Lesson 4: Aggregate Functions and GROUP BY

Duration: 20 minutes
Deliverable: `lesson4_aggregates.sql`

🎯 Learning Objectives

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

- Use aggregate functions (COUNT, AVG, MAX, MIN, SUM)
- Group data with GROUP BY
- Filter groups with HAVING
- Understand the difference between WHERE and HAVING
- Calculate statistics on your data

📚 What are Aggregate Functions?

Aggregate functions perform calculations on multiple rows and return a single result. They help you answer questions like:

- How many characters are in the database?
- What's the average height?
- Who is the tallest character?
- How many characters of each species are there?

🛠️ Part 1: Basic Aggregate Functions (8 minutes)

Step 1: Create Your SQL File

1. Navigate to the `lessons/` folder
2. Create a new file: `lesson4_aggregates.sql`
3. Add a header:

```
-- Lesson 4: Aggregate Functions and GROUP BY
-- Student Name: [Your Name]
-- Date: [Today's Date]
--
-- This script demonstrates aggregate functions and grouping data
```

The Five Main Aggregate Functions

Function	Purpose	Example Result
`COUNT()`	Count rows	11
`AVG()`	Calculate average	165.7
`MAX()`	Find maximum value	228
`MIN()`	Find minimum value	66
`SUM()`	Add up values	1823

Step 2: Count All Characters

```
-- Query 1: Count how many characters are in the table
SELECT COUNT(*) FROM characters;
```

Result: Total number of rows (e.g., 11)

Step 3: Count Non-NULL Values

```
-- Query 2: Count characters who have a height recorded
SELECT COUNT(height) FROM characters;
```

Note: `COUNT(column)` ignores NULL values, but `COUNT(*)` counts all rows.

Step 4: Find Maximum Height

```
-- Query 3: Find the tallest character's height
SELECT MAX(height) FROM characters;
```

Result: 228 (Chewbacca's height)

Step 5: Find Minimum Height

```
-- Query 4: Find the shortest character's height
SELECT MIN(height) FROM characters;
```

Result: 66 (Yoda's height)

Step 6: Calculate Average Height

```
-- Query 5: Calculate the average height of all characters
SELECT AVG(height) FROM characters;
```

Result: Approximately 158 cm (average of all characters)

Step 7: Sum All Heights

```
-- Query 6: Add up all character heights
SELECT SUM(height) FROM characters;
```

Step 8: Use Multiple Aggregates Together

```
-- Query 7: Get multiple statistics at once
SELECT
    COUNT(*) AS total_characters,
    AVG(height) AS average_height,
    MAX(height) AS tallest,
    MIN(height) AS shortest
FROM characters;
```

Explanation: `AS` creates an alias (friendly name) for the result column.

📊 Part 2: GROUP BY Clause (7 minutes)

GROUP BY groups rows that have the same values and performs aggregate functions on each group.

GROUP BY Syntax

```
SELECT column, AGGREGATE_FUNCTION(column)
FROM table_name
GROUP BY column;
```

Step 9: Count Characters by Species

```
-- Query 8: Count how many characters of each species
SELECT species, COUNT(*) AS character_count
FROM characters
GROUP BY species;
```

Result: Shows each species and how many characters belong to it.

Step 10: Average Height by Species

```
-- Query 9: Find the average height for each species
SELECT species, AVG(height) AS average_height
FROM characters
WHERE height IS NOT NULL
GROUP BY species;
```

Note: `WHERE height IS NOT NULL` filters out characters without height data before grouping.

Step 11: Count Characters by Homeworld

```
-- Query 10: Count characters from each homeworld
SELECT homeworld, COUNT(*) AS character_count
FROM characters
GROUP BY homeworld
ORDER BY character_count DESC;
```

Explanation: Groups by homeworld, counts each group, then sorts by count (highest first).

Step 12: Add an Affiliation Column

Before the next examples, let's add affiliation data:

```
-- Add affiliation column
ALTER TABLE characters ADD COLUMN affiliation TEXT;

-- Update characters with affiliations
UPDATE characters SET affiliation = 'Rebel Alliance' WHERE name IN ('Luke Skywalker',
UPDATE characters SET affiliation = 'Jedi Order' WHERE name IN ('Obi-Wan Kenobi', 'Yo
UPDATE characters SET affiliation = 'Galactic Empire' WHERE name = 'Darth Vader';
UPDATE characters SET affiliation = 'Independent' WHERE name = 'R2-D2';
```

Execute these to add affiliation data.

Step 13: Count by Affiliation

```
-- Query 11: Count characters in each affiliation
SELECT affiliation, COUNT(*) AS members
FROM characters
WHERE affiliation IS NOT NULL
GROUP BY affiliation
ORDER BY members DESC;
```

🎯 Part 3: HAVING Clause (5 minutes)

`HAVING` filters groups (like WHERE filters rows). Use HAVING with GROUP BY to filter aggregated results.

WHERE vs HAVING

Clause	Filters	Used With	Example
`WHERE`	Individual rows	Before grouping	`WHERE species = 'Human'`
`HAVING`	Groups	After grouping	`HAVING COUNT(*) > 2`

HAVING Syntax

```
SELECT column, AGGREGATE_FUNCTION(column)
FROM table_name
GROUP BY column
HAVING condition;
```

Step 14: Find Species with Multiple Characters

```
-- Query 12: Show only species with 2 or more characters
SELECT species, COUNT(*) AS character_count
FROM characters
GROUP BY species
HAVING COUNT(*) >= 2;
```

Result: Only shows species that have 2+ characters.

Step 15: Affiliations with Above-Average Membership

```
-- Query 13: Find affiliations with more than the average number of members
SELECT affiliation, COUNT(*) AS member_count
FROM characters
WHERE affiliation IS NOT NULL
GROUP BY affiliation
HAVING COUNT(*) > (SELECT AVG(cnt) FROM (SELECT COUNT(*) AS cnt FROM characters WHERE
```

Note: This is complex! It calculates average group size then filters.

Step 16: Combine WHERE and HAVING

```
-- Query 14: Count humans by homeworld, only showing planets with 2+ humans
SELECT homeworld, COUNT(*) AS human_count
FROM characters
WHERE species = 'Human'
GROUP BY homeworld
HAVING COUNT(*) >= 2;
```

Explanation:

1. `WHERE species = 'Human'` - Filter to humans first
2. `GROUP BY homeworld` - Group remaining rows by planet
3. `HAVING COUNT(*) >= 2` - Only show groups with 2+ characters

🔢 Part 4: COUNT DISTINCT

`COUNT(DISTINCT column)` counts unique values, ignoring duplicates.

Step 17: Count Unique Species

```
-- Query 15: How many different species are there?
SELECT COUNT(DISTINCT species) AS unique_species
FROM characters;
```

Step 18: Count Unique Homeworlds

```
-- Query 16: How many different homeworlds are represented?
SELECT COUNT(DISTINCT homeworld) AS unique_homeworlds
FROM characters;
```

🎓 Practice Exercises

Complete these queries in your `lesson4_aggregates.sql` file:

Exercise 1: Basic Aggregates

```
-- Exercise 1: Find the total height of all characters combined
SELECT SUM(height) AS total_height
FROM characters;
```

Exercise 2: Group and Count

```
-- Exercise 2: Count characters from each homeworld, sorted alphabetically
SELECT homeworld, COUNT(*) AS character_count
FROM characters
GROUP BY homeworld
ORDER BY homeworld;
```

Exercise 3: Average with Grouping

```
-- Exercise 3: Find average height by affiliation
SELECT affiliation, AVG(height) AS avg_height
FROM characters
WHERE height IS NOT NULL AND affiliation IS NOT NULL
GROUP BY affiliation;
```

Exercise 4: Using HAVING

```
-- Exercise 4: Show homeworlds that have exactly 1 character
SELECT homeworld, COUNT(*) AS character_count
FROM characters
GROUP BY homeworld
HAVING COUNT(*) = 1;
```

🐛 Common Errors & Troubleshooting

Error: "misuse of aggregate function"

Problem: Using aggregate function without GROUP BY when other columns are selected.

Wrong:

```
SELECT species, COUNT(*)
FROM characters;
```

Correct:

```
SELECT species, COUNT(*)
FROM characters
GROUP BY species;
```

Rule: If you SELECT a column and an aggregate, you must GROUP BY that column.

Error: "no such column in GROUP BY"

Problem: Grouping by a column not in SELECT or misspelt.

Solution: Ensure column exists and is spelt correctly.

WHERE vs HAVING Confusion

Wrong:

```
-- Can't use aggregate in WHERE
SELECT species, COUNT(*)
FROM characters
WHERE COUNT(*) > 2
GROUP BY species;
```

Correct:

```
-- Use HAVING for aggregate conditions
SELECT species, COUNT(*)
FROM characters
GROUP BY species
HAVING COUNT(*) > 2;
```

NULL Values in Aggregates

Important:

- `COUNT(*)` counts all rows (including NULLs)
- `COUNT(column)` counts non-NULL values
- `AVG()` , `SUM()` , `MAX()` , `MIN()` ignore NULLs

Example:

```
-- This counts all characters
SELECT COUNT(*) FROM characters;

-- This counts only characters with height data
SELECT COUNT(height) FROM characters;
```

Order of Clauses

Correct order:

```
SELECT columns
FROM table
WHERE condition       -- Filter rows first
GROUP BY column       -- Then group
HAVING condition      -- Filter groups
ORDER BY column       -- Finally sort
LIMIT number;
```

✅ Checkpoint: What You've Learnt

Before moving on, make sure you can:

- ✅ Use COUNT(), AVG(), MAX(), MIN(), SUM()
- ✅ Group data with GROUP BY
- ✅ Filter groups with HAVING
- ✅ Understand WHERE (rows) vs HAVING (groups)
- ✅ Use COUNT(DISTINCT) for unique values
- ✅ Combine aggregates with ORDER BY and LIMIT
- ✅ Write column aliases with AS

🎯 Challenge Problem (Optional)

Task: Find which affiliation has the tallest average height, but only include affiliations with 2 or more members. Show the affiliation name, average height, and member count, sorted by average height (tallest first).

Requirements:

- Use AVG(), COUNT()
- Use GROUP BY
- Use HAVING to filter groups
- Use ORDER BY
- Round average to 2 decimal places: `ROUND(AVG(height), 2)`

Write your solution:

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

Click to reveal the solution

```
SELECT
    affiliation,
    COUNT(*) AS member_count,
    ROUND(AVG(height), 2) AS avg_height
FROM characters
WHERE affiliation IS NOT NULL AND height IS NOT NULL
GROUP BY affiliation
HAVING COUNT(*) >= 2
ORDER BY avg_height DESC;
```

💾 Save Your Work with Git

Save your progress!

Commands:

```
git status
git add lessons/lesson4_aggregates.sql database/starwars.db
git commit -m "Completed Lesson 4: Aggregate functions and GROUP BY"
git push
```

📖 Key SQL Commands Learnt

Command	**Purpose**	**Example**
`COUNT()`	Count rows	`SELECT COUNT(*) FROM characters`
`AVG()`	Calculate average	`SELECT AVG(height) FROM characters`
`MAX()`	Find maximum	`SELECT MAX(height) FROM characters`
`MIN()`	Find minimum	`SELECT MIN(height) FROM characters`
`SUM()`	Add values	`SELECT SUM(height) FROM characters`
`GROUP BY`	Group rows	`GROUP BY species`
`HAVING`	Filter groups	`HAVING COUNT(*) > 2`
`AS`	Column alias	`COUNT(*) AS total`
`DISTINCT`	Unique values	`COUNT(DISTINCT species)`
`ROUND()`	Round numbers	`ROUND(AVG(height), 2)`

🎉 Excellent Work!

You can now perform calculations and analyse your data! In the next lesson, you'll learn about database relationships and creating multiple related tables.

Ready to continue? Move on to `lesson5_instructions.md`

Need Help?

- Remember: WHERE before GROUP BY, HAVING after
- Check for NULL values affecting calculations
- Verify all non-aggregate columns are in GROUP BY
- Ask your instructor
- Compare with the solution file (after attempting yourself!)