1 Описание проекта

Вы — аналитик крупного интернет-магазина. Вместе с отделом маркетинга вы подготовили список гипотез для увеличения выручки. Приоритизируйте гипотезы, запустите A/B-тест и проанализируйте результаты.

2 Часть 1. Приоритизация гипотез.

2.1 Описание данных

Файл `/datasets/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.

2.2 Загрузите данные и подготовьте их к анализу

In [1]:

```
import pandas as pd

data=pd.read_csv('/datasets/hypothesis.csv')
data

```

Out[1]:

	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

Приведем название столбцов к общему формату. Необходимо сделать переименование колонок. (предобработка данных)

In [2]:

```
data.columns = [x.lower() for x in data.columns.values]
data.columns
```

Out[2]:

```
Index(['hypothesis', 'reach', 'impact', 'confidence', 'efforts'], dtyp
e='object')
```

2.3 Примените фреймворк ICE для приоритизации гипотез. Отсортируйте их по убыванию приоритета.

In [3]:

```
data['ICE'] = (data['impact'] * data['confidence']) / data['efforts']
data[['hypothesis', 'ICE']].sort_values(by='ICE', ascending=False)
```

Out[3]:

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

2.4 Примените фреймворк RICE для приоритизации гипотез. Отсортируйте их по убыванию приоритета.

In [4]:

```
data['RICE'] = (data['reach'] * data['impact'] * data['confidence']) / data['ef
data[['hypothesis', 'RICE']].sort_values(by='RICE', ascending=False)
```

Out[4]:

	hypothesis	RICE
7	Добавить форму подписки на все основные страни...	112.0
2	Добавить блоки рекомендаций товаров на сайт ин...	56.0
0	Добавить два новых канала привлечения трафика,...	40.0
6	Показать на главной странице баннеры с актуаль...	40.0
8	Запустить акцию, дающую скидку на товар в день...	16.2
3	Изменить структура категорий, что увеличит кон...	9.0
1	Запустить собственную службу доставки, что сок...	4.0
5	Добавить страницу отзывов клиентов о магазине,...	4.0
4	Изменить цвет фона главной страницы, чтобы уве...	3.0

2.5 Укажите, как изменилась приоритизация гипотез при применении RICE вместо ICE. Объясните, почему так произошло.

Метод `RICE` учитывает больше параметров чем `ICE` . При расчете `RICE` мы учитываем такой параметр как `Reach` — скольких пользователей затронет изменение, которое вы хотите внести. Из-за этого он более точно позволяет приоретизировать гипотезы. Данный метод целесообразно применять, когда оцениваемые гипотезы затрагивают разные по обьемам группы пользователей Гипотеза №7 затронет всех пользователей интернет магазина, в то время как гипотеза №4 коснется меньшего числа пользователей (не все пользователи начинают свой путь по сайту с главной страницы) В нашем случае использование метода `RICE` более предпочтительнее, и приоретизация по данному методу более точно отражает значимость гипотез. Гипотеза №7 имеет максимальный приоритет.

3 Часть 2. Анализ A/B-теста

3.1 Описание данных

Вы провели A/B-тест и получили результаты, которые описаны в файлах `/datasets/orders.csv` и `/datasets/visitors.csv`

`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

3.2 Постройте график кумулятивной выручки по группам. Сделайте выводы и предположения.

In [5]:

```
import pandas as pd
import datetime as dt
import numpy as np

import warnings

warnings.simplefilter('ignore')

orders = pd.read_csv('//datasets/orders.csv')
orders['date'] = orders['date'].map(lambda x: dt.datetime.strptime(x, '%Y-%m-%d

visitors = pd.read_csv('/datasets/visitors.csv')
visitors['date'] = visitors['date'].map(lambda x: dt.datetime.strptime(x, '%Y-%

display(orders.head())
display(visitors.head())

```

	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

	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 [6]:

```
# 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_v

# получаем агрегированные кумулятивные по дням данные о посетителях интернет-ма
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).so

# объединяем кумулятивные данные в одной таблице и присваиваем ее столбцам поня
cumulativeData = ordersAggregated.merge(visitorsAggregated, left_on=['date', 'g
                                        right_on=['date', 'group'])
cumulativeData.columns = ['date', 'group', 'orders', 'buyers', 'revenue', 'visi

cumulativeData.head(5)
```

Out[6]:

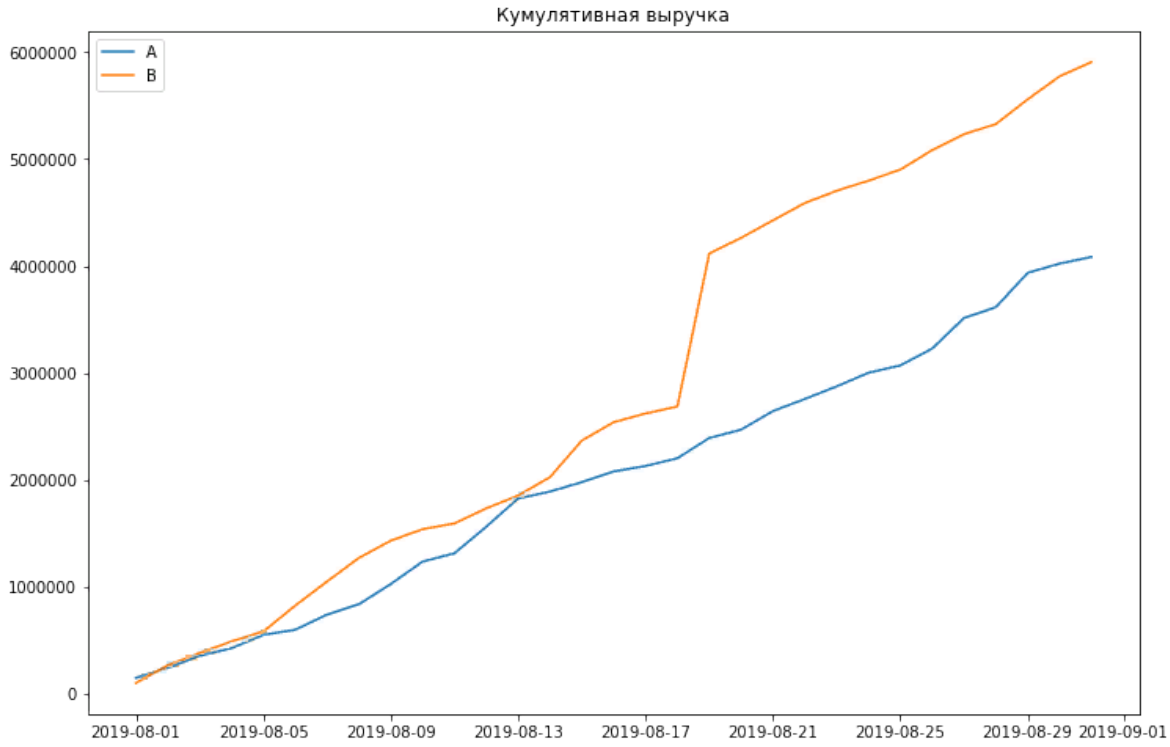
	date	group	orders	buyers	revenue	visitors
0	2019-08-01	A	24	20	148579	719
1	2019-08-01	B	21	20	101217	713
2	2019-08-02	A	44	38	242401	1338
3	2019-08-02	B	45	43	266748	1294
4	2019-08-03	A	68	62	354874	1845

In [7]:

```
import matplotlib.pyplot as plt

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням
cumulativeRevenueA = cumulativeData[cumulativeData['group']=='A'][['date','reve

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

# Строим график выручки группы A
plt.figure(figsize=(12,8))
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')
plt.legend()
```

Out[7]:

```
<matplotlib.legend.Legend at 0x7f2c78240810>
```



Вывод: Выручка почти равномерно увеличивается в течение всего теста. Хороший знак. Однако график выручки группы B в нескольких точках резко растет. Это может сигнализировать о всплесках числа заказов, либо о появлении очень дорогих заказов в выборке.

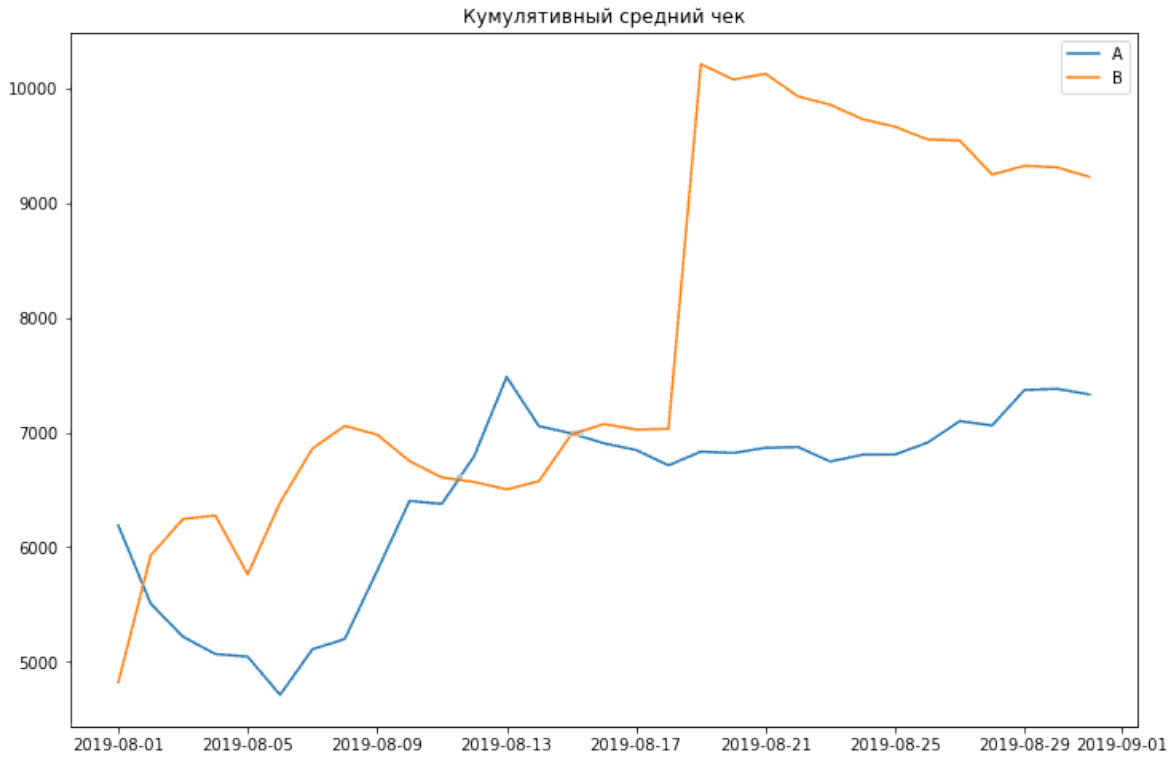
3.3 Постройте график кумулятивного среднего чека по группам. Сделайте выводы и предположения.

In [8]:

```
# Построим графики среднего чека по группам — разделим кумулятивную выручку на
plt.figure(figsize=(12,8))
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']/cumulativeRe
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']/cumulativeRe
plt.title('Кумулятивный средний чек')
plt.legend()
```

Out[8]:

```
<matplotlib.legend.Legend at 0x7f2c6c1cf8d0>
```



Вывод: За месяц средний чек по группе A становится более равномерным, а вот по группе B мы наблюдаем резкий всплес в районе 18-19 числа. Скорее всего была совершена аномально большая покупка (такое бывает). Данный заказ исказил данные по группе. Для получения корректных данных на графике необходимо большее число наблюдений за группой B . Иными словами, опираясь на собранные данные мы можем сделать предположение о среднем чеке по группе A на уровне 7000 руб. Но для группы B данных недостаточно.

3.4 Постройте график относительного изменения кумулятивного среднего чека группы B к группе A.

In [21]:

```
# собираем данные в одном датафрейме

mergedCumulativeRevenue = cumulativeRevenueA.merge(cumulativeRevenueB,
                                                   left_on='date',
                                                   right_on='date',
                                                   how='left', suffixes=['A', '

# строим отношение средних чеков
plt.figure(figsize=(12,8))
plt.plot(mergedCumulativeRevenue['date'], (mergedCumulativeRevenue['revenueB']/
plt.title('Относительное изменение кумулятивного среднего чека группы B к групп

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

Out[21]:

```
<matplotlib.lines.Line2D at 0x7f2c6bee85d0>
```

Вывод: В нескольких точках график различия между сегментами резко «скачет». Это говорит о крупных заказах.

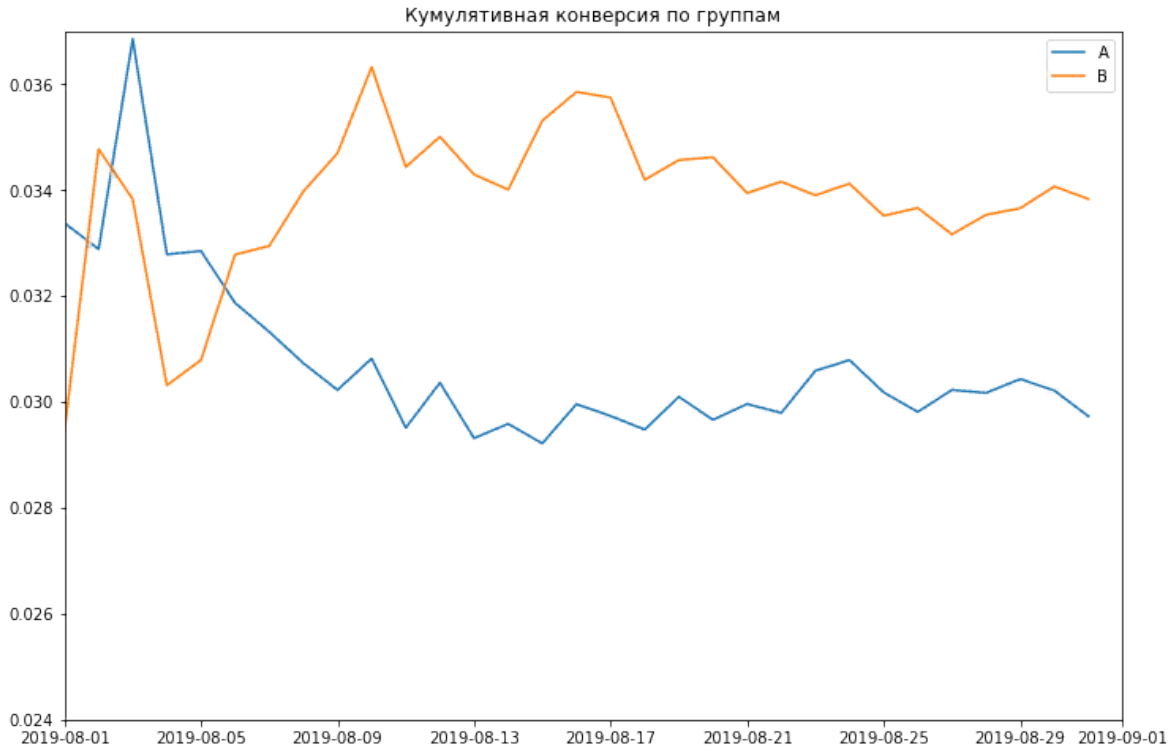
3.5 Постройте график кумулятивной конверсии по группам.

In [22]:

```
# считаем кумулятивную конверсию
cumulativeData['conversion'] = cumulativeData['orders']/cumulativeData['visitor

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

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

# строим графики
plt.figure(figsize=(12,8))
plt.plot(cumulativeDataA['date'], cumulativeDataA['conversion'], label='A')
plt.plot(cumulativeDataB['date'], cumulativeDataB['conversion'], label='B')
plt.legend()
plt.title('Кумулятивная конверсия по группам')

# задаем масштаб осей
plt.axis(["2019-08-01", '2019-09-01', 0.024, 0.037])
```

Out[22]:

```
['2019-08-01', '2019-09-01', 0.024, 0.037]
```



Вывод: Конверсия группы B стабильно выше. Мы наблюдаем стандартный график. В начале идут большие скачки в связи с небольшим промежутком анализа, и далее они сгажываются. К концу месяца можно говорить об установившемся значении кумулятивной конверсии для обеих групп.

3.6 Постройте график относительного изменения кумулятивной конверсии

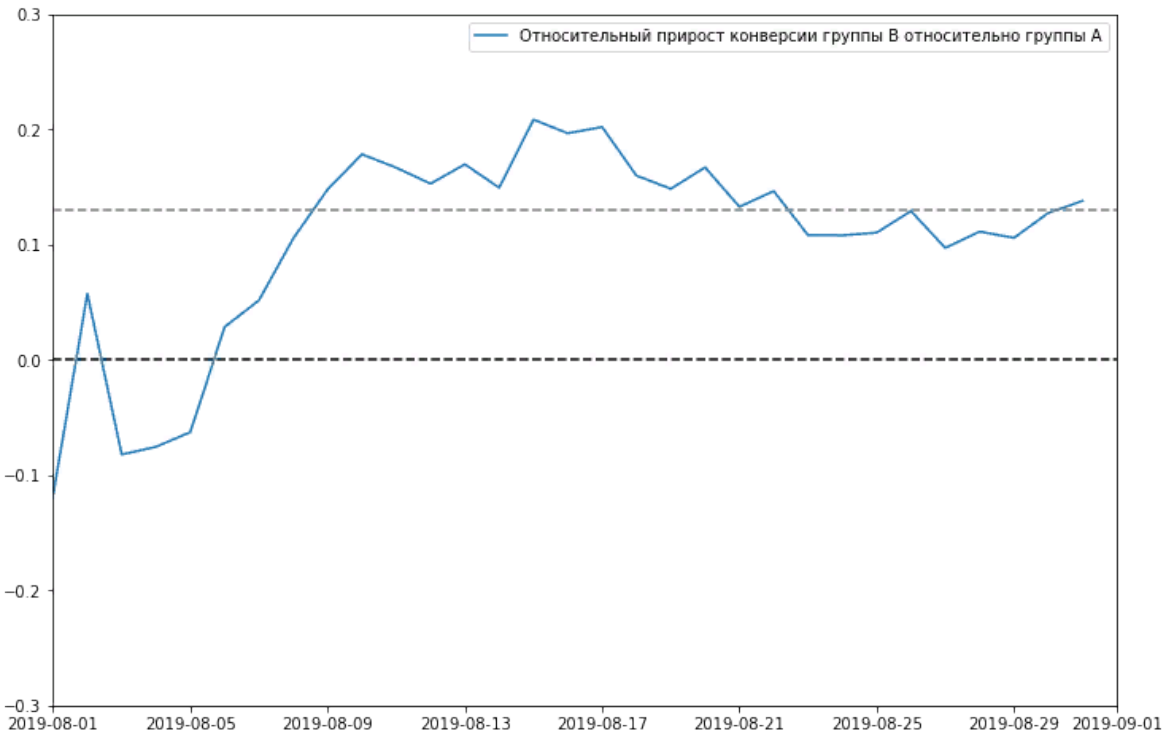
группы B к группе A.

In [11]:

```
mergedCumulativeConversions = cumulativeDataA[['date','conversion']].merge(cumu
                                                                           left_
                                                                           righ
                                                                           how=

# сторим график
plt.figure(figsize=(12,8))
plt.plot(mergedCumulativeConversions['date'],
         mergedCumulativeConversions['conversionB']/mergedCumulativeConversions
         label="Относительный прирост конверсии группы B относительно группы A"
plt.legend()

plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=0.13, color='grey', linestyle='--')
plt.axis(["2019-08-01", '2019-09-01', -0.3, 0.3])
```

Out[11]:

```
['2019-08-01', '2019-09-01', -0.3, 0.3]
```



Вывод: Мы наблюдаем постоянный рост относительного изменения конверсии группы B к группе A. В целом отношение конверсии ещё не установилось, и сейчас делать какие-либо выводы по тесту нельзя. Впрочем, сперва стоит проанализировать аномалии, возможно, они изменят картину

3.7 Постройте точечный график количества заказов по пользователям

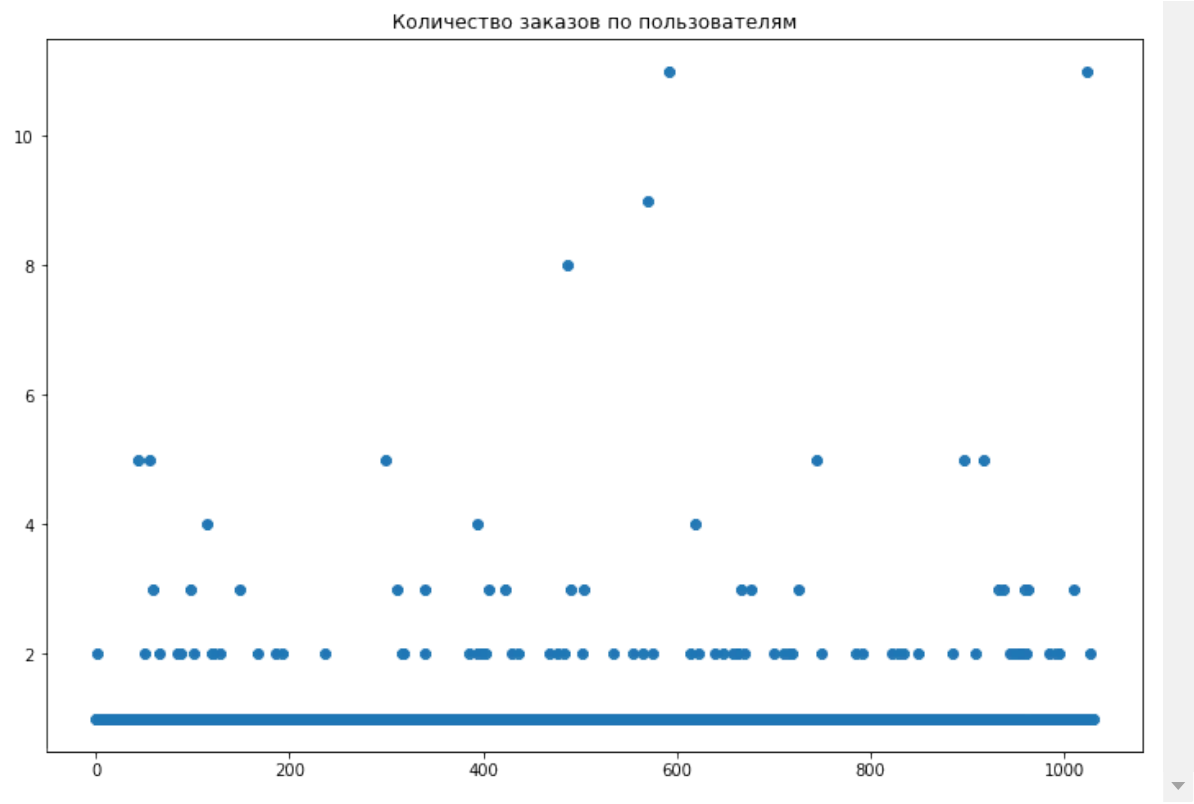
In [12]:

```
# получам данные о количестве заказов
ordersByUsers = (
    orders.groupby('visitorId', as_index=False)
    .agg({'transactionId': pd.Series.nunique})
)
ordersByUsers.columns = ['userId', 'orders']

display(ordersByUsers.sort_values(by='orders', ascending=False).head(10))

# серия из чисел от 0 до количества наблюдений в ordersByUsers
plt.figure(figsize=(12,8))
x_values = pd.Series(range(0, len(ordersByUsers)))
plt.title('Количество заказов по пользователям')
plt.scatter(x_values, ordersByUsers['orders'])
```

	userId	orders
1023	4256040402	11
591	2458001652	11
569	2378935119	9
487	2038680547	8
44	199603092	5
744	3062433592	5
55	237748145	5
917	3803269165	5
299	1230306981	5
897	3717692402	5

Out[12]:

```
<matplotlib.collections.PathCollection at 0x7f2c75300f50>
```



Вывод: Большенство пользователей делали 1-3 заказа, но есть и "уникальные" которые совершили 11 заказов.Их точная доля не ясна — непонятно, считать их аномалиями или нет.

3.8 Посчитайте 95-й и 99-й перцентили количества заказов на пользователя. Выберите границу для определения аномальных пользователей.

In [13]:

```
display(ordersByUsers.sort_values(by='orders', ascending=False).head(10))

# Метод np.percentile('column', [percentile1, percentile2, percentile3]) находи
np.percentile(ordersByUsers['orders'], [95, 99])
```

	userId	orders
1023	4256040402	11
591	2458001652	11
569	2378935119	9
487	2038680547	8
44	199603092	5
744	3062433592	5
55	237748145	5
917	3803269165	5
299	1230306981	5
897	3717692402	5

Out[13]:

```
array([2., 4.])
```

Вывод: Всего 5% покупателей оформили более 2 заказов и всего 1% офрмили более 4. Разумно выбрать 2 заказа на одного пользователя за нижнюю границу числа заказов, и отсеять аномальных пользователей по ней. Пользователи, совершившие много заказов, влияют на числитель формулы конверсии. Скорее всего, их поведение отличается от нормального. Если речь не идёт об интернет-магазине с регулярным спросом, «обычный» пользователь совершает не больше двух заказов за короткий срок.

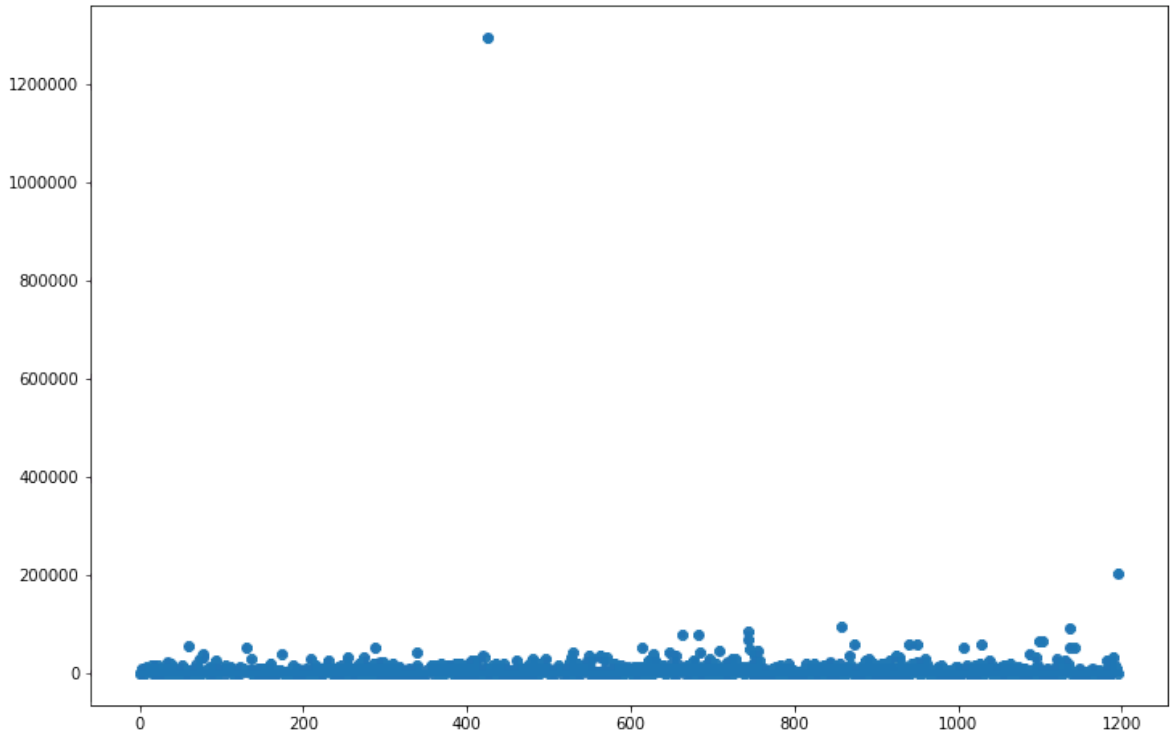
3.9 Постройте точечный график стоимостей заказов. Сделайте выводы и предположения.

In [14]:

```
x_values = pd.Series(range(0,len(orders['revenue'])))
plt.figure(figsize=(12,8))
plt.title('Точечный график стоимостей заказов')
plt.scatter(x_values, orders['revenue'])
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x7f2c6bf5b9d0>
```



Вывод: Мы видим один аномальный заказ более чем на 1 млн и еще один чуть больше чем 200 тыс. Эти заказы явно оказывают негативное влияние на нашу выборку. Вполне возможно, что в дальнейших расчетах мы не будем их учитывать.

3.10 Посчитайте 95-й и 99-й перцентили стоимости заказов. Выберите

границу для определения аномальных заказов.

In [15]:

```
1 np.percentile(orders['revenue'], [95, 99])
```

Out[15]:

```
array([28000.  , 58233.2])
```

Вывод: 5% пользователей оформили заказ на сумму свыше 28 тыс. Если такие дорогие заказы попадут в одну из групп теста, они сразу же исказят результаты и сделает победителем ту группу, где оказались. При этом такие покупки — редкость, исключение, которое проявилось не из-за тестирования гипотезы, а случайно.

Такие аномально дорогие заказы следует удалять из теста. Можно провести и анализ самых дешёвых заказов. Однако из-за маленькой стоимости они слабо влияют на средний чек.

3.11 Посчитайте статистическую значимость различий в конверсии между группами по «сырым» данным.

In [16]:

```python
import scipy.stats as stats

# пользователи группы A по дням
visitorsADaily = visitors[visitors['group'] == 'A'][['date', 'visitors']]
visitorsADaily.columns = ['date', 'visitorsPerDateA']

# кумулятивная суума пользователей A по дням
visitorsACummulative = visitorsADaily.apply(
    lambda x: visitorsADaily[visitorsADaily['date'] <= x['date']].agg(
        {'date': 'max', 'visitorsPerDateA': 'sum'}
    ),
    axis=1,
)
visitorsACummulative.columns = ['date', 'visitorsCummulativeA']

#пользователи группы B по дням
visitorsBDaily = visitors[visitors['group'] == 'B'][['date', 'visitors']]
visitorsBDaily.columns = ['date', 'visitorsPerDateB']

# кумулятивная суума пользователей B по дням
visitorsBCummulative = visitorsBDaily.apply(
    lambda x: visitorsBDaily[visitorsBDaily['date'] <= x['date']].agg(
        {'date': 'max', 'visitorsPerDateB': 'sum'}
    ),
    axis=1,
)
visitorsBCummulative.columns = ['date', 'visitorsCummulativeB']

# пользователи группы A совершившие хотя бы 1 заказ
ordersADaily = (
    orders[orders['group'] == 'A'][['date', 'transactionId', 'visitorId', 'reve
    .groupby('date', as_index=False)
    .agg({'transactionId': pd.Series.nunique, 'revenue': 'sum'})
)
ordersADaily.columns = ['date', 'ordersPerDateA', 'revenuePerDateA']

# кумулятивные данные по группе A
ordersACummulative = ordersADaily.apply(
    lambda x: ordersADaily[ordersADaily['date'] <= x['date']].agg(
        {'date': 'max', 'ordersPerDateA': 'sum', 'revenuePerDateA': 'sum'}
    ),
    axis=1,
).sort_values(by=['date'])
ordersACummulative.columns = [
    'date',
    'ordersCummulativeA',
    'revenueCummulativeA',
]

# пользователи группы B совершившие хотя бы 1 заказ
ordersBDaily = (
    orders[orders['group'] == 'B'][['date', 'transactionId', 'visitorId', 'reve
    .groupby('date', as_index=False)
    .agg({'transactionId': pd.Series.nunique, 'revenue': 'sum'})
)
ordersBDaily.columns = ['date', 'ordersPerDateB', 'revenuePerDateB']

ordersBCummulative = ordersBDaily.apply(
    lambda x: ordersBDaily[ordersBDaily['date'] <= x['date']].agg(
```

```
        {'date': 'max', 'ordersPerDateB': 'sum', 'revenuePerDateB': 'sum'}
    ),
    axis=1,
).sort_values(by=['date'])
ordersBCummulative.columns = [
    'date',
    'ordersCummulativeB',
    'revenueCummulativeB',
]

# общая таблица с кумулятивными данными
data = (
    ordersADaily.merge(
        ordersBDaily, left_on='date', right_on='date', how='left'
    )
    .merge(ordersACummulative, left_on='date', right_on='date', how='left')
    .merge(ordersBCummulative, left_on='date', right_on='date', how='left')
    .merge(visitorsADaily, left_on='date', right_on='date', how='left')
    .merge(visitorsBDaily, left_on='date', right_on='date', how='left')
    .merge(visitorsACummulative, left_on='date', right_on='date', how='left')
    .merge(visitorsBCummulative, left_on='date', right_on='date', how='left')
)

display(data.head(5))
ordersByUsersA = (
    orders[orders['group'] == 'A']
    .groupby('visitorId', as_index=False)
    .agg({'transactionId': pd.Series.nunique})
)
ordersByUsersA.columns = ['userId', 'orders']

ordersByUsersB = (
    orders[orders['group'] == 'B']
    .groupby('visitorId', as_index=False)
    .agg({'transactionId': pd.Series.nunique})
)
ordersByUsersB.columns = ['userId', 'orders']
sampleA = pd.concat([ordersByUsersA['orders'],
                     pd.Series(0, index=np.arange(data['visitorsPerDateA'].sum(

sampleB = pd.concat([ordersByUsersB['orders'],
                     pd.Series(0, index=np.arange(data['visitorsPerDateB'].sum(
ordersByUsersA = (
    orders[orders['group'] == 'A']
    .groupby('visitorId', as_index=False)
    .agg({'transactionId': pd.Series.nunique})
)
ordersByUsersA.columns = ['userId', 'orders']

ordersByUsersB = (
    orders[orders['group'] == 'B']
    .groupby('visitorId', as_index=False)
    .agg({'transactionId': pd.Series.nunique})
)
ordersByUsersB.columns = ['userId', 'orders']

sampleA = pd.concat(
    [
        ordersByUsersA['orders'],
        pd.Series(
            0,
```

```
            index=np.arange(
                data['visitorsPerDateA'].sum() - len(ordersByUsersA['orders'])
            ),
            name='orders',
        ),
    ],
    axis=0,
)

sampleB = pd.concat(
    [
        ordersByUsersB['orders'],
        pd.Series(
            0,
            index=np.arange(
                data['visitorsPerDateB'].sum() - len(ordersByUsersB['orders'])
            ),
            name='orders',
        ),
    ],
    axis=0,
)
display("{0:.5f}".format(stats.mannwhitneyu(sampleA, sampleB)[1]))

display("{0:.3f}".format(sampleB.mean() / sampleA .mean() - 1))
```

	date	ordersPerDateA	revenuePerDateA	ordersPerDateB	revenuePerDateB	ordersCummula
0	2019-08-01	24	148579	21	101217	
1	2019-08-02	20	93822	24	165531	
2	2019-08-03	24	112473	16	114248	
3	2019-08-04	16	70825	17	108571	
4	2019-08-05	25	124218	23	92428	

```
'0.00840'
```

```
'0.138'
```

Вывод P-value значительно меньше 0.05, потому нулевую гипотезу отвергаем (различий между группами нет). Анализ сырых данных показывает, что в конверсии между группами есть статистически значимые различия. Относительный прирост конверсии группы B к конверсии группы A равен 13,8%

3.12 Посчитайте статистическую значимость различий в среднем чеке заказа между группами по «сырым» данным.

In [17]:

```
display('{0:.3f}'.format(stats.mannwhitneyu(orders[orders['group']=='A']['reven
                                            orders[orders['group']=='B']['reven
display('{0:.3f}'.format(orders[orders['group']=='B']['revenue'].mean()/
                         orders[orders['group']=='A']['revenue'].mean()-1))
```

```
'0.365'

'0.259'
```

Вывод P-value больше 0.05 - статистически значимых отличий в среднем чеке между группами нет.Хотя разница между группами значительна почти 26%. Это говорит нам о том что ввыборке есть выбросы, аномально дорогие заказы.

3.13 Посчитайте статистическую значимость различий в конверсии между группами по «очищенным» данным.

In [18]:

```
# выборка по количеству закзаов
usersWithManyOrders = pd.concat(
    [
        ordersByUsersA[ordersByUsersA['orders'] > 2]['userId'],
        ordersByUsersB[ordersByUsersB['orders'] > 2]['userId'],
    ],
    axis=0,
)

# выборка по сумме заказов
usersWithExpensiveOrders = orders[orders['revenue'] > 28000]['visitorId']
abnormalUsers = (
    pd.concat([usersWithManyOrders, usersWithExpensiveOrders], axis=0)
    .drop_duplicates()
    .sort_values()
)
display(abnormalUsers.head(5))
```

```
1099    148427295
18      199603092
928     204675465
23      237748145
37      249864742
dtype: int64
```

In [19]:

```
sampleAFiltered = pd.concat([ordersByUsersA[np.logical_not(ordersByUsersA['user
        pd.Series(0,
            index=np.arange(data['visitorsPerDateA'].sum() - len(ordersByUsersA
            name='orders')],axis=0,)

sampleBFiltered = pd.concat([ordersByUsersB[np.logical_not(ordersByUsersB['user
        ]['orders'],
        pd.Series(0, index=np.arange(data['visitorsPerDateB'].sum() - len(order
            ),
            name='orders')], axis=0,)
print('{0:.5f}'.format(stats.mannwhitneyu(sampleAFiltered, sampleBFiltered)[1])
print('{0:.3f}'.format(sampleBFiltered.mean()/sampleAFiltered.mean()-1))
```

```
0.00652
0.173
```

Вывод: P-value меньше 0.05 поэтому отвергаем нуливую гипотезу. Как и сырыми данными статистическая значимочть достигнута. Сегмент B значительно лучше сегмента A. (на 17%)

3.14 Посчитайте статистическую значимость различий в среднем чеке заказа между группами по «очищенным» данным.

In [20]:

```
print('{0:.3f}'.format(stats.mannwhitneyu(orders[np.logical_and(orders['group']
                    np.logical_not(orders['visitorId'].isin(abnormalUsers)))]['
            orders[np.logical_and(orders['group'] == 'B', np.logical_not(orders
                )]['revenue'])[1]))

print(
    "{0:.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
    )
)
```

```
0.369
-0.020
```

Выводы: По все видимости изменения которые проверяем не оказывают существенного влияния средний чек. Это хорошо показывают очищинные данные. P-value значительно больше 0.05

следовательно статистически значемых различий в среднем чеке нет. Да и относительная разница всего 2%

4 Общий вывод и решение по результатам теста

В рамках предоставленных данным мы не видим какие изменения в работе интернет магазина мы тестируем. Если изменения направленны на изменения среднего чека, то результат не достигнут и требуется дальнейшее продолжения теста. Визуально видно, что средний чек по группе B вырос. Правда сновнай рост достигнут благодаря аномальным заказам случившимся в период тестирования. Но тем не менен изменения среднего чека по "сырым" и по "очищенным" данным не показали стаистической значимости

Если же изменения направлены на изменения конверсии, то результат достигнут. Конверсия по группе B лучше, что и показали статистические тесты по "сырым" и очищенным данным. Как видно из графика кумулятивной конверсии по группам, показатели конверсии в конце месяца достаточно стабильна.