ГЛЕБ ЗАЙЦЕВ

ROW_NUMBER В SQL

18 задач по ранжированию

```
ROW_NUMBER() OVER (
  PARTITION BY client_id
  ORDER BY paid_at DESC
) AS rn
```

последняя строка · топ внутри группы

дубли · статусы · очереди

все запросы запускаются в SQLite

Глеб Зайцев

ROW_NUMBER в SQL. 18 задач по ранжированию

«Автор»
2026

Зайцев Г.

ROW_NUMBER в SQL. 18 задач по ранжированию / Г. Зайцев — «Автор», 2026

Практическая книга по ROW_NUMBER и ранжированию в SQL для аналитиков. Внутри 18 задач на маленьких таблицах: последние заказы, топ товаров в категории, дубли, статусы, первая покупка, очередь заявок, лучшая цена и стабильная сортировка. Все примеры запускаются в SQLite, данные встроены в книгу.

Содержание

Глеб Зайцев
ROW_NUMBER в SQL. 18 задач по ранжированию

Перед первой задачей

Зачем аналитику ROW_NUMBER

ROW_NUMBER помогает выбрать первую, последнюю, вторую или лучшую строку внутри группы без ручного перебора.

Самые частые задачи: последний статус заказа, топ товаров по категории, первая покупка клиента и дубли после первой строки.

Ключевой вопрос перед запросом: внутри какой группы нумеруем строки и по какому правилу сортируем.

Что проверять в каждом примере

PARTITION BY отвечает за группу: клиент, заказ, товар, день или категория.

ORDER BY внутри OVER отвечает за смысл номера: новый к старому, дорогой к дешёвому, высокий приоритет к низкому.

Если у строк возможна ничья, добавляйте дополнительное правило сортировки, иначе номер может быть техническим, а не бизнесовым.

Маршрут по задачам

Практические задачи

Задача 1. Пронумеровать заказы клиента от новых к старым

Рабочий вопрос

Нужно для каждого клиента поставить номер заказа по убыванию даты, чтобы найти самый свежий заказ без ручной сортировки.

Данные

```
    CREATE TABLE orders(client_id INTEGER, order_id INTEGER,
order_date TEXT);
    INSERT INTO orders VALUES
    (10,1,'2026-09-01'),(10,2,'2026-09-03'),(20,3,'2026-09-02'),
(20,4,'2026-09-05');
```

Запрос

```
    SELECT client_id, order_id, ROW_NUMBER() OVER (
    PARTITION BY client_id ORDER BY order_date DESC
    ) AS rn
    FROM orders
    ORDER BY client_id, rn;
```

Ожидаемый результат

```
    [[10, 2, 1], [10, 1, 2], [20, 4, 1], [20, 3, 2]]
```

Почему работает

ROW_NUMBER нумерует строки внутри каждого клиента, а PARTITION BY начинает счёт заново для следующего клиента.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

ORDER BY или явная агрегация делают вывод устойчивым: строки не зависят от случайного порядка хранения данных.

Где легко ошибиться

Если забыть PARTITION BY, получится общий рейтинг всех заказов, а не отдельная нумерация по клиентам.

Самостоятельная проверка

Какой заказ клиента 10 получит номер 1? Объясните ответ, опираясь на строки исходных данных.

Ответ: Заказ 2, потому что дата 2026-09-03 новее 2026-09-01.

Задача 2. Выбрать последний заказ каждого клиента

Рабочий вопрос

Нужно оставить по одному последнему заказу на клиента, чтобы построить витрину актуального состояния.

Данные

```
    CREATE TABLE orders(client_id INTEGER, order_id INTEGER,
order_date TEXT);
    INSERT INTO orders VALUES
    (10,1,'2026-09-01'),(10,2,'2026-09-03'),(20,3,'2026-09-02'),
(20,4,'2026-09-05');
```

Запрос

```
    WITH ranked AS (
    SELECT client_id, order_id, order_date,
    ROW_NUMBER() OVER (PARTITION BY client_id ORDER BY order_date
DESC) AS rn
    FROM orders
    )
    SELECT client_id, order_id, order_date
    FROM ranked
    WHERE rn = 1
    ORDER BY client_id;
```

Ожидаемый результат

```
    [[10, 2, "2026-09-03"], [20, 4, "2026-09-05"]]
```

Почему работает

CTE сначала ставит номер, а внешний SELECT оставляет только rn = 1 для каждого клиента.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

ORDER BY или явная агрегация делают вывод устойчивым: строки не зависят от случайного порядка хранения данных.

Где легко ошибиться

Фильтровать rn в том же SELECT через WHERE нельзя: алиас оконной функции появляется после расчёта строк.

Самостоятельная проверка

Почему нужен внешний запрос или CTE? Объясните ответ, опираясь на строки исходных данных.

Ответ: Потому что сначала нужно посчитать rn, а уже потом фильтровать по нему.

Задача 3. Ранжировать товары внутри категории

Рабочий вопрос

Нужно увидеть место товара по выручке внутри каждой категории, чтобы сравнивать не все товары сразу, а похожие товары.

Данные

```
    CREATE TABLE sales(category TEXT, sku TEXT, revenue INTEGER);
    INSERT INTO sales VALUES
    ('books','A',500),('books','B',700),('books','C',300),
('apps','D',900),('apps','E',400);
```

Запрос

```
    SELECT category, sku, revenue,
    ROW_NUMBER() OVER (PARTITION BY category ORDER BY revenue DESC)
AS place
    FROM sales
    ORDER BY category, place;
```

Ожидаемый результат

```
    [["apps", "D", 900, 1], ["apps", "E", 400, 2], ["books", "B",
700, 1], ["books", "A", 500, 2], ["books", "C", 300, 3]]
```

Почему работает

PARTITION BY category делит рейтинг по категориям, а ORDER BY revenue DESC ставит большие продажи выше.

Запрос намеренно работает на маленькой таблице, поэтому результат можно проверить глазами до переноса приёма в рабочий отчёт.

Конец ознакомительного фрагмента.