SQL Challenge: Find Numbers Appearing at Least Three Times Consecutively

Problem Statement

You are given a table `Logs` with the following schema:

Column Name	Type
id	int
num	varchar

- `id` is the **primary key** and is auto-incremented starting from 1.
- Your task is to **find all numbers** that appear **at least three times consecutively** (i.e., in three consecutive rows with the same value).

Example Input

Logs Table:

id	num
1	1
2	1
3	1
4	2
5	1
6	2
7	2

Expected Output

ConsecutiveNums
1

Explanation:
Only the number `1` appears **three times in a row** (at `id` 1, 2, and 3). No other number appears three or more times consecutively.

SQL Query

```sql
SELECT DISTINCT l1.num AS ConsecutiveNums
FROM Logs l1
```

```
JOIN Logs l2 ON l1.id = l2.id - 1
JOIN Logs l3 ON l1.id = l3.id - 2
WHERE l1.num = l2.num AND l2.num = l3.num;
```

Understanding the Problem

We need to:

1. Find numbers that appear at least 3 times consecutively in the Logs table
2. Return these numbers as a distinct list
3. A number is "consecutive" if it appears in rows with sequential IDs

Breaking Down the SQL Query

The provided solution is clever but might be confusing at first glance:

```sql
SELECT DISTINCT l1.num AS ConsecutiveNums
FROM Logs l1
JOIN Logs l2 ON l1.id = l2.id - 1
JOIN Logs l3 ON l1.id = l3.id - 2
WHERE l1.num = l2.num AND l2.num = l3.num;
```

Let's break this down piece by piece:

1. The Three Table Copies

```
FROM Logs l1
JOIN Logs l2 ON l1.id = l2.id - 1
JOIN Logs l3 ON l1.id = l3.id - 2
```

This creates three "copies" of the same table with different aliases:

- `l1` : The first occurrence of a number
- `l2` : The row immediately after `l1` (since l1.id = l2.id - 1)
- `l3` : The row two positions after `l1` (since l1.id = l3.id - 2)

Think of it like sliding a 3-row window across the table.

2. The JOIN Conditions

- `l1.id = l2.id - 1` : This ensures `l2` is the next row after `l1`
- `l1.id = l3.id - 2` : This ensures `l3` is two rows after `l1`

Together, these JOIN conditions create a 3-row consecutive window.

3. Checking for Same Value

```
WHERE l1.num = l2.num AND l2.num = l3.num
```

This WHERE clause ensures all three rows contain the same number. If all three match, we have found a consecutive triplet.

4. Handling Duplicates

```
SELECT DISTINCT l1.num AS ConsecutiveNums
```

The `DISTINCT` keyword ensures we only report each number once, even if it appears consecutive multiple times.

Working Through an Example

Let's use our example table:

id	num
1	1
2	1
3	1
4	2
5	1
6	2
7	2

Let's walk through how the query processes this data:

1. The query creates three copies of the table: `l1`, `l2`, and `l3`

2. It then aligns these copies based on the JOIN conditions:

   ```
   l1.id | l1.num | l2.id | l2.num | l3.id | l3.num
   ------|--------|-------|--------|-------|-------
      1  |   1    |   2   |   1    |   3   |   1    □ (all match)
      2  |   1    |   3   |   1    |   4   |   2    □ (l3 doesn't match)
      3  |   1    |   4   |   2    |   5   |   1    □ (l2 doesn't match)
      4  |   2    |   5   |   1    |   6   |   2    □ (l2 doesn't match)
      5  |   1    |   6   |   2    |   7   |   2    □ (l2 doesn't match)
   ```

3. After filtering with the WHERE clause, only the first row remains (where all three numbers are 1)

4. The query then selects the distinct values from those results, giving us only `1`

Alternative Approach for Beginners

If this approach seems complex, here's an alternative way to think about it:

1. We're checking if each row, the row after it, and the row after that have the same number
2. To do this, we JOIN the table with itself three times
3. We position these three copies so that they represent consecutive rows
4. Then we check if all three rows contain the same number

Key Concepts for SQL Beginners

1. **Self-Join**: Joining a table to itself (using different aliases)
2. **JOIN Conditions**: Using math in JOIN conditions to create specific relationships
3. **DISTINCT**: Removing duplicate values from results
4. **Consecutive Records**: Using ID differences to identify consecutive records

This technique of self-joining tables to find patterns across consecutive rows is very powerful in SQL and can be used for many similar problems.