ГЛЕБ ЗАЙЦЕВ

HAVING В SQL

18 задач для отчётов

```
SELECT category, SUM(revenue)
FROM orders
GROUP BY category
HAVING SUM(revenue) > 1000;
```

COUNT · SUM · AVG · маржа

повторные покупки · сегменты

все запросы запускаются в SQLite

Глеб Зайцев

HAVING в SQL. 18 задач для отчётов

«Автор»
2026

Зайцев Г.

HAVING в SQL. 18 задач для отчётов / Г. Зайцев — «Автор», 2026

Практическая книга по HAVING в SQL для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: фильтрация групп, повторные покупки, маржа, конверсия, средний чек, остатки, бюджеты, сегменты и аудит проблемных заказов. Все запросы запускаются в SQLite, данные встроены прямо в книгу.

Содержание

Глеб Зайцев
HAVING в SQL. 18 задач для отчётов

Перед первой задачей

Почему HAVING часто путают с WHERE

WHERE фильтрует отдельные строки до группировки, а HAVING фильтрует уже собранные группы.

Если вопрос звучит как «оставить категории, где сумма больше порога» или «клиентов с двумя заказами», почти всегда нужен GROUP BY и HAVING.

Эта книга тренирует не абстрактный синтаксис, а реальные отчётные вопросы: выручка, маржа, повторные покупки, достаточная выборка и проблемные заказы.

Как работать с задачами

Сначала прочитайте бизнес-вопрос и определите зерно отчёта: категория, клиент, дата, источник или товар.

Затем посмотрите, какие агрегаты участвуют в условии: COUNT, SUM, AVG или COUNT DISTINCT.

Все примеры маленькие и запускаются в SQLite, поэтому результат можно проверить вручную и только потом переносить идею в рабочие данные.

Маршрут по задачам

Практические задачи

Задача 1. Оставить категории с двумя и более заказами

Рабочий вопрос

Нужно найти категории, где было хотя бы два заказа, чтобы не делать выводы по единичной покупке.

Данные

```
    CREATE TABLE orders(category TEXT, order_id INTEGER, revenue
INTEGER);
    INSERT INTO orders VALUES
    ('books',1,300),('books',2,500),('courses',3,1200),
('templates',4,200),('templates',5,250);
```

Запрос

```
    SELECT category, COUNT(*) AS orders_count
    FROM orders
    GROUP BY category
    HAVING COUNT(*) >= 2
    ORDER BY category;
```

Ожидаемый результат

```
    [["books", 2], ["templates", 2]]
```

Почему работает

HAVING фильтрует уже собранные группы, поэтому условие COUNT(*) >= 2 применяется после GROUP BY.

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 написать COUNT(*) в WHERE, SQLite выдаст ошибку: агрегаты нельзя использовать до того, как группы собраны.

Самостоятельная проверка

Какая категория исчезнет, если порог поднять до трёх заказов?
Ответ: Исчезнут обе категории из результата, потому что ни у одной нет трёх заказов.

Задача 2. Найти продавцов с выручкой от 1000 рублей

Рабочий вопрос

Нужно быстро выделить продавцов, у которых суммарная выручка уже достаточно большая для отдельного разбора.

Данные

```
CREATE TABLE sales(seller TEXT, revenue INTEGER);
INSERT INTO sales VALUES
('Анна',400),('Анна',700),('Борис',900),('Вера',300),
('Вера',500);
```

Запрос

```
SELECT seller, SUM(revenue) AS total_revenue
FROM sales
GROUP BY seller
HAVING SUM(revenue) >= 1000
ORDER BY seller;
```

Ожидаемый результат

```
[["Анна", 1100]]
```

Почему работает

SUM(revenue) считается внутри каждого продавца, а HAVING оставляет только группы с нужной суммой.

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

Фильтрация отдельных строк через WHERE revenue >= 1000 потеряла бы продавца Анна, хотя сумма её двух продаж больше порога.

Самостоятельная проверка

Почему Борис не попал в результат? Объясните ответ, опираясь на строки исходных данных.

Ответ: У Бориса одна продажа на 900 рублей, а это меньше порога 1000 рублей.

Задача 3. Отобрать товары с высоким средним чеком

Рабочий вопрос

Нужно найти товары, у которых средний чек выше 400 рублей, чтобы сравнить их с массовыми дешёвыми товарами.

Данные

```
CREATE TABLE order_items(sku TEXT, total INTEGER);
INSERT INTO order_items VALUES
('A1',300),('A1',500),('B2',900),('C3',150),('C3',250);
```

Запрос

```
SELECT sku, ROUND(AVG(total), 1) AS avg_total
FROM order_items
GROUP BY sku
HAVING AVG(total) > 400
ORDER BY sku;
```

Ожидаемый результат

```
[["B2", 900.0]]
```

Почему работает

AVG(total) считается по каждой группе SKU, а ROUND нужен только для аккуратного отображения результата.

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

Конец ознакомительного фрагмента.