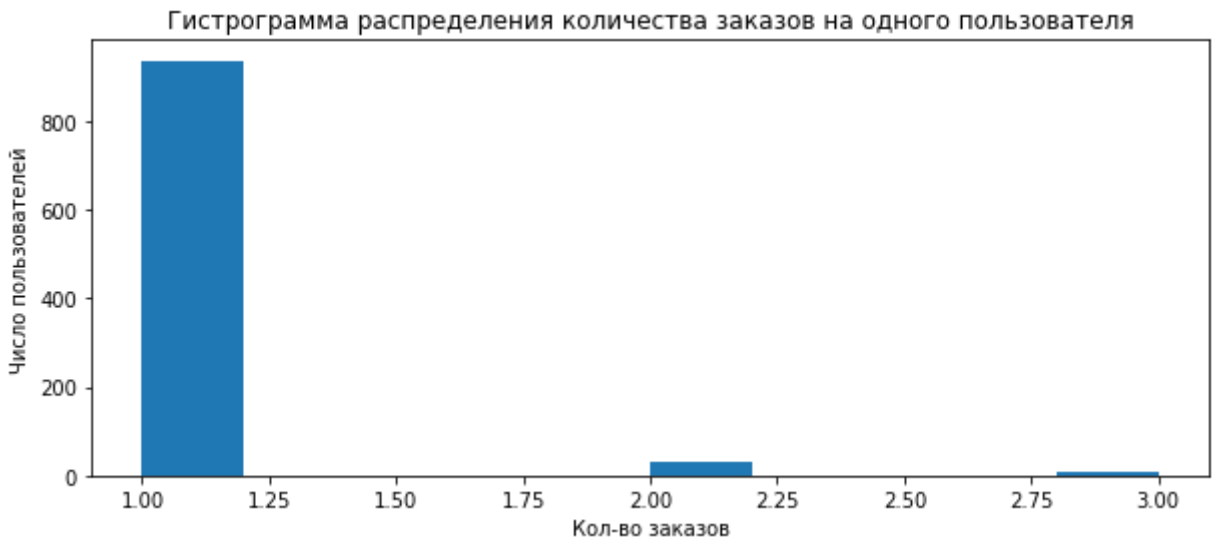
Подтверждает предыдущий вывод:

Количество заказов в группе B стабильно больше, чем в группе A, начиная с 9-го числа.

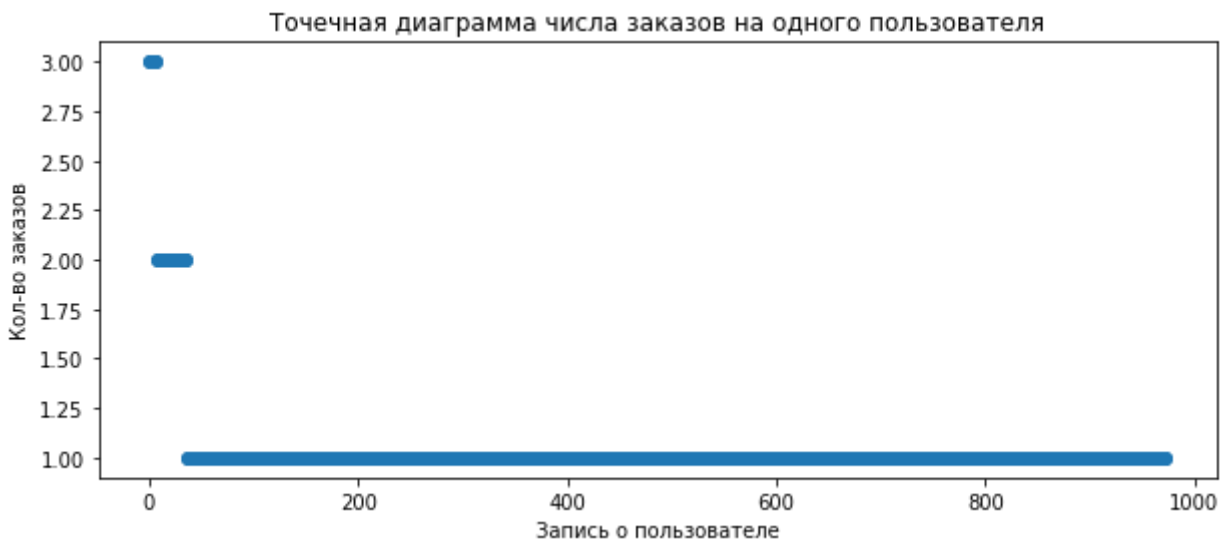
2.6 Точечный график количества заказов по пользователям

In [42]:
```
# Массив для анализа
ordersByUsers = orders.groupby('visitorId', as_index = False)\
.agg({'transactionId':'nunique'})\
.sort_values(by='transactionId', ascending = False) # для удобства

ordersByUsers.columns = ['visitorId', 'orders']
```

In [43]:
```
# Размер поля построения
plt.figure(figsize=(10,4))

plt.hist(ordersByUsers['orders'], bins = 10)
plt.title('Гистрограмма распределения количества заказов на одного пользователя')
plt.ylabel("Число пользователей")
plt.xlabel("Кол-во заказов")
plt.show();
```



In [44]:
```
# Размер поля построения
plt.figure(figsize=(10,4))

x_values = pd.Series(range(0, len(ordersByUsers)))
plt.scatter(x_values, ordersByUsers['orders'])
plt.title('Точечная диаграмма числа заказов на одного пользователя')
plt.ylabel("Кол-во заказов")
plt.xlabel("Запись о пользователе")
plt.show()
```



Выводы:

Большинство совершило 1 или 2 заказа.

2.7 95-й и 99-й перцентили количества заказов на пользователя. Выбор границы для определения аномальных пользователей

In [45]:
```
print('Перцентили количества заказов на пользователя [95, 99]:',
      np.percentile(ordersByUsers['orders'], [95, 99])
     )
```

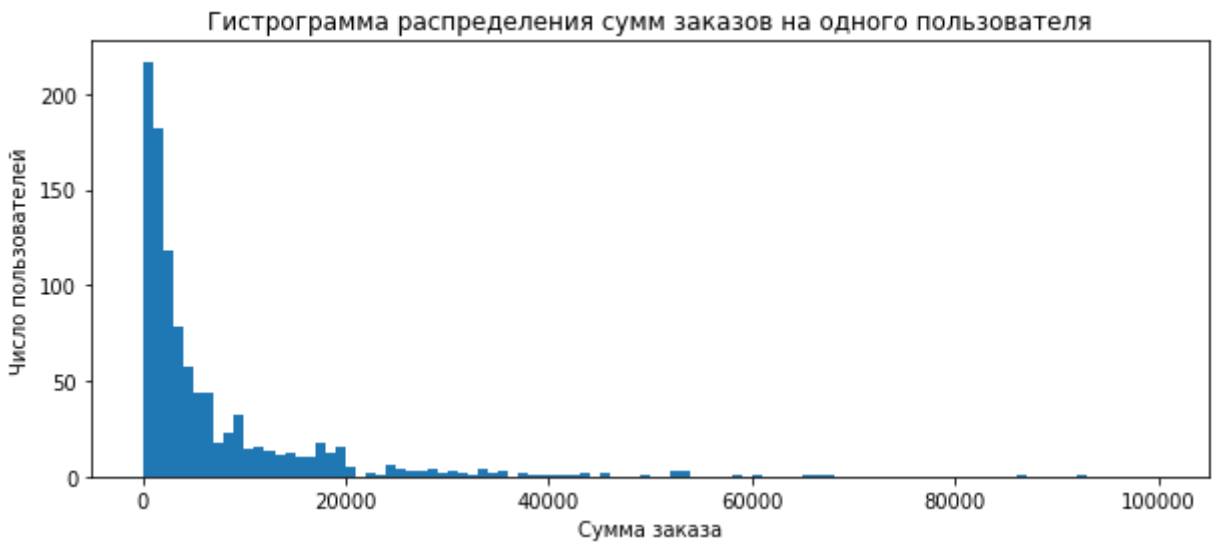
```
Перцентили количества заказов на пользователя [95, 99]: [1. 2.]
```

Выводы:

Не более 1% пользователей оформляли больше 2-ух заказов - выберем это значение в качестве границы аномальных пользоваетелей.

2.8 Точечный график стоимостей заказов

In [46]:
```
# Размер поля построения
plt.figure(figsize=(10,4))

plt.hist(orders['revenue'], range=(0,100000), bins = 100)
plt.title('Гистрограмма распределения сумм заказов на одного пользователя')
plt.ylabel("Число пользователей")
plt.xlabel("Сумма заказа")
plt.show()
```



In [47]:
```
x_values = pd.Series(range(0, len(orders)))

# Размер поля построения
plt.figure(figsize=(10,4))

plt.scatter(x_values, orders['revenue'])
plt.title('Точечная диаграмма сумм заказов на одного пользователя')
plt.axis([0, 1050, 0, 110000])
plt.ylabel("Сумма заказа")
plt.xlabel("Запись о пользователе")
plt.show()
```

Выводы:

В подовляющем большинстве суммы заказов не превышали 30 000 ₽.

2.9 95-й и 99-й перцентили стоимости заказов. Выбор границ для определения аномальных заказов

In [48]:
```
print('Перцентили количества заказов на пользователя [95, 99]:',
      np.percentile(orders['revenue'], [95, 99])
     )
```

```
Перцентили количества заказов на пользователя [95, 99]: [26785. 53904.]
```

Выводы:

У не более 5% пользователей стоимость заказа была более 26 785 ₽ - будем считать границей.

2.10 Статистическая значимость различий в среднем количестве заказов на посетителя между группами по «сырым» данным

In [49]:
```
# Информация о группах
ordersByUsersA = orders[orders['group']=='A'].groupby('visitorId', as_index=False).agg({'transactionId':'nunique'})
ordersByUsersA.columns = ['visitorId', 'orders']
ordersByUsersB = orders[orders['group']=='B'].groupby('visitorId', as_index=False).agg({'transactionId':'nunique'})
ordersByUsersB.columns = ['visitorId', 'orders']

print('Кол-во покупателей в группе A: {}'.format(len(ordersByUsersA)))
print('Кол-во покупок в группе A: {}'.format(ordersByUsersA['orders'].sum()))
print('Кол-во посетителей в группе A: {}\n'.format(visitors[visitors['group']=='A']['visitors'].sum()))

print('Кол-во покупателей в группе B: {}'.format(len(ordersByUsersB)))
print('Кол-во покупок в группе B: {}'.format(ordersByUsersB['orders'].sum()))
print('Кол-во посетителей в группе B: {}'.format(visitors[visitors['group']=='B']['visitors'].sum()))
```

```
Кол-во покупателей в группе A: 445
Кол-во покупок в группе A: 468
Кол-во посетителей в группе A: 18647

Кол-во покупателей в группе B: 528
Кол-во покупок в группе B: 548
Кол-во посетителей в группе B: 18824
```

Гипотезы

Согласно рейтингу ICE была принята к проверке гипотеза:

`Запустить акцию, дающую скидку на товар в день рождения`

Для проверки применим статистический критерий Манна-Уитни к полученным выборкам, `alpha = 5%` .

Провека №1

Относительный прирост конверсии группы B по отношению к группе A:

H_0: Запуск акции, дающей скидку на товар в день рождения вызовет прирост консверсии.
H_1: Запуск акции, дающей скидку на товар в день рождения не изменит консверсию.

In [50]:
```
# Составим списки кол-ва заказов sampleA и sampleB со всеми пользователями (+ без покупок) по группам
list_orders_1 = []
for i in range(0,(visitors[visitors['group'] == 'A']['visitors'].sum() - len(ordersByUsersA))):
    list_orders_1.append(0)
orders_by_non_purchased_users_A = pd.Series(data = list_orders_1, name = 'orders')

list_orders_2 = []
for i in range(0,(visitors[visitors['group'] == 'B']['visitors'].sum()-len(ordersByUsersB))):
    list_orders_2.append(0)
orders_by_non_purchased_users_B = pd.Series(data = list_orders_2, name = 'orders')

sampleA = pd.concat([ordersByUsersA['orders'], orders_by_non_purchased_users_A], axis=0)
sampleB = pd.concat([ordersByUsersB['orders'], orders_by_non_purchased_users_B], axis=0)
print('Относительный прирост конверсии группы B по отношению к группе A: {:.3f}'.format(sampleB.mean()/sampleA.mean()-1))

alpha = 0.05 # выбранный уровень
results = stats.mannwhitneyu(sampleA, sampleB) # тест

print('P-value: {}'.format(results.pvalue), '\n')

if results.pvalue < alpha:
    print('Вывод:')
    print('Различие в количестве заказов СТАТИСТИЧЕСКИ ЗНАЧИМО')
else:
    print('Вывод:')
    print('Различие в количестве заказов нельзя считать значимым')
```

```
Относительный прирост конверсии группы B по отношению к группе A: 0.160
P-value: 0.0109567830835148

Вывод:
Различие в количестве заказов СТАТИСТИЧЕСКИ ЗНАЧИМО
```

Выводы:

С учетом пользователей без покупок различие в среднем количестве заказов по «сырым» данным является **статистически значимым**.

2.11 Статистическая значимость различий в среднем чеке заказа между группами по «сырым» данным

Провека №2

Относительное изменение среднего чека в группе B по отношению группы A:

H_0: Запуск акции, дающей скидку на товар в день рождения увеличит средний чек заказа.
H_1: Запуск акции, дающей скидку на товар в день рождения не изменит средний чек заказа.

In [51]:
```
print(
    'Относительное изменение среднего чека в группе B по отношению группы A: {:.3f}'
      .format(
          orders[orders['group'] == 'B']['revenue'].mean() / orders[orders['group'] == 'A']['revenue'].mean()-1
      )
)

results = stats.mannwhitneyu(
    orders[orders['group'] == 'A']['revenue'], orders[orders['group'] == 'B']['revenue']
)

print('P-value: {}'.format(results.pvalue), '\n')

if results.pvalue < alpha:
    print('Вывод:')
    print('Различия в среднем чеке заказа СТАТИСТИЧЕСКИ ЗНАЧИМЫ')
else:
    print('Вывод:')
    print('Различия в среднем чеке заказа нельзя считать статистически значимыми')
```

```
Относительное изменение среднего чека в группе B по отношению группы A: 0.287
P-value: 0.8294908998149533

Вывод:
Различия в среднем чеке заказа нельзя считать статистически значимыми
```

Выводы:

Статистическую значимость **различий в среднем чеке** заказа между группами по «сырым» **нельзя считать значимой**.

2.12 Статистическая значимость различий в среднем количестве заказов на посетителя между группами по «очищенным» данным

In [52]:
```
# Пределы для аномалий
limit_orders = 2
limit_revenue = 26785
```

In [53]:
```
# Кол-во пользователей с аномалиями
usersWithManyOrders = pd.concat([ordersByUsersA[ordersByUsersA['orders']>limit_orders]['visitorId'],
                                 ordersByUsersB[ordersByUsersB['orders']>limit_orders]['visitorId']],
                                 axis = 0
                                )

usersWithExpensiveOrders = orders[orders['revenue'] > limit_revenue]['visitorId']

abnormalUsers = pd.concat([usersWithManyOrders,
                           usersWithExpensiveOrders],
                           axis = 0
                          ).drop_duplicates().sort_values()

print('Кол-во пользователей с аномалиями: {}'
      .format(len(abnormalUsers)),
      'шт.'
     )
```

```
Кол-во пользователей с аномалиями: 58 шт.
```

Провека №3

Относительный прирост конверсии группы B по отношению к группе A:

H_0: Запуск акции, дающей скидку на товар в день рождения вызовет прирост консверсии.
H_1: Запуск акции, дающей скидку на товар в день рождения не изменит консверсию.

In [54]:
```
sampleAFiltered = pd.concat([
    ordersByUsersA[np.logical_not(ordersByUsersA['visitorId'].isin(abnormalUsers))]['orders'],
    orders_by_non_purchased_users_A
], axis = 0
)

sampleBFiltered = pd.concat([
    ordersByUsersB[np.logical_not(ordersByUsersB['visitorId'].isin(abnormalUsers))]['orders'],
    orders_by_non_purchased_users_B
], axis = 0
)

print('Относительный прирост конверсии группы B по отношению к группе A после очистки данных: {:.3f}'
      .format(sampleBFiltered.mean() / sampleAFiltered.mean() - 1))

results = stats.mannwhitneyu(sampleAFiltered,
                             sampleBFiltered
                            )

print('P-value: {}'.format(results.pvalue), '\n')

if results.pvalue < alpha:
    print('Вывод:')
    print('В среднем количестве заказов на посетителя РАЗНИЦА СТАТИСТИЧЕСКИ ЗНАЧИМА')
else:
    print('Вывод:')
    print('В среднем количестве заказов на посетителя разницу нельзя считать значимой')
```

```
Относительный прирост конверсии группы B по отношению к группе A после очистки данных: 0.182
P-value: 0.012285273602771688

Вывод:
В среднем количестве заказов на посетителя РАЗНИЦА СТАТИСТИЧЕСКИ ЗНАЧИМА
```

Выводы:

Статистическая значимость **различий в среднем количестве заказов** на посетителя между группами по «очищенным» данным **СТАТИСТИЧЕСКИ ЗНАЧИМА**.

Запуск акции, дающей скидку на товар в день рождения вызвал прирост консверсии. Группа B на 18,9% лучше A.

2.13 Статистическая значимость различий в среднем чеке заказа между группами по «очищенным» данным

Провека №4

Относительное изменение среднего чека в группе B по отношению группы A:

H_0: Запуск акции, дающей скидку на товар в день рождения увеличит средний чек заказа.

H_1: Запуск акции, дающей скидку на товар в день рождения не изменит средний чек заказа.

In [55]:
```
print('Относительное изменение среднего чека в группе B по отношению группы A после очистки данных: {:.3f}'
      .format(
orders[np.logical_and(orders['group'] == 'B', np.logical_not(orders['visitorId']
                                                             .isin(abnormalUsers)
                                                             )
                      )]['revenue'].mean()\
/ orders[np.logical_and(orders['group'] == 'A',
                        np.logical_not(orders['visitorId'].isin(abnormalUsers))
                       )]['revenue'].mean() - 1
      )
     )


results = stats.mannwhitneyu(orders[np.logical_and(orders['group'] == 'A',
                                                   np.logical_not(orders['visitorId'].isin(abnormalUsers))
                                                  )]['revenue'],
                             orders[np.logical_and(orders['group'] == 'B',
                                                   np.logical_not(orders['visitorId'].isin(abnormalUsers))
                                                  )]['revenue'])

print('P-value: {}'.format(results.pvalue), '\n')

if results.pvalue < alpha:
    print('Вывод:')
    print('Различия в среднем чеке заказа СТАТИСТИЧЕСКИ ЗНАЧИМЫ')
else:
    print('Вывод:')
    print('Различия в среднем чеке заказа нельзя считать статистически значимыми')
```

```
Относительное изменение среднего чека в группе B по отношению группы A после очистки данных: -0.048
P-value: 0.6458964038091206

Вывод:
Различия в среднем чеке заказа нельзя считать статистически значимыми
```

Выводы:

Статистическую значимость **различий в среднем чеке** заказа между группами по «очищенным» данным **нельзя считать статистически значимой**.

запуск акции, дающей скидку на товар в день рождения не изменил средний чек заказа.

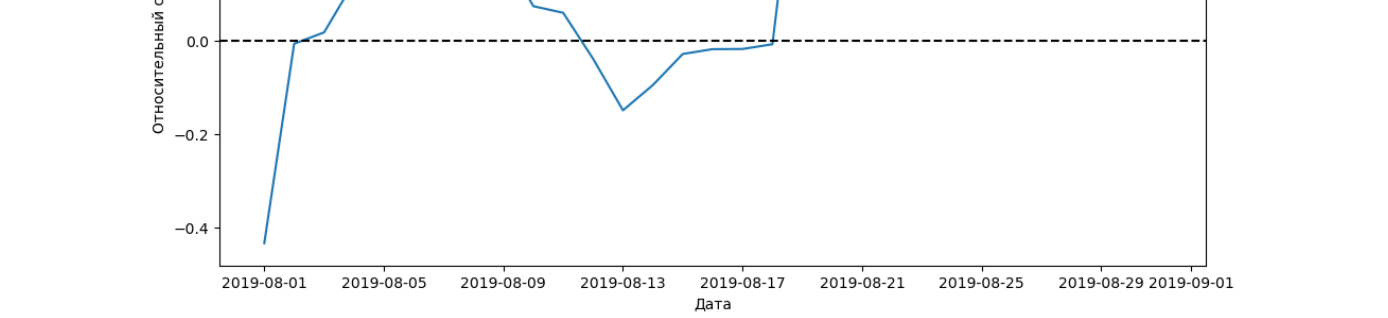
До очистки 0.287, после -0.022. Это говорит о значительном влиянии аномалий.

2.14 Решение по результатам теста

Выводы

- **Различия в количестве** заказов между группами по «сырым» и по «очищенным» отсутствуют. Между группами A и B они **статистически значимы**.
- **Различия в среднем чеке** заказа между группами по «сырым» и по «очищенным» отсутствуют. Между группами A и B их нельзя считать статистически значимыми.
- График различия среднего количества заказов между группами сообщает, что результаты группы B лучше группы A, кол-во заказов продолжает расти.

```
"Количество заказов в группе B стабильно больше, чем в группе A, начиная с 9-го числа"
```



- График различия среднего чека говорит о том, что результаты группы B лучше результатов группы A:

В кумулятивных метриках наблюдается лидерство группы B, однако были обнаружены "всплески" на графиках выше. Была проведена очистка аномалий, после которой различия между группами уменьшились.

До очистки показатель конверсии 16,0%, а после 18,9% - разница в 2,9%.

Исходя из обнаруженных фактов, тест *можно* остановить и стоит **признать победу группы B**.

Заметки для других отделов:

Также стоит отметить, что в данных были обнаружены пользователи, которые попали в обе группы тестирования - они были удалены, однако стоит рассмотреть механизм распределения, так как он может в дальнейшем повлиять на другие тесты. Ниже предствалены даты, в которых нет "проблемных" пользователей.

```
2019-08-03
2019-08-13
2019-08-16
2019-08-26
2019-08-19
2019-08-20
```