**SQL Bug Fixing: Fix the QUERY - Totaling**

Oh no! Timmys been moved into the database divison of his software company but as we know Timmy loves making mistakes. Help Timmy keep his job by fixing his query...

Timmy works for a statistical analysis company and has been given a task of totaling the number of sales on a given day grouped by each department name and then each day.

Resultant table:

day (type: date) {group by} [order by asc]

department (type: text) {group by} [In a real world situation it is bad practice to name a column after a table]

sale\_count (type: int)

Tables and relationship below:

A diagram of a data flow

Description automatically generated with medium confidence

**Solution:**

**SELECT DISTINCT DATE (s.transaction\_date) AS day, d.name AS department, COUNT(s.id) AS sale\_count**

**FROM department d**

**INNER JOIN sale s ON d.id = s.department\_id**

**GROUP BY d.name, day**

**ORDER BY day ASC**

**Calculating Batting Average**

In baseball, the batting average is a simple and most common way to measure a hitter's performace. Batting average is calculated by taking all the players hits and dividing it by their number of at\_bats, and it is usually displayed as a 3 digit decimal (i.e. 0.300).

Given a yankees table with the following schema,

-player\_id STRING

-player\_name STRING

-primary\_position STRING

-games INTEGER

-at\_bats INTEGER

-hits INTEGER

return a table with player\_name, games, and batting\_average.

We want batting\_average to be rounded to the nearest thousandth, since that is how baseball fans are used to seeing it. Format it as text and make sure it has 3 digits to the right of the decimal (pad with zeroes if neccesary).

Next, order our resulting table by batting\_average, with the highest average in the first row.

Finally, since batting\_average is a rate statistic, a small number of at\_bats can change the average dramatically. To correct for this, exclude any player who doesn't have at least 100 at bats.

Expected Output Table

-player\_name STRING

-games INTEGER

-batting\_average STRING

**Solution:**

**SELECT player\_name, games, CAST(ROUND(hits::NUMERIC / at\_bats, 3) AS DECIMAL(10,3))::TEXT AS batting\_average**

**FROM yankees**

**WHERE at\_bats >= 100**

**ORDER BY batting\_average DESC**

**SQL: Disorder**

You are given a table numbers with just one column, number. It holds some numbers that are already ordered.

You need to write a query that makes them un-ordered, as in, every possible ordering should appear equally often.

**Solution:**

**SELECT \***

**FROM numbers**

**ORDER BY RANDOM()**

**SQL with Pokemon: Damage Multipliers**

You have arrived at the Celadon Gym to battle Erika for the Rainbow Badge.

She will be using Grass-type Pokemon. Any fire pokemon you have will be strong against grass, but your water types will be weakened. The multipliers table within your Pokedex will take care of that.

Using the following tables, return the pokemon\_name, modifiedStrength and element of the Pokemon whose strength, after taking these changes into account, is greater than or equal to 40, ordered from strongest to weakest.

pokemon schema

id

pokemon\_name

element\_id

str

multipliers schema

id

element

multiplier

**Solution:**

**SELECT p.pokemon\_name, (p.str \* m.multiplier) AS modifiedStrength, m.element**

**FROM pokemon p**

**LEFT JOIN multipliers m ON p.element\_id = m.id**

**WHERE modifiedStrength >= 40**

**ORDER BY modifiedStrength DESC**