Lesson 3: Sorting and Limiting Results

Duration: 20 minutes
Deliverable: `lesson3_sorting.sql`

🎯 Learning Objectives

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

- Sort query results with ORDER BY
- Sort in ascending and descending order
- Sort by multiple columns
- Limit the number of results with LIMIT
- Use OFFSET for pagination
- Combine WHERE, ORDER BY, and LIMIT

📚 Why Sort and Limit?

Databases don't store data in any particular order. When you query a table, rows can appear in any sequence. Sorting and limiting helps you:

- **Organise** data logically (alphabetically, numerically)
- **Find** the top or bottom values (highest, lowest, first, last)
- **Improve performance** by retrieving only what you need
- **Create rankings** and leaderboards
- **Paginate** results (like pages in a search engine)

🛠️ Part 1: Sorting with ORDER BY (8 minutes)

Step 1: Create Your SQL File

1. Navigate to the `lessons/` folder
2. Create a new file: `lesson3_sorting.sql`
3. Add a header:

```
-- Lesson 3: Sorting and Limiting Results
-- Student Name: [Your Name]
-- Date: [Today's Date]
--
-- This script demonstrates ORDER BY and LIMIT clauses
```

ORDER BY Syntax

```
SELECT columns FROM table_name ORDER BY column_name;
```

By default, ORDER BY sorts in **ascending** order (A-Z, 0-9, smallest to largest).

Step 2: Add Height Data First

Before we can sort by height, let's add a height column to our characters table:

```
-- Add height column to characters table
ALTER TABLE characters ADD COLUMN height INTEGER;

-- Update characters with height data (in centimetres)
UPDATE characters SET height = 172 WHERE name = 'Luke Skywalker';
UPDATE characters SET height = 150 WHERE name = 'Leia Organa';
UPDATE characters SET height = 180 WHERE name = 'Han Solo';
UPDATE characters SET height = 228 WHERE name = 'Chewbacca';
UPDATE characters SET height = 182 WHERE name = 'Obi-Wan Kenobi';
UPDATE characters SET height = 202 WHERE name = 'Darth Vader';
UPDATE characters SET height = 66 WHERE name = 'Yoda';
UPDATE characters SET height = 96 WHERE name = 'R2-D2';
```

Execute these queries to add height data.

Step 3: Sort Alphabetically by Name

```
-- Query 1: View all characters sorted by name (A-Z)
SELECT name, species, homeworld FROM characters ORDER BY name;
```

Result: Characters appear in alphabetical order.

Step 4: Sort by Species

```
-- Query 2: View characters sorted by species
SELECT name, species FROM characters ORDER BY species;
```

Notice: Groups similar species together.

Step 5: Sort Numerically by Height

```
-- Query 3: View characters sorted by height (shortest to tallest)
SELECT name, height FROM characters ORDER BY height;
```

📊 Part 2: Ascending vs Descending (5 minutes)

You can control sort direction with `ASC` (ascending) or `DESC` (descending).

ASC: Smallest to Largest (Default)

```
-- Query 4: Explicitly sort ascending (same as Query 3)
SELECT name, height FROM characters ORDER BY height ASC;
```

DESC: Largest to Smallest

```
-- Query 5: Sort by height (tallest to shortest)
SELECT name, height FROM characters ORDER BY height DESC;
```

Use Case: Finding the tallest character!

Sort Names in Reverse Alphabetical Order

```
-- Query 6: Sort names Z-A
SELECT name FROM characters ORDER BY name DESC;
```

🔗 Part 3: Sorting by Multiple Columns (7 minutes)

You can sort by multiple columns. The first column is the primary sort, the second breaks ties.

Syntax

```
SELECT columns FROM table_name ORDER BY column1, column2;
```

Step 6: Sort by Species, Then Name

```
-- Query 7: Sort by species first, then by name within each species
SELECT name, species, homeworld
FROM characters
ORDER BY species, name;
```

Result: Characters are grouped by species, and within each species group, they're alphabetically sorted by name.

Step 7: Mix Ascending and Descending

```
-- Query 8: Sort by species (A-Z), then height (tallest to shortest)
SELECT name, species, height
FROM characters
ORDER BY species ASC, height DESC;
```

Explanation:

- First, groups by species alphabetically
- Within each species group, sorts by height (tallest first)

Step 8: Sort by Homeworld and Species

```
-- Query 9: Group by homeworld, then by species
SELECT homeworld, species, name
FROM characters
ORDER BY homeworld, species;
```

🎯 Part 4: Limiting Results with LIMIT (5 minutes)

The `LIMIT` clause restricts how many rows are returned.

LIMIT Syntax

```
SELECT columns FROM table_name LIMIT number;
```

Step 9: Get Top 5 Results

```
-- Query 10: View only the first 5 characters
SELECT name FROM characters LIMIT 5;
```

Use Case: Showing preview data or improving performance.

Step 10: Find the Tallest Character

```
-- Query 11: Find the tallest character
SELECT name, height
FROM characters
ORDER BY height DESC
LIMIT 1;
```

Result: Chewbacca (228 cm)

Step 11: Find the 3 Shortest Characters

```
-- Query 12: Find the three shortest characters
SELECT name, height
FROM characters
ORDER BY height ASC
LIMIT 3;
```

Step 12: Top 5 Names Alphabetically

```
-- Query 13: Get the first 5 names alphabetically
SELECT name FROM characters ORDER BY name LIMIT 5;
```

📄 Part 5: Pagination with OFFSET (3 minutes)

`OFFSET` skips a specified number of rows. Combined with LIMIT, it enables pagination.

OFFSET Syntax

```
SELECT columns FROM table_name LIMIT number OFFSET number;
```

Step 13: Skip the First 3 Characters

```
-- Query 14: Get characters 4-8 (skip first 3)
SELECT name FROM characters ORDER BY name LIMIT 5 OFFSET 3;
```

Explanation:

- Skip first 3 rows
- Then return the next 5 rows

Step 14: Pagination Example

```
-- Page 1: First 3 characters
SELECT name FROM characters ORDER BY name LIMIT 3 OFFSET 0;

-- Page 2: Next 3 characters
SELECT name FROM characters ORDER BY name LIMIT 3 OFFSET 3;

-- Page 3: Next 3 characters
SELECT name FROM characters ORDER BY name LIMIT 3 OFFSET 6;
```

Use Case: Displaying results across multiple pages in an application.

🔗 Part 6: Combining WHERE, ORDER BY, and LIMIT (2 minutes)

You can combine all the techniques you've learnt!

Order of Clauses

```
SELECT columns
FROM table_name
WHERE condition
ORDER BY column
LIMIT number;
```

Important: The order matters! Always: SELECT → FROM → WHERE → ORDER BY → LIMIT

Step 15: Filter, Sort, and Limit

```
-- Query 15: Find the tallest human
SELECT name, species, height
FROM characters
WHERE species = 'Human'
ORDER BY height DESC
LIMIT 1;
```

Step 16: Top 3 Characters from Specific Planets

```
-- Query 16: Find 3 characters NOT from Tatooine, sorted by name
SELECT name, homeworld
FROM characters
WHERE homeworld != 'Tatooine'
ORDER BY name
LIMIT 3;
```

🎓 Practice Exercises

Complete these queries in your `lesson3_sorting.sql` file:

Exercise 1: Sort and Limit

```
-- Exercise 1: Find the 5 tallest characters
SELECT name, height
FROM characters
ORDER BY height DESC
LIMIT 5;
```

Exercise 2: Alphabetical Species

```
-- Exercise 2: List all unique species in alphabetical order
SELECT DISTINCT species
FROM characters
ORDER BY species;
```

Exercise 3: Filter and Sort

```
-- Exercise 3: Find all humans sorted by height (shortest first)
SELECT name, species, height
FROM characters
WHERE species = 'Human'
ORDER BY height ASC;
```

Exercise 4: Complex Query

```
-- Exercise 4: Find the second and third tallest characters
SELECT name, height
FROM characters
ORDER BY height DESC
LIMIT 2 OFFSET 1;
```

🐛 Common Errors & Troubleshooting

Error: "no such column: height"

Problem: Height column doesn't exist.

Solution: Run the ALTER TABLE and UPDATE statements from Step 2.

Results in Wrong Order

Problem: Forgot ORDER BY or used wrong column.

Solution:

```
-- WRONG: No ordering specified
SELECT name FROM characters LIMIT 5;

-- CORRECT: Explicit ordering
SELECT name FROM characters ORDER BY name LIMIT 5;
```

ORDER BY Clauses in Wrong Order

Problem: SQL clauses must be in specific order.

Solution:

```
-- WRONG: LIMIT before ORDER BY
SELECT name FROM characters LIMIT 5 ORDER BY name;

-- CORRECT: ORDER BY before LIMIT
SELECT name FROM characters ORDER BY name LIMIT 5;
```

Sorting NULL Values

Problem: NULL values appear first or last depending on database.

Behaviour in SQLite: NULL values sort FIRST in ascending order.

Solution: Filter out NULLs if needed:

```
SELECT name, height
FROM characters
WHERE height IS NOT NULL
ORDER BY height;
```

OFFSET Without ORDER BY

Problem: Using OFFSET without ORDER BY gives unpredictable results.

Solution: Always use ORDER BY with OFFSET:

```
-- WRONG: Unpredictable results
SELECT name FROM characters LIMIT 3 OFFSET 3;

-- CORRECT: Predictable, ordered results
SELECT name FROM characters ORDER BY name LIMIT 3 OFFSET 3;
```

Case Sensitivity in Sorting

Note: Sorting is case-sensitive in SQLite by default.

Example: 'a' comes after 'Z' because lowercase has higher ASCII values.

Solution: Use `COLLATE NOCASE` for case-insensitive sorting:

```
SELECT name FROM characters ORDER BY name COLLATE NOCASE;
```

✅ Checkpoint: What You've Learnt

Before moving on, make sure you can:

- ✅ Sort results with ORDER BY
- ✅ Use ASC and DESC for sort direction
- ✅ Sort by multiple columns
- ✅ Limit results with LIMIT
- ✅ Use OFFSET for pagination
- ✅ Combine WHERE, ORDER BY, and LIMIT correctly
- ✅ Understand the order of SQL clauses

🎯 Challenge Problem (Optional)

Task: Find characters whose names contain the letter 'a', sorted by height (shortest first), and show only the second and third results.

Requirements:

- Use WHERE with LIKE
- Use ORDER BY
- Use LIMIT and OFFSET

Write your solution:

```
-- Challenge Problem
-- YOUR CODE HERE
```

Click to reveal the solution

```
SELECT name, height
FROM characters
WHERE name LIKE '%a%'
ORDER BY height ASC
LIMIT 2 OFFSET 1;
```

💾 Save Your Work with Git

Save your progress to GitHub!

Step 1: Check Status

```
git status
```

Step 2: Stage Your File

```
git add lessons/lesson3_sorting.sql database/starwars.db
```

Step 3: Commit

```
git commit -m "Completed Lesson 3: Sorting and limiting query results"
```

Step 4: Push

```
git push
```

📖 Key SQL Commands Learnt

Command	Purpose	Example
`ORDER BY`	Sort results	`ORDER BY name`
`ASC`	Sort ascending (default)	`ORDER BY height ASC`
`DESC`	Sort descending	`ORDER BY height DESC`
`LIMIT`	Restrict number of results	`LIMIT 5`
`OFFSET`	Skip rows	`LIMIT 5 OFFSET 3`
`ALTER TABLE`	Modify table structure	`ALTER TABLE characters ADD COLUMN height INTEGER`
`UPDATE`	Modify existing data	`UPDATE characters SET height = 172 WHERE name = 'Luke'`

📊 SQL Clause Order Reference

Remember, SQL clauses must appear in this order:

```
SELECT columns
FROM table
WHERE condition
ORDER BY column
LIMIT number
OFFSET number;
```

🎉 Great Progress!

You can now organise and control your query results! In the next lesson, you'll learn how to perform calculations on groups of data using aggregate functions.

Ready to continue? Move on to `lesson4_instructions.md`

Need Help?

- Check clause order carefully
- Test without LIMIT first to see all results
- Verify column names exist
- Ask your instructor
- Compare with the solution file (after attempting yourself!)