**Day 1 of the SQL Challenge:**

**Main Idea:**  
You're given a users table and an events table. The goal is to find the percentage of users who haven't commented or liked any content.

A screenshot of a computer

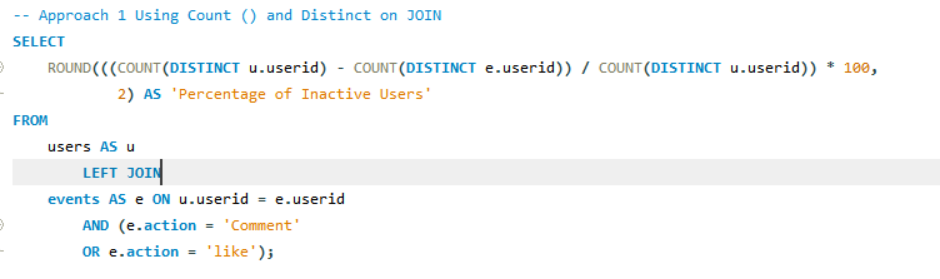
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Question:**  
Find the percentage of users who have never liked or commented.

**Approach 1: Using LEFT JOIN with Aggregate Functions**

* **LEFT JOIN** the users table with the events table on user\_id.
* Filter the events for 'Comment' and 'Like'.
* Calculate the number of distinct users who haven't performed these actions.
* Use the formula (inactive\_users / total\_users) \* 100 to calculate the percentage.
* 

**Approach 2: Using Variables and Subquery**

* Use COUNT(\*) on the Users table to get the total number of users and store this result in a variable @total\_users.
* Use a NOT EXISTS subquery on the Events table to identify users who haven't performed either "comment" or "like" actions. Count these users and store the result in a variable @notcommentednorliked\_users.
* Divide the number of inactive users (@notcommentednorliked\_users) by the total number of users (@total\_users). Multiply by 100 to get the percentage and round the result to 2 decimal places for display.
* A screenshot of a computer program

  AI-generated content may be incorrect.

**Key Takeaway:**

I learned that **LEFT JOIN with aggregate functions** is more efficient and easier to implement than using subqueries and session variables. The LEFT JOIN allows for faster execution and simpler query structure, making it more scalable for larger datasets.

**Business Impact:**

Tracking the percentage of inactive users provides valuable insights into user behavior, platform health, and engagement levels. This data helps businesses:

* Create more effective retention strategies.
* Improve customer support and marketing efforts.
* Optimize features and content to engage users better.
* Measure the success of new product offerings or platform changes.

This allows businesses to proactively address user engagement challenges and implement targeted actions that lead to higher retention and success.

**Which is the Best Optimized Query?**

**Approach 1** is better optimized for the following reasons:

* **Single query execution:** It processes everything in one go, reducing overhead.
* **Efficient use of LEFT JOIN and aggregate functions:** It is more efficient than using subqueries and variables.
* **Scalability:** Approach 1 is more likely to scale better as the dataset grows, due to the efficiency of JOIN operations compared to subqueries in Approach 2.