ГЛЕБ ЗАЙЦЕВ

ВОРОНКИ В SQL

18 задач для аналитика продукта

```
WITH flags AS (...)
SELECT SUM(view) viewers,
       SUM(cart) carts,
       SUM(pay) pays
FROM flags;
```

просмотр · корзина · оплата

конверсия · потери · сегменты

все запросы запускаются в SQLite

Глеб Зайцев

Воронки в SQL. 18 задач для аналитика продукта

https://litres.ru/74419834
SelfPub; 2026

Аннотация

Практическая книга по продуктовым воронкам в SQL. Внутри 18 задач на маленьких таблицах: просмотры, корзины, оплаты, флаги пользователей, конверсии между шагами, потери, порядок событий, сегменты по каналам, версиям карточки и устройствам. Все запросы запускаются в SQLite.

Содержание

Глеб Зайцев
Воронки в SQL. 18 задач для аналитика продукта

Перед первой задачей

Почему воронка ломается от неправильного зерна

Просмотры, корзины и оплаты можно считать событиями, сессиями или пользователями — и это будут разные ответы.

Перед расчётом нужно решить, что является единицей анализа: пользователь, сессия, товар, день или версия карточки.

Эта книга показывает компактные SQL-приёмы, которые помогают не спутать клики с людьми и оплату с прохождением полного сценария.

Как читать задачи

Сначала определите шаги воронки и проверьте, можно ли их пропускать.

Затем соберите флаги по выбранной единице анализа и только после этого считайте конверсии.

Все примеры учебные: они не требуют внешних сервисов, файлов или личных данных покупателей.

Маршрут по задачам

Практические задачи

Задача 1. Посчитать шаги воронки по событиям

Рабочий вопрос

Нужно увидеть, сколько пользователей дошло до просмотра, корзины и оплаты, чтобы оценить первый разрыв в сценарии.

Данные

```
  CREATE TABLE events(user_id INTEGER,
event_name TEXT);
  INSERT INTO events VALUES
  (1,'view'),(1,'cart'),(1,'pay'),
(2,'view'),(2,'cart'),(3,'view'),
(4,'view'),(4,'pay');
```

Запрос

```
  SELECT    event_name,    COUNT(DISTINCT
user_id) AS users
  FROM events
  WHERE           event_name           IN
('view','cart','pay')
  GROUP BY event_name
  ORDER BY CASE event_name WHEN 'view' THEN
1 WHEN 'cart' THEN 2 ELSE 3 END;
```

Ожидаемый результат

```
  [["view", 4], ["cart", 2], ["pay", 2]]
```

Почему работает

COUNT(DISTINCT user_id) считает людей на каждом шаге, а не число событий, поэтому повторные клики не завышают воронку.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

ORDER BY или явная агрегация делают вывод устойчи-

вым: строки не зависят от случайного порядка хранения данных.

Где легко ошибиться

Переходы между шагами здесь ещё не проверяются по времени; это срез наличия событий в периоде.

Самостоятельная проверка

Почему pay равен двум пользователям? Объясните ответ, опираясь на строки исходных данных.

Ответ: Оплату имеют пользователи 1 и 4.

Задача 2. Построить флаги прохождения шагов

Рабочий вопрос

Нужно собрать по одному ряду на пользователя: был просмотр, была корзина, была оплата.

Данные

```
  CREATE TABLE events(user_id INTEGER,
event_name TEXT);
  INSERT INTO events VALUES
  (1,'view'),(1,'cart'),(1,'pay'),
(2,'view'),(2,'cart'),(3,'view');
```

Запрос

```
  SELECT user_id,
  MAX(event_name = 'view') AS had_view,
  MAX(event_name = 'cart') AS had_cart,
  MAX(event_name = 'pay') AS had_pay
  FROM events
```

```
GROUP BY user_id
ORDER BY user_id;
```

Ожидаемый результат

```
[[1, 1, 1, 1], [2, 1, 1, 0], [3, 1, 0, 0]]
```

Почему работает

Логические выражения в SQLite возвращают 0 или 1, а MAX превращает события пользователя в флаг наличия шага.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

ORDER BY или явная агрегация делают вывод устойчивым: строки не зависят от случайного порядка хранения данных.

Где легко ошибиться

Если один пользователь совершил шаг несколько раз, флаг останется 1 и не превратится в счётчик кликов.

Самостоятельная проверка

Какой had_pay у пользователя 2? Объясните ответ, опираясь на строки исходных данных.

Ответ: 0, потому что у пользователя 2 нет события pay.

Задача 3. Посчитать конверсию из просмотра в корзину

Рабочий вопрос

Нужно получить долю пользователей с корзиной среди пользователей с просмотром, чтобы оценить качество карточки или оффера.

Данные

```
  CREATE TABLE events(user_id INTEGER,
event_name TEXT);
  INSERT INTO events VALUES
  (1,'view'),(1,'cart'),(2,'view'),
(2,'cart'),(3,'view'),(4,'view');
```

Запрос

```
  WITH flags AS (
  SELECT user_id,
  MAX(event_name = 'view') AS had_view,
  MAX(event_name = 'cart') AS had_cart
```

```
  FROM events
  GROUP BY user_id
  )
  SELECT  ROUND(1.0  *  SUM(had_cart)  /
SUM(had_view), 2) AS view_to_cart
  FROM flags;
```

Ожидаемый результат

```
  [[0.5]]
```

Почему работает

Флаги дают одного пользователя в знаменатель один раз, а дробное деление показывает конверсию.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

ORDER BY или явная агрегация делают вывод устойчивым: строки не зависят от случайного порядка хранения данных.

Где легко ошибиться

Если считать строки событий, активные пользователи с

повторными событиями могут исказить долю.

Самостоятельная проверка

Чему равна конверсия в этих данных? Объясните ответ, опираясь на строки исходных данных.

Ответ: 0.5: два пользователя с корзиной из четырёх с просмотром.

Задача 4. Посчитать конверсию из корзины в оплату

Конец ознакомительного фрагмента.