

Глеб Зайцев

ORDER BY и LIMIT в SQL. 18 задач на сортировку

Глеб Зайцев

ORDER BY и LIMIT в SQL.
18 задач на сортировку

«Автор»

2026

Зайцев Г.

**ORDER BY и LIMIT в SQL. 18 задач на сортировку / Г. Зайцев —
«Автор», 2026**

Практическая книга по ORDER BY и LIMIT в SQL для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: топ заказов, последние события, вторичная сортировка, страницы через OFFSET, сортировка по выражению, приоритеты через CASE, NULL в конце, топ внутри категории и финальный отчёт по марже. Все примеры запускаются в SQLite.

© Зайцев Г., 2026

© Автор, 2026

Содержание

Перед первой задачей	5
Почему сортировка — это не косметика	5
Как проверять себя	6
Маршрут по задачам	7
Практические задачи	8
Задача 1. Показать заказы от самого крупного	8
Задача 2. Добавить вторичную сортировку	9
Задача 3. Взять топ-3 товара по выручке	10
Конец ознакомительного фрагмента.	11

Глеб Зайцев

ORDER BY и LIMIT в SQL.

18 задач на сортировку

Перед первой задачей

Почему сортировка — это не косметика

ORDER BY часто воспринимают как украшение отчёта, но на самом деле он меняет то, какие строки попадут в топ, какую страницу увидит менеджер и какой заказ будет считаться последним.

LIMIT без понятной сортировки почти всегда опасен: база может вернуть любые первые строки, и отчёт будет выглядеть стабильным только случайно.

В этой книге каждая задача показывает маленькую рабочую ситуацию, где порядок строк влияет на решение: кого проверить, что продвигать, какой товар считать лидером.

Как проверять себя

Перед запуском запроса выпишите вручную ключ сортировки для каждой строки: сумма, дата, приоритет, вычисленная маржа или номер страницы.

Если у двух строк одинаковый главный ключ, проверьте вторичный ORDER BY. Именно он делает результат воспроизводимым, а не зависящим от внутреннего порядка базы.

Все запросы рассчитаны на SQLite. Приёмы ORDER BY, LIMIT, OFFSET, CASE и сортировки выражений полезны почти в любой SQL-системе.

Маршрут по задачам

- [1. Показать заказы от самого крупного](#)
- [2. Добавить вторичную сортировку](#)
- [3. Взять топ-3 товара по выручке](#)
- [4. Получить вторую страницу каталога](#)
- [5. Сортировать по сумме строки заказа](#)
- [6. Сортировать статусы по приоритету](#)
- [7. Поставить NULL в конец списка](#)
- [8. Выбрать самый дорогой товар в категории](#)
- [9. Показать последние события](#)
- [10. Сортировать категории по суммарной выручке](#)
- [11. Сортировать текст без учёта регистра](#)
- [12. Сортировать числовые строки как числа](#)
- [13. Найти товары с самым низким остатком](#)
- [14. Взять топ после фильтра](#)
- [15. Стабилизировать лидерборд](#)
- [16. Сортировать по алиасу агрегата](#)
- [17. Выбрать нижний хвост продаж](#)
- [18. Собрать финальный топ по марже](#)

Практические задачи

Задача 1. Показать заказы от самого крупного

Рабочий вопрос

Нужно вывести заказы так, чтобы менеджер сначала увидел самые дорогие покупки.

Данные

```
CREATE TABLE orders(id INTEGER, total INTEGER);  
INSERT INTO orders VALUES (1,300), (2,900), (3,500), (4,1200);
```

Запрос

```
SELECT id, total  
FROM orders  
ORDER BY total DESC;
```

Ожидаемый результат

```
[[4, 1200], [2, 900], [3, 500], [1, 300]]
```

Почему работает

ORDER BY total DESC сортирует суммы от большей к меньшей.
Заказ 4 самый крупный, поэтому он идёт первым.
Заказ 1 самый маленький и оказывается внизу.

Где легко ошибиться

Без DESC сортировка пошла бы по возрастанию, и отчёт начал бы показывать самые маленькие заказы.

Самостоятельная проверка

Добавьте заказ на 700. Между какими строками он окажется?
Ответ: Он окажется между 900 и 500, потому что сортировка идёт по убыванию total.

Задача 2. Добавить вторичную сортировку

Рабочий вопрос

В отчёте есть одинаковые суммы, и нужно получить стабильный порядок строк внутри ничьей.

Данные

```
CREATE TABLE orders(id INTEGER, total INTEGER, order_date
TEXT);
INSERT INTO orders VALUES
(1, 500, '2026-09-03'),
(2, 900, '2026-09-01'),
(3, 500, '2026-09-01'),
(4, 900, '2026-09-02');
```

Запрос

```
SELECT id, total, order_date
FROM orders
ORDER BY total DESC, order_date ASC, id ASC;
```

Ожидаемый результат

```
[[2, 900, "2026-09-01"], [4, 900, "2026-09-02"], [3, 500,
"2026-09-01"], [1, 500, "2026-09-03"]]
```

Почему работает

Сначала строки делятся по total от большего к меньшему.
При равной сумме раньше идёт более ранняя дата.
id добавлен как финальная страховка от случайного порядка.

Где легко ошибиться

Если оставить только ORDER BY total DESC, строки с одинаковой суммой могут возвращаться в неочевидном порядке.

Самостоятельная проверка

Почему заказ 2 идёт выше заказа 4, хотя суммы у них одинаковые?

Ответ: Потому что вторичная сортировка order_date ASC ставит 2026-09-01 раньше 2026-09-02.

Задача 3. Взять топ-3 товара по выручке

Рабочий вопрос

Нужно показать три товара с наибольшей выручкой, чтобы быстро выбрать лидеров каталога.

Данные

```
CREATE TABLE products(sku TEXT, revenue INTEGER);
INSERT INTO products VALUES ('A1',300), ('B2',900), ('C3',500),
('D4',1200), ('E5',100);
```

Запрос

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.