SQL Challenge: Calculate Task Difficulty Level

Problem Statement

You are given two tables:

- `tasks` : contains the list of task IDs and their names.
- `reports` : contains candidate scores for each task.

Your task is to **calculate the difficulty level** of each task based on the **average score** of all candidates who attempted it.

Difficulty Classification Rules

Avg Score Range	Difficulty
≤ 20	Hard
> 20 and ≤ 60	Medium
> 60	Easy

Input Tables

`tasks`

id	name
101	MinDist
123	Equi
142	Median
300	Tricoloring

`reports` (candidate performance)

id	task_id	candidate	score
13	101	John Smith	100
24	123	Delaney Lloyd	34
37	300	Monroe Jimenez	50
49	101	Stanley Price	45
51	142	Tanner Sears	37
68	142	Lara Fraser	3
83	300	Tanner Sears	0

Expected Output

task_id	task_name	difficulty
101	MinDist	Easy
123	Equi	Medium
142	Median	Hard
300	Tricoloring	Medium

SQL Solution

```sql
SELECT
    t.id,
    t.name,
    CASE
        WHEN AVG(rp.score) <= 20 THEN 'Hard'
        WHEN AVG(rp.score) > 20 AND AVG(rp.score) <= 60 THEN 'Medium'
        ELSE 'Easy'
    END AS difficulty
FROM tasks t
JOIN reports rp
    ON t.id = rp.task_id
GROUP BY t.id, t.name
ORDER BY t.id, t.name;
```

Understanding the Problem

We need to:

1. Join the `tasks` and `reports` tables
2. Calculate the average score for each task
3. Classify the difficulty level based on the average score:
    - ≤ 20: Hard
    - > *20 and ≤ 60: Medium*
    - > *60: Easy*
4. Return the task ID, name, and difficulty level

Breaking Down the Solution

Let's analyze the SQL query that solves this problem:

```sql
SELECT
    t.id,
    t.name,
    CASE
        WHEN AVG(rp.score) <= 20 THEN 'Hard'
        WHEN AVG(rp.score) > 20 AND AVG(rp.score) <= 60 THEN 'Medium'
        ELSE 'Easy'
    END AS difficulty
```

```
FROM tasks t
JOIN reports rp
    ON t.id = rp.task_id
GROUP BY t.id, t.name
ORDER BY t.id, t.name;
```

Step 1: Joining the Tables

First, we need to connect the task information with the candidate scores:

```
FROM tasks t
JOIN reports rp
    ON t.id = rp.task_id
```

This joins the `tasks` table with the `reports` table where the task ID in both tables matches. After this join, we get:

t.id	t.name	rp.id	rp.task_id	rp.candidate	rp.score
101	MinDist	13	101	John Smith	100
101	MinDist	49	101	Stanley Price	45
123	Equi	24	123	Delaney Lloyd	34
142	Median	51	142	Tanner Sears	37
142	Median	68	142	Lara Fraser	3
300	Tricoloring	37	300	Monroe Jimenez	50
300	Tricoloring	83	300	Tanner Sears	0

Step 2: Grouping and Calculating Averages

Next, we group the results by task ID and name to calculate the average score for each task:

```
GROUP BY t.id, t.name
```

This allows us to use the `AVG()` function to calculate the average score for each task. Let's manually calculate these averages:

- Task 101 (MinDist): (100 + 45) / 2 = 72.5
- Task 123 (Equi): 34 / 1 = 34
- Task 142 (Median): (37 + 3) / 2 = 20
- Task 300 (Tricoloring): (50 + 0) / 2 = 25

Step 3: Classifying Difficulty Levels

The CASE statement determines the difficulty level based on the average score:

```
CASE
    WHEN AVG(rp.score) <= 20 THEN 'Hard'
    WHEN AVG(rp.score) > 20 AND AVG(rp.score) <= 60 THEN 'Medium'
    ELSE 'Easy'
END AS difficulty
```

Applying these rules to our calculated averages:

- Task 101 (MinDist): 72.5 > 60, so it's 'Easy'
- Task 123 (Equi): 34 > 20 and ≤ 60, so it's 'Medium'
- Task 142 (Median): 20 ≤ 20, so it's 'Hard'
- Task 300 (Tricoloring): 25 > 20 and ≤ 60, so it's 'Medium'

Step 4: Final Result

Finally, we order the results by task ID and name:

```
ORDER BY t.id, t.name;
```

This gives us our final output:

task_id	task_name	difficulty
101	MinDist	Easy
123	Equi	Medium
142	Median	Hard
300	Tricoloring	Medium

Why This Works (Detailed Explanation)

1. **JOIN Operation**: The JOIN connects task details with performance reports. This is a standard INNER JOIN, meaning only tasks that have at least one report will appear in the results.
2. **Aggregate Function**: The `AVG()` function calculates the mean of all scores for each task. This works because we're grouping by task ID and name.
3. **CASE Statement**: This implements a simple decision tree:
    - If the average score is less than or equal to 20, the task is classified as 'Hard'
    - If the average score is between 20 and 60 (inclusive), it's 'Medium'
    - Otherwise (average score > 60), it's 'Easy'
4. **GROUP BY Clause**: By grouping on task ID and name, we ensure each task appears exactly once in the output, with its calculated difficulty level.

Alternative Approaches

For better readability, you could calculate the average score in a subquery or CTE (Common Table Expression):

```
WITH task_averages AS (
    SELECT
        t.id,
        t.name,
        AVG(rp.score) AS avg_score
    FROM tasks t
```

```
    JOIN reports rp ON t.id = rp.task_id
    GROUP BY t.id, t.name
)
SELECT
    id AS task_id,
    name AS task_name,
    CASE
        WHEN avg_score <= 20 THEN 'Hard'
        WHEN avg_score <= 60 THEN 'Medium'
        ELSE 'Easy'
    END AS difficulty
FROM task_averages
ORDER BY id, name;
```

This approach makes the query more readable and easier to maintain, especially when the classification rules might change in the future.