Глеб Зайцев

Подзапросы SQL. 18 задач для аналитика

Глеб Зайцев

Подзапросы SQL. 18 задач для аналитика

«Автор»
2026

Зайцев Г.

Подзапросы SQL. 18 задач для аналитика / Г. Зайцев — «Автор», 2026

Практическая книга по SQL-подзапросам для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: IN, EXISTS, NOT EXISTS, скалярные подзапросы, подзапросы в SELECT и FROM, средние значения, последние события, анти-join и контроль дублей. Каждый пример запускается в SQLite, данные встроены прямо в книгу.

Содержание

Глеб Зайцев
Подзапросы SQL. 18 задач для аналитика

Перед первой задачей

Зачем нужны подзапросы

Подзапрос помогает сначала выразить маленький вопрос, а потом использовать его как фильтр, значение или временную таблицу.

Это особенно удобно в аналитике: найти клиентов с покупками, товары без продаж, заказы выше среднего, последние события и подозрительные несостыковки.

В этой книге нет абстрактной теории ради теории. Каждая задача начинается с рабочей ситуации и маленьких данных, где результат можно проверить глазами.

Как не сломать отчёт

Главная ловушка подзапросов — не синтаксис, а уровень детализации. Нужно понимать, возвращает подзапрос одно значение, список значений или таблицу.

Второй важный вопрос: подзапрос независимый или коррелированный. Коррелированный подзапрос выполняет проверку для строки внешнего запроса.

Все примеры рассчитаны на SQLite. Мы используем базовые конструкции SQL, поэтому смысл легко перенести в PostgreSQL, MySQL и другие СУБД.

Маршрут по задачам

Практические задачи

Задача 1. Найти клиентов с оплаченными заказами через IN

Рабочий вопрос

Маркетологу нужен список клиентов, у которых есть хотя бы один успешный заказ.

Данные

```
    CREATE TABLE clients(id INTEGER, name TEXT);
    CREATE TABLE orders(id INTEGER, client_id INTEGER, status
TEXT);
    INSERT INTO clients VALUES (1,'Анна'),(2,'Илья'),(3,'Олег'),
(4,'Нина');
    INSERT INTO orders VALUES (101,1,'paid'),(102,1,'cancelled'),
(103,3,'paid');
```

Запрос

```
    SELECT id, name
    FROM clients
    WHERE id IN (
    SELECT client_id
    FROM orders
    WHERE status = 'paid'
    )
    ORDER BY id;
```

Ожидаемый результат

```
    [[1, "Анна"], [3, "Олег"]]
```

Почему работает

Внутренний запрос возвращает client_id только из оплаченных заказов.

Внешний запрос оставляет клиентов, чей id входит в этот список.

У Анны есть оплаченный заказ, хотя рядом есть и отменённый: фильтр смотрит на наличие хотя бы одного paid.

Где легко ошибиться

Не нужно соединять таблицы, если нужен только список клиентов по факту наличия заказа: IN делает проверку короче и понятнее.

Самостоятельная проверка

Добавьте заказ 104 для Нины со статусом paid. Какие клиенты будут в результате?

Ответ: В результате будут Анна, Олег и Нина, потому что id Нины появится во внутреннем списке.

Задача 2. Оставить заказы выше среднего чека

Рабочий вопрос

Нужно быстро увидеть заказы, сумма которых выше среднего значения по всем заказам.

Данные

```
CREATE TABLE orders(id INTEGER, total INTEGER);
INSERT INTO orders VALUES (1,300),(2,900),(3,600),(4,200);
```

Запрос

```
SELECT id, total
FROM orders
WHERE total > (
SELECT AVG(total)
FROM orders
)
ORDER BY total DESC;
```

Ожидаемый результат

```
[[2, 900], [3, 600]]
```

Почему работает

Скалярный подзапрос возвращает одно число: средний чек.
Среднее равно 500, поэтому остаются только суммы больше 500.
Внешний запрос сортирует уже отфильтрованные заказы по убыванию суммы.

Где легко ошибиться

Такой подзапрос обязан вернуть одно значение. Если случайно вернуть несколько строк, сравнение total > (...) потеряет смысл.

Самостоятельная проверка

Что изменится, если заказ 4 увеличить с 200 до 800?
Ответ: Среднее станет 650, поэтому останутся заказы 2 и 4; заказ 3 на 600 уже не выше среднего.

Задача 3. Найти товары без продаж через NOT EXISTS

Рабочий вопрос

Каталог содержит товары, но не все продавались. Нужно найти товары без единой строки продажи.

Данные

```
CREATE TABLE products(sku TEXT, name TEXT);
CREATE TABLE sales(id INTEGER, sku TEXT, qty INTEGER);
INSERT INTO products VALUES ('A1','Планер'),('B2','Кур-
с'),('C3','Шаблон'),('D4','Чек-лист');
INSERT INTO sales VALUES (1,'A1',2),(2,'C3',1),(3,'A1',1);
```

Запрос

```
SELECT p.sku, p.name
FROM products p
WHERE NOT EXISTS (
SELECT 1
FROM sales s
WHERE s.sku = p.sku
)
ORDER BY p.sku;
```

Ожидаемый результат

```
[["B2", "Курс"], ["D4", "Чек-лист"]]
```

Почему работает

Конец ознакомительного фрагмента.