Глеб Зайцев

CTE в SQL. 18 задач с WITH для аналитика

Глеб Зайцев

CTE в SQL. 18 задач с WITH для аналитика

https://litres.ru/74419472
SelfPub; 2026

Аннотация

Практическая книга по CTE и WITH в SQL для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: именованные шаги, цепочки расчётов, повторное использование агрегатов, последние события, проценты от итога, проверка дублей, анти-join и простые рекурсивные последовательности. Все примеры запускаются в SQLite.

Содержание

Глеб Зайцев
CTE в SQL. 18 задач с WITH для аналитика

Перед первой задачей

Зачем выносить шаги в WITH

CTE помогает дать имя промежуточному результату и читать длинный запрос как последовательность маленьких решений.

В аналитике это почти всегда снижает риск: отдельно собрать оплаченные заказы, отдельно посчитать итоги, отдельно добавить долю или фильтр.

В этой книге WITH используется только там, где он делает отчёт прозрачнее. Мы не прячем сложность, а раскладываем её на проверяемые слои.

Как проверять себя

Сначала прочитайте каждый CTE как временную таблицу: какие колонки она возвращает и сколько строк в ней должно быть.

Затем проверьте финальный SELECT. Ошибки обычно появляются на стыке: перепутали зерно, повторно размножили строки или взяли среднее не на том уровне.

Все запросы написаны для SQLite. Конструкция WITH и большинство приёмов переносимы в PostgreSQL, MySQL и другие SQL-системы.

Маршрут по задачам

Практические задачи

Задача 1. Вынести оплаченные заказы в отдельный шаг

Рабочий вопрос

Нужно посчитать выручку только по оплаченным заказам, но сделать фильтр явным и читаемым.

Данные

```
  CREATE TABLE orders(id INTEGER, status
TEXT, total INTEGER);
  INSERT INTO orders VALUES (1,'paid',500),
(2,'new',700),(3,'paid',300),
(4,'cancelled',900);
```

Запрос

```
  WITH paid_orders AS (
```

```
SELECT id, total
FROM orders
WHERE status = 'paid'
)
SELECT SUM(total) AS paid_revenue
FROM paid_orders;
```

Ожидаемый результат

```
[[800]]
```

Почему работает

CTE paid_orders содержит только оплаченные строки.

Финальный запрос считает сумму уже по очищенному набору.

Так проще увидеть, что new и cancelled не участвуют в выручке.

Где легко ошибиться

Если фильтр оставить внутри большого финального запроса, легко случайно смешать оплаченные и неоплаченные строки.

Самостоятельная проверка

Добавьте paid-заказ на 200. Каким станет paid_revenue и почему?

Ответ: paid_revenue станет 1000, потому что новый заказ попадёт в paid_orders.

Задача 2. Собрать выручку по клиентам

Рабочий вопрос

Нужно сначала получить итог по каждому клиенту, а затем отсортировать клиентов по выручке.

Данные

```
  CREATE TABLE orders(id INTEGER, client_id
INTEGER, total INTEGER);
  INSERT INTO orders VALUES (1,1,300),
(2,1,500),(3,2,900),(4,3,100);
```

Запрос

```
  WITH client_revenue AS (
  SELECT client_id, SUM(total) AS revenue
  FROM orders
  GROUP BY client_id
  )
  SELECT client_id, revenue
```

```
FROM client_revenue
ORDER BY revenue DESC, client_id;
```

Ожидаемый результат

```
[[2, 900], [1, 800], [3, 100]]
```

Почему работает

CTE делает одну строку на client_id.

Финальный SELECT уже не думает о деталях заказов.

Сортировка показывает клиентов от самого денежного к слабому.

Где легко ошибиться

Если позже присоединять справочник клиентов, сначала проверьте, что client_revenue действительно имеет одну строку на клиента.

Самостоятельная проверка

Добавьте клиенту 3 заказ на 850. Как изменится порядок строк?

Ответ: Клиент 3 станет первым с выручкой 950, затем кли-

ент 2 и клиент 1.

Задача 3. Отфильтровать итоги после группировки

Рабочий вопрос

Менеджеру нужны только товары, которые дали выручку от 500, а не все позиции каталога.

Данные

```
  CREATE TABLE sales(sku TEXT, revenue
INTEGER);
  INSERT INTO sales VALUES ('A1',200),
('A1',350),('B2',100),('C3',700);
```

Запрос

```
  WITH sku_revenue AS (
  SELECT sku, SUM(revenue) AS revenue
  FROM sales
  GROUP BY sku
  )
  SELECT sku, revenue
```

```
FROM sku_revenue
WHERE revenue >= 500
ORDER BY revenue DESC;
```

Ожидаемый результат

```
[["C3", 700], ["A1", 550]]
```

Почему работает

Внутри CTE суммы собираются по SKU.

Финальный WHERE фильтрует уже агрегированную выручку.

B2 не проходит, потому что его итог всего 100.

Где легко ошибиться

WHERE revenue >= 500 до группировки проверял бы отдельные строки продаж, а не итог товара.

Самостоятельная проверка

Почему A1 проходит фильтр, хотя ни одна отдельная строка A1 не равна 500?

Ответ: Потому что CTE сначала складывает 200 и 350, а

фильтр видит итог 550.

Конец ознакомительного фрагмента.