ГЛЕБ ЗАЙЦЕВ

ЗАКАЗЫ В SQL

18 задач для контроля продаж

```
SELECT status,
       COUNT(*) AS orders
FROM orders
GROUP BY status
ORDER BY status;
```

оплаты · отмены · возвраты

дубли · доставка · клиенты

все запросы запускаются в SQLite

Глеб Зайцев

Заказы в SQL. 18 задач для контроля продаж

«Автор»
2026

Зайцев Г.

Заказы в SQL. 18 задач для контроля продаж / Г. Зайцев — «Автор», 2026

Практическая книга по контролю заказов в SQL. Внутри 18 задач на маленьких таблицах: оплаченные заказы, статусы, выручка, отмены, дубли, платежи, доставка, возвраты, повторные покупатели, лаги и короткий отчёт продаж. Все запросы запускаются в SQLite.

Содержание

Глеб Зайцев
Заказы в SQL. 18 задач для контроля продаж

Перед первой задачей

Почему заказы надо считать аккуратно

Заказ может быть создан, оплачен, отменён или доставлен, и каждая строка не всегда означает деньги.

Эта книга тренирует короткие SQL-запросы для контроля продаж: где оплата, где дубль, где возврат, где задержка выдачи.

Примеры маленькие, но вопросы рабочие: их удобно переносить в реальные выгрузки маркетплейса, CRM или магазина.

Как читать задачи

Сначала проверьте единицу анализа: заказ, клиент, товарная строка, день или канал.

Затем смотрите, какие статусы попадают в отчёт, а какие должны быть исключены.

Все запросы исполняются в SQLite и имеют ожидаемый результат, поэтому ответ можно перепроверить без внешних файлов.

Маршрут по задачам

Практические задачи

Задача 1. Посчитать оплаченные заказы

Рабочий вопрос

Нужно быстро отделить реальные оплаченные заказы от созданных, отменённых и тестовых строк.

Данные

```
    CREATE TABLE orders(order_id INTEGER, status TEXT, amount INTEGER);
    INSERT INTO orders VALUES (1,'paid',900),(2,'created',500),(3,'paid',700),(4,'cancelled',400);
```

Запрос

```
    SELECT COUNT(*) AS paid_orders
    FROM orders
    WHERE status = 'paid';
```

Ожидаемый результат

```
    [[2]]
```

Почему работает

WHERE оставляет только строки со статусом paid, а COUNT считает их количество.

Запрос сделан на маленькой синтетической таблице, поэтому ответ можно проверить глазами и повторить в SQLite.

Явный ORDER BY или отдельная агрегация фиксируют смысл результата, а не оставляют его на случайный порядок строк.

Где легко ошибиться

COUNT(*) по всей таблице показал бы четыре строки, но две из них ещё не являются выручкой.

Самостоятельная проверка

Сколько оплаченных заказов в данных? Объясните ответ по исходным строкам и условию запроса.

Ответ: Два заказа: 1 и 3. Это следует из исходных строк и условия запроса.

Задача 2. Сложить дневную выручку

Рабочий вопрос

Нужно получить выручку по дням, но учитывать только оплаченные заказы, а не все созданные корзины.

Данные

```
    CREATE TABLE orders(order_id INTEGER, paid_at TEXT, status
TEXT, amount INTEGER);
    INSERT INTO orders VALUES
    (1,'2026-09-01','paid',900),(2,'2026-09-01','cancelled',500),
    (3,'2026-09-02','paid',700),(4,'2026-09-02','paid',300);
```

Запрос

```
SELECT paid_at, SUM(amount) AS revenue
FROM orders
WHERE status = 'paid'
GROUP BY paid_at
ORDER BY paid_at;
```

Ожидаемый результат

```
[["2026-09-01", 900], ["2026-09-02", 1000]]
```

Почему работает

Фильтр по paid выполняется до группировки, поэтому отменённая сумма не попадает в дневную выручку.

Запрос сделан на маленькой синтетической таблице, поэтому ответ можно проверить глазами и повторить в SQLite.

Явный ORDER BY или отдельная агрегация фиксируют смысл результата, а не оставляют его на случайный порядок строк.

Где легко ошибиться

Если группировать все статусы вместе, отчёт завысит деньги на день с отменой.

Самостоятельная проверка

Какая выручка 2026-09-02? Объясните ответ по исходным строкам и условию запроса.

Ответ: 1000: 700 плюс 300. Это следует из исходных строк и условия запроса.

Задача 3. Разложить заказы по статусам

Рабочий вопрос

Нужно увидеть, где застревает поток заказов: создано, оплачено, доставлено или отменено.

Данные

```
CREATE TABLE orders(order_id INTEGER, status TEXT);
INSERT INTO orders VALUES (1,'created'),(2,'paid'),(3,'paid'),
(4,'delivered'),(5,'cancelled');
```

Запрос

```
SELECT status, COUNT(*) AS orders_count
FROM orders
GROUP BY status
ORDER BY status;
```

Ожидаемый результат

```
[["cancelled", 1], ["created", 1], ["delivered", 1], ["paid",
2]]
```

Почему работает

GROUP BY status превращает журнал заказов в компактный счётчик по состояниям.

Запрос сделан на маленькой синтетической таблице, поэтому ответ можно проверить глазами и повторить в SQLite.

Явный ORDER BY или отдельная агрегация фиксируют смысл результата, а не оставляют его на случайный порядок строк.

Где легко ошибиться

Без явного словаря статусов легко сравнивать разные состояния как будто они одинаково ценны.

Конец ознакомительного фрагмента.