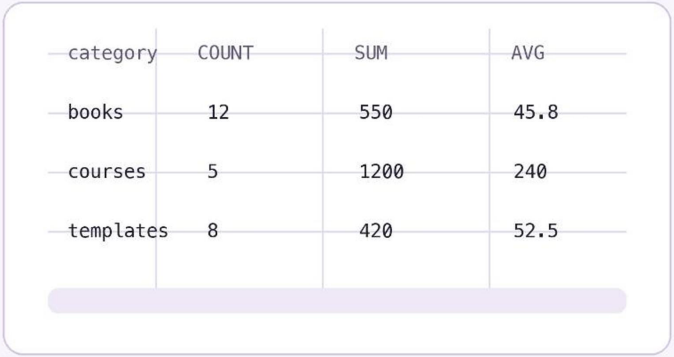
ГЛЕБ ЗАЙЦЕВ

GROUP BY В SQL

18 задач для отчётов



category	COUNT	SUM	AVG
books	12	550	45.8
courses	5	1200	240
templates	8	420	52.5

HAVING · COUNT DISTINCT · CASE · доли

месяцы · дубли · итоговые строки

все запросы запускаются в SQLite

Глеб Зайцев

GROUP BY в SQL. 18 задач для отчётов и аналитики

https://litres.ru/74418883
SelfPub; 2026

Аннотация

Практическая книга по GROUP BY, HAVING и агрегатам SQL для тех, кто хочет делать отчёты без угадывания. Внутри 18 задач на маленьких таблицах: выручка по категориям, средний чек, месяцы, COUNT DISTINCT, поиск дублей, сегменты через CASE, объединение со справочником, доли от итога, взвешенное среднее и итоговый отчёт. Все примеры запускаются в SQLite, данные встроены прямо в книгу.

Содержание

Глеб Зайцев
GROUP BY в SQL. 18 задач для отчётов и аналитики

Перед первым отчётом

Зачем отдельная книга про группировки

GROUP BY выглядит простым, пока отчёт не начинает отвечать на неправильный вопрос. Одна лишняя колонка, забытый фильтр или неверный уровень группировки меняют смысл результата.

Эта книга идёт от базовых агрегатов к отчётным конструкциям: HAVING, CASE, COUNT DISTINCT, подзапросы, CTE и объединение со справочниками.

Мы используем маленькие искусственные таблицы магазина, чтобы каждая задача проверялась глазами и запускалась локально без внешней базы.

Как работать с примерами

Перед запуском запроса назовите единицу результата: строка на статус, категория, месяц, клиент, пара месяц-канал или весь отчёт. Это защищает от случайной группировки.

После результата проверьте не только числа, но и количество строк. В отчётах ошибка часто прячется именно там: строк стало больше или меньше, чем ожидалось.

Все запросы написаны под SQLite. В PostgreSQL, MySQL или ClickHouse могут отличаться функции дат, но сама логика группировки останется той же.

Маршрут по задачам

Восемнадцать задач

Задача 1. Посчитать заказы по статусам

Рабочий вопрос

Нужно понять, сколько заказов находится в каждом статусе, чтобы увидеть баланс между новыми, оплаченными и отменёнными строками.

Данные

```
  CREATE TABLE orders(id INTEGER, status
TEXT);
  INSERT INTO orders VALUES
  (1,'paid'),(2,'new'),(3,'paid'),
(4,'cancelled'),(5,'new'),(6,'paid');
```

Запрос

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

Ожидаемый результат

```
[["cancelled", 1], ["new", 2], ["paid",
3]]
```

Почему работает

GROUP BY status собирает строки с одинаковым статусом в одну группу.

COUNT(*) считает все строки внутри каждой группы.

ORDER BY нужен не для расчёта, а для стабильного порядка отчёта.

Где легко ошибиться

Если убрать GROUP BY, COUNT(*) вернёт одну строку по всей таблице, а не разрез по статусам.

Самостоятельная проверка

Добавьте ещё один paid-заказ и проверьте, что счётчик paid станет равен 4.

Ответ: После добавления новой строки paid результат будет cancelled=1, new=2, paid=4.

Задача 2. Сложить выручку по категориям

Рабочий вопрос

Есть продажи разных категорий. Нужно узнать, какие категории принесли больше денег за период.

Данные

```
  CREATE TABLE sales(id INTEGER, category
TEXT, revenue INTEGER);
  INSERT INTO sales VALUES
  (1,'books',300),(2,'courses',900),
(3,'books',250),(4,'templates',420),
(5,'courses',300);
```

Запрос

```
  SELECT category, SUM(revenue) AS revenue
  FROM sales
  GROUP BY category
  ORDER BY revenue DESC, category;
```

Ожидаемый результат

```
[["courses", 1200], ["books", 550],
["templates", 420]]
```

Почему работает

SUM складывает revenue только внутри текущей категории.

Сортировка по revenue DESC показывает самые денежные категории сверху.

Дополнительная сортировка по category делает порядок предсказуемым при равенстве сумм.

Где легко ошибиться

Нельзя читать этот отчёт как прибыль: здесь нет комиссий, налогов и себестоимости.

Самостоятельная проверка

Добавьте продажу templates на 900. Какая категория выйдет первой?

Ответ: templates станет первой с 1320, потому что 420 +

900 больше 1200.

Задача 3. Найти средний чек клиента

Рабочий вопрос

Менеджер хочет сравнить клиентов по среднему заказу, а не по общей сумме покупок.

Данные

```
  CREATE TABLE orders(client TEXT, total
INTEGER);
  INSERT INTO orders VALUES
  ('Анна',1000),('Ан-
на',500),('Илья',900),('Олег',300),
('Олег',900);
```

Запрос

```
  SELECT client, ROUND(AVG(total), 2) AS
avg_total
  FROM orders
  GROUP BY client
  ORDER BY avg_total DESC, client;
```

Ожидаемый результат

```
[["Илья", 900.0], ["Анна", 750.0], ["Олег", 600.0]]
```

Почему работает

AVG считает среднее значение total внутри группы клиента.

Конец ознакомительного фрагмента.