

# Table of Contents

- [1 Описание данных:](#)
- [2 Приоретизация гипотез](#)
- [3 Анализ A/B теста](#)
  - [3.1 Предобработка](#)
  - [3.2 График кумулятивной выручки по группам](#)
  - [3.3 График кумулятивного среднего чека по группам.](#)
  - [3.4 График относительного изменения кумулятивного среднего чека группы В к группе А.](#)
  - [3.5 График кумулятивной конверсии по группам](#)
  - [3.6 График относительного изменения кумулятивной конверсии группы В к группе А](#)
  - [3.7 Точечный график количества заказов по пользователям](#)
  - [3.8 Посчитаем выборочные 95-й и 99-й перцентили количества заказов на одного пользователя](#)
  - [3.9 Точечный график стоимости заказов](#)
  - [3.10 Посчитаем выборочные 95-й и 99-й перцентили стоимости заказов](#)
  - [3.11 Статистическая значимость различия в конверсии между группами по "сырым" данным.](#)
  - [3.12 Статистическая значимость различий в среднем чеке заказа между группами по «сырым» данным.](#)
  - [3.13 Статистическая значимость различий в конверсии между группами по «очищенным» данным.](#)
  - [3.14 Статистическая значимость различий в среднем чеке заказа между группами по «очищенным» данным](#)
- [4 Вывод](#)

## Название проекта: Анализ гипотез для увеличения выручки интернет магазина.

Как аналитику крупного интернет-магазина перед нами стоит задача. Вместе с отделом маркетинга мы подготовили список гипотез для увеличения выручки. Нам предстоит провести приоритизацию гипотез, запустить A/B-тест и проанализировать результаты.

### Описание данных:

#### Данные для первой части

Файл `/datasets/hypothesis.csv`. [Скачать датасет](#)

- `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.

#### Данные для второй части

Файл `/datasets/orders.csv`. [Скачать датасет](#)

- `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`. [Скачать датасет](#)

- `date` — дата;
- `group` — группа A/B-теста.

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

## Приоретизация гипотез

Импортируем все необходимые библиотеки

In [1]:

```
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
from scipy import stats as st
```

In [2]:

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

Out[2]:

|   | 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       |

In [3]:

```
data.columns = data.columns.str.lower()
```

Проведем приоритизацию гипотез с помощью фреймворка `ICE` (умножаем значимость для пользовательского опыта `impact` на нашу уверенность в этом `confidence` и делим на цену уровень стоимости проверки `efforts` )

In [4]:

```
data['ICE'] = round (data['impact']*data['confidence']/data['efforts'], 2) # рассчитываем ICE, результат записываем в отдельный столбец
pd.set_option('max_colwidth', 150) # увеличиваем максимальную ширину столбца, чтобы была возможность прочесть гипотезы полностью
display(data[['hypothesis', 'ICE']].sort_values(by='ICE', ascending=False)) # выводим на экран гипотезы в порядке приоритизации по ICE
```

|   | hypothesis  | ICE   |
|---|---|-------|
| 8 | Запустить акцию, дающую скидку на товар в день рождения   | 16.20 |
| 0 | Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей    | 13.33 |
| 7 | Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок      | 11.20 |
| 6 | Показать на главной странице баннеры с актуальными акциями и предложениями, чтобы увеличить конверсию | 8.00  |

|   |  | hypothesis | ICE  |
|---|--|------------|------|
| 2 | Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа |            | 7.00 |
| 1 | Запустить собственную службу доставки, что сократит срок доставки заказов                                    |            | 2.00 |
| 5 | Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов                     |            | 1.33 |
| 3 | Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар          |            | 1.12 |
| 4 | Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей                             |            | 1.00 |

Теперь воспользуемся фреймворком `RICE`. При использовании этого фреймворка в множителе числителя добавляем значение оценку количества пользователей, которых затронут изменения `reach`

In [5]:

```
data['RICE'] = data['reach']*data['impact']*data['confidence']/data['efforts'] # рассчитываем RICE, результат записываем в отдельный столбец
display(data[['hypothesis', 'RICE']].sort_values(by='RICE', ascending=False)) # выводим на экран гипотезы в порядке приоритизации по RICE
```

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

Для наглядности выведем рядом очередность приоритизации гипотез разными фреймворками

| ICE | RICE |
|-----|------|
| 8   | 7    |
| 0   | 2    |
| 7   | 0    |
| 6   | 6    |
| 2   | 8    |
| 1   | 3    |
| 5   | 1    |
| 3   | 5    |
| 4   | 4    |

Результаты несколько разнятся. Это связано с тем, что `RICE` учитывает оценку количества пользователей, которых затронут изменения и чем их больше, тем приоритет гипотезы выше. Для `ICE` этот критерий не учитывается.

Стоит отметить, что в обоих случаях можно выделить приоритетными гипотезы под номерами `7, 0, 6`. Несмотря на то, что их приоритет разнится при оценке разными фреймворками, в обоих случаях они вверху нашего "рейтинга". Гипотезы же `4, 5, 3, 1` в обоих случаях имеют низкий приоритет.

# Анализ A/B теста

## Предобработка

Для начала запишем данные из имеющихся датафреймов в переменные `orders` и `visitors` соответственно и посмотрим как выглядят таблицы.

In [6]:

```
orders = pd.read_csv('/datasets/orders.csv', sep=',')
orders.head()
```

Out[6]:

|   | 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 [7]:

```
visitors = pd.read_csv('/datasets/visitors.csv')
visitors.head()
```

Out[7]:

|   | 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      |

Приведем тип данных в столбцах `date` в обоих датафреймах к типу `datetime`

In [8]:

```
orders['date'] = orders['date'].map(
    lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
)
visitors['date'] = visitors['date'].map(
    lambda x: dt.datetime.strptime(x, '%Y-%m-%d')
)
```

Проверяем корректность разделения пользователей по группам.

In [9]:

```
# отбираем группы из данных с заказами
group_a=orders.query('group == "A"')
group_b=orders.query('group == "B"')

# получаем id пользователей, попавших в обе группы
group_intersections=list(np.intersect1d(group_a['visitorId'], group_b['visitorId']))
len(group_intersections)
```

Out[9]:

Получается **58** пользователей попали в обе группы. Это может исказить выводы дальнейшего исследования, поэтому от записей о покупках этих пользователей придется избавиться

In [10]:

```
# перезаписываем orders
orders = orders.query('visitorId not in @group_intersections')
```

Проверим, что в данных не осталось записей о пользователях попавших в обе группы

In [11]:

```
# отбираем группы из данных с заказами
group_a=orders.query('group == "A"')
group_b=orders.query('group == "B"')

# получаем пользователей, попавших в обе группы
group_intersections=list(np.intersect1d(group_a['visitorId'], group_b['visitorId']))
len(group_intersections)
```

Out[11]:

0

Чтобы построить графики, нужно собрать кумулятивные данные. Создадим датафрейм `cumulativeData` со столбцами:

- `date` — дата;
- `group` — группа **A/B**-теста (**A** или **B**);
- `orders` — кумулятивное количество заказов на указанную дату в указанной группе;
- `buyers` — кумулятивное количество пользователей, совершивших хотя бы один заказ, на указанную дату в указанной группе;
- `revenue` — кумулятивная выручка на указанную дату в указанной группе (средний чек);
- `visitors` — кумулятивное количество посетителей интернет-магазина на указанную дату в определённой группе.

Для этого вначале подготовимся.

In [12]:

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

Получим строки таблицы `orders`, дата которых меньше или равна дате элемента из `datesGroups`, а группа теста равна группе из `datesGroups`.

Агрегируем значения. Вычислим максимальную дату. Для группы тоже рассчитаем максимум, и хотя она будет определённой, сделаем это, чтобы столбец `group` не пропал из итогового вывода. Найдём число уникальных **ID** заказов и пользователей. Подсчитаем сумму средних чеков

Применим методы к каждой строке датафрейма и отсортируем результаты по столбцам `date` и `group`.

Результат запишем в датафрейм `ordersAgregated`

In [13]:

```
ordersAgregated = 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'])
```

Аналогично `visitorsAgregated`

In [14]:

```
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'])
```

Таким образом в `ordersAggregated` и `visitorsAggregated` будут находиться такие данные как будто мы ежедневно считали результаты тестирования до выбранного дня включительно и сохраняли их в строках таблиц.

In [15]:

```
# 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']

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

|   | date       | group | orders | buyers | revenue | visitors |
|---|------------|-------|--------|--------|---------|----------|
| 0 | 2019-08-01 | A     | 23     | 19     | 142779  | 719      |
| 1 | 2019-08-01 | B     | 17     | 17     | 59758   | 713      |
| 2 | 2019-08-02 | A     | 42     | 36     | 234381  | 1338     |
| 3 | 2019-08-02 | B     | 40     | 39     | 221801  | 1294     |
| 4 | 2019-08-03 | A     | 66     | 60     | 346854  | 1845     |

## График кумулятивной выручки по группам

In [16]:

```
# параметры визуализации
sns.set(font_scale=2, style='whitegrid', rc={'figure.figsize':(20,7)})
```

In [17]:

```
# 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']]

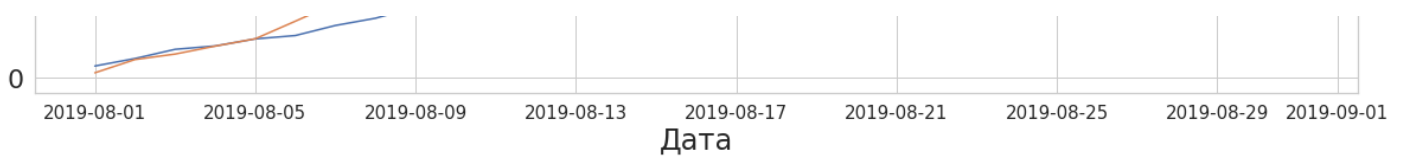
# Строим график выручки группы A
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue'], label='A')

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

# размер шрифта по оси x
plt.tick_params(axis='x', which='major', labelsize=15)

plt.title('График кумулятивной выручки по группам')
plt.xlabel('Дата')
plt.ylabel('Выручка')
plt.legend();
```





Вывод:

- выручка стабильно растет на протяжении всего теста в обеих группах;
- в начале теста метрика немного колеблется, но показывает близкие значения в обеих группах;
- у группы В наблюдается резкий всплеск после **17.08**, который может быть связан с резким увеличением заказов в этот день, либо о появлении очень дорогих заказов в выборке;
- после выброса в группе В выручки в обеих группах стабилизируются и растут практически одинаково.

Однозначно утверждать, что группа В лучше на данном шаге нельзя.

## График кумулятивного среднего чека по группам.

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

In [18]:

```
# график среднего чека группы A
plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']/cumulativeRevenueA['orders'], label='A')

# график среднего чека группы B
plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']/cumulativeRevenueB['orders'], label='B')

# размер шрифта по оси x
plt.tick_params(axis='x', which='major', labels=15)

plt.title('График кумулятивного среднего чека по группам')
plt.xlabel('Дата')
plt.ylabel('Кумулятивный средний чек')
plt.legend();
```



Вывод:

- В целом к концу теста метрика стабилизируется в обеих группах.
- Группа А, в отличие от группы В более равномерна, к концу теста средний чек почти сравнялся с максимальным значением, достигнутым 13 августа.
- Выделяется аномальный рост среднего чека 18-19 августа по группе В, а затем снижение. Возможно, в группу В в первой половине теста попали крупные заказы (резкий всплеск на графике). Тогда ей нужно больше данных, чтобы прийти к реальному среднему чеку и установиться на его уровне;

## График относительного изменения кумулятивного среднего чека группы В к группе А.

Построим график относительного различия для среднего чека.

In [19]:

```
# 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='--')

# размер шрифта по оси x
plt.tick_params(axis='x', which='major', labelsize=15)

plt.title('График относительного изменения кумулятивного среднего чека группы В к группе А')
plt.xlabel('Дата')
plt.ylabel('Кумулятивный средний чек');
```



Вывод:

- В начале теста лучше себя чувствовали показатели группы В, затем они стали ухудшаться относительно А
- Ближе к середине теста **13.08** А достигает своих лучших показателей, поэтому график опускается в худшую за время теста для В точку
- Однако важно отметить, что на графике различия между сегментами резко «скачет» в некоторых днях, например **05.08** или **08.08**, но особенно выделяется, конечно скачок **18.08**. Очевидно есть выбросы.
- После выброса метрика снижается и стремится стабилизироваться.

## График кумулятивной конверсии по группам

Теперь посмотрим на кумулятивную конверсию

In [20]:

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



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

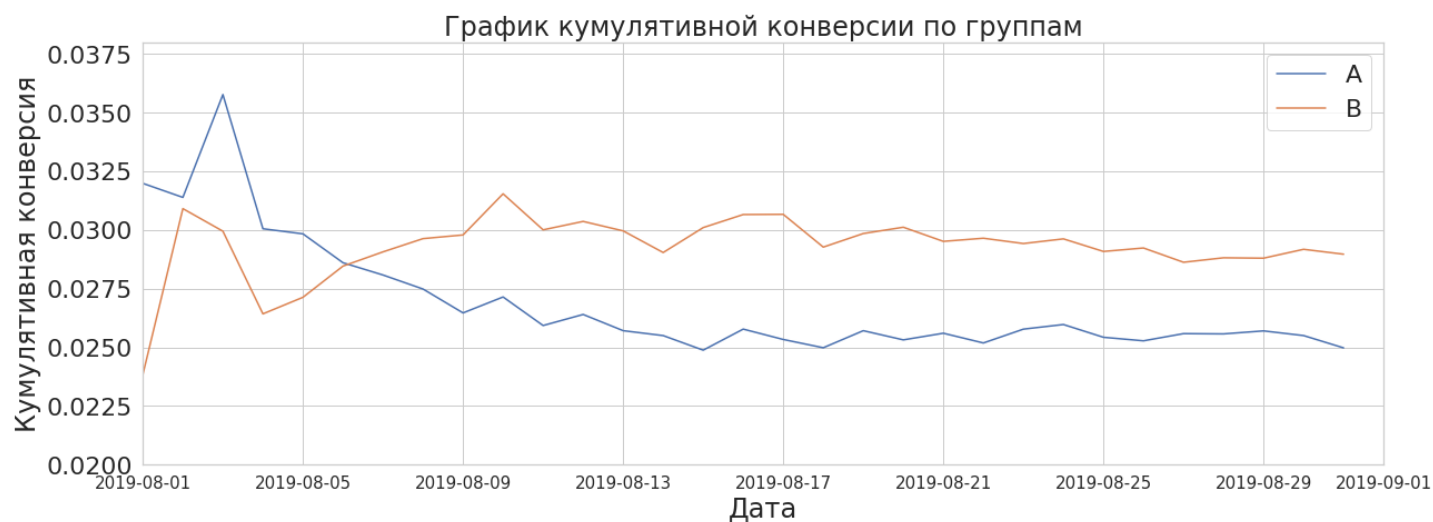
# отделяем данные по группе B
cumulativeDataB = cumulativeData[cumulativeData['group']=='B']

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

# размер шрифта по оси x
plt.tick_params(axis='x', which='major', labelsize=15)

# задаем масштаб осей
plt.axis([dt.datetime(2019, 8, 1), dt.datetime(2019, 9, 1), 0.020, 0.038])

plt.title('График кумулятивной конверсии по группам')
plt.xlabel('Дата')
plt.ylabel('Кумулятивная конверсия');
```



Вывод:

- В начале теста наблюдаем заметные колебания. В начале конверсия у группы A была больше группы B, но уже к концу первой недели тестирования конверсия группы B выросла, а группы A снизилась, после чего графики начали стремиться к стабилизации
- К 11-12 августа различие в конверсии у обеих групп становится очевидно, и начинает фиксироваться.
- Примерно к 20 августа колебания максимально сглаживаются
- В целом, конверсия у группы B стабильно лучше по накопленным показателям чем A

## График относительного изменения кумулятивной конверсии группы B к группе A

Построим график относительного изменения кумулятивных конверсий

In [21]:

```
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.legend()

plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=0.1, color='grey', linestyle='--')
plt.axis([dt.datetime(2019, 8, 1), dt.datetime(2019, 9, 1), -0.3, 0.3])
```

```
# размер шрифта по оси x
plt.tick_params(axis='x', which='major', labelsize=15)

plt.title('График относительного изменения кумулятивной конверсии группы В к группе А')
plt.xlabel('Дата')
plt.ylabel('Кумулятивная конверсия');
```



Вывод:

- С самого начала теста, метрика группа В меньше группы А, но с 6 августа вырвалась вперед и стабильно росла до своего пика **15.08 в 21%**, далее метрика начинается снижаться и стремиться к стабилизации.
- В целом отношение конверсии стремится примерно к уровню **13-17%** в пользу В, но в последние дни теста еще растет. Скорее всего отношение конверсии еще полностью не установилось, и сейчас делать какие-либо однозначные выводы по тесту еще нельзя, но на первый взгляд В выглядит интереснее в плане конверсии.

### Точечный график количества заказов по пользователям

Для начала посмотрим топ количества заказов в табличном виде

```
In [22]:
ordersByUsers = (orders.groupby('visitorId', as_index=False).agg({'transactionId': pd.Series.nunique}))
ordersByUsers.columns = ['visitorId', 'orders']

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

Out[22]:
```

|     | visitorId  | orders |
|-----|------------|--------|
| 908 | 3967698036 | 3      |
| 55  | 249864742  | 3      |
| 478 | 2108163459 | 3      |
| 687 | 2988190573 | 3      |
| 890 | 3908431265 | 3      |
| 138 | 611059232  | 3      |
| 632 | 2742574263 | 3      |
| 157 | 678354126  | 2      |
| 323 | 1404560065 | 2      |
| 452 | 1985475298 | 2      |

Теперь построим точечный график количества заказов по пользователям

In [23]:

```
# серия из чисел от 0 до количества наблюдений в ordersByUsers
x_values = pd.Series(range(0, len(ordersByUsers)))
plt.scatter(x_values, ordersByUsers['orders']);

plt.title('Точечный график количества заказов по пользователям')
plt.ylabel('Количество заказов');
```



Вывод:

Есть пользователи, которые совершали по **1, 2, 3** заказов. Чаще всего **1** заказ, не редки случаи и **2**-х заказов. Их точная доля не ясна — непонятно, считать их аномалиями или нет. Для этого необходимо посчитать выборочные перцентили количества заказов на одного пользователя.

Посчитаем выборочные **95-й** и **99-й** перцентили количества заказов на одного пользователя

In [24]:

```
print(np.percentile(ordersByUsers['orders'], [95, 99]))
```

[1. 2.]

Вывод:

Не более **5%** пользователей оформляли больше чем один и не более **1%** пользователей - больше двух заказов. Целесообразно выбрать **1** заказ на одного пользователя за нижнюю границу "нормального" числа заказов, и отсеять аномальных пользователей.

Точечный график стоимости заказов

Для начала посмотрим в табличном виде топ по стоимости заказов

In [25]:

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

Out[25]:

|      | transactionId | visitorId  | date       | revenue | group |
|------|---------------|------------|------------|---------|-------|
| 425  | 590470918     | 1920142716 | 2019-08-19 | 1294500 | B     |
| 1196 | 3936777065    | 2108080724 | 2019-08-15 | 202740  | B     |
| 448  | 600016100     | 1007000100 | 2019-08-10 | 20550   | A     |

| id   | transactionid | visitorid  | date       | revenue | group |
|------|---------------|------------|------------|---------|-------|
| 744  | 366308183     | 888512513  | 2019-08-27 | 86620   | B     |
| 743  | 3603576309    | 4133034833 | 2019-08-09 | 67990   | A     |
| 1103 | 1348774318    | 1164614297 | 2019-08-12 | 66350   | A     |
| 1099 | 316924019     | 148427295  | 2019-08-12 | 65710   | A     |
| 949  | 1347999392    | 887908475  | 2019-08-21 | 60450   | A     |
| 940  | 2420050534    | 4003628586 | 2019-08-08 | 58550   | B     |
| 131  | 3163614039    | 2254586615 | 2019-08-22 | 53904   | A     |

даже в этой таблице уже виден пользователь из группы **B**, который скорее всего и стал причиной аномального всплеска, потратив **1 294 500** рублей

Теперь построим точечный график

In [26]:

```
x_values = pd.Series(range(0, len(orders['revenue'])))
plt.scatter(x_values, orders['revenue']);
plt.title('Точечный график стоимостей заказов')
plt.ylabel('Стоимость заказа в млн.');
```



Вывод:

На графике зафиксирован один самый крупный заказ на сумму **1 294 500**. Далее виден заказ в размере **202 740**. Все остальные заказы ниже **92 550**. Для того, чтобы точно определить стоимость заказа, которую следует считать аномальной посчитаем выборочные перцентили.

## Посчитаем выборочные 95-й и 99-й перцентили стоимости заказов

In [27]:

```
print(np.percentile(orders['revenue'], [95, 99]))
[26785. 53904.]
```

Вывод:

Не более, чем у **5%** пользователей чек дороже **26 785**, и только не более **1%** пользователей оформили на чек дороже **53 904**. Границей для определения аномальных заказов следует обозначить до **26 785**.

**Статистическая значимость различия в конверсии между группами по "сырым" данным.**

Перед тестом Манна-Уитни проверим гипотезу о нормальном распределении с помощью критерия Шапиро-Уилка:

$$\begin{cases} H_0 \\ : \text{Конверсии группы A распределены нормально} \\ H_1 \\ : \text{Конверсии группы A не распределены нормально} \end{cases}$$

$\alpha=5\%$  критический уровень статистической значимости.

Аналогично для группы B.

In [28]:

```
visitorsADaily = visitors[visitors['group'] == 'A'][['date', 'visitors']]
visitorsADaily.columns = ['date', 'visitorsPerDateA']

visitorsACummulative = visitorsADaily.apply(
    lambda x: visitorsADaily[visitorsADaily['date'] <= x['date']].agg(
        {'date': 'max', 'visitorsPerDateA': 'sum'}
    ),
    axis=1,
)
visitorsACummulative.columns = ['date', 'visitorsCummulativeA']

visitorsBDaily = visitors[visitors['group'] == 'B'][['date', 'visitors']]
visitorsBDaily.columns = ['date', 'visitorsPerDateB']

visitorsBCummulative = visitorsBDaily.apply(
    lambda x: visitorsBDaily[visitorsBDaily['date'] <= x['date']].agg(
        {'date': 'max', 'visitorsPerDateB': 'sum'}
    ),
    axis=1,
)
visitorsBCummulative.columns = ['date', 'visitorsCummulativeB']

ordersADaily = (
    orders[orders['group'] == 'A'][['date', 'transactionId', 'visitorId', 'revenue']]
    .groupby('date', as_index=False)
    .agg({'transactionId': pd.Series.nunique, 'revenue': 'sum'})
)
ordersADaily.columns = ['date', 'ordersPerDateA', 'revenuePerDateA']

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',
]

ordersBDaily = (
    orders[orders['group'] == 'B'][['date', 'transactionId', 'visitorId', 'revenue']]
    .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',
    ]

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')
)

# для пользователей, совершивших хотя бы 1 заказ, будет указано число заказов
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']

# пользователям с заказами будет соответствовать число заказов пользователя, а пользо-
# вателям без заказов — нули
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)

# зададим уровень значимости 5%
alpha=0.05
p=st.shapiro(sampleA)
print("Shapiro-Wilk normality test, W-statistic: %f, p-value: %f" % p)
if p[1] > alpha:
    print('Принять гипотезу о нормальности распределении группа A')
else:
    print('Отклонить гипотезу о нормальности распределении группа A')

p=st.shapiro(sampleB)
print("Shapiro-Wilk normality test, W-statistic: %f, p-value: %f" % p)
if p[1] > alpha:
    print('Принять гипотезу о нормальности распределении группа B')
else:
    print('Отклонить гипотезу о нормальности распределении группа B')

Shapiro-Wilk normality test, W-statistic: 0.132646, p-value: 0.000000
Отклонить гипотезу о нормальности распределении группа A
Shapiro-Wilk normality test, W-statistic: 0.150587, p-value: 0.000000
Отклонить гипотезу о нормальности распределении группа B

/opt/conda/lib/python3.9/site-packages/scipy/stats/_morestats.py:1761: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")

```

Итак данные в обеих группах данные не подходят под нормальное распределение, поэтому воспользуемся **U-критерием Манна — Уитни**

$$\left\{ \begin{array}{l} H_0 \\ H_1 \end{array} \right. : \text{Конверсии групп A и B равны, статистически значимых отличий нет}$$

$\{$  : Конверсии групп А и В различны, статистически значимые отличия есть

$\alpha=5\%$  критический уровень статистической значимости.

Применим тест **U**-критерий Манна — Уитни

In [29]:

```
print("{0:.3f}".format(stats.mannwhitneyu(sampleA, sampleB)[1]))  
print("{0:.2f}".format(sampleB.mean() / sampleA.mean() - 1))
```

0.011  
0.16

Вывод:

- Первое число — **p-value = 0.011** меньше **0.05**. Значит, нулевую гипотезу о том, что конверсии равны и статистически значимых различий в конверсии между группами нет **отвергаем**. Конверсии различаются значимо.
- Относительный проигрыш группы **А** составляет **16%**

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

$$\left\{ \begin{array}{l} H_0 \\ \text{: Средние чеки групп А и В равны, статистически значимых отличий нет} \\ H_1 \\ \text{: Средние чеки групп А и В различны, статистически значимые отличия есть} \end{array} \right.$$

$\alpha=5\%$  критический уровень статистической значимости.

In [30]:

```
print('{0:.3f}'.format(stats.mannwhitneyu(orders[orders['group']=='A']['revenue'], orders[orders['group']=='B']['revenue'])[1]))  
print('{0:.2f}'.format(orders[orders['group']=='B']['revenue'].mean()/orders[orders['group']=='A']['revenue'].mean()-1))
```

0.829  
0.29

Вывод:

- Первое число — **p-value = 0.829** значительно больше **0.05**. Значит, нулевую гипотезу о том, что средние чеки равны и статистически значимых различий между группами в размере среднего чека нет **принимает**.
- При этом средний чек группы **В** значительно выше чем у **А**, почти на **29%**, но тут свою роль могли сыграть выбросы, проверим это далее на очищенных данных

**Статистическая значимость различий в конверсии между группами по «очищенным» данным.**

Как мы определились ранее, за аномальных пользователей примем тех, кто совершил более **1** заказа или совершил заказ дороже **26 785** рублей. Сделаем срезы пользователей с числом заказов больше **2** — `usersWithManyOrders` и пользователей, совершивших заказы дороже **26 785** — `usersWithExpensiveOrders`. Объединим их в таблице `abnormalUsers`.

In [31]:

```
usersWithManvOrders = pd.concat(
```

```

[
    ordersByUsersA[ordersByUsersA['orders'] > 1]['userId'],
    ordersByUsersB[ordersByUsersB['orders'] > 1]['userId'],
],
axis=0,
)
usersWithExpensiveOrders = orders[orders['revenue'] > 26785]['visitorId']
abnormalUsers = (
    pd.concat([usersWithManyOrders, usersWithExpensiveOrders], axis=0)
    .drop_duplicates()
    .sort_values()
)
print(abnormalUsers.head(5))
print(abnormalUsers.shape[0])

```

```

568      113298937
1099      148427295
928       204675465
33        249864742
684       358944393
dtype: int64
86

```

Получаем 86 аномальных пользователей, проверим как они повлияли на результат теста.

Посчитаем статистическую значимость различий в конверсии между группами теста по очищенным данным. Сначала подготовим выборки количества заказов по пользователям по группам теста:

$$\left\{ \begin{array}{ll} H_0 & \\ : \text{Конверсии групп A и B равны, статистически значимых отличий нет} & \\ H_1 & \\ : \text{Конверсии групп A и B различны, статистически значимые отличия есть} & \end{array} \right.$$

$\alpha=5\%$  критический уровень статистической значимости.

In [32]:

```

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

sampleBFiltered = pd.concat(
    [
        ordersByUsersB[
            np.logical_not(ordersByUsersB['userId'].isin(abnormalUsers))
        ]['orders'],
        pd.Series(
            0,
            index=np.arange(
                data['visitorsPerDateB'].sum() - len(ordersByUsersB['orders'])
            ),
            name='orders',
        ),
    ],
    axis=0,
)

```



In [33]:

```
print('{0:.3f}'.format(stats.mannwhitneyu(sampleAFiltered, sampleBFiltered)[1]))
print('{0:.2f}'.format(sampleBFiltered.mean()/sampleAFiltered.mean()-1))
```

```
0.016
0.17
```

Вывод:

Результаты по конверсии значимо не изменились. **p-value** стал чуть больше на **0,005**, что никак не меняет наших выводов. Проигрыш группы **A** на очищенных данных немного увеличился **17%** против **16%** на сырых

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

$$\begin{cases} H_0 \\ : \text{Средние чеки групп A и B равны, статистически значимых отличий нет} \\ H_1 \\ : \text{Средние чеки групп A и B различны, статистически значимые отличия есть} \end{cases}$$

$\alpha=5\%$  критический уровень статистической значимости.

In [34]:

```
print(
    '{0:.3f}'.format(
        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' ],
        )[1]
    )
)

print(
    "{0:.2f}".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
    )
)
```

```
0.727
-0.03
```

## Вывод:

- **p-value** уменьшился почти на **0.1**, но по прежнему значительно выше **0,05**
- Особенно нужно обратить внимание на разницу между средним чеком. Она упала с **29%** в пользу **В** до **3%** в пользу **А**! Это еще раз заставляет убедиться, что статистически значимых различий по этой метрике между группами нет, а наблюдаемая на неочищенных данных разница как мы и предполагали была связана с выбросами.

## Вывод

Резюмируем имеющиеся факты:

- есть статистически значимые различия по конверсии между группами (как по сырым так и по очищенным данным), при этом преимущество у группы **В** порядка **16-17%**;
- нет статистически значимых различий по среднему чеку между группами (как по сырым так и по очищенным данным);
- график различия конверсий между группами показывает, что результаты группы **В** лучше группы **А** в конце исследования приблизительно на **13-17%**, при этом есть тенденция к еще небольшому улучшению;
- график различий кумулятивного среднего чека говорит нам о преимуществе группы **В** только из - за выброса, после чего метрика начинает снижаться к более естественному положению

Исходя из обнаруженных фактов тест следует остановить и признать его успешным в части улучшения конверсии в целевой группе **В**. В части среднего чека нельзя утверждать, что какая - либо из групп имеет преимущество над другой. Оснований полагать, что при продолжении теста разница между группами в конверсиях либо среднем чеке значимо изменятся нет, т.к. колебания в графиках носят затухающий характер.