SQLite Tutorial: Demonstrating INNER and LEFT JOINs

Dr. Frank Xu

October 18, 2025

Introduction

This tutorial demonstrates the use of **INNER JOIN** and **LEFT JOIN** in SQLite using a dataset with three users: two with orders (Alice and Bob) and one without (Charlie). We will create tables, insert data, display the data after insertion, and compare the results of both JOIN operations to highlight their differences.

Step 1: Create the Tables

We create two tables: `users` to store user information and `orders` to store order details.

```
-- Create users table
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT
);

-- Create orders table
CREATE TABLE orders (
    order_id INTEGER PRIMARY KEY,
    user_id INTEGER,
    product TEXT,
    amount REAL,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

Step 2: Insert Data for Three Users

We insert three users—Alice, Bob, and Charlie—and orders for only Alice and Bob. Charlie has no orders.

```sql
-- Insert three users
INSERT INTO users (name, email) VALUES
('Alice', 'alice@example.com'),
('Bob', 'bob@example.com'),
('Charlie', 'charlie@example.com');

-- Insert orders for Alice and Bob only
INSERT INTO orders (user_id, product, amount) VALUES
(1, 'Laptop', 999.99),  -- Alice's order
(1, 'Mouse', 29.99),    -- Alice's order
(2, 'Keyboard', 59.99); -- Bob's order
```

0.1 Results After Step 2

We display the contents of both tables after inserting the data.

0.1.1 Users Table

```sql
SELECT * FROM users;
```

Result:

user_id	name	email
1	Alice	alice@example.com
2	Bob	bob@example.com
3	Charlie	charlie@example.com

0.1.2 Orders Table

```sql
SELECT * FROM orders;
```

Result:

order_id	user_id	product	amount
1	1	Laptop	999.99
2	1	Mouse	29.99
3	2	Keyboard	59.99

Explanation: The `users` table contains three users: Alice (`user_id = 1`), Bob (`user_id = 2`), and Charlie (`user_id = 3`). The `orders` table contains three orders: two for Alice (`user_id = 1`, Laptop and Mouse) and one for Bob (`user_id = 2`, Keyboard). Charlie (`user_id = 3`) has no orders.

Step 3: INNER JOIN Example

An **INNER JOIN** returns only rows where there is a match in both tables.

```sql
SELECT users.name, orders.product, orders.amount
FROM users
INNER JOIN orders
ON users.user_id = orders.user_id;
```

Result:

```
name    product     amount
Alice   Laptop      999.99
Alice   Mouse       29.99
Bob     Keyboard    59.99
```

Explanation: Only users with orders appear (Alice and Bob). Charlie has no orders, so he is excluded because there is no matching `user_id = 3` in the `orders` table. The INNER JOIN shows only rows where `users.user_id` equals `orders.user_id`.

Old-Style Equivalent (for reference):

```sql
-- Older implicit join syntax (not recommended)
SELECT users.name, orders.product, orders.amount
FROM users, orders
WHERE users.user_id = orders.user_id;
```

Note: The old-style query produces the same result as the modern `INNER JOIN`, but it is harder to read and can lead to mistakes if the `WHERE` condition is omitted. Always use the explicit `JOIN ... ON ...` form in modern SQL.

Step 4: LEFT JOIN Example

A **LEFT JOIN** returns all rows from the left table (`users`), with matching rows from the right table (`orders`). If there is no match, NULL values are returned for the right table's columns.

```sql
SELECT users.name, orders.product, orders.amount
FROM users
LEFT JOIN orders
ON users.user_id = orders.user_id;
```

Result:

```
name    product     amount
Alice   Laptop      999.99
Alice   Mouse       29.99
Bob     Keyboard    59.99
Charlie NULL        NULL
```

Explanation: All users are included because `users` is the left table. Alice and Bob have orders, so their rows show the corresponding `product` and `amount`. Charlie has no orders, so his row shows `NULL` for `product` and `amount`. The LEFT JOIN preserves all rows from `users`, even when there is no match in `orders`.

Key Differences Between INNER JOIN and LEFT JOIN

- **INNER JOIN**:
  - Only includes rows with matches in both tables.
  - Excludes Charlie because he has no orders.
  - Result: 3 rows (Alice's two orders, Bob's one order).
- **LEFT JOIN**:
  - Includes all rows from the left table (`users`).
  - Shows Charlie with `NULL` for order details.
  - Result: 4 rows (Alice's two orders, Bob's one order, Charlie's `NULL` row).

Try It Yourself

To practice:

1. Modify the INNER JOIN to show only orders with `amount > 100`.
2. Use LEFT JOIN to find users with no orders (hint: add `WHERE orders.user_id IS NULL`).
3. Add an order for Charlie and rerun both JOINs to see how the results change.

You can run these queries in DB Browser for SQLite or `sqliteonline.com`.