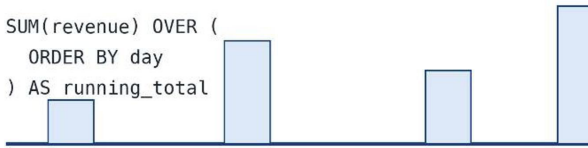
ГЛЕБ ЗАЙЦЕВ

ОКОННЫЕ ФУНКЦИИ SQL

18 задач для аналитика

```
ROW_NUMBER() OVER (
  PARTITION BY client
  ORDER BY order_date DESC
) AS rn

SUM(revenue) OVER (
  ORDER BY day
) AS running_total
```



rank · lag · lead · фреймы · когорты

все запросы запускаются в SQLite

практикум без внешних баз и аккаунтов

Глеб Зайцев

Оконные функции SQL. 18 задач для аналитика

https://litres.ru/74418588
SelfPub; 2026

Аннотация

Практическая книга по оконным функциям SQL для аналитиков, которые уже знают SELECT, JOIN и GROUP BY и хотят уверенно применять ROW_NUMBER, RANK, DENSE_RANK, LAG, LEAD, накопительные суммы, скользящие средние, фреймы, когорты и воронки. Внутри 18 самостоятельных задач на маленьких таблицах: последний заказ клиента, топы внутри категории, доля в месяце, дедупликация, серии дней, ретеншен и итоговый отчёт. Все данные встроены в книгу, каждый запрос можно запустить локально в SQLite.

Содержание

Глеб Зайцев
Оконные функции SQL. 18 задач для аналитика

Перед первым окном

Зачем отдельная книга по окнам

Оконная функция отвечает на вопрос внутри строки, но смотрит на соседние строки той же таблицы. Это похоже на GROUP BY, но результат не схлопывается до одной строки на группу. Поэтому окна особенно полезны в аналитике: нужно показать заказ и одновременно знать место заказа в истории клиента, долю товара в месяце или накопительный итог.

Главная ошибка новичка — ожидать от окна поведения обычной группировки. GROUP BY уменьшает число строк. SUM(amount) OVER (...) оставляет строки на месте и добавляет к каждой контекст. В отчётах именно это часто и требуется: не потерять детализацию, но видеть сравнение с группой.

В книге используется SQLite. Синтаксис выбран так, что-

бы запросы можно было прогнать без сервера и аккаунтов. На PostgreSQL, MySQL 8 и других СУБД логика почти та же, но функции дат и округление могут отличаться.

Как читать практикум

Каждая задача начинается с рабочего вопроса: что должен узнать аналитик. Затем идёт схема и данные, потом один запрос, ожидаемый результат, объяснение и типичная ошибка. Такой порядок защищает от красивого SQL, который отвечает не на бизнес-вопрос.

Если запрос у вас дал другой порядок строк, сначала проверьте ORDER BY. У окон есть свой ORDER BY внутри OVER, но он не обязан задавать порядок итоговой выдачи. В книге итоговый ORDER BY всегда прописан отдельно.

Не переносите примеры на реальные клиентские данные без обезличивания. Для тренировки достаточно синтетических строк; цель здесь — понять механику окон, а не гонять чужие выгрузки в неизвестной среде.

Маршрут по задачам

Восемнадцать задач

Задача 1. ROW_NUMBER: последний заказ каждого клиента

Рабочий вопрос

Нужно оставить по одному последнему заказу на клиента, но сохранить id заказа и сумму, а не только максимальную дату.

Данные

```
  CREATE TABLE orders(client TEXT, order_id
TEXT, order_date TEXT, amount INTEGER);
  INSERT INTO orders VALUES
  ('alice','A1','2026-09-01',1200),
  ('alice','A2','2026-09-05',700),
  ('bob','B1','2026-09-02',400),
  ('bob','B2','2026-09-02',900),
  ('bob','B3','2026-09-06',300);
```

Запрос

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

Ожидаемый результат

```
[
[
"alice",
```

```
"A2",
700
],
[
"bob",
"B3",
300
]
]
```

Почему это работает

PARTITION BY client запускает нумерацию заново для каждого клиента. Поэтому у alice и bob будет свой номер один.

ORDER BY внутри окна задаёт, что считать последним заказом. В примере дата сортируется по убыванию, а order_id добавлен как стабильный второй ключ.

Обычный MAX(order_date) не вернул бы сумму и id заказа без дополнительного соединения. ROW_NUMBER позволяет выбрать целую строку.

Типичная ошибка

Если у двух заказов одна дата, без второго ключа порядок

может стать неустойчивым. Для отчёта это опасно: сегодня выберется B1, завтра B2.

Самостоятельная проверка

Добавьте клиенту alice заказ A3 на ту же дату 2026-09-05 с суммой 900. Какой заказ останется при текущем втором ключе?

Вернуться к маршруту

Задача 2. RANK и DENSE_RANK: места с одинаковой выручкой

Рабочий вопрос

Нужно показать места товаров внутри категории и увидеть разницу между спортивным рангом с пропусками и плотным рангом без пропусков.

Данные

```
  CREATE TABLE sku_sales(category TEXT, sku
TEXT, revenue INTEGER);
  INSERT INTO sku_sales VALUES
  ('books','A',1000),
  ('books','B',1000),
  ('books','C',800),
  ('templates','D',700),
  ('templates','E',600),
  ('templates','F',600);
```

Запрос

```
  SELECT
  category,
  sku,
  revenue,
  RANK() OVER (PARTITION BY category ORDER
BY revenue DESC) AS rnk,
  DENSE_RANK() OVER (PARTITION BY category
ORDER BY revenue DESC) AS dense_rnk
  FROM sku_sales
  ORDER BY category, rnk, sku;
```

Ожидаемый результат

```
  [
  [
  "books",
  "A",
  1000,
  1,
  1
  ],
  [
```

```
"books",
"B",
1000,
1,
1
],
[
"books",
"C",
800,
3,
2
],
[
"templates",
"D",
700,
1,
1
],
[
"templates",
"E",
600,
2,
2
```

```
],
[
"templates",
"F",
600,
2,
2
]
]
```

Почему это работает

RANK оставляет пропуск после ничьей: две первые строки делят первое место, следующая становится третьей.

DENSE_RANK не оставляет дырок: после первого места идёт второе. Для витрины или фильтра “топ-2 уровня” это часто удобнее.

Обе функции смотрят только на строки своей категории, потому что задан PARTITION BY category.

Типичная ошибка

Нельзя автоматически говорить, что RANK хуже DENSE_RANK. Для спортивной таблицы пропуски нормальны, а для группировки уровней обычно удобнее плот-

ный ранг.

Самостоятельная проверка

Замените C на 1000. Какие ранги получит каждая книга в категории books?

Вернуться к маршруту

Задача 3. Накопительная выручка по дням

Рабочий вопрос

Нужно построить ежедневную строку отчёта и рядом показать сумму выручки с начала периода по текущий день.

Данные

```
  CREATE   TABLE   daily_sales(day   TEXT,
revenue INTEGER);
  INSERT INTO daily_sales VALUES
  ('2026-09-01',1200),
  ('2026-09-02',500),
  ('2026-09-03',900),
  ('2026-09-04',400);
```

Запрос

```
  SELECT
```

Конец ознакомительного фрагмента.