SQL Query to Calculate Team Rankings

I'll walk through creating this SQL query in small, easily understood steps that even beginners can follow.

The Problem

We need to:

1. Calculate how many points each soccer team has earned across all matches
2. Rank the teams based on their total points
3. If teams have the same points, rank them by team_id

Step 1: Understanding the Data and Rules

We have two tables:

- `teams`: Contains information about each team
- `matches`: Contains information about each match played

The scoring rules are:

- Win (more goals than opponent): 3 points
- Draw (same number of goals): 1 point
- Loss (fewer goals than opponent): 0 points

Step 2: Planning Our Approach

For each team, we need to:

1. Find all matches where they played (either as host or guest)
2. Calculate points earned in each match
3. Sum up all points
4. Display results ordered by points (descending) and team_id (ascending) if tied

Step 3: Calculating Points When Team is Host

First, let's calculate points when a team plays as host:

```
SELECT
    host_team AS team_id,
    CASE
        WHEN host_goals > guest_goals THEN 3  -- Win
        WHEN host_goals = guest_goals THEN 1  -- Draw
        ELSE 0                                -- Loss
    END AS points
FROM matches
```

This query:

- Takes each match from the `matches` table
- Identifies the host team as `team_id`
- Uses a CASE statement to assign points based on the scoring rules
- Returns a table with two columns: team_id and points earned in that match

Step 4: Calculating Points When Team is Guest

Similarly, we calculate points when a team plays as guest:

```
SELECT
    guest_team AS team_id,
    CASE
        WHEN guest_goals > host_goals THEN 3  -- Win
        WHEN guest_goals = host_goals THEN 1  -- Draw
        ELSE 0                                -- Loss
    END AS points
FROM matches
```

Step 5: Combining Both Results

We use UNION ALL to combine both results:

```
SELECT
    host_team AS team_id,
    CASE
        WHEN host_goals > guest_goals THEN 3
        WHEN host_goals = guest_goals THEN 1
        ELSE 0
    END AS points
FROM matches

UNION ALL

SELECT
    guest_team AS team_id,
    CASE
        WHEN guest_goals > host_goals THEN 3
        WHEN guest_goals = host_goals THEN 1
        ELSE 0
    END AS points
FROM matches
```

This gives us a list of all teams and the points they earned in each match.

Step 6: Using a Common Table Expression (CTE)

We'll use a CTE to store this combined result temporarily:

```
WITH team_points AS (
    -- Points for teams when they were the host
    SELECT
        host_team AS team_id,
        CASE
            WHEN host_goals > guest_goals THEN 3
            WHEN host_goals = guest_goals THEN 1
            ELSE 0
        END AS points
    FROM matches

    UNION ALL

    -- Points for teams when they were the guest
    SELECT
        guest_team AS team_id,
        CASE
            WHEN guest_goals > host_goals THEN 3
            WHEN guest_goals = host_goals THEN 1
            ELSE 0
        END AS points
    FROM matches
)
```

Step 7: Summing Up the Points

Now we need to sum up all points for each team:

```
SELECT
    team_id,
    SUM(points) AS total_points
FROM team_points
GROUP BY team_id
```

This groups all records by team_id and sums the points for each team.

Step 8: Adding Team Names

We need to join with the teams table to get the team names:

```
SELECT
    t.team_id,
    t.team_name,
    COALESCE(SUM(tp.points), 0) AS num_points
FROM
    teams t
LEFT JOIN
    team_points tp ON t.team_id = tp.team_id
GROUP BY
    t.team_id, t.team_name
```

Note:

- We use LEFT JOIN to include all teams, even if they didn't play any matches
- COALESCE ensures that teams with no matches get 0 points instead of NULL

Step 9: Ordering the Results

Finally, we order the results as required:

```
SELECT
    t.team_id,
    t.team_name,
    COALESCE(SUM(tp.points), 0) AS num_points
FROM
    teams t
LEFT JOIN
    team_points tp ON t.team_id = tp.team_id
GROUP BY
    t.team_id, t.team_name
ORDER BY
    num_points DESC,  -- Highest points first
    t.team_id ASC     -- In case of a tie, order by team_id
```

Step 10: The Complete SQL Query

Putting it all together:

```
WITH team_points AS (
    -- Points for teams when they were the host
    SELECT
        host_team AS team_id,
        CASE
            WHEN host_goals > guest_goals THEN 3  -- Win
            WHEN host_goals = guest_goals THEN 1  -- Draw
            ELSE 0                                -- Loss
        END AS points
    FROM matches

    UNION ALL

    -- Points for teams when they were the guest
    SELECT
        guest_team AS team_id,
        CASE
            WHEN guest_goals > host_goals THEN 3  -- Win
            WHEN guest_goals = host_goals THEN 1  -- Draw
            ELSE 0                                -- Loss
        END AS points
    FROM matches
)

SELECT
    t.team_id,
    t.team_name,
    COALESCE(SUM(tp.points), 0) AS num_points
FROM
    teams t
LEFT JOIN
    team_points tp ON t.team_id = tp.team_id
GROUP BY
    t.team_id, t.team_name
ORDER BY
    num_points DESC,
    t.team_id ASC;
```

Testing with the Example Data

If we run this query on the example data provided, it would produce the correct output:

```
team_id | team_name | num_points
---------------------------------
20      | Never     | 4
50      | Gonna     | 4
10      | Give      | 3
30      | You       | 3
40      | Up        | 0
```

This shows the teams ranked by points with ties broken by team_id, exactly as required.