Databases
Lab 05: A still 'gentle' Introduction to Intermediate SQL.

Andrés Oswaldo Calderón Romero, Ph.D.

September 4, 2025

1 Introduction

In this lab, we will take a hands-on approach to explore data analysis using SQL within the context of European football leagues. Working with a rich dataset that spans multiple seasons and thousands of players and matches, you will gain practical experience designing SQL queries that summarize meaningful insights from relational data.

2 Getting Some Data

This time, we will work with European football leagues, specifically 11 leagues from countries such as England, France, and the Netherlands. The database contains data from the 2008 to 2016 seasons, covering more than 25,000 matches and 10,000 players. More information about the database can be found at `https://www.kaggle.com/datasets/hugomathien/soccer`. The dataset we will use is a simplified version that includes only a few player attributes and no team attributes. Figure 1 shows the Entity-Relationship Diagram (ERD) of the database.

Please follow these steps:

1. Download the `euroleagues.sql` file to an accessible location.
2. Create a database named *euroleagues*.
3. Connect to the database.
4. Run the command `\i euroleagues.sql`. Be sure to update the path according.
5. Enjoy!

3 SQL Sorting

1. Reorder the SQL query lines in Table 1 to determine which player ranks higher –Messi or Cristiano Ronaldo. Hint: You should use two temporal relations for select the involved players and for computing statistics about their overall performance.
2. Reorder the SQL query lines in Table 2 to list players whose potential exceeds the global average and whose height is $\leq 170\,\text{cm}$. Hint: Use different temporal relations two compute the average potential of each player and the general average for all of them.

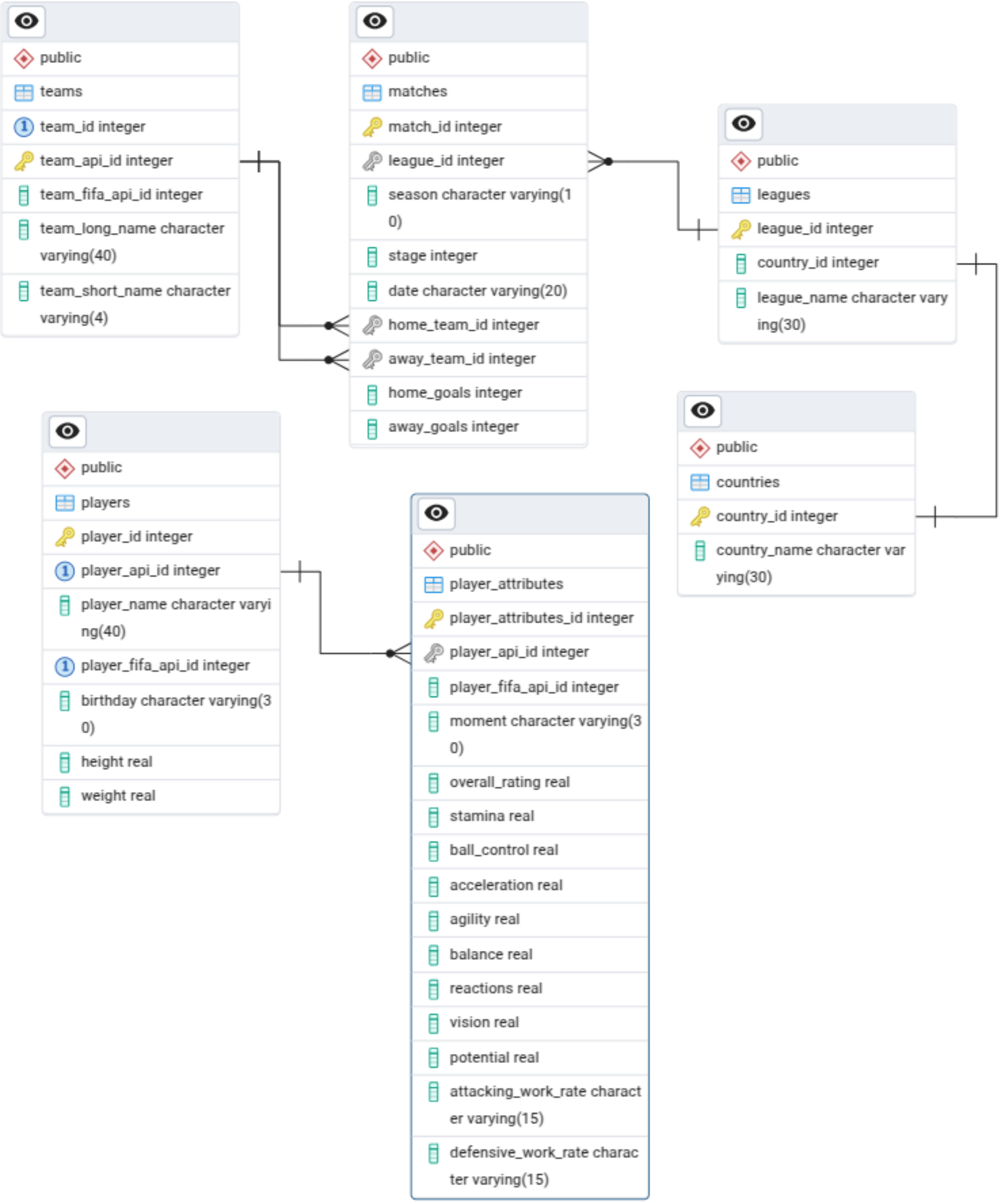

Figure 1: E-R Diagram of the Euroleagues Database.

Table 1: Messi or Cristiano

Pos	SQL Line
1	FROM stats
2	FROM candidates c
3	JOIN public.player_attributes pa
4	ORDER BY avg_overall DESC, max_overall DESC;
5	)
6	FROM public.players p
7	WHERE p.player_name LIKE ‘%Messi%’ OR p.player_name LIKE ‘%Cristiano R%’
8	WITH candidates AS (
9	MAX(pa.overall_rating) AS max_overall
10	AVG(pa.overall_rating) AS avg_overall,
11	SELECT p.player_api_id, p.player_name AS who
12	SELECT *
13	ON pa.player_api_id = c.player_api_id
14	SELECT c.who,
15	stats AS (
16	),
17	GROUP BY c.who

4 SQL Interpretation

For each listing in this section, provide a concise yet thorough explanation of what the SQL statement does –its purpose, key operations (joins, filters, aggregations), and the expected result.

1.

```sql
WITH A AS (
  SELECT home_team_id AS team_api_id
  FROM public.matches
  WHERE home_goals = 0 AND away_goals = 0
  UNION
  SELECT away_team_id
  FROM public.matches
  WHERE home_goals = 0 AND away_goals = 0
),
B AS (
  SELECT home_team_id AS team_api_id
  FROM public.matches
  WHERE (home_goals - away_goals) >= 5
  UNION
  SELECT away_team_id
  FROM public.matches
  WHERE (away_goals - home_goals) >= 5
)
```

Table 2: Above potential

Pos	SQL Line
1	FROM A
2	GROUP BY pa.player_api_id, p.player_name
3	p.height $\leq$ 170
4	NATURAL JOIN public.players p
5	),
6	SELECT AVG(player_potential) AS general_potential
7	B AS (
8	FROM public.players p
9	)
10	WHERE pa.potential IS NOT NULL
11	WHERE a.player_potential $>$ b.general_potential AND
12	WITH A AS (
13	SELECT pa.player_api_id, p.player_name, AVG(pa.potential) AS player_potential
14	NATURAL JOIN A a, B b
15	FROM public.player_attributes pa
16	SELECT p.player_name, a.player_potential, b.general_potential

```
SELECT
  t.team_long_name
FROM (
  SELECT team_api_id FROM A
  INTERSECT
  SELECT team_api_id FROM B) AS foo
JOIN
  public.teams t
USING
  (team_api_id);
```

2.

```
WITH M AS (
  SELECT
    league_id, season,
    CASE WHEN home_goals = away_goals THEN 1 ELSE 0 END AS is_d
  FROM
    public.matches
)
SELECT
  l.league_name, m.season,
  COUNT(*) AS C,
  SUM(m.is_d) AS S,
```

```sql
  ROUND(100.0 * SUM(m.is_d)::numeric / COUNT(*), 2) AS rate
FROM
  M m
JOIN
  public.leagues l ON l.league_id = m.league_id
GROUP BY
  l.league_name, m.season
ORDER BY
  rate DESC
LIMIT
  15;
```

5 What We Expect

Just have fun.

Happy Hacking 😎!