

Problem 3580: Find Consistently Improving Employees

Problem Information

Difficulty: Medium

Acceptance Rate: 56.09%

Paid Only: No

Tags: Database

Problem Description

Table: `employees`

+-----+-----+ | Column Name | Type | +-----+-----+ | employee_id | int | | name | varchar | +-----+-----+ employee_id is the unique identifier for this table. Each row contains information about an employee.

Table: `performance_reviews`

+-----+-----+ | Column Name | Type | +-----+-----+ | review_id | int | | employee_id | int | | review_date | date | | rating | int | +-----+-----+ review_id is the unique identifier for this table. Each row represents a performance review for an employee. The rating is on a scale of 1-5 where 5 is excellent and 1 is poor.

Write a solution to find employees who have consistently improved their performance over **their last three reviews**.

* An employee must have **at least** `3` **review** to be considered
* The employee's **last** `3` **reviews** must show **strictly increasing ratings** (each review better than the previous)
* Use the most recent `3` reviews based on `review_date` for each employee
* Calculate the **improvement score** as the difference between the latest rating and the earliest rating among the last `3` reviews

Return _the result table ordered by**improvement score** in **descending** order, then by **name** in **ascending** order_.

The result format is in the following example.

****Example:****

****Input:****

employees table:

	employee_id	name									
Alice Johnson	1	Alice Johnson	2	Bob Smith	3	Carol Davis	4	David Wilson	5	Emma Brown	

performance_reviews table:

	review_id	employee_id	review_date	rating							
1 2023-01-15 1 1 2023-04-15 2 2 1 2023-05-01 2 3 3 1 2023-07-15 4 4 1 2023-10-15 5 5 2 2023-02-01 3 6 2 2023-08-01 4 8 2 2023-11-01 5 9 3 2023-03-10 1 10 3 2023-06-10 2 11 3 2023-09-10 3 12 3 2023-12-10 4 13 4 2023-01-20 4 14 4 2023-04-20 4 15 4 2023-07-20 4 16 5 2023-02-15 3 17 5 2023-05-15 2											

****Output:****

	employee_id	name	improvement_score								
Carol Davis	2	Bob Smith	3	1	Alice Johnson	2	3				

****Explanation:****

* **Alice Johnson (employee_id = 1):** * Has 4 reviews with ratings: 2, 3, 4, 5 * Last 3 reviews (by date): 2023-04-15 (3), 2023-07-15 (4), 2023-10-15 (5) * Ratings are strictly increasing: 3 -> 4 -> 5 * Improvement score: $5 - 3 = 2$ * **Carol Davis (employee_id = 3):** * Has 4 reviews with ratings: 1, 2, 3, 4 * Last 3 reviews (by date): 2023-06-10 (2), 2023-09-10 (3), 2023-12-10 (4) * Ratings are strictly increasing: 2 -> 3 -> 4 * Improvement score: $4 - 2 = 2$ * **Bob Smith (employee_id = 2):** * Has 4 reviews with ratings: 3, 2, 4, 5 * Last 3 reviews (by date): 2023-05-01 (2), 2023-08-01 (4), 2023-11-01 (5) * Ratings are strictly increasing: 2 -> 4 -> 5 * Improvement score: $5 - 2 = 3$ * **Employees not included:** * David Wilson (employee_id = 4): Last 3 reviews are all 4 (no improvement) * Emma Brown (employee_id = 5): Only has 2 reviews (needs at least 3)

The output table is ordered by improvement_score in descending order, then by name in ascending order.

Code Snippets

MySQL:

```
# Write your MySQL query statement below
```

MS SQL Server:

```
/* Write your T-SQL query statement below */
```

PostgreSQL:

```
-- Write your PostgreSQL query statement below
```