DESCRIPTION

Table: Queries

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| query_name | varchar |

| result | varchar |

| position | int |

| rating | int |

+-------------+---------+

This table may have duplicate rows.

This table contains information collected from some queries on a database.

The position column has a value from **1** to **500**.

The rating column has a value from **1** to **5**. Query with rating less than 3 is a poor query.

We define query quality as:

The average of the ratio between query rating and its position.

We also define poor query percentage as:

The percentage of all queries with rating less than 3.

Write a solution to find each query_name, the quality and poor_query_percentage.

Both quality and poor_query_percentage should be **rounded to 2 decimal places**.

Return the result table in **any order**.

The result format is in the following example.

Example 1:

Input:

Queries table:

+------------+-------------------+----------+--------+

| query_name | result | position | rating |

+------------+------------------+----------+--------+

| Dog | Golden Retriever | 1 | 5 |

| Dog | German Shepherd | 2 | 5 |

| Dog | Mule | 200 | 1 |

| Cat | Shirazi | 5 | 2 |

| Cat | Siamese | 3 | 3 |

| Cat | Sphynx | 7 | 4 |

+------------+------------------+----------+--------+

Output:

+------------+---------+----------------------+

| query_name | quality | poor_query_percentage |

+------------+---------+----------------------+

| Dog | 2.50 | 33.33 |

| Cat | 0.66 | 33.33 |

+------------+---------+----------------------+

Explanation:

Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50

Dog queries poor_ query_percentage is (1 / 3) * 100 = 33.33

Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66

Cat queries poor_ query_percentage is (1 / 3) * 100 = 33.33

SOLUTION

MySQL:

- Select query_name, calculate quality using AVG(), and round the result to 2 decimals using ROUND()
- GROUP BY query_name

```
SELECT query_name, ROUND(AVG(rating / position), 2) quality, ROUND(SUM(IF(rating < 3, 1, 0)) * 100 /
COUNT(rating), 2) poor_query_percentage
FROM Queries
GROUP BY query_name;
```

PostgreSQL:

- Add a CTE (WITH) to find number of users from Users table using COUNT()
- Find percentage (number of users (Register table) * 100 / total users from CTE), and ROUND the result to 2 decimals using ROUND()
- Join tables using JOIN
- GROUP BY contest_id and ORDER BY percentage in descending order, and contest_id in ascending order

```
WITH t1 AS(
    SELECT query_name, ROUND(SUM(ROUND(rating, 2)/position)/COUNT(result), 2) quality,
COUNT(result) total_different_animals
    FROM Queries
    GROUP BY 1),
t2 AS(
    SELECT query_name, COUNT(rating) rating_less_than_3
    FROM Queries
    WHERE rating < 3
    GROUP BY 1)
SELECT t1.query_name, t1.quality, COALESCE(NULLIF(ROUND(ROUND(t2.rating_less_than_3, 2)
*100 / t1.total_different_animals, 2), 0), 0) poor_query_percentage
FROM t1
LEFT JOIN t2
ON t1.query_name = t2.query_name;
```