

# Lesson 6: Table Joins

---

**Duration:** 20 minutes

**Deliverable:** `lesson6_joins.sql`

## Learning Objectives

---

By the end of this lesson, you will be able to:

- Understand why we need JOINs
- Write INNER JOIN queries
- Write LEFT JOIN queries
- Combine data from multiple tables
- Handle NULL values in joins
- Use joins with WHERE, ORDER BY, and aggregate functions



## What are JOINs?

---

**JOINs** combine rows from two or more tables based on a related column. They allow you to retrieve data spread across multiple tables in a single query.

**Without JOINs:** - Query characters: Get `homeworld_id = 1` - Query planets: Find that ID 1 = "Tatooine" - Manually connect the information

**With JOINs:** - One query returns character name AND planet name together!

## Types of JOINS

Join Type	Returns	Use When
<b>INNER JOIN</b>	Only matching rows from both tables	You want records that exist in both tables
<b>LEFT JOIN</b>	All rows from left table, matching from right	You want all from table A, even if no match in table B
<b>RIGHT JOIN</b>	All rows from right table, matching from left	SQLite doesn't support; use LEFT JOIN instead



## Part 1: INNER JOIN (8 minutes)

---

### Step 1: Create Your SQL File

1. Create: lesson6\_joins.sql
2. Add header:

```
-- Lesson 6: Table Joins
-- Student Name: [Your Name]
-- Date: [Today's Date]
--
-- This script demonstrates INNER and LEFT joins
```

### INNER JOIN Syntax

```
SELECT columns
FROM table1
INNER JOIN table2 ON table1.column = table2.column;
```

### Step 2: Join Characters and Planets

```
-- Query 1: Show characters with their homeworld details
SELECT
    characters.name AS character_name,
    characters.species,
    planets.name AS homeworld_name,
    planets.climate
FROM characters
INNER JOIN planets ON characters.homeworld_id = planets.id;
```

**Explanation:** - characters.name - Specify which table the column comes from - AS character\_name - Rename column in results (avoids confusion when both tables have "name") - ON characters.homeworld\_id = planets.id - How tables relate

**Execute this query** to see characters with planet details!

### Step 3: Using Table Aliases

Table names can be long. Use **aliases** to shorten them:

```
-- Query 2: Same query with table aliases
SELECT
    c.name AS character_name,
    c.species,
    p.name AS planet_name,
    p.climate,
    p.population
FROM characters c
INNER JOIN planets p ON c.homeworld_id = p.id;
```

**Much cleaner!** `c` is shorthand for `characters`, `p` for `planets`.

## Step 4: Join Three Tables

Let's combine characters, vehicles, and the junction table:

```
-- Query 3: Show which characters pilot which vehicles
SELECT
    c.name AS character_name,
    v.name AS vehicle_name,
    v.vehicle_class
FROM characters c
INNER JOIN character_vehicles cv ON c.id = cv.character_id
INNER JOIN vehicles v ON cv.vehicle_id = v.id
ORDER BY c.name;
```

**Explanation:** 1. Start with `characters` table 2. Join to `character_vehicles` (junction table) linking characters to vehicles 3. Join to `vehicles` table to get vehicle details 4. Order by character name

## Step 5: JOIN with WHERE

```
-- Query 4: Find all humans and their homeworlds
SELECT
    c.name,
    c.species,
    p.name AS homeworld
FROM characters c
INNER JOIN planets p ON c.homeworld_id = p.id
WHERE c.species = 'Human';
```

**Explanation:** JOIN first, then filter with WHERE.

## Step 6: JOIN with Aggregate Functions

```
-- Query 5: Count how many characters are from each planet
SELECT
    p.name AS planet_name,
    COUNT(c.id) AS character_count
FROM planets p
INNER JOIN characters c ON p.id = c.homeworld_id
GROUP BY p.name
ORDER BY character_count DESC;
```

**Notice:** Only shows planets that have characters (INNER JOIN requirement).



## Part 2: LEFT JOIN (8 minutes)

LEFT JOIN returns ALL rows from the left table, plus matching rows from the right table. If there's no match, NULL appears for right table columns.

### LEFT JOIN Syntax

```
SELECT columns
FROM table1
LEFT JOIN table2 ON table1.column = table2.column;
```

### Step 7: Find Characters Without Vehicles

```
-- Query 6: List all characters and their vehicles (including those with no vehicles)
SELECT
    c.name AS character_name,
    v.name AS vehicle_name
FROM characters c
LEFT JOIN character_vehicles cv ON c.id = cv.character_id
LEFT JOIN vehicles v ON cv.vehicle_id = v.id
ORDER BY c.name;
```

**Result:** Characters without vehicles show NULL for vehicle\_name.

### Step 8: Filter for Characters WITHOUT Vehicles

```
-- Query 7: Find characters who don't pilot any vehicles
SELECT
    c.name AS character_name,
    c.species
FROM characters c
LEFT JOIN character_vehicles cv ON c.id = cv.character_id
WHERE cv.vehicle_id IS NULL;
```

**Explanation:** After LEFT JOIN, rows without matches have NULL. We filter for those.

### Step 9: Find Vehicles Without Pilots

```
-- Query 8: Find vehicles that no character pilots
SELECT
    v.name AS vehicle_name,
    v.vehicle_class
FROM vehicles v
LEFT JOIN character_vehicles cv ON v.id = cv.vehicle_id
WHERE cv.character_id IS NULL;
```

## Step 10: Count Including Empty Groups

```
-- Query 9: Count characters per planet (including planets with 0 characters)
SELECT
    p.name AS planet_name,
    COUNT(c.id) AS character_count
FROM planets p
LEFT JOIN characters c ON p.id = c.homeworld_id
GROUP BY p.name
ORDER BY character_count DESC;
```

**Difference from Query 5:** This shows ALL planets, even those with no characters (count = 0).

## Part 3: Complex JOIN Queries (4 minutes)

### Step 11: Multiple JOINS with Filtering

```
-- Query 10: Find humans who pilot starfighters
SELECT
    c.name AS character_name,
    v.name AS vehicle_name,
    v.vehicle_class
FROM characters c
INNER JOIN character_vehicles cv ON c.id = cv.character_id
INNER JOIN vehicles v ON cv.vehicle_id = v.id
WHERE c.species = 'Human' AND v.vehicle_class = 'Starfighter';
```

### Step 12: JOIN with Aggregate and HAVING

```
-- Query 11: Find characters who pilot more than one vehicle
SELECT
    c.name AS character_name,
    COUNT(v.id) AS vehicle_count
FROM characters c
INNER JOIN character_vehicles cv ON c.id = cv.character_id
INNER JOIN vehicles v ON cv.vehicle_id = v.id
GROUP BY c.name
HAVING COUNT(v.id) > 1;
```

### Step 13: Comprehensive Query

```
-- Query 12: Character summary with all related data
SELECT
    c.name AS character,
    c.species,
    p.name AS homeworld,
    p.climate,
    COUNT(v.id) AS vehicles_piloted
FROM characters c
LEFT JOIN planets p ON c.homeworld_id = p.id
LEFT JOIN character_vehicles cv ON c.id = cv.character_id
LEFT JOIN vehicles v ON cv.vehicle_id = v.id
GROUP BY c.name, c.species, p.name, p.climate
ORDER BY c.name;
```

## Practice Exercises

---

### Exercise 1: Simple INNER JOIN

```
-- Exercise 1: List all characters with their homeworld's population
SELECT
    c.name,
    p.name AS homeworld,
    p.population
FROM characters c
INNER JOIN planets p ON c.homeworld_id = p.id;
```

### Exercise 2: Multiple JOINS

```
-- Exercise 2: Show all vehicle-pilot pairs with character species
SELECT
    c.name AS pilot,
    c.species,
    v.name AS vehicle
FROM characters c
INNER JOIN character_vehicles cv ON c.id = cv.character_id
INNER JOIN vehicles v ON cv.vehicle_id = v.id;
```

### Exercise 3: LEFT JOIN with NULL Check

```
-- Exercise 3: Find all planets with no characters
SELECT
    p.name AS planet_name,
    p.climate
FROM planets p
LEFT JOIN characters c ON p.id = c.homeworld_id
WHERE c.id IS NULL;
```

### Exercise 4: Aggregate with JOIN

```
-- Exercise 4: Show each vehicle with the count of who pilots it
SELECT
    v.name AS vehicle_name,
    COUNT(c.id) AS pilot_count
FROM vehicles v
LEFT JOIN character_vehicles cv ON v.id = cv.vehicle_id
LEFT JOIN characters c ON cv.character_id = c.id
GROUP BY v.name;
```



## Common Errors & Troubleshooting

---

### Error: "ambiguous column name"

**Problem:** Column exists in multiple tables and you didn't specify which.

#### Wrong:

```
SELECT name FROM characters
INNER JOIN planets ON homeworld_id = id;
```

#### Correct:

```
SELECT characters.name FROM characters
INNER JOIN planets ON characters.homeworld_id = planets.id;
```

### Error: "no such column"

**Problem:** Misspelt column or using wrong table prefix.

**Solution:** Verify column names:

```
-- Check column names in each table
PRAGMA table_info(characters);
PRAGMA table_info(planets);
```

### Wrong JOIN Type

**Symptom:** Missing expected rows.

**Problem:** Used INNER JOIN when you needed LEFT JOIN.

**Remember:** - INNER JOIN: Only rows with matches in BOTH tables - LEFT JOIN: ALL rows from left table, matches from right

### Incorrect ON Clause

**Problem:** Joining on wrong columns.

#### Wrong:

```
FROM characters c
INNER JOIN planets p ON c.id = p.id -- Wrong columns!
```

#### Correct:

```
FROM characters c
INNER JOIN planets p ON c.homeworld_id = p.id -- Correct relationship
```

## Cartesian Product (Too Many Results)

**Problem:** Missing ON clause creates every possible combination.

**Wrong:**

```
SELECT * FROM characters, planets; -- Returns 11 × 8 = 88 rows!
```

**Correct:**

```
SELECT * FROM characters
INNER JOIN planets ON characters.homeworld_id = planets.id;
```

## Checkpoint: What You've Learnt

---

Before moving on, make sure you can:

-  Explain the purpose of JOINs
-  Write INNER JOIN queries
-  Write LEFT JOIN queries
-  Use table aliases for clarity
-  Join three or more tables
-  Combine JOINs with WHERE, GROUP BY, and HAVING
-  Find records with no matches using LEFT JOIN and NULL

## Challenge Problem (Optional)

**Task:** Create a query that shows each planet with: - Planet name - Climate - Number of characters from that planet - Average height of characters from that planet - Include planets with NO characters (show 0 for count, NULL for average) - Order by character count (descending), then planet name

**Hints:** - Use LEFT JOIN - Use COUNT() and AVG() - Use GROUP BY - Use ORDER BY with multiple columns

Click to reveal the solution

```
SELECT
    p.name AS planet_name,
    p.climate,
    COUNT(c.id) AS character_count,
    ROUND(AVG(c.height), 1) AS avg_height
FROM planets p
LEFT JOIN characters c ON p.id = c.homeworld_id
GROUP BY p.name, p.climate
ORDER BY character_count DESC, p.name;
```

## Save Your Work with Git

---

```
git status  
git add lessons/lesson6_joins.sql  
git commit -m "Completed Lesson 6: INNER and LEFT joins across multiple tables"  
git push
```



## Key SQL Commands Learnt

Command	Purpose	Example
INNER JOIN	Return only matching rows	INNER JOIN planets ON c.homeworld_id = p.id
LEFT JOIN	Return all from left table	LEFT JOIN vehicles ON c.id = v.pilot_id
ON	Specify join condition	ON table1.id = table2.foreign_id
Table Alias	Shorten table names	FROM characters c
IS NULL	Check for NULL values	WHERE cv.vehicle_id IS NULL



## JOIN Visual Reference

---

### INNER JOIN:

Table A: [1, 2, 3]

Table B: [2, 3, 4]

Result: [2, 3] ← Only matches

### LEFT JOIN:

Table A: [1, 2, 3]

Table B: [2, 3, 4]

Result: [1, 2, 3] ← All from A, with B data where available



## Fantastic Work!

---

You can now combine data from multiple tables! In the next lesson, you'll learn how to modify data with UPDATE and DELETE statements.

**Ready to continue?** Move on to `lesson7_instructions.md`

**Need Help?** - Draw table relationships on paper - Test JOIN without WHERE first - Check which table has which columns - Verify foreign key relationships - Ask your instructor!