Lesson 2: Selecting and Filtering Data (SELECT & WHERE)

Duration: 20 minutes
Deliverable: `lesson2_queries.sql`

🎯 Learning Objectives

By the end of this lesson, you will be able to: - Retrieve specific columns from a table using SELECT - Filter data using the WHERE clause - Use comparison operators (=, !=, <, >, <=, >=) - Combine conditions with AND and OR - Use pattern matching with LIKE and wildcards

📚 Introduction to Querying Data

In Lesson 1, you created a database and inserted data. Now you'll learn how to **retrieve** specific information from your database. This is what makes databases so powerful!

The `SELECT` statement is the most commonly used SQL command. It allows you to: - View all data in a table - Select specific columns - Filter rows based on conditions - Search for patterns in text

🛠️ Part 1: The SELECT Statement (8 minutes)

Step 1: Create Your SQL File

1. Navigate to the `lessons/` folder
2. Create a new file: `lesson2_queries.sql`
3. Add a header comment:

```
-- Lesson 2: Selecting and Filtering Data
-- Student Name: [Your Name]
-- Date: [Today's Date]
--
-- This script demonstrates SELECT queries with WHERE clauses
```

Basic SELECT Syntax

The simplest SELECT statement retrieves all data from a table:

```
SELECT * FROM table_name;
```

- `SELECT` - Command to retrieve data
- `*` - Means "all columns"
- `FROM` - Specifies which table
- `table_name` - Name of the table

Step 2: Select All Characters

Add this query to your file:

```
-- Query 1: View all characters and all columns
SELECT * FROM characters;
```

Execute this query to see all your characters.

Step 3: Select Specific Columns

What if you only want to see names and species?

```
-- Query 2: View only names and species
SELECT name, species FROM characters;
```

Notice: The output now only shows two columns!

Step 4: Select Columns in Different Order

```
-- Query 3: View columns in a different order
SELECT homeworld, name, species FROM characters;
```

Key Point: Columns appear in the order you specify, not the table's order.

🔍 Part 2: Filtering with WHERE (8 minutes)

The `WHERE` clause filters rows based on conditions. Only rows that meet the condition are returned.

Basic WHERE Syntax

```
SELECT columns FROM table_name WHERE condition;
```

Comparison Operators

Operator	Meaning	Example
`=`	Equal to	`species = 'Human'`
`!=` or `<>`	Not equal to	`species != 'Droid'`
`<`	Less than	`id < 5`
`>`	Greater than	`id > 3`
`<=`	Less than or equal	`id <= 10`
`>=`	Greater than or equal	`id >= 1`

Step 5: Find All Humans

```
-- Query 4: Find all human characters
SELECT * FROM characters WHERE species = 'Human';
```

Execute this query. You should see only the human characters.

Step 6: Find Characters from a Specific Planet

```
-- Query 5: Find all characters from Tatooine
SELECT name, homeworld FROM characters WHERE homeworld = 'Tatooine';
```

Step 7: Find Characters NOT from a Planet

```
-- Query 6: Find all characters who are NOT human
SELECT name, species FROM characters WHERE species != 'Human';
```

Note: Both `!=` and `<>` work for "not equal".

🔗 Part 3: Combining Conditions (5 minutes)

You can combine multiple conditions using `AND` and `OR`.

AND: Both Conditions Must Be True

```
-- Query 7: Find humans from Tatooine
SELECT name, species, homeworld
FROM characters
WHERE species = 'Human' AND homeworld = 'Tatooine';
```

Result: Only characters who are human AND from Tatooine.

OR: At Least One Condition Must Be True

```
-- Query 8: Find characters who are either Droids OR from Naboo
SELECT name, species, homeworld
FROM characters
WHERE species = 'Droid' OR homeworld = 'Naboo';
```

Result: Characters who match either condition.

Combining AND with OR

Use parentheses to group conditions:

```
-- Query 9: Find humans from either Tatooine or Alderaan
SELECT name, species, homeworld
FROM characters
WHERE species = 'Human' AND (homeworld = 'Tatooine' OR homeworld = 'Alderaan');
```

Why parentheses? They ensure the OR is evaluated first, then the AND.

🎨 Part 4: Pattern Matching with LIKE (7 minutes)

The `LIKE` operator searches for patterns in text using wildcards.

Wildcards

Wildcard	Meaning	Example
`%`	Any number of characters (including zero)	`'L%'` matches "Luke", "Leia"
`_`	Exactly one character	`'_oda'` matches "Yoda", "Boda"

Step 8: Find Names Starting with a Letter

```
-- Query 10: Find all characters whose names start with 'L'
SELECT name FROM characters WHERE name LIKE 'L%';
```

Result: Luke Skywalker, Leia Organa

Step 9: Find Names Ending with a Pattern

```
-- Query 11: Find all characters whose names end with 'o'
SELECT name FROM characters WHERE name LIKE '%o';
```

Step 10: Find Names Containing a Pattern

```
-- Query 12: Find all characters with 'Darth' in their name
SELECT name FROM characters WHERE name LIKE '%Darth%';
```

Key Point: The `%` matches any characters before and after "Darth".

Step 11: Species Containing Specific Letters

```
-- Query 13: Find all species containing 'oid'
SELECT DISTINCT species FROM characters WHERE species LIKE '%oid%';
```

Note: `DISTINCT` removes duplicates from results.

🎓 Practice Exercises

Complete these queries in your `lesson2_queries.sql` file:

Exercise 1: Find Specific Characters

```
-- Exercise 1: Find all characters from Kashyyyk
SELECT name, homeworld FROM characters WHERE homeworld = 'Kashyyyk';
```

Exercise 2: Exclude Species

```
-- Exercise 2: Find all characters who are NOT droids
SELECT name, species FROM characters WHERE species != 'Droid';
```

Exercise 3: Multiple Conditions

```
-- Exercise 3: Find all humans NOT from Tatooine
SELECT name, species, homeworld
FROM characters
WHERE species = 'Human' AND homeworld != 'Tatooine';
```

Exercise 4: Pattern Matching

```
-- Exercise 4: Find all characters whose names contain 'Sky'
SELECT name FROM characters WHERE name LIKE '%Sky%';
```

🐛 Common Errors & Troubleshooting

Error: "no such column"

Problem: Misspelt column name.

Solution:

```
-- WRONG:
SELECT naam FROM characters;  -- 'naam' doesn't exist

-- CORRECT:
SELECT name FROM characters;
```

Error: "no such table"

Problem: Table doesn't exist or misspelt table name.

Solution: - Check the table was created in Lesson 1 - Verify spelling: `characters` not `character`

No Results Returned

Problem: WHERE condition doesn't match any rows.

Solution: - Check your condition logic - Verify data exists: `SELECT * FROM characters;` - Check for case sensitivity: `'human'` vs `'Human'`

Case Sensitivity in SQLite

Important: SQLite text comparisons are case-sensitive by default.

```
-- These are DIFFERENT:
WHERE species = 'human'   -- Won't match 'Human'
WHERE species = 'Human'   -- Correct
```

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

```
SELECT * FROM characters WHERE species = 'human' COLLATE NOCASE;
```

LIKE Not Finding Results

Problem: Forgetting wildcards.

Solution:

```
-- WRONG: Looks for exact match 'Luke'
WHERE name LIKE 'Luke'

-- CORRECT: Finds names containing 'Luke'
WHERE name LIKE '%Luke%'
```

Mixing Up AND/OR

Problem: Logic error in conditions.

Example:

```
-- This returns NO results if no one is both a Droid AND Human
WHERE species = 'Droid' AND species = 'Human'

-- This returns characters who are either
WHERE species = 'Droid' OR species = 'Human'
```

✅ Checkpoint: What You've Learnt

Before moving on, make sure you can:

- ✅ Use SELECT to retrieve all columns (*)
- ✅ Select specific columns by name
- ✅ Filter rows with WHERE
- ✅ Use comparison operators (=, !=, <, >, <=, >=)
- ✅ Combine conditions with AND/OR
- ✅ Use LIKE with wildcards (%, _)
- ✅ Understand case sensitivity in text comparisons

🎯 Challenge Problem (Optional)

Write a query that finds all characters whose **species contains "oid"** AND whose **homeworld starts with 'N'**.

Hint: You'll need to combine WHERE, LIKE, and AND.

Write your solution:

```
-- Challenge Problem: Species with 'oid' AND homeworld starting with 'N'
-- YOUR CODE HERE
```

Click to reveal the solution

```
SELECT name, species, homeworld
FROM characters
WHERE species LIKE '%oid%' AND homeworld LIKE 'N%';
```

💾 Save Your Work with Git

Time to save your progress!

Step 1: Check Your Changes

```
git status
```

Step 2: Stage Your File

```
git add lessons/lesson2_queries.sql
```

Step 3: Commit with a Descriptive Message

```
git commit -m "Completed Lesson 2: SELECT queries with WHERE filtering"
```

Step 4: Push to GitHub

```
git push
```

📖 Key SQL Commands Learnt

Command	Purpose	Example
`SELECT`	Retrieve data from tables	`SELECT name FROM characters;`
`WHERE`	Filter rows based on conditions	`WHERE species = 'Human'`
`AND`	Both conditions must be true	`WHERE species = 'Human' AND homeworld = 'Tatooine'`
`OR`	At least one condition must be true	`WHERE species = 'Droid' OR species = 'Human'`
`LIKE`	Pattern matching	`WHERE name LIKE 'L%'`
`%`	Wildcard: any characters	`'%Sky%'`
`_`	Wildcard: exactly one character	`'_oda'`
`DISTINCT`	Remove duplicate values	`SELECT DISTINCT species`

🎉 Excellent Work!

You can now query your database and find exactly the information you need! In the next lesson, you'll learn how to sort and limit your results.

Ready to continue? Move on to `lesson3_instructions.md`

Need Help? - Review the troubleshooting section above - Test queries one piece at a time - Use `SELECT *` first to see all data - Ask your instructor - Compare with the solution file (after attempting yourself!)