ГЛЕБ ЗАЙЦЕВ

UNION
В SQL

18 задач на объединение таблиц

```
SELECT email FROM buyers_2025
UNION
SELECT email FROM buyers_2026
ORDER BY email;
```

UNION · UNION ALL · дубли

источники · периоды · отчёты

все запросы запускаются в SQLite

Глеб Зайцев

UNION в SQL. 18 задач на объединение таблиц

https://litres.ru/74419702
SelfPub; 2026

Аннотация

Практическая книга по UNION и UNION ALL в SQL для аналитиков и начинающих разработчиков. Внутри 18 задач на маленьких таблицах: объединение периодов, удаление дублей, сохранение дублей, общий список клиентов, витрина событий, источники лидов, категории из разных справочников, совместимые колонки и финальный отчёт из нескольких потоков. Все примеры запускаются в SQLite, данные встроены прямо в книгу.

Содержание

Глеб Зайцев
UNION в SQL. 18 задач на объединение таблиц

Перед первой задачей

Зачем нужен UNION

UNION нужен, когда данные лежат в нескольких похожих таблицах или запросах, а в отчёте нужна одна общая лента строк.

Типичная ситуация: продажи за разные периоды, клиенты из разных источников, события сайта и приложения, старый и новый справочник.

Главная развилка проста: UNION убирает дубли, UNION ALL сохраняет всё. В отчёте это меняет не только внешний вид, но и итоговые числа.

Как не сломать объединение

У всех частей UNION должно быть одинаковое количество колонок в одинаковом порядке. Названия важны для читаемости, но структура важнее.

Сортировку обычно ставят в самом конце общего запроса. Если сортировать каждую часть отдельно, итоговый отчёт может остаться неупорядоченным.

Перед объединением полезно добавить колонку source. Тогда после UNION ALL видно, откуда пришла каждая строка.

Маршрут по задачам

Практические задачи

Задача 1. Собрать клиентов из двух таблиц

Рабочий вопрос

Нужно получить общий список email из двух источников клиентов и убрать повторяющиеся адреса.

Данные

```
  CREATE TABLE buyers_2025(email TEXT);
  CREATE TABLE buyers_2026(email TEXT);
  INSERT    INTO    buyers_2025    VALUES
('anna@example.test'),
('oleg@example.test');
  INSERT    INTO    buyers_2026    VALUES
('oleg@example.test'),
('nina@example.test');
```

Запрос

```
SELECT email FROM buyers_2025
UNION
SELECT email FROM buyers_2026
ORDER BY email;
```

Ожидаемый результат

```
[["anna@example.test"],
["nina@example.test"],
["oleg@example.test"]]
```

Почему работает

UNION объединяет строки из двух SELECT.
Повтор oleg@example.test остаётся один раз.
ORDER BY в конце сортирует уже общий результат.

Где легко ошибиться

UNION удаляет дубли. Если повторная покупка важна как отдельный факт, такой запрос может занизить количество событий.

Самостоятельная проверка

Почему oleg@example.test показан один раз, хотя есть в обеих исходных таблицах?

Ответ: Потому что UNION возвращает уникальные строки общего результата.

Задача 2. Сохранить повторы через UNION ALL

Рабочий вопрос

Нужно объединить продажи двух дней и сохранить каждую строку, даже если суммы совпадают.

Данные

```
  CREATE TABLE sales_day1(id INTEGER, total
INTEGER);
  CREATE TABLE sales_day2(id INTEGER, total
INTEGER);
  INSERT INTO sales_day1 VALUES (1,500),
(2,300);
  INSERT INTO sales_day2 VALUES (3,500),
(4,200);
```

Запрос

```
  SELECT total FROM sales_day1
  UNION ALL
```

```
SELECT total FROM sales_day2
ORDER BY total DESC;
```

Ожидаемый результат

```
[[500], [500], [300], [200]]
```

Почему работает

UNION ALL не удаляет одинаковые строки.
Две продажи на 500 остаются двумя строками.
Сортировка выполняется после объединения двух дней.

Где легко ошибиться

Если заменить UNION ALL на UNION, одна из одинаковых сумм 500 исчезнет из результата.

Самостоятельная проверка

Почему для отчёта по фактам продаж UNION ALL безопаснее, чем UNION?

Ответ: Факты продаж могут иметь одинаковые значения, и их нельзя удалять как дубли без отдельного правила.

Задача 3. Добавить источник каждой строки

Рабочий вопрос

Нужно объединить лиды из сайта и формы, сохранив колонку, которая показывает исходный канал.

Данные

```
CREATE TABLE site_leads(email TEXT);
CREATE TABLE form_leads(email TEXT);
INSERT INTO site_leads VALUES
('a@example.test'),('b@example.test');
INSERT INTO form_leads VALUES
('b@example.test'),('c@example.test');
```

Запрос

```
SELECT 'site' AS source, email FROM
site_leads
UNION ALL
SELECT 'form' AS source, email FROM
```

```
form_leads
  ORDER BY email, source;
```

Ожидаемый результат

```
  [["site",  "a@example.test"],  ["form",
"b@example.test"],                ["site",
"b@example.test"],                ["form",
"c@example.test"]]
```

Почему работает

Каждый SELECT добавляет постоянную колонку source.

UNION ALL сохраняет одинаковый email из разных каналов.

Конец ознакомительного фрагмента.