31_Find Students Who Improved Medium - Solution

Source: https://leetcode.com/problems/find-students-who-improved/description/

Running Notes

- Write a solution to find the **students who have shown improvement**.
- Improvement criteria

  A student is considered to have shown improvement if they meet **both** of these conditions:

  - Have taken exams in the **same subject** on at least two different dates
  - Their **latest score** in that subject is **higher** than their **first score**
- Return *the result table ordered by* `student_id,` `subject` *in* ***ascending*** *order*.

I knew somewhere this was wrong, but I wanted to try it out anyway and document it :) (documentation is half the problem solved :))

```
SELECT *
FROM Scores
WHERE count(student_id) > 1
```

You cannot use aggregate functions (like `COUNT`, `SUM`, `AVG`, etc.) directly in the `WHERE` clause because the `WHERE` clause is applied **before** the aggregation happens. Instead, aggregate functions are used in the `HAVING` clause, which filters the results **after** the aggregation is complete.

```
-- Write your PostgreSQL query statement below
SELECT student_result_1.student_id, student_result_1.subject,
student_result_1.score AS first_score,
student_result_2.score AS latest_score
FROM Scores as student_result_1
INNER JOIN Scores as student_result_2
ON student_result_1.student_id = student_result_2.student_id
AND student_result_2.exam_date > student_result_1.exam_date
AND student_result_1.subject = student_result_2.subject
AND student_result_2.score > student_result_1.score
```

So I very happily wrote this code and it has passed the 1st test case, and I felt like I've been acing the "Medium" problems just to be hit by the fact that it failed in the next test case lol

So then I realised the problem and I also realized why "Medium"

So the part of the question that I technically didn't cater to was **latest score and the fact that a student can give exams multiple times and the fact that we need to find the difference between their first score and latest score**

```
-- Write your PostgreSQL query statement below
WITH correct_dates_CTE AS (
SELECT
subject,
student_id,
MIN(exam_date) AS first_date,
MAX(exam_date) AS latest_date
FROM Scores
GROUP BY student_id,subject)

SELECT student_result_1.student_id,
student_result_1.subject,
student_result_1.score AS first_score,
student_result_2.score AS latest_score
FROM Scores as student_result_1
INNER JOIN Scores as student_result_2
ON student_result_1.student_id = student_result_2.student_id
```

```
AND student_result_1.subject = student_result_2.subject
AND student_result_2.score > student_result_1.score
INNER JOIN correct_dates_CTE
ON student_result_1.student_id = correct_dates_CTE.student_id
AND student_result_1.exam_date = correct_dates_CTE.first_date
AND student_result_2.exam_date = correct_dates_CTE.latest_date
AND student_result_1.subject = correct_dates_CTE.subject
ORDER BY student_result_1.student_id,student_result_1.subject
```

So I'll break down my code

- find the first date and latest date when the exam was taken for each student and each subject
- then I did a join with the scores table considering 1 table as 1st results of student and another as 2nd results of students ... so I joined both tables ensuring
    - student id is same
    - subject is same
    - and they should've scored more during their second results
- and then I have joined with my dates table to ensure that:
    - the 1st result is from the first date
    - the 2nd results is from the latest date

But clearly, this query needs to be optimized, so I wrote this code

```
WITH final_CTE AS (
    SELECT *,
    FIRST_VALUE(score) OVER(
    PARTITION BY student_id, subject
    ORDER BY exam_date
    ) AS first_score,
    LAST_VALUE(score) OVER(
        PARTITION BY student_id, subject
    ) AS latest_score
FROM Scores
)

SELECT
student_id,
subject,
first_score,
latest_score
FROM final_CTE
WHERE score=first_score AND latest_score>first_score
```

But this again failed at 1 test case, it give a duplicate row

and the reason is there are two rows for a `student_id` and `subject` have the same `first_score`, they both satisfy the `score = first_score` condition

Hence the final answer is:

```
WITH final_CTE AS (
    SELECT *,
    FIRST_VALUE(score) OVER(
    PARTITION BY student_id, subject
    ORDER BY exam_date
    ) AS first_score,
    FIRST_VALUE(score) OVER(
        PARTITION BY student_id, subject
        ORDER BY exam_date DESC
    ) AS latest_score
FROM Scores
)

SELECT DISTINCT
student_id,
subject,
first_score,
latest_score
```

```
FROM final_CTE
WHERE score=first_score AND latest_score>first_score
```

Written By,
Harshee Pitroda