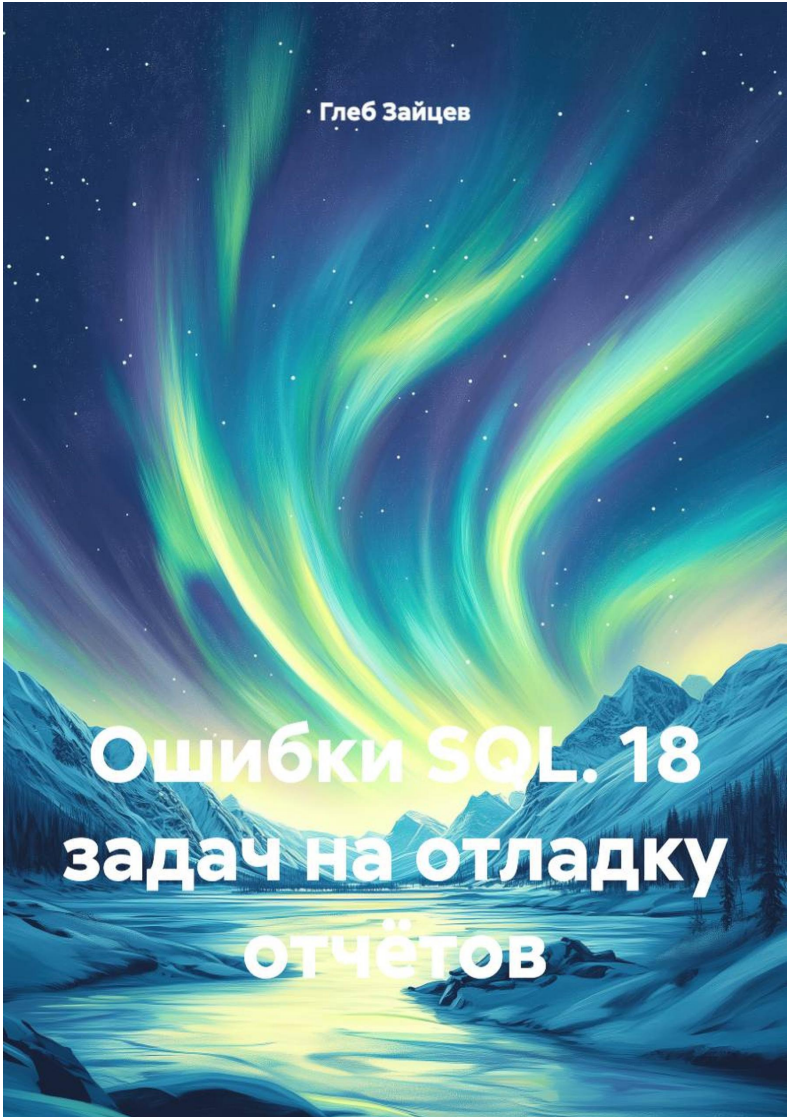
Глеб Зайцев
Ошибки SQL. 18 задач на отладку отчётов

Глеб Зайцев

Ошибки SQL. 18 задач на отладку отчётов

https://litres.ru/74419354
SelfPub; 2026

Аннотация

Практическая книга о типовых ошибках SQL-отчётов для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: потерянный фильтр, дубли после JOIN, LEFT JOIN с условием в WHERE, COUNT и NULL, границы дат, HAVING, UNION, NOT IN, средние значения, сортировка, пустые категории и финальный аудит отчёта. Все примеры запускаются в SQLite, данные встроены прямо в книгу.

Содержание

Глеб Зайцев
Ошибки SQL. 18 задач на отладку отчётов

Перед первой задачей

Почему отчёт может выглядеть правильным и всё равно врать

SQL редко падает с ошибкой, когда бизнес-логика выбрана неверно. Он спокойно вернёт число, которое выглядит аккуратно и попадает в таблицу.

Самые опасные ошибки не синтаксические: лишний JOIN размножил строки, WHERE убрал пустые связи, дата потеряла вечерние заказы, COUNT пропустил NULL.

Эта книга тренирует привычку отлаживать отчёт маленькими контрольными таблицами, а не верить первому красивому результату.

Как работать с задачами

В каждой задаче есть короткий рабочий вопрос и правильный запрос. Перед чтением объяснения полезно назвать, какая ошибка могла бы появиться в наивном варианте.

Смотрите не только на итоговое число, но и на уровень строки: заказ, товарная позиция, клиент, день или категория.

Все примеры запускаются в SQLite. В других СУБД функции дат и строк могут отличаться, но сами ловушки отчётов остаются теми же.

Маршрут по задачам

Практические задачи

Задача 1. Не забыть фильтр оплаченных заказов

Рабочий вопрос

Нужно посчитать только оплаченную выручку и не дать новым или отменённым заказам попасть в финансовый итог.

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

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

Ожидаемый результат

```
[[800]]
```

Почему работает

WHERE оставляет только оплаченные строки.
Оплаченные суммы равны 500 и 300.
new и cancelled не входят в денежный итог.

Где легко ошибиться

SUM(total) без фильтра вернул бы 2400 и выглядел бы как оборот, хотя это не оплаченная выручка.

Самостоятельная проверка

Какая строка сильнее всего искажает результат, если за-

быть фильтр status = paid?

Ответ: Отменённый заказ на 900 добавляет деньги, которых не должно быть в оплаченной выручке.

Задача 2. Не размножить заказы после JOIN

Рабочий вопрос

Нужно посчитать число оплаченных заказов после соединения с товарными строками, не считая позиции как заказы.

Данные

```
  CREATE TABLE orders(id INTEGER, status
TEXT);
  CREATE TABLE items(order_id INTEGER, sku
TEXT);
  INSERT INTO orders VALUES (1,'paid'),
(2,'paid'),(3,'new');
  INSERT INTO items VALUES (1,'A'),(1,'B'),
(2,'C'),(3,'D');
```

Запрос

```
  SELECT COUNT(DISTINCT o.id) AS
paid_orders
```

```
FROM orders o
JOIN items i ON i.order_id = o.id
WHERE o.status = 'paid';
```

Ожидаемый результат

```
[[2]]
```

Почему работает

Заказ 1 после JOIN даёт две строки из-за двух товаров.
COUNT(DISTINCT o.id) считает заказ 1 один раз.
Фильтр paid исключает заказ 3.

Где легко ошибиться

COUNT(*) после JOIN вернул бы 3, потому что считал бы товарные строки оплаченных заказов.

Самостоятельная проверка

Почему COUNT(*) даёт неверный ответ именно после соединения с items?

Ответ: Один заказ может иметь несколько товарных строк, и JOIN размножает его по позициям.

Задача 3. Не сломать LEFT JOIN фильтром WHERE

Рабочий вопрос

Нужно показать все категории и выручку только оплаченных продаж, включая категории без оплат.

Данные

```
  CREATE TABLE categories(name TEXT);
  CREATE TABLE sales(category TEXT, status
TEXT, total INTEGER);
  INSERT INTO categories VALUES ('books'),
('courses'),('templates');
  INSERT      INTO      sales      VALUES
('books','paid',300),('books','new',500),
('courses','paid',700);
```

Конец ознакомительного фрагмента.